

JIU - The Java Imaging Utilities – API documentation

Marco Schmidt

July 7, 2004

Contents

Class Hierarchy	7
1 Package net.sourceforge.jiu.apps	12
1.1 Interfaces	14
1.1.1 <i>Interface</i> JiuInfo	14
1.1.2 <i>Interface</i> MenuIndexConstants	15
1.1.3 <i>Interface</i> StringIndexConstants	20
1.2 Classes	33
1.2.1 <i>Class</i> ColorIndexer	33
1.2.2 <i>Class</i> dumpcodecs	35
1.2.3 <i>Class</i> EditorState	36
1.2.4 <i>Class</i> ImageDescriptionCreator	44
1.2.5 <i>Class</i> ImageLoadTester	45
1.2.6 <i>Class</i> jiuawt	47
1.2.7 <i>Class</i> jiuawtapplet	48
1.2.8 <i>Class</i> jiuconvert	49
1.2.9 <i>Class</i> MenuWrapper	50
1.2.10 <i>Class</i> OperationProcessor	52
1.2.11 <i>Class</i> StringLoader	60
1.2.12 <i>Class</i> Strings	61
2 Package net.sourceforge.jiu.codecs	64
2.1 Classes	66
2.1.1 <i>Class</i> BMPCodec	66
2.1.2 <i>Class</i> CodecMode	70
2.1.3 <i>Class</i> GIFCodec	71
2.1.4 <i>Class</i> IFFCodec	76
2.1.5 <i>Class</i> ImageCodec	79
2.1.6 <i>Class</i> ImageLoader	92
2.1.7 <i>Class</i> PalmCodec	96
2.1.8 <i>Class</i> PCDCodec	105
2.1.9 <i>Class</i> PNGCodec	109
2.1.10 <i>Class</i> PNMCodec	115
2.1.11 <i>Class</i> PSDCodec	120
2.1.12 <i>Class</i> RASCodec	122
2.2 Exceptions	125
2.2.1 <i>Class</i> InvalidFileStructureException	125
2.2.2 <i>Class</i> InvalidImageIndexException	126
2.2.3 <i>Class</i> UnsupportedCodecModeException	127

2.2.4	<i>Class</i> UnsupportedTypeException	128
2.2.5	<i>Class</i> WrongFormatException	129
3	Package net.sourceforge.jiu.codecs.tiff	130
3.1	Interfaces	132
3.1.1	<i>Interface</i> TIFFConstants	132
3.2	Classes	137
3.2.1	<i>Class</i> TIFFCodec	137
3.2.2	<i>Class</i> TIFFDecoder	143
3.2.3	<i>Class</i> TIFFDecoderDeflated	147
3.2.4	<i>Class</i> TIFFDecoderLogLuv	149
3.2.5	<i>Class</i> TIFFDecoderModifiedHuffman	151
3.2.6	<i>Class</i> TIFFDecoderPackbits	153
3.2.7	<i>Class</i> TIFFDecoderUncompressed	154
3.2.8	<i>Class</i> TIFFFaxCodes	155
3.2.9	<i>Class</i> TIFFImageFileDirectory	157
3.2.10	<i>Class</i> TIFFRational	164
3.2.11	<i>Class</i> TIFFTag	166
4	Package net.sourceforge.jiu.color	169
4.1	Interfaces	170
4.1.1	<i>Interface</i> YCbCrIndex	170
4.2	Classes	171
4.2.1	<i>Class</i> Invert	171
4.2.2	<i>Class</i> WebsafePaletteCreator	172
5	Package net.sourceforge.jiu.color.data	173
5.1	Interfaces	175
5.1.1	<i>Interface</i> CoOccurrenceFrequencyMatrix	175
5.1.2	<i>Interface</i> CoOccurrenceMatrix	178
5.1.3	<i>Interface</i> Histogram1D	180
5.1.4	<i>Interface</i> Histogram3D	182
5.2	Classes	184
5.2.1	<i>Class</i> ArrayHistogram1D	184
5.2.2	<i>Class</i> BaseCoOccurrenceFrequencyMatrix	186
5.2.3	<i>Class</i> MemoryCoOccurrenceFrequencyMatrix	188
5.2.4	<i>Class</i> MemoryCoOccurrenceMatrix	190
5.2.5	<i>Class</i> NaiveHistogram3D	192
5.2.6	<i>Class</i> OnDemandHistogram3D	195
6	Package net.sourceforge.jiu.color.adjustment	198
6.1	Classes	199
6.1.1	<i>Class</i> Brightness	199
6.1.2	<i>Class</i> Contrast	201
6.1.3	<i>Class</i> EqualizeHistogram	203
6.1.4	<i>Class</i> GammaCorrection	204
6.1.5	<i>Class</i> HueSaturationValue	206
6.1.6	<i>Class</i> NormalizeHistogram	208

7	Package net.sourceforge.jiu.color.analysis	209
7.1	Classes	210
7.1.1	Class Histogram1DCreator	210
7.1.2	Class Histogram3DCreator	213
7.1.3	Class MatrixCreator	215
7.1.4	Class MeanDifference	217
7.1.5	Class TextureAnalysis	219
8	Package net.sourceforge.jiu.color.conversion	222
8.1	Classes	223
8.1.1	Class CMYKConversion	223
8.1.2	Class LogLuvConversion	225
8.1.3	Class PCDYCbCrConversion	227
9	Package net.sourceforge.jiu.color.dithering	228
9.1	Interfaces	229
9.1.1	Interface SpotFunction	229
9.2	Classes	230
9.2.1	Class ClusteredDotDither	230
9.2.2	Class DiamondSpotFunction	233
9.2.3	Class ErrorDiffusionDithering	234
9.2.4	Class LineSpotFunction	240
9.2.5	Class OrderedDither	241
9.2.6	Class RoundSpotFunction	244
10	Package net.sourceforge.jiu.color.io	245
10.1	Classes	246
10.1.1	Class HistogramSerialization	246
10.1.2	Class MatrixSerialization	248
10.1.3	Class PaletteSerialization	249
11	Package net.sourceforge.jiu.color.promotion	251
11.1	Classes	252
11.1.1	Class PromotionGray16	252
11.1.2	Class PromotionGray8	253
11.1.3	Class PromotionPaletted8	254
11.1.4	Class PromotionRGB24	255
11.1.5	Class PromotionRGB48	256
12	Package net.sourceforge.jiu.color.quantization	257
12.1	Interfaces	259
12.1.1	Interface RGBQuantizer	259
12.2	Classes	261
12.2.1	Class ArbitraryPaletteQuantizer	261
12.2.2	Class MedianCutContourRemoval	264
12.2.3	Class MedianCutNode	269
12.2.4	Class MedianCutQuantizer	273
12.2.5	Class OctreeColorQuantizer	279
12.2.6	Class OctreeNode	282
12.2.7	Class PopularityQuantizer	285
12.2.8	Class RGBColor	288

12.2.9	<i>Class</i> RGBColorComparator	290
12.2.10	<i>Class</i> RGBColorList	291
12.2.11	<i>Class</i> UniformPaletteQuantizer	293
13	Package net.sourceforge.jiu.color.reduction	295
13.1	Classes	296
13.1.1	<i>Class</i> AutoDetectColorType	296
13.1.2	<i>Class</i> ReduceRGB	299
13.1.3	<i>Class</i> ReduceShadesOfGray	301
13.1.4	<i>Class</i> ReduceToBilevelThreshold	303
13.1.5	<i>Class</i> RGBToGrayConversion	305
14	Package net.sourceforge.jiu.data	308
14.1	Interfaces	311
14.1.1	<i>Interface</i> BilevelImage	311
14.1.2	<i>Interface</i> ByteChannelImage	314
14.1.3	<i>Interface</i> Gray16Image	317
14.1.4	<i>Interface</i> Gray8Image	318
14.1.5	<i>Interface</i> GrayImage	319
14.1.6	<i>Interface</i> GrayIntegerImage	321
14.1.7	<i>Interface</i> IntegerImage	322
14.1.8	<i>Interface</i> Paletted8Image	325
14.1.9	<i>Interface</i> PalettedImage	326
14.1.10	<i>Interface</i> PalettedIntegerImage	327
14.1.11	<i>Interface</i> PixelImage	328
14.1.12	<i>Interface</i> RGB24Image	331
14.1.13	<i>Interface</i> RGB48Image	332
14.1.14	<i>Interface</i> RGBImage	333
14.1.15	<i>Interface</i> RGBIndex	334
14.1.16	<i>Interface</i> RGBIntegerImage	336
14.1.17	<i>Interface</i> ShortChannelImage	337
14.1.18	<i>Interface</i> TransparencyInformation	340
14.2	Classes	342
14.2.1	<i>Class</i> MemoryBilevelImage	342
14.2.2	<i>Class</i> MemoryByteChannelImage	345
14.2.3	<i>Class</i> MemoryGray16Image	350
14.2.4	<i>Class</i> MemoryGray8Image	352
14.2.5	<i>Class</i> MemoryPaletted8Image	354
14.2.6	<i>Class</i> MemoryRGB24Image	356
14.2.7	<i>Class</i> MemoryRGB48Image	357
14.2.8	<i>Class</i> MemoryShortChannelImage	358
14.2.9	<i>Class</i> Palette	363
15	Package net.sourceforge.jiu.filters	366
15.1	Classes	367
15.1.1	<i>Class</i> AreaFilterOperation	367
15.1.2	<i>Class</i> BorderSampleGenerator	371
15.1.3	<i>Class</i> ConvolutionKernelData	374
15.1.4	<i>Class</i> ConvolutionKernelFilter	377
15.1.5	<i>Class</i> MaximumFilter	381

15.1.6	<i>Class</i> MeanFilter	383
15.1.7	<i>Class</i> MedianFilter	385
15.1.8	<i>Class</i> MinimumFilter	387
15.1.9	<i>Class</i> OilFilter	388
15.1.10	<i>Class</i> UnsharpMaskKernel	390
16	Package net.sourceforge.jiu.geometry	391
16.1	Classes	393
16.1.1	<i>Class</i> BellFilter	393
16.1.2	<i>Class</i> BoxFilter	395
16.1.3	<i>Class</i> BSplineFilter	396
16.1.4	<i>Class</i> Crop	397
16.1.5	<i>Class</i> Flip	399
16.1.6	<i>Class</i> HermiteFilter	400
16.1.7	<i>Class</i> Lanczos3Filter	401
16.1.8	<i>Class</i> Mirror	402
16.1.9	<i>Class</i> MitchellFilter	404
16.1.10	<i>Class</i> Resample	405
16.1.11	<i>Class</i> ResampleFilter	409
16.1.12	<i>Class</i> Rotate180	411
16.1.13	<i>Class</i> Rotate90Left	413
16.1.14	<i>Class</i> Rotate90Right	414
16.1.15	<i>Class</i> ScaleReplication	416
16.1.16	<i>Class</i> Shear	418
16.1.17	<i>Class</i> TriangleFilter	420
17	Package net.sourceforge.jiu.gui.awt	421
17.1	Classes	422
17.1.1	<i>Class</i> AwtInfo	422
17.1.2	<i>Class</i> AwtMenuWrapper	423
17.1.3	<i>Class</i> AwtOperationProcessor	425
17.1.4	<i>Class</i> BufferedRGB24Image	433
17.1.5	<i>Class</i> ImageCanvas	437
17.1.6	<i>Class</i> ImageCreator	440
17.1.7	<i>Class</i> JiuAwtFrame	443
17.1.8	<i>Class</i> RGBA	448
17.1.9	<i>Class</i> ToolkitLoader	452
18	Package net.sourceforge.jiu.gui.awt.dialogs	454
18.1	Classes	456
18.1.1	<i>Class</i> CropDialog	456
18.1.2	<i>Class</i> Dialogs	459
18.1.3	<i>Class</i> GammaCorrectionDialog	460
18.1.4	<i>Class</i> HueSaturationValueDialog	462
18.1.5	<i>Class</i> InfoDialog	464
18.1.6	<i>Class</i> IntegerDialog	466
18.1.7	<i>Class</i> MapToArbitraryPaletteDialog	468
18.1.8	<i>Class</i> MedianCutDialog	471
18.1.9	<i>Class</i> OctreeDialog	474
18.1.10	<i>Class</i> ReduceGrayscaleDialog	477

18.1.11	<i>Class</i> ScaleDialog	480
18.1.12	<i>Class</i> ShearDialog	483
18.1.13	<i>Class</i> UniformPaletteQuantizerDialog	485
18.1.14	<i>Class</i> WindowSizeDialog	488
18.1.15	<i>Class</i> YesNoDialog	490
19	Package net.sourceforge.jiu.ops	492
19.1	Interfaces	493
19.1.1	<i>Interface</i> ProgressListener	493
19.2	Classes	495
19.2.1	<i>Class</i> BatchProcessorOperation	495
19.2.2	<i>Class</i> ImagesToImageOperation	498
19.2.3	<i>Class</i> ImageToImageOperation	501
19.2.4	<i>Class</i> LookupTableOperation	505
19.2.5	<i>Class</i> Operation	508
19.3	Exceptions	512
19.3.1	<i>Class</i> MissingParameterException	512
19.3.2	<i>Class</i> OperationFailedException	513
19.3.3	<i>Class</i> WrongParameterException	514
20	Package net.sourceforge.jiu.util	515
20.1	Interfaces	516
20.1.1	<i>Interface</i> ComparatorInterface	516
20.2	Classes	517
20.2.1	<i>Class</i> ArrayConverter	517
20.2.2	<i>Class</i> ArrayRotation	522
20.2.3	<i>Class</i> ArrayScaling	525
20.2.4	<i>Class</i> Median	526
20.2.5	<i>Class</i> SeekableByteArrayOutputStream	528
20.2.6	<i>Class</i> Sort	532
20.2.7	<i>Class</i> Statistics	533
20.2.8	<i>Class</i> SystemInfo	537

Class Hierarchy

Classes

- java.lang.Object
 - java.awt.Component
 - java.awt.Canvas
 - net.sourceforge.jiu.gui.awt.ImageCanvas (in 17.1.5, page 437)
 - java.awt.Container
 - java.awt.Panel
 - java.applet.Applet
 - net.sourceforge.jiu.apps.jiuawtapplet (in 1.2.7, page 48)
 - java.awt.Window
 - java.awt.Dialog
 - net.sourceforge.jiu.gui.awt.dialogs.CropDialog (in 18.1.1, page 456)
 - net.sourceforge.jiu.gui.awt.dialogs.GammaCorrectionDialog (in 18.1.3, page 460)
 - net.sourceforge.jiu.gui.awt.dialogs.HueSaturationValueDialog (in 18.1.4, page 462)
 - net.sourceforge.jiu.gui.awt.dialogs.InfoDialog (in 18.1.5, page 464)
 - net.sourceforge.jiu.gui.awt.dialogs.IntegerDialog (in 18.1.6, page 466)
 - net.sourceforge.jiu.gui.awt.dialogs.MapToArbitraryPaletteDialog (in 18.1.7, page 468)
 - net.sourceforge.jiu.gui.awt.dialogs.MedianCutDialog (in 18.1.8, page 471)
 - net.sourceforge.jiu.gui.awt.dialogs.OctreeDialog (in 18.1.9, page 474)
 - net.sourceforge.jiu.gui.awt.dialogs.ReduceGrayscaleDialog (in 18.1.10, page 477)
 - net.sourceforge.jiu.gui.awt.dialogs.ScaleDialog (in 18.1.11, page 480)
 - net.sourceforge.jiu.gui.awt.dialogs.ShearDialog (in 18.1.12, page 483)
 - net.sourceforge.jiu.gui.awt.dialogs.UniformPaletteQuantizerDialog (in 18.1.13, page 485)
 - net.sourceforge.jiu.gui.awt.dialogs.WindowSizeDialog (in 18.1.14, page 488)
 - net.sourceforge.jiu.gui.awt.dialogs.YesNoDialog (in 18.1.15, page 490)
 - java.awt.Frame
 - net.sourceforge.jiu.gui.awt.JiuAwtFrame (in 17.1.7, page 443)
 - java.io.OutputStream
 - net.sourceforge.jiu.util.SeekableByteArrayOutputStream (in 20.2.5, page 528)
 - net.sourceforge.jiu.apps.EditorState (in 1.2.3, page 36)
 - net.sourceforge.jiu.apps.ImageDescriptionCreator (in 1.2.4, page 44)
 - net.sourceforge.jiu.apps.MenuWrapper (in 1.2.9, page 50)

- `net.sourceforge.jiu.gui.awt.AwtMenuWrapper` (in 17.1.2, page 423)
- `net.sourceforge.jiu.apps.OperationProcessor` (in 1.2.10, page 52)
 - `net.sourceforge.jiu.gui.awt.AwtOperationProcessor` (in 17.1.3, page 425)
- `net.sourceforge.jiu.apps.StringLoader` (in 1.2.11, page 60)
- `net.sourceforge.jiu.apps.Strings` (in 1.2.12, page 61)
- `net.sourceforge.jiu.apps.dumpcodecs` (in 1.2.2, page 35)
- `net.sourceforge.jiu.apps.jiuawt` (in 1.2.6, page 47)
- `net.sourceforge.jiu.apps.jiuconvert` (in 1.2.8, page 49)
- `net.sourceforge.jiu.codecs.CodecMode` (in 2.1.2, page 70)
- `net.sourceforge.jiu.codecs.ImageLoader` (in 2.1.6, page 92)
- `net.sourceforge.jiu.codecs.tiff.TIFFDecoder` (in 3.2.2, page 143)
 - `net.sourceforge.jiu.codecs.tiff.TIFFDecoderDeflated` (in 3.2.3, page 147)
 - `net.sourceforge.jiu.codecs.tiff.TIFFDecoderLogLuv` (in 3.2.4, page 149)
 - `net.sourceforge.jiu.codecs.tiff.TIFFDecoderModifiedHuffman` (in 3.2.5, page 151)
 - `net.sourceforge.jiu.codecs.tiff.TIFFDecoderPackbits` (in 3.2.6, page 153)
 - `net.sourceforge.jiu.codecs.tiff.TIFFDecoderUncompressed` (in 3.2.7, page 154)
- `net.sourceforge.jiu.codecs.tiff.TIFFFaxCodes` (in 3.2.8, page 155)
- `net.sourceforge.jiu.codecs.tiff.TIFFImageFileDirectory` (in 3.2.9, page 157)
- `net.sourceforge.jiu.codecs.tiff.TIFFRational` (in 3.2.10, page 164)
- `net.sourceforge.jiu.codecs.tiff.TIFFTag` (in 3.2.11, page 166)
- `net.sourceforge.jiu.color.WebsafePaletteCreator` (in 4.2.2, page 172)
- `net.sourceforge.jiu.color.analysis.MatrixCreator` (in 7.1.3, page 215)
- `net.sourceforge.jiu.color.conversion.CMYKConversion` (in 8.1.1, page 223)
- `net.sourceforge.jiu.color.conversion.LogLuvConversion` (in 8.1.2, page 225)
- `net.sourceforge.jiu.color.conversion.PCDYCbCrConversion` (in 8.1.3, page 227)
- `net.sourceforge.jiu.color.data.ArrayHistogram1D` (in 5.2.1, page 184)
- `net.sourceforge.jiu.color.data.BaseCoOccurrenceFrequencyMatrix` (in 5.2.2, page 186)
 - `net.sourceforge.jiu.color.data.MemoryCoOccurrenceFrequencyMatrix` (in 5.2.3, page 188)
- `net.sourceforge.jiu.color.data.MemoryCoOccurrenceMatrix` (in 5.2.4, page 190)
- `net.sourceforge.jiu.color.data.NaiveHistogram3D` (in 5.2.5, page 192)
- `net.sourceforge.jiu.color.data.OnDemandHistogram3D` (in 5.2.6, page 195)
- `net.sourceforge.jiu.color.dithering.DiamondSpotFunction` (in 9.2.2, page 233)
- `net.sourceforge.jiu.color.dithering.LineSpotFunction` (in 9.2.4, page 240)
- `net.sourceforge.jiu.color.dithering.RoundSpotFunction` (in 9.2.6, page 244)
- `net.sourceforge.jiu.color.io.HistogramSerialization` (in 10.1.1, page 246)
- `net.sourceforge.jiu.color.io.MatrixSerialization` (in 10.1.2, page 248)
- `net.sourceforge.jiu.color.io.PaletteSerialization` (in 10.1.3, page 249)
- `net.sourceforge.jiu.color.quantization.MedianCutNode` (in 12.2.3, page 269)
- `net.sourceforge.jiu.color.quantization.OctreeNode` (in 12.2.6, page 282)
- `net.sourceforge.jiu.color.quantization.RGBColor` (in 12.2.8, page 288)
- `net.sourceforge.jiu.color.quantization.RGBColorComparator` (in 12.2.9, page 290)
- `net.sourceforge.jiu.color.quantization.RGBColorList` (in 12.2.10, page 291)
- `net.sourceforge.jiu.data.MemoryBilevelImage` (in 14.2.1, page 342)
- `net.sourceforge.jiu.data.MemoryByteChannelImage` (in 14.2.2, page 345)
 - `net.sourceforge.jiu.data.MemoryGray8Image` (in 14.2.4, page 352)
 - `net.sourceforge.jiu.data.MemoryPaletted8Image` (in 14.2.5, page 354)
 - `net.sourceforge.jiu.data.MemoryRGB24Image` (in 14.2.6, page 356)
- `net.sourceforge.jiu.data.MemoryShortChannelImage` (in 14.2.8, page 358)
 - `net.sourceforge.jiu.data.MemoryGray16Image` (in 14.2.3, page 350)
 - `net.sourceforge.jiu.data.MemoryRGB48Image` (in 14.2.7, page 357)

- `net.sourceforge.jiu.data.Palette` (in 14.2.9, page 363)
- `net.sourceforge.jiu.filters.BorderSampleGenerator` (in 15.1.2, page 371)
- `net.sourceforge.jiu.filters.ConvolutionKernelData` (in 15.1.3, page 374)
 - `net.sourceforge.jiu.filters.UnsharpMaskKernel` (in 15.1.10, page 390)
- `net.sourceforge.jiu.geometry.ResampleFilter` (in 16.1.11, page 409)
 - `net.sourceforge.jiu.geometry.BSplineFilter` (in 16.1.3, page 396)
 - `net.sourceforge.jiu.geometry.BellFilter` (in 16.1.1, page 393)
 - `net.sourceforge.jiu.geometry.BoxFilter` (in 16.1.2, page 395)
 - `net.sourceforge.jiu.geometry.HermiteFilter` (in 16.1.6, page 400)
 - `net.sourceforge.jiu.geometry.Lanczos3Filter` (in 16.1.7, page 401)
 - `net.sourceforge.jiu.geometry.MitchellFilter` (in 16.1.9, page 404)
 - `net.sourceforge.jiu.geometry.TriangleFilter` (in 16.1.17, page 420)
- `net.sourceforge.jiu.gui.awt.AwtInfo` (in 17.1.1, page 422)
- `net.sourceforge.jiu.gui.awt.BufferedRGB24Image` (in 17.1.4, page 433)
- `net.sourceforge.jiu.gui.awt.ImageCreator` (in 17.1.6, page 440)
- `net.sourceforge.jiu.gui.awt.RGBA` (in 17.1.8, page 448)
- `net.sourceforge.jiu.gui.awt.ToolkitLoader` (in 17.1.9, page 452)
- `net.sourceforge.jiu.gui.awt.dialogs.Dialogs` (in 18.1.2, page 459)
- `net.sourceforge.jiu.ops.Operation` (in 19.2.5, page 508)
 - `net.sourceforge.jiu.codecs.ImageCodec` (in 2.1.5, page 79)
 - `net.sourceforge.jiu.codecs.BMPCodec` (in 2.1.1, page 66)
 - `net.sourceforge.jiu.codecs.GIFCodec` (in 2.1.3, page 71)
 - `net.sourceforge.jiu.codecs.IFFCodec` (in 2.1.4, page 76)
 - `net.sourceforge.jiu.codecs.PCDCodec` (in 2.1.8, page 105)
 - `net.sourceforge.jiu.codecs.PNGCodec` (in 2.1.9, page 109)
 - `net.sourceforge.jiu.codecs.PNMCodec` (in 2.1.10, page 115)
 - `net.sourceforge.jiu.codecs.PSDCodec` (in 2.1.11, page 120)
 - `net.sourceforge.jiu.codecs.PalmCodec` (in 2.1.7, page 96)
 - `net.sourceforge.jiu.codecs.RASCodec` (in 2.1.12, page 122)
 - `net.sourceforge.jiu.codecs.tiff.TIFFCodec` (in 3.2.1, page 137)
 - `net.sourceforge.jiu.color.analysis.Histogram1DCreator` (in 7.1.1, page 210)
 - `net.sourceforge.jiu.color.analysis.Histogram3DCreator` (in 7.1.2, page 213)
 - `net.sourceforge.jiu.color.analysis.MeanDifference` (in 7.1.4, page 217)
 - `net.sourceforge.jiu.color.analysis.TextureAnalysis` (in 7.1.5, page 219)
 - `net.sourceforge.jiu.color.reduction.AutoDetectColorType` (in 13.1.1, page 296)
 - `net.sourceforge.jiu.ops.BatchProcessorOperation` (in 19.2.1, page 495)
 - `net.sourceforge.jiu.apps.ColorIndexer` (in 1.2.1, page 33)
 - `net.sourceforge.jiu.apps.ImageLoadTester` (in 1.2.5, page 45)
 - `net.sourceforge.jiu.ops.ImageToImageOperation` (in 19.2.3, page 501)
 - `net.sourceforge.jiu.color.Invert` (in 4.2.1, page 171)
 - `net.sourceforge.jiu.color.adjustment.HueSaturationValue` (in 6.1.5, page 206)
 - `net.sourceforge.jiu.color.dithering.ClusteredDotDither` (in 9.2.1, page 230)
 - `net.sourceforge.jiu.color.dithering.ErrorDiffusionDithering` (in 9.2.3, page 234)
 - `net.sourceforge.jiu.color.dithering.OrderedDither` (in 9.2.5, page 241)
 - `net.sourceforge.jiu.color.promotion.PromotionGray16` (in 11.1.1, page 252)
 - `net.sourceforge.jiu.color.promotion.PromotionGray8` (in 11.1.2, page 253)
 - `net.sourceforge.jiu.color.promotion.PromotionPaletted8` (in 11.1.3, page 254)
 - `net.sourceforge.jiu.color.promotion.PromotionRGB24` (in 11.1.4, page 255)
 - `net.sourceforge.jiu.color.promotion.PromotionRGB48` (in 11.1.5, page 256)
 - `net.sourceforge.jiu.color.quantization.ArbitraryPaletteQuantizer` (in 12.2.1, page 257)

261)

- `net.sourceforge.jiu.color.quantization.MedianCutContourRemoval` (in 12.2.2, page 261)

264)

- `net.sourceforge.jiu.color.quantization.MedianCutQuantizer` (in 12.2.4, page 273)
- `net.sourceforge.jiu.color.quantization.OctreeColorQuantizer` (in 12.2.5, page 279)
- `net.sourceforge.jiu.color.quantization.PopularityQuantizer` (in 12.2.7, page 285)
- `net.sourceforge.jiu.color.quantization.UniformPaletteQuantizer` (in 12.2.11, page 293)

293)

- `net.sourceforge.jiu.color.reduction.RGBToGrayConversion` (in 13.1.5, page 305)
- `net.sourceforge.jiu.color.reduction.ReduceRGB` (in 13.1.2, page 299)
- `net.sourceforge.jiu.color.reduction.ReduceShadesOfGray` (in 13.1.3, page 301)
- `net.sourceforge.jiu.color.reduction.ReduceToBilevelThreshold` (in 13.1.4, page 303)
- `net.sourceforge.jiu.filters.AreaFilterOperation` (in 15.1.1, page 367)
 - `net.sourceforge.jiu.filters.MaximumFilter` (in 15.1.5, page 381)
 - `net.sourceforge.jiu.filters.MeanFilter` (in 15.1.6, page 383)
 - `net.sourceforge.jiu.filters.MedianFilter` (in 15.1.7, page 385)
 - `net.sourceforge.jiu.filters.MinimumFilter` (in 15.1.8, page 387)
 - `net.sourceforge.jiu.filters.OilFilter` (in 15.1.9, page 388)
- `net.sourceforge.jiu.filters.ConvolutionKernelFilter` (in 15.1.4, page 377)
- `net.sourceforge.jiu.geometry.Crop` (in 16.1.4, page 397)
- `net.sourceforge.jiu.geometry.Flip` (in 16.1.5, page 399)
- `net.sourceforge.jiu.geometry.Mirror` (in 16.1.8, page 402)
- `net.sourceforge.jiu.geometry.Resample` (in 16.1.10, page 405)
- `net.sourceforge.jiu.geometry.Rotate180` (in 16.1.12, page 411)
- `net.sourceforge.jiu.geometry.Rotate90Left` (in 16.1.13, page 413)
- `net.sourceforge.jiu.geometry.Rotate90Right` (in 16.1.14, page 414)
- `net.sourceforge.jiu.geometry.ScaleReplication` (in 16.1.15, page 416)
- `net.sourceforge.jiu.geometry.Shear` (in 16.1.16, page 418)
- `net.sourceforge.jiu.ops.LookupTableOperation` (in 19.2.4, page 505)
 - `net.sourceforge.jiu.color.adjustment.Brightness` (in 6.1.1, page 199)
 - `net.sourceforge.jiu.color.adjustment.Contrast` (in 6.1.2, page 201)
 - `net.sourceforge.jiu.color.adjustment.EqualizeHistogram` (in 6.1.3, page 203)
 - `net.sourceforge.jiu.color.adjustment.GammaCorrection` (in 6.1.4, page 204)
 - `net.sourceforge.jiu.color.adjustment.NormalizeHistogram` (in 6.1.6, page 208)
- `net.sourceforge.jiu.ops.ImagesToImageOperation` (in 19.2.2, page 498)
- `net.sourceforge.jiu.util.ArrayConverter` (in 20.2.1, page 517)
- `net.sourceforge.jiu.util.ArrayRotation` (in 20.2.2, page 522)
- `net.sourceforge.jiu.util.ArrayScaling` (in 20.2.3, page 525)
- `net.sourceforge.jiu.util.Median` (in 20.2.4, page 526)
- `net.sourceforge.jiu.util.Sort` (in 20.2.6, page 532)
- `net.sourceforge.jiu.util.Statistics` (in 20.2.7, page 533)
- `net.sourceforge.jiu.util.SystemInfo` (in 20.2.8, page 537)

Interfaces

- `net.sourceforge.jiu.apps.JiuInfo` (in 1.1.1, page 14)
- `net.sourceforge.jiu.apps.MenuIndexConstants` (in 1.1.2, page 15)
- `net.sourceforge.jiu.apps.StringIndexConstants` (in 1.1.3, page 20)
- `net.sourceforge.jiu.codecs.tiff.TIFFConstants` (in 3.1.1, page 132)

- `net.sourceforge.jiu.color.YCbCrIndex` (in 4.1.1, page 170)
- `net.sourceforge.jiu.color.data.CoOccurrenceFrequencyMatrix` (in 5.1.1, page 175)
- `net.sourceforge.jiu.color.data.CoOccurrenceMatrix` (in 5.1.2, page 178)
- `net.sourceforge.jiu.color.data.Histogram1D` (in 5.1.3, page 180)
- `net.sourceforge.jiu.color.data.Histogram3D` (in 5.1.4, page 182)
- `net.sourceforge.jiu.color.dithering.SpotFunction` (in 9.1.1, page 229)
- `net.sourceforge.jiu.color.quantization.RGBQuantizer` (in 12.1.1, page 259)
- `net.sourceforge.jiu.data.GrayImage` (in 14.1.5, page 319)
 - `net.sourceforge.jiu.data.GrayIntegerImage` (in 14.1.6, page 321)
 - `net.sourceforge.jiu.data.BilevelImage` (in 14.1.1, page 311)
 - `net.sourceforge.jiu.data.Gray16Image` (in 14.1.3, page 317)
 - `net.sourceforge.jiu.data.Gray8Image` (in 14.1.4, page 318)
- `net.sourceforge.jiu.data.PalettedImage` (in 14.1.9, page 326)
 - `net.sourceforge.jiu.data.PalettedIntegerImage` (in 14.1.10, page 327)
- `net.sourceforge.jiu.data.PixelImage` (in 14.1.11, page 328)
 - `net.sourceforge.jiu.data.IntegerImage` (in 14.1.7, page 322)
 - `net.sourceforge.jiu.data.ByteChannelImage` (in 14.1.2, page 314)
 - `net.sourceforge.jiu.data.Paletted8Image` (in 14.1.8, page 325)
 - `net.sourceforge.jiu.data.RGB24Image` (in 14.1.12, page 331)
 - `net.sourceforge.jiu.data.RGBIntegerImage` (in 14.1.16, page 336)
 - `net.sourceforge.jiu.data.ShortChannelImage` (in 14.1.17, page 337)
 - `net.sourceforge.jiu.data.RGB48Image` (in 14.1.13, page 332)
 - `net.sourceforge.jiu.data.RGBImage` (in 14.1.14, page 333)
- `net.sourceforge.jiu.data.RGBIndex` (in 14.1.15, page 334)
- `net.sourceforge.jiu.data.TransparencyInformation` (in 14.1.18, page 340)
- `net.sourceforge.jiu.ops.ProgressListener` (in 19.1.1, page 493)
- `net.sourceforge.jiu.util.ComparatorInterface` (in 20.1.1, page 516)

Exceptions

- `java.lang.Object`
 - `java.lang.Throwable`
 - `java.lang.Exception`
 - `net.sourceforge.jiu.ops.OperationFailedException` (in 19.3.2, page 513)
 - `net.sourceforge.jiu.codecs.InvalidFileStructureException` (in 2.2.1, page 125)
 - `net.sourceforge.jiu.codecs.InvalidImageIndexException` (in 2.2.2, page 126)
 - `net.sourceforge.jiu.codecs.UnsupportedCodecModeException` (in 2.2.3, page 127)
 - `net.sourceforge.jiu.codecs.UnsupportedTypeException` (in 2.2.4, page 128)
 - `net.sourceforge.jiu.codecs.WrongFormatException` (in 2.2.5, page 129)
 - `net.sourceforge.jiu.ops.MissingParameterException` (in 19.3.1, page 512)
 - `net.sourceforge.jiu.ops.WrongParameterException` (in 19.3.3, page 514)

Chapter 1

Package net.sourceforge.jiu.apps

Package Contents

Page

Interfaces

JiuInfo	14
<i>Contains several constants with information on JIU.</i>	
MenuIndexConstants	15
<i>Constant int values for all menu items.</i>	
StringIndexConstants	20
<i>Interface of integer index values to be used with (in 1.2.12, page 61).</i>	

Classes

ColorIndexer	33
<i>Loads image files and generates color index information for them.</i>	
dumpcodecs	35
<i>Command line program that lists all codecs registered with ImageLoader.</i>	
EditorState	36
<i>Represents the state of the editor, including image(s), modified flag, current file name and directories and more.</i>	
ImageDescriptionCreator	44
<i>Returns textual descriptions of the properties of JIU image objects.</i>	
ImageLoadTester	45
<i>Command line program that tries to load images from files, thus testing the built-in and / or Toolkit's codecs.</i>	
jiuawt	47
<i>Graphical user interface application based on the AWT (Abstract Windowing Toolkit, part of Java's standard runtime library since 1.0) that demonstrates features of JIU.</i>	
jiuawtapplet	48
<i>Applet version of jiuawt.</i>	
jiuconvert	49
<i>A command line program to convert between file formats.</i>	
MenuWrapper	50
<i>Abstract menu wrapper.</i>	
OperationProcessor	52
<i>Abstract base class for performing JIU operations in combination with an (in 1.2.3, page 36).</i>	
StringLoader	60

<i>This class loads a (in 1.2.12, page 61)resource from a text file.</i>	
Strings	61
<i>String resource for the various apps.</i>	

Smaller and larger applications demonstrating how to use JIU.

1.1 Interfaces

1.1.1 *Interface* JiuInfo

Contains several constants with information on JIU.

Declaration

```
public interface JiuInfo
```

All known subclasses

jiuawtapplet (in 1.2.7, page 48), JiuAwtFrame (in 17.1.7, page 443)

All classes known to implement interface

jiuawtapplet (in 1.2.7, page 48), JiuAwtFrame (in 17.1.7, page 443)

Field summary

JIU_FEEDBACK_ADDRESS Address for feedback on JIU.

JIU_HOMEPAGE Homepage of the JIU project.

JIU_NUMERICAL_VERSION Three int values for the JIU version, in order (major, minor, patch).

JIU_VERSION Version as String, created from (in 1.1.1, page 14).

Fields

- int **JIU_NUMERICAL_VERSION**
 - Three int values for the JIU version, in order (major, minor, patch).
- java.lang.String **JIU_VERSION**
 - Version as String, created from (in 1.1.1, page 14).
- java.lang.String **JIU_HOMEPAGE**
 - Homepage of the JIU project.
- java.lang.String **JIU_FEEDBACK_ADDRESS**
 - Address for feedback on JIU.

1.1.2 Interface MenuIndexConstants

Constant int values for all menu items. Some of these menu items denote operations which can be performed by (in 1.2.10, page 52).

Declaration

```
public interface MenuIndexConstants
```

All known subclasses

OperationProcessor (in 1.2.10, page 52), EditorState (in 1.2.3, page 36), AwtOperationProcessor (in 17.1.3, page 425)

All classes known to implement interface

OperationProcessor (in 1.2.10, page 52), EditorState (in 1.2.3, page 36)

Field summary

```
COLOR
COLOR_ADJUST
COLOR_ADJUST_BRIGHTNESS
COLOR_ADJUST_CONTRAST
COLOR_ADJUST_GAMMA
COLOR_ADJUST_HUESATURATIONVALUE
COLOR_CONVERTTOMINIMUMCOLORTYPE
COLOR_HISTOGRAM
COLOR_HISTOGRAM_COUNTCOLORSUSED
COLOR_HISTOGRAM_EQUALIZE
COLOR_HISTOGRAM_NORMALIZE
COLOR_HISTOGRAM_SAVECOOCCURRENCEFREQUENCYMATRIXAS
COLOR_HISTOGRAM_SAVECOOCCURRENCEMATRIXAS
COLOR_HISTOGRAM_SAVEHISTOGRAMAS
COLOR_HISTOGRAM_TEXTUREPROPERTIES
COLOR_INVERT
COLOR_PALETTE
COLOR_PALETTE_SAVEAS
COLOR_PROMOTE
COLOR_PROMOTE_PROMOTETOGRAY16
COLOR_PROMOTE_PROMOTETOGRAY8
COLOR_PROMOTE_PROMOTETOPALETTED
COLOR_PROMOTE_PROMOTETORGB24
COLOR_PROMOTE_PROMOTETORGB48
COLOR_REDUCE
COLOR_REDUCE_CONVERTTOGRAYSCALE
COLOR_REDUCE_MAPTOARBITRARYPALETTE
COLOR_REDUCE_MEDIANCUT
COLOR_REDUCE_OCTREE
COLOR_REDUCE_REDUCENUMBEROFSHADESOFGRAY
COLOR_REDUCE_REDUCETOBILEVELTHRESHOLD
```


COLOR_REDUCE_UNIFORMPALETTE
EDIT
EDIT_REDO
EDIT_UNDO
FILE
FILE_CLOSE
FILE_EXIT
FILE_OPEN
FILE_SAVEAS
FILE_SAVEAS_GIF
FILE_SAVEAS_PALM
FILE_SAVEAS_PBM
FILE_SAVEAS_PGM
FILE_SAVEAS_PNG
FILE_SAVEAS_PPM
FILE_SAVEAS_SUNRASTER
FILE_SAVEAS_WINDOWSBMP
FILTERS
FILTERS_BLUR
FILTERS_EDGEDETECTION
FILTERS_EMBOSS
FILTERS_HORIZONTALPREWITT
FILTERS_HORIZONTALSOBEL
FILTERS_LITHOGRAPH
FILTERS_MAXIMUM
FILTERS_MEAN
FILTERS_MEDIAN
FILTERS_MINIMUM
FILTERS_OIL
FILTERS_PSYCHEDELICDISTILLATION
FILTERS_SHARPEN
FILTERS_VERTICALPREWITT
FILTERS_VERTICALSOBEL
HELP
HELP_ABOUT
HELP_SYSTEMINFORMATION
NUM_CONSTANTS
TRANSFORMATIONS
TRANSFORMATIONS_CROP
TRANSFORMATIONS_FLIP
TRANSFORMATIONS_MIRROR
TRANSFORMATIONS_ROTATE180
TRANSFORMATIONS_ROTATELEFT90
TRANSFORMATIONS_ROTATERIGHT90
TRANSFORMATIONS_SCALE
TRANSFORMATIONS_SHEAR
VIEW
VIEW_INTERPOLATIONTYPE
VIEW_INTERPOLATIONTYPE_BICUBIC
VIEW_INTERPOLATIONTYPE_BILINEAR

VIEW_INTERPOLATIONTYPE_NEARESTNEIGHBOR
VIEW_SETORIGINALSIZE
VIEW_ZOOMIN
VIEW_ZOOMOUT

Fields

- int **FILE**
- int **FILE_OPEN**
- int **FILE_SAVEAS**
- int **FILE_SAVEAS_GIF**
- int **FILE_SAVEAS_PALM**
- int **FILE_SAVEAS_PBM**
- int **FILE_SAVEAS_PGM**
- int **FILE_SAVEAS_PNG**
- int **FILE_SAVEAS_PPM**
- int **FILE_SAVEAS_SUNRASTER**
- int **FILE_SAVEAS_WINDOWS BMP**
- int **FILE_CLOSE**
- int **FILE_EXIT**
- int **EDIT**
- int **EDIT_UNDO**
- int **EDIT_REDO**
- int **COLOR**
- int **COLOR_ADJUST**
- int **COLOR_ADJUST_BRIGHTNESS**
- int **COLOR_ADJUST_CONTRAST**
- int **COLOR_ADJUST_GAMMA**
- int **COLOR_ADJUST_HUESATURATIONVALUE**
- int **COLOR_HISTOGRAM**
- int **COLOR_HISTOGRAM_COUNTCOLORSUSED**
- int **COLOR_HISTOGRAM_EQUALIZE**

- int **COLOR_HISTOGRAM_NORMALIZE**
- int **COLOR_HISTOGRAM_TEXTUREPROPERTIES**
- int **COLOR_HISTOGRAM_SAVEHISTOGRAMAS**
- int **COLOR_HISTOGRAM_SAVECOOCCURRENCEMATRIXAS**
- int **COLOR_HISTOGRAM_SAVECOOCCURRENCEFREQUENCYMATRIXAS**
- int **COLOR_PALETTE**
- int **COLOR_PALETTE_SAVEAS**
- int **COLOR_PROMOTE**
- int **COLOR_PROMOTE_PROMOTETOPALETTED**
- int **COLOR_PROMOTE_PROMOTETOGRAY8**
- int **COLOR_PROMOTE_PROMOTETOGRAY16**
- int **COLOR_PROMOTE_PROMOTETORGB24**
- int **COLOR_PROMOTE_PROMOTETORGB48**
- int **COLOR_REDUCE**
- int **COLOR_REDUCE_REDUCETOBILEVELTHRESHOLD**
- int **COLOR_REDUCE_REDUCENUMBEROFSHADESOFGRAY**
- int **COLOR_REDUCE_CONVERTTOGRAYSCALE**
- int **COLOR_REDUCE_MEDIANCUT**
- int **COLOR_REDUCE_OCTREE**
- int **COLOR_REDUCE_UNIFORMPALETTE**
- int **COLOR_REDUCE_MAPTOARBITRARYPALETTE**
- int **COLOR_INVERT**
- int **COLOR_CONVERTTOMINIMUMCOLORTYPE**
- int **TRANSFORMATIONS**
- int **TRANSFORMATIONS_FLIP**
- int **TRANSFORMATIONS_MIRROR**
- int **TRANSFORMATIONS_ROTATELEFT90**
- int **TRANSFORMATIONS_ROTATERIGHT90**
- int **TRANSFORMATIONS_ROTATE180**
- int **TRANSFORMATIONS_CROP**

- int **TRANSFORMATIONS_SCALE**
- int **TRANSFORMATIONS_SHEAR**
- int **FILTERS**
- int **FILTERS_BLUR**
- int **FILTERS_SHARPEN**
- int **FILTERS_EDGEDETECTION**
- int **FILTERS_EMBOSS**
- int **FILTERS_PSYCHEDELICDISTILLATION**
- int **FILTERS_LITHOGRAPH**
- int **FILTERS_HORIZONTALSOBEL**
- int **FILTERS_VERTICALSOBEL**
- int **FILTERS_HORIZONTALPREWITT**
- int **FILTERS_VERTICALPREWITT**
- int **FILTERS_MINIMUM**
- int **FILTERS_MAXIMUM**
- int **FILTERS_MEDIAN**
- int **FILTERS_MEAN**
- int **FILTERS_OIL**
- int **VIEW**
- int **VIEW_ZOOMIN**
- int **VIEW_ZOOMOUT**
- int **VIEW_SETORIGINALSIZE**
- int **VIEW_INTERPOLATIONTYPE**
- int **VIEW_INTERPOLATIONTYPE_NEARESTNEIGHBOR**
- int **VIEW_INTERPOLATIONTYPE_BILINEAR**
- int **VIEW_INTERPOLATIONTYPE_BICUBIC**
- int **HELP**
- int **HELP_ABOUT**
- int **HELP_SYSTEMINFORMATION**
- int **NUM_CONSTANTS**

1.1.3 *Interface* StringIndexConstants

Interface of integer index values to be used with (in 1.2.12, page 61).

Declaration

```
public interface StringIndexConstants
```

All known subclasses

Strings (in 1.2.12, page 61), SystemInfo (in 20.2.8, page 537)

All classes known to implement interface

Strings (in 1.2.12, page 61), SystemInfo (in 20.2.8, page 537)

Field summary

ABOUT
ADJUST
ADJUST_BRIGHTNESS
ADJUST_CONTRAST
ADJUST_GAMMA
ADJUST_HUE_SATURATION_AND_VALUE
ALGORITHMS_NONE
APPLY_MAXIMUM_FILTER
APPLY_MEAN_FILTER
APPLY_MEDIAN_FILTER
APPLY_MINIMUM_FILTER
APPLY_OIL_FILTER
BILEVEL
BITS_PER_PIXEL
BLUR
BOTTOM_ROW
BRIGHTNESS_MENU_ITEM
BURKES_ERROR_DIFFUSION
CANCEL
CHOOSE_DITHERING_METHOD
CHOOSE_PALETTE_TYPE
CLOSE
CLOSE_FILE
COLOR
COLOR_IMAGE_QUANTIZATION
CONTOUR_REMOVAL
CONTOUR_REMOVAL_NUM_PASSES
CONTOUR_REMOVAL_TAU
CONTRAST
CONTRAST_MENU_ITEM
CONVERT_TO_GRAYSCALE
CONVERT_TO_MINIMUM_COLOR_TYPE_MENU_ITEM
CORRELATION

COULD_NOT_CREATE_HISTOGRAM
COUNT_COLORS_USED
CPU_ENDIANNES
CPU_ISALIST
CROP_IMAGE
CROP_MENU_ITEM
DISK_SPACE
DISSIMILARITY
DITHERING_METHOD
DITHERING_NONE
DO_YOU_REALLY_WANT_TO_CLOSE_WITHOUT_SAVING
DO_YOU_REALLY_WANT_TO_QUIT_WITHOUT_SAVING
EDGE_DETECTION
EDIT
EDIT_REDO
EDIT_UNDO
EMBOSS
ENERGY
ENTER_BRIGHTNESS_VALUE
ENTER_CONTRAST_VALUE
ENTER_GAMMA_VALUE
ENTER_THRESHOLD_VALUE
ENTER_WINDOW_SIZE
ENTROPY
EQUALIZE_HISTOGRAM_MENU_ITEM
ERROR_DIFFUSION
ERROR_LOADING_IMAGE
ERROR_MESSAGE
ERROR_NO_MORE_THAN_8_BITS
EXIT
FEEDBACK
FILE
FILE_FORMAT_UNKNOWN
FILTERS
FLIP
FLOYD_STEINBERG_ERROR_DIFFUSION
FREE_MEMORY
GAMMA_MENU_ITEM
GIF
GRAYSCALE
HELP
HISTOGRAM
HOMEPAGE
HOMOGENEITY
HORIZONTAL_PREWITT
HORIZONTAL_SOBEL
HUE
HUE_SATURATION_VALUE_MENU_ITEM
IMAGE_TYPE
IMAGE_TYPE_UNKNOWN

INVERT
JARVIS_JUDICE_NINKE_ERROR_DIFFUSION
LEFT_COLUMN
LITHOGRAPH
LOAD_IMAGE_FILE
LOAD_PALETTE
MAINTAIN_ASPECT_RATIO
MAP_TO_ARBITRARY_PALETTE
MAP_TO_ARBITRARY_PALETTE_MENU_ITEM
MAXIMUM_COLOR_DISTANCE
MAXIMUM_FILTER_MENU_ITEM
MEAN_FILTER_MENU_ITEM
MEDIAN_CUT
MEDIAN_CUT_COLOR_QUANTIZATION
MEDIAN_CUT_CONTOUR_REMOVAL
MEDIAN_FILTER_MENU_ITEM
MEMORY
METHOD
METHOD_REPR_COLOR
METHOD_REPR_COLOR_AVERAGE
METHOD_REPR_COLOR_MEDIAN
METHOD_REPR_COLOR_WEIGHTED_AVERAGE
MINIMUM_FILTER_MENU_ITEM
MIRROR
NEW_HEIGHT
NEW_WIDTH
NO
NORMALIZE_HISTOGRAM_MENU_ITEM
NUM_COLORS
NUMBER_OF_BITS
NUMBER_OF_BITS_BLUE
NUMBER_OF_BITS_GREEN
NUMBER_OF_BITS_RED
NUMBER_OF_COLORS_SMALL_ENOUGH
NUMBER_OF_SHADES_OF_GRAY
NUMBER_OF_USED_COLORS
OCTREE_COLOR_QUANTIZATION
OCTREE_COLOR_QUANTIZATION_MENU_ITEM
OIL_FILTER_MENU_ITEM
OK
OPEN
ORDERED_DITHERING
OUTPUT_COLOR_TYPE
OUTPUT_COLOR_TYPE_PALETTERED
OUTPUT_COLOR_TYPE_RGB
OUTPUT_QUALITY_IMPROVEMENT_ALGORITHM
PALETTE_FROM_FILE
PALETTE_MENU_ITEM
PALETTE_PALM_16_COLORS
PALETTE_PALM_16_GRAY

PALETTE_PALM_256_COLORS
PALETTE_PALM_4_GRAY
PALETTE_SAVE_AS_MENU_ITEM
PALETTED
PALM
PIXELS
PORTABLE_BITMAP
PORTABLE_GRAYMAP
PORTABLE_NETWORK_GRAPHICS
PORTABLE_PIXMAP
PROMOTE
PROMOTE_TO_GRAY16
PROMOTE_TO_GRAY8
PROMOTE_TO_PALETTED
PROMOTE_TO_RGB
PROMOTE_TO_RGB24
PROMOTE_TO_RGB48
PROPERTY_JAVA_CLASS_PATH
PROPERTY_JAVA_CLASS_VERSION
PROPERTY_JAVA_HOME
PROPERTY_JAVA_SPECIFICATION_NAME
PROPERTY_JAVA_SPECIFICATION_VENDOR
PROPERTY_JAVA_SPECIFICATION_VERSION
PROPERTY_JAVA_VENDOR
PROPERTY_JAVA_VENDOR_URL
PROPERTY_JAVA_VERSION
PROPERTY_JAVA_VM_NAME
PROPERTY_JAVA_VM_SPECIFICATION_NAME
PROPERTY_JAVA_VM_SPECIFICATION_VENDOR
PROPERTY_JAVA_VM_SPECIFICATION_VERSION
PROPERTY_JAVA_VM_VENDOR
PROPERTY_JAVA_VM_VERSION
PROPERTY_OS_ARCH
PROPERTY_OS_NAME
PROPERTY_OS_VERSION
PSYCHEDELIC_DISTILLATION
QUIT_PROGRAM
REDUCE
REDUCE_NUMBER_OF_SHADES_OF_GRAY
REDUCE_NUMBER_OF_SHADES_OF_GRAY_MENU_ITEM
REDUCE_TO_BILEVEL_ORDERED_DITHERING
REDUCE_TO_BILEVEL_THRESHOLD
REDUCE_TO_BILEVEL_THRESHOLD_MENU_ITEM
RGB_TRUECOLOR
RIGHT_COLUMN
ROTATE_180
ROTATE_90_LEFT
ROTATE_90_RIGHT
ROTATE_OTHER
SATURATION

SAVE_AS
SAVE_COOCCURRENCE_FREQUENCY_MATRIX
SAVE_COOCCURRENCE_FREQUENCY_MATRIX_MENU_ITEM
SAVE_COOCCURRENCE_MATRIX
SAVE_COOCCURRENCE_MATRIX_MENU_ITEM
SAVE_HISTOGRAM_AS
SAVE_HISTOGRAM_AS_MENU_ITEM
SAVE_IMAGE_AS
SAVE_PALETTE
SAVEAS
SCALE
SCALE_IMAGE
SCREEN_RESOLUTION
SET_HUE
SHARPEN
SHEAR_ENTER_ANGLE
SHEAR_IMAGE
SHEAR_MENU_ITEM
SIERRA_ERROR_DIFFUSION
STEVENSON_ARCE_ERROR_DIFFUSION
STUCKI_ERROR_DIFFUSION
SUN_RASTER
SYSTEM
SYSTEM_INFORMATION
TEXTURE_PROPERTIES
TEXTURE_PROPERTIES_MENU_ITEM
TOP_ROW
TOTAL_MEMORY
TOTAL_NUMBER_OF_BITS_AND_COLORS
TRANSFORMATIONS
UNIFORM_PALETTE_COLOR_QUANTIZATION
UNIFORM_PALETTE_COLOR_QUANTIZATION_MENU_ITEM
USED_MEMORY
VALUE
VERTICAL_PREWITT
VERTICAL_SOBEL
VIEW
VIEW_INTERPOLATIONTYPE
VIEW_INTERPOLATIONTYPE_BICUBIC
VIEW_INTERPOLATIONTYPE_BILINEAR
VIEW_INTERPOLATIONTYPE_NEARESTNEIGHBOR
VIEW_SETORIGINALSIZE
VIEW_ZOOMIN
VIEW_ZOOMOUT
WEBSAFE_PALETTE
WINDOW_HEIGHT
WINDOW_WIDTH
WINDOWS_BITMAP
YES

Fields

- int **ERROR_LOADING_IMAGE**
- int **FILE_FORMAT_UNKNOWN**
- int **LOAD_IMAGE_FILE**
- int **SCREEN_RESOLUTION**
- int **COULD_NOT_CREATE_HISTOGRAM**
- int **NUMBER_OF_USED_COLORS**
- int **COUNT_COLORS_USED**
- int **CLOSE**
- int **HELP**
- int **ABOUT**
- int **SYSTEM**
- int **SYSTEM_INFORMATION**
- int **COLOR**
- int **INVERT**
- int **CONVERT_TO_GRAYSCALE**
- int **FILE**
- int **OPEN**
- int **SAVEAS**
- int **EXIT**
- int **PROPERTY_JAVA_VERSION**
- int **PROPERTY_JAVA_VENDOR**
- int **PROPERTY_JAVA_VENDOR_URL**
- int **PROPERTY_JAVA_HOME**
- int **PROPERTY_JAVA_VM_SPECIFICATION_VERSION**
- int **PROPERTY_JAVA_VM_SPECIFICATION_VENDOR**
- int **PROPERTY_JAVA_VM_SPECIFICATION_NAME**
- int **PROPERTY_JAVA_VM_VERSION**
- int **PROPERTY_JAVA_VM_VENDOR**
- int **PROPERTY_JAVA_VM_NAME**

- int **PROPERTY_JAVA_SPECIFICATION_VERSION**
- int **PROPERTY_JAVA_SPECIFICATION_VENDOR**
- int **PROPERTY_JAVA_SPECIFICATION_NAME**
- int **PROPERTY_JAVA_CLASS_VERSION**
- int **PROPERTY_JAVA_CLASS_PATH**
- int **PROPERTY_OS_NAME**
- int **PROPERTY_OS_ARCH**
- int **PROPERTY_OS_VERSION**
- int **HOMEPAGE**
- int **FEEDBACK**
- int **MEDIAN_CUT**
- int **ERROR_MESSAGE**
- int **NUMBER_OF_COLORS_SMALL_ENOUGH**
- int **CPU_ENDIANNES**
- int **CPU_ISALIST**
- int **FREE_MEMORY**
- int **USED_MEMORY**
- int **TOTAL_MEMORY**
- int **SAVE_AS**
- int **TRANSFORMATIONS**
- int **FLIP**
- int **MIRROR**
- int **MEDIAN_CUT_CONTOUR_REMOVAL**
- int **PROMOTE_TO_RGB**
- int **REDUCE_TO_BILEVEL_THRESHOLD_MENU_ITEM**
- int **REDUCE_TO_BILEVEL_ORDERED_DITHERING**
- int **ROTATE_90_LEFT**
- int **ROTATE_90_RIGHT**
- int **ROTATE_180**
- int **ROTATE_OTHER**
- int **SCALE**

- int **SCALE_IMAGE**
- int **NEW_WIDTH**
- int **NEW_HEIGHT**
- int **MAINTAIN_ASPECT_RATIO**
- int **OK**
- int **CANCEL**
- int **NUM_COLORS**
- int **MEDIAN_CUT_COLOR_QUANTIZATION**
- int **OUTPUT_COLOR_TYPE**
- int **METHOD_REPR_COLOR_AVERAGE**
- int **METHOD_REPR_COLOR_WEIGHTED_AVERAGE**
- int **METHOD_REPR_COLOR_MEDIAN**
- int **OUTPUT_COLOR_TYPE_PALETTED**
- int **OUTPUT_COLOR_TYPE_RGB**
- int **METHOD_REPR_COLOR**
- int **CONTOUR_REMOVAL**
- int **METHOD**
- int **FILTERS**
- int **SHARPEN**
- int **BLUR**
- int **EMBOSS**
- int **PSYCHEDELIC_DISTILLATION**
- int **LITHOGRAPH**
- int **MAXIMUM_COLOR_DISTANCE**
- int **PORTABLE_BITMAP**
- int **PORTABLE_GRAYMAP**
- int **PORTABLE_PIXMAP**
- int **UNIFORM_PALETTE_COLOR_QUANTIZATION_MENU_ITEM**
- int **NUMBER_OF_BITS_RED**
- int **NUMBER_OF_BITS_GREEN**
- int **NUMBER_OF_BITS_BLUE**

- int **ORDERED_DITHERING**
- int **DITHERING_NONE**
- int **DITHERING_METHOD**
- int **UNIFORM_PALETTE_COLOR_QUANTIZATION**
- int **SAVE_IMAGE_AS**
- int **ERROR_NO_MORE_THAN_8_BITS**
- int **TOTAL_NUMBER_OF_BITS_AND_COLORS**
- int **EDGE_DETECTION**
- int **REDUCE_NUMBER_OF_SHADES_OF_GRAY**
- int **REDUCE_NUMBER_OF_SHADES_OF_GRAY_MENU_ITEM**
- int **NUMBER_OF_BITS**
- int **NUMBER_OF_SHADES_OF_GRAY**
- int **SUN_RASTER**
- int **ADJUST**
- int **CONTRAST_MENU_ITEM**
- int **BRIGHTNESS_MENU_ITEM**
- int **GAMMA_MENU_ITEM**
- int **ADJUST_CONTRAST**
- int **ENTER_CONTRAST_VALUE**
- int **ADJUST_BRIGHTNESS**
- int **ENTER_BRIGHTNESS_VALUE**
- int **ADJUST_GAMMA**
- int **ENTER_GAMMA_VALUE**
- int **CROP_IMAGE**
- int **LEFT_COLUMN**
- int **TOP_ROW**
- int **RIGHT_COLUMN**
- int **BOTTOM_ROW**
- int **CROP_MENU_ITEM**
- int **CONVERT_TO_MINIMUM_COLOR_TYPE_MENU_ITEM**
- int **HISTOGRAM**

- int **REDUCE_TO_BILEVEL_THRESHOLD**
- int **ENTER_THRESHOLD_VALUE**
- int **FLOYD_STEINBERG_ERROR_DIFFUSION**
- int **STUCKI_ERROR_DIFFUSION**
- int **BURKES_ERROR_DIFFUSION**
- int **SIERRA_ERROR_DIFFUSION**
- int **JARVIS_JUDICE_NINKE_ERROR_DIFFUSION**
- int **STEVENSON_ARCE_ERROR_DIFFUSION**
- int **OUTPUT_QUALITY_IMPROVEMENT_ALGORITHM**
- int **ALGORITHMS_NONE**
- int **ERROR_DIFFUSION**
- int **COLOR_IMAGE_QUANTIZATION**
- int **HORIZONTAL_SOBEL**
- int **VERTICAL_SOBEL**
- int **HORIZONTAL_PREWITT**
- int **VERTICAL_PREWITT**
- int **SHEAR_MENU_ITEM**
- int **SHEAR_IMAGE**
- int **SHEAR_ENTER_ANGLE**
- int **HUE_SATURATION_VALUE_MENU_ITEM**
- int **ADJUST_HUE_SATURATION_AND_VALUE**
- int **SET_HUE**
- int **HUE**
- int **SATURATION**
- int **VALUE**
- int **MEAN_FILTER_MENU_ITEM**
- int **MEDIAN_FILTER_MENU_ITEM**
- int **OIL_FILTER_MENU_ITEM**
- int **APPLY_MEAN_FILTER**
- int **APPLY_MEDIAN_FILTER**
- int **APPLY_OIL_FILTER**

- int **WINDOW_WIDTH**
- int **WINDOW_HEIGHT**
- int **ENTER_WINDOW_SIZE**
- int **CONTOUR_REMOVAL_NUM_PASSES**
- int **CONTOUR_REMOVAL_TAU**
- int **PALETTE_MENU_ITEM**
- int **PALETTE_SAVE_AS_MENU_ITEM**
- int **MAP_TO_ARBITRARY_PALETTE_MENU_ITEM**
- int **SAVE_PALETTE**
- int **LOAD_PALETTE**
- int **CHOOSE_DITHERING_METHOD**
- int **WEBSAFE_PALETTE**
- int **PALETTE_FROM_FILE**
- int **CHOOSE_PALETTE_TYPE**
- int **MAP_TO_ARBITRARY_PALETTE**
- int **EQUALIZE_HISTOGRAM_MENU_ITEM**
- int **NORMALIZE_HISTOGRAM_MENU_ITEM**
- int **OCTREE_COLOR_QUANTIZATION_MENU_ITEM**
- int **OCTREE_COLOR_QUANTIZATION**
- int **SAVE_COOCCURRENCE_MATRIX**
- int **SAVE_COOCCURRENCE_MATRIX_MENU_ITEM**
- int **SAVE_COOCCURRENCE_FREQUENCY_MATRIX**
- int **SAVE_COOCCURRENCE_FREQUENCY_MATRIX_MENU_ITEM**
- int **WINDOWS_BITMAP**
- int **PROMOTE**
- int **PROMOTE_TO_PALETTED**
- int **PROMOTE_TO_GRAY8**
- int **PROMOTE_TO_GRAY16**
- int **PROMOTE_TO_RGB24**
- int **PROMOTE_TO_RGB48**
- int **REDUCE**

- int **SAVE_HISTOGRAM_AS_MENU_ITEM**
- int **SAVE_HISTOGRAM_AS**
- int **EDIT**
- int **EDIT_UNDO**
- int **EDIT_REDO**
- int **TEXTURE_PROPERTIES_MENU_ITEM**
- int **CONTRAST**
- int **ENERGY**
- int **ENTROPY**
- int **HOMOGENEITY**
- int **TEXTURE_PROPERTIES**
- int **CORRELATION**
- int **DISSIMILARITY**
- int **MINIMUM_FILTER_MENU_ITEM**
- int **MAXIMUM_FILTER_MENU_ITEM**
- int **APPLY_MINIMUM_FILTER**
- int **APPLY_MAXIMUM_FILTER**
- int **VIEW**
- int **VIEW_ZOOMIN**
- int **VIEW_ZOOMOUT**
- int **VIEW_SETORIGINALSIZE**
- int **VIEW_INTERPOLATIONTYPE**
- int **VIEW_INTERPOLATIONTYPE_NEARESTNEIGHBOR**
- int **VIEW_INTERPOLATIONTYPE_BILINEAR**
- int **VIEW_INTERPOLATIONTYPE_BICUBIC**
- int **PALM**
- int **PALETTE_PALM_256_COLORS**
- int **PALETTE_PALM_16_COLORS**
- int **PALETTE_PALM_16_GRAY**
- int **PALETTE_PALM_4_GRAY**
- int **DO_YOU REALLY_WANT_TO_QUIT_WITHOUT_SAVING**

- int **QUIT_PROGRAM**
- int **YES**
- int **NO**
- int **DO_YOU_REALLY_WANT_TO_CLOSE_WITHOUT_SAVING**
- int **CLOSE_FILE**
- int **BITS_PER_PIXEL**
- int **BILEVEL**
- int **GRAYSCALE**
- int **RGB_TRUECOLOR**
- int **IMAGE_TYPE_UNKNOWN**
- int **IMAGE_TYPE**
- int **PALETTED**
- int **PIXELS**
- int **PORTABLE_NETWORK_GRAPHICS**
- int **MEMORY**
- int **DISK_SPACE**
- int **GIF**

1.2 Classes

1.2.1 *Class* ColorIndexer

Loads image files and generates color index information for them.

Declaration

```
public class ColorIndexer
  extends net.sourceforge.jiu.ops.BatchProcessorOperation (in 19.2.1, page 495)
```

Field summary

BLACK
BLUE
COLOR_NAMES
CYAN
GREEN
MAGENTA
RED
WHITE
YELLOW

Constructor summary

ColorIndexer()

Method summary

main(String[])
processFile(String, String, String)

Fields

- public static final int **BLACK**
- public static final int **RED**
- public static final int **GREEN**
- public static final int **BLUE**
- public static final int **YELLOW**
- public static final int **MAGENTA**
- public static final int **CYAN**
- public static final int **WHITE**
- public static final java.lang.String **COLOR_NAMES**

Constructors

- *ColorIndexer*
`public ColorIndexer()`

Methods

- *main*
`public static void main(java.lang.String[] args)`
- *processFile*
`public abstract void processFile(java.lang.String inputDirectory,
java.lang.String inputFileName, java.lang.String outputDirectory)`
 - **Description copied from net.sourceforge.jiu.ops.BatchProcessorOperation (in 19.2.1, page 495)**
Method to be called on each file given to this operation. Non-abstract heirs of this class must implement this method to add functionality.
 - **Parameters**
 - * `inputDirectory` – name of directory where the file to be processed resides
 - * `inputFileName` – name of file to be processed
 - * `outputDirectory` – output directory for that file, need not necessarily be used

1.2.2 Class dumpcodecs

Command line program that lists all codecs registered with ImageLoader. All program arguments are ignored.

Declaration

```
public class dumpcodecs
  extends java.lang.Object
```

Constructor summary

dumpcodecs()

Method summary

main(String[])

Constructors

- *dumpcodecs*
public **dumpcodecs**()

Methods

- *main*
public static void **main**(java.lang.String[] args) throws
java.lang.Exception

1.2.3 Class EditorState

Represents the state of the editor, including image(s), modified flag, current file name and directories and more. This class must not know GUI-specific information like Frame or JFrame objects. These GUI classes (more precisely, the JIU classes that extend them) will have to know EditorState and update according to the information they retrieve from an EditorState object associated with them. EditorState is a pure data container.

Declaration

```
public class EditorState
  extends java.lang.Object
  implements MenuIndexConstants
```

Field summary

DEFAULT_INTERPOLATION The default interpolation type, one of the three INTERPOLATION_xyz constants.

DEFAULT_MAX_REDO_IMAGES The default number of redo steps possible.

DEFAULT_MAX_UNDO_IMAGES The default number of undo steps possible.

INTERPOLATION_BICUBIC Integer constant for bicubic interpolation.

INTERPOLATION_BILINEAR Integer constant for bilinear neighbor interpolation.

INTERPOLATION_NEAREST_NEIGHBOR Integer constant for nearest neighbor interpolation.

ORIGINAL_SIZE_ZOOM_INDEX The index into the (in 1.2.3, page 37)array that holds the original size zoom level (100 percent).

ZOOM_LEVELS All allowed zoom levels, as percentage values in ascending order.

Constructor summary

EditorState() Create new EditorState object and initialize its private fields to default values.

Method summary

addProgressListener(ProgressListener) Adds the argument progress listener to the internal list of progress listeners to be notified by progress updates.

canRedo() Returns if a redo operation is possible right now.

canUndo() Returns if an undo operation is possible right now.

clearRedo()

clearUndo()

ensureStringsAvailable()

getCurrentDirectory() Returns the current directory.

getFileName() Returns the name of the file from which the current image was loaded.

getImage() Returns the image object currently loaded.

getInterpolation() Returns the current interpolation type, one of the INTERPOLATION_xyz constants.

getLocale() Returns the Locale object currently used.

getModified() Returns the current modified state (true if image was modified and not saved after modification, false otherwise).

getProgressListeners() Returns the internal list of progress listeners.

getStartupImageName()

getStrings() Returns the Strings object currently in use.

getZoomFactorX() Returns the current zoom factor in horizontal direction.

getZoomFactorY() Returns the current zoom factor in vertical direction.

getZoomToFit() Returns if image display is currently set to "zoom to fit" Zoom to fit means that the image is always zoomed to fit exactly into the window.

hasImage() Returns if this state encapsulates an image object.

installProgressListeners(Operation) Adds all ProgressListener objects from the internal list of listeners to the argument operation.

isMaximumZoom() Returns if the image is displayed at maximum zoom level.

isMinimumZoom() Returns if the image is displayed at minimum zoom level.

isZoomOriginalSize() Returns if the current zoom level is set to original size (each image pixel is displayed as one pixel).

redo() Perform a redo operation, restore the state before the last undo operation.

resetZoomFactors()

setCurrentDirectory(String) Sets a new current directory.

setFileName(String) Sets a new file name.

setImage(PixelImage, boolean) Sets image and modified state to argument values.

setInterpolation(int) Sets a new interpolation type to be used for display.

setLocale(Locale) Defines a new Locale to be used.

setStartupImageName(String)

setStrings(String) Set new Strings resource.

setZoomFactors(double, double) Sets the zoom factors to the argument values.

undo() Perform an undo step - the previous state will be set, the current state will be saved for a redo operation

zoomIn() Increase the zoom level by one.

zoomOut() Decrease the zoom level by one.

zoomSetOriginalSize() Set the zoom level to 100 percent (1:1).

Fields

- public static final int **DEFAULT_MAX_UNDO_IMAGES**
 - The default number of undo steps possible.
- public static final int **DEFAULT_MAX_REDO_IMAGES**
 - The default number of redo steps possible.
- public static final int **ZOOM_LEVELS**
 - All allowed zoom levels, as percentage values in ascending order.
- public static final int **ORIGINAL_SIZE_ZOOM_INDEX**
 - The index into the (in 1.2.3, page 37)array that holds the original size zoom level (100 percent). So, ZOOM_LEVELS[ORIGINAL_SIZE_ZOOM_INDEX] must be equal to 100.
- public static final int **INTERPOLATION_NEAREST_NEIGHBOR**

- Integer constant for nearest neighbor interpolation. A fast but ugly method.
- public static final int **INTERPOLATION_BILINEAR**
 - Integer constant for bilinear neighbor interpolation. A slow but nice method.
- public static final int **INTERPOLATION_BICUBIC**
 - Integer constant for bicubic interpolation. A very slow method, but with the nicest output of the three supported interpolation types.
- public static final int **DEFAULT_INTERPOLATION**
 - The default interpolation type, one of the three INTERPOLATION_xyz constants.

Constructors

- *EditorState*
public **EditorState**()
 - **Description**
Create new EditorState object and initialize its private fields to default values.

Methods

- *addProgressListener*
public void **addProgressListener**(net.sourceforge.jiu.ops.ProgressListener pl)
 - **Description**
Adds the argument progress listener to the internal list of progress listeners to be notified by progress updates.
 - **Parameters**
* pl – object implementing ProgressListener to be added
- *canRedo*
public boolean **canRedo**()
 - **Description**
Returns if a redo operation is possible right now.
- *canUndo*
public boolean **canUndo**()
 - **Description**
Returns if an undo operation is possible right now.
- *clearRedo*
public void **clearRedo**()
- *clearUndo*
public void **clearUndo**()

- *ensureStringsAvailable*

public void ensureStringsAvailable()
- *getCurrentDirectory*
public java.lang.String getCurrentDirectory()
 - **Description**
Returns the current directory. This directory will be used when file dialogs are opened.

- *getFileName*
public java.lang.String getFileName()
 - **Description**
Returns the name of the file from which the current image was loaded.

- *getImage*
public net.sourceforge.jiu.data.PixelImage getImage()
 - **Description**
Returns the image object currently loaded.

- *getInterpolation*
public int getInterpolation()
 - **Description**
Returns the current interpolation type, one of the INTERPOLATION_xyz constants.

- *getLocale*
public java.util.Locale getLocale()
 - **Description**
Returns the Locale object currently used.

- *getModified*
public boolean getModified()
 - **Description**
Returns the current modified state (true if image was modified and not saved after modification, false otherwise).

- *getProgressListeners*
public java.util.Vector getProgressListeners()
 - **Description**
Returns the internal list of progress listeners.

- *getStartupImageName*
public java.lang.String getStartupImageName()

- *getStrings*
public Strings getStrings()
 - **Description**
Returns the Strings object currently in use.

- *getZoomFactorX*

public double **getZoomFactorX**()

- **Description**

Returns the current zoom factor in horizontal direction. The value 1.0 means that the image is displayed at its original size. Anything smaller means that the image is scaled down, anything larger means that the image is scaled up. The value must not be smaller than or equal to 0.0.

- **Returns** – zoom factor in horizontal direction

- **See also**

* `EditorState.getZoomFactorY()` (in 1.2.3, page 40)

- *getZoomFactorY*

public double **getZoomFactorY**()

- **Description**

Returns the current zoom factor in vertical direction. The value 1.0 means that the image is displayed at its original size. Anything smaller means that the image is scaled down, anything larger means that the image is scaled up. The value must not be smaller than or equal to 0.0.

- **Returns** – zoom factor in vertical direction

- **See also**

* `EditorState.getZoomFactorX()` (in 1.2.3, page 40)

- *getZoomToFit*

public boolean **getZoomToFit**()

- **Description**

Returns if image display is currently set to "zoom to fit" Zoom to fit means that the image is always zoomed to fit exactly into the window.

- *hasImage*

public boolean **hasImage**()

- **Description**

Returns if this state encapsulates an image object.

- *installProgressListeners*

public void **installProgressListeners**(net.sourceforge.jiu.ops.Operation op)

- **Description**

Adds all ProgressListener objects from the internal list of listeners to the argument operation.

- *isMaximumZoom*

public boolean **isMaximumZoom**()

- **Description**

Returns if the image is displayed at maximum zoom level.

- *isMinimumZoom*

public boolean **isMinimumZoom**()

– **Description**

Returns if the image is displayed at minimum zoom level.

- *isZoomOriginalSize*

public boolean isZoomOriginalSize()

– **Description**

Returns if the current zoom level is set to original size (each image pixel is displayed as one pixel).

- *redo*

public void redo()

– **Description**

Perform a redo operation, restore the state before the last undo operation. Before that is done, save the current state for an undo.

- *resetZoomFactors*

public void resetZoomFactors()

- *setCurrentDirectory*

public void setCurrentDirectory(java.lang.String newCurrentDirectory)

– **Description**

Sets a new current directory.

– **Parameters**

* **newCurrentDirectory** – the directory to be used as current directory from now on

- *setFileName*

public void setFileName(java.lang.String newFileName)

– **Description**

Sets a new file name. This is used mostly after a new image was loaded from a file or if the current image is closed (then a null value would be given to this method).

– **Parameters**

* **newFileName** – new name of the current file

- *setImage*

public void setImage(net.sourceforge.jiu.data.PixelImage image, boolean newModifiedState)

– **Description**

Sets image and modified state to argument values.

– **Parameters**

* **image** – new current image

* **newModifiedState** – new state of modified flag

- *setInterpolation*

public void setInterpolation(int newInterpolation)

– **Description**

Sets a new interpolation type to be used for display.

– **Parameters**

* **newInterpolation** – an int for the interpolation type, must be one of the INTERPOLATION_xyz constants

- *setLocale*

public void **setLocale**(java.util.Locale **newLocale**)

- **Description**

Defines a new Locale to be used.

- **Parameters**

* **newLocale** – Locale object used from now on

- **See also**

* `EditorState.setStrings(java.lang.String)` (in 1.2.3, page 42)

- *setStartupImageName*

public void **setStartupImageName**(java.lang.String **name**)

- *setStrings*

public void **setStrings**(java.lang.String **iso639Code**)

- **Description**

Set new Strings resource.

- **Parameters**

* **iso639Code** – language of the new Strings resource

- *setZoomFactors*

public void **setZoomFactors**(double **zoomX**, double **zoomY**)

- **Description**

Sets the zoom factors to the argument values.

- *undo*

public void **undo**()

- **Description**

Perform an undo step - the previous state will be set, the current state will be saved for a redo operation

- **See also**

* `EditorState.redo()` (in 1.2.3, page 41)

- *zoomIn*

public void **zoomIn**()

- **Description**

Increase the zoom level by one.

- **See also**

* `EditorState.zoomOut()` (in 1.2.3, page 42)

* `EditorState.zoomSetOriginalSize()` (in 1.2.3, page 43)

- *zoomOut*

public void **zoomOut**()

- **Description**

Decrease the zoom level by one.

- **See also**

- * `EditorState.zoomIn()` (in 1.2.3, page 42)

- * `EditorState.zoomSetOriginalSize()` (in 1.2.3, page 43)

- *zoomSetOriginalSize*

`public void zoomSetOriginalSize()`

- **Description**

Set the zoom level to 100 percent (1:1). Each image pixel will be displayed as one pixel

- **See also**

- * `EditorState.zoomIn()` (in 1.2.3, page 42)

- * `EditorState.zoomOut()` (in 1.2.3, page 42)

1.2.4 Class ImageDescriptionCreator

Returns textual descriptions of the properties of JIU image objects.

Declaration

```
public class ImageDescriptionCreator
    extends java.lang.Object
```

Method summary

getDescription(PixelImage, Locale, Strings) Returns a description of the argument image using the language as specified by the argument locale's two-letter language code.

Methods

- *getDescription*

```
public static java.lang.String getDescription(
    net.sourceforge.jiu.data.PixelImage image, java.util.Locale locale, Strings
    strings )
```

 - **Description**
Returns a description of the argument image using the language as specified by the argument locale's two-letter language code.
 - **Parameters**
 - * **image** – the image for which a textual description is to be returned
 - * **locale** – the Locale storing the natural language to be used for formatting
 - **Returns** – a textual description of the image

1.2.5 Class ImageLoadTester

Command line program that tries to load images from files, thus testing the built-in and / or Toolkit's codecs. Start the program with file names or directory names as arguments. Directory names will lead to the inclusion of that complete directory tree in the list of files to test. Add the argument `--notoolkit` in order to keep this program from trying to load images via `java.awt.Toolkit`. The program will process all files and print one line per file that tells the result of the loading process. At the end, three lines with statistics (failed / successful / total loading attempts) are printed. Note that you may want to give the JVM more memory if large images are stored in those files you want to test. Example:

```
java -mx300m net.sourceforge.jiu.apps.ImageLoadTester *.jpg
```

This gives 300 MB to the JVM.

Declaration

```
public class ImageLoadTester
  extends net.sourceforge.jiu.ops.BatchProcessorOperation (in 19.2.1, page 495)
```

Constructor summary

ImageLoadTester()

Method summary

main(String[]) Main method of this command line program.

processFile(String, String, String) Tries to load an image from a file.

setUseToolkit(boolean) Specifies whether `java.awt.Toolkit` is supposed to be used when trying to load an image.

Constructors

- *ImageLoadTester*
public **ImageLoadTester**()

Methods

- *main*
public static void **main**(java.lang.String[] args) throws java.lang.Exception
 - **Description**
Main method of this command line program.
 - **Parameters**
 - * **args** – program arguments, provided by the Java Virtual Machine, must be file or directory names
- *processFile*
public void **processFile**(java.lang.String inputDirectory, java.lang.String inputFileName, java.lang.String outputDirectory)

- **Description**

Tries to load an image from a file. Prints a message to standard output and increases certain internal counters for statistics.

- **Parameters**

- * `inputDirectory` – directory where the file resides
 - * `inputFileName` – name of file
 - * `outputDirectory` – not used, this argument is demanded by the parent class+
-

- *setUseToolkit*

```
public void setUseToolkit( boolean newValue )
```

- **Description**

Specifies whether java.awt.Toolkit is supposed to be used when trying to load an image.

- **Parameters**

- * `newValue` – boolean, true if Toolkit is to be used, false otherwise

1.2.6 Class jiuawt

Graphical user interface application based on the AWT (Abstract Windowing Toolkit, part of Java's standard runtime library since 1.0) that demonstrates features of JIU.

Memory shortage

One of the errors experienced most frequently with jiuawt is the 'out of memory' error. Note that only whoever starts jiuawt can give it more memory by giving more memory to the Java virtual machine. Example:

`java -mx300m jiu.jar` starts jiuawt and provides it with 300 MB of memory.

Command line switches

`--dir DIRECTORY` set working directory to `DIRECTORY`

`--help` print help screen to standard output and exit

`--lang LANGUAGE` set language to `LANGUAGE`, where `en` is English, `de` is German and `es` is Spanish

`--system` print system information to standard output and exit

`--version` print version information to standard output and exit

Declaration

```
public class jiuawt
extends java.lang.Object
```

Method summary

main(String[]) Creates a (in 17.1.7, page 443)object.

Methods

- *main*

```
public static void main( java.lang.String[] args )
```

- **Description**

Creates a (in 17.1.7, page 443)object.

- **Parameters**

- * **args** – program arguments, call jiuawt with `--help` as single argument to get a help screen

1.2.7 Class jiuawtapplet

Applet version of jiuawt. Not really useful because no images can be loaded.

Declaration

```
public class jiuawtapplet
  extends java.applet.Applet
  implements JiuInfo
```

Constructor summary

jiuawtapplet()

Method summary

getAppletInfo()
init()
start()
stop()

Serializable Fields

- private jiuawt **jiuawtObject**

Constructors

- *jiuawtapplet*
public jiuawtapplet()

Methods

- *getAppletInfo*
public java.lang.String getAppletInfo()
- *init*
public void init()
- *start*
public void start()
- *stop*
public void stop()

1.2.8 Class jiuconvert

A command line program to convert between file formats.

Declaration

```
public class jiuconvert  
extends java.lang.Object
```

Method summary

main(String[])

Methods

- *main*
public static void **main**(java.lang.String[] args)

1.2.9 Class MenuWrapper

Abstract menu wrapper. A menu consists of a number of menu elements, each of which have a text, an enabled status and an int constant from MenuIndexConstants associated with it.

Declaration

```
public abstract class MenuWrapper
extends java.lang.Object
```

All known subclasses

AwtMenuWrapper (in 17.1.2, page 423)

Constructor summary

MenuWrapper()

Method summary

- findIndex(Object)** Attempts to find the index of a given object that represents a menu element.
- getStringIndex(int)** For one of the values in (in 1.1.2, page 15), returns the corresponding constant in (in 1.1.3, page 20).
- setEnabled(int, boolean)** Sets the enabled status of one of the menu items to either true or false.
- setLabel(int, String)** Sets the text of one of the menu elements to a new value.

Constructors

- *MenuWrapper*
public **MenuWrapper**()

Methods

- *findIndex*
public abstract int **findIndex**(java.lang.Object o)
 - **Description**
Attempts to find the index of a given object that represents a menu element.
 - **Parameters**
* o – some object representing part of the menu
 - **Returns** – corresponding index value from (in 1.1.2, page 15) on success or -1 on failure
- *getStringIndex*
public int **getStringIndex**(int menuIndex)
 - **Description**
For one of the values in (in 1.1.2, page 15), returns the corresponding constant in (in 1.1.3, page 20).

- **Parameters**

- * **menuIndex** – int value from the MenuIndexConstants interface

- **Returns** – int value from the StringIndexConstants interface

- *setEnabled*

```
public abstract void setEnabled( int index, boolean enabled )
```

- **Description**

Sets the enabled status of one of the menu items to either `true` or `false`.

- **Parameters**

- * **index** – menu index of the component whose status is to be reset
 - * **enabled** – boolean with the new value

- *setLabel*

```
public abstract void setLabel( int index, java.lang.String text )
```

- **Description**

Sets the text of one of the menu elements to a new value. This method is usually called when the language settings have changed and new words have to be assigned.

- **Parameters**

- * **index** – integer index of the menu element
 - * **text** – new text value to be used for this element

1.2.10 *Class* OperationProcessor

Abstract base class for performing JIU operations in combination with an (in 1.2.3, page 36).

Declaration

```
public abstract class OperationProcessor
extends java.lang.Object
implements MenuIndexConstants
```

All known subclasses

AwtOperationProcessor (in 17.1.3, page 425)

Constructor summary

OperationProcessor(EditorState) Create an object of this class, storing the state argument for later use.

Method summary

colorAdjustBrightness() Adjust the brightness of the current image.
colorAdjustContrast() Adjust the contrast of the current image.
colorAdjustGamma() Adjust the gamma value of the current image.
colorAdjustHueSaturationValue() Adjust hue, saturation and value of the current image.
colorConvertToMinimumColorType()
colorHistogramCountColorsUsed() Count the number of colors used in the current image.
colorHistogramEqualize()
colorHistogramNormalize()
colorHistogramSaveCoOccurrenceFrequencyMatrixAs()
colorHistogramSaveCoOccurrenceMatrixAs()
colorHistogramSaveHistogramAs()
colorHistogramTextureProperties()
colorInvert()
colorPaletteSaveAs()
colorPromotePromoteToGray16()
colorPromotePromoteToGray8()
colorPromotePromoteToPaletted()
colorPromotePromoteToRgb24()
colorPromotePromoteToRgb48()
colorReduceConvertToGrayscale()
colorReduceMapToArbitraryPalette()
colorReduceMedianCut()
colorReduceOctree()
colorReduceReduceNumberOfShadesOfGray()
colorReduceReduceToBilevelThreshold()
colorReduceUniformPalette()
editRedo()
editUndo()

fileClose() If there is an image loaded in the application, remove the image.
fileExit() Terminate the application.
fileOpen() Load an image in the application.
fileSaveAsBmp() Save the current image as a Windows BMP file.
fileSaveAsGif() Save the current image as a GIF file.
fileSaveAsPalm() Save the current image as a Palm image file.
fileSaveAsPbm() Save the current image as a Portable Bitmap file.
fileSaveAsPgm() Save the current image as a Portable Graymap file.
fileSaveAsPng() Save the current image as a Portable Network Graphics file.
fileSaveAsPpm() Save the current image as a Portable Pixmap file.
fileSaveAsRas() Save the current image as a Sun Raster file.
filtersBlur()
filtersEdgeDetection()
filtersEmboss()
filtersHorizontalPrewitt()
filtersHorizontalSobel()
filtersLithograph()
filtersMaximum()
filtersMean()
filtersMedian()
filtersMinimum()
filtersOil()
filtersPsychedelicDistillation()
filtersSharpen()
filtersVerticalPrewitt()
filtersVerticalSobel()
getEditorState() Returns the EditorState object given to this object's constructor.
helpAbout() Display information about the application: name, version, feedback email address, website.
helpSystemInformation() Display information on the system this application is currently running on.
isAvailable(int) Returns if the operation given by the menu index (from (in 1.1.2, page 15) is available regarding the current editor state.
process(int)
transformationsCrop()
transformationsFlip()
transformationsMirror()
transformationsRotate180()
transformationsRotate90Left()
transformationsRotate90Right()
transformationsScale()
transformationsShear()
viewInterpolationTypeBicubic()
viewInterpolationTypeBilinear()
viewInterpolationTypeNearestNeighbor()
viewSetOriginalSize()
viewZoomIn()
viewZoomOut()

Constructors

- *OperationProcessor*
public OperationProcessor(EditorState editorState)
 - **Description**
 Create an object of this class, storing the state argument for later use.
 - **Parameters**
 * **editorState** – EditorState object to be used for processing

Methods

- *colorAdjustBrightness*
public abstract void colorAdjustBrightness()
 - **Description**
 Adjust the brightness of the current image.
- *colorAdjustContrast*
public abstract void colorAdjustContrast()
 - **Description**
 Adjust the contrast of the current image.
- *colorAdjustGamma*
public abstract void colorAdjustGamma()
 - **Description**
 Adjust the gamma value of the current image.
- *colorAdjustHueSaturationValue*
public abstract void colorAdjustHueSaturationValue()
 - **Description**
 Adjust hue, saturation and value of the current image.
- *colorConvertToMinimumColorType*
public abstract void colorConvertToMinimumColorType()
- *colorHistogramCountColorsUsed*
public abstract void colorHistogramCountColorsUsed()
 - **Description**
 Count the number of colors used in the current image.
- *colorHistogramEqualize*
public abstract void colorHistogramEqualize()
- *colorHistogramNormalize*
public abstract void colorHistogramNormalize()

- *colorHistogramSaveCoOccurrenceFrequencyMatrixAs*
public abstract void colorHistogramSaveCoOccurrenceFrequencyMatrixAs()
- *colorHistogramSaveCoOccurrenceMatrixAs*
public abstract void colorHistogramSaveCoOccurrenceMatrixAs()
- *colorHistogramSaveHistogramAs*
public abstract void colorHistogramSaveHistogramAs()
- *colorHistogramTextureProperties*
public abstract void colorHistogramTextureProperties()
- *colorInvert*
public abstract void colorInvert()
- *colorPaletteSaveAs*
public abstract void colorPaletteSaveAs()
- *colorPromotePromoteToGray16*
public abstract void colorPromotePromoteToGray16()
- *colorPromotePromoteToGray8*
public abstract void colorPromotePromoteToGray8()
- *colorPromotePromoteToPaletted*
public abstract void colorPromotePromoteToPaletted()
- *colorPromotePromoteToRgb24*
public abstract void colorPromotePromoteToRgb24()
- *colorPromotePromoteToRgb48*
public abstract void colorPromotePromoteToRgb48()
- *colorReduceConvertToGrayscale*
public abstract void colorReduceConvertToGrayscale()
- *colorReduceMapToArbitraryPalette*
public abstract void colorReduceMapToArbitraryPalette()
- *colorReduceMedianCut*
public abstract void colorReduceMedianCut()
- *colorReduceOctree*
public abstract void colorReduceOctree()
- *colorReduceReduceNumberOfShadesOfGray*
public abstract void colorReduceReduceNumberOfShadesOfGray()
- *colorReduceReduceToBilevelThreshold*
public abstract void colorReduceReduceToBilevelThreshold()
- *colorReduceUniformPalette*
public abstract void colorReduceUniformPalette()
- *editRedo*
public abstract void editRedo()
- *editUndo*
public abstract void editUndo()

- *fileClose*
public abstract void **fileClose**()
 - **Description**
If there is an image loaded in the application, remove the image.

- *fileExit*
public abstract void **fileExit**()
 - **Description**
Terminate the application. If changes were not saved, the user should be asked whether these changes should be discarded.

- *fileOpen*
public abstract void **fileOpen**()
 - **Description**
Load an image in the application.

- *fileSaveAsBmp*
public abstract void **fileSaveAsBmp**()
 - **Description**
Save the current image as a Windows BMP file.

- *fileSaveAsGif*
public abstract void **fileSaveAsGif**()
 - **Description**
Save the current image as a GIF file.

- *fileSaveAsPalm*
public abstract void **fileSaveAsPalm**()
 - **Description**
Save the current image as a Palm image file.

- *fileSaveAsPbm*
public abstract void **fileSaveAsPbm**()
 - **Description**
Save the current image as a Portable Bitmap file.

- *fileSaveAsPgm*
public abstract void **fileSaveAsPgm**()
 - **Description**
Save the current image as a Portable Graymap file.

- *fileSaveAsPng*
public abstract void **fileSaveAsPng**()
 - **Description**
Save the current image as a Portable Network Graphics file.

- *fileSaveAsPpm*
 public abstract void **fileSaveAsPpm**()
 – **Description**
 Save the current image as a Portable Pixmap file.

- *fileSaveAsRas*
 public abstract void **fileSaveAsRas**()
 – **Description**
 Save the current image as a Sun Raster file.

- *filtersBlur*
 public abstract void **filtersBlur**()

- *filtersEdgeDetection*
 public abstract void **filtersEdgeDetection**()

- *filtersEmboss*
 public abstract void **filtersEmboss**()

- *filtersHorizontalPrewitt*
 public abstract void **filtersHorizontalPrewitt**()

- *filtersHorizontalSobel*
 public abstract void **filtersHorizontalSobel**()

- *filtersLithograph*
 public abstract void **filtersLithograph**()

- *filtersMaximum*
 public abstract void **filtersMaximum**()

- *filtersMean*
 public abstract void **filtersMean**()

- *filtersMedian*
 public abstract void **filtersMedian**()

- *filtersMinimum*
 public abstract void **filtersMinimum**()

- *filtersOil*
 public abstract void **filtersOil**()

- *filtersPsychedelicDistillation*
 public abstract void **filtersPsychedelicDistillation**()

- *filtersSharpen*
 public abstract void **filtersSharpen**()

- *filtersVerticalPrewitt*
 public abstract void **filtersVerticalPrewitt**()

- *filtersVerticalSobel*
 public abstract void **filtersVerticalSobel**()

- *getEditorState*
 public EditorState **getEditorState**()

 - **Description**
 Returns the EditorState object given to this object's constructor.
 - **Returns** – EditorState object used by this processor
- *helpAbout*
 public abstract void **helpAbout**()

 - **Description**
 Display information about the application: name, version, feedback email address, website.
- *helpSystemInformation*
 public abstract void **helpSystemInformation**()

 - **Description**
 Display information on the system this application is currently running on.
- *isAvailable*
 public boolean **isAvailable**(int **menuIndex**)

 - **Description**
 Returns if the operation given by the menu index (from (in 1.1.2, page 15)) is available regarding the current editor state. This method is used to update the enabled status of menu items so that they reflect what can be done in the current state of an application. Thus, things that cannot be done cannot be chosen in the menu because they are disabled. Example: the File — Save as... items are disabled as long as there is no image loaded, simply because there is nothing to save.
 - **Parameters**
 * **menuIndex** – index of menu item to be checked
 - **Returns** – whether the operation is available (if true, menu item should be enabled)
- *process*
 public void **process**(int **menuIndex**)

- *transformationsCrop*
 public abstract void **transformationsCrop**()

- *transformationsFlip*
 public abstract void **transformationsFlip**()

- *transformationsMirror*
 public abstract void **transformationsMirror**()

- *transformationsRotate180*
 public abstract void **transformationsRotate180**()

- *transformationsRotate90Left*
 public abstract void **transformationsRotate90Left**()

- *transformationsRotate90Right*
 public abstract void **transformationsRotate90Right**()

- *transformationsScale*
public abstract void transformationsScale()
- *transformationsShear*
public abstract void transformationsShear()
- *viewInterpolationTypeBicubic*
public abstract void viewInterpolationTypeBicubic()
- *viewInterpolationTypeBilinear*
public abstract void viewInterpolationTypeBilinear()
- *viewInterpolationTypeNearestNeighbor*
public abstract void viewInterpolationTypeNearestNeighbor()
- *viewSetOriginalSize*
public abstract void viewSetOriginalSize()
- *viewZoomIn*
public abstract void viewZoomIn()
- *viewZoomOut*
public abstract void viewZoomOut()

1.2.11 Class StringLoader

This class loads a (in 1.2.12, page 61)resource from a text file. The text file must contain one

Declaration

```
public class StringLoader
  extends java.lang.Object
```

Field summary

resourceDirectory The directory of language resource files, default:
/resources/lang/.

Constructor summary

StringLoader()
StringLoader(String)

Method summary

load()

Fields

- public static java.lang.String **resourceDirectory**
 - The directory of language resource files, default: /resources/lang/.

Constructors

- *StringLoader*
public **StringLoader**()
- *StringLoader*
public **StringLoader**(java.lang.String iso639Code)

Methods

- *load*
public Strings **load**() throws java.io.IOException

1.2.12 Class Strings

String resource for the various apps. Each index value from (in 1.1.3, page 20) has a corresponding String value for all supported natural languages.

Declaration

```
public class Strings
  extends java.lang.Object
  implements StringIndexConstants
```

Field summary

DEFAULT_LANGUAGE Constant of the default language, (in 1.2.12, page 61).

DEFAULT_LANGUAGE_ISO_639_CODE The ISO 639 code for the default language (in 1.2.12, page 61).

LANG_ENGLISH Constant int value for the natural language English.

LANG_GERMAN Constant int value for the natural language German.

LANG_SPANISH Constant int value for the natural language Spanish.

Constructor summary

Strings(Integer, String[]) Create a new String object for the given language and fill it with the String array.

Method summary

determineIsoCodeFromDefaultLocale()

determineSuitableIsoCode() Determines an ISO 639 code of a language suitable for the environment in which the JVM is currently running.

findLanguageCode(String)

get(int) Gets the String denoted by the argument index.

getFileName(int)

getLanguage() Returns the language of this object as one of the LANG_xyz constants of this class.

set(Integer, String[])

Fields

- public static final java.lang.Integer **LANG_ENGLISH**
 - Constant int value for the natural language English.
- public static final java.lang.Integer **LANG_GERMAN**
 - Constant int value for the natural language German.
- public static final java.lang.Integer **LANG_SPANISH**
 - Constant int value for the natural language Spanish.
- public static final java.lang.Integer **DEFAULT_LANGUAGE**
 - Constant of the default language, (in 1.2.12, page 61).

- public static final java.lang.String **DEFAULT_LANGUAGE_ISO_639_CODE**
 - The ISO 639 code for the default language (in 1.2.12, page 61).

Constructors

- *Strings*
 public **Strings**(java.lang.Integer languageConstant, java.lang.String[] stringValues)
 - **Description**
 Create a new String object for the given language and fill it with the String array.

Methods

- *determineIsoCodeFromDefaultLocale*
 public static java.lang.String **determineIsoCodeFromDefaultLocale**()
- *determineSuitableIsoCode*
 public static java.lang.String **determineSuitableIsoCode**()
 - **Description**
 Determines an ISO 639 code of a language suitable for the environment in which the JVM is currently running. First calls (in 1.2.12, page 62). If that yields null, the ISO code for (in 1.2.12, page 61) is returned. So different from (in 1.2.12, page 62) this method always returns a non-null value.
 - **Returns** – String with ISO 639 code of a language that fits the JVM environment, or the default language as fallback solution
- *findLanguageCode*
 public static java.lang.Integer **findLanguageCode**(java.lang.String iso639LanguageCode)
- *get*
 public java.lang.String **get**(int index)
 - **Description**
 Gets the String denoted by the argument index. This index must be one of the int constants defined in (in 1.1.3, page 20).
 - **Returns** – String with given index in the current language
 - **Throws**
 * java.lang.IllegalArgumentException – is not a valid index from (in 1.1.3, page 20)
- *getFileName*
 public static java.lang.String **getFileName**(int languageCode)
- *getLanguage*
 public java.lang.Integer **getLanguage**()
 - **Description**
 Returns the language of this object as one of the LANG_xyz constants of this class.

-
- *set*
`public void set(java.lang.Integer languageConstant, java.lang.String[]
values)`

Chapter 2

Package net.sourceforge.jiu.codecs

Package Contents

Page

Classes

BMPCodec	66
<i>A codec to read and write Windows BMP image files.</i>	
CodecMode	70
<i>This class is an enumeration type for the two modes that an image codec can be used in, (in 2.1.2, page 70) and (in 2.1.2, page 70).</i>	
GIFCodec	71
<i>A codec to write CompuServe GIF (Graphics Interchange Format) files.</i>	
IFFCodec	76
<i>A codec to read Amiga IFF image files.</i>	
ImageCodec	79
<i>The base class for image codecs, operations to read images from or write them to streams.</i>	
ImageLoader	92
<i>A convenience class with static methods to load images from files using JIU codecs.</i>	
PalmCodec	96
<i>A codec to read and write image files in the native image file format of Palm OS (at http://www.palmos.com/), an operating system for handheld devices.</i>	
PCDCodec	105
<i>A codec to read Kodak Photo-CD (image pac) image files.</i>	
PNGCodec	109
<i>A codec for the Portable Network Graphics (PNG) format.</i>	
PNMCodec	115
<i>A codec to read and write Portable Anymap (PNM) image files.</i>	
PSDCodec	120
<i>A codec to read images from Photoshop PSD files.</i>	
RASCodec	122
<i>A codec to read and write Sun Raster (RAS) image files.</i>	

Provides classes to read images from and save them to files (or streams) in various file formats. In some cases, it will be sufficient for codecs to use `File` and `InputStream`. This approach should be picked when possible, as it allows for maximum flexibility—input and output streams can be files, network streams, standard input / output (so that data can be piped on the command line) and more. However, in some cases, it will be necessary for codecs to use `RandomAccessFile` in order to seek to various places in

the file.

Note that the codecs are (sometimes more, sometimes less) far from being finished or even stable. Please do not rely on them for important data. Treat the codecs (and the rest of JIU) as beta software.

Package Specification

All image codecs must extend the `ImageFormat` class (in 2.1.5, page 79). They may support only a subset of all possible flavors of an image file format. As an example, they may choose to support only reading or only writing. As `ImageFormat` extends `ImageFormat` (in 19.2.5, page 508), the progress notification system can (and should) be used. If the codecs are used in GUI (graphical user interface) applications, users could be shown a progress bar—loading and saving can be time-consuming.

`ImageFormat` provides methods to specify a rectangular part of an image. The information should be used by the codec to read or write only that part of the image. That way, loading only a part of a huge image is typically faster and consumes less memory. When saving a part of a huge image, an additional crop operation becomes unnecessary this way.

Related Documentation

Obviously, JIU can only benefit from supporting more file formats. If you want to contribute codecs to JIU (remember that you must provide your code under the GNU General Public License), please contact the maintainer at **the JIU homepage** (at <http://jiu.sourceforge.net>). However, note that no code will be integrated that uses patented algorithms. For better or worse, algorithms like LZW compression (used optionally in TIFF and mandatory in GIF) or arithmetic entropy coding (used in some parts of JPEG) are patented in several countries. In order to use these algorithms, one has to pay license fees. This is not acceptable for JIU and therefore no code will be integrated that uses such algorithms. In cases like GIF or TIFF/LZW that is very unfortunate because these formats have a certain popularity that would make them interesting to support. Read the **GIF section** (at

http://dmoz.org/Computers/Data_Formats/Graphics/2D/GIF/) of the Open Directory, it contains several links to sites that explain the situation (from different points of view).

File format specifications can be found at the following resources:

- **Wotsit.org** (at <http://www.wotsit.org>), one of the biggest repositories for file format specifications
 - **Open Directory** (at http://dmoz.org/Computers/Data_Formats/), section on data formats
-

2.1 Classes

2.1.1 *Class* BMPCodec

A codec to read and write Windows BMP image files.

Typical file extensions are `.bmp` and `.rle` (the latter is only used for compressed files).

Bounds

This codec supports the bounds concept for loading and saving.

Supported BMP types when loading

- Bilevel, 1 bit per pixel, uncompressed. BMP supports palettes for bilevel images, but the content of that palette is ignored and 0 is considered black and 1 white. Any class implementing `ImageObject` (in 14.1.1, page 311) can be given to the codec and it will load the image to that object (if the image's resolution is sufficient). If no image object is given to the codec, a new `ImageObject` (in 14.2.1, page 342) will be created.
- Paletted, 4 bits per pixel, uncompressed or RLE4 compression. Both types are loaded to a `ImageObject` (in 14.1.8, page 325). This requires 50 % more space than is necessary, but there is no dedicated 4 bit image data class in JIU.
- Paletted, 8 bits per pixel, uncompressed or RLE8 compression. Both types are loaded to a `ImageObject` (in 14.1.8, page 325).
- RGB truecolor, 24 bits per pixel, uncompressed. This is loaded to a `ImageObject` (in 14.1.12, page 331).

There is no support for 16 bpp images or `BI_BITFIELDS` compression (for lack of test files).

Supported JIU image data classes when saving to BMP

- `ImageObject` (in 14.1.1, page 311) objects are stored as 1 bit per pixel BMP files.
- `ImageObject` (in 14.1.4, page 318) and `ImageObject` (in 14.1.8, page 325) objects are stored as paletted 8 bits per pixel files. It doesn't really matter how many entries the palette has, the BMP file's palette will always have 256 entries, filled up with zero entries if necessary.
- `ImageObject` (in 14.1.12, page 331) objects are stored as 24 bpp BMP files.

There is no support for compressed BMP files when saving.

I/O classes

BMPCodec works with all input and output classes supported by ImageCodec (`InputStream`, `OutputStream`, `File`, `RandomAccessFile`).

Problems

The RLE-compressed BMP files that I could test this codec on seem to have an end-of-line code at the end of every line instead of relying on the decoder to know when it has unpacked enough bytes for a line. Whenever this codec encounters an EOL symbol and has a current column value of 0, the EOL is ignored.

Usage examples

Write an image to a BMP file.

```
BMPCodec codec = new BMPCodec();
codec.setImage(image);
codec.setFile("out.bmp", CodecMode.SAVE);
codec.process();
codec.close();
```

Read an image from a BMP file.

```
BMPCodec codec = new BMPCodec();
codec.setFile("image.bmp", CodecMode.LOAD);
codec.process();
codec.close();
PixelImage image = codec.getImage();
```

Declaration

```
public class BMPCodec
extends net.sourceforge.jiu.codecs.ImageCodec (in 2.1.5, page 79)
```

Constructor summary

BMPCodec()

Method summary

```
getFileExtensions()
getFormatName()
getMimeTypes()
isLoadingSupported()
isSavingSupported()
process()
suggestFileExtension(PixelImage)
```

Constructors

- *BMPCodec*
public **BMPCodec**()

Methods

- *getFileExtensions*
public java.lang.String[] **getFileExtensions**()
– **Description copied from ImageCodec (in 2.1.5, page 79)**
Returns all file extensions that are typical for this file format. The default implementation in ImageCodec returns null. The file extension strings should include a leading dot and are supposed to be lower case (if that is allowed for the given file format). Example: {".jpg", ".jpeg"} for the JPEG file format.

- **Returns** – String array with typical file extensions
-

- *getFormatName*

public abstract java.lang.String getFormatName()

- **Description copied from ImageCodec (in 2.1.5, page 79)**

Returns the name of the file format supported by this codec. All classes extending (in 2.1.5, page 79) must override this method. When overriding, leave out any words in a particular language so that this format name can be understood by everyone. Usually it is enough to return the format creator plus a typical abbreviation, e.g. **Microsoft BMP** or **Portable Anymap (PNM)**.

- **Returns** – name of the file format supported by this codec
-

- *getMimeTypes*

public abstract java.lang.String[] getMimeTypes()

- **Description copied from ImageCodec (in 2.1.5, page 79)**

Return the **MIME** (at <http://www.faqs.org/rfcs/rfc2045.html>) (Multipurpose Internet Mail Extensions) type strings for this format, or null if none are available.

- **Returns** – MIME type strings or null
-

- *isLoadingSupported*

public abstract boolean isLoadingSupported()

- **Description copied from ImageCodec (in 2.1.5, page 79)**

Returns if this codec is able to load images in the file format supported by this codec. If true is returned this does not necessarily mean that all files in this format can be read, but at least some.

- **Returns** – if loading is supported
-

- *isSavingSupported*

public abstract boolean isSavingSupported()

- **Description copied from ImageCodec (in 2.1.5, page 79)**

Returns if this codec is able to save images in the file format supported by this codec. If true is returned this does not necessarily mean that all types files in this format can be written, but at least some.

- **Returns** – if saving is supported
-

- *process*

public void process() throws
net.sourceforge.jiu.ops.MissingParameterException,
net.sourceforge.jiu.ops.OperationFailedException,
net.sourceforge.jiu.ops.WrongParameterException

- **Description copied from net.sourceforge.jiu.ops.Operation (in 19.2.5, page 508)**

This method does the actual work of the operation. It must be called after all parameters have been given to the operation object.

- **Throws**

- * `net.sourceforge.jiu.ops.WrongParameterException` – if at least one of the input parameters was not initialized appropriately (values out of the valid interval, etc.)
- * `net.sourceforge.jiu.ops.MissingParameterException` – if any mandatory parameter was not given to the operation
- * `net.sourceforge.jiu.ops.OperationFailedException` –

- *suggestFileExtension*

```
public java.lang.String suggestFileExtension(
net.sourceforge.jiu.data.PixelImage image )
```

- **Description copied from ImageCodec (in 2.1.5, page 79)**

Attempts to suggest a filename extension. The type of the argument image will be taken into consideration, although this will be necessary for some file formats only (as an example, PNM has different extensions for different image types, see (in 2.1.10, page 115)). This default implementation always returns `null`.

- **Parameters**

- * `image` – the image that is to be written to a file

- **Returns** – the file extension, including a leading dot, or `null` if no file extension can be recommended

2.1.2 Class CodecMode

This class is an enumeration type for the two modes that an image codec can be used in, (in 2.1.2, page 70) and (in 2.1.2, page 70). These values are used as arguments in some of the methods of (in 2.1.5, page 79).

Declaration

```
public final class CodecMode
    extends java.lang.Object
```

Field summary

LOAD Codec mode load, one of the two possible values of CodecMode.
SAVE Codec mode save, one of the two possible values of CodecMode.

Fields

- public static final CodecMode **LOAD**
 - Codec mode load, one of the two possible values of CodecMode. To be used with a codec to indicate that an image is to be read from a stream.
- public static final CodecMode **SAVE**
 - Codec mode save, one of the two possible values of CodecMode. To be used with a codec to indicate that an image is to be written to a stream.

2.1.3 Class GIFCodec

A codec to write Compuserve GIF (Graphics Interchange Format) files.

Only writing GIF files is supported right now. Reading GIF files with JIU can be done with the (in 17.1.9, page 452)class which uses the image reader built into the Java runtime library (class). That reader has supported GIF since Java 1.0.

Supported image types

When saving, classes implementing the following image data interfaces are supported: (in 14.1.1, page 311), (in 14.1.4, page 318)and (in 14.1.8, page 325). GIF only supports up to 256 colors in an image, so you will have to use one of the quantization classes to reduce a truecolor image to 256 or less colors before you can save it with this codec.

Supported I/O classes

This codec supports , and .

Bounds

(in 2.1.5, page 79)'s bounds concept is supported. A user of this codec can specify a rectangular part of the input image that will be saved instead of the complete image.

Comments

GIF - at least in its 89a version - allows for the inclusion of textual comments. When saving an image to a GIF file, each comment given to a codec will be stored in a comment extension block of its own.

Usage example

Save an image using this codec:

```
GIFCodec codec = new GIFCodec();
codec.appendComment("Bob and Susan at the Munich airport (2002-06-13).");
codec.setImage(image); // BilevelImage, Gray8Image or Paletted8Image
codec.setInterlacing(true);
codec.setFile("output.gif", CodecMode.SAVE);
codec.process();
```

Interlaced storage

This codec allows creating interlaced and non-interlaced GIF files. The default is non-interlaced storage. Non-interlaced files store the rows top-to-bottom.

Interlaced files store the image in four passes, progressively adding rows until the complete image is stored. When decoding, the progressive display of interlaced files makes it supposedly quicker to find out what's displayed in the image.

On the other hand, transmission typically takes longer, because interlacing often leads to slightly larger files. When using interlaced mode, lines that get stored one after another have some room between them in the image, so there are less similarities between consecutive lines, which worsens compression ratio (compression works better with a lot of similarities in the data to be compressed).

GIF versions

There are two versions of GIF, 87a and 89a. In 89a, several things were added to the file format specification. From the 89a features this codec only uses the possibility of storing textual comments in GIF files. Thus, the version used for writing depends on the return value of (in 2.1.5, page 86). If there is at least one comment to be written to the file, version 89a will be used, 87a otherwise.

Licensing of the LZW algorithm

Unisys Corp. had a patent in several countries on the LZW algorithm used within GIF. However, this patent has expired (Japan being the last country where the patent expired, on July 7th 2004) so that LZW can be used freely.

Licensing of the file format

GIF was defined by Compuserve. In a technical document file called `Gif89a.txt` that I found somewhere on the Net they grant a royalty-free license for use of the format to anyone - in order to improve the popularity of the format, I guess. I don't think that it should be possible to put a file format under a copyright, but all that Compuserve asks for in exchange for freely using the format is the inclusion of a message. So, here is that message: "The Graphics Interchange Format(c) is the Copyright property of CompuServe Incorporated. GIF(sm) is a Service Mark property of CompuServe Incorporated."

File format background

I've compiled a web page with **technical information on GIF** (at <http://www.geocities.com/marcoschmidt.geo/gif-image-file-format.html>).

Declaration

```
public class GIFCodec
  extends net.sourceforge.jiu.codecs.ImageCodec (in 2.1.5, page 79)
```

Constructor summary

GIFCodec()

Method summary

getBackgroundColor() Returns the index of the background color.
getFileExtensions()
getFormatName()
getMimeTypes()
isInterlaced() Returns if the image is or will be stored in interlaced (**true**) or non-interlaced mode (**false**).
isLoadingSupported()
isSavingSupported()
process()
setBackgroundColor(int) Specify the value of the background color.
setInterlacing(boolean) Specifies whether the image will be stored in interlaced mode (**true**) or non-interlaced mode (**false**).

Constructors

- *GIFCodec*
public **GIFCodec**()

Methods

- *getBackgroundColor*
public int **getBackgroundColor**()
 - **Description**
Returns the index of the background color.
 - **Returns** – int value with the color (index into the palette) of the background color
 - **See also**
* `GIFCodec.setBackgroundColor(int)` (in 2.1.3, page 74)
- *getFileExtensions*
public java.lang.String[] **getFileExtensions**()
 - **Description copied from ImageCodec (in 2.1.5, page 79)**
Returns all file extensions that are typical for this file format. The default implementation in ImageCodec returns null. The file extension strings should include a leading dot and are supposed to be lower case (if that is allowed for the given file format). Example: {".jpg", ".jpeg"} for the JPEG file format.
 - **Returns** – String array with typical file extensions
- *getFormatName*
public abstract java.lang.String **getFormatName**()
 - **Description copied from ImageCodec (in 2.1.5, page 79)**
Returns the name of the file format supported by this codec. All classes extending (in 2.1.5, page 79) must override this method. When overriding, leave out any words in a particular language so that this format name can be understood by everyone. Usually it is enough to return the format creator plus a typical abbreviation, e.g. Microsoft BMP or Portable Anymap (PNM).
 - **Returns** – name of the file format supported by this codec
- *getMimeTypes*
public abstract java.lang.String[] **getMimeTypes**()
 - **Description copied from ImageCodec (in 2.1.5, page 79)**
Return the **MIME** (at <http://www.faqs.org/rfcs/rfc2045.html>) (Multipurpose Internet Mail Extensions) type strings for this format, or null if none are available.
 - **Returns** – MIME type strings or null
- *isInterlaced*
public boolean **isInterlaced**()
 - **Description**
Returns if the image is or will be stored in interlaced (`true`) or non-interlaced mode (`false`).

– **Returns** – interlacing mode

– **See also**

* `GIFCodec.setInterlacing(boolean)` (in 2.1.3, page 75)

• *isLoadingSupported*

`public abstract boolean isLoadingSupported()`

– **Description copied from ImageCodec (in 2.1.5, page 79)**

Returns if this codec is able to load images in the file format supported by this codec. If `true` is returned this does not necessarily mean that all files in this format can be read, but at least some.

– **Returns** – if loading is supported

• *isSavingSupported*

`public abstract boolean isSavingSupported()`

– **Description copied from ImageCodec (in 2.1.5, page 79)**

Returns if this codec is able to save images in the file format supported by this codec. If `true` is returned this does not necessarily mean that all types files in this format can be written, but at least some.

– **Returns** – if saving is supported

• *process*

`public void process() throws`
`net.sourceforge.jiu.ops.MissingParameterException,`
`net.sourceforge.jiu.ops.OperationFailedException,`
`net.sourceforge.jiu.ops.WrongParameterException`

– **Description copied from net.sourceforge.jiu.ops.Operation (in 19.2.5, page 508)**

This method does the actual work of the operation. It must be called after all parameters have been given to the operation object.

– **Throws**

* `net.sourceforge.jiu.ops.WrongParameterException` – if at least one of the input parameters was not initialized appropriately (values out of the valid interval, etc.)

* `net.sourceforge.jiu.ops.MissingParameterException` – if any mandatory parameter was not given to the operation

* `net.sourceforge.jiu.ops.OperationFailedException` –

• *setBackgroundColor*

`public void setBackgroundColor(int colorIndex)`

– **Description**

Specify the value of the background color. Default is 0.

– **Parameters**

* `colorIndex` – int value with the color (index into the palette) of the background color

– **See also**

* `GIFCodec.getBackgroundColor()` (in 2.1.3, page 73)

- *setInterlacing*

`public void setInterlacing(boolean useInterlacing)`

- **Description**

Specifies whether the image will be stored in interlaced mode (`true`) or non-interlaced mode (`false`).

- **Parameters**

* `useInterlacing` – boolean, if true interlaced mode, otherwise non-interlaced mode

- **See also**

* `GIFCodec.isInterlaced()` (in 2.1.3, page 73)

2.1.4 Class IFFCodec

A codec to read Amiga IFF image files. IFF (Interchange File Format) is an Amiga wrapper file format for texts, images, animations, sound and other kinds of data. This codec only deals with image IFF files. Typical file extensions for IFF image files are `.lbm` and `.iff`.

Loading / saving

Only loading is supported by this codec.

Supported file types

Both uncompressed and run-length encoded files are read.

- 1 to 8 bit indexed (paletted) color
- 24 bit RGB truecolor
- HAM6 and HAM8 images (which are a mixture of paletted and truecolor)

Usage example

```
IFFCodec codec = new IFFCodec();
codec.setFile("image.iff", CodecMode.LOAD);
codec.process();
PixelImage image = codec.getImage();
```

Declaration

```
public class IFFCodec
  extends net.sourceforge.jiu.codecs.ImageCodec (in 2.1.5, page 79)
```

Constructor summary

IFFCodec()

Method summary

getFileExtensions()
getFormatName()
getMimeTypes()
isLoadingSupported()
isSavingSupported()
process()

Constructors

- *IFFCodec*
`public IFFCodec()`

Methods

- *getFileExtensions*

```
public java.lang.String[] getFileExtensions( )
```

- **Description copied from ImageCodec (in 2.1.5, page 79)**

Returns all file extensions that are typical for this file format. The default implementation in ImageCodec returns null. The file extension strings should include a leading dot and are supposed to be lower case (if that is allowed for the given file format). Example: {".jpg", ".jpeg"} for the JPEG file format.

- **Returns** – String array with typical file extensions

- *getFormatName*

```
public abstract java.lang.String getFormatName( )
```

- **Description copied from ImageCodec (in 2.1.5, page 79)**

Returns the name of the file format supported by this codec. All classes extending (in 2.1.5, page 79) must override this method. When overriding, leave out any words in a particular language so that this format name can be understood by everyone. Usually it is enough to return the format creator plus a typical abbreviation, e.g. Microsoft BMP or Portable Anymap (PNM).

- **Returns** – name of the file format supported by this codec

- *getMimeTypes*

```
public abstract java.lang.String[] getMimeTypes( )
```

- **Description copied from ImageCodec (in 2.1.5, page 79)**

Return the **MIME** (at <http://www.faqs.org/rfcs/rfc2045.html>) (Multipurpose Internet Mail Extensions) type strings for this format, or null if none are available.

- **Returns** – MIME type strings or null

- *isLoadingSupported*

```
public abstract boolean isLoadingSupported( )
```

- **Description copied from ImageCodec (in 2.1.5, page 79)**

Returns if this codec is able to load images in the file format supported by this codec. If true is returned this does not necessarily mean that all files in this format can be read, but at least some.

- **Returns** – if loading is supported

- *isSavingSupported*

```
public abstract boolean isSavingSupported( )
```

- **Description copied from ImageCodec (in 2.1.5, page 79)**

Returns if this codec is able to save images in the file format supported by this codec. If true is returned this does not necessarily mean that all types files in this format can be written, but at least some.

- **Returns** – if saving is supported

- *process*

```
public void process( ) throws  
net.sourceforge.jiu.ops.MissingParameterException,  
net.sourceforge.jiu.ops.OperationFailedException,  
net.sourceforge.jiu.ops.WrongParameterException
```

- **Description copied from net.sourceforge.jiu.ops.Operation** (in 19.2.5, page 508)

This method does the actual work of the operation. It must be called after all parameters have been given to the operation object.

- **Throws**

- * `net.sourceforge.jiu.ops.WrongParameterException` – if at least one of the input parameters was not initialized appropriately (values out of the valid interval, etc.)
- * `net.sourceforge.jiu.ops.MissingParameterException` – if any mandatory parameter was not given to the operation
- * `net.sourceforge.jiu.ops.OperationFailedException` –

2.1.5 Class ImageCodec

The base class for image codecs, operations to read images from or write them to streams. A codec should support one file format only. The word codec is derived from enCOder DECoder.

Usage

The codecs differ quite a bit in what they support. But here are two code snippets that demonstrate how to do loading and saving in general.

Load image

```
ImageCodec codec = new BMPCodec(); // BMPCodec is just an example
codec.setFile("image.bmp", CodecMode.LOAD);
codec.process();
PixelImage image = codec.getImage();
```

Save image

```
PixelImage image = ...; // the image to be saved
ImageCodec codec = new BMPCodec(); // BMPCodec is just an example
codec.setFile("image.bmp", CodecMode.SAVE);
codec.setImage(image);
codec.process();
```

I/O objects

There are several set and get methods for I/O objects, including `DataInput`, `DataOutput`, `InputStream`, `OutputStream` and `RandomAccessFile`. If you are just using the codec (and not developing one) make it easier for yourself and use `InputStream` (in 2.1.5, page 89). That way the picking of the right type of I/O class and the creation of a buffered stream wrapper is done automatically. Codecs have different requirements concerning I/O objects. If an image is to be loaded, it's enough for some formats to linearly read from an `InputStream` to load the image. However, some formats (like TIFF) require random access.

When implementing a codec, take care that as many I/O classes as possible can be used. If possible, call `setInputStream` (in 2.1.5, page 86) when loading and `setOutputStream` (in 2.1.5, page 86) when saving. That way, input / output streams, `RandomAccessFiles` and arbitrary `DataInput` / `DataOutput` objects can be used.

Mode

Codecs can be used to save images or load them, or both. As was g; by default, no mode (of enumeration type `CodecMode` (in 2.1.2, page 70)) is specified and `getMode` (in 2.1.5, page 86) returns `null`. Mode only has two possible values, `LOAD` (in 2.1.2, page 70) and `SAVE` (in 2.1.2, page 70). In some cases, the codec can find out whether to load or save from the I/O objects that were given to it; if it has an input stream, something must be loaded, if it has an output stream, something is to be saved. If a codec demands a `Mode`, there is no way to find out the mode automatically, that is why `setMode` (in 2.1.5, page 90) also has an argument of type `Mode` (in 2.1.2, page 70).

Bounds; to load or save only part of an image. Defining bounds is optional; by default, the complete image is loaded or saved (no bounds). Using (in 2.1.5, page 88), one can specify the rectangle which will be loaded or saved.

PixelImage object; get and set methods for the image which is to be loaded or saved. If an image is to be loaded, a PixelImage object can optionally be specified so that the image will be written to that object; image type and resolution must of course match the image from input. Normally, the codec will create the appropriate image object itself. If an image is to be saved, an image object must be provided, otherwise there is nothing to do.

Image index; the index of the image that is to be loaded (int value, default is 0). For image formats that support more than one image in one stream, the index of the image to be loaded (zero-based) can be specified using (in 2.1.5, page 90).

Textual comments

Some file formats allow for the inclusion of textual comments, to store a description, creator, copyright owner or anything else within the image file without actually drawing that text on the image itself. Some codecs support reading and writing of comments.

Other methods

Each file format must be able to return its name ((in 2.1.5, page 85)) and file extensions that are typical for it ((in 2.1.5, page 85)).

A related method suggests a file extension for a given PixelImage object ((in 2.1.5, page 91)).

That method need not be implemented, the default version returns simply `null`. However, it is encouraged that codec implementors provide this method as well. Most file formats only have one typical extension (e. g. `.bmp`). However, for a file format like PNM, the extension depends on the image type (a grayscale image would end in `.pgm`, a color image in `.ppm`).

Declaration

```
public abstract class ImageCodec
extends net.sourceforge.jiu.ops.Operation (in 19.2.5, page 508)
```

All known subclasses

RASCodec (in 2.1.12, page 122), PSDCodec (in 2.1.11, page 120), PNMCodec (in 2.1.10, page 115), PNGCodec (in 2.1.9, page 109), PCDCCodec (in 2.1.8, page 105), PalmCodec (in 2.1.7, page 96), IFFCodec (in 2.1.4, page 76), GIFCodec (in 2.1.3, page 71), BMPCodec (in 2.1.1, page 66), TIFFCodec (in 3.2.1, page 137)

Constructor summary

ImageCodec() This constructor will be called by descendants.

Method summary

appendComment(String) Appends a comment to the internal list of comments.

checkBounds(int, int) If bounds were defined for this codec, this method tests if the bounds rectangle fits into the rectangle (0, 0) / (width - 1, height - 1).

checkImageResolution() If an image object was provided to be used for loading via (in 2.1.5, page 90), this method checks if its resolution is the same as the bounds' resolution.

close() Calls the close method of all input and output I/O objects that were given to this object.

getBoundsHeight() Returns the height of the rectangle specified by bounds.

getBoundsWidth() Returns the width of the rectangle specified by bounds.

getBoundsX1() Returns x coordinate of the upper left corner of the bounds.

getBoundsX2() Returns x coordinate of the lower right corner of the bounds.

getBoundsY1() Returns y coordinate of the upper left corner of the bounds.

getBoundsY2() Returns y coordinate of the lower right corner of the bounds.

getComment(int) Returns a comment from the internal list of comments.

getDataInput() Returns a `Object` if one was provided via `setDataInput()` (in 2.1.5, page 88) or `null` otherwise.

getDataOutput() Returns a `Object` if one was provided via `setDataOutput()` (in 2.1.5, page 89) or `null` otherwise.

getDpiX() Returns the horizontal physical resolution of the image associated with this codec.

getDpiY() Returns the vertical physical resolution of the image associated with this codec.

getFileExtensions() Returns all file extensions that are typical for this file format.

getFormatName() Returns the name of the file format supported by this codec.

getImage() Returns the image object stored in this codec.

getImageIndex() Returns the zero-based index of the image to be loaded.

getInputAsDataInput() Returns a `Object` if one was specified using `setInputAsDataInput()` (in 2.1.5, page 88), or creates a `RandomAccessFile` if an `int` was specified, or returns a `Object` if one was specified (RandomAccessFile implements DataInput).

getInputStream() Returns an `Object` that was given to this codec via `setInputStream()` (in 2.1.5, page 90) or `null` otherwise.

getMimeTypeTypes() Return the **MIME** (at <http://www.faqs.org/rfcs/rfc2045.html>) (Multipurpose Internet Mail Extensions) type strings for this format, or `null` if none are available.

getMode() Returns the mode this codec is in.

getNumComments() Returns the current number of comments in the internal comment list.

getOutputAsDataOutput() Attempts to return an output object as a `Object`.

getOutputStream() Returns an `Object` that was given to this codec via `setOutputStream()` (in 2.1.5, page 90) or `null` otherwise.

getRandomAccessFile() Returns a `Object` that was given to this codec via `setRandomAccessFile()` (in 2.1.5, page 90) or `null` otherwise.

hasBounds() Returns if bounds have been specified.

initModeFromIOObjects()

isLoadingSupported() Returns if this codec is able to load images in the file format supported by this codec.

isRowRequired(int) Returns if an image row given by its number (zero-based) must be loaded in the context of the current bounds.

isSavingSupported() Returns if this codec is able to save images in the file format supported by this codec.

isTileRequired(int, int, int, int) Returns if the tile formed by the argument coordinates form a rectangle that overlaps with the bounds.

removeAllComments() Removes all entries from the internal list of comments.

removeBounds() If bounds were set using `setBounds()` (in 2.1.5, page 88), these bounds are no longer regarded after the call to this method.

setBounds(int, int, int, int) Sets the bounds of a rectangular part of the image that is to be loaded or saved, instead of the complete image.

setBoundsIfNecessary(int, int) If no bounds have been set ((in 2.1.5, page 87) returns *false*), this method will set the bounds to 0, 0, *width - 1*, *height - 1*.

setDataInput(DataInput) Specifies a *DataInput* object to be used for loading.

setDataOutput(DataOutput) Sets a *DataOutput* object to be used for saving an image.

setDpi(int, int) Sets the DPI values to be stored in the file to the argument values.

setFile(File, CodecMode) Gives a *File* object and a codec mode to this codec and attempts to initialize the appropriate I/O objects.

setFile(String, CodecMode) Gives a file name and codec mode to the codec which will then try to create the corresponding I/O object.

setImage(PixelImage) Give an image to this codec to be used for loading an image into it or saving the image.

setImageIndex(int) Sets the index of the image to be loaded to the argument value (which must be zero or larger).

setInputStream(InputStream) An *InputStream* can be given to this codec using this method.

setOutputStream(OutputStream) A method to give an *OutputStream* to this codec to be used for saving an image.

setRandomAccessFile(RandomAccessFile, CodecMode) A method to give a *RandomAccessFile* to this codec to be used for loading or saving an image.

suggestFileExtension(PixelImage) Attempts to suggest a filename extension.

Constructors

- *ImageCodec*

```
public ImageCodec( )
```

- **Description**

This constructor will be called by descendants. The bounds state is initialized to no bounds.

Methods

- *appendComment*

```
public void appendComment( java.lang.String comment )
```

- **Description**

Appends a comment to the internal list of comments. If the argument comment is non-null, it will be added to the internal list of comment strings.

- **Parameters**

* *comment* – the comment to be added

- *checkBounds*

```
public void checkBounds( int width, int height ) throws
net.sourceforge.jiu.ops.WrongParameterException
```

- **Description**

If bounds were defined for this codec, this method tests if the bounds rectangle fits into the rectangle (0, 0) / (width - 1, height - 1). If the bounds are incorrect, a (in

19.3.3, page 514) is thrown, otherwise nothing happens. To be used within codecs that support the bounds concept.

- *checkImageResolution*

public void checkImageResolution() throws
net.sourceforge.jiu.ops.WrongParameterException

- **Description**

If an image object was provided to be used for loading via (in 2.1.5, page 90), this method checks if its resolution is the same as the bounds' resolution. If the two differ, a (in 19.3.3, page 514) is thrown.

- **Throws**

* **net.sourceforge.jiu.ops.WrongParameterException** – if image resolution and bounds dimension differ

- *close*

public void close()

- **Description**

Calls the close method of all input and output I/O objects that were given to this object. Catches and ignores any IOException objects that may be thrown in the process. Note that not all I/O objects have a close method (e.g. and have not).

- *getBoundsHeight*

public int getBoundsHeight()

- **Description**

Returns the height of the rectangle specified by bounds. Bounds must have been specified using (in 2.1.5, page 88), otherwise the return value is undefined. This equals (in 2.1.5, page 84) - (in 2.1.5, page 84) + 1.

- **Returns** – height of bounds rectangle

- *getBoundsWidth*

public int getBoundsWidth()

- **Description**

Returns the width of the rectangle specified by bounds. Bounds must have been specified using (in 2.1.5, page 88), otherwise the return value is undefined. This equals (in 2.1.5, page 83) - (in 2.1.5, page 83) + 1.

- **Returns** – width of bounds rectangle

- *getBoundsX1*

public int getBoundsX1()

- **Description**

Returns x coordinate of the upper left corner of the bounds. Bounds must have been specified using (in 2.1.5, page 88), otherwise the return value is undefined.

- **Returns** – x coordinate of the upper left corner of the bounds

- *getBoundsX2*

public int getBoundsX2()

– **Description**

Returns x coordinate of the lower right corner of the bounds. Bounds must have been specified using (in 2.1.5, page 88), otherwise the return value is undefined.

– **Returns** – x coordinate of the lower right corner of the bounds

• *getBoundsY1*

`public int getBoundsY1()`

– **Description**

Returns y coordinate of the upper left corner of the bounds. Bounds must have been specified using (in 2.1.5, page 88), otherwise the return value is undefined.

– **Returns** – y coordinate of the upper left corner of the bounds

• *getBoundsY2*

`public int getBoundsY2()`

– **Description**

Returns y coordinate of the lower right corner of the bounds. Bounds must have been specified using (in 2.1.5, page 88), otherwise the return value is undefined.

– **Returns** – y coordinate of the lower right corner of the bounds

• *getComment*

`public java.lang.String getComment(int index)`

– **Description**

Returns a comment from the internal list of comments.

– **Parameters**

* **index** – the index of the comment to be returned, must be from 0 to (in 2.1.5, page 86)- 1; if this is not the case, null will be returned

– **See also**

* `ImageCodec.getNumComments()` (in 2.1.5, page 86)

* `ImageCodec.appendComment(java.lang.String)` (in 2.1.5, page 82)

* `ImageCodec.removeAllComments()` (in 2.1.5, page 88)

• *getDataInput*

`public java.io.DataInput getDataInput()`

– **Description**

Returns a `object` if one was provided via (in 2.1.5, page 88) or null otherwise.

– **Returns** – the `DataInput` object

• *getDataOutput*

`public java.io.DataOutput getDataOutput()`

– **Description**

Returns a `object` if one was provided via (in 2.1.5, page 89) or null otherwise.

– **Returns** – the `DataInput` object

• *getDpiX*

`public int getDpiX()`

– **Description**

Returns the horizontal physical resolution of the image associated with this codec. This resolution value was either retrieved from an image file or set via (in 2.1.5, page 89).

– **Returns** – horizontal physical resolution in dpi

– **See also**

* ImageCodec.getDpiY() (in 2.1.5, page 85)

• *getDpiY*

public int **getDpiY**()

– **Description**

Returns the vertical physical resolution of the image associated with this codec. This resolution value was either retrieved from an image file or set via (in 2.1.5, page 89).

– **Returns** – horizontal physical resolution in dpi

– **See also**

* ImageCodec.getDpiX() (in 2.1.5, page 84)

• *getFileExtensions*

public java.lang.String[] **getFileExtensions**()

– **Description**

Returns all file extensions that are typical for this file format. The default implementation in ImageCodec returns null. The file extension strings should include a leading dot and are supposed to be lower case (if that is allowed for the given file format). Example: {".jpg", ".jpeg"} for the JPEG file format.

– **Returns** – String array with typical file extensions

• *getFormatName*

public abstract java.lang.String **getFormatName**()

– **Description**

Returns the name of the file format supported by this codec. All classes extending (in 2.1.5, page 79) must override this method. When overriding, leave out any words in a particular language so that this format name can be understood by everyone. Usually it is enough to return the format creator plus a typical abbreviation, e.g. Microsoft BMP or Portable Anymap (PNM).

– **Returns** – name of the file format supported by this codec

• *getImage*

public net.sourceforge.jiu.data.PixelImage **getImage**()

– **Description**

Returns the image object stored in this codec. This is either an image given to this object via (in 2.1.5, page 90) or it was created by the codec itself during a loading operation.

– **Returns** – PixelImage object stored in this codec

• *getImageIndex*

public int **getImageIndex**()

– **Description**

Returns the zero-based index of the image to be loaded. Default is zero.

- **Returns** – zero-based image index value
-

- *getInputAsDataInput*

```
public java.io.DataInput getInputAsDataInput( )
```

- **Description**

Returns a `Object` if one was specified using `setInput` (in 2.1.5, page 88), or creates a `RandomAccessFile` if an `Image` was specified, or returns a `RandomAccessFile` if one was specified (`RandomAccessFile` implements `DataInput`). If neither of those has been given to this object, `null` is returned.

- **Returns** – `DataInput` object or `null`
-

- *getInputStream*

```
public java.io.InputStream getInputStream( )
```

- **Description**

Returns an `Object` that was given to this codec via `setInput` (in 2.1.5, page 90)(or `null` otherwise).

- **Returns** – `InputStream` object
-

- *getMimeTypes*

```
public abstract java.lang.String[] getMimeTypes( )
```

- **Description**

Return the **MIME** (at <http://www.faqs.org/rfcs/rfc2045.html>) (Multipurpose Internet Mail Extensions) type strings for this format, or `null` if none are available.

- **Returns** – MIME type strings or `null`
-

- *getMode*

```
public CodecMode getMode( )
```

- **Description**

Returns the mode this codec is in. Can be `null`, so that the codec will have to find out itself what to do.

- **Returns** – codec mode (load or save)
-

- *getNumComments*

```
public int getNumComments( )
```

- **Description**

Returns the current number of comments in the internal comment list.

- **Returns** – number of comments in the internal comment list
-

- *getOutputAsDataOutput*

```
public java.io.DataOutput getOutputAsDataOutput( )
```

- **Description**

Attempts to return an output object as a `Object`.

- **Returns** – a `DataOutput` object or `null` if that was not possible
-

- *getOutputStream*

```
public java.io.OutputStream getOutputStream( )
```

- **Description**

Returns an object that was given to this codec via (in 2.1.5, page 90)(or null otherwise).

- **Returns** – OutputStream object

- *getRandomAccessFile*

```
public java.io.RandomAccessFile getRandomAccessFile( )
```

- **Description**

Returns a object that was given to this codec via (in 2.1.5, page 90)(or null otherwise).

- **Returns** – RandomAccessFile object

- *hasBounds*

```
public boolean hasBounds( )
```

- **Description**

Returns if bounds have been specified.

- **Returns** – if bounds have been specified

- **See also**

* ImageCodec.removeBounds() (in 2.1.5, page 88)

* ImageCodec.setBounds(int,int,int,int) (in 2.1.5, page 88)

- *initModeFromIOObjects*

```
protected void initModeFromIOObjects( ) throws  
net.sourceforge.jiu.ops.MissingParameterException
```

- *isLoadingSupported*

```
public abstract boolean isLoadingSupported( )
```

- **Description**

Returns if this codec is able to load images in the file format supported by this codec. If true is returned this does not necessarily mean that all files in this format can be read, but at least some.

- **Returns** – if loading is supported

- *isRowRequired*

```
public boolean isRowRequired( int row )
```

- **Description**

Returns if an image row given by its number (zero-based) must be loaded in the context of the current bounds.

Example: if vertical bounds have been set to 34 and 37, image rows 34 to 37 as arguments to this method would result in true, anything else (e.g. 12 or 45) would result in false.

- **Parameters**

* row – the number of the row to be checked

- **Returns** – if row must be loaded, regarding the current bounds

- *isSavingSupported*

```
public abstract boolean isSavingSupported( )
```

– **Description**

Returns if this codec is able to save images in the file format supported by this codec. If `true` is returned this does not necessarily mean that all types files in this format can be written, but at least some.

– **Returns** – if saving is supported

• *isTileRequired*

`public boolean isTileRequired(int x1, int y1, int x2, int y2)`

– **Description**

Returns if the tile formed by the argument coordinates form a rectangle that overlaps with the bounds. If no bounds were defined, returns `true`.

– **Parameters**

* `x1` –
 * `y1` –
 * `x2` –
 * `y2` –

– **Returns** – if the argument tile is required

• *removeAllComments*

`public void removeAllComments()`

– **Description**

Removes all entries from the internal list of comments.

• *removeBounds*

`public void removeBounds()`

– **Description**

If bounds were set using (in 2.1.5, page 88), these bounds are no longer regarded after the call to this method.

• *setBounds*

`public void setBounds(int x1, int y1, int x2, int y2)`

– **Description**

Sets the bounds of a rectangular part of the image that is to be loaded or saved, instead of the complete image.

• *setBoundsIfNecessary*

`public void setBoundsIfNecessary(int width, int height)`

– **Description**

If no bounds have been set ((in 2.1.5, page 87) returns `false`), this method will set the bounds to 0, 0, `width - 1`, `height - 1`. By calling this method somewhere in the codec, no distinction has to be made for the two cases bounds have been defined and bounds have not been defined.

– **Parameters**

* `width` – width of the image to be loaded or saved
 * `height` – height of the image to be loaded or saved

• *setDataInput*

`public void setDataInput(java.io.DataInput dataInput)`

– **Description**

Specifies a `DataInput` object to be used for loading.

– **Parameters**

* `dataInput` – `DataInput` object to be used for loading an image

• *setDataOutput*

`public void setDataOutput(java.io.DataOutput dataOutput)`

– **Description**

Sets a `Object` to be used for saving an image.

– **Parameters**

* `dataOutput` – the object to be used for output

• *setDpi*

`public void setDpi(int horizontalDpi, int verticalDpi)`

– **Description**

Sets the DPI values to be stored in the file to the argument values.

– **Parameters**

* `horizontalDpi` – horizontal physical resolution in DPI (dots per inch)

* `verticalDpi` – vertical physical resolution in DPI (dots per inch)

– **See also**

* `ImageCodec.getDpiX()` (in 2.1.5, page 84)

* `ImageCodec.getDpiY()` (in 2.1.5, page 85)

• *setFile*

`public void setFile(java.io.File file, CodecMode codecMode) throws
java.io.IOException,
net.sourceforge.jiu.codecs.UnsupportedCodecModeException`

– **Description**

Gives a `File` object and a codec mode to this codec and attempts to initialize the appropriate I/O objects. Simply calls `setFile(String, CodecMode)` (in 2.1.5, page 89) with the absolute path of the `File` object.

– **Parameters**

* `file` – `File` object for the file to be used

* `codecMode` – defines whether an image is to be loaded from or saved to the file

• *setFile*

`public void setFile(java.lang.String fileName, CodecMode codecMode)
throws java.io.IOException,
net.sourceforge.jiu.codecs.UnsupportedCodecModeException`

– **Description**

Gives a file name and codec mode to the codec which will then try to create the corresponding I/O object. The default implementation in `ImageCodec` creates a `DataInputStream` object wrapped around a `BufferedInputStream` wrapped around a `FileInputStream` for `CodecMode.LOAD`. Similar for `CodecMode.SAVE`: a `DataOutputStream` around a `BufferedOutputStream` object around a `FileOutputStream` object. Codecs that need different I/O objects must override this method (some codecs may need random access and thus require a `RandomAccessFile` object).

– **Parameters**

- * `fileName` – name of the file to be used for loading or saving
 - * `codecMode` – defines whether file is to be used for loading or saving
-

• *setImage*

`public void setImage(net.sourceforge.jiu.data.PixelImage img)`

– **Description**

Give an image to this codec to be used for loading an image into it or saving the image.

– **Parameters**

- * `img` – image object to save or to load data into
-

• *setImageIndex*

`public void setImageIndex(int index)`

– **Description**

Sets the index of the image to be loaded to the argument value (which must be zero or larger).

– **Parameters**

- * `index` – int index value (zero-based) of the image to be loaded

– **Throws**

- * `java.lang.IllegalArgumentException` – if the argument is negative
-

• *setInputStream*

`public void setInputStream(java.io.InputStream inputStream)`

– **Description**

An can be given to this codec using this method.

– **Parameters**

- * `inputStream` – InputStream object to read from
-

• *setOutputStream*

`public void setOutputStream(java.io.OutputStream outputStream)`

– **Description**

A method to give an to this codec to be used for saving an image.

– **Parameters**

- * `outputStream` – the output stream to be used by this codec
-

• *setRandomAccessFile*

`public void setRandomAccessFile(java.io.RandomAccessFile randomAccessFile, CodecMode codecMode)`

– **Description**

A method to give a to this codec to be used for loading or saving an image. It is not possible to determine from a RandomAccessFile object whether it was opened in read-only or read-and-write mode. To let the codec know whether the object is to be used for loading or saving the second argument is of type CodecMode.

– **Parameters**

- * `randomAccessFile` – the file to be used for loading or saving
- * `codecMode` – tells the codec whether the file is to be used for loading or saving

- *suggestFileExtension*

```
public java.lang.String suggestFileExtension(  
net.sourceforge.jiu.data.PixelImage image )
```

- **Description**

Attempts to suggest a filename extension. The type of the argument image will be taken into consideration, although this will be necessary for some file formats only (as an example, PNM has different extensions for different image types, see (in 2.1.10, page 115)). This default implementation always returns `null`.

- **Parameters**

- * `image` – the image that is to be written to a file

- **Returns** – the file extension, including a leading dot, or `null` if no file extension can be recommended

2.1.6 Class ImageLoader

A convenience class with static methods to load images from files using JIU codecs. The load methods of this class try to load an image with all codecs registered with this class. This includes almost every codec that resides in the `net.sourceforge.jiu.codecs` package. You can register additional codecs with (in 2.1.6, page 94) or remove the usage of codecs with (in 2.1.6, page 95). A Codec that cannot safely identify a file to be in the format that it supports must not be used with ImageLoader. The failure to identify typically comes from the lack of magic byte sequences defined for the format. In order to load such a file, use the codec manually. Example: (in 2.1.7, page 96).

In order to load an image via (JPEG, PNG or GIF), use (in 17.1.9, page 452). It combines the loading features of `java.awt.Toolkit` and JIU's ImageLoader.

Usage example

```

PixelImage image = null;
try
{
    image = ImageLoader.load("image.tif");
}
catch (Exception e)
{
    // handle exception
}

```

Declaration

```

public class ImageLoader
extends java.lang.Object

```

Method summary

- createCodec(int)** Creates an instance of one of the codec classes known to ImageLoader.
- createFilenameFilter()** Returns a filename filter () that accepts files with name extensions typical for the image file formats known to ImageLoader.
- getNumCodecs()** Returns the number of codec classes currently known to ImageLoader.
- load(File)** Attempts to load an image from a file.
- load(File, Vector)** Attempts to load an image from a file, notifying the argument progress listeners.
- load(String)** Load an image from a file given by its name.
- load(String, Vector)** Attempts to load an image from the file with the given name, using the given list of progress listeners.
- registerCodecClass(ImageCodec)** Registers a codec class with ImageLoader.
- removeAllCodecClasses()** Removes all codec classes from the internal list of codec classes.
- removeCodecClass(ImageCodec)** Removes a codec class from the internal list of codec classes.

Methods

- *createCodec*

```
public static ImageCodec createCodec( int index )
```

- **Description**

Creates an instance of one of the codec classes known to ImageLoader.

- **Parameters**

* **index** – 0-based index of codec number, maximum value is (in 2.1.6, page 93)– 1

- **Returns** – new codec object or null if no object could be instantiated

- *createFilenameFilter*

```
public static java.io.FilenameFilter createFilenameFilter( )
```

- **Description**

Returns a filename filter () that accepts files with name extensions typical for the image file formats known to ImageLoader. The filter could then be used in an file dialog like .

Note that this filter does not include file formats supported by the AWT (GIF and JPEG, also PNG since Java 1.3).

- **Returns** – filter for image file names

- *getNumCodecs*

```
public static int getNumCodecs( )
```

- **Description**

Returns the number of codec classes currently known to ImageLoader. This number can be changed by registering additional codecs ((in 2.1.6, page 94)) or by removing codec classes ((in 2.1.6, page 95)).

- **Returns** – number of known codec classes

- *load*

```
public static net.sourceforge.jiu.data.PixelImage load( java.io.File file )
throws java.io.IOException,
net.sourceforge.jiu.codecs.InvalidFileStructureException,
net.sourceforge.jiu.codecs.InvalidImageIndexException,
net.sourceforge.jiu.codecs.UnsupportedTypeException
```

- **Description**

Attempts to load an image from a file.

- **Parameters**

* **file** – the file from which an image is to be loaded

- **Returns** – the image on success or null on failure

- *load*

```
public static net.sourceforge.jiu.data.PixelImage load( java.io.File file,
java.util.Vector listeners ) throws java.io.IOException,
net.sourceforge.jiu.codecs.InvalidFileStructureException,
net.sourceforge.jiu.codecs.InvalidImageIndexException,
net.sourceforge.jiu.codecs.UnsupportedTypeException
```

– **Description**

Attempts to load an image from a file, notifying the argument progress listeners.

– **Parameters**

- * **file** – the file to load an image from
- * **listeners** – a Vector of ProgressListener objects to be notified

– **Returns** – an instance of a class implementing (in 14.1.11, page 328)

• *load*

```
public static net.sourceforge.jiu.data.PixelImage load( java.lang.String
fileName ) throws java.io.IOException,
net.sourceforge.jiu.codecs.InvalidFileStructureException,
net.sourceforge.jiu.codecs.InvalidImageIndexException,
net.sourceforge.jiu.codecs.UnsupportedTypeException
```

– **Description**

Load an image from a file given by its name. Simply calls load(fileName, null).

– **Parameters**

- * **fileName** – name of the file from which an image is to be loaded

– **Returns** – the loaded image on success, null on failure

• *load*

```
public static net.sourceforge.jiu.data.PixelImage load( java.lang.String
fileName, java.util.Vector listeners ) throws java.io.IOException,
net.sourceforge.jiu.codecs.InvalidFileStructureException,
net.sourceforge.jiu.codecs.InvalidImageIndexException,
net.sourceforge.jiu.codecs.UnsupportedTypeException
```

– **Description**

Attempts to load an image from the file with the given name, using the given list of progress listeners.

– **Parameters**

- * **fileName** – name of the file from which an image is to be loaded
- * **listeners** – a list of objects implementing ProgressListener

– **Returns** – the loaded image

• *registerCodecClass*

```
public static void registerCodecClass( ImageCodec codec )
```

– **Description**

Registers a codec class with ImageLoader. The argument is an instance of the class to be registered. Note that the codec class must have an empty constructor.

Example: let's say you have written a new codec called ACMEImageCodec. Your codec supports loading images. Then you could register it like that:

```
ImageLoader.registerCodecClass(new ACMEImageCodec());
```

– **Parameters**

- * **codec** – an instance of the codec class to be registered
-

• *removeAllCodecClasses*

```
public static void removeAllCodecClasses( )
```

- **Description**

Removes all codec classes from the internal list of codec classes. After a call to this method, ImageLoader will not load anything unless new codecs get registered.

- *removeCodecClass*

```
public static void removeCodecClass( ImageCodec codec )
```

- **Description**

Removes a codec class from the internal list of codec classes.

- **Parameters**

- * `codec` – an instance of the codec class to be removed

2.1.7 Class PalmCodec

A codec to read and write image files in the native image file format of **Palm OS** (at <http://www.palmos.com/>), an operating system for handheld devices.

Supported file types when loading

This codec reads uncompressed, scan line compressed and RLE compressed Palm files with bit depths of 1, 2, 4, 8 and 16 bits per pixel. Not supported are the Packbits compression algorithm or any color depths other than the aforementioned.

Supported image types when saving

Compression types Uncompressed, Scan line and RLE are written. When saving an image as a Palm, the image data classes will be mapped to file types as follows:

- (in 14.1.1, page 311): will be saved as a 1 bit per pixel, monochrome file.
- (in 14.1.4, page 318): will be saved as an 8 bits per pixel file with a custom palette which will contain the 256 shades of gray from black - (0, 0, 0) - to white - (255, 255, 255).
- (in 14.1.8, page 325): it is first checked if the image is using the Palm system 8 bits per pixel palette. If so, an 8 bits per pixel file with no custom palette is written, otherwise an 8 bits per pixel file with a custom palette (of the original length) is written.
- (in 14.1.12, page 331): will be saved as a 16 bits per pixel, direct color file. Some information will get lost when converting from 24 to 16 bits per pixel. Instead of 256 shades for each red, green and blue (and thus, $256^3 = 16,777,216$ possible colors) the resulting file will only use 32 shades of red and blue and 64 shades of green (65,536 possible colors).

I/O objects

This codec supports all the I/O classes that are considered in ImageCodec. If you save images and want a correct compressed size field in the resulting Palm file, make sure to give a RandomAccessFile object to the codec. Or simply use (in 2.1.7, page 103) which does that automatically.

File extension

This codec suggests .palm as file extension for this file format. This is by no means official, but I find it helpful.

Transparency information

The transparency index in a Palm file is saved and loaded, but a loaded index is not stored in the image object as there is no support for transparency information of any kind in PixelImage yet. The RGB transparency color that is present in a file only in direct color mode is read but not written.

Bounds

The bounds concept of ImageCodec is supported so that you can load or save only part of an image.

Open questions on the Palm file format

- How does Packbits compression work? Where can I get sample files or a Windows converter that writes those?
- How is FLAG_4_BYTE_FIELDS interpreted? When are four byte fields used?
- When loading a 4 bpp Palm image file without a custom palette, how is the decoder supposed to know whether to take the predefined color or grayscale palette with 16 entries?

If you can answer any of these, please **contact me** (at <http://www.geocities.com/marcoschmidt.geo/contact.html>)!

Known problems

- Unfortunately, the Palm image file format does not include a signature that makes it easy to identify such a file. Various checks on allowed combinations of color depth, compression type etc. will prevent the codec from trying to interpret all files as Palm image files, but there is still a probability of false identification.

Usage examples

Load an image from a Palm image file:

```
PalmCodec codec = new PalmCodec();
codec.setFile("test.palm", CodecMode.LOAD);
codec.process();
PixelImage image = codec.getImage();
codec.close();
```

Save an image to a Palm file using RLE compression:

```
PalmCodec codec = new PalmCodec();
codec.setImage(image);
codec.setCompression(PalmCodec.COMPRESSION_RLE);
codec.setFile("out.palm", CodecMode.SAVE);
codec.process();
codec.close();
```

Background

The code is based on:

- the specification **Palm Native Image Format** (at <http://www.kawt.de/doc/palmimage.html>),
- the source code of the utilities `pnmtopalm` and `palmtopnm` that are part of the **Netpbm** (at <http://netpbm.sourceforge.net>) package,
- **Palm OS Compressed Bitmaps** (at <http://oasis.palm.com/dev/kb/papers/1831.cfm>) by Ken Krugler, a Palm Developer Knowledge Base article on the scan line compression algorithm and
- **Palm OS Bitmaps** (at <http://oasis.palm.com/dev/kb/manuals/sdk/Bitmap.cfm>), also part of the Palm Developer Knowledge Base, contains general information on the structure of Palm images.

I also received helpful feedback and test images from Bill Janssen.

Declaration

```
public class PalmCodec
extends net.sourceforge.jiu.codecs.ImageCodec (in 2.1.5, page 79)
```

Field summary

COMPRESSION_NONE Constant for compression type Uncompressed.
COMPRESSION_PACKBITS Constant for compression type Packbits.
COMPRESSION_RLE Constant for compression type RLE (run length encoding).
COMPRESSION_SCANLINE Constant for compression type Scanline.

Constructor summary

PalmCodec()

Method summary

createSystem2BitGrayscalePalette() Creates the 2 bits per pixel Palm system palette with grayscale values.
createSystem4BitColorPalette() Creates the 4 bits per pixel Palm system palette with color values.
createSystem4BitGrayscalePalette() Creates the 4 bits per pixel Palm system palette with grayscale values.
createSystem8BitPalette() Creates the 8 bits per pixel Palm system palette.
getCompression() Returns the Palm compression method.
getFormatName()
getMimeTypes()
getTransparencyIndex() Returns the transparency index if one is available (in 2.1.7, page 100) returns **true**) or an undefined value otherwise.
hasTransparencyIndex() Returns whether a transparency index is available and can be retrieved via (in 2.1.7, page 100).
isLoadingSupported()
isPalmSystemPalette256(Palette) Returns if the argument palette is the Palm system palette with 256 colors.
isPalmSystemPaletteColor16(Palette) Returns if the argument palette is the Palm system color palette with 16 entries.
isPalmSystemPaletteGray16(Palette) Returns if the argument palette is the Palm system grayscale palette with 16 entries.
isPalmSystemPaletteGray4(Palette) Returns if the argument palette is the Palm system grayscale palette with 4 entries.
isSavingSupported()
process()
removeTransparencyIndex() Removes the transparency index if one has been set.
setCompression(int) Sets the compression algorithm to be used for saving an image.
setFile(String, CodecMode) Reuses super.setFile when used for CodecMode.LOAD, but creates a RandomAccessFile instead of a FileOutputStream in write mode so that the compressed size can be written correctly (requires a seek operation).
setTransparencyIndex(int) Sets a new transparency index when saving an image.
suggestFileExtension(PixelImage)

Fields

- public static final int **COMPRESSION_NONE**
 - Constant for compression type Uncompressed.
- public static final int **COMPRESSION_PACKBITS**
 - Constant for compression type Packbits.
- public static final int **COMPRESSION_RLE**
 - Constant for compression type RLE (run length encoding).
- public static final int **COMPRESSION_SCANLINE**
 - Constant for compression type Scanline.

Constructors

- *PalmCodec*
public **PalmCodec**()

Methods

- *createSystem2BitGrayscalePalette*
public static net.sourceforge.jiu.data.Palette **createSystem2BitGrayscalePalette**()
 - **Description**
Creates the 2 bits per pixel Palm system palette with grayscale values. This palette is used when no custom palette is defined in a 2 bpp image.
 - **Returns** – Palm’s default palette for 2 bits per pixel (grayscale), with 4 entries
- *createSystem4BitColorPalette*
public static net.sourceforge.jiu.data.Palette **createSystem4BitColorPalette**()
 - **Description**
Creates the 4 bits per pixel Palm system palette with color values. This palette (or the 4 bpp grayscale palette) is used when no custom palette is defined in a 4 bpp image.
 - **Returns** – Palm’s default palette for 4 bits per pixel (color), with 16 entries
- *createSystem4BitGrayscalePalette*
public static net.sourceforge.jiu.data.Palette **createSystem4BitGrayscalePalette**()
 - **Description**
Creates the 4 bits per pixel Palm system palette with grayscale values. This palette (or the 4 bpp color palette) is used when no custom palette is defined in a 4 bpp image.
 - **Returns** – Palm’s default palette for 4 bits per pixel (grayscale), with 16 entries

- *createSystem8BitPalette*

```
public static net.sourceforge.jiu.data.Palette createSystem8BitPalette( )
```

- **Description**

Creates the 8 bits per pixel Palm system palette. This palette is used when no custom palette is defined in an 8 bpp image.

- **Returns** – Palm’s default palette for 8 bits per pixel, with 256 entries

- *getCompression*

```
public int getCompression( )
```

- **Description**

Returns the Palm compression method. This should be one of the COMPRESSION_xyz constants of this class.

- **Returns** – integer value with the compression method (found in a file when loading or to be used for saving)

- **See also**

* PalmCodec.setCompression(int) (in 2.1.7, page 103)

- *getFormatName*

```
public abstract java.lang.String getFormatName( )
```

- **Description copied from ImageCodec (in 2.1.5, page 79)**

Returns the name of the file format supported by this codec. All classes extending (in 2.1.5, page 79) must override this method. When overriding, leave out any words in a particular language so that this format name can be understood by everyone. Usually it is enough to return the format creator plus a typical abbreviation, e.g. Microsoft BMP or Portable Anymap (PNM).

- **Returns** – name of the file format supported by this codec

- *getMimeType*

```
public abstract java.lang.String[] getMimeType( )
```

- **Description copied from ImageCodec (in 2.1.5, page 79)**

Return the **MIME** (at <http://www.faqs.org/rfcs/rfc2045.html>) (Multipurpose Internet Mail Extensions) type strings for this format, or null if none are available.

- **Returns** – MIME type strings or null

- *getTransparencyIndex*

```
public int getTransparencyIndex( )
```

- **Description**

Returns the transparency index if one is available ((in 2.1.7, page 100) returns true) or an undefined value otherwise.

- **See also**

* PalmCodec.hasTransparencyIndex() (in 2.1.7, page 100)

* PalmCodec.removeTransparencyIndex() (in 2.1.7, page 102)

* PalmCodec.setTransparencyIndex(int) (in 2.1.7, page 103)

- *hasTransparencyIndex*

```
public boolean hasTransparencyIndex( )
```

– **Description**

Returns whether a transparency index is available and can be retrieved via (in 2.1.7, page 100).

– **Returns** – transparency index, a positive value that is a valid index into the palette

– **See also**

- * `PalmCodec.getTransparencyIndex()` (in 2.1.7, page 100)
 - * `PalmCodec.removeTransparencyIndex()` (in 2.1.7, page 102)
 - * `PalmCodec.setTransparencyIndex(int)` (in 2.1.7, page 103)
-

• *isLoadingSupported*

```
public abstract boolean isLoadingSupported( )
```

– **Description copied from ImageCodec (in 2.1.5, page 79)**

Returns if this codec is able to load images in the file format supported by this codec. If `true` is returned this does not necessarily mean that all files in this format can be read, but at least some.

– **Returns** – if loading is supported

• *isPalmSystemPalette256*

```
public static boolean isPalmSystemPalette256(
net.sourceforge.jiu.data.Palette palette )
```

– **Description**

Returns if the argument palette is the Palm system palette with 256 colors.

– **Parameters**

- * `palette` – to be checked

– **Returns** – if the argument is an 8 bits per pixel Palm system palette

– **See also**

- * `PalmCodec.createSystem8BitPalette()` (in 2.1.7, page 100)
-

• *isPalmSystemPaletteColor16*

```
public static boolean isPalmSystemPaletteColor16(
net.sourceforge.jiu.data.Palette palette )
```

– **Description**

Returns if the argument palette is the Palm system color palette with 16 entries.

– **Parameters**

- * `palette` – to be checked

– **See also**

- * `PalmCodec.createSystem4BitColorPalette()` (in 2.1.7, page 99)
-

• *isPalmSystemPaletteGray16*

```
public static boolean isPalmSystemPaletteGray16(
net.sourceforge.jiu.data.Palette palette )
```

– **Description**

Returns if the argument palette is the Palm system grayscale palette with 16 entries.

– **Parameters**

- * `palette` – to be checked

– **See also**

* `PalmCodec.createSystem4BitGrayscalePalette()` (in 2.1.7, page 99)

• *isPalmSystemPaletteGray4*

```
public static boolean isPalmSystemPaletteGray4(
    net.sourceforge.jiu.data.Palette palette )
```

– **Description**

Returns if the argument palette is the Palm system grayscale palette with 4 entries.

– **Parameters**

* `palette` – to be checked

– **See also**

* `PalmCodec.createSystem2BitGrayscalePalette()` (in 2.1.7, page 99)

• *isSavingSupported*

```
public abstract boolean isSavingSupported( )
```

– **Description copied from ImageCodec (in 2.1.5, page 79)**

Returns if this codec is able to save images in the file format supported by this codec. If `true` is returned this does not necessarily mean that all types files in this format can be written, but at least some.

– **Returns** – if saving is supported

• *process*

```
public void process( ) throws
    net.sourceforge.jiu.ops.MissingParameterException,
    net.sourceforge.jiu.ops.OperationFailedException,
    net.sourceforge.jiu.ops.WrongParameterException
```

– **Description copied from net.sourceforge.jiu.ops.Operation (in 19.2.5, page 508)**

This method does the actual work of the operation. It must be called after all parameters have been given to the operation object.

– **Throws**

- * `net.sourceforge.jiu.ops.WrongParameterException` – if at least one of the input parameters was not initialized appropriately (values out of the valid interval, etc.)
 - * `net.sourceforge.jiu.ops.MissingParameterException` – if any mandatory parameter was not given to the operation
 - * `net.sourceforge.jiu.ops.OperationFailedException` –
-

• *removeTransparencyIndex*

```
public void removeTransparencyIndex( )
```

– **Description**

Removes the transparency index if one has been set.

– **See also**

- * `PalmCodec.getTransparencyIndex()` (in 2.1.7, page 100)
 - * `PalmCodec.hasTransparencyIndex()` (in 2.1.7, page 100)
 - * `PalmCodec.setTransparencyIndex(int)` (in 2.1.7, page 103)
-

- *setCompression*

```
public void setCompression( int newCompressionType )
```

- **Description**

Sets the compression algorithm to be used for saving an image.

- **Parameters**

- * **newCompressionType** – int value that is one of the COMPRESSION_xyz constants of this class

- **Throws**

- * `java.lang.IllegalArgumentException` – if the compression type is unsupported

- **See also**

- * `PalmCodec.getCompression()` (in 2.1.7, page 100)

- *setFile*

```
public void setFile( java.lang.String fileName, CodecMode codecMode )
```

```
throws java.io.IOException,
```

```
net.sourceforge.jiu.codecs.UnsupportedCodecModeException
```

- **Description**

Reuses `super.setFile` when used for `CodecMode.LOAD`, but creates a `RandomAccessFile` instead of a `FileOutputStream` in write mode so that the compressed size can be written correctly (requires a seek operation).

- **Parameters**

- * **fileName** – name of the file to be opened

- * **codecMode** – defines whether this codec object is to be used for loading or saving

- *setTransparencyIndex*

```
public void setTransparencyIndex( int newIndex )
```

- **Description**

Sets a new transparency index when saving an image. If this method is called, the argument value is used as an index into the palette for a color that is supposed to be transparent. When the resulting Palm image file is drawn onto some background, all pixels in the color pointed to by the transparency index are not supposed to be overdrawn so that the background is visible at those places.

- **Parameters**

- * **newIndex** – the new transparency index, must be smaller than the number of entries in the palette

- **See also**

- * `PalmCodec.getTransparencyIndex()` (in 2.1.7, page 100)

- * `PalmCodec.hasTransparencyIndex()` (in 2.1.7, page 100)

- * `PalmCodec.removeTransparencyIndex()` (in 2.1.7, page 102)

- *suggestFileExtension*

```
public java.lang.String suggestFileExtension(
```

```
net.sourceforge.jiu.data.PixelImage image )
```

- **Description copied from ImageCodec (in 2.1.5, page 79)**

Attempts to suggest a filename extension. The type of the argument image will be taken into consideration, although this will be necessary for some file formats only (as an example, PNM has different extensions for different image types, see (in 2.1.10, page 115)). This default implementation always returns `null`.

– **Parameters**

* **image** – the image that is to be written to a file

– **Returns** – the file extension, including a leading dot, or `null` if no file extension can be recommended

2.1.8 Class PCDCodec

A codec to read Kodak Photo-CD (image pac) image files. Typical file extension is .pcd. PCD is designed to store the same image in several resolutions. Not all resolutions are always present in a file. Typically, the first five resolutions are available and the file size is between four and six megabytes. Lossless compression (Huffman encoding) is used to store the higher resolution images. All images are in 24 bit YCbCr colorspace, with a component subsampling of 4:1:1 (Y:Cb:Cr) in both horizontal and vertical direction.

Limitations

Only the lowest three resolutions are supported by this codec.

Sample PCD files

You can download sample PCD image files from **Kodak's website** (at <http://www.kodak.com/digitalImaging/samples/imageIntro.shtml>).

Declaration

```
public class PCDCodec
extends net.sourceforge.jiu.codecs.ImageCodec (in 2.1.5, page 79)
implements net.sourceforge.jiu.color.YCbCrIndex
```

Field summary

PCD_RESOLUTION_1 Base/16, the minimum pixel resolution, 192 x 128 pixels.
PCD_RESOLUTION_2 Base/4, the second pixel resolution, 384 x 256 pixels.
PCD_RESOLUTION_3 Base, the third pixel resolution, 768 x 512 pixels.
PCD_RESOLUTION_4 Base*4, the fourth pixel resolution, 1536 x 1024 pixels.
PCD_RESOLUTION_5 Base*16, the fifth pixel resolution, 3072 x 2048 pixels.
PCD_RESOLUTION_6 Base*64, the sixth pixel resolution, 6144 x 4096 pixels.
PCD_RESOLUTION_DEFAULT Index for the default resolution , Base (in 2.1.8, page 106)).
PCD_RESOLUTIONS This two-dimensional int array holds all possible pixel resolutions for a PCD file.

Constructor summary

PCDCodec() This constructor chooses the default settings for PCD image loading:

- load color image (all channels, not only luminance)
- perform color conversion from PCD's native YCbCr color space to RGB
- load the image in the default resolution (in 2.1.8, page 106), 768 x 512 pixels (or vice versa)

Method summary

```
getFileExtensions()
getFormatName()
getMimeTypes()
isLoadingSupported()
```

isSavingSupported()

process() Checks the parameter and loads an image.

setColorConversion(boolean) Specify whether color is converted from PCD's YCbCr color space to RGB color space.

setFile(String, CodecMode)

setMonochrome(boolean) Specifies whether the image is to be loaded as gray or color image.

setResolutionIndex(int)**Fields**

- public static final int **PCD_RESOLUTION_1**
 - Base/16, the minimum pixel resolution, 192 x 128 pixels.
- public static final int **PCD_RESOLUTION_2**
 - Base/4, the second pixel resolution, 384 x 256 pixels.
- public static final int **PCD_RESOLUTION_3**
 - Base, the third pixel resolution, 768 x 512 pixels.
- public static final int **PCD_RESOLUTION_4**
 - Base*4, the fourth pixel resolution, 1536 x 1024 pixels. Unsupported
- public static final int **PCD_RESOLUTION_5**
 - Base*16, the fifth pixel resolution, 3072 x 2048 pixels. Unsupported
- public static final int **PCD_RESOLUTION_6**
 - Base*64, the sixth pixel resolution, 6144 x 4096 pixels. Unsupported
- public static final int **PCD_RESOLUTION_DEFAULT**
 - Index for the default resolution , Base (in 2.1.8, page 106)).
- public static final int **PCD_RESOLUTIONS**
 - This two-dimensional int array holds all possible pixel resolutions for a PCD file. Use one of the PCD resolution constants (e.g. (in 2.1.8, page 106)as first index. The second index must be 0 or 1 and leads to either width or height. Example:
PCD_RESOLUTION[PCD_RESOLUTION_3][1] will evaluate as 512, which can be width or height, depending on the image being in landscape or portrait mode. You may want to use these resolution values in your program to prompt the user which resolution to load from the file.

Constructors

- *PCDCodec*
public **PCDCodec()**
 - **Description**
This constructor chooses the default settings for PCD image loading:

- * load color image (all channels, not only luminance)
- * perform color conversion from PCD's native YCbCr color space to RGB
- * load the image in the default resolution (in 2.1.8, page 106), 768 x 512 pixels (or vice versa)

Methods

- *getFileExtensions*

`public java.lang.String[] getFileExtensions()`

- **Description copied from ImageCodec (in 2.1.5, page 79)**

Returns all file extensions that are typical for this file format. The default implementation in ImageCodec returns `null`. The file extension strings should include a leading dot and are supposed to be lower case (if that is allowed for the given file format). Example: `{" .jpg", " .jpeg"}` for the JPEG file format.

- **Returns** – String array with typical file extensions
-

- *getFormatName*

`public abstract java.lang.String getFormatName()`

- **Description copied from ImageCodec (in 2.1.5, page 79)**

Returns the name of the file format supported by this codec. All classes extending (in 2.1.5, page 79) must override this method. When overriding, leave out any words in a particular language so that this format name can be understood by everyone. Usually it is enough to return the format creator plus a typical abbreviation, e.g. `Microsoft BMP` or `Portable Anymap (PNM)`.

- **Returns** – name of the file format supported by this codec
-

- *getMimeTypes*

`public abstract java.lang.String[] getMimeTypes()`

- **Description copied from ImageCodec (in 2.1.5, page 79)**

Return the **MIME** (at <http://www.faqs.org/rfcs/rfc2045.html>) (Multipurpose Internet Mail Extensions) type strings for this format, or `null` if none are available.

- **Returns** – MIME type strings or null
-

- *isLoadingSupported*

`public abstract boolean isLoadingSupported()`

- **Description copied from ImageCodec (in 2.1.5, page 79)**

Returns if this codec is able to load images in the file format supported by this codec. If `true` is returned this does not necessarily mean that all files in this format can be read, but at least some.

- **Returns** – if loading is supported
-

- *isSavingSupported*

`public abstract boolean isSavingSupported()`

- **Description copied from ImageCodec (in 2.1.5, page 79)**

Returns if this codec is able to save images in the file format supported by this codec. If `true` is returned this does not necessarily mean that all types files in this format can be written, but at least some.

- **Returns** – if saving is supported
-

- *process*

```
public void process( ) throws
net.sourceforge.jiu.codecs.InvalidFileStructureException,
net.sourceforge.jiu.ops.MissingParameterException,
net.sourceforge.jiu.ops.OperationFailedException,
net.sourceforge.jiu.codecs.UnsupportedTypeException,
net.sourceforge.jiu.codecs.WrongFileFormatException
```

- **Description**

Checks the parameter and loads an image.

- *setColorConversion*

```
public void setColorConversion( boolean performColorConversion )
```

- **Description**

Specify whether color is converted from PCD's YCbCr color space to RGB color space. The default is `true`, and you should only change this if you really know what you are doing. If you simply want the luminance (gray) channel, use (in 2.1.8, page 108) with `true` as parameter.

- **Parameters**

- * `performColorConversion` – boolean that determines whether color conversion is applied

- *setFile*

```
public void setFile( java.lang.String fileName, CodecMode codecMode )
throws java.io.IOException,
net.sourceforge.jiu.codecs.UnsupportedCodecModeException
```

- **Description copied from ImageCodec (in 2.1.5, page 79)**

Gives a file name and codec mode to the codec which will then try to create the corresponding I/O object. The default implementation in ImageCodec creates a `DataInputStream` object wrapped around a `BufferedInputStream` wrapped around a `FileInputStream` for `CodecMode.LOAD`. Similar for `CodecMode.SAVE`: a `DataOutputStream` around a `BufferedOutputStream` object around a `FileOutputStream` object. Codecs that need different I/O objects must override this method (some codecs may need random access and thus require a `RandomAccessFile` object).

- **Parameters**

- * `fileName` – name of the file to be used for loading or saving
- * `codecMode` – defines whether file is to be used for loading or saving

- *setMonochrome*

```
public void setMonochrome( boolean monochrome )
```

- **Description**

Specifies whether the image is to be loaded as gray or color image. If argument is `true`, only the gray channel is loaded.

- *setResolutionIndex*

```
public void setResolutionIndex( int resolutionIndex )
```

2.1.9 Class PNGCodec

A codec for the Portable Network Graphics (PNG) format. Supports both loading and saving of images.

Usage examples

Load an image

The following example code loads an image from a PNG file. Note that you could also use (in 2.1.6, page 92) or (in 17.1.9, page 452) which require only a single line of code and can load all formats supported by JIU, including PNG.

```
PNGCodec codec = new PNGCodec();
codec.setFile("image.png", CodecMode.LOAD);
codec.process();
PixelImage image = codec.getImage();
```

Save an image

```
PNGCodec codec = new PNGCodec();
codec.setFile("out.png", CodecMode.SAVE);
codec.setImage(image);
codec.setCompressionLevel(Deflater.BEST_COMPRESSION);
codec.appendComment("Copyright (c) 1992 John Doe");
// sets last modification time to current time
codec.setModification(new GregorianCalendar(
    new SimpleTimeZone(0, "UTC")));
codec.process();
```

Supported storage order types

Loading

This codec reads both non-interlaced and Adam7 interlaced PNG files.

Saving

This codec only writes non-interlaced PNG files.

Supported color types

Loading

- Grayscale 1 bit streams are loaded as (in 14.1.1, page 311) objects, 2, 4 and 8 bit streams as (in 14.1.4, page 318) and 16 bit as (in 14.1.3, page 317) objects.
- Indexed 1, 2, 4 and 8 bit streams are all loaded as (in 14.1.8, page 325).
- RGB truecolor 24 bit streams are loaded as (in 14.1.12, page 331), 48 bit streams as (in 14.1.13, page 332) objects.

Saving

- (in 14.1.1, page 311)objects are stored as grayscale 1 bit PNG streams.
- (in 14.1.8, page 325)objects are stored as indexed 8 bit PNG streams. Images will always be stored as 8 bit files, even if the palette has only 16, 4 or 2 entries.
- (in 14.1.4, page 318)objects are stored as 8 bit grayscale PNG streams.
- (in 14.1.3, page 317)objects are stored as 16 bit grayscale PNG streams.
- (in 14.1.12, page 331)objects are stored as 24 bit RGB truecolor PNG streams.
- (in 14.1.13, page 332)objects are stored as 48 bit RGB truecolor PNG streams.

Transparency information

PNG allows to store different types of transparency information. Full alpha channels, transparent index values, and more. Right now, this JIU codec does not make use of this information and simply skips over it when encountered.

Bounds

This codec regards the bounds concept. If bounds are specified with (in 2.1.5, page 88), the codec will only load or save part of an image.

Metadata

Loading

- Physical resolution information is loaded from `pHYs` chunks. Use (in 2.1.5, page 84)and (in 2.1.5, page 85)to retrieve that information. after the call to (in 2.1.9, page 112).
- Textual comments are read from `tEXt` chunks and can be retrieved with (in 2.1.5, page 84)after the call to (in 2.1.9, page 112).

Saving

- Physical resolution information (specified with (in 2.1.5, page 89)) is stored in a `pHYs` chunk.
- Textual comments (specified with (in 2.1.5, page 82)) are stored as `tEXt` chunks. The keyword used is `Comment`. Each of the (in 2.1.5, page 86)is stored in a chunk of its own.
- Time of modification is stored in a `tIME` chunk. Use (in 2.1.9, page 113)to give a point in time to this codec.

Implementation details

This class relies heavily on the Java runtime library for decompression and checksum creation.

Background

To learn more about the PNG file format, visit its **official homepage** (at <http://www.libpng.org/pub/png/>). There you can find a detailed specification, test images and existing PNG libraries and PNG-aware applications. The book PNG - The Definitive Guide by Greg Roelofs, published by O'Reilly, 1999, ISBN 1-56592-542-4 is a valuable source of information on PNG. It is out of print, but it can be viewed online and downloaded for offline reading in its entirety from the site.

Declaration

```
public class PNGCodec
extends net.sourceforge.jiu.codecs.ImageCodec (in 2.1.5, page 79)
```

Constructor summary

PNGCodec()

Method summary

getFormatName()
getMimeTypes()
isLoadingSupported()
isSavingSupported()
main(String[])
process()
setCompressionLevel(int) Sets the compression level to be used with the underlying object which does the compression.
setCompressionStrategy(int) Sets the compression strategy to be used with the underlying object which does the compression.
setEncodingIdatSize(int) Sets the size of IDAT chunks generated when encoding.
setFile(String, CodecMode)
setModification(Calendar) Sets date and time of last modification of the image to be stored in a PNG stream when saving.
suggestFileExtension(PixelImage)

Constructors

- *PNGCodec*
 public **PNGCodec()**

Methods

- *getFormatName*
 public abstract java.lang.String **getFormatName()**
 - **Description copied from ImageCodec (in 2.1.5, page 79)**
 Returns the name of the file format supported by this codec. All classes extending (in 2.1.5, page 79) must override this method. When overriding, leave out any words in a particular language so that this format name can be understood by everyone. Usually it is enough to return the format creator plus a typical abbreviation, e.g. **Microsoft BMP** or **Portable Anymap (PNM)**.
 - **Returns** – name of the file format supported by this codec
- *getMimeTypes*
 public abstract java.lang.String[] **getMimeTypes()**
 - **Description copied from ImageCodec (in 2.1.5, page 79)**
 Return the **MIME** (at <http://www.faqs.org/rfcs/rfc2045.html>) (Multipurpose Internet Mail Extensions) type strings for this format, or null if none are available.

- **Returns** – MIME type strings or null
-

- *isLoadingSupported*

`public abstract boolean isLoadingSupported()`

- **Description copied from ImageCodec (in 2.1.5, page 79)**

Returns if this codec is able to load images in the file format supported by this codec. If `true` is returned this does not necessarily mean that all files in this format can be read, but at least some.

- **Returns** – if loading is supported
-

- *isSavingSupported*

`public abstract boolean isSavingSupported()`

- **Description copied from ImageCodec (in 2.1.5, page 79)**

Returns if this codec is able to save images in the file format supported by this codec. If `true` is returned this does not necessarily mean that all types files in this format can be written, but at least some.

- **Returns** – if saving is supported
-

- *main*

`public static void main(java.lang.String[] args) throws
java.lang.Exception`

- *process*

`public void process() throws
net.sourceforge.jiu.ops.MissingParameterException,
net.sourceforge.jiu.ops.OperationFailedException,
net.sourceforge.jiu.ops.WrongParameterException`

- **Description copied from net.sourceforge.jiu.ops.Operation (in 19.2.5, page 508)**

This method does the actual work of the operation. It must be called after all parameters have been given to the operation object.

- **Throws**

- * `net.sourceforge.jiu.ops.WrongParameterException` – if at least one of the input parameters was not initialized appropriately (values out of the valid interval, etc.)
 - * `net.sourceforge.jiu.ops.MissingParameterException` – if any mandatory parameter was not given to the operation
 - * `net.sourceforge.jiu.ops.OperationFailedException` –
-

- *setCompressionLevel*

`public void setCompressionLevel(int newLevel)`

- **Description**

Sets the compression level to be used with the underlying object which does the compression. If no value is specified, is used.

- **Parameters**

- * `newLevel` – compression level, from 0 to 9, 0 being fastest and compressing worst and 9 offering highest compression and taking the most time
-

- *setCompressionStrategy*

```
public void setCompressionStrategy( int newStrategy )
```

- **Description**

Sets the compression strategy to be used with the underlying object which does the compression. If no value is specified, is used.

- **Parameters**

* **newStrategy** – one of Deflater’s strategy values: , ,

- *setEncodingIdatSize*

```
public void setEncodingIdatSize( int newSize )
```

- **Description**

Sets the size of IDAT chunks generated when encoding. If this method is never called, a default value of 32768 bytes (32 KB) is used. Note that a byte array of the size of the value you specify here is allocated, so make sure that you keep the value small enough to stay within a system’s memory.

Compressed image data is spread over several IDAT chunks by this codec. The length of the compressed data of a complete image is known only after the complete image has been encoded. With PNG, that length value has to be stored before the compressed data as a chunk size value. This codec is supposed to work with objects, so seeking back to adjust the chunk size value of an IDAT chunk is not possible. That’s why all data of a chunk is compressed into a memory buffer. Whenever the buffer gets full, it is written to output as an IDAT chunk.

Note that the last IDAT chunk may be smaller than the size defined here.

- **Parameters**

* **newSize** – size of encoding compressed data buffer

- *setFile*

```
public void setFile( java.lang.String fileName, CodecMode codecMode )
throws java.io.IOException,
net.sourceforge.jiu.codecs.UnsupportedCodecModeException
```

- **Description copied from ImageCodec (in 2.1.5, page 79)**

Gives a file name and codec mode to the codec which will then try to create the corresponding I/O object. The default implementation in ImageCodec creates a DataInputStream object wrapped around a BufferedInputStream wrapped around a FileInputStream for CodecMode.LOAD. Similar for CodecMode.SAVE: a DataOutputStream around a BufferedOutputStream object around a FileOutputStream object. Codecs that need different I/O objects must override this method (some codecs may need random access and thus require a RandomAccessFile object).

- **Parameters**

* **fileName** – name of the file to be used for loading or saving
 * **codecMode** – defines whether file is to be used for loading or saving

- *setModification*

```
public void setModification( java.util.Calendar time )
```

- **Description**

Sets date and time of last modification of the image to be stored in a PNG stream when saving. Make sure the argument object has UTC as time zone (**as demanded by the PNG specs**) (at <http://www.w3.org/TR/PNG#C.tIME>). If you want the current time and date, use `new GregorianCalendar(new SimpleTimeZone(0, "UTC"))` as parameter for this method.

– **Parameters**

* `time` – time of last modification of the image

• *suggestFileExtension*

```
public java.lang.String suggestFileExtension(
net.sourceforge.jiu.data.PixelImage image )
```

– **Description copied from ImageCodec (in 2.1.5, page 79)**

Attempts to suggest a filename extension. The type of the argument image will be taken into consideration, although this will be necessary for some file formats only (as an example, PNM has different extensions for different image types, see (in 2.1.10, page 115)). This default implementation always returns `null`.

– **Parameters**

* `image` – the image that is to be written to a file

– **Returns** – the file extension, including a leading dot, or `null` if no file extension can be recommended

2.1.10 Class PNMCodec

A codec to read and write Portable Anymap (PNM) image files. This format includes three file types well-known in the Unix world:

- PBM (Portable Bitmap - 1 bit per pixel bilevel image),
- PGM (Portable Graymap - grayscale image) and
- PPM (Portable Pixmap - RGB truecolor image).

Compression

The file format only allows for uncompressed storage.

ASCII mode / binary mode

PNM streams can be stored in binary mode or ASCII mode. ASCII mode files are text files with numbers representing the pixels. They become larger than their binary counterparts, but as they are very redundant they can be compressed well with archive programs. ASCII PGM and PPM files can have all kinds of maximum sample values, thus allowing for arbitrary precision. They are not restricted by byte limits. PBM streams always have two colors, no matter if they are ASCII or binary.

Color depth for PGM / PPM

The header of a PGM and PPM file stores a maximum sample value (such a value is not stored for PBM, where the maximum value is always 1). When in binary mode, PGM and PPM typically have a maximum sample value of 255, which makes PGM 8 bits per pixel and PPM 24 bits per pixel large. One sample will be stored as a single byte. However, there also exist binary PGM files with a maximum sample value larger than 255 and smaller than 65536. These files use two bytes per sample, in network byte order (big endian). I have yet to see PPM files with that property, but they are of course imagineable. 16 bpp

DPI values

PNM files cannot store the physical resolution in DPI.

Number of images

Only one image can be stored in a PNM file.

Usage example - load an image from a PNM file

```
PNMCodec codec = new PNMCodec();
codec.setFile("test.ppm", CodecMode.LOAD);
codec.process();
codec.close();
PixelImage image = codec.getImage();
```

Usage example - save an image to a PNM file

```
PNMCodec codec = new PNMCodec();
BilevelImage myFax = ...; // initialize
codec.setImage(myFax);
codec.setFile("out.pbm", CodecMode.SAVE);
codec.process();
codec.close();
```

Declaration

```
public class PNMCodec
  extends net.sourceforge.jiu.codecs.ImageCodec (in 2.1.5, page 79)
```

Field summary

IMAGE_TYPE_BILEVEL Image type constant for bilevel images, stored in PBM files.

IMAGE_TYPE_COLOR Image type constant for RGB truecolor images, stored in PPM files.

IMAGE_TYPE_GRAY Image type constant for grayscale images, stored in PGM files.

IMAGE_TYPE_UNKNOWN Image type constant for images of unknown type.

Constructor summary

PNMCodec()

Method summary

determineImageTypeFromFileName(String) Attempts to find the appropriate image type by looking at a file's name.

getAscii() Returns if ASCII mode was used for loading an image or will be used to store an image.

getFormatName()

getMimeTypeNames()

getTypicalFileExtension(int) Returns the typical file extension (including leading dot) for an image type.

isLoadingSupported()

isSavingSupported()

process()

setAscii(boolean) Specify whether ASCII mode is to be used when saving an image.

suggestFileExtension(PixelImage)

Fields

- public static final int **IMAGE_TYPE_UNKNOWN**
 - Image type constant for images of unknown type.

- public static final int **IMAGE_TYPE_BILEVEL**
 - Image type constant for bilevel images, stored in PBM files.
- public static final int **IMAGE_TYPE_GRAY**
 - Image type constant for grayscale images, stored in PGM files.
- public static final int **IMAGE_TYPE_COLOR**
 - Image type constant for RGB truecolor images, stored in PPM files.

Constructors

- *PNMCodec*
public **PNMCodec**()

Methods

- *determineImageTypeFromFileName*
public static int **determineImageTypeFromFileName**(java.lang.String **fileName**)
 - **Description**
Attempts to find the appropriate image type by looking at a file's name. Ignores case when comparing. Returns (in 2.1.10, page 117)for .pbm, (in 2.1.10, page 117)for .pgm and (in 2.1.10, page 117)for .ppm. Otherwise, (in 2.1.10, page 116)is returned. To get a file extension given that you have an image type, use (in 2.1.10, page 118).
 - **Parameters**
* **fileName** – the file name to be examined
 - **Returns** – one of the **IMAGE_TYPE_XXX** constants of this class
- *getAscii*
public java.lang.Boolean **getAscii**()
 - **Description**
Returns if ASCII mode was used for loading an image or will be used to store an image.
 - **Returns** – true for ASCII mode, false for binary mode, null if that information is not available
 - **See also**
* **PNMCodec.setAscii(boolean)** (in 2.1.10, page 119)
- *getFormatName*
public abstract java.lang.String **getFormatName**()
 - **Description copied from ImageCodec (in 2.1.5, page 79)**
Returns the name of the file format supported by this codec. All classes extending (in 2.1.5, page 79)must override this method. When overriding, leave out any words in a particular language so that this format name can be understood by everyone. Usually it is enough to return the format creator plus a typical abbreviation, e.g. **Microsoft BMP** or **Portable Anymap (PNM)**.

- **Returns** – name of the file format supported by this codec
-

- *getMimeTypes*

```
public abstract java.lang.String[] getMimeTypes( )
```

- **Description copied from ImageCodec (in 2.1.5, page 79)**
Return the **MIME** (at <http://www.faqs.org/rfcs/rfc2045.html>) (Multipurpose Internet Mail Extensions) type strings for this format, or null if none are available.
 - **Returns** – MIME type strings or null
-

- *getTypicalFileExtension*

```
public static java.lang.String getTypicalFileExtension( int imageType )
```

- **Description**
Returns the typical file extension (including leading dot) for an image type. Returns null for (in 2.1.10, page 116). To get the image type given that you have a file name, use (in 2.1.10, page 117).
 - **Parameters**
* *imageType* – the image type for which the extension is required
 - **Returns** – the file extension or null
-

- *isLoadingSupported*

```
public abstract boolean isLoadingSupported( )
```

- **Description copied from ImageCodec (in 2.1.5, page 79)**
Returns if this codec is able to load images in the file format supported by this codec. If true is returned this does not necessarily mean that all files in this format can be read, but at least some.
 - **Returns** – if loading is supported
-

- *isSavingSupported*

```
public abstract boolean isSavingSupported( )
```

- **Description copied from ImageCodec (in 2.1.5, page 79)**
Returns if this codec is able to save images in the file format supported by this codec. If true is returned this does not necessarily mean that all types files in this format can be written, but at least some.
 - **Returns** – if saving is supported
-

- *process*

```
public void process( ) throws  
net.sourceforge.jiu.ops.MissingParameterException,  
net.sourceforge.jiu.ops.OperationFailedException,  
net.sourceforge.jiu.ops.WrongParameterException
```

- **Description copied from net.sourceforge.jiu.ops.Operation (in 19.2.5, page 508)**
This method does the actual work of the operation. It must be called after all parameters have been given to the operation object.
- **Throws**

- * `net.sourceforge.jiu.ops.WrongParameterException` – if at least one of the input parameters was not initialized appropriately (values out of the valid interval, etc.)
 - * `net.sourceforge.jiu.ops.MissingParameterException` – if any mandatory parameter was not given to the operation
 - * `net.sourceforge.jiu.ops.OperationFailedException` –
-

- *setAscii*

```
public void setAscii( boolean asciiMode )
```

- **Description**

Specify whether ASCII mode is to be used when saving an image. Default is binary mode.

- **Parameters**

- * `asciiMode` – if true, ASCII mode is used, binary mode otherwise
-

- *suggestFileExtension*

```
public java.lang.String suggestFileExtension(
net.sourceforge.jiu.data.PixelImage image )
```

- **Description copied from ImageCodec (in 2.1.5, page 79)**

Attempts to suggest a filename extension. The type of the argument image will be taken into consideration, although this will be necessary for some file formats only (as an example, PNM has different extensions for different image types, see (in 2.1.10, page 115)). This default implementation always returns `null`.

- **Parameters**

- * `image` – the image that is to be written to a file

- **Returns** – the file extension, including a leading dot, or `null` if no file extension can be recommended

2.1.11 Class PSDCodec

A codec to read images from Photoshop PSD files. PSD was created by Adobe for their **Photoshop** (at <http://www.adobe.com/store/products/photoshop.html>) image editing software. Note that only a small subset of valid PSD files is supported by this codec. Typical file extension is `.psd`.

Declaration

```
public class PSDCodec
extends net.sourceforge.jiu.codecs.ImageCodec (in 2.1.5, page 79)
```

Constructor summary

PSDCodec()

Method summary

getFormatName()
getMimeTypes()
isLoadingSupported()
isSavingSupported()
process()

Constructors

- *PSDCodec*
public PSDCodec()

Methods

- *getFormatName*
public abstract java.lang.String getFormatName()
 - **Description copied from ImageCodec (in 2.1.5, page 79)**
Returns the name of the file format supported by this codec. All classes extending (in 2.1.5, page 79) must override this method. When overriding, leave out any words in a particular language so that this format name can be understood by everyone. Usually it is enough to return the format creator plus a typical abbreviation, e.g. **Microsoft BMP** or **Portable Anymap (PNM)**.
 - **Returns** – name of the file format supported by this codec

- *getMimeTypes*
public abstract java.lang.String[] getMimeTypes()
 - **Description copied from ImageCodec (in 2.1.5, page 79)**
Return the **MIME** (at <http://www.faqs.org/rfcs/rfc2045.html>) (Multipurpose Internet Mail Extensions) type strings for this format, or `null` if none are available.
 - **Returns** – MIME type strings or null

- *isLoadingSupported*

`public abstract boolean isLoadingSupported()`

- **Description copied from ImageCodec (in 2.1.5, page 79)**

Returns if this codec is able to load images in the file format supported by this codec. If `true` is returned this does not necessarily mean that all files in this format can be read, but at least some.

- **Returns** – if loading is supported

- *isSavingSupported*

`public abstract boolean isSavingSupported()`

- **Description copied from ImageCodec (in 2.1.5, page 79)**

Returns if this codec is able to save images in the file format supported by this codec. If `true` is returned this does not necessarily mean that all types files in this format can be written, but at least some.

- **Returns** – if saving is supported

- *process*

`public void process() throws`

`net.sourceforge.jiu.ops.MissingParameterException,`
`net.sourceforge.jiu.ops.OperationFailedException,`
`net.sourceforge.jiu.ops.WrongParameterException`

- **Description copied from net.sourceforge.jiu.ops.Operation (in 19.2.5, page 508)**

This method does the actual work of the operation. It must be called after all parameters have been given to the operation object.

- **Throws**

- * `net.sourceforge.jiu.ops.WrongParameterException` – if at least one of the input parameters was not initialized appropriately (values out of the valid interval, etc.)
- * `net.sourceforge.jiu.ops.MissingParameterException` – if any mandatory parameter was not given to the operation
- * `net.sourceforge.jiu.ops.OperationFailedException` –

2.1.12 Class RASCodec

A codec to read and write Sun Raster (RAS) image files. The typical file extension for this format is `.ras`.

Usage example

This code snippet demonstrate how to read a RAS file.

```
RASCodec codec = new RASCodec();
codec.setFile("image.ras", CodecMode.LOAD);
codec.process();
PixelImage loadedImage = codec.getImage();
```

Supported file types when reading

Only uncompressed RAS files are read. Only 8 bit (gray and paletted) and 24 bit are supported when reading.

Supported image types when writing

Only (in 14.1.8, page 325)/ uncompressed is supported when writing.

Bounds

The bounds concept of ImageCodec is supported so that you can load or save only part of an image.

File format documentation

This file format is documented as a man page `rasterfile(5)` on Sun Unix systems. That documentation can also be found online, e.g. at

<http://www.doc.ic.ac.uk/mac/manuals/sunos-manual-pages/sunos4/usr/man/man5/rasterfile.5.html> (at

<http://www.doc.ic.ac.uk/mac/manuals/sunos-manual-pages/sunos4/usr/man/man5/rasterfile.5.html>).

A **web search for rasterfile(5)** (at <http://www.google.com/search?q=rasterfile%285%29&sourceid=opera&num=0>) brings up other places as well.

Declaration

```
public class RASCodec
extends net.sourceforge.jiu.codecs.ImageCodec (in 2.1.5, page 79)
```

Constructor summary

RASCodec()

Method summary

```

getFormatName()
getMimeTypes()
isLoadingSupported()
isSavingSupported()
process()
suggestFileExtension(PixelImage)

```

Constructors

- *RASCodec*

```
public RASCodec( )
```

Methods

- *getFormatName*

```
public abstract java.lang.String getFormatName( )
```

 - **Description copied from ImageCodec (in 2.1.5, page 79)**
Returns the name of the file format supported by this codec. All classes extending (in 2.1.5, page 79) must override this method. When overriding, leave out any words in a particular language so that this format name can be understood by everyone. Usually it is enough to return the format creator plus a typical abbreviation, e.g. **Microsoft BMP** or **Portable Anymap (PNM)**.
 - **Returns** – name of the file format supported by this codec

- *getMimeTypes*

```
public abstract java.lang.String[] getMimeTypes( )
```

 - **Description copied from ImageCodec (in 2.1.5, page 79)**
Return the **MIME** (at <http://www.faqs.org/rfcs/rfc2045.html>) (Multipurpose Internet Mail Extensions) type strings for this format, or `null` if none are available.
 - **Returns** – MIME type strings or `null`

- *isLoadingSupported*

```
public abstract boolean isLoadingSupported( )
```

 - **Description copied from ImageCodec (in 2.1.5, page 79)**
Returns if this codec is able to load images in the file format supported by this codec. If `true` is returned this does not necessarily mean that all files in this format can be read, but at least some.
 - **Returns** – if loading is supported

- *isSavingSupported*

```
public abstract boolean isSavingSupported( )
```

 - **Description copied from ImageCodec (in 2.1.5, page 79)**
Returns if this codec is able to save images in the file format supported by this codec. If `true` is returned this does not necessarily mean that all types files in this format can be written, but at least some.

- **Returns** – if saving is supported
-

- *process*

`public void process()` throws
`net.sourceforge.jiu.ops.MissingParameterException`,
`net.sourceforge.jiu.ops.OperationFailedException`,
`net.sourceforge.jiu.ops.WrongParameterException`

- **Description copied from `net.sourceforge.jiu.ops.Operation` (in 19.2.5, page 508)**

This method does the actual work of the operation. It must be called after all parameters have been given to the operation object.

- **Throws**

- * `net.sourceforge.jiu.ops.WrongParameterException` – if at least one of the input parameters was not initialized appropriately (values out of the valid interval, etc.)
 - * `net.sourceforge.jiu.ops.MissingParameterException` – if any mandatory parameter was not given to the operation
 - * `net.sourceforge.jiu.ops.OperationFailedException` –
-

- *suggestFileExtension*

`public java.lang.String suggestFileExtension(
net.sourceforge.jiu.data.PixelImage image)`

- **Description copied from `ImageCodec` (in 2.1.5, page 79)**

Attempts to suggest a filename extension. The type of the argument image will be taken into consideration, although this will be necessary for some file formats only (as an example, PNM has different extensions for different image types, see (in 2.1.10, page 115)). This default implementation always returns `null`.

- **Parameters**

- * `image` – the image that is to be written to a file

- **Returns** – the file extension, including a leading dot, or `null` if no file extension can be recommended

2.2 Exceptions

2.2.1 Class InvalidFileStructureException

This exception is thrown during image loading, when the decoding process is made impossible by errors in the image file. If a codec has recognized the file format but finds irregularities in the data and cannot continue loading, it is supposed to throw an instance of this exception class. An unexpected end of the input stream also falls into this category. This typically means that the file is corrupt, but of course it could also be because of an error in the codec implementation.

If the format is not recognized at all, a (in 2.2.5, page 129) should be thrown.

If the format is recognized but cannot be loaded because the codec does not fully support the file format, a (in 2.2.4, page 128) should be thrown.

See also

- `UnsupportedTypeException` (in 2.2.4, page 128)
- `WrongFileFormatException` (in 2.2.5, page 129)

Declaration

```
public class InvalidFileStructureException
extends net.sourceforge.jiu.ops.OperationFailedException (in 19.3.2, page 513)
```

Constructor summary

InvalidFileStructureException(String)

Constructors

- *InvalidFileStructureException*
`public InvalidFileStructureException(java.lang.String message)`

2.2.2 Class InvalidImageIndexException

This exception is thrown when the caller has defined an image index that specifies the image to be loaded in a multiple-image file and that index is unavailable. Example: user has specified an image index of 5, which is the sixth image in the file (counting starts at 0), but only three images are available.

Declaration

```
public class InvalidImageIndexException
extends net.sourceforge.jiu.ops.OperationFailedException (in 19.3.2, page 513)
```

Constructor summary

InvalidImageIndexException(String) Creates new exception object with a given error message.

Constructors

- *InvalidImageIndexException*
public InvalidImageIndexException(java.lang.String message)
 - **Description**
Creates new exception object with a given error message.
 - **Parameters**
 - * **message** – String with text describing the exact problem

2.2.3 *Class* **UnsupportedCodecModeException**

This exception is thrown when a codec does not support the codec mode wanted by the user.

Example: A user gives an `OutputStream` to a codec, indicating that an image is to be saved, but the codec only supports loading.

Declaration

```
public class UnsupportedCodecModeException
extends net.sourceforge.jiu.ops.OperationFailedException (in 19.3.2, page 513)
```

Constructor summary

UnsupportedCodecModeException(String)

Constructors

- *UnsupportedCodecModeException*
`public UnsupportedCodecModeException(java.lang.String message)`

2.2.4 Class *UnsupportedTypeException*

This exception is thrown during image loading. If a codec recognizes the file format but does not support the exact subtype it encounters (the compression type is unknown or the color depth unsupported), an instance of this exception class is created.

If the format is not recognized at all, a *WrongFileFormatException* (in 2.2.5, page 129) should be thrown.

If there were errors during loading because of file corruption, an *InvalidFileStructureException* (in 2.2.1, page 125) must be thrown.

See also

- *InvalidFileStructureException* (in 2.2.1, page 125)
- *WrongFileFormatException* (in 2.2.5, page 129)

Declaration

```
public class UnsupportedTypeException
extends net.sourceforge.jiu.ops.OperationFailedException (in 19.3.2, page 513)
```

Constructor summary

UnsupportedTypeException(String)

Constructors

- *UnsupportedTypeException*
public **UnsupportedTypeException**(java.lang.String message)

2.2.5 Class WrongFormatException

This exception is thrown during image loading. If a codec is sure that the file or input stream that was given to it is not in the format supported by that codec, an instance of this exception class is thrown. This is usually the case if the magic byte sequence of that format is not found at the beginning of the stream.

If there were errors during loading because of file corruption, an (in 2.2.1, page 125) should be thrown.

If the format is recognized but cannot be loaded because the codec does not fully support the file format, a (in 2.2.4, page 128) should be thrown.

See also

- `InvalidFileStructureException` (in 2.2.1, page 125)
- `UnsupportedTypeException` (in 2.2.4, page 128)

Declaration

```
public class WrongFormatException
extends net.sourceforge.jiu.ops.OperationFailedException (in 19.3.2, page 513)
```

Constructor summary

WrongFormatException(String)

Constructors

- *WrongFormatException*
`public WrongFormatException(java.lang.String message)`

Chapter 3

Package `net.sourceforge.jiu.codecs.tiff`

Package Contents

Page

Interfaces

TIFFConstants	132
<i>This class provides a lot of constant values for a TIFF encoder or decoder.</i>	

Classes

TIFFCodec	137
<i>A codec to read Tagged Image File Format (TIFF) image files.</i>	
TIFFDecoder	143
<i>The abstract base class for a TIFF decoder, a class that decompresses one tile or strip of image data and understands one or more compression types.</i>	
TIFFDecoderDeflated	147
<i>A TIFF decoder for files compressed with the Deflated method.</i>	
TIFFDecoderLogLuv	149
<i>A TIFF decoder for files compressed with the LogLuv RLE method.</i>	
TIFFDecoderModifiedHuffman	151
<i>A TIFF decoder for files compressed with the Modified Huffman method (also known as CCITT 1D Modified Huffman Run Length Encoding).</i>	
TIFFDecoderPackbits	153
<i>A TIFF decoder for files compressed with the Packbits method.</i>	
TIFFDecoderUncompressed	154
<i>A TIFF decoder for uncompressed TIFF files.</i>	
TIFFFaxCodes	155
<i>Information to be used to decode and encode TIFF files in one of the bilevel compression types Modified Huffman, CCITT Group 3 or CCITT Group 4.</i>	
TIFFImageFileDirectory	157
<i>This class encapsulates all data of a TIFF image file directory (IFD).</i>	
TIFFRational	164
<i>Data class to store a TIFF rational number.</i>	
TIFFTag	166
<i>This encapsulates the data stored in a TIFF tag (a single image file directory entry).</i>	

Classes to handle the Tagged Image File Format (TIFF). See **the TIFFCodec documentation** (at [TIFFCodec.html](#)) for the amount of support that is built into this package.

Package Specification

The most important class is `TIFFCodec`, extending the base class for image codecs, `ImageCodec`. `TIFFCodec` reads the TIFF header, then the image file directory of the image to be loaded (TIFF can store more than one image in a file). The information of an image file directory is put into an object of class `TIFFImageFileDirectory`. It contains the tags of that directory (each tag is of type `TIFFTag`), and the most important information of a directory can also be retrieved from the various get methods (e.g. `getCompression`). TIFF files can be stored using all kinds of compression methods. When reading TIFFs, each supported compression method gets its own class extending `TIFFDecoder`, which provides basic methods required by all decoders (like storing decompressed data). A `TIFFCodec` object that is supposed to read an image creates an appropriate `TIFFDecoder` (e.g. `TIFFDecoderUncompressed` for compression type 1, no compression) for each strip or tile and lets them do the image loading.

3.1 Interfaces

3.1.1 *Interface* TIFFConstants

This class provides a lot of constant values for a TIFF encoder or decoder.

Declaration

```
public interface TIFFConstants
```

All known subclasses

TIFFTag (in 3.2.11, page 166), TIFFImageFileDirectory (in 3.2.9, page 157), TIFFCodec (in 3.2.1, page 137)

All classes known to implement interface

TIFFTag (in 3.2.11, page 166), TIFFImageFileDirectory (in 3.2.9, page 157), TIFFCodec (in 3.2.1, page 137)

Field summary

```

COMPRESSION_CCITT_GROUP3_1D_MODIFIED_HUFFMAN
COMPRESSION_CCITT_T4
COMPRESSION_CCITT_T6
COMPRESSION_DEFLATED_INOFFICIAL
COMPRESSION_DEFLATED_OFFICIAL
COMPRESSION_JBIG
COMPRESSION_JBIG2
COMPRESSION_JPEG_6_0
COMPRESSION_JPEG_POST_6_0
COMPRESSION_LZW
COMPRESSION_NEXT
COMPRESSION_NONE
COMPRESSION_NONE_WORD_ALIGNED
COMPRESSION_PACKBITS
COMPRESSION_SGI_LOG_24_PACKED
COMPRESSION_SGI_LOG_RLE
COMPRESSION_THUNDERSCAN
PHOTOMETRIC_BLACK_IS_ZERO
PHOTOMETRIC_LOGL
PHOTOMETRIC_PALETTED
PHOTOMETRIC_TRUECOLOR_CMYK
PHOTOMETRIC_TRUECOLOR_LOGLUV
PHOTOMETRIC_TRUECOLOR_RGB
PHOTOMETRIC_WHITE_IS_ZERO
PLANAR_CONFIGURATION_CHUNKY
PLANAR_CONFIGURATION_PLANAR
TAG_ARTIST
TAG_BAD_FAX_LINES
TAG_BITS_PER_SAMPLE

```

TAG_CELL_LENGTH
TAG_CELL_WIDTH
TAG_CLEAN_FAX_DATA
TAG_COLOR_MAP
TAG_COMPRESSION
TAG_CONSECUTIVE_BAD_FAX_LINES
TAG_COPYRIGHT
TAG_DATE_TIME
TAG_DOCUMENT_NAME
TAG_EXTRA_SAMPLES
TAG_FILL_ORDER
TAG_FREE_BYTE_COUNTS
TAG_FREE_OFFSETS
TAG_GRAY_RESPONSE_CURVE
TAG_GRAY_RESPONSE_UNIT
TAG_HOST_COMPUTER
TAG_IMAGE_DESCRIPTION
TAG_IMAGE_LENGTH
TAG_IMAGE_WIDTH
TAG_LENGTH Length of a tag (an image file directory entry) in bytes (12).
TAG_MAKE
TAG_MAX_SAMPLE_VALUE
TAG_MIN_SAMPLE_VALUE
TAG_MODEL
TAG_NEW_SUBFILE_TYPE
TAG_ORIENTATION
TAG_PHOTOMETRIC_INTERPRETATION
TAG_PHOTOSHOP_IMAGE_RESOURCES
TAG_PLANAR_CONFIGURATION
TAG_PREDICTOR
TAG_RESOLUTION_UNIT
TAG_RESOLUTION_X
TAG_RESOLUTION_Y
TAG_ROWS_PER_STRIP
TAG_SAMPLES_PER_PIXEL
TAG_SOFTWARE
TAG_STRIP_BYTE_COUNTS
TAG_STRIP_OFFSETS
TAG_T4_OPTIONS
TAG_T6_OPTIONS
TAG_TILE_BYTE_COUNTS
TAG_TILE_HEIGHT
TAG_TILE_OFFSETS
TAG_TILE_WIDTH
TAG_TYPE_ASCII
TAG_TYPE_BYTE
TAG_TYPE_DOUBLE
TAG_TYPE_FLOAT
TAG_TYPE_LONG
TAG_TYPE_RATIONAL

TAG_TYPE_SBYTE
TAG_TYPE_SHORT
TAG_TYPE_SLONG
TAG_TYPE_SRATIONAL
TAG_TYPE_SSHORT
TAG_TYPE_UNDEFINED

Fields

- int **COMPRESSION_NONE**
- int **COMPRESSION_CCITT_GROUP3_1D_MODIFIED_HUFFMAN**
- int **COMPRESSION_CCITT_T4**
- int **COMPRESSION_CCITT_T6**
- int **COMPRESSION_LZW**
- int **COMPRESSION_JPEG_6_0**
- int **COMPRESSION_JPEG_POST_6_0**
- int **COMPRESSION_DEFLATED_OFFICIAL**
- int **COMPRESSION_NEXT**
- int **COMPRESSION_NONE_WORD_ALIGNED**
- int **COMPRESSION_PACKBITS**
- int **COMPRESSION_THUNDERSCAN**
- int **COMPRESSION_DEFLATED_INOFFICIAL**
- int **COMPRESSION_JBIG**
- int **COMPRESSION_SGI_LOG_RLE**
- int **COMPRESSION_SGI_LOG_24_PACKED**
- int **COMPRESSION_JBIG2**
- int **PHOTOMETRIC_WHITE_IS_ZERO**
- int **PHOTOMETRIC_BLACK_IS_ZERO**
- int **PHOTOMETRIC_PALETTED**
- int **PHOTOMETRIC_TRUECOLOR_RGB**
- int **PHOTOMETRIC_TRUECOLOR_CMYK**
- int **PHOTOMETRIC_LOGL**
- int **PHOTOMETRIC_TRUECOLOR_LOGLUV**

- int **PLANAR_CONFIGURATION_CHUNKY**
- int **PLANAR_CONFIGURATION_PLANAR**
- int **TAG_LENGTH**
 - Length of a tag (an image file directory entry) in bytes (12).
- int **TAG_TYPE_BYTE**
- int **TAG_TYPE_ASCII**
- int **TAG_TYPE_SHORT**
- int **TAG_TYPE_LONG**
- int **TAG_TYPE_RATIONAL**
- int **TAG_TYPE_SBYTE**
- int **TAG_TYPE_UNDEFINED**
- int **TAG_TYPE_SSHORT**
- int **TAG_TYPE_SLONG**
- int **TAG_TYPE_SRATIONAL**
- int **TAG_TYPE_FLOAT**
- int **TAG_TYPE_DOUBLE**
- int **TAG_ARTIST**
- int **TAG_BAD_FAX_LINES**
- int **TAG_BITS_PER_SAMPLE**
- int **TAG_CELL_LENGTH**
- int **TAG_CELL_WIDTH**
- int **TAG_CLEAN_FAX_DATA**
- int **TAG_COLOR_MAP**
- int **TAG_COMPRESSION**
- int **TAG_CONSECUTIVE_BAD_FAX_LINES**
- int **TAG_COPYRIGHT**
- int **TAG_DATE_TIME**
- int **TAG_DOCUMENT_NAME**
- int **TAG_EXTRA_SAMPLES**
- int **TAG_FILL_ORDER**
- int **TAG_FREE_BYTE_COUNTS**

- int **TAG_FREE_OFFSETS**
- int **TAG_GRAY_RESPONSE_CURVE**
- int **TAG_GRAY_RESPONSE_UNIT**
- int **TAG_HOST_COMPUTER**
- int **TAG_IMAGE_DESCRIPTION**
- int **TAG_IMAGE_LENGTH**
- int **TAG_IMAGE_WIDTH**
- int **TAG_MAKE**
- int **TAG_MAX_SAMPLE_VALUE**
- int **TAG_MIN_SAMPLE_VALUE**
- int **TAG_MODEL**
- int **TAG_NEW_SUBFILE_TYPE**
- int **TAG_ORIENTATION**
- int **TAG_PHOTOMETRIC_INTERPRETATION**
- int **TAG_PHOTOSHOP_IMAGE_RESOURCES**
- int **TAG_PLANAR_CONFIGURATION**
- int **TAG_PREDICTOR**
- int **TAG_RESOLUTION_UNIT**
- int **TAG_RESOLUTION_X**
- int **TAG_RESOLUTION_Y**
- int **TAG_ROWS_PER_STRIP**
- int **TAG_SAMPLES_PER_PIXEL**
- int **TAG_SOFTWARE**
- int **TAG_STRIP_BYTE_COUNTS**
- int **TAG_STRIP_OFFSETS**
- int **TAG_T4_OPTIONS**
- int **TAG_T6_OPTIONS**
- int **TAG_TILE_BYTE_COUNTS**
- int **TAG_TILE_HEIGHT**
- int **TAG_TILE_OFFSETS**
- int **TAG_TILE_WIDTH**

3.2 Classes

3.2.1 *Class* TIFFCodec

A codec to read Tagged Image File Format (TIFF) image files.

Usage example

Load an image from a TIFF file.

```
TIFFCodec codec = new TIFFCodec();
codec.setFile("image.tif", CodecMode.LOAD);
codec.process();
PixelImage loadedImage = codec.getImage();
Saving images is not supported by this codec.
```

Compression types

Reading

The TIFF package supports the following compression types when reading:

- Uncompressed. Compression method number 1. Works with all types of image data. See (in 3.2.7, page 154).
- Packbits. Compression method number 32773. Works with all types of image data. See (in 3.2.6, page 153).
- CCITT Group 3 1-Dimensional Modified Huffman runlength encoding. Compression method number 2. Works with bilevel image data only. See (in 3.2.5, page 151).
- Deflated. Compression method number 8 or 32946. Works with all types of image data. See (in 3.2.3, page 147).
- LogLuv RLE and LogLuv 24. Compression method numbers 34676 and 34677. Works only with LogLuv color data. See (in 3.2.4, page 149).

Note that you can write your own decoder (extending (in 3.2.2, page 143)) for any compression type you want.

Image types

Reading

The TIFF package supports the following image / color types when reading:

- Black & white. JIU image data type (in 14.1.1, page 311).
- Grayscale, 4 and 8 bits per pixel. JIU image data type (in 14.1.4, page 318).
- TODO add other image types

Note that you can write your own decoder (extending (in 3.2.2, page 143)) for any compression type you want.

Writing

Writing TIFFs is not supported. I don't know if or when it will be supported.

Strips and tiles

The early versions of TIFF considered an image to be a sequence of strips. Each strip was a rectangular part of the image, as wide as the complete image, and with a certain height defined by the rows per strip tag. So with a number of rows per strip of 10, and an image height of 200, you would have to store 20 strips. It was recommended that a strip should not be larger than 8 KB (RAM was tighter in those days). The rule of thumb to define the number of rows per strip was to see how many rows would fit into 8 KB.

Later, the concept of tiles was added to the TIFF specs. Tiled TIFFs are separated into rectangles that not only had a defineable height but also a defineable width (tile width and tile height are also stored in corresponding tags).

Obviously, strips are just a special case of tiles, with the tile width being equal to image width. That is why JIU internally only deals with tiles. The only difference: No row padding takes place for strips. In a tiled image with a tile height of 10 and an image height of 14, the image is two tiles high.

Number of images

TIFF allows for multiple images in a single file. This codec regards the image index, queries (in 2.1.5, page 85) and skips to the correct image.

Bounds

The bounds concept of JIU is supported by this codec. So you can specify bounds of a rectangular part of an image that you want to load instead of loading the complete image.

Color spaces

The following color spaces are understood when reading truecolor TIFF files.

- RGB - should cover most truecolor files.
- CMYK - is supported, but colors may not be exactly right. CMYK data is converted to RGB on the fly, so the codec user never accesses CMYK data.
- LogLuv - is supported, but not all flavors yet.

Physical resolution

DPI information can be stored in TIFF files. If that information is available, this codec retrieves it so that it can be queried using (in 2.1.5, page 84) and (in 2.1.5, page 85).

Background information on TIFF

TIFF is an important image file format for DTP (desktop publishing). The advantages of TIFF include its flexibility, availability of libraries to read and write TIFF files and its good support in existing software. The major disadvantage of TIFF is its complexity, which makes it hard for software to support all possible valid TIFF files.

TIFF was created by Aldus and now belongs to Adobe, who offer a specification document:

TIFF (Tagged Image File Format) 6.0 Specification (at

<http://partners.adobe.com/asn/developer/PDFS/TN/TIFF6.pdf>) (updated on Web September, 20 1995, document dated June, 3 1992) (PDF: 385 KB / 121 pages).

Other good references include the **homepage of libtiff** (at <http://www.libtiff.org>), a free C library to read and write TIFF files and **The Unofficial TIFF homepage** (at

<http://home.earthlink.net/~ritter/tiff/>) by **Niles Ritter** (at <mailto:ritter@earthlink.net>). Also see **the TIFF section** (at http://dmoz.org/Computers/Data_Formats/Graphics/Pixmap/TIFF/) of the **Open Directory** (at <http://www.dmoz.org>).

TIFF is used for various specialized tasks. As an example, see **GeoTIFF** (at <http://www.remotesensing.org/geotiff/geotiff.html>) (geographical data) or **EXIF** (at <http://www.ba.wakwak.com/~tsuruzoh/index-e.html>) (digital camera metadata; this is actually a TIFF directory embedded in a JPEG header).

Here's a list of features that make TIFF quite complex:

- More than one image can be stored in a TIFF file.
- Integer values that are larger than one byte can be in either little or big endian byte order.
- Various color types are supported (bilevel, gray, paletted, all kinds of color spaces (RGB / YCbCr / CMYK). It's easy to add new color types, so this list can grow.
- The meta data (information that describes the image and how it is stored) can be distributed all over the file.
- Image data is stored as packed bytes, 4-bit-nibbles, bytes and 16-bit-integers. Other types are possible as well.
- Various compression types are supported; not all types can be used on all color types.
- Image data can be stored in strips or tiles.
- An arbitrary number of non-image-data samples can stored within the image data.
- Color types with more than one sample per pixel can store data in an interleaved (chunky) way or in planes.
- Different ways of defining black and white are possible with bilevel and grayscale images.

Declaration

```
public class TIFFCodec
  extends net.sourceforge.jiu.codecs.ImageCodec (in 2.1.5, page 79)
  implements TIFFConstants
```

Field summary

BYTE_ORDER_INTEL
BYTE_ORDER_MOTOROLA

Constructor summary

TIFFCodec()

Method summary

getByteOrder() Returns the current byte order, either (in 3.2.1, page 140) or (in 3.2.1, page 140).
getFormatName()
getMimeType()
getTagName(int) Returns the name of a tag in English.
isLoadingSupported()
isSavingSupported()

```

process()
registerDecoder(Class) Register a (in 3.2.2, page 143)class.
setFile(String, CodecMode)

```

Fields

- public static final int **BYTE_ORDER_MOTOROLA**
- public static final int **BYTE_ORDER_INTEL**

Constructors

- *TIFFCodec*
public **TIFFCodec**()

Methods

- *getByteOrder*
public int **getByteOrder**()
 - **Description**
Returns the current byte order, either (in 3.2.1, page 140) or (in 3.2.1, page 140).
 - **Returns** – current byte order

- *getFormatName*
public abstract java.lang.String **getFormatName**()
 - **Description copied from net.sourceforge.jiu.codecs.ImageCodec (in 2.1.5, page 79)**
Returns the name of the file format supported by this codec. All classes extending (in 2.1.5, page 79) must override this method. When overriding, leave out any words in a particular language so that this format name can be understood by everyone. Usually it is enough to return the format creator plus a typical abbreviation, e.g. Microsoft BMP or Portable Anymap (PNM).
 - **Returns** – name of the file format supported by this codec

- *getMimeType*
public abstract java.lang.String[] **getMimeType**()
 - **Description copied from net.sourceforge.jiu.codecs.ImageCodec (in 2.1.5, page 79)**
Return the **MIME** (at <http://www.faqs.org/rfcs/rfc2045.html>) (Multipurpose Internet Mail Extensions) type strings for this format, or null if none are available.
 - **Returns** – MIME type strings or null

- *getTagName*
public static java.lang.String **getTagName**(int id)

– **Description**

Returns the name of a tag in English.

– **Parameters**

* `id` – of the tag for which a name is to be returned

– **Returns** – tag name as String or a question mark ?

• *isLoadingSupported*

`public abstract boolean isLoadingSupported()`

– **Description copied from net.sourceforge.jiu.codecs.ImageCodec (in 2.1.5, page 79)**

Returns if this codec is able to load images in the file format supported by this codec. If `true` is returned this does not necessarily mean that all files in this format can be read, but at least some.

– **Returns** – if loading is supported

• *isSavingSupported*

`public abstract boolean isSavingSupported()`

– **Description copied from net.sourceforge.jiu.codecs.ImageCodec (in 2.1.5, page 79)**

Returns if this codec is able to save images in the file format supported by this codec. If `true` is returned this does not necessarily mean that all types files in this format can be written, but at least some.

– **Returns** – if saving is supported

• *process*

`public void process()` throws
`net.sourceforge.jiu.ops.MissingParameterException`,
`net.sourceforge.jiu.ops.OperationFailedException`,
`net.sourceforge.jiu.ops.WrongParameterException`

– **Description copied from net.sourceforge.jiu.ops.Operation (in 19.2.5, page 508)**

This method does the actual work of the operation. It must be called after all parameters have been given to the operation object.

– **Throws**

- * `net.sourceforge.jiu.ops.WrongParameterException` – if at least one of the input parameters was not initialized appropriately (values out of the valid interval, etc.)
 - * `net.sourceforge.jiu.ops.MissingParameterException` – if any mandatory parameter was not given to the operation
 - * `net.sourceforge.jiu.ops.OperationFailedException` –
-

• *registerDecoder*

`public static void registerDecoder(java.lang.Class decoderClass)`

– **Description**

Register a (in 3.2.2, page 143)class. TIFF knows many compression types, and JIU only supports some of them. To register an external TIFFDecoder class with TIFFCodec, call this method with the class field of your decoder. As an example, for your

TIFFDecoderMyCompression class, call

TIFFCodec.registerDecoder(TIFFDecoderMyCompression.class). It will be checked if `decoderClass.newInstance() instanceof TIFFDecoder` is true and, if so, the class will be added to an internal list. Whenever a TIFF file is to be decoded, the correct decoder is determined (each decoder knows about the compression types it supports via the `getCompressionTypes` method) and for each tile or strip such a decoder object will be created.

- *setFile*

```
public void setFile( java.lang.String fileName,
net.sourceforge.jiu.codecs.CodecMode codecMode ) throws
java.io.IOException,
net.sourceforge.jiu.codecs.UnsupportedCodecModeException
```

- **Description copied from net.sourceforge.jiu.codecs.ImageCodec (in 2.1.5, page 79)**

Gives a file name and codec mode to the codec which will then try to create the corresponding I/O object. The default implementation in ImageCodec creates a DataInputStream object wrapped around a BufferedInputStream wrapped around a FileInputStream for CodecMode.LOAD. Similar for CodecMode.SAVE: a DataOutputStream around a BufferedOutputStream object around a FileOutputStream object. Codecs that need different I/O objects must override this method (some codecs may need random access and thus require a RandomAccessFile object).

- **Parameters**

- * `fileName` – name of the file to be used for loading or saving
- * `codecMode` – defines whether file is to be used for loading or saving

3.2.2 Class TIFFDecoder

The abstract base class for a TIFF decoder, a class that decompresses one tile or strip of image data and understands one or more compression types. Each child class implements the decoding of a particular TIFF compression type in its `decode()` method.

This class does all the work of storing decompressed data (given as a byte array) in the image object. Given the many variants (sample order, color depth, color space etc.) this is a larger portion of code.

Declaration

```
public abstract class TIFFDecoder
extends java.lang.Object
```

All known subclasses

TIFFDecoderUncompressed (in 3.2.7, page 154), TIFFDecoderPackbits (in 3.2.6, page 153), TIFFDecoderModifiedHuffman (in 3.2.5, page 151), TIFFDecoderLogLuv (in 3.2.4, page 149), TIFFDecoderDeflated (in 3.2.3, page 147)

Constructor summary

TIFFDecoder()

Method summary

decode() Decode data from input and write the decompressed pixel data to the image associated with this decoder.

getBytesPerRow() Returns the number of bytes per row for the strip or tile that this decoder deals with.

getCodec() Returns the codec from which this decoder is used.

getCompressionTypes() Returns an array with Integer values of all compression types supported by this decoder (see the `COMPRESSION_xyz` constants in (in 3.1.1, page 132).

getImageFileDirectory() Returns the IFD for the image this decoder is supposed to uncompress (partially).

getInput() Returns the input stream from which this decoder is supposed to read data.

getTileIndex() Returns the zero-based index of the tile or strip this decoder is supposed to be decompressing.

getX1() Returns the leftmost column of the image strip / tile to be read by this decoder.

getX2() Returns the rightmost column of the image strip / tile to be read by this decoder.

getY1() Returns the top row of the image strip / tile to be read by this decoder.

getY2() Returns the bottom row of the image strip / tile to be read by this decoder.

initialize() Check if all necessary parameters have been given to this decoder and initialize several internal fields from them.

putBytes(byte[], int, int) Adds a number of bytes to the internal row buffer.

setCodec(TIFFCodec) Specify the codec to be used with this decoder.

setImageFileDirectory(TIFFImageFileDirectory) Specify the IFD to be used with this decoder.

setTileIndex(int) Specify the zero-based tile index for the tile or strip to be decompressed by this decoder.

Constructors

- *TIFFDecoder*
public **TIFFDecoder**()

Methods

- *decode*
public abstract void **decode**() throws
net.sourceforge.jiu.codecs.InvalidFileStructureException,
java.io.IOException
 - **Description**
Decode data from input and write the decompressed pixel data to the image associated with this decoder. Child classes must override this method to implement the decoding for a particular compression type.
- *getBytesPerRow*
public int **getBytesPerRow**()
 - **Description**
Returns the number of bytes per row for the strip or tile that this decoder deals with. So with a tiled TIFF and an image width of 500 and a tile width of 100, for an eight bit grayscale image this would return 100 (not 500).
 - **Returns** – number of bytes per row
- *getCodec*
public TIFFCodec **getCodec**()
 - **Description**
Returns the codec from which this decoder is used.
 - **Returns** – TIFFCodec object using this decoder
- *getCompressionTypes*
public abstract java.lang.Integer[] **getCompressionTypes**()
 - **Description**
Returns an array with Integer values of all compression types supported by this decoder (see the COMPRESSION_xyz constants in (in 3.1.1, page 132). Normally, this is only one value, but some compression types got assigned more than one constant (e.g. deflated). Also, a decoder could be capable of dealing with more than one type of compression if the compression types are similar enough to justify that. However, typically a decoder can only deal with one type of compression.
 - **Returns** – array with Integer objects of all TIFF compression constants supported by this decoder

- *getImageFileDirectory*

public TIFFImageFileDirectory **getImageFileDirectory**()

- **Description**

Returns the IFD for the image this decoder is supposed to uncompress (partially).

- **Returns** – IFD object

- *getInput*

public java.io.DataInput **getInput**()

- **Description**

Returns the input stream from which this decoder is supposed to read data.

- *getTileIndex*

public int **getTileIndex**()

- **Description**

Returns the zero-based index of the tile or strip this decoder is supposed to be decompressing.

- **Returns** – tile index

- *getX1*

public int **getX1**()

- **Description**

Returns the leftmost column of the image strip / tile to be read by this decoder.

- *getX2*

public int **getX2**()

- **Description**

Returns the rightmost column of the image strip / tile to be read by this decoder.

- *getY1*

public int **getY1**()

- **Description**

Returns the top row of the image strip / tile to be read by this decoder.

- *getY2*

public int **getY2**()

- **Description**

Returns the bottom row of the image strip / tile to be read by this decoder.

- *initialize*

public void **initialize**() throws java.io.IOException,
net.sourceforge.jiu.ops.MissingParameterException

- **Description**

Check if all necessary parameters have been given to this decoder and initialize several internal fields from them. Required parameters are a TIFFCodec object, a TIFFImageFileDirectory object and a tile index.

- *putBytes*

```
public void putBytes( byte[] data, int offset, int number )
```

- **Description**

Adds a number of bytes to the internal row buffer. If the row buffer gets full (a complete line is available) that data will be copied to the image. Note that more than one line, exactly one line or only part of a line can be stored in the **number** bytes in **data**.

- **Parameters**

- * **data** – byte array with image data that has been decoded
 - * **offset** – int index into data where the first byte to be stored is situated
 - * **number** – int number of bytes to be stored
-

- *setCodec*

```
public void setCodec( TIFFCodec tiffCodec )
```

- **Description**

Specify the codec to be used with this decoder. This is a mandatory parameter - without it, (in 3.2.2, page 145) will throw an exception.

- **Parameters**

- * **tiffCodec** – TIFFCodec object to be used by this decoder

- **See also**

- * `TIFFDecoder.getCodec()` (in 3.2.2, page 144)
-

- *setImageFileDirectory*

```
public void setImageFileDirectory( TIFFImageFileDirectory tiffIfd )
```

- **Description**

Specify the IFD to be used with this decoder. This is a mandatory parameter - without it, (in 3.2.2, page 145) will throw an exception.

- **Parameters**

- * **tiffIfd** – object to be used by this decoder

- **See also**

- * `TIFFDecoder.getImageFileDirectory()` (in 3.2.2, page 145)
-

- *setTileIndex*

```
public void setTileIndex( int index )
```

- **Description**

Specify the zero-based tile index for the tile or strip to be decompressed by this decoder. This is a mandatory parameter - without it, (in 3.2.2, page 145) will throw an exception.

- **Parameters**

- * **index** – zero-based tile / strip index

- **See also**

- * `TIFFDecoder.getTileIndex()` (in 3.2.2, page 145)

3.2.3 Class TIFFDecoderDeflated

A TIFF decoder for files compressed with the Deflated method. This compression algorithm has the values 31946 ((in 3.1.1, page 134)) and 8 ((in 3.1.1, page 134)) in the compression tag of an image file directory. All types of image data can be compressed with this method. This decoder makes use of the package java.util.zip which comes with an Inflater class which does most of the use. All the decoder has to do is feed it with compressed data from the input file. and give decompressed data received from the Inflater to the putBytes method.

Declaration

```
public class TIFFDecoderDeflated
extends net.sourceforge.jiu.codecs.tiff.TIFFDecoder (in 3.2.2, page 143)
```

Constructor summary

TIFFDecoderDeflated()

Method summary

decode()
getCompressionTypes()
initialize()

Constructors

- *TIFFDecoderDeflated*
public TIFFDecoderDeflated()

Methods

- *decode*
public abstract void decode() throws
net.sourceforge.jiu.codecs.InvalidFileStructureException,
java.io.IOException
 - **Description copied from TIFFDecoder (in 3.2.2, page 143)**
Decode data from input and write the decompressed pixel data to the image associated with this decoder. Child classes must override this method to implement the decoding for a particular compression type.
- *getCompressionTypes*
public abstract java.lang.Integer[] getCompressionTypes()
 - **Description copied from TIFFDecoder (in 3.2.2, page 143)**
Returns an array with Integer values of all compression types supported by this decoder (see the COMPRESSION_xyz constants in (in 3.1.1, page 132). Normally, this is only one value, but some compression types got assigned more than one constant (e.g. deflated). Also, a decoder could be capable of dealing with more than one type of compression if the compression types are similar enough to justify that. However, typically a decoder can only deal with one type of compression.

- **Returns** – array with Integer objects of all TIFF compression constants supported by this decoder

- *initialize*

public void **initialize**() throws java.io.IOException,
net.sourceforge.jiu.ops.MissingParameterException

- **Description copied from TIFFDecoder (in 3.2.2, page 143)**
Check if all necessary parameters have been given to this decoder and initialize several internal fields from them. Required parameters are a TIFFCodec object, a TIFFImageFileDirectory object and a tile index.

3.2.4 Class TIFFDecoderLogLuv

A TIFF decoder for files compressed with the LogLuv RLE method. This compression algorithm has the value 34676 ((in 3.1.1, page 134)) in the compression tag of an image file directory. Only image data with a photometric interpretation value of (in 3.1.1, page 134) can be compressed with this method.

This implementation is based on the file `tif_luv.c` which is part of the TIFF library **libtiff** (at <http://www.libtiff.org>). The original implementation was written by Greg W. Larson.

Learn more about the color type and its encoding on Greg's page **LogLuv Encoding for TIFF Images** (at <http://positron.cs.berkeley.edu/~gwlarsen/pixformat/tiffluv.html>). You will also find numerous sample image files there.

Declaration

```
public class TIFFDecoderLogLuv
  extends net.sourceforge.jiu.codecs.tiff.TIFFDecoder (in 3.2.2, page 143)
```

Constructor summary

TIFFDecoderLogLuv()

Method summary

decode()
getCompressionTypes()
initialize()

Constructors

- *TIFFDecoderLogLuv*
 public **TIFFDecoderLogLuv**()

Methods

- *decode*
 public abstract void **decode**() throws
 net.sourceforge.jiu.codecs.InvalidFileStructureException,
 java.io.IOException
 – **Description copied from TIFFDecoder (in 3.2.2, page 143)**
 Decode data from input and write the decompressed pixel data to the image associated with this decoder. Child classes must override this method to implement the decoding for a particular compression type.
- *getCompressionTypes*
 public abstract java.lang.Integer[] **getCompressionTypes**()

– **Description copied from TIFFDecoder (in 3.2.2, page 143)**

Returns an array with Integer values of all compression types supported by this decoder (see the COMPRESSION_xyz constants in (in 3.1.1, page 132). Normally, this is only one value, but some compression types got assigned more than one constant (e.g. deflated). Also, a decoder could be capable of dealing with more than one type of compression if the compression types are similar enough to justify that. However, typically a decoder can only deal with one type of compression.

– **Returns** – array with Integer objects of all TIFF compression constants supported by this decoder

• *initialize*

public void **initialize**() throws java.io.IOException,
net.sourceforge.jiu.ops.MissingParameterException

– **Description copied from TIFFDecoder (in 3.2.2, page 143)**

Check if all necessary parameters have been given to this decoder and initialize several internal fields from them. Required parameters are a TIFFCodec object, a TIFFImageFileDirectory object and a tile index.

3.2.5 Class TIFFDecoderModifiedHuffman

A TIFF decoder for files compressed with the Modified Huffman method (also known as CCITT 1D Modified Huffman Run Length Encoding). This compression algorithm has the value 2 in the compression tag of an image file directory. Only bilevel images can be encoded with that method.

Declaration

```
public class TIFFDecoderModifiedHuffman
  extends net.sourceforge.jiu.codecs.tiff.TIFFDecoder (in 3.2.2, page 143)
```

Constructor summary

TIFFDecoderModifiedHuffman()

Method summary

decode()
getCompressionTypes()
initialize()

Constructors

- *TIFFDecoderModifiedHuffman*
public TIFFDecoderModifiedHuffman()

Methods

- *decode*
public abstract void decode() throws
net.sourceforge.jiu.codecs.InvalidFileStructureException,
java.io.IOException
 - **Description copied from TIFFDecoder (in 3.2.2, page 143)**
 Decode data from input and write the decompressed pixel data to the image associated with this decoder. Child classes must override this method to implement the decoding for a particular compression type.
- *getCompressionTypes*
public abstract java.lang.Integer[] getCompressionTypes()
 - **Description copied from TIFFDecoder (in 3.2.2, page 143)**
 Returns an array with Integer values of all compression types supported by this decoder (see the COMPRESSION_xyz constants in (in 3.1.1, page 132). Normally, this is only one value, but some compression types got assigned more than one constant (e.g. deflated). Also, a decoder could be capable of dealing with more than one type of compression if the compression types are similar enough to justify that. However, typically a decoder can only deal with one type of compression.
 - **Returns** – array with Integer objects of all TIFF compression constants supported by this decoder

-
- *initialize*
`public void initialize()` throws `java.io.IOException`,
`net.sourceforge.jiu.ops.MissingParameterException`
 - **Description copied from TIFFDecoder (in 3.2.2, page 143)**
Check if all necessary parameters have been given to this decoder and initialize several internal fields from them. Required parameters are a TIFFCodec object, a TIFFImageFileDirectory object and a tile index.

3.2.6 Class TIFFDecoderPackbits

A TIFF decoder for files compressed with the Packbits method. This compression algorithm has the value 32773 in the compression tag of an image file directory.

Declaration

```
public class TIFFDecoderPackbits
  extends net.sourceforge.jiu.codecs.tiff.TIFFDecoder (in 3.2.2, page 143)
```

Constructor summary

TIFFDecoderPackbits()

Method summary

decode()
getCompressionTypes()

Constructors

- *TIFFDecoderPackbits*
 public **TIFFDecoderPackbits**()

Methods

- *decode*
 public abstract void **decode**() throws
 net.sourceforge.jiu.codecs.InvalidFileStructureException,
 java.io.IOException
 - **Description copied from TIFFDecoder (in 3.2.2, page 143)**
 Decode data from input and write the decompressed pixel data to the image associated with this decoder. Child classes must override this method to implement the decoding for a particular compression type.
- *getCompressionTypes*
 public abstract java.lang.Integer[] **getCompressionTypes**()
 - **Description copied from TIFFDecoder (in 3.2.2, page 143)**
 Returns an array with Integer values of all compression types supported by this decoder (see the COMPRESSION_xyz constants in (in 3.1.1, page 132). Normally, this is only one value, but some compression types got assigned more than one constant (e.g. deflated). Also, a decoder could be capable of dealing with more than one type of compression if the compression types are similar enough to justify that. However, typically a decoder can only deal with one type of compression.
 - **Returns** – array with Integer objects of all TIFF compression constants supported by this decoder

3.2.7 Class TIFFDecoderUncompressed

A TIFF decoder for uncompressed TIFF files.

Declaration

```
public class TIFFDecoderUncompressed
  extends net.sourceforge.jiu.codecs.tiff.TIFFDecoder (in 3.2.2, page 143)
```

Constructor summary

TIFFDecoderUncompressed()

Method summary

decode()
getCompressionTypes()

Constructors

- *TIFFDecoderUncompressed*
 public **TIFFDecoderUncompressed**()

Methods

- *decode*
 public abstract void **decode**() throws
 net.sourceforge.jiu.codecs.InvalidFileStructureException,
 java.io.IOException
 - **Description copied from TIFFDecoder (in 3.2.2, page 143)**
 Decode data from input and write the decompressed pixel data to the image associated with this decoder. Child classes must override this method to implement the decoding for a particular compression type.
- *getCompressionTypes*
 public abstract java.lang.Integer[] **getCompressionTypes**()
 - **Description copied from TIFFDecoder (in 3.2.2, page 143)**
 Returns an array with Integer values of all compression types supported by this decoder (see the COMPRESSION_xyz constants in (in 3.1.1, page 132). Normally, this is only one value, but some compression types got assigned more than one constant (e.g. deflated). Also, a decoder could be capable of dealing with more than one type of compression if the compression types are similar enough to justify that. However, typically a decoder can only deal with one type of compression.
 - **Returns** – array with Integer objects of all TIFF compression constants supported by this decoder

3.2.8 Class TIFFFaxCodes

Information to be used to decode and encode TIFF files in one of the bilevel compression types Modified Huffman, CCITT Group 3 or CCITT Group 4.

Declaration

```
public class TIFFFaxCodes
    extends java.lang.Object
```

Field summary

BLACK_CODES The code words and their meanings for black codes.
INDEX_CODE_VALUE Index of the code value in the int[] value pairs.
INDEX_CODE_WORD Index of the code word in the int[] value pairs.
MIN_BLACK_CODE_SIZE Minimum code length in bits of black codes.
MIN_WHITE_CODE_SIZE Minimum code length in bits of white codes.
WHITE_CODES The code words and their meanings for white codes.

Constructor summary

TIFFFaxCodes()

Fields

- public static final int **INDEX_CODE_WORD**
 - Index of the code word in the int[] value pairs.
- public static final int **INDEX_CODE_VALUE**
 - Index of the code value in the int[] value pairs.
- public static final int **MIN_BLACK_CODE_SIZE**
 - Minimum code length in bits of black codes.
- public static final int **MIN_WHITE_CODE_SIZE**
 - Minimum code length in bits of white codes.
- public static final int **BLACK_CODES**
 - The code words and their meanings for black codes. In ascending order, starting at MIN_BLACK_CODE_SIZE bits, each int[][] object contains all the code word / code value pairs for one bit length.
- public static final int **WHITE_CODES**
 - The code words and their meanings for white codes. In ascending order, starting at MIN_WHITE_CODE_SIZE bits, each int[][] object contains all the code word / code value pairs for one bit length.

Constructors

- *TIFFFaxCodes*
`public TIFFFaxCodes()`

3.2.9 Class TIFFImageFileDirectory

This class encapsulates all data of a TIFF image file directory (IFD).

Declaration

```
public class TIFFImageFileDirectory
extends java.lang.Object
implements TIFFConstants
```

Field summary

TYPE_BILEVEL_BYTE
TYPE_BILEVEL_PACKED
TYPE_CMYK32_INTERLEAVED
TYPE_CMYK32_PLANAR
TYPE_GRAY16
TYPE_GRAY4
TYPE_GRAY8
TYPE_LOGL
TYPE_LOGLUV32_INTERLEAVED
TYPE_PALETTED4
TYPE_PALETTED8
TYPE_RGB24_INTERLEAVED
TYPE_RGB48_INTERLEAVED

Constructor summary

TIFFImageFileDirectory() Initializes all members to null or -1 and creates an internal list for the tags that will be make up this directory.

Method summary

append(TIFFTag) Adds a tag to the end of the internal list of tags.
computeNumBytes(int) TODO: regard extra samples
getArtist() Returns information on the person who created the image (as stored in tag (in 3.1.1, page 135)).
getBitsPerPixel() Returns the number of bits per pixel (not including transparency information).
getByteCount(int) Returns the number of compressed byte for a given tile.
getBytesPerRow()
getCompression() Returns the compression method, encoded as a number as found in (in 3.1.1, page 132)(more specifically, the COMPRESSION_xyz constants).
getCompressionName(int) Returns the name of a TIFF compression method.
getCopyright()
getDateTime() If a date / time tag was found in this image file directory and (in 3.2.9, page 162)was called already, it was attempted to create a object from it.
getDateTimeString() If there was a date / time tag in this IFD, its String value is returned.
getDpiX()
getDpiY()

```

getHeight()
getHostComputer()
getImageDescription()
getImageType()
getModel()
getNumHorizontalTiles()
getNumStrips()
getNumTiles()
getNumVerticalTiles()
getPalette()
getPhotometricInterpretation()
getPredictor()
getRowsPerStrip()
getSamplesPerPixel()
getSoftware()
getStripOffsets()
getT4Options()
getT6Options()
getTileHeight()
getTileOffset(int)
getTileWidth()
getTileX1(int)
getTileX2(int)
getTileY1(int)
getTileY2(int)
getWidth()
initFromTags(boolean)
initMembers()
isGrayscale()
isPaletted()
isStriped() Returns true if the image belonging to this IFD is stored as strips, false
              otherwise.
isTiled() Returns true if the image belonging to this IFD is stored as tiles, false
           otherwise.
setTimeZone(TimeZone) Sets the time zone to be used when trying to interpret
                      dates found in a (in 3.1.1, page 135)tag.

```

Fields

- public static final int **TYPE_BILEVEL_PACKED**
- public static final int **TYPE_GRAY4**
- public static final int **TYPE_GRAY8**
- public static final int **TYPE_GRAY16**
- public static final int **TYPE_PALETTED4**
- public static final int **TYPE_PALETTED8**

- public static final int **TYPE_RGB24_INTERLEAVED**
- public static final int **TYPE_RGB48_INTERLEAVED**
- public static final int **TYPE_BILEVEL_BYTE**
- public static final int **TYPE_CMYK32_INTERLEAVED**
- public static final int **TYPE_CMYK32_PLANAR**
- public static final int **TYPE_LOGLUV32_INTERLEAVED**
- public static final int **TYPE_LOGL**

Constructors

- *TIFFImageFileDirectory*
public **TIFFImageFileDirectory**()
 – **Description**
 Initializes all members to null or -1 and creates an internal list for the tags that will be make up this directory.

Methods

- *append*
public void **append**(TIFFTag tag)
 – **Description**
 Adds a tag to the end of the internal list of tags.
 – **Parameters**
 * tag – the TIFFTag instance to be appended
- *computeNumBytes*
public int **computeNumBytes**(int numPixels)
 – **Description**
 TODO: regard extra samples
- *getArtist*
public java.lang.String **getArtist**()
 – **Description**
 Returns information on the person who created the image (as stored in tag (in 3.1.1, page 135)).
- *getBitsPerPixel*
public int **getBitsPerPixel**()
 – **Description**
 Returns the number of bits per pixel (not including transparency information).

- *getByteCount*

```
public int getByteCount( int tileIndex )
```

- **Description**

Returns the number of compressed byte for a given tile. Tile index must not be negative and must be smaller than the number of tiles.

- **Parameters**

* **tileIndex** – zero-based index of tile or strip for which the number of compressed bytes is to be returned

- *getBytesPerRow*

```
public int getBytesPerRow( )
```

- *getCompression*

```
public int getCompression( )
```

- **Description**

Returns the compression method, encoded as a number as found in (in 3.1.1, page 132)(more specifically, the COMPRESSION_xyz constants). Use (in 3.2.9, page 160)to get the English name of this compression method.

- **Returns** – compression method

- *getCompressionName*

```
public static java.lang.String getCompressionName( int method )
```

- **Description**

Returns the name of a TIFF compression method. If the name is unknown, Unknown method plus the method number is returned. This static method can be used in combination with the value from (in 3.2.9, page 160).

- **Parameters**

* **method** – the compression method number

- **Returns** – the compression method name

- *getCopyright*

```
public java.lang.String getCopyright( )
```

- *getDateTime*

```
public java.util.Date getDateTime( )
```

- **Description**

If a date / time tag was found in this image file directory and (in 3.2.9, page 162)was called already, it was attempted to create a object from it. This object (or null) is returned. Use (in 3.2.9, page 163)to provide a time zone before the date parsing is done.

- **See also**

* TIFFImageFileDirectory.getDateTimeString() (in 3.2.9, page 160)

- *getDateTimeString*

```
public java.lang.String getDateTimeString( )
```

- **Description**

If there was a date / time tag in this IFD, its String value is returned.

- **See also**

* `TIFFImageFileDirectory.getDateTime()` (in 3.2.9, page 160)

- *getDpiX*
`public int getDpiX()`
- *getDpiY*
`public int getDpiY()`
- *getHeight*
`public int getHeight()`
- *getHostComputer*
`public java.lang.String getHostComputer()`
- *getImageDescription*
`public java.lang.String getImageDescription()`
- *getImageType*
`public int getImageType()`
- *getModel*
`public java.lang.String getModel()`
- *getNumHorizontalTiles*
`public int getNumHorizontalTiles()`
- *getNumStrips*
`public int getNumStrips()`
- *getNumTiles*
`public int getNumTiles()`
- *getNumVerticalTiles*
`public int getNumVerticalTiles()`
- *getPalette*
`public net.sourceforge.jiu.data.Palette getPalette()`
- *getPhotometricInterpretation*
`public int getPhotometricInterpretation()`
- *getPredictor*
`public int getPredictor()`
- *getRowsPerStrip*
`public int getRowsPerStrip()`
- *getSamplesPerPixel*
`public int getSamplesPerPixel()`
- *getSoftware*
`public java.lang.String getSoftware()`
- *getStripOffsets*
`public java.util.Vector getStripOffsets()`
- *getT4Options*
`public int getT4Options()`

- *getT6Options*
public int **getT6Options**()
 - *getTileHeight*
public int **getTileHeight**()
 - *getTileOffset*
public long **getTileOffset**(int tileIndex)
 - *getTileWidth*
public int **getTileWidth**()
 - *getTileX1*
public int **getTileX1**(int tileIndex)
 - *getTileX2*
public int **getTileX2**(int tileIndex)
 - *getTileY1*
public int **getTileY1**(int tileIndex)
 - *getTileY2*
public int **getTileY2**(int tileIndex)
 - *getWidth*
public int **getWidth**()
 - *initFromTags*
public void **initFromTags**(boolean check) throws
net.sourceforge.jiu.codecs.InvalidFileStructureException,
net.sourceforge.jiu.codecs.UnsupportedTypeException
 - *initMembers*
public void **initMembers**()
 - *isGrayscale*
public boolean **isGrayscale**()
 - *isPaletted*
public boolean **isPaletted**()
 - *isStriped*
public boolean **isStriped**()
 - **Description**
Returns true if the image belonging to this IFD is stored as strips, false otherwise.
 - **See also**
* TIFFImageFileDirectory.isTiled() (in 3.2.9, page 162)
-
- *isTiled*
public boolean **isTiled**()
 - **Description**
Returns true if the image belonging to this IFD is stored as tiles, false otherwise.
 - **See also**
* TIFFImageFileDirectory.isStriped() (in 3.2.9, page 162)
-

- *setTimeZone*

```
public void setTimeZone( java.util.TimeZone tz )
```

- **Description**

Sets the time zone to be used when trying to interpret dates found in a (in 3.1.1, page 135)tag. Example call: `setTimeZone(TimeZone.getTimeZone("America/New_York"));`

- **Parameters**

- * `tz` – TimeZone object

3.2.10 Class *TIFFRational*

Data class to store a TIFF rational number. A TIFF rational number is a fraction given by 32 bit integer numerator and denominator values. It is one of the data types used in TIFF tags (in 3.2.11, page 166)). For more information on TIFF's internals, see (in 3.2.1, page 137), which lists a few links to TIFF specification documents.

Declaration

```
public class TIFFRational
extends java.lang.Object
```

Constructor summary

TIFFRational(int, int) Creates a TiffRational object from the arguments.

Method summary

getAsDouble() Returns the fraction as a double value.

getAsFloat() Returns the fraction as a float value.

getDenominator() Returns the denominator value that was given to the constructor.

getNumerator() Returns the numerator value that was given to the constructor.

Constructors

- *TIFFRational*

```
public TIFFRational( int numerator, int denominator )
```

- **Description**

Creates a TiffRational object from the arguments.

- **Parameters**

- * **numerator** – the numerator of the fraction stored in this object

- * **denominator** – the denominator of the fraction stored in this object

- **Throws**

- * **java.lang.IllegalArgumentException** – if denominator is 0 (division by zero is not allowed)

Methods

- *getAsDouble*

```
public double getAsDouble( )
```

- **Description**

Returns the fraction as a double value.

- **Returns** – the fraction stored in this object

- **See also**

- * **TIFFRational.getAsFloat()** (in 3.2.10, page 164)

- *getAsFloat*

```
public float getAsFloat( )
```

- **Description**

Returns the fraction as a `float` value.

- **Returns** – the fraction stored in this object

- **See also**

* `TIFFRational.getAsDouble()` (in 3.2.10, page 164)

- *getDenominator*

`public int getDenominator()`

- **Description**

Returns the denominator value that was given to the constructor.

- **Returns** – denominator value

- *getNumerator*

`public int getNumerator()`

- **Description**

Returns the numerator value that was given to the constructor.

- **Returns** – numerator value

3.2.11 Class **TIFFTag**

This encapsulates the data stored in a TIFF tag (a single image file directory entry). That includes the following items:

- identification number, 254 or higher; see the many TAG_xyz constants in (in 3.1.1, page 132) for a list of allowed values
- type; the allowed types include integer and floating point numbers, Strings etc.; see the TAG_TYPE_xyz constants in (in 3.1.1, page 132)
- count; the number of values of the given type that are stored in this tag; 1 or higher
- offset; if count is 1 and the type fits into four bytes, this is the complete value of this tag; otherwise, it is an offset into the file to the position that will hold the value(s)

See the TIFF specification manual linked in the description of (in 3.2.1, page 137) for more details.

See also

- **TIFFImageFileDirectory** (in 3.2.9, page 157)

Declaration

```
public class TIFFTag
  extends java.lang.Object
  implements TIFFConstants
```

Constructor summary

TIFFTag(int, int, int, int) Creates a new tag with the given ID, type, number of objects / primitives stored in it and offset value.

TIFFTag(int, int, int, int, Vector)

Method summary

getCount() Returns the number of items stored in this tag.

getElementAsInt(int) Returns an item stored in this tag an int value.

Returns the ID of this tag, which may be one of the TAG_xyz constants.

getObject(int) Returns an object from this tag's Vector of items, or null if no such Vector exists.

getOffset() Returns the offset value stored in this tag.

getString() If this tag has a Vector of items and if the first item is a String, that String is returned, null otherwise.

getType() Returns the type of this tag's content as a TAG_TYPE_xyz constant.

getVector() Returns the Vector encapsulating the items stored in this tag.

isInt() Returns if the value(s) stored in this tag are of type BYTE, SHORT or LONG.

setVector(Vector) If this tag encapsulates more than one item or a single item that does not fit into four bytes, this Vector will store all elements in it.

Constructors

- *TIFFTag*
`public TIFFTag(int id, int type, int count, int offset)`
 - **Description**
Creates a new tag with the given ID, type, number of objects / primitives stored in it and offset value.
- *TIFFTag*
`public TIFFTag(int id, int type, int count, int offset, java.util.Vector vector)`

Methods

- *getCount*
`public int getCount()`
 - **Description**
Returns the number of items stored in this tag.
- *getElementAsInt*
`public int getElementAsInt(int index)`
 - **Description**
Returns an item stored in this tag an `int` value.
 - **Parameters**
 - * `index` – zero-based index of the integer item to be returned
- *getId*
`public int getId()`
 - **Description**
Returns the ID of this tag, which may be one of the `TAG_xyz` constants.
- *getObject*
`public java.lang.Object getObject(int index)`
 - **Description**
Returns an object from this tag's Vector of items, or `null` if no such Vector exists.
- *getOffset*
`public int getOffset()`
 - **Description**
Returns the offset value stored in this tag.
- *getString*
`public java.lang.String getString()`

– **Description**

If this tag has a Vector of items and if the first item is a String, that String is returned, null otherwise.

• *getType*

public int **getType**()

– **Description**

Returns the type of this tag's content as a TAG_TYPE_xyz constant.

• *getVector*

public java.util.Vector **getVector**()

– **Description**

Returns the Vector encapsulating the items stored in this tag.

– **See also**

* TIFFTag.setVector(java.util.Vector) (in 3.2.11, page 168)

• *isInt*

public boolean **isInt**()

– **Description**

Returns if the value(s) stored in this tag are of type BYTE, SHORT or LONG. Note that BYTE and SHORT have the same meaning as in Java (one and two bytes large) while LONG is a 32-bit-value, just like int in Java.

– **Returns** – if this tag's contains integer values ≤ 32 bits

• *setVector*

public void **setVector**(java.util.Vector vector)

– **Description**

If this tag encapsulates more than one item or a single item that does not fit into four bytes, this Vector will store all elements in it. The size() method called on that Vector object returns the same value as getCount().

– **Parameters**

* **vector** – the Vector with the items to be encapsulated by this tag

– **See also**

* TIFFTag.getVector() (in 3.2.11, page 168)

Chapter 4

Package net.sourceforge.jiu.color

Package Contents

Page

Interfaces

YCbCrIndex	170
<i>This interface simply provides three integer constants as index values for the three channels of an YCbCr image: gray, blue chrominance and red chrominance.</i>	

Classes

Invert	171
<i>Creates an inverted (negated) version of an image.</i>	
WebsafePaletteCreator	172
<i>This class creates (in 14.2.9, page 363) objects that contain the so-called web-safe palette.</i>	

Contains color-related operations that did not fit into one of the subpackages.

4.1 Interfaces

4.1.1 Interface YCbCrIndex

This interface simply provides three integer constants as index values for the three channels of an YCbCr image: gray, blue chrominance and red chrominance. The three values are guaranteed to lie in the interval 0 to 2. Furthermore, all three values are different from each other, so that the complete interval from 0 to 2 is used.

See also

- `net.sourceforge.jiu.data.RGBIndex` (in 14.1.15, page 334)

Declaration

```
public interface YCbCrIndex
```

All known subclasses

PCDCodec (in 2.1.8, page 105), PCDYCbCrConversion (in 8.1.3, page 227)

All classes known to implement interface

PCDCodec (in 2.1.8, page 105), PCDYCbCrConversion (in 8.1.3, page 227)

Field summary

INDEX_CB Index value for the blue chrominance component.
INDEX_CR Index value for the red chrominance component.
INDEX_Y Index value for the luminance (gray) component.

Fields

- `int INDEX_Y`
 - Index value for the luminance (gray) component.
- `int INDEX_CB`
 - Index value for the blue chrominance component.
- `int INDEX_CR`
 - Index value for the red chrominance component.

4.2 Classes

4.2.1 Class Invert

Creates an inverted (negated) version of an image. This is done by subtracting each sample value of a channel from the maximum sample for that channel. The maximum sample for a channel is given by (in 14.1.7, page 323). For paletted images, just the palette is treated that way. Supported image types: (in 14.1.7, page 322). Input and output image can be the same object.

Declaration

```
public class Invert
extends net.sourceforge.jiu.ops.ImageToImageOperation (in 19.2.3, page 501)
```

Constructor summary

Invert()

Method summary

process() Inverts the input image, reusing an output image if one has been specified.

Constructors

- *Invert*
public **Invert**()

Methods

- *process*
public void **process**() throws
net.sourceforge.jiu.ops.MissingParameterException,
net.sourceforge.jiu.ops.WrongParameterException
 - **Description**
Inverts the input image, reusing an output image if one has been specified. For paletted images, inverts the palette. For all other types, subtracts each sample of each channel from the maximum value of that channel.
 - **Throws**
 - * net.sourceforge.jiu.ops.MissingParameterException – if the input image is missing
 - * net.sourceforge.jiu.ops.WrongParameterException – if any of the specified image parameters are unsupported or of the wrong width or height

4.2.2 Class WebsafePaletteCreator

This class creates (in 14.2.9, page 363) objects that contain the so-called websafe palette. This palette has 216 entries which are uniformly spread over the RGB color cube. Each component (red / green / blue) takes each of the six values 0, 51, 101, 153, 204 and 255 (note that the difference is almost equal between two consecutive values, between 50 and 52). Therefore, the palette will have $6^3 = 6 * 6 * 6 = 216$ entries.

This palette was designed with computer systems in mind that can only display 256 colors at a time. With the 216 colors that are uniformly spread over RGB color space, there is at least a somewhat similar match for each possible input color.

Declaration

```
public class WebsafePaletteCreator
    extends java.lang.Object
    implements net.sourceforge.jiu.data.RGBIndex
```

Method summary

create() Creates a new palette with the 216 websafe colors.

Methods

- *create*
`public static net.sourceforge.jiu.data.Palette create()`
 - **Description**
Creates a new palette with the 216 websafe colors.
 - **Returns** – new palette object

Chapter 5

Package net.sourceforge.jiu.color.data

Package Contents

Page

Interfaces

CoOccurrenceFrequencyMatrix	175
<i>An interface for a co-occurrence frequency matrix.</i>	
CoOccurrenceMatrix	178
<i>An interface for co-occurrence matrices.</i>	
Histogram1D	180
<i>An interface for a one-dimensional histogram.</i>	
Histogram3D	182
<i>An interface for classes that store three-dimensional histograms.</i>	

Classes

ArrayHistogram1D	184
<i>A one-dimensional histogram data class that stores its counters in memory.</i>	
BaseCoOccurrenceFrequencyMatrix	186
<i>This abstract class encapsulates all data of a co-occurrence frequency matrix except for the frequency values.</i>	
MemoryCoOccurrenceFrequencyMatrix	188
<i>Implements the (in 5.1.1, page 175) interface by using a large array of values in memory.</i>	
MemoryCoOccurrenceMatrix	190
<i>This class stores a co-occurrence matrix, a two-dimensional array of int counters.</i>	
NaiveHistogram3D	192
<i>A class for a three-dimensional histogram that allocates one int value per counter at construction time.</i>	
OnDemandHistogram3D	195
<i>A data class for a three-dimensional histogram, creating counters on demand only, not allocating counters for all possible entries at the beginning.</i>	

Provides classes to store images and data directly related to them.

Package Specification

The base interface for image data in JIU is (in 14.1.11, page 328). The concept of a pixel image includes the following properties:

- The image data is arranged as a rectangular grid of pixels. The image has a fixed number of columns (width) and rows (height). Width and height can be queried using the method `getWidth` and `getHeight`.
- Each pixel is made up of one or more samples. The exact number of samples per pixel equals the number of channels. Nothing is said about the nature of the samples (e. g., their type) and no ways to access samples or pixels are provided in the `PixelImage` interface.
- A class implementing `PixelImage` must provide the `createCompatibleImage` method which gets a width and a height value as parameters and must return a new object of the same class with that pixel resolution.
- Helper methods in `PixelImage` include ways to get a textual description of the type and the number of bytes allocated for a specific object implementing `PixelImage`.

The interface (in 14.1.7, page 322) extends the (in 14.1.11, page 328) interface. All sample values belonging to an object of a class implementing `IntegerImage` are supposed to be integer values that can be stored in an `int` value (a signed 32 bit value).

Related Documentation

5.1 Interfaces

5.1.1 Interface CoOccurrenceFrequencyMatrix

An interface for a co-occurrence frequency matrix. Also provides access to some statistical data. This class is not a pure data type for it also demands a method (in 5.1.1, page 175) which takes the matrix coefficients and computes mean, standard deviation and other properties from it.

Declaration

```
public interface CoOccurrenceFrequencyMatrix
```

All known subclasses

MemoryCoOccurrenceFrequencyMatrix (in 5.2.3, page 188), BaseCoOccurrenceFrequencyMatrix (in 5.2.2, page 186)

All classes known to implement interface

BaseCoOccurrenceFrequencyMatrix (in 5.2.2, page 186)

Method summary

clear() Sets all frequency values in this matrix to 0.0.
computeStatistics() Computes mean, standard deviation and the sum of those two so that these values can be queried by the appropriate get methods.
getDimension() Returns the dimension of this matrix.
getMean(int) Returns the mean for all pairs (index, i), with i running from 0 to (in 5.1.1, page 176)- 1.
getScofMean() Returns the sum of mean and standard deviation for all pairs (index, x), with x running from 0 to getDimension() - 1.
getScofStddev() Returns the standard deviation for all pairs (i, i), with i running from 0 to getDimension() - 1.
getScofSum()
getStddev(int) Returns the standard deviation of the values getValue(index, i) with i running from 0 to (in 5.1.1, page 176)- 1.
getValue(int) Returns the value for the self co-occurrence frequency of i (i being from 0 to (in 5.1.1, page 176)- 1).
getValue(int, int)
setValue(int, int, double)

Methods

- *clear*
 void **clear**()
 – **Description**
 Sets all frequency values in this matrix to 0.0.
- *computeStatistics*
 void **computeStatistics**()

– **Description**

Computes mean, standard deviation and the sum of those two so that these values can be queried by the appropriate get methods.

- *getDimension*

`int getDimension()`

– **Description**

Returns the dimension of this matrix.

- *getMean*

`double getMean(int index)`

– **Description**

Returns the mean for all pairs (index, i), with i running from 0 to (in 5.1.1, page 176)- 1.

- *getScofMean*

`double getScofMean()`

– **Description**

Returns the sum of mean and standard deviation for all pairs (index, x), with x running from 0 to getDimension() - 1. The result is equal to (in 5.1.1, page 176)+ (in 5.1.1, page 176)

- *getScofStddev*

`double getScofStddev()`

– **Description**

Returns the standard deviation for all pairs (i, i), with i running from 0 to getDimension() - 1.

– **Returns** – standard deviation for pairs

- *getScofSum*

`double getScofSum()`

- *getStddev*

`double getStddev(int index)`

– **Description**

Returns the standard deviation of the values getValue(index, i) with i running from 0 to (in 5.1.1, page 176)- 1.

– **Parameters**

* **index** – first argument to all calls of getValue used to determine the standard deviation

- *getValue*

`double getValue(int i)`

– **Description**

Returns the value for the self co-occurrence frequency of i (i being from 0 to (in 5.1.1, page 176)- 1). The result is the same as a call to getValue(i, i).

– **Parameters**

* **i** – index into the matrix, must be larger than or equal to 0 and smaller than (in 5.1.1, page 176)

-
- *getValue*
double **getValue**(**int** **i**, **int** **j**)
 - *setValue*
void **setValue**(**int** **i**, **int** **j**, **double** **newValue**)

5.1.2 Interface CoOccurrenceMatrix

An interface for co-occurrence matrices. An implementing class stores `int` counter values for pairs of pixels. These counters represent the number of times two pixels are direct neighbors in an image.

Declaration

```
public interface CoOccurrenceMatrix
```

All known subclasses

MemoryCoOccurrenceMatrix (in 5.2.4, page 190)

All classes known to implement interface

MemoryCoOccurrenceMatrix (in 5.2.4, page 190)

Method summary

clear() Sets all counters to zero.
getDimension() Returns the dimension of this matrix.
getValue(int, int) Returns the matrix value at a given position.
incValue(int, int) Increases the counter for pair (i, j) by one.
setValue(int, int, int) Sets the counter for pair (i, j) to a new value.

Methods

- *clear*
void clear()
 - **Description**
Sets all counters to zero.
- *getDimension*
int getDimension()
 - **Description**
Returns the dimension of this matrix. This is the number of rows and columns.
 - **Returns** – matrix dimension (larger than zero)
- *getValue*
int getValue(int i, int j)
 - **Description**
Returns the matrix value at a given position.
 - **Parameters**
 - * **i** – column index, from 0 to (in 5.1.2, page 178)- 1
 - * **j** – row index, from 0 to (in 5.1.2, page 178)- 1
 - **Throws**
 - * **java.lang.IllegalArgumentException** – for invalid index pairs (i, j)

- *incValue*

```
void incValue( int i, int j )
```

- **Description**

Increases the counter for pair (i, j) by one. This method can be implemented by the call `setValue(i, j, getValue(i, j) + 1);`.

- **Parameters**

- * i – column index, from 0 to (in 5.1.2, page 178)- 1
 - * j – row index, from 0 to (in 5.1.2, page 178)- 1

- **Throws**

- * `java.lang.IllegalArgumentException` – for invalid index pairs (i, j)
-

- *setValue*

```
void setValue( int i, int j, int newValue )
```

- **Description**

Sets the counter for pair (i, j) to a new value.

- **Parameters**

- * i – column index, from 0 to (in 5.1.2, page 178)- 1
 - * j – row index, from 0 to (in 5.1.2, page 178)- 1

- **Throws**

- * `java.lang.IllegalArgumentException` – for invalid index pairs (i, j)

5.1.3 Interface Histogram1D

An interface for a one-dimensional histogram.

See also

- Histogram3D (in 5.1.4, page 182)

Declaration

```
public interface Histogram1D
```

All known subclasses

ArrayHistogram1D (in 5.2.1, page 184)

All classes known to implement interface

ArrayHistogram1D (in 5.2.1, page 184)

Method summary

- clear()** Sets all counters to zero.
- getEntry(int)** Returns the counter value for the given index.
- getMaxValue()** Returns the maximum allowed index.
- getNumUsedEntries()** Returns the number of used entries (those entries with a counter value larger than zero).
- increaseEntry(int)** Increases the counter value of the given index by one.
- setEntry(int, int)** Sets one counter to a new value.

Methods

- *clear*
void **clear**()
 - **Description**
Sets all counters to zero.
- *getEntry*
int **getEntry**(int index)
 - **Description**
Returns the counter value for the given index.
 - **Parameters**
* index – the zero-based index of the desired counter value
 - **Returns** – the counter value
 - **Throws**
* java.lang.IllegalArgumentException – if the argument is not a valid index
- *getMaxValue*
int **getMaxValue**()

- **Description**

Returns the maximum allowed index. The minimum is always 0.

- **Returns** – the maximum index value

- *getNumUsedEntries*

int **getNumUsedEntries**()

- **Description**

Returns the number of used entries (those entries with a counter value larger than zero).

- **Returns** – number of non-zero counter values

- *increaseEntry*

void **increaseEntry**(**int** **index**)

- **Description**

Increases the counter value of the given index by one. Same semantics as `setEntry(index, getEntry(index) + 1);`

- **Parameters**

* **index** – index into the histogram

- **Returns** – the counter value of the given index

- **Throws**

* `java.lang.IllegalArgumentException` – if the argument index is invalid

- *setEntry*

void **setEntry**(**int** **index**, **int** **newValue**)

- **Description**

Sets one counter to a new value.

- **Parameters**

* **index** – index of the counter to be changed

* **newValue** – new value for that counter

- **Throws**

* `java.lang.IllegalArgumentException` – if the index is invalid

5.1.4 Interface Histogram3D

An interface for classes that store three-dimensional histograms. Histograms count the occurrence of values, so a three-dimensional histogram has counters for three-dimensional values. The 3D histograms are used (as an example) to count the occurrence of each color in an RGB image.

See also

- Histogram1D (in 5.1.3, page 180)

Declaration

```
public interface Histogram3D
```

All known subclasses

OnDemandHistogram3D (in 5.2.6, page 195), NaiveHistogram3D (in 5.2.5, page 192)

All classes known to implement interface

OnDemandHistogram3D (in 5.2.6, page 195), NaiveHistogram3D (in 5.2.5, page 192)

Method summary

clear() Sets all counters to zero.
getEntry(int, int, int) Returns the counter value of (index1, index2, index3).
getMaxValue(int) Returns the maximum index value for one of the three indexes.
getNumUsedEntries() Returns the number of used entries (those entries with a counter value larger than zero).
increaseEntry(int, int, int) Increases the counter value of (index1, index2, index3) by one.
setEntry(int, int, int, int) Sets the counter value of (index1, index2, index3) to newValue.

Methods

- *clear*
 void **clear**()
 – **Description**
 Sets all counters to zero.
- *getEntry*
 int **getEntry**(int index1, int index2, int index3)
 – **Description**
 Returns the counter value of (index1, index2, index3).
 – **Parameters**
 - * **index1** – first of the three values forming the threedimensional index
 - * **index2** – second of the three values forming the threedimensional index
 - * **index3** – three of the three values forming the threedimensional index

- **Returns** – the counter value of the desired index
- **Throws**
 - * `java.lang.IllegalArgumentException` – if the index formed by the arguments is invalid

- *getMaxValue*

`int getMaxValue(int index)`

- **Description**
Returns the maximum index value for one of the three indexes.
- **Throws**
 - * `java.lang.IllegalArgumentException` – if the index formed by the arguments is invalid

- *getNumUsedEntries*

`int getNumUsedEntries()`

- **Description**
Returns the number of used entries (those entries with a counter value larger than zero).
- **Returns** – number of non-zero counter values

- *increaseEntry*

`void increaseEntry(int index1, int index2, int index3)`

- **Description**
Increases the counter value of (index1, index2, index3) by one. This method can easily be implemented by the one-liner `setEntry(index1, index2, index3, getEntry(index1, index2, index3) + 1)`; However, this method is expected to be faster in some contexts.
- **Parameters**
 - * `index1` – first of the three values forming the three-dimensional index
 - * `index2` – second of the three values forming the three-dimensional index
 - * `index3` – three of the three values forming the three-dimensional index
- **Returns** – the counter value of the desired index
- **Throws**
 - * `java.lang.IllegalArgumentException` – if the index formed by the arguments is invalid

- *setEntry*

`void setEntry(int index1, int index2, int index3, int newValue)`

- **Description**
Sets the counter value of (index1, index2, index3) to newValue.
- **Parameters**
 - * `index1` – first of the three values forming the three-dimensional index
 - * `index2` – second of the three values forming the three-dimensional index
 - * `index3` – three of the three values forming the three-dimensional index
 - * `newValue` – the counter value that is assigned to the argument index
- **Throws**
 - * `java.lang.IllegalArgumentException` – if the index formed by the first three arguments is invalid

5.2 Classes

5.2.1 Class ArrayHistogram1D

A one-dimensional histogram data class that stores its counters in memory. Counters are stored in an `int` array of length (in 5.2.1, page 185)+ 1 so that `k` values will require `k * 4` bytes.

Declaration

```
public class ArrayHistogram1D
  extends java.lang.Object
  implements Histogram1D
```

Constructor summary

ArrayHistogram1D(int) Creates a histogram with the argument's number of values, from 0 to `numValues - 1`.

Method summary

```
clear()
getEntry(int)
getMaxValue()
getNumUsedEntries()
increaseEntry(int)
setEntry(int, int)
```

Constructors

- *ArrayHistogram1D*
public ArrayHistogram1D(int numValues)
 - **Description**
 Creates a histogram with the argument's number of values, from 0 to `numValues - 1`.
 - **Parameters**
 * `numValues` – the number of counters in the histogram; must be one or larger
 - **Throws**
 * `java.lang.IllegalArgumentException` – if the argument is smaller than one

Methods

- *clear*
void clear()
 - **Description copied from Histogram1D (in 5.1.3, page 180)**
 Sets all counters to zero.
- *getEntry*
int getEntry(int index)

- **Description copied from Histogram1D (in 5.1.3, page 180)**
Returns the counter value for the given index.
 - **Parameters**
 - * **index** – the zero-based index of the desired counter value
 - **Returns** – the counter value
 - **Throws**
 - * `java.lang.IllegalArgumentException` – if the argument is not a valid index
-

- *getMaxValue*

`int getMaxValue()`

- **Description copied from Histogram1D (in 5.1.3, page 180)**
Returns the maximum allowed index. The minimum is always 0.
 - **Returns** – the maximum index value
-

- *getNumUsedEntries*

`int getNumUsedEntries()`

- **Description copied from Histogram1D (in 5.1.3, page 180)**
Returns the number of used entries (those entries with a counter value larger than zero).
 - **Returns** – number of non-zero counter values
-

- *increaseEntry*

`void increaseEntry(int index)`

- **Description copied from Histogram1D (in 5.1.3, page 180)**
Increases the counter value of the given index by one. Same semantics as `setEntry(index, getEntry(index) + 1);`
 - **Parameters**
 - * **index** – index into the histogram
 - **Returns** – the counter value of the given index
 - **Throws**
 - * `java.lang.IllegalArgumentException` – if the argument index is invalid
-

- *setEntry*

`void setEntry(int index, int newValue)`

- **Description copied from Histogram1D (in 5.1.3, page 180)**
Sets one counter to a new value.
- **Parameters**
 - * **index** – index of the counter to be changed
 - * **newValue** – new value for that counter
- **Throws**
 - * `java.lang.IllegalArgumentException` – if the index is invalid

5.2.2 Class BaseCoOccurrenceFrequencyMatrix

This abstract class encapsulates all data of a co-occurrence frequency matrix except for the frequency values. The method `computeStatistics` is implemented. Any class extending this class only has to deal with storing the frequency values (in 5.2.3, page 188) does this by using a one-dimensional array internally).

Declaration

```
public abstract class BaseCoOccurrenceFrequencyMatrix
extends java.lang.Object
implements CoOccurrenceFrequencyMatrix
```

All known subclasses

MemoryCoOccurrenceFrequencyMatrix (in 5.2.3, page 188)

Constructor summary

BaseCoOccurrenceFrequencyMatrix()

Method summary

computeStatistics() Assumes that the co-occurrence frequency values have been initialized.

getMean(int) Returns the mean of the co-occurrence frequency values.

getScofMean() Returns the mean of all self co-occurrence frequency values.

getScofStddev() Returns the standard deviation of all self co-occurrence frequency values.

getScofSum() Return the sum of mean and standard deviation of the self co-occurrence frequency values.

getStddev(int)

Constructors

- *BaseCoOccurrenceFrequencyMatrix*
public **BaseCoOccurrenceFrequencyMatrix**()

Methods

- *computeStatistics*
public void **computeStatistics**()
 - **Description**
Assumes that the co-occurrence frequency values have been initialized. Computes mean and standard deviation for co-occurrence and self co-occurrence frequency values.
- *getMean*
public double **getMean**(int index)

– **Description**

Returns the mean of the co-occurrence frequency values.

• *getScofMean*

`public double getScofMean()`

– **Description**

Returns the mean of all self co-occurrence frequency values. This value is called μ_S in Shufelt’s paper. This value is determined once within `computeStatistics()`.

• *getScofStddev*

`public double getScofStddev()`

– **Description**

Returns the standard deviation of all self co-occurrence frequency values. This value is called σ_S in Shufelt’s paper. This value is determined once within a call to `computeStatistics()`.

• *getScofSum*

`public double getScofSum()`

– **Description**

Return the sum of mean and standard deviation of the self co-occurrence frequency values. Assumes that `computeStatistics()` has been called already.

– **Returns** – sum of mean and standard deviation of the self co-occurrence frequency values

• *getStddev*

`double getStddev(int index)`

– **Description copied from CoOccurrenceFrequencyMatrix (in 5.1.1, page 175)**

Returns the standard deviation of the values `getValue(index, i)` with `i` running from 0 to `index - 1`.

– **Parameters**

* `index` – first argument to all calls of `getValue` used to determine the standard deviation

5.2.3 Class MemoryCoOccurrenceFrequencyMatrix

Implements the (in 5.1.1, page 175)interface by using a large array of values in memory.

Declaration

```
public class MemoryCoOccurrenceFrequencyMatrix
extends net.sourceforge.jiu.color.data.BaseCoOccurrenceFrequencyMatrix (in 5.2.2, page 186)
```

Constructor summary

MemoryCoOccurrenceFrequencyMatrix(int) Creates a co-occurrence frequency matrix of given dimension; allocates dimension times dimension double values for internal array; does not call clear() to set everything to zero, must be done by user (or automatically in init).

Method summary

clear() Sets all values of this matrix to zero.
getDimension()
getValue(int) Returns the value of this matrix at row i, column i.
getValue(int, int) Returns the value of this matrix at row j, column i.
setValue(int, int, double) Sets value at row j, column i to newValue.

Constructors

- *MemoryCoOccurrenceFrequencyMatrix*
 public **MemoryCoOccurrenceFrequencyMatrix(int dimension)**
 - **Description**
 Creates a co-occurrence frequency matrix of given dimension; allocates dimension times dimension double values for internal array; does not call clear() to set everything to zero, must be done by user (or automatically in init). Dimension should be number of colors in palette.
 - **Throws**
 * java.lang.IllegalArgumentException – if dimension is smaller than 1

Methods

- *clear*
 public void **clear()**
 - **Description**
 Sets all values of this matrix to zero.
- *getDimension*
 public int **getDimension()**
- *getValue*
 public double **getValue(int i)** throws java.lang.IllegalArgumentException

– **Description**

Returns the value of this matrix at row *i*, column *i*. Argument is zero-based, so make sure that $0 \leq i < \text{getDimension}()$. Other values will raise an `IllegalArgumentException`. Simply calls `getValue(i, i)`.

• *getValue*

`public double getValue(int i, int j) throws
java.lang.IllegalArgumentException`

– **Description**

Returns the value of this matrix at row *j*, column *i*. Both arguments are zero-based, so make sure that $0 \leq i, j < \text{getDimension}()$. Other values will raise an `IllegalArgumentException`.

• *setValue*

`public void setValue(int i, int j, double newValue) throws
java.lang.IllegalArgumentException`

– **Description**

Sets value at row *j*, column *i* to `newValue`. Both arguments are zero-based, so make sure that $0 \leq i, j < \text{getDimension}()$. Other values will raise an `IllegalArgumentException`.

5.2.4 Class MemoryCoOccurrenceMatrix

This class stores a co-occurrence matrix, a two-dimensional array of int counters. The dimension is given to the constructor which allocates a corresponding array.

Declaration

```
public class MemoryCoOccurrenceMatrix
  extends java.lang.Object
  implements CoOccurrenceMatrix
```

Constructor summary

MemoryCoOccurrenceMatrix(int) Creates a new matrix that stores dimension times dimension int values in memory.

Method summary

```
clear()
getDimension()
getValue(int, int)
incValue(int, int)
setValue(int, int, int)
```

Constructors

- *MemoryCoOccurrenceMatrix*
 public **MemoryCoOccurrenceMatrix(int dimension)**
 - **Description**
 Creates a new matrix that stores dimension times dimension int values in memory. Given that array index values are of type int, this limits dimension to about 46000 (sqrt(Integer.MAX_VALUE)). In practice, dimension leads to dimension times dimension times 4 bytes being allocated, so that memory available to the JVM may become a decisive factor.
 - **Parameters**
 * **dimension** – the matrix' dimension, which is both the number of rows and columns

Methods

- *clear*
 void **clear()**
 - **Description copied from CoOccurrenceMatrix (in 5.1.2, page 178)**
 Sets all counters to zero.
- *getDimension*
 int **getDimension()**

- **Description copied from CoOccurrenceMatrix (in 5.1.2, page 178)**
Returns the dimension of this matrix. This is the number of rows and columns.
 - **Returns** – matrix dimension (larger than zero)
-

- *getValue*

int getValue(int i, int j)

- **Description copied from CoOccurrenceMatrix (in 5.1.2, page 178)**
Returns the matrix value at a given position.
 - **Parameters**
 - * i – column index, from 0 to (in 5.1.2, page 178)- 1
 - * j – row index, from 0 to (in 5.1.2, page 178)- 1
 - **Throws**
 - * `java.lang.IllegalArgumentException` – for invalid index pairs (i, j)
-

- *incValue*

void incValue(int i, int j)

- **Description copied from CoOccurrenceMatrix (in 5.1.2, page 178)**
Increases the counter for pair (i, j) by one. This method can be implemented by the call `setValue(i, j, getValue(i, j) + 1);`.
 - **Parameters**
 - * i – column index, from 0 to (in 5.1.2, page 178)- 1
 - * j – row index, from 0 to (in 5.1.2, page 178)- 1
 - **Throws**
 - * `java.lang.IllegalArgumentException` – for invalid index pairs (i, j)
-

- *setValue*

void setValue(int i, int j, int newValue)

- **Description copied from CoOccurrenceMatrix (in 5.1.2, page 178)**
Sets the counter for pair (i, j) to a new value.
- **Parameters**
 - * i – column index, from 0 to (in 5.1.2, page 178)- 1
 - * j – row index, from 0 to (in 5.1.2, page 178)- 1
- **Throws**
 - * `java.lang.IllegalArgumentException` – for invalid index pairs (i, j)

5.2.5 Class NaiveHistogram3D

A class for a three-dimensional histogram that allocates one `int` value per counter at construction time. This means that a histogram with 8 bits for each channel will have $2^{8+8+8} = 2^{24} = 16,777,216$ `int` values, making it 64 MB large.

Declaration

```
public class NaiveHistogram3D
  extends java.lang.Object
  implements Histogram3D
```

Constructor summary

NaiveHistogram3D(int) Creates a histogram with the same number of values for all three dimensions.

NaiveHistogram3D(int, int, int) Creates a histogram

Method summary

clear() Sets all counters to zero.

getEntry(int, int, int) Returns the counter value of (index1, index2, index3).

getMaxValue(int)

getNumUsedEntries() Returns the number of used entries (those entries with a counter value larger than zero).

increaseEntry(int, int, int) Increases the counter value of (index1, index2, index3) by one.

setEntry(int, int, int, int) Sets the counter value of (index1, index2, index3) to newValue.

Constructors

- *NaiveHistogram3D*

```
public NaiveHistogram3D( int numValues ) throws
  java.lang.IllegalArgumentException, java.lang.OutOfMemoryError
```

- **Description**

Creates a histogram with the same number of values for all three dimensions. Calls (int, int, int) (in 5.2.5, page 192), repeating numValues three times.

- **Parameters**

* numValues – the number of levels for all three dimensions

- *NaiveHistogram3D*

```
public NaiveHistogram3D( int numValuesLevel1, int numValuesLevel2, int
  numValuesLevel3 ) throws java.lang.IllegalArgumentException,
  java.lang.OutOfMemoryError
```

- **Description**

Creates a histogram

Methods

- *clear*

`public void clear()`

- **Description**

Sets all counters to zero.

- *getEntry*

`public int getEntry(int index1, int index2, int index3)` throws
`java.lang.IllegalArgumentException`

- **Description**

Returns the counter value of (index1, index2, index3).

- **Parameters**

- * `index1` – first of the three values forming the three-dimensional index
- * `index2` – second of the three values forming the three-dimensional index
- * `index3` – three of the three values forming the three-dimensional index

- **Returns** – the counter value of the desired index

- **Throws**

- * `java.lang.IllegalArgumentException` – could be (read: need not necessarily) be thrown if the index formed by the arguments is invalid
-

- *getMaxValue*

`int getMaxValue(int index)`

- **Description copied from Histogram3D (in 5.1.4, page 182)**

Returns the maximum index value for one of the three indexes.

- **Throws**

- * `java.lang.IllegalArgumentException` – if the index formed by the arguments is invalid
-

- *getNumUsedEntries*

`public int getNumUsedEntries()`

- **Description**

Returns the number of used entries (those entries with a counter value larger than zero).

- **Returns** – number of non-zero counter values

- *increaseEntry*

`public void increaseEntry(int index1, int index2, int index3)` throws
`java.lang.IllegalArgumentException`

- **Description**

Increases the counter value of (index1, index2, index3) by one. This method can easily be implemented by the one-liner `setEntry(index1, index2, index3, getEntry(index1, index2, index3) + 1)`; However, this method is expected to be faster in some contexts.

- **Parameters**

- * `index1` – first of the three values forming the three-dimensional index
- * `index2` – second of the three values forming the three-dimensional index

- * **index3** – three of the three values forming the three-dimensional index
 - **Returns** – the counter value of the desired index
 - **Throws**
 - * `java.lang.IllegalArgumentException` – could be (read: need not necessarily) be thrown if the index formed by the arguments is invalid
-

- *setEntry*

```
public void setEntry( int index1, int index2, int index3, int newValue )
throws java.lang.IllegalArgumentException
```

- **Description**

Sets the counter value of (index1, index2, index3) to newValue.
- **Parameters**
 - * **index1** – first of the three values forming the three-dimensional index
 - * **index2** – second of the three values forming the three-dimensional index
 - * **index3** – three of the three values forming the three-dimensional index
 - * **newValue** – the counter value that is assigned to the argument index
- **Throws**
 - * `java.lang.IllegalArgumentException` – could be (read: need not necessarily) be thrown if the index formed by the first three arguments is invalid

5.2.6 Class OnDemandHistogram3D

A data class for a three-dimensional histogram, creating counters on demand only, not allocating counters for all possible entries at the beginning. The creation on demand happens to save space. A naive implementation can become huge - for eight bits per component, you'd need $2^{(8+8+8)} = 2^{24} = 16,777,216$ int values (64 MB), while a typical 24 bit image uses only a fraction of these possible colors.

Declaration

```
public class OnDemandHistogram3D
  extends java.lang.Object
  implements Histogram3D
```

Constructor summary

- OnDemandHistogram3D(int)** Creates a new object of this class with the argument as maximum value for all three index positions and the component values 0, 1, 2.
- OnDemandHistogram3D(int, int, int)** Creates a new object of this class with maximum values as specified by the arguments and the component values 0, 1, 2.
- OnDemandHistogram3D(int, int, int, int, int, int)** Creates a new object of this class with specified maximum values for all three indexes.

Method summary

- clear()** Resets all counters to zero.
- getEntry(int, int, int)** Returns counter for the argument color given by its red, green and blue intensity.
- getMaxValue(int)**
- getNumUsedEntries()** Returns the number of entries in this histogram with a counter value of one or higher (in other words: the number of colors that are in use).
- increaseEntry(int, int, int)** Increases the counter for the color given by the arguments red, green and blue.
- setEntry(int, int, int, int)** Sets one counter to a new value.

Constructors

- *OnDemandHistogram3D*
 public **OnDemandHistogram3D**(int **maxValue**)
 - **Description**
 Creates a new object of this class with the argument as maximum value for all three index positions and the component values 0, 1, 2. Simply calls **this(maxValue, maxValue, maxValue)**;
 - **Parameters**
 * **maxValue** – the maximum value for all indexes
- *OnDemandHistogram3D*
 public **OnDemandHistogram3D**(int **maxValue1**, int **maxValue2**, int **maxValue3**)

- **Description**

Creates a new object of this class with maximum values as specified by the arguments and the component values 0, 1, 2. Simply calls `this(maxValue1, maxValue2, maxValue3, 0, 1, 2);`

- **Parameters**

- * `maxValue1` – the maximum value for the first index
- * `maxValue2` – the maximum value for the second index
- * `maxValue3` – the maximum value for the third index

- *OnDemandHistogram3D*

`public OnDemandHistogram3D(int maxValue1, int maxValue2, int maxValue3, int c1, int c2, int c3) throws java.lang.IllegalArgumentException`

- **Description**

Creates a new object of this class with specified maximum values for all three indexes.

- **Parameters**

- * `maxValue1` – the maximum value for the first index
- * `maxValue2` – the maximum value for the second index
- * `maxValue3` – the maximum value for the third index
- * `c1` – the top level component for the internal tree
- * `c2` – the second-level component for the internal tree
- * `c3` – the third-level component for the internal tree

Methods

- *clear*

`public void clear()`

- **Description**

Resets all counters to zero. As the Java VM's garbage collector is responsible for releasing unused memory, it is unclear when memory allocated for the histogram will be available again.

- *getEntry*

`public int getEntry(int index1, int index2, int index3) throws java.lang.IllegalArgumentException`

- **Description**

Returns counter for the argument color given by its red, green and blue intensity.

- **Parameters**

- * `index1` – the first value of the index
- * `index2` –
- * `index3` –

- **Throws**

- * `java.lang.IllegalArgumentException` – this exception is thrown if the argument color is not in the valid interval, i.e., if at least one of the components is not from 0 .. max

- *getMaxValue*

`int getMaxValue(int index)`

- **Description copied from Histogram3D (in 5.1.4, page 182)**

Returns the maximum index value for one of the three indexes.

- **Throws**

* `java.lang.IllegalArgumentException` – if the index formed by the arguments is invalid

- *getNumUsedEntries*

`public int getNumUsedEntries()`

- **Description**

Returns the number of entries in this histogram with a counter value of one or higher (in other words: the number of colors that are in use).

- **Returns** – the number of unique colors

- *increaseEntry*

`public void increaseEntry(int red, int green, int blue) throws
java.lang.IllegalArgumentException`

- **Description**

Increases the counter for the color given by the arguments red, green and blue. This method could be implemented by the following code snippet:

```
setEntry(red, green, blue, getEntry(red, green, blue) + 1);
```

However, this is not done to avoid slow-downs by accessing a counter value twice.

- **Parameters**

* **red** – the red intensity of the color entry whose counter will be increased
 * **green** – the green intensity of the color entry whose counter will be increased
 * **blue** – the blue intensity of the color entry whose counter will be increased

- **Throws**

* `java.lang.IllegalArgumentException` – if the argument color is not valid (minimum is 0, maximum is defined for each component independently)

- *setEntry*

`public void setEntry(int r, int g, int b, int newValue) throws
java.lang.IllegalArgumentException`

- **Description**

Sets one counter to a new value.

- **Parameters**

* **r** – red component
 * **g** – green component
 * **b** – blue component
 * **newValue** – the new value for the counter

- **Throws**

* `java.lang.IllegalArgumentException` – if the components do not form a valid color

Chapter 6

Package

net.sourceforge.jiu.color.adjustment

Package Contents

Page

Classes

Brightness	199
<i>Adjusts the brightness of an image.</i>	
Contrast	201
<i>Adjusts the contrast of an image.</i>	
EqualizeHistogram	203
<i>Equalize the image using histogram information separately for each channel.</i>	
GammaCorrection	204
<i>Corrects the gamma of an image.</i>	
HueSaturationValue	206
<i>Adjusts saturation and value of a color image, optionally hue as well.</i>	
NormalizeHistogram	208
<i>Normalize the image using histogram information, separately for each channel.</i>	

Contains operations that modify pixel colors independent from other pixels.

6.1 Classes

6.1.1 Class Brightness

Adjusts the brightness of an image. The amount of adjustment is given to the constructor as a percentage value between -100 and 100. -100 will make the resulting image black, 0 will leave it unchanged, 100 will make it white.

Usage example

This code snippet will increase `image`'s brightness by 30 percent.

```
Brightness brightness = new Brightness();
brightness.setInputImage(image);
brightness.setBrightness(30);
brightness.process();
PixelImage adjustedImage = brightness.getOutputImage();
```

Declaration

```
public class Brightness
extends net.sourceforge.jiu.ops.LookupTableOperation (in 19.2.4, page 505)
```

Constructor summary

Brightness()

Method summary

process()

setBrightness(int) Sets the brightness adjustment value in percent (between -100 and 100).

Constructors

- *Brightness*
`public Brightness()`

Methods

- *process*
`public void process()` throws
`net.sourceforge.jiu.ops.MissingParameterException`,
`net.sourceforge.jiu.ops.OperationFailedException`,
`net.sourceforge.jiu.ops.WrongParameterException`
 - **Description copied from `net.sourceforge.jiu.ops.Operation` (in 19.2.5, page 508)**
This method does the actual work of the operation. It must be called after all parameters have been given to the operation object.

- **Throws**

- * `net.sourceforge.jiu.ops.WrongParameterException` – if at least one of the input parameters was not initialized appropriately (values out of the valid interval, etc.)
 - * `net.sourceforge.jiu.ops.MissingParameterException` – if any mandatory parameter was not given to the operation
 - * `net.sourceforge.jiu.ops.OperationFailedException` –
-

- *setBrightness*

`public void setBrightness(int newBrightness)`

- **Description**

Sets the brightness adjustment value in percent (between -100 and 100).

- **Parameters**

- * `newBrightness` – the amount of change to be applied to the brightness of the input image

- **Throws**

- * `java.lang.IllegalArgumentException` – if the argument is smaller than -100 or larger than 100

6.1.2 Class Contrast

Adjusts the contrast of an image. The amount of adjustment is given to the constructor as a percentage value between -100 and 100. -100 will make the resulting image middle-gray, 0 will leave it unchanged, 100 will map it to the eight corners of the color cube.

Usage example

This code snippet will reduce `image`'s contrast by 40 percent.

```
Contrast contrast = new Contrast();
contrast.setInputImage(image);
contrast.setContrast(-40);
contrast.process();
PixelImage adjustedImage = contrast.getOutputImage();
```

Declaration

```
public class Contrast
  extends net.sourceforge.jiu.ops.LookupTableOperation (in 19.2.4, page 505)
```

Constructor summary

Contrast()

Method summary

getContrast() Returns the contrast adjustment value associated with this operation.

process()

setContrast(int) Sets the value for contrast adjustment to be used within this operation.

Constructors

- *Contrast*
`public Contrast()`

Methods

- *getContrast*
`public int getContrast()`
 - **Description**
Returns the contrast adjustment value associated with this operation. The value lies between -100 and 100 (including both values).
 - **Returns** – contrast adjustment
 - **See also**
 - * `Contrast.setContrast(int)` (in 6.1.2, page 202)

- *process*

`public void process()` throws
`net.sourceforge.jiu.ops.MissingParameterException`,
`net.sourceforge.jiu.ops.OperationFailedException`,
`net.sourceforge.jiu.ops.WrongParameterException`

- **Description** copied from `net.sourceforge.jiu.ops.Operation` (in 19.2.5, page 508)

This method does the actual work of the operation. It must be called after all parameters have been given to the operation object.

- **Throws**

- * `net.sourceforge.jiu.ops.WrongParameterException` – if at least one of the input parameters was not initialized appropriately (values out of the valid interval, etc.)
- * `net.sourceforge.jiu.ops.MissingParameterException` – if any mandatory parameter was not given to the operation
- * `net.sourceforge.jiu.ops.OperationFailedException` –

- *setContrast*

`public void setContrast(int newContrast)`

- **Description**

Sets the value for contrast adjustment to be used within this operation.

- **Parameters**

- * `newContrast` – new contrast, between -100 and 100 (including both values)

- **Throws**

- * `java.lang.IllegalArgumentException` – if the new contrast value is not in the above mentioned interval

- **See also**

- * `Contrast.getContrast()` (in 6.1.2, page 201)

6.1.3 Class EqualizeHistogram

Equalize the image using histogram information separately for each channel. Works for intensity-based image types like (in 14.1.4, page 318) or (in 14.1.12, page 331).

Declaration

```
public class EqualizeHistogram
extends net.sourceforge.jiu.ops.LookupTableOperation (in 19.2.4, page 505)
```

Constructor summary

EqualizeHistogram(IntegerImage) Creates an object of this class and initializes the lookup tables with the argument input image.

Constructors

- *EqualizeHistogram*
`public EqualizeHistogram(net.sourceforge.jiu.data.IntegerImage in)` throws `net.sourceforge.jiu.ops.OperationFailedException`
 - **Description**
Creates an object of this class and initializes the lookup tables with the argument input image.
 - **Parameters**
 - * `in` – the input image

6.1.4 Class GammaCorrection

Corrects the gamma of an image. Works with (in 14.1.6, page 321), (in 14.1.16, page 336) and (in 14.1.8, page 325). Only the palette is manipulated for paletted images.

Changes intensity values by applying the formula $f(x) = \text{MAX} * (x / \text{MAX})^{(1/\text{gamma})}$ to each x from [0 ; MAX] to them. The MAX value is the maximum value allowed for an intensity value of the corresponding channel. It is determined by calling (in 14.1.7, page 323) on the input image.

The gamma parameter must be given to a GammaCorrection operation before the call to (in 6.1.4, page 205) is made. The valid interval for gamma is (0.0 ; (in 6.1.4, page 204)] (so 0.0 is not a valid value). Gamma values smaller than 1 will make the image darker, values larger than 1 will make it brighter.

Usage example

```
GammaCorrection gamma = new GammaCorrection();
gamma.setInputImage(image);
gamma.setGamma(2.2);
gamma.process();
PixelImage correctedImage = gamma.getOutputImage();
```

Declaration

```
public class GammaCorrection
extends net.sourceforge.jiu.ops.LookupTableOperation (in 19.2.4, page 505)
```

Field summary

MAX_GAMMA The maximum allowed value for gamma.

Constructor summary

GammaCorrection()

Method summary

getGamma() Returns the gamma value to be used for this operation.

process()

setGamma(double) Sets a new gamma value to be used in this operation.

Fields

- public static final double **MAX_GAMMA**
 - The maximum allowed value for gamma.

Constructors

- *GammaCorrection*
public **GammaCorrection()**

Methods

- *getGamma*

`public double getGamma()`

- **Description**

Returns the gamma value to be used for this operation.

- **Returns** – gamma value between 0 (not included) and (in 6.1.4, page 204)

- *process*

`public void process()` throws

`net.sourceforge.jiu.ops.MissingParameterException`,

`net.sourceforge.jiu.ops.OperationFailedException`,

`net.sourceforge.jiu.ops.WrongParameterException`

- **Description** copied from `net.sourceforge.jiu.ops.Operation` (in 19.2.5, page 508)

This method does the actual work of the operation. It must be called after all parameters have been given to the operation object.

- **Throws**

- * `net.sourceforge.jiu.ops.WrongParameterException` – if at least one of the input parameters was not initialized appropriately (values out of the valid interval, etc.)

- * `net.sourceforge.jiu.ops.MissingParameterException` – if any mandatory parameter was not given to the operation

- * `net.sourceforge.jiu.ops.OperationFailedException` –

- *setGamma*

`public void setGamma(double newGamma)`

- **Description**

Sets a new gamma value to be used in this operation.

- **Parameters**

- * `newGamma` – the new gamma value must be >0.0 and $\leq \text{MAX_GAMMA}$

- **Throws**

- * `java.lang.IllegalArgumentException` – if the argument is not in the described interval

6.1.5 Class HueSaturationValue

Adjusts saturation and value of a color image, optionally hue as well.
Supported image types: (in 14.1.12, page 331), (in 14.1.8, page 325).

Declaration

```
public class HueSaturationValue
extends net.sourceforge.jiu.ops.ImageToImageOperation (in 19.2.3, page 501)
implements net.sourceforge.jiu.data.RGBIndex
```

Constructor summary

HueSaturationValue()

Method summary

process()
setHueSaturationValue(int, int, int) Set the values for the adjustment of hue, saturation and value (brightness).
setSaturationValue(int, int) Set the amount of change to saturation and value (brightness) for this operation, between -100 and 100.

Constructors

- *HueSaturationValue*
public HueSaturationValue()

Methods

- *process*
public void process() throws
net.sourceforge.jiu.ops.MissingParameterException,
net.sourceforge.jiu.ops.OperationFailedException,
net.sourceforge.jiu.ops.WrongParameterException
 - **Description copied from net.sourceforge.jiu.ops.Operation** (in 19.2.5, page 508)
This method does the actual work of the operation. It must be called after all parameters have been given to the operation object.
 - **Throws**
 - * net.sourceforge.jiu.ops.WrongParameterException – if at least one of the input parameters was not initialized appropriately (values out of the valid interval, etc.)
 - * net.sourceforge.jiu.ops.MissingParameterException – if any mandatory parameter was not given to the operation
 - * net.sourceforge.jiu.ops.OperationFailedException –
- *setHueSaturationValue*
public void setHueSaturationValue(int hue, int saturation, int value)

- **Description**

Set the values for the adjustment of hue, saturation and value (brightness). Saturation and value must be from the interval -100 to 100 (also see (in 6.1.5, page 207)). Hue must be from the interval 0 to 359.

- **Parameters**

- * **hue** – the hue to be used for the complete image, between 0 and 359
- * **saturation** – change of saturation, between -100 and 100
- * **value** – change of saturation, between -100 and 100

- **Throws**

- * `java.lang.IllegalArgumentException` – if one of the arguments does not stay within the valid interval

- *setSaturationValue*

```
public void setSaturationValue( int saturation, int value )
```

- **Description**

Set the amount of change to saturation and value (brightness) for this operation, between -100 and 100. Calling this method also tells the operation not to modify the hue of the image.

- **Parameters**

- * **saturation** – change of saturation, between -100 and 100
- * **value** – change of saturation, between -100 and 100

- **Throws**

- * `java.lang.IllegalArgumentException` – if one of the two arguments does not stay within the -100 .. 100 interval

6.1.6 Class NormalizeHistogram

Normalize the image using histogram information, separately for each channel. Works for intensity-based image types like (in 14.1.4, page 318) or (in 14.1.12, page 331).

Declaration

```
public class NormalizeHistogram
extends net.sourceforge.jiu.ops.LookupTableOperation (in 19.2.4, page 505)
```

Constructor summary

NormalizeHistogram(IntegerImage) Creates an object of this class and initializes the lookup tables with the argument input image.

Constructors

- *NormalizeHistogram*
public NormalizeHistogram(net.sourceforge.jiu.data.IntegerImage in)
throws net.sourceforge.jiu.ops.OperationFailedException
 - **Description**
Creates an object of this class and initializes the lookup tables with the argument input image.

Chapter 7

Package

net.sourceforge.jiu.color.analysis

Package Contents

Page

Classes

Histogram1DCreator	210
<i>This class creates one-dimensional histograms for images with integer samples.</i>	
Histogram3DCreator	213
<i>This class creates three-dimensional histograms for images with integer samples.</i>	
MatrixCreator	215
<i>This class creates and initializes co-occurrence matrices and co-occurrence frequency matrices.</i>	
MeanDifference	217
<i>This operation determines the mean difference between two images.</i>	
TextureAnalysis	219
<i>This class determines a number of properties for a given co-occurrence matrix.</i>	

Contains classes that analyze pixel or sample values and thus determine certain image properties.

7.1 Classes

7.1.1 Class Histogram1DCreator

This class creates one-dimensional histograms for images with integer samples. Only (in 14.1.7, page 322)objects are supported.

Existing histogram objects can be given to this operation to be reused. Give an existing (in 5.1.3, page 180)object to this operation via (in 7.1.1, page 211).

The histogram can be created for any channel of an IntegerImage. The first channel (index 0) is the default channel. Use (in 7.1.1, page 211)to specify another one.

Note: Before JIU 0.10.0 there was a single HistogramCreator class.

Usage example

Creates a histogram for the third channel of an image.

```
Histogram1DCreator hc = new Histogram1DCreator();
hc.setImage(image, 2);
hc.process();
Histogram1D hist = hc.getHistogram();
```

Declaration

```
public class Histogram1DCreator
  extends net.sourceforge.jiu.ops.Operation (in 19.2.5, page 508)
```

Constructor summary

Histogram1DCreator()

Method summary

getHistogram() Returns the histogram used in this operation.

process()

setHistogram(Histogram1D) Sets a histogram object to be used for this operation.

setImage(IntegerImage) Set the image for which the histogram is to be initialized.

setImage(IntegerImage, int) Set the image and the channel index for which the histogram is to be initialized.

Constructors

- *Histogram1DCreator*
public **Histogram1DCreator**()

Methods

- *getHistogram*
public net.sourceforge.jiu.color.data.Histogram1D **getHistogram**()

- **Description**

Returns the histogram used in this operation.

- **Returns** – histogram object, newly-created or reused one

- **See also**

*

Histogram1DCreator.setHistogram(net.sourceforge.jiu.color.data.Histogram1D)
(in 7.1.1, page 211)

- *process*

public void process() throws

net.sourceforge.jiu.ops.MissingParameterException,

net.sourceforge.jiu.ops.OperationFailedException,

net.sourceforge.jiu.ops.WrongParameterException

- **Description copied from net.sourceforge.jiu.ops.Operation** (in 19.2.5, page 508)

This method does the actual work of the operation. It must be called after all parameters have been given to the operation object.

- **Throws**

* net.sourceforge.jiu.ops.WrongParameterException – if at least one of the input parameters was not initialized appropriately (values out of the valid interval, etc.)

* net.sourceforge.jiu.ops.MissingParameterException – if any mandatory parameter was not given to the operation

* net.sourceforge.jiu.ops.OperationFailedException –

- *setHistogram*

public void setHistogram(net.sourceforge.jiu.color.data.Histogram1D
histogram)

- **Description**

Sets a histogram object to be used for this operation. Within (in 7.1.1, page 211) it will be checked if this histogram is large enough for the image.

- **See also**

* Histogram1DCreator.getHistogram() (in 7.1.1, page 210)

- *setImage*

public void setImage(net.sourceforge.jiu.data.IntegerImage newImage)

- **Description**

Set the image for which the histogram is to be initialized. The first channel (index 0) is used by default.

- **Parameters**

* newImage – image object to be used

- **See also**

* Histogram1DCreator.setImage(net.sourceforge.jiu.data.IntegerImage, int)
(in 7.1.1, page 211)

- *setImage*

public void setImage(net.sourceforge.jiu.data.IntegerImage newImage, int
imageChannelIndex)

– **Description**

Set the image and the channel index for which the histogram is to be initialized.

– **Parameters**

- * **newImage** – image object to be used
- * **imageChannelIndex** – must not be negative and must be smaller than `newImage.getNumChannels()`

– **See also**

- * `Histogram1DCreator.setImage(net.sourceforge.jiu.data.IntegerImage)`
(in 7.1.1, page 211)

7.1.2 Class Histogram3DCreator

This class creates three-dimensional histograms for images with integer samples. Only (in 14.1.7, page 322) is supported. Existing histogram objects can be given to this operation to be reused. Note: Before JIU 0.10.0 there was a single HistogramCreator class.

Declaration

```
public class Histogram3DCreator
extends net.sourceforge.jiu.ops.Operation (in 19.2.5, page 508)
```

Constructor summary

Histogram3DCreator()

Method summary

getHistogram() Returns the histogram initialized in this operation.
process()
setHistogram3D(Histogram3D) Sets the histogram object to be reused for this operation.
setImage(IntegerImage) The image for which a histogram will be initialized.
setImage(IntegerImage, int, int, int) The image for which a histogram will be initialized.

Constructors

- *Histogram3DCreator*
public Histogram3DCreator()

Methods

- *getHistogram*
public net.sourceforge.jiu.color.data.Histogram3D getHistogram()
 – **Description**
 Returns the histogram initialized in this operation.
- *process*
public void process() throws
net.sourceforge.jiu.ops.MissingParameterException,
net.sourceforge.jiu.ops.OperationFailedException,
net.sourceforge.jiu.ops.WrongParameterException
 – **Description copied from net.sourceforge.jiu.ops.Operation (in 19.2.5, page 508)**
 This method does the actual work of the operation. It must be called after all parameters have been given to the operation object.
 – **Throws**

- * `net.sourceforge.jiu.ops.WrongParameterException` – if at least one of the input parameters was not initialized appropriately (values out of the valid interval, etc.)
- * `net.sourceforge.jiu.ops.MissingParameterException` – if any mandatory parameter was not given to the operation
- * `net.sourceforge.jiu.ops.OperationFailedException` –

- *setHistogram3D*

```
public void setHistogram3D( net.sourceforge.jiu.color.data.Histogram3D
    histogram )
```

- **Description**

Sets the histogram object to be reused for this operation. If this method is not called, a new histogram will be created.

- **Parameters**

- * `histogram` – the histogram object to be used in this operation
-

- *setImage*

```
public void setImage( net.sourceforge.jiu.data.IntegerImage newImage )
```

- **Description**

The image for which a histogram will be initialized. Simply calls (in 7.1.2, page 214) with 0, 1 and 2 as parameters.

- **Parameters**

- * `newImage` – the image for the histogram initialization
-

- *setImage*

```
public void setImage( net.sourceforge.jiu.data.IntegerImage newImage, int
    channelIndex1, int channelIndex2, int channelIndex3 )
```

- **Description**

The image for which a histogram will be initialized. Simply calls (in 7.1.2, page 214) with 0, 1 and 2 as parameters.

- **Parameters**

- * `newImage` –

7.1.3 Class MatrixCreator

This class creates and initializes co-occurrence matrices and co-occurrence frequency matrices.

Declaration

```
public class MatrixCreator
extends java.lang.Object
```

Method summary

createCoOccurrenceFrequencyMatrix(CoOccurrenceMatrix) Creates a new co-occurrence frequency with the same dimension as the argument co-occurrence matrix, calls (in 7.1.3, page 216) with them to initialize the new matrix, then returns it.

createCoOccurrenceMatrix(Gray8Image)

createCoOccurrenceMatrix(IntegerImage, int)

createCoOccurrenceMatrix(Palett8Image)

initCoOccurrenceFrequencyMatrix(CoOccurrenceMatrix, CoOccurrenceFrequencyMatrix) Initializes a co-occurrence frequency matrix from a co-occurrence matrix.

initCoOccurrenceMatrix(IntegerImage, int, CoOccurrenceMatrix) Initializes a co-occurrence matrix from the input image, using the direct four neighbor pixels.

Methods

- *createCoOccurrenceFrequencyMatrix*

```
public static net.sourceforge.jiu.color.data.CoOccurrenceFrequencyMatrix
createCoOccurrenceFrequencyMatrix(
net.sourceforge.jiu.color.data.CoOccurrenceMatrix A )
```

- **Description**

Creates a new co-occurrence frequency with the same dimension as the argument co-occurrence matrix, calls (in 7.1.3, page 216) with them to initialize the new matrix, then returns it. A (in 5.2.3, page 188) is created.

- **Parameters**

- * A – the co-occurrence matrix from which the resulting matrix will be initialized

- **Returns** – the newly-created co-occurrence frequency matrix

- **Throws**

- * java.lang.IllegalArgumentException – if the argument matrix is null

- *createCoOccurrenceMatrix*

```
public static net.sourceforge.jiu.color.data.CoOccurrenceMatrix
createCoOccurrenceMatrix( net.sourceforge.jiu.data.Gray8Image image )
```

- *createCoOccurrenceMatrix*

```
public static net.sourceforge.jiu.color.data.CoOccurrenceMatrix
createCoOccurrenceMatrix( net.sourceforge.jiu.data.IntegerImage image, int
channelIndex )
```


- *createCoOccurrenceMatrix*

```
public static net.sourceforge.jiu.color.data.CoOccurrenceMatrix
createCoOccurrenceMatrix( net.sourceforge.jiu.data.Paletted8Image image )
```

- *initCoOccurrenceFrequencyMatrix*

```
public static void initCoOccurrenceFrequencyMatrix(
net.sourceforge.jiu.color.data.CoOccurrenceMatrix A,
net.sourceforge.jiu.color.data.CoOccurrenceFrequencyMatrix cofm )
```

- **Description**

Initializes a co-occurrence frequency matrix from a co-occurrence matrix. The two argument matrices must be non-null and have the same dimension.

- **Parameters**

- * A – co-occurrence matrix used as input
- * cofm – co-occurrence matrix, will be initialized by this method

- **Throws**

- * `java.lang.IllegalArgumentException` – if either matrix is null or if the dimensions are not equal
-

- *initCoOccurrenceMatrix*

```
public static void initCoOccurrenceMatrix(
net.sourceforge.jiu.data.IntegerImage image, int channelIndex,
net.sourceforge.jiu.color.data.CoOccurrenceMatrix matrix )
```

- **Description**

Initializes a co-occurrence matrix from the input image, using the direct four neighbor pixels. The number of entries in the palette of the argument image must be equal to the dimension of the argument matrix.

- **Parameters**

- * image – the image that will be used to initialize the matrix
- * matrix – the matrix that will first be cleared and then initialized from the image

- **Throws**

- * `java.lang.IllegalArgumentException` – if at least one of the arguments is null or if the palette size is not equal to the matrix dimension

7.1.4 Class MeanDifference

This operation determines the mean difference between two images. It requires two images of the same resolution and adds the absolute difference of all samples. Then it divides by the number of samples in the image (width times height times number of channels).

Supported combinations of image types

- One of the two images is of type (in 14.1.12, page 331), the other of type (in 14.1.8, page 325).
- Both images are of the same type and that type implements (in 14.1.16, page 336).
- Both images are of the same type and that type implements (in 14.1.6, page 321).

Usage example

```
MeanDifference diff = new MeanDifference();
diff.setImages(img1, img2);
diff.process();
double meanDifference = diff.getDifference();
```

Declaration

```
public class MeanDifference
  extends net.sourceforge.jiu.ops.Operation (in 19.2.5, page 508)
```

Constructor summary

MeanDifference()

Method summary

getDifference() After a call to process, returns the determined mean difference value.
process()
setImages(PixelImage, PixelImage) Sets the two images for which the mean difference is to be determined.

Constructors

- *MeanDifference*
 public **MeanDifference()**

Methods

- *getDifference*
 public double **getDifference()**
 – **Description**
 After a call to process, returns the determined mean difference value.

- **Returns** – difference value, 0.0 or larger

- *process*

`public void process()` throws
`net.sourceforge.jiu.ops.MissingParameterException`,
`net.sourceforge.jiu.ops.OperationFailedException`,
`net.sourceforge.jiu.ops.WrongParameterException`

- **Description** copied from `net.sourceforge.jiu.ops.Operation` (in 19.2.5, page 508)

This method does the actual work of the operation. It must be called after all parameters have been given to the operation object.

- **Throws**

- * `net.sourceforge.jiu.ops.WrongParameterException` – if at least one of the input parameters was not initialized appropriately (values out of the valid interval, etc.)
- * `net.sourceforge.jiu.ops.MissingParameterException` – if any mandatory parameter was not given to the operation
- * `net.sourceforge.jiu.ops.OperationFailedException` –

- *setImages*

`public void setImages(net.sourceforge.jiu.data.PixelImage firstImage,`
`net.sourceforge.jiu.data.PixelImage secondImage)`

- **Description**

Sets the two images for which the mean difference is to be determined.

- **Parameters**

- * `firstImage` – first image
- * `secondImage` – second image

- **Throws**

- * `java.lang.IllegalArgumentException` – if either of the images is null, if their resolution is different or if their types are not supported by this operation

7.1.5 Class TextureAnalysis

This class determines a number of properties for a given co-occurrence matrix. The only input parameter is a mandatory co-occurrence matrix object to be specified using (in 7.1.5, page 221). Then (in 7.1.5, page 220) must be called. After that, the various properties can be retrieved using the corresponding get methods, e.g. (in 7.1.5, page 219), (in 7.1.5, page 220) etc. The following resources were helpful when creating this class:

- Article Suchen ohne Worte by Henning Mller in German computer magazine c't **15 / 2001** (at <http://www.heise.de/ct/01/15/004/>), p. 162ff.
- **GLCM Texture: A Tutorial** (at <http://www.ucalgary.ca/~mhallbey/texture/texture.tutorial.html>) by Mryka Hall-Beyer.
- **Texture Mapping of Neurological Magnetic Resonance Images** (at <http://www.burrill.demon.co.uk/meddoc/tmmri.html>) by J.H.P. Burrill

Declaration

```
public class TextureAnalysis
extends net.sourceforge.jiu.ops.Operation (in 19.2.5, page 508)
```

Constructor summary

TextureAnalysis()

Method summary

getContrast() Returns the contrast value determined in (in 7.1.5, page 220).
getCorrelation() Returns the correlation determined in (in 7.1.5, page 220).
getDissimilarity() Returns the dissimilarity value determined in (in 7.1.5, page 220).
getEnergy() Returns the energy value determined in (in 7.1.5, page 220).
getEntropy() Returns the entropy value determined in (in 7.1.5, page 220).
getHomogeneity() Returns the homogeneity value determined in (in 7.1.5, page 220).
getSum() Returns the sum of all entries in the matrix.
isSymmetrical()
process() Run over the input matrix and determine contrast, energy, entropy and homogeneity of that matrix.
setMatrix(CoOccurrenceMatrix) Sets the matrix to be used by this operation to the argument value.

Constructors

- *TextureAnalysis*
public **TextureAnalysis**()

Methods

- *getContrast*
public int **getContrast**()

– **Description**

Returns the contrast value determined in (in 7.1.5, page 220). Also called inertia.

- *getCorrelation*

`public double getCorrelation()`

– **Description**

Returns the correlation determined in (in 7.1.5, page 220).

- *getDissimilarity*

`public int getDissimilarity()`

– **Description**

Returns the dissimilarity value determined in (in 7.1.5, page 220).

- *getEnergy*

`public int getEnergy()`

– **Description**

Returns the energy value determined in (in 7.1.5, page 220).

- *getEntropy*

`public double getEntropy()`

– **Description**

Returns the entropy value determined in (in 7.1.5, page 220).

- *getHomogeneity*

`public double getHomogeneity()`

– **Description**

Returns the homogeneity value determined in (in 7.1.5, page 220). Also called inverse difference moment.

- *getSum*

`public int getSum()`

– **Description**

Returns the sum of all entries in the matrix.

- *isSymmetrical*

`public boolean isSymmetrical()`

- *process*

`public void process() throws
net.sourceforge.jiu.ops.MissingParameterException`

– **Description**

Run over the input matrix and determine contrast, energy, entropy and homogeneity of that matrix.

– **Throws**

* `net.sourceforge.jiu.ops.MissingParameterException` – if no co-occurrence matrix was provided using (in 7.1.5, page 221)

- *setMatrix*

```
public void setMatrix( net.sourceforge.jiu.color.data.CoOccurrenceMatrix m
)
```

- **Description**

Sets the matrix to be used by this operation to the argument value.

- **Parameters**

- * **m** – the matrix for which the various properties will be computed

Chapter 8

Package net.sourceforge.jiu.color.conversion

Package Contents *Page*

Classes

CMYKConversion	223
<i>Convert from CMYK color space to RGB color space.</i>	
LogLuvConversion	225
<i>Convert from LogLuv color representation to RGB color space and from LogL to grayscale.</i>	
PCDYCbCrConversion	227
<i>Convert from YCbCr color space (as used in Kodak PCD files) to RGB.</i>	

Classes to improve the results of color reduction algorithms. Some algorithms work by modifying pixel values before they are reduced, others compute the error of a pixel reduction and distributes that error over neighboring pixels that are yet to be reduced.

8.1 Classes

8.1.1 Class CMYKConversion

Convert from CMYK color space to RGB color space.

Declaration

```
public class CMYKConversion
  extends java.lang.Object
```

Method summary

convertCMYK32InterleavedToRGB24Planar(byte[], int, byte[], int, byte[], int, byte[], int, int) Converts a number of CMYK pixels stored in interleaved order (all samples of one pixel together: CMYKCMYKCMYK...) to RGB pixels which are stored as planes (all red samples together, etc.).

convertCMYK32PlanarToRGB24Planar(byte[], int, byte[], int, byte[], int, byte[], int, byte[], int, byte[], int, byte[], int, int) Converts a 32 bit CMYK pixel to a 24 bit RGB pixel.

Methods

- *convertCMYK32InterleavedToRGB24Planar*
 public static void **convertCMYK32InterleavedToRGB24Planar**(byte[] cmyk, int cmykOffset, byte[] red, int redOffset, byte[] green, int greenOffset, byte[] blue, int blueOffset, int numPixels)
 - **Description**
 Converts a number of CMYK pixels stored in interleaved order (all samples of one pixel together: CMYKCMYKCMYK...) to RGB pixels which are stored as planes (all red samples together, etc.).
 - **Parameters**
 - * **cmyk** – a byte array with numPixels times four samples stored in order C-M-Y-K
 - * **cmykOffset** – the index of the first byte that is to be accessed
 - * **red** – the byte array to which the red samples will be written by this method
 - * **redOffset** – the offset into the red array of the first sample to be written
 - * **green** – the byte array to which the green samples will be written by this method
 - * **greenOffset** – the offset into the green array of the first sample to be written
 - * **blue** – the byte array to which the blue samples will be written by this method
 - * **blueOffset** – the offset into the blue array of the first sample to be written
- *convertCMYK32PlanarToRGB24Planar*
 public static void **convertCMYK32PlanarToRGB24Planar**(byte[] cyan, int cyanOffset, byte[] magenta, int magentaOffset, byte[] yellow, int yellowOffset, byte[] black, int blackOffset, byte[] red, int redOffset, byte[] green, int greenOffset, byte[] blue, int blueOffset, int numPixels)
- *convertCMYK32ToRGB24*
 public static void **convertCMYK32ToRGB24**(int cyan, int magenta, int yellow, int black, int[] rgb)

– **Description**

Converts a 32 bit CMYK pixel to a 24 bit RGB pixel. Red, green and blue sample will be written at the indexes that (in 14.1.15, page 334) defines for them.

– **Parameters**

- * **cyan** – the cyan sample, must lie in the interval 0 to 255
- * **magenta** – the magenta sample, must lie in the interval 0 to 255
- * **yellow** – the yellow sample, must lie in the interval 0 to 255
- * **black** – the black sample, must lie in the interval 0 to 255
- * **rgb** – byte array for the destination R-G-B pixel, must have length 3 or larger, will be accessed using RGBIndex, each sample will lie in the interval 0 to 255

8.1.2 Class LogLuvConversion

Convert from LogLuv color representation to RGB color space and from LogL to grayscale. This implementation is based on the file `tif_luv.c` which is part of the TIFF library **libtiff** (at <http://www.libtiff.org>). The original implementation was written by Greg W. Larson. Learn more about the color type and its encoding on Greg's page **LogLuv Encoding for TIFF Images** (at <http://positron.cs.berkeley.edu/~gwlarson/pixformat/tiffluv.html>).

Declaration

```
public class LogLuvConversion
    extends java.lang.Object
```

Method summary

- convertLogL10toY(int)** Converts an unsigned 10 bit value (the argument must lie in the interval 0 to 1023) to a `double` luminance (brightness) value between 0.0 and 1.0.
- convertLogL16toGray8(byte[], byte[], int)** Converts a number of 16 bit LogL samples to 8 bit grayscale samples.
- convertLogL16toY(int)** Converts a signed 16 bit value (the argument must lie in the interval -32768 to 32767) to a `double` luminance (brightness) value between 0.0 and 1.0.
- convertLogLuv24InterleavedtoRGB24Planar(byte[], byte[], byte[], byte[], int)** Converts a number of 24 bit LogLuv pixels to 24 bit RGB pixels.
- convertLogLuv32InterleavedtoRGB24Planar(byte[], byte[], byte[], byte[], int)** Converts a number of 32 bit LogLuv pixels to 24 bit RGB pixels.

Methods

- *convertLogL10toY*

```
public static double convertLogL10toY( int p10 )
```

 - **Description**
 Converts an unsigned 10 bit value (the argument must lie in the interval 0 to 1023) to a `double` luminance (brightness) value between 0.0 and 1.0. This conversion is needed by both LogLuv to XYZ and LogL to grayscale.
 - **Parameters**
 * `p10` – input LogL value
 - **Returns** – `double` value with luminance, between 0 and 1
- *convertLogL16toGray8*

```
public static void convertLogL16toGray8( byte[] logl, byte[] gray, int num )
```

 - **Description**
 Converts a number of 16 bit LogL samples to 8 bit grayscale samples.
 - **Parameters**

- * **logl** – byte array with LogL samples, each 16 bit sample is stored as two consecutive bytes in order most-significant-byte least-significant-byte (network byte order); the array must be at least `num * 2` entries large
- * **gray** – the byte array to which the converted samples will be written
- * **num** – the number of samples to be converted

- *convertLogL16toY*

```
public static double convertLogL16toY( int p16 )
```

- **Description**

Converts a signed 16 bit value (the argument must lie in the interval -32768 to 32767) to a double luminance (brightness) value between 0.0 and 1.0. This conversion is needed by both LogLuv to XYZ and LogL to grayscale.

- **Parameters**

- * **p16** – input LogL value

- **Returns** – double value with luminance, between 0 and 1

- *convertLogLuv24InterleavedtoRGB24Planar*

```
public static void convertLogLuv24InterleavedtoRGB24Planar( byte[]
logluv, byte[] red, byte[] green, byte[] blue, int num )
```

- **Description**

Converts a number of 24 bit LogLuv pixels to 24 bit RGB pixels. Each LogLuv pixel is stored as three consecutive bytes in the `logluv` byte array. The first byte and the top two bits of the second are the LogL value, the remaining 14 bits are an index that encodes *u* and *v*.

- **Parameters**

- * **logluv** – byte array with LogLuv data, must be at least `num * 3` bytes large
- * **red** – the byte samples for the red channel will be written to this array
- * **green** – the byte samples for the green channel will be written to this array
- * **blue** – the byte samples for the blue channel will be written to this array
- * **num** – number of pixels to be converted

- *convertLogLuv32InterleavedtoRGB24Planar*

```
public static void convertLogLuv32InterleavedtoRGB24Planar( byte[]
logluv, byte[] red, byte[] green, byte[] blue, int num )
```

- **Description**

Converts a number of 32 bit LogLuv pixels to 24 bit RGB pixels. Each LogLuv pixel is stored as four consecutive bytes in the `logluv` byte array. The first two bytes represent the LogL value (most significant bytefirst), followed by the *u* value and then the *v* value.

- **Parameters**

- * **logluv** – byte array with LogLuv data, must be at least `num * 4` bytes large
- * **red** – the byte samples for the red channel will be written to this array
- * **green** – the byte samples for the green channel will be written to this array
- * **blue** – the byte samples for the blue channel will be written to this array
- * **num** – number of pixels to be converted

8.1.3 Class PCDYCbCrConversion

Convert from YCbCr color space (as used in Kodak PCD files) to RGB. Only works for 24 bits per pixel (8 bits per channel) image data.

Declaration

```
public class PCDYCbCrConversion
  extends java.lang.Object
  implements net.sourceforge.jiu.data.RGBIndex, net.sourceforge.jiu.color.YCbCrIndex
```

Method summary

convertYccToRgb(byte[], byte[], byte[], int, byte[], byte[], byte[], int, int)
 Converts pixels from YCbCr to RGB color space.

Methods

- *convertYccToRgb*

```
public static void convertYccToRgb( byte[] y, byte[] cb, byte[] cr, int
yccOffset, byte[] r, byte[] g, byte[] b, int rgbOffset, int num ) throws
java.lang.IllegalArgumentException
```

- **Description**

Converts pixels from YCbCr to RGB color space. Input pixels are given as three byte arrays for luminance and the two chroma components. Same for output pixels, three other arrays for red, green and blue. Offset values can be specified separately for the YCbCr and the RGB arrays.

- **Parameters**

- * **y** – the array of gray source samples
- * **cb** – the array of chroma blue source samples
- * **cr** – the array of chroma red source samples
- * **yccOffset** – offset value into the arrays y, cb and cr; color conversion will be started at the yccOffset'th value of each array
- * **r** – the array of red destination samples
- * **g** – the array of green destination samples
- * **b** – the array of blue destination samples
- * **rgbOffset** – offset value into the arrays r, g and b; destination samples will be written to the three arrays starting at the rgbOffset'th value of each array
- * **num** – the number of pixels to be converted

- **Throws**

- * **java.lang.IllegalArgumentException** – if one of the int values is negative or one of the arrays is null or too small

Chapter 9

Package

net.sourceforge.jiu.color.dithering

Package Contents

Page

Interfaces

SpotFunction	229
<i>An interface for spot functions to be used for clustered dot dithering.</i>	

Classes

ClusteredDotDither	230
<i>Apply a clustered dot ordered dither to a grayscale image, converting it to a bilevel image in the process.</i>	
DiamondSpotFunction	233
<i>A diamond spot function.</i>	
ErrorDiffusionDithering	234
<i>This class is used to apply error diffusion dithering to images that are being reduced in their color depth.</i>	
LineSpotFunction	240
<i>A line spot function.</i>	
OrderedDither	241
<i>This operation reduces the color depth of RGB truecolor images and grayscale images by applying ordered dithering.</i>	
RoundSpotFunction	244
<i>A round spot function.</i>	

Classes for conversion between color spaces. Right now, this mostly includes conversion to the RGB (red / green / blue) color space.

9.1 Interfaces

9.1.1 Interface SpotFunction

An interface for spot functions to be used for clustered dot dithering.

See also

- `ClusteredDotDither` (in 9.2.1, page 230)

Declaration

```
public interface SpotFunction
```

All known subclasses

`RoundSpotFunction` (in 9.2.6, page 244), `LineSpotFunction` (in 9.2.4, page 240), `DiamondSpotFunction` (in 9.2.2, page 233)

All classes known to implement interface

`RoundSpotFunction` (in 9.2.6, page 244), `LineSpotFunction` (in 9.2.4, page 240), `DiamondSpotFunction` (in 9.2.2, page 233)

Method summary

compute(double, double) Compute the spot intensity at the given position.
isBalanced() Returns if this spot function is balanced.

Methods

- *compute*
`double compute(double x, double y)`
 - **Description**
Compute the spot intensity at the given position.
 - **Parameters**
 - * `x` – horizontal position, must be between -1.0 and 1.0 (including both)
 - * `y` – vertical position, must be between -1.0 and 1.0 (including both)
 - **Returns** – the function value, must be between 0.0 and 1.0 (including both)
- *isBalanced*
`boolean isBalanced()`
 - **Description**
Returns if this spot function is balanced.

9.2 Classes

9.2.1 *Class* ClusteredDotDither

Apply a clustered dot ordered dither to a grayscale image, converting it to a bilevel image in the process. Works with (in 14.1.6, page 321)objects as input and (in 14.1.1, page 311)objects as output. Resolution of both must be the same. Use one of the predefined dither matrices or have one compiled from a spot function (given by an object of a class implementing (in 9.1.1, page 229)). There are a couple of classes implementing that spot function interface in the dithering package. If no matrix is specified, (in 9.2.1, page 231)creates a default matrix by calling (in 9.2.1, page 232).

Usage example

```
ClusteredDotDither dither = new ClusteredDotDither();
dither.setInputImage(image); // some GrayIntegerImage
dither.setDitherMatrix(8, 8, new DiamondSpotFunction());
dither.process();
PixelImage ditheredImage = dither.getOutputImage();
```

Credits

The predefined dither matrices were taken from **Netpbm** (at <http://netpbm.sourceforge.net/>)’s **pgmtopbm** program (the matrices are stored in the file `dithers.h`). I learned about spot functions and their use from Austin Donnelly’s GIMP plugin **newsprint** - it has extensive comments, and the **newsprint website** (at <http://www.cl.cam.ac.uk/~and1000/newsprint/>) explains the **theoretical background** (at <http://www.cl.cam.ac.uk/~and1000/newsprint/clustered-dot.html>).

Declaration

```
public class ClusteredDotDither
extends net.sourceforge.jiu.ops.ImageToImageOperation (in 19.2.3, page 501)
```

Constructor summary

ClusteredDotDither()

Method summary

process()
setDitherMatrix(int, int, int[]) Sets the dither matrix to be used in this operation.
setDitherMatrix(int, int, SpotFunction) Creates and sets a dither matrix of user-defined size and compiles it from a spot function.
setOrder3DitherMatrix() Sets a 6 times 6 elements matrix to be used for dithering.
setOrder4DitherMatrix() Sets an 8 times 8 elements matrix to be used for dithering.
setOrder8DitherMatrix() Sets a 16 times 16 elements matrix to be used for dithering.

Constructors

- *ClusteredDotDither*
`public ClusteredDotDither()`

Methods

- *process*
`public void process() throws`
`net.sourceforge.jiu.ops.MissingParameterException,`
`net.sourceforge.jiu.ops.OperationFailedException,`
`net.sourceforge.jiu.ops.WrongParameterException`
 - **Description** copied from `net.sourceforge.jiu.ops.Operation` (in 19.2.5, page 508)
This method does the actual work of the operation. It must be called after all parameters have been given to the operation object.
 - **Throws**
 - * `net.sourceforge.jiu.ops.WrongParameterException` – if at least one of the input parameters was not initialized appropriately (values out of the valid interval, etc.)
 - * `net.sourceforge.jiu.ops.MissingParameterException` – if any mandatory parameter was not given to the operation
 - * `net.sourceforge.jiu.ops.OperationFailedException` –
- *setDitherMatrix*
`public void setDitherMatrix(int width, int height, int[] data)`
 - **Description**
Sets the dither matrix to be used in this operation.
 - **Parameters**
 - * `width` – number of entries in horizontal direction
 - * `height` – number of entries in vertical direction
 - * `data` – matrix entries, in order top to bottom, and in each row left to right
 - **Throws**
 - * `java.lang.IllegalArgumentException` – if width or height are smaller than one or data is null or has not at least width times height entries
- *setDitherMatrix*
`public void setDitherMatrix(int width, int height, SpotFunction f)`
 - **Description**
Creates and sets a dither matrix of user-defined size and compiles it from a spot function. Creates a matrix from the spot function and calls (in 9.2.1, page 231).
 - **Parameters**
 - * `width` – width of matrix, must be one or larger
 - * `height` – height of matrix, must be one or larger
 - * `f` – the spot function to be used for compiling the matrix

- *setOrder3DitherMatrix*

```
public void setOrder3DitherMatrix( )
```

- **Description**

Sets a 6 times 6 elements matrix to be used for dithering.

- *setOrder4DitherMatrix*

```
public void setOrder4DitherMatrix( )
```

- **Description**

Sets an 8 times 8 elements matrix to be used for dithering.

- *setOrder8DitherMatrix*

```
public void setOrder8DitherMatrix( )
```

- **Description**

Sets a 16 times 16 elements matrix to be used for dithering.

9.2.2 Class *DiamondSpotFunction*

A diamond spot function.

See also

- *ClusteredDotDither* (in 9.2.1, page 230)

Declaration

```
public class DiamondSpotFunction
  extends java.lang.Object
  implements SpotFunction
```

Constructor summary

DiamondSpotFunction()

Method summary

compute(double, double)
isBalanced()

Constructors

- *DiamondSpotFunction*
public **DiamondSpotFunction()**

Methods

- *compute*
double **compute(double x, double y)**
 - **Description copied from SpotFunction (in 9.1.1, page 229)**
Compute the spot intensity at the given position.
 - **Parameters**
 - * **x** – horizontal position, must be between -1.0 and 1.0 (including both)
 - * **y** – vertical position, must be between -1.0 and 1.0 (including both)
 - **Returns** – the function value, must be between 0.0 and 1.0 (including both)
- *isBalanced*
boolean **isBalanced()**
 - **Description copied from SpotFunction (in 9.1.1, page 229)**
Returns if this spot function is balanced.

9.2.3 Class ErrorDiffusionDithering

This class is used to apply error diffusion dithering to images that are being reduced in their color depth. Works with (in 14.1.6, page 321) and (in 14.1.16, page 336) objects. For RGB images, a quantizer must be specified via (in 9.2.3, page 237). That quantizer must have been initialized (it must have searched for / given a palette that it can map to).

This class offers six predefined types of error diffusion dithering. In addition, user-defined types can be integrated by providing a information on how the error is to be distributed; see the description of (in 9.2.3, page 237).

Usage examples

Color

This small program maps some RGB24Image object to a Paletted8Image with 120 entries in its palette, using Stucki error diffusion dithering in combination with an octree color quantizer.

```
MemoryRGB24Image image = ...; // some RGB image
OctreeColorQuantizer quantizer = new OctreeColorQuantizer();
quantizer.setInputImage(image);
quantizer.setPaletteSize(120);
quantizer.init();
ErrorDiffusionDithering edd = new ErrorDiffusionDithering();
edd.setType(ErrorDiffusionDithering.TYPE_STUCKI);
edd.setQuantizer(quantizer);
edd.setInputImage(image);
edd.process();
PixelImage quantizedImage = edd.getOutputImage();
```

Grayscale to black and white

In this example, a (in 14.1.4, page 318) object is reduced to black and white using Floyd-Steinberg dithering.

```
Gray8Image image = ...; // some grayscale image
ErrorDiffusionDithering edd = new ErrorDiffusionDithering();
edd.setGrayscaleOutputBits(1);
edd.setInputImage(image);
edd.process();
PixelImage ditheredImage = edd.getOutputImage();
// if you need something more specific than PixelImage:
BilevelImage output = null;
// ditheredImage should be a BilevelImage...
if (ditheredImage instanceof BilevelImage)
{
    // ... and it is!
    output = (BilevelImage)ditheredImage;
}
```

TODO

Adjust this class to be able to process 16 bits per sample.

Theoretical background

The predefined templates were taken from the book Bit-mapped graphics (2nd edition) by Steve Rimmer, published by Windcrest / McGraw-Hill, ISBN 0-8306-4208-0. The part on error diffusion dithering starts on page 375.

Several sources recommend Robert Ulichney's book **Digital Halftoning** (at <http://crl.research.compaq.com/who/people/ulichney/bib/DigitalHalftoning/Digital-Halftoning.html>) for this topic (published by The MIT Press, ISBN 0-262-21009-6).

Unfortunately, I wasn't able to get a copy (or the CD-ROM version published by **Dr. Dobb's Journal** (at <http://www.ddj.com>)).

Declaration

```
public class ErrorDiffusionDithering
  extends net.sourceforge.jiu.ops.ImageToImageOperation (in 19.2.3, page 501)
  implements net.sourceforge.jiu.data.RGBIndex
```

Field summary

DEFAULT_TYPE The default error diffusion type, to be used if none is specified by the user.

INDEX_ERROR_DENOMINATOR The index of the denominator of the relative part of the error of a neighbor pixel.

INDEX_ERROR_NUMERATOR The index of the numerator of the relative part of the error of a neighbor pixel.

INDEX_X_POS The index for the horizontal position of a neighbor pixel.

INDEX_Y_POS The index for the vertical position of a neighbor pixel.

TYPE_BURKES Constant for Burkes error diffusion.

TYPE_FLOYD_STEINBERG Constant for Floyd-Steinberg error diffusion.

TYPE_JARVIS_JUDICE_NINKE Constant for Burkes error diffusion.

TYPE_SIERRA Constant for Burkes error diffusion.

TYPE_STEVENSON_ARCE Constant for Burkes error diffusion.

TYPE_STUCKI Constant for Stucki error diffusion.

Constructor summary

ErrorDiffusionDithering() Creates a new object of this class and set the dithering type to (in 9.2.3, page 236).

Method summary

process() Quantizes the input image, distributing quantization errors to neighboring pixels.

setGrayscaleOutputBits(int) Sets the number of bits to be in the output image when a grayscale image is quantized.

setQuantizer(Quantizer) Sets the color quantizer to be used (if the input image is a truecolor image).

setTemplateData(int[][]) Set information on how errors are to be distributed by this error diffusion dithering operation.

setTemplateType(int) Sets a new template type.

setTruecolorOutput(boolean) When dithering an RGB input image, this method specifies whether the output will be an (in 14.1.16, page 336)(**true**) or a (in 14.1.8, page 325)(**false**).

Fields

- public static final int **TYPE_FLOYD_STEINBERG**
 - Constant for Floyd-Steinberg error diffusion. The quantization error is distributed to four neighboring pixels.
- public static final int **TYPE_STUCKI**
 - Constant for Stucki error diffusion. The quantization error is distributed to twelve neighboring pixels.
- public static final int **TYPE_BURKES**
 - Constant for Burkes error diffusion. The quantization error is distributed to seven neighboring pixels.
- public static final int **TYPE_SIERRA**
 - Constant for Burkes error diffusion. The quantization error is distributed to ten neighboring pixels.
- public static final int **TYPE_JARVIS_JUDICE_NINKE**
 - Constant for Burkes error diffusion. The quantization error is distributed to twelve neighboring pixels.
- public static final int **TYPE_STEVENSON_ARCE**
 - Constant for Burkes error diffusion. The quantization error is distributed to twelve neighboring pixels.
- public static final int **DEFAULT_TYPE**
 - The default error diffusion type, to be used if none is specified by the user.
- public static final int **INDEX_X_POS**
 - The index for the horizontal position of a neighbor pixel. For a description, see the constructor (in 9.2.3, page 237).
- public static final int **INDEX_Y_POS**
 - The index for the vertical position of a neighbor pixel. For a description, see the constructor (in 9.2.3, page 237).
- public static final int **INDEX_ERROR_NUMERATOR**
 - The index of the numerator of the relative part of the error of a neighbor pixel. For a description, see the constructor (in 9.2.3, page 237).
- public static final int **INDEX_ERROR_DENOMINATOR**
 - The index of the denominator of the relative part of the error of a neighbor pixel. For a description, see the constructor (in 9.2.3, page 237).

Constructors

- *ErrorDiffusionDithering*

public ErrorDiffusionDithering()

– **Description**

Creates a new object of this class and set the dithering type to (in 9.2.3, page 236).

Methods

- *process*

public void process() throws
 net.sourceforge.jiu.ops.MissingParameterException,
 net.sourceforge.jiu.ops.WrongParameterException

– **Description**

Quantizes the input image, distributing quantization errors to neighboring pixels. Works for (in 14.1.4, page 318)(then (in 9.2.3, page 237) must have been called to set a number of output bits between 1 and 7) objects and (in 14.1.12, page 331)(then a quantizer must be specified using (in 9.2.3, page 237)) objects.

- *setGrayscaleOutputBits*

public void setGrayscaleOutputBits(int numBits)

– **Description**

Sets the number of bits to be in the output image when a grayscale image is quantized. If the input image is of type (in 14.1.4, page 318), only values between 1 and 7 are valid.

– **Parameters**

* **numBits** – the number of bits in the output image

- *setQuantizer*

public void setQuantizer(
 net.sourceforge.jiu.color.quantization.RGBQuantizer **q)**

– **Description**

Sets the color quantizer to be used (if the input image is a truecolor image).

– **Parameters**

* **q** – an object of a class implementing the RGBQuantizer interface

- *setTemplateData*

public void setTemplateData(int[] [] data)

– **Description**

Set information on how errors are to be distributed by this error diffusion dithering operation.

Error diffusion dithering works by quantizing each pixel and distributing the resulting error to neighboring pixels. Quantizing maps a pixel to another pixel. Each pixel is made up of one or more samples (as an example, three samples r_{orig} , g_{orig} and b_{orig} for the original pixel of an RGB image and r_{quant} , g_{quant} and b_{quant} for the quantized pixel).

The process of quantization attempts to find a quantized pixel that is as close to the original as possible. In the ideal case, the difference between original and quantized pixel is zero for each sample. Otherwise, this quantization error is non-zero, positive or negative. Example: original pixel (12, 43, 33), quantized pixel (10, 47, 40); the error is $(12 - 10, 43 - 47, 40 - 33) = (2, -4, 7)$. The error (2, -4, 7) is to be distributed to neighboring pixels.

The `data` argument of this constructor describes how to do that. It is a two-dimensional array of int values. Each of the one-dimensional int arrays of `data` describe one neighboring pixel and the relative amount of the error that it gets. That is why `data.length` specifies the number of neighboring pixels involved in distributing the error. Let's call the pixel that was just quantized the current pixel. It is at image position (x, y).

Each of the one-dimensional arrays that are part of `data` must have a length of 4. The meaning of these four values is now described. The values can be accessed by the `INDEX_xyz` constants of this class. These four values describe the position of one neighboring pixel and the relative amount of the error that will be added to or subtracted from it.

- * (in 9.2.3, page 236)(0): the difference between the horizontal position of the current pixel, x, and the neighboring pixel; can take a positive or negative value, or zero; exception: the y position of the current pixel is zero; in that case, this value must be larger than zero, because neighboring pixels that get part of the error must be to the right of or below the current pixel
- * (in 9.2.3, page 236)(1): the difference between the vertical position of the current pixel, y, and the neighboring pixel; must be equal to or larger than 0
- * (in 9.2.3, page 236)(2): the numerator of the relative part of the error that will be added to this neighboring pixel; must not be equal to 0
- * (in 9.2.3, page 236)(3): the denominator of the relative part of the error that will be added to this neighboring pixel; must not be equal to 0

Example: the predefined dithering type Floyd-Steinberg. It has the following `data` array:

```
int[] [] FLOYD_STEINBERG = {{ 1,  0, 7, 16},
    {-1,  1, 3, 16},
    { 0,  1, 5, 16},
    { 1,  1, 1, 16}};
```

Each of the one-dimensional arrays is of length 4. Accidentally, there are also four one-dimensional arrays. The number of arrays is up to the designer. The first array {1, 0, 7, 16} is interpreted as follows—go to the pixel with a horizontal difference of 1 and a vertical difference of 0 (so, the pixel to the right of the current pixel) and add 7 / 16th of the quantization error to it. Then go to the pixel at position (-1, 1) (one to the left, one row below the current row) and add 3 / 16th of the error to it. The other two one-dimensional arrays are processed just like that.

As you can see, the four relative errors 1/16, 3/16, 5/16 and 7/16 sum up to 1 (or 16/16); this is in a precondition to make sure that the error is distributed completely.

– Parameters

- * `data` – contains a description of how the error is to be distributed

• *setTemplateType*

```
public void setTemplateType( int type )
```

– Description

Sets a new template type. The argument must be one of the TYPE_xyz constants of this class.

– **Parameters**

* **type** – int value, one of the TYPE_xyz constants of this class

– **Throws**

* `java.lang.IllegalArgumentException` – if the argument is not of the TYPE_xyz constants

• *setTruecolorOutput*

`public void setTruecolorOutput(boolean truecolor)`

– **Description**

When dithering an RGB input image, this method specifies whether the output will be an (in 14.1.16, page 336)(`true`) or a (in 14.1.8, page 325)(`false`).

– **Parameters**

* **truecolor** – true if truecolor output is wanted

9.2.4 Class LineSpotFunction

A line spot function.

See also

- `ClusteredDotDither` (in 9.2.1, page 230)

Declaration

```
public class LineSpotFunction
  extends java.lang.Object
  implements SpotFunction
```

Constructor summary

LineSpotFunction()

Method summary

compute(double, double)
isBalanced()

Constructors

- *LineSpotFunction*
public LineSpotFunction()

Methods

- *compute*
double compute(double x, double y)
 - **Description copied from SpotFunction (in 9.1.1, page 229)**
Compute the spot intensity at the given position.
 - **Parameters**
 - * **x** – horizontal position, must be between -1.0 and 1.0 (including both)
 - * **y** – vertical position, must be between -1.0 and 1.0 (including both)
 - **Returns** – the function value, must be between 0.0 and 1.0 (including both)
- *isBalanced*
boolean isBalanced()
 - **Description copied from SpotFunction (in 9.1.1, page 229)**
Returns if this spot function is balanced.

9.2.5 Class OrderedDither

This operation reduces the color depth of RGB truecolor images and grayscale images by applying ordered dithering. A relatively nice output image (for a human observer) will be created at the cost of introducing noise into the output image. The algorithm is relatively fast, but its quality is usually inferior to what can be reached by using (in 9.2.3, page 234) or similar other methods.

Supported conversions

- **Grayscale to bilevel** maps `GrayIntegerImage` objects to `BilevelImage` objects.
- **Grayscale to grayscale** maps `GrayIntegerImage` objects to other `GrayIntegerImage` objects. Right now, only `Gray8Image` objects are supported. After reducing the number of bits, each sample is immediately scaled back to 8 bits per pixel in order to fit into another `Gray8Image` object.
- **Truecolor to paletted** maps `RGBIntegerImage` objects to `Paletted8Image` objects; the sum of the output bits must not be larger than 8

Usage example

The following code snippet demonstrates how to create a paletted image with 256 colors, using three bits for red and green and two bits for blue, from an RGB truecolor image.

```
OrderedDither od = new OrderedDither();
od.setRgbBits(3, 3, 2);
od.setInputImage(image);
od.process();
Paletted8Image ditheredImage = (Paletted8Image)od.getOutputImage();
```

Declaration

```
public class OrderedDither
  extends net.sourceforge.jiu.ops.ImageToImageOperation (in 19.2.3, page 501)
  implements net.sourceforge.jiu.data.RGBIndex
```

Constructor summary

OrderedDither()

Method summary

process()
setOutputBits(int)
setRgbBits(int, int, int) Sets the number of bits to be used for each RGB component in the output image.
setStandardThresholdValues() Calls (in 9.2.5, page 243) with a 16 x 16 matrix.
setThresholdValues(int[], int, int) Defines a matrix of threshold values that will be used for grayscale dithering.

Constructors

- *OrderedDither*
`public OrderedDither()`

Methods

- *process*
`public void process() throws`
`net.sourceforge.jiu.ops.MissingParameterException,`
`net.sourceforge.jiu.ops.OperationFailedException,`
`net.sourceforge.jiu.ops.WrongParameterException`
 - **Description copied from net.sourceforge.jiu.ops.Operation (in 19.2.5, page 508)**
This method does the actual work of the operation. It must be called after all parameters have been given to the operation object.
 - **Throws**
 - * `net.sourceforge.jiu.ops.WrongParameterException` – if at least one of the input parameters was not initialized appropriately (values out of the valid interval, etc.)
 - * `net.sourceforge.jiu.ops.MissingParameterException` – if any mandatory parameter was not given to the operation
 - * `net.sourceforge.jiu.ops.OperationFailedException` –
- *setOutputBits*
`public void setOutputBits(int bits)`
- *setRgbBits*
`public void setRgbBits(int red, int green, int blue)`
 - **Description**
Sets the number of bits to be used for each RGB component in the output image. Each argument must be one or larger. The values defined by this method are only used if the input image implements `RGBIntegerImage`. Later, in (in 9.2.5, page 242), these values are checked against the actual number of bits per component in the input image. If any of the arguments of this method is equal to or larger than those actual bits per channel values, a `WrongParameterException` will then be thrown. Right now, there is no way how this can be checked, because the input image may not have been defined yet.
 - **Parameters**
 - * **red** – number of bits for the red channel in the output image
 - * **green** – number of bits for the green channel in the output image
 - * **blue** – number of bits for the blue channel in the output image
 - **Throws**
 - * `java.lang.IllegalArgumentException` – if at least one argument is smaller than 1
- *setStandardThresholdValues*
`public void setStandardThresholdValues()`

- **Description**

Calls (in 9.2.5, page 243) with a 16 x 16 matrix.

- *setThresholdValues*

```
public void setThresholdValues( int[] values, int valueWidth, int  
valueHeight )
```

- **Description**

Defines a matrix of threshold values that will be used for grayscale dithering.

- **Parameters**

- * **values** – the int values to use for comparing
- * **valueWidth** –
- * **valueHeight** –

9.2.6 Class RoundSpotFunction

A round spot function.

See also

- `ClusteredDotDither` (in 9.2.1, page 230)

Declaration

```
public class RoundSpotFunction
  extends java.lang.Object
  implements SpotFunction
```

Constructor summary

RoundSpotFunction()

Method summary

compute(double, double)
isBalanced()

Constructors

- *RoundSpotFunction*
`public RoundSpotFunction()`

Methods

- *compute*
`double compute(double x, double y)`
 - **Description copied from SpotFunction** (in 9.1.1, page 229)
Compute the spot intensity at the given position.
 - **Parameters**
 - * **x** – horizontal position, must be between -1.0 and 1.0 (including both)
 - * **y** – vertical position, must be between -1.0 and 1.0 (including both)
 - **Returns** – the function value, must be between 0.0 and 1.0 (including both)
- *isBalanced*
`boolean isBalanced()`
 - **Description copied from SpotFunction** (in 9.1.1, page 229)
Returns if this spot function is balanced.

Chapter 10

Package net.sourceforge.jiu.color.io

Package Contents

Page

Classes

HistogramSerialization	246
<i>This class has static methods for saving histograms.</i>	
MatrixSerialization	248
<i>Write co-occurrence and co-occurrence frequency matrices to text files.</i>	
PaletteSerialization	249
<i>This class loads and saves palettes.</i>	

Classes to read and write color-related data from and to files.

10.1 Classes

10.1.1 Class HistogramSerialization

This class has static methods for saving histograms. Text files (actually, any `PrintStream`, so you could write to standard output using `System.out`) are used to store the histogram information. Hint: When creating a `Histogram` object yourself, set the `autoFlush` argument of the constructor to `false`. You should also wrap your `PrintStream` object into a `PrintWriter` object. That may speed things up.

A simple format is used for storing the histograms. The first line holds the number of components. This would be 3 for a three-dimensional histogram, e.g. for RGB color images, or 1 for a one-dimensional histogram as used for a grayscale image.

Next, as many lines as dimensions follow. Each line holds the maximum value allowed for that component. The minimum value is always zero. Typically, the maximum values are all the same, e.g. 255 for each component of a 24 bit RGB truecolor image.

Following these header lines is the actual histogram. Each line holds a non-zero counter value for one pixel. The counter is always the last integer value in the line.

Example:

```
34 0 55 4033
```

For the histogram of an `RGB24Image`, this would mean that the pixel red=34, green=0, blue=55 occurs 4033 times.

```
0 2
```

For the histogram of any one channel image, this means that the value 0 occurs twice.

Declaration

```
public class HistogramSerialization
    extends java.lang.Object
```

Method summary

save(Histogram1D, PrintStream) Saves a one-dimensional histogram to a text output stream.

save(Histogram3D, PrintStream) Saves a three-dimensional histogram to a text output stream.

Methods

- *save*

```
public static void save( net.sourceforge.jiu.color.data.Histogram1D hist,
    java.io.PrintStream out )
```

 - **Description**
Saves a one-dimensional histogram to a text output stream.
 - **Parameters**
 - * `hist` – the histogram that will be written to a stream
 - * `out` – the stream that will be written to
- *save*

```
public static void save( net.sourceforge.jiu.color.data.Histogram3D hist,
    java.io.PrintStream out )
```

– **Description**

Saves a three-dimensional histogram to a text output stream.

– **Parameters**

- * **hist** – the histogram to be saved
- * **out** – the output stream where the histogram will be saved to

10.1.2 Class MatrixSerialization

Write co-occurrence and co-occurrence frequency matrices to text files.

Declaration

```
public class MatrixSerialization
    extends java.lang.Object
```

Method summary

```
    save(CoOccurrenceFrequencyMatrix, PrintStream)
    save(CoOccurrenceMatrix, PrintStream)
```

Methods

- *save*

```
public static void save(
    net.sourceforge.jiu.color.data.CoOccurrenceFrequencyMatrix matrix,
    java.io.PrintStream out )
```
- *save*

```
public static void save( net.sourceforge.jiu.color.data.CoOccurrenceMatrix
    matrix, java.io.PrintStream out )
```

10.1.3 Class PaletteSerialization

This class loads and saves palettes. Loading is done using the (in 2.1.6, page 92)class - an image is loaded which is supposed to have no more than 256 pixels, the palette entries. When saving, the (in 2.1.10, page 115)is used to store palettes as .ppm files.

Declaration

```
public class PaletteSerialization
extends java.lang.Object
implements net.sourceforge.jiu.data.RGBIndex
```

Method summary

convertImageToPalette(RGB24Image) Create a palette from the pixels of the argument image.

convertPaletteToImage(Palette) Creates an RGB24Image from the palette entries, each entry becomes a pixel in an image of width 1 and height palette.getNumEntries().

load(File) Loads a palette from the argument file.

save(Palette, File) Saves the palette to the given file as a PPM image file.

Methods

- *convertImageToPalette*

```
public static net.sourceforge.jiu.data.Palette convertImageToPalette(
net.sourceforge.jiu.data.RGB24Image image )
```

- **Description**

Create a palette from the pixels of the argument image.

- *convertPaletteToImage*

```
public static net.sourceforge.jiu.data.RGB24Image convertPaletteToImage(
net.sourceforge.jiu.data.Palette palette )
```

- **Description**

Creates an RGB24Image from the palette entries, each entry becomes a pixel in an image of width 1 and height palette.getNumEntries().

- *load*

```
public static net.sourceforge.jiu.data.Palette load( java.io.File paletteFile
)
```

- **Description**

Loads a palette from the argument file. Uses (in 2.1.6, page 92)to load an image from the argument file, then calls (in 10.1.3, page 249)and returns the palette created that way.

- *save*

```
public static void save( net.sourceforge.jiu.data.Palette palette,
java.io.File paletteFile ) throws java.io.IOException
```

– **Description**

Saves the palette to the given file as a PPM image file. Uses (in 2.1.10, page 115).

Chapter 11

Package

net.sourceforge.jiu.color.promotion

Package Contents

Page

Classes

PromotionGray16	252
<i>Converts BilevelImage and Gray8Image objects to Gray16Image objects.</i>	
PromotionGray8	253
<i>Converts BilevelImage objects to Gray8Image objects.</i>	
PromotionPaletted8	254
<i>Converts (in 14.1.1, page 311) and (in 14.1.4, page 318) objects to (in 14.1.8, page 325) objects.</i>	
PromotionRGB24	255
<i>Converts several image types to RGB.</i>	
PromotionRGB48	256
<i>Converts several image types to (in 14.1.13, page 332).</i>	

Classes to convert JIU image objects to other image types that require more memory. This is a lossless operation, all information is kept. However, the new image object typically requires more memory to store the same information. A promotion operation is necessary if a certain operation cannot be performed on a smaller image type, but once the image has been promoted to a larger type, the operation becomes possible.

Example: Convolution kernel filtering cannot be applied to paletted images. They require intensity information on each pixel. Once a paletted image has been promoted to be an RGB truecolor image, filtering is possible.

11.1 Classes

11.1.1 Class PromotionGray16

Converts BilevelImage and Gray8Image objects to Gray16Image objects. Promotion is a lossless operation that will lead to an output image that holds the same image in a way that demands more memory.

Declaration

```
public class PromotionGray16
extends net.sourceforge.jiu.ops.ImageToImageOperation (in 19.2.3, page 501)
```

Constructor summary

PromotionGray16()

Method summary

process()

Constructors

- *PromotionGray16*
public PromotionGray16()

Methods

- *process*
public void process() throws
`net.sourceforge.jiu.ops.MissingParameterException`,
`net.sourceforge.jiu.ops.OperationFailedException`,
`net.sourceforge.jiu.ops.WrongParameterException`
 - **Description** copied from `net.sourceforge.jiu.ops.Operation` (in 19.2.5, page 508)
This method does the actual work of the operation. It must be called after all parameters have been given to the operation object.
 - **Throws**
 - * `net.sourceforge.jiu.ops.WrongParameterException` – if at least one of the input parameters was not initialized appropriately (values out of the valid interval, etc.)
 - * `net.sourceforge.jiu.ops.MissingParameterException` – if any mandatory parameter was not given to the operation
 - * `net.sourceforge.jiu.ops.OperationFailedException` –

11.1.1.2 Class PromotionGray8

Converts BilevelImage objects to Gray8Image objects. Promotion is a lossless operation that will lead to an output image that holds the same image in a way that demands more memory.

Declaration

```
public class PromotionGray8
extends net.sourceforge.jiu.ops.ImageToImageOperation (in 19.2.3, page 501)
```

Constructor summary

PromotionGray8()

Method summary

process()

Constructors

- *PromotionGray8*
public **PromotionGray8**()

Methods

- *process*
public void **process**() throws
net.sourceforge.jiu.ops.MissingParameterException,
net.sourceforge.jiu.ops.OperationFailedException,
net.sourceforge.jiu.ops.WrongParameterException
 - **Description copied from net.sourceforge.jiu.ops.Operation** (in 19.2.5, page 508)
This method does the actual work of the operation. It must be called after all parameters have been given to the operation object.
 - **Throws**
 - * net.sourceforge.jiu.ops.WrongParameterException – if at least one of the input parameters was not initialized appropriately (values out of the valid interval, etc.)
 - * net.sourceforge.jiu.ops.MissingParameterException – if any mandatory parameter was not given to the operation
 - * net.sourceforge.jiu.ops.OperationFailedException –

11.1.3 Class PromotionPaletted8

Converts (in 14.1.1, page 311) and (in 14.1.4, page 318) objects to (in 14.1.8, page 325) objects. This lossless operation will only lead to an output image that holds the input image in a way that demands more memory.

Declaration

```
public class PromotionPaletted8
  extends net.sourceforge.jiu.ops.ImageToImageOperation (in 19.2.3, page 501)
```

Constructor summary

PromotionPaletted8()

Method summary

process()

Constructors

- *PromotionPaletted8*
public **PromotionPaletted8**()

Methods

- *process*
public void **process**() throws
net.sourceforge.jiu.ops.MissingParameterException,
net.sourceforge.jiu.ops.OperationFailedException,
net.sourceforge.jiu.ops.WrongParameterException
 - **Description copied from net.sourceforge.jiu.ops.Operation** (in 19.2.5, page 508)
This method does the actual work of the operation. It must be called after all parameters have been given to the operation object.
 - **Throws**
 - * net.sourceforge.jiu.ops.WrongParameterException – if at least one of the input parameters was not initialized appropriately (values out of the valid interval, etc.)
 - * net.sourceforge.jiu.ops.MissingParameterException – if any mandatory parameter was not given to the operation
 - * net.sourceforge.jiu.ops.OperationFailedException –

11.1.1.4 Class PromotionRGB24

Converts several image types to RGB. Promoting is a lossless operation that will only lead to an output image that holds the same image in a way that demands more memory.

If you give an image implementing RGB24Image to this operation, a WrongParameterException will be thrown. This operation could also return the input image, but this might lead to the wrong impression that a copy of the input was produced which can be modified without changing the original.

Declaration

```
public class PromotionRGB24
  extends net.sourceforge.jiu.ops.ImageToImageOperation (in 19.2.3, page 501)
```

Constructor summary

PromotionRGB24()

Method summary

process()

Constructors

- *PromotionRGB24*
public **PromotionRGB24**()

Methods

- *process*
public void **process**() throws
net.sourceforge.jiu.ops.MissingParameterException,
net.sourceforge.jiu.ops.OperationFailedException,
net.sourceforge.jiu.ops.WrongParameterException
 - **Description copied from net.sourceforge.jiu.ops.Operation (in 19.2.5, page 508)**
This method does the actual work of the operation. It must be called after all parameters have been given to the operation object.
 - **Throws**
 - * net.sourceforge.jiu.ops.WrongParameterException – if at least one of the input parameters was not initialized appropriately (values out of the valid interval, etc.)
 - * net.sourceforge.jiu.ops.MissingParameterException – if any mandatory parameter was not given to the operation
 - * net.sourceforge.jiu.ops.OperationFailedException –

11.1.1.5 Class PromotionRGB48

Converts several image types to (in 14.1.13, page 332). Promotion is a lossless operation that will only lead to an output image that holds the same image in a way that demands more memory. If you give an image implementing RGB24Image to this operation, a WrongParameterException will be thrown. This operation could also return the input image, but this might lead to the wrong impression that a copy of the input was produced which can be modified without changing the original.

Declaration

```
public class PromotionRGB48
  extends net.sourceforge.jiu.ops.ImageToImageOperation (in 19.2.3, page 501)
```

Constructor summary

PromotionRGB48()

Method summary

process()

Constructors

- *PromotionRGB48*
public **PromotionRGB48**()

Methods

- *process*
public void **process**() throws
net.sourceforge.jiu.ops.MissingParameterException,
net.sourceforge.jiu.ops.OperationFailedException,
net.sourceforge.jiu.ops.WrongParameterException
 - **Description copied from net.sourceforge.jiu.ops.Operation (in 19.2.5, page 508)**
This method does the actual work of the operation. It must be called after all parameters have been given to the operation object.
 - **Throws**
 - * net.sourceforge.jiu.ops.WrongParameterException – if at least one of the input parameters was not initialized appropriately (values out of the valid interval, etc.)
 - * net.sourceforge.jiu.ops.MissingParameterException – if any mandatory parameter was not given to the operation
 - * net.sourceforge.jiu.ops.OperationFailedException –

Chapter 12

Package

net.sourceforge.jiu.color.quantization

Package Contents

Page

Interfaces

RGBQuantizer	259
<i>An interface for an RGB color quantizer.</i>	

Classes

ArbitraryPaletteQuantizer	261
<i>A color quantizer that maps an (in 14.1.14, page 333) to any given palette.</i>	
MedianCutContourRemoval	264
<i>Performs the Median Cut color quantization algorithm in combination with a contour removal algorithm.</i>	
MedianCutNode	269
<i>An instance of this node class represents a cuboid part of the color cube representing the three-dimensional RGB color space.</i>	
MedianCutQuantizer	273
<i>Performs the Median Cut color quantization algorithm for a given list of colors.</i>	
OctreeColorQuantizer	279
<i>Performs the octree color quantization algorithm for a given RGB truecolor image.</i>	
OctreeNode	282
<i>A single node in an octree.</i>	
PopularityQuantizer	285
<i>Performs the popularity color quantization algorithm that maps an image to the colors occurring most frequently in the input image.</i>	
RGBColor	288
<i>Encapsulates a single color from RGB (red, green, blue) color space plus a frequency counter.</i>	
RGBColorComparator	290
<i>Compares two (in 12.2.8, page 288) objects.</i>	
RGBColorList	291
<i>Holds an array of (in 12.2.8, page 288) objects.</i>	
UniformPaletteQuantizer	293
<i>A color quantizer that maps to a palette which is equidistantly distributed in the RGB color cube.</i>	

Classes to perform color image quantization, the reduction of the number of unique colors in an image. This is a lossy operation. Usually a number of colors in the destination image is specified by the user (e.g. 256), then the quantization algorithm creates a copy of the input image that has no more than that number of colors. The goal is to be as close to the original as possible. Depending on the actual number of colors specified and the image content this can lead to varying results.

Quantization is usually done to reduce the amount of data necessary to represent the image. The cost for this reduction is a loss of information.

See **the dithering package** (at ../dithering/package-summary.html) for algorithms that improve the result of color image quantization algorithms.

12.1 Interfaces

12.1.1 Interface RGBQuantizer

An interface for an RGB color quantizer. Color quantizers take an input image and produce an output image that looks like the original but uses less colors. Keeping the error (the difference between input and output image) as small as possible is important - the more similar input and output are, the better.

Similarity between two pixels (or, more accurately, the colors of two pixels) can be defined by their distance in color space. Imagine two colors given by (r_1, g_1, b_1) and (r_2, g_2, b_2) . The distance can then be defined as $\text{sqrt}((r_1 - r_2)^2 + (g_1 - g_2)^2 + (b_1 - b_2)^2)$ (with sqrt being the square root).

A quantizer has two main tasks:

1. **Find a palette.** Some quantizers create custom palettes for a given input (e.g. (in 12.2.4, page 273) or (in 12.2.5, page 279), other quantizers use fixed palettes (e.g. (in 12.2.11, page 293)). Using a custom palette typically results in a better output image (it is more similar because it takes into consideration the content of the input). However, using fixed palettes requires less CPU time and memory and is sufficient in many cases from a point of view of output quality.

If a quantizer does use a fixed palette, this first step obviously is not so much about finding the palette but about specifying it.

2. **Map the input image to the palette.** For each pixel in the truecolor input image the mapping procedure must find the color in the palette that is closest to that input pixel so that the difference between source and destination pixel is as small as possible.

The code that does the mapping from the original to any given palette could be shared among quantizers - after all, the goal is always the same, picking the palette entry with the smallest distance in color space to the original pixel. However, sometimes the data structures built while finding the palette can be reused for faster mapping from the original to output. This is the case for both the MedianCutQuantizer and the OctreeColorQuantizer.

Dithering methods like error diffusion dithering may be used to increase the quality of the output. Note however that dithering introduces noise that makes the quantized image harder to compress and in some cases unusable for post-processing (the noise may be an obstacle for image processing algorithms).

This quantizer interface was designed with JIU's error diffusion dithering operation (in 9.2.3, page 234) in mind.

Declaration

```
public interface RGBQuantizer
```

All known subclasses

UniformPaletteQuantizer (in 12.2.11, page 293), PopularityQuantizer (in 12.2.7, page 285), OctreeColorQuantizer (in 12.2.5, page 279), MedianCutQuantizer (in 12.2.4, page 273), ArbitraryPaletteQuantizer (in 12.2.1, page 261)

All classes known to implement interface

UniformPaletteQuantizer (in 12.2.11, page 293), PopularityQuantizer (in 12.2.7, page 285), OctreeColorQuantizer (in 12.2.5, page 279), MedianCutQuantizer (in 12.2.4, page 273), ArbitraryPaletteQuantizer (in 12.2.1, page 261)

Method summary

createPalette() Return a Palette object with the list of colors to be used in the quantization process.

map(int[], int[]) This method maps a triplet of intensity values to its quantized counterpart and returns the palette index of that quantized color.

Methods

- *createPalette*

`net.sourceforge.jiu.data.Palette createPalette()`

- **Description**

Return a Palette object with the list of colors to be used in the quantization process. That palette may be fixed or created specifically for a given input image.

- **Returns** – Palette object for destination image

- *map*

`int map(int[] origRgb, int[] quantizedRgb)`

- **Description**

This method maps a triplet of intensity values to its quantized counterpart and returns the palette index of that quantized color. The index values for the two arrays are taken from RGBIndex.

- **Parameters**

- * **origRgb** – the three samples red, green and blue for which a good match is searched in the palette
- * **quantizedRgb** – will hold the three samples found to be closest to origRgb after the call to this method

- **Returns** – int index in the palette of the match quantizedRgb

12.2 Classes

12.2.1 Class ArbitraryPaletteQuantizer

A color quantizer that maps an (in 14.1.14, page 333) to any given palette. This operation is restricted to RGB24Image and palettes with up to 256 colors. It picks the color from the palette which is closest to the color to be quantized (with the minimum distance). This is a rather naive implementation which, for any given color to be quantized, computes the distance between it and each color in the palette (read: this operation is rather slow with a large palette and input image). It uses **Manhattan distance** (at <http://hiss.nist.gov/dads/HTML/manhtndstnc.html>) (L_1) instead of **Euclidean distance** (at <http://hiss.nist.gov/dads/HTML/euclidndstnc.html>) (L_2). This saves a square root operation per distance computation. There are more sophisticated **nearest neighbor** (at <http://www.cs.sunysb.edu/algorithm/files/nearest-neighbor.shtml>) algorithms available, left for future extensions.

Usage example

This example maps an RGB truecolor image to some palette we create.

```
RGB24Image image = ...; // initialize this
// create some Palette object that you want to map the image to
Palette palette = new Palette(3); // create palette with three entries
palette.put(0, 33, 00, 244); // set first color
palette.put(1, 0, 240, 193); // set second color
palette.put(2, 245, 126, 136); // set third color
ArbitraryPaletteQuantizer quantizer = new ArbitraryPaletteQuantizer(palette);
quantizer.setInputImage(image);
quantizer.process();
PixelImage quantizedImage = quantizer.getOutputImage();
```

Declaration

```
public class ArbitraryPaletteQuantizer
  extends net.sourceforge.jiu.ops.ImageToImageOperation (in 19.2.3, page 501)
  implements net.sourceforge.jiu.data.RGBIndex, RGBQuantizer
```

Constructor summary

ArbitraryPaletteQuantizer(Palette) Creates a quantizer that will be able to map pixels (or a complete image) to the argument palette.

Method summary

createPalette() Returns a copy of the palette that was given to the constructor of this class.
map(int[], int[])
map(int, int, int) Finds the best match for the argument color in the palette and returns its index.
process() Maps the input image to an output image, using the palette given to the constructor.

Constructors

- *ArbitraryPaletteQuantizer*

`public ArbitraryPaletteQuantizer(net.sourceforge.jiu.data.Palette palette)`

- **Description**

Creates a quantizer that will be able to map pixels (or a complete image) to the argument palette.

Methods

- *createPalette*

`public net.sourceforge.jiu.data.Palette createPalette()`

- **Description**

Returns a copy of the palette that was given to the constructor of this class.

- **Returns** – new copy of the palette this quantizer works on

- *map*

`int map(int[] origRgb, int[] quantizedRgb)`

- **Description copied from RGBQuantizer (in 12.1.1, page 259)**

This method maps a triplet of intensity values to its quantized counterpart and returns the palette index of that quantized color. The index values for the two arrays are taken from RGBIndex.

- **Parameters**

- * **origRgb** – the three samples red, green and blue for which a good match is searched in the palette
- * **quantizedRgb** – will hold the three samples found to be closest to origRgb after the call to this method

- **Returns** – int index in the palette of the match quantizedRgb

- *map*

`public int map(int red, int green, int blue)`

- **Description**

Finds the best match for the argument color in the palette and returns its index. Similar to (in 12.2.1, page 262), the quantized color is not required and thus a few assignments are saved by this method.

- **Parameters**

- * **red** – red sample of the pixel to be quantized
- * **green** – green sample of the pixel to be quantized
- * **blue** – blue sample of the pixel to be quantized

- **Returns** – index of the color in the palette that is closest to the argument color

- *process*

`public void process() throws
net.sourceforge.jiu.ops.MissingParameterException,
net.sourceforge.jiu.ops.WrongParameterException`

– **Description**

Maps the input image to an output image, using the palette given to the constructor.

12.2.2 Class MedianCutContourRemoval

Performs the Median Cut color quantization algorithm in combination with a contour removal algorithm.

Quantization is an operation that reduces the number of colors in an image while trying to remain as close to the original image as possible. Standard Median Cut quantization is implemented in the [\(in 12.2.4, page 273\)](#) class.

This class implements an algorithm that improves the standard implementation. It repeatedly calls the original quantizer and adjusts the palette in order to reduce the amount of contouring errors.

Image types

This operation requires an [\(in 14.1.12, page 331\)](#) object as input and produces a [\(in 14.1.8, page 325\)](#) as output.

Usage example

```

RGB24Image inputImage = ...; // image to be processed, from a file etc.
MedianCutQuantizer quantizer = new MedianCutQuantizer();
quantizer.setPaletteSize(256);
MedianCutContourRemoval removal = new MedianCutContourRemoval();
removal.setQuantizer(quantizer);
removal.setInputImage(inputImage);
removal.setTau(11.0);
removal.setNumPasses(3);
removal.process();
PixelImage outputImage = removal.getOutputImage();

```

Rationale - why an extension to Median Cut?

Quantization without dithering can lead to contouring (banding) in the output image. The contours introduced that way are not only ugly but they may lead to erroneous results when processing that quantized image. Dithering, an alternative group of algorithms used in combination with quantizers to improve output quality, leads to output which is more pleasant to the human eye. However, it introduces noise that may not be acceptable when the output image is to be further processed by image processing algorithms. Instead, this algorithm attempts to adjust the palette found by the Median Cut algorithm. The adjustments aim at reducing the amount of contouring caused by a palette found in a previous Median Cut operation.

How the contour removal algorithm works

- First, a normal Median Cut quantization operation is performed. The class [\(in 12.2.4, page 273\)](#) is used for that purpose. This results in a palette and an output image that was mapped from the original using the palette.
- Now a [\(in 5.1.1, page 175\)](#) is created from a [\(in 5.1.2, page 178\)](#), which is in turn created from the palleted image that was produced in the previous step. The co-occurrence frequency matrix stores how often a pixel value *j* is the neighbor of pixel *i* in the image, in relation to

all occurrences of *i*. The matrix stores this information as `double` values between 0.0 and 1.0. If the value is 0.6, *j* makes 60 percent of all neighboring pixels of *i*.

- Using certain heuristics that take advantage of the above mentioned matrices, colors are classified into three groups:
 - contouring color pairs which contribute significantly to the contouring,
 - compressible color pairs, two colors which are similar to each other and not contouring, and
 - all colors which are not part of one of the two color pair types described before.

The tau value which can be specified with `tau` (in 12.2.2, page 267) is a distance value in RGB space (tau is adjusted for an RGB cube where each axis can take values from 0 to 255). It is used to define a threshold for similar colors. A pair of compressible colors may not differ by more than tau. It is guaranteed that no color can be both compressible and contouring.

- Note that each palette color generated by the Median Cut algorithm represents a cuboid part of the RGB color cube. For each pair of compressible colors, their corresponding cuboids are neighbors in the cube. Now a number *N* of palette entries is changed. That number *N* is either the number of compressible color pairs found, or the number of contouring color pairs found, whichever is smaller. If *N* equals zero, nothing can be done and the algorithm terminates.
- Now *N* swap operations are performed on the palette. Two palette entries of a compressible color pair are merged to form one palette entry. Their colors are similar so it will not decrease image quality much. The newly freed palette entry is used to split one color of a contouring color pair into two, thus allowing to represent a gradient type section in the image with an additional color.
- The original truecolor image is mapped to the new, modified palette. This whole process can now be performed again with the modified palette. That's why this operation has a `repeat` (in 12.2.2, page 267) method. Usually, more than eight iterations do not make a difference.

For an in-depth description of the algorithm see the journal article mentioned in the Credits section below.

Credits

The algorithm was developed by **Jefferey Shufelt** (at <http://www-2.cs.cmu.edu/afs/cs.cmu.edu/user/js/www/homepage.html>) and described in his article Texture Analysis for Enhanced Color Image Quantization. CVGIP: Graphical Model and Image Processing 59(3): 149-163 (1997).

See also

- `MedianCutQuantizer` (in 12.2.4, page 273)

Declaration

```
public class MedianCutContourRemoval
  extends net.sourceforge.jiu.ops.ImageToImageOperation (in 19.2.3, page 501)
```

Field summary

DEFAULT_NUM_PASSES The default number of passes, used if they are not specified with (in 12.2.2, page 267).

DEFAULT_TAU The default tau value, used if none is specified with (in 12.2.2, page 267).

Constructor summary

MedianCutContourRemoval()

Method summary

main(String[]) Small command line application that performs a contour removal on an image.

process()

setNumPasses(int) Specify the number of contour removal passes to be performed.

setQuantizer(MedianCutQuantizer) Set the (in 12.2.4, page 273) object to be used with this contour removal operation.

setTau(double) Specify the tau value to be used by this operation.

Fields

- public static final double **DEFAULT_TAU**
 - The default tau value, used if none is specified with (in 12.2.2, page 267). Check the class documentation to find out more about the meaning of tau: (in 12.2.2, page 264).
- public static final int **DEFAULT_NUM_PASSES**
 - The default number of passes, used if they are not specified with (in 12.2.2, page 267). Check the class documentation to find out more about the meaning of that number of passes: (in 12.2.2, page 264).

Constructors

- *MedianCutContourRemoval*
public **MedianCutContourRemoval()**

Methods

- *main*
public static void **main**(java.lang.String[] args) throws java.lang.Exception
 - **Description**
Small command line application that performs a contour removal on an image. The first and only argument must be the name of image file from which the image to be quantized is loaded.
 - **Parameters**

- * **args** – program arguments; must have length one, the only argument being the input image file name
 - **Throws**
 - * `java.lang.Exception` –

- *process*

```
public void process( ) throws
net.sourceforge.jiu.ops.MissingParameterException,
net.sourceforge.jiu.ops.OperationFailedException,
net.sourceforge.jiu.ops.WrongParameterException
```

 - **Description copied from `net.sourceforge.jiu.ops.Operation` (in 19.2.5, page 508)**
This method does the actual work of the operation. It must be called after all parameters have been given to the operation object.
 - **Throws**
 - * `net.sourceforge.jiu.ops.WrongParameterException` – if at least one of the input parameters was not initialized appropriately (values out of the valid interval, etc.)
 - * `net.sourceforge.jiu.ops.MissingParameterException` – if any mandatory parameter was not given to the operation
 - * `net.sourceforge.jiu.ops.OperationFailedException` –

- *setNumPasses*

```
public void setNumPasses( int newValue )
```

 - **Description**
Specify the number of contour removal passes to be performed. Check out the section **How the contour removal algorithm works** (at [#howitworks](#)) to learn more about the meaning of this value. If this method is not called the default value (in 12.2.2, page 266) is used.
 - **Parameters**
 - * **newValue** – number of passes, 1 or higher
 - **Throws**
 - * `java.lang.IllegalArgumentException` – if the argument is 0 or smaller

- *setQuantizer*

```
public void setQuantizer( MedianCutQuantizer medianCutQuantizer )
```

 - **Description**
Set the (in 12.2.4, page 273) object to be used with this contour removal operation. This is a mandatory parameter. If process gets called before the quantizer object was specified, a (in 19.3.1, page 512) is thrown.
 - **Parameters**
 - * **medianCutQuantizer** – the quantizer object that will get used by this operation

- *setTau*

```
public void setTau( double newValue )
```

- **Description**

Specify the tau value to be used by this operation. Check out the section **How the contour removal algorithm works** (at [#howitworks](#)) to learn more about the meaning of this value. If this method is not called the default value (in 12.2.2, page 266) is used.

- **Parameters**

- * `newValue` – tau value, 0.0 or higher

- **Throws**

- * `java.lang.IllegalArgumentException` – if the argument is smaller than 0.0

12.2.3 Class MedianCutNode

An instance of this node class represents a cuboid part of the color cube representing the three-dimensional RGB color space.

See also

- `MedianCutQuantizer` (in 12.2.4, page 273)

Declaration

```
public class MedianCutNode
  extends java.lang.Object
  implements net.sourceforge.jiu.data.RGBIndex
```

Constructor summary

MedianCutNode(MedianCutNode, int, int) Creates a node for a Median Cut tree of nodes with index values for some external color array and the parent node.

Method summary

canBeSplit() Returns if this node can be split into two.

computeRgbDistance(MedianCutNode) Computes the distance in RGB color space between the representative color of this node and the argument node and returns it as non-negative value.

getAxisOfLargestDistribution() Returns the axis of the channel whose samples are most widely distributed among the colors that belong to this node.

getDifferenceOfLargestDistribution()

getLeftIndex()

getLeftSuccessor() Returns left successor node (or null if this node is a leaf).

getMaxColorSample(int)

getMedianValue()

getMiddleIndex()

getMinColorSample(int)

getNumColors()

getPaletteIndex()

getParentNode() Returns parent node (or null if this node is the root node).

getRepresentativeColor()

getRightIndex()

getRightSuccessor() Returns right successor node (or null if this node is a leaf).

getSuccessor(int[])

isAxisDetermined()

isLeaf() Returns if this node is a leaf by checking if both successors are null.

setLargestDistribution(int, int)

setMaxColor(int, int, int)

setMaxColorSample(int, int)

setMedianValue(int)

setMinColor(int, int, int)

setMinColorSample(int, int)

setPaletteIndex(int)

setRepresentativeColor(int[])

setSuccessors(MedianCutNode, MedianCutNode) Sets the successor nodes for this node.

Constructors

- *MedianCutNode*

public MedianCutNode(MedianCutNode parent, int index1, int index2)

- **Description**

Creates a node for a Median Cut tree of nodes with index values for some external color array and the parent node. This parent is null for the root node.

- **Parameters**

- * **parent** – the parent node of this new node, should be null only for the root node
- * **index1** – the index value of the first element of colors in the color list
- * **index2** – the index value of the last element of colors in the color list; must be larger than or equal to index1

- **Throws**

- * **java.lang.IllegalArgumentException** – if index1 is larger than index2

Methods

- *canBeSplit*

public boolean canBeSplit()

- **Description**

Returns if this node can be split into two. This is true if and only if this is a leaf and if the color list index values represent an interval of at least length 2.

- **Returns** – if this node can be split into two nodes

- *computeRgbDistance*

public double computeRgbDistance(MedianCutNode node)

- **Description**

Computes the distance in RGB color space between the representative color of this node and the argument node and returns it as non-negative value.

- *getAxisOfLargestDistribution*

public int getAxisOfLargestDistribution()

- **Description**

Returns the axis of the channel whose samples are most widely distributed among the colors that belong to this node.

- **Returns** – index of axis, one of the (in 14.1.15, page 334) constants

- **Throws**

- * **java.lang.IllegalArgumentException** – if that axis has not been determined

- *getDifferenceOfLargestDistribution*

public int getDifferenceOfLargestDistribution()

- *getLeftIndex*

public int getLeftIndex()

- *getLeftSuccessor*
public MedianCutNode getLeftSuccessor()

 - **Description**
Returns left successor node (or null if this node is a leaf).

- *getMaxColorSample*

public int getMaxColorSample(int index)

- *getMedianValue*

public int getMedianValue()

- *getMiddleIndex*

public int getMiddleIndex()

- *getMinColorSample*

public int getMinColorSample(int index)

- *getNumColors*

public int getNumColors()

- *getPaletteIndex*

public int getPaletteIndex()

- *getParentNode*
public MedianCutNode getParentNode()

 - **Description**
Returns parent node (or null if this node is the root node).

- *getRepresentativeColor*

public int[] getRepresentativeColor()

- *getRightIndex*

public int getRightIndex()

- *getRightSuccessor*
public MedianCutNode getRightSuccessor()

 - **Description**
Returns right successor node (or null if this node is a leaf).

- *getSuccessor*

public MedianCutNode getSuccessor(int[] rgb)

- *isAxisDetermined*

public boolean isAxisDetermined()

- *isLeaf*
public boolean isLeaf()

 - **Description**
Returns if this node is a leaf by checking if both successors are null. Note that the case of one successor being null and the other non-null should never happen.

– **Returns** – if this node is a leaf (true)

• *setLargestDistribution*

public void setLargestDistribution(int newAxis, int newDifference)

• *setMaxColor*

public void setMaxColor(int red, int green, int blue)

• *setMaxColorSample*

public void setMaxColorSample(int index, int value)

• *setMedianValue*

public void setMedianValue(int newMedianValue)

• *setMinColor*

public void setMinColor(int red, int green, int blue)

• *setMinColorSample*

public void setMinColorSample(int index, int value)

• *setPaletteIndex*

public void setPaletteIndex(int newPaletteIndex)

• *setRepresentativeColor*

public void setRepresentativeColor(int[] aRepresentativeColor)

• *setSuccessors*

public void setSuccessors(MedianCutNode left, MedianCutNode right)

– **Description**

Sets the successor nodes for this node. The successors must be either both null or both initialized. They must not be equal.

– **Parameters**

- * **left** – the left successor node
- * **right** – the left successor node

12.2.4 Class MedianCutQuantizer

Performs the Median Cut color quantization algorithm for a given list of colors.

Supported image types

The input image must implement (in 14.1.12, page 331).

Usage example

The following code snippet uses the default settings with a palette of 256 entries.

```
MedianCutQuantizer quantizer = new MedianCutQuantizer();
quantizer.setInputImage(image);
quantizer.setPaletteSize(256);
quantizer.process();
PixelImage quantizedImage = quantizer.getOutputImage();
```

If you want to combine Median Cut quantization with error diffusion dithering to improve the visual quality of the output, try the (in 9.2.3, page 234)class. However, note that noise is introduced into the image with dithering methods so that the resulting image may not be suitable for automatic processing.

Credits

The Median Cut algorithm was designed by **Paul Heckbert** (at <http://www.cs.cmu.edu/~ph>). He described it in the article Color image quantization for frame buffer display. Comput. Graphics 16(3), 1982, 297 - 304. **CiteSeer page of the article** (at <http://citeseer.nj.nec.com/heckbert80color.html>).

See also

- `MedianCutContourRemoval` (in 12.2.2, page 264)
- `net.sourceforge.jiu.color.dithering.ErrorDiffusionDithering` (in 9.2.3, page 234)

Declaration

```
public class MedianCutQuantizer
extends net.sourceforge.jiu.ops.ImageToImageOperation (in 19.2.3, page 501)
implements net.sourceforge.jiu.data.RGBIndex, RGBQuantizer
```

Field summary

- DEFAULT_METHOD_REPR_COLOR** The default method to determine the representative color from a list of colors.
- METHOD_REPR_COLOR_AVERAGE** Constant value for a method of determining the representative color for a set of colors by computing the average of all samples for each of the three components red, green and blue.
- METHOD_REPR_COLOR_MEDIAN** Constant value for a method of determining the representative color for a set of colors by picking the median value of all samples for each of the three components red, green and blue.
- METHOD_REPR_COLOR_WEIGHTED_AVERAGE** Constant value for a method of determining the representative color for a set of colors by computing the

weighted average of all samples for each of the three components red, green and blue.

Constructor summary

MedianCutQuantizer() Creates a MedianCutQuantizer object and initializes its fields to default values.

Method summary

createLeafList() Creates a linear list of leaf nodes.
createPalette() Creates a palette with the representative colors of all leaf nodes.
findAllRepresentativeColors()
findNearestNeighbor(int[]) For a given RGB value, searches the node in the internal node tree whose representative color is closest to this color.
findNearestNeighbor(MedianCutNode[], int, int, int) For each node in the argument array computes the distance between the representative color of that node and the color given by the three argument samples.
findPalette()
getMethodToDetermineRepresentativeColors() Returns the method (to be) used to determine the representative color for the list of colors of a node.
getPaletteSize() Returns the number of colors in the destination image.
getTruecolorOutput() Returns if this operation is supposed to generate truecolor or paletted output.
map(int[], int[])
mapImage(RGB24Image, Paletted8Image)
mapImage(RGB24Image, RGB24Image)
process()
setAllPaletteIndexValues()
setMapping(boolean) Defines whether process will map the input image to an output image.
setMethodToDetermineRepresentativeColors(int) Sets the method to determine the representative color for a list of colors.
setPaletteSize(int) Sets the number of colors that this operations is supposed to reduce the original image to.
setTruecolorOutput(boolean) Lets the user specify if the output image is to be truecolor (argument useTruecolor is true) or paletted (argument useTruecolor is false).
splitNode(MedianCutNode)

Fields

- public static final int **METHOD_REPR_COLOR_AVERAGE**
 - Constant value for a method of determining the representative color for a set of colors by computing the average of all samples for each of the three components red, green and blue. #getMethodToDetermineRepresentativeColors
#setMethodToDetermineRepresentativeColors
- public static final int **METHOD_REPR_COLOR_WEIGHTED_AVERAGE**

- Constant value for a method of determining the representative color for a set of colors by computing the weighted average of all samples for each of the three components red, green and blue. Weighted means that each color is multiplied by the number of times it occurs in the input image. The values of samples multiplied by their frequency are then divided by the total number of times the colors appear in the image.
`#getMethodToDetermineRepresentativeColors`
`#setMethodToDetermineRepresentativeColors`
- public static final int **METHOD_REPR_COLOR_MEDIAN**
 - Constant value for a method of determining the representative color for a set of colors by picking the median value of all samples for each of the three components red, green and blue. `#getMethodToDetermineRepresentativeColors`
`#setMethodToDetermineRepresentativeColors`
- public static final int **DEFAULT_METHOD_REPR_COLOR**
 - The default method to determine the representative color from a list of colors. Will be used if none is set by the user of this class via (in 12.2.4, page 277).

Constructors

- *MedianCutQuantizer*
`public MedianCutQuantizer()`
 - **Description**
Creates a MedianCutQuantizer object and initializes its fields to default values.

Methods

- *createLeafList*
`public MedianCutNode[] createLeafList()`
 - **Description**
Creates a linear list of leaf nodes. Assumes that (in 12.2.4, page 276) was successfully run before.
- *createPalette*
`public net.sourceforge.jiu.data.Palette createPalette()`
 - **Description**
Creates a palette with the representative colors of all leaf nodes. Assumes that (in 12.2.4, page 276) was successfully run before.
 - **Returns** – palette with all representative colors
- *findAllRepresentativeColors*
`public void findAllRepresentativeColors()`
- *findNearestNeighbor*
`public MedianCutNode findNearestNeighbor(int[] rgb)`

– **Description**

For a given RGB value, searches the node in the internal node tree whose representative color is closest to this color.

– **Parameters**

* **rgb** – the color for which a match is searched; the array must have at least three entries; (in 14.1.15, page 334) constants are used to address the samples

– **Returns** – node with best match

• *findNearestNeighbor*

```
public int findNearestNeighbor( MedianCutNode[] nodes, int red, int green,
int blue )
```

– **Description**

For each node in the argument array computes the distance between the representative color of that node and the color given by the three argument samples.

– **Returns** – index of the node with the smallest distance or -1 if the array has a length of 0

• *findPalette*

```
public void findPalette( )
```

• *getMethodToDetermineRepresentativeColors*

```
public int getMethodToDetermineRepresentativeColors( )
```

– **Description**

Returns the method (to be) used to determine the representative color for the list of colors of a node. Default is (in 12.2.4, page 275).

– **Returns** – the method, one of the METHOD_xyz constants

• *getPaletteSize*

```
public int getPaletteSize( )
```

– **Description**

Returns the number of colors in the destination image. If output is paletted, this is also the number of entries in the palette.

– **Returns** – number of colors in the destination

• *getTruecolorOutput*

```
public boolean getTruecolorOutput( )
```

– **Description**

Returns if this operation is supposed to generate truecolor or paletted output.

– **Returns** – if truecolor images are to be generated

– **See also**

* `MedianCutQuantizer.setTruecolorOutput(boolean)` (in 12.2.4, page 278)

• *map*

```
int map( int[] origRgb, int[] quantizedRgb )
```

– **Description copied from RGBQuantizer (in 12.1.1, page 259)**

This method maps a triplet of intensity values to its quantized counterpart and returns the palette index of that quantized color. The index values for the two arrays are taken from RGBIndex.

– **Parameters**

- * **origRgb** – the three samples red, green and blue for which a good match is searched in the palette
- * **quantizedRgb** – will hold the three samples found to be closest to origRgb after the call to this method

– **Returns** – int index in the palette of the match quantizedRgb

• *mapImage*

```
public void mapImage( net.sourceforge.jiu.data.RGB24Image in,
net.sourceforge.jiu.data.Paletted8Image out )
```

• *mapImage*

```
public void mapImage( net.sourceforge.jiu.data.RGB24Image in,
net.sourceforge.jiu.data.RGB24Image out )
```

• *process*

```
public void process( ) throws
net.sourceforge.jiu.ops.MissingParameterException,
net.sourceforge.jiu.ops.OperationFailedException,
net.sourceforge.jiu.ops.WrongParameterException
```

– **Description** copied from net.sourceforge.jiu.ops.Operation (in 19.2.5, page 508)

This method does the actual work of the operation. It must be called after all parameters have been given to the operation object.

– **Throws**

- * net.sourceforge.jiu.ops.WrongParameterException – if at least one of the input parameters was not initialized appropriately (values out of the valid interval, etc.)
 - * net.sourceforge.jiu.ops.MissingParameterException – if any mandatory parameter was not given to the operation
 - * net.sourceforge.jiu.ops.OperationFailedException –
-

• *setAllPaletteIndexValues*

```
public void setAllPaletteIndexValues( )
```

• *setMapping*

```
public void setMapping( boolean doMap )
```

– **Description**

Defines whether process will map the input image to an output image. If not, only the palette is determined.

• *setMethodToDetermineRepresentativeColors*

```
public void setMethodToDetermineRepresentativeColors( int newMethod )
```

– **Description**

Sets the method to determine the representative color for a list of colors. After the algorithm has determined sets of colors that lie closely together in color space and will be mapped to the same color in the destination image, the algorithm will determine that color

– **Parameters**

- * **newMethod** – the new method, one of the METHOD_xyz constants in this class

- *setPaletteSize*

```
public void setPaletteSize( int newPaletteSize )
```

- **Description**

Sets the number of colors that this operations is supposed to reduce the original image to.

- **Parameters**

- * `newPaletteSize` – the number of colors

- **Throws**

- * `java.lang.IllegalArgumentException` – if the argument is smaller than 1 or larger than 256

- **See also**

- * `MedianCutQuantizer.getPaletteSize()` (in 12.2.4, page 276)

- *setTruecolorOutput*

```
public void setTruecolorOutput( boolean useTruecolor )
```

- **Description**

Lets the user specify if the output image is to be truecolor (argument `useTruecolor` is `true`) or paletted (argument `useTruecolor` is `false`). If the color type is to be changed afterwards, use `PromoteToRgb24` to convert from paletted to truecolor. Reducing a truecolor image that uses only 256 or less colors to a paletted image can be done with `AutoDetectColorType`.

- **Parameters**

- * `useTruecolor` –

- *splitNode*

```
public void splitNode( MedianCutNode node )
```

12.2.5 Class OctreeColorQuantizer

Performs the octree color quantization algorithm for a given RGB truecolor image. The quality is usually somewhat inferior to the results of (in 12.2.4, page 273). Note that you can improve the quality by applying a dithering algorithm. See (in 9.2.3, page 234).

Usage example

This reduces some RGB24Image image to a 16 color paletted image:

```
MemoryRGB24Image image = ...; // initialize
OctreeColorQuantizer ocq = new OctreeColorQuantizer();
ocq.setInputImage(image);
ocq.setPaletteSize(16);
ocq.process();
PixelImage quantizedImage = ocq.getOutputImage();
```

Credits

Declaration

```
public class OctreeColorQuantizer
extends net.sourceforge.jiu.ops.ImageToImageOperation (in 19.2.3, page 501)
implements net.sourceforge.jiu.data.RGBIndex, RGBQuantizer
```

Field summary

DEFAULT_PALETTE_SIZE The default number of colors in the palette.

Constructor summary

OctreeColorQuantizer()

Method summary

createPalette()

init() Creates an octree and prepares this quantizer so that colors can be mapped to palette index values.

map(int[], int[]) Maps an RGB color origRgb to one of the colors in the color map; that color will be written to quantizedRgb and its palette index will be returned.

process() Initializes an octree, reduces it have as many leaves (or less) as the desired palette size and maps the original image to the newly-created palette.

setPaletteSize(int)

Fields

- public static final int **DEFAULT_PALETTE_SIZE**
 - The default number of colors in the palette. Will be used when no other value is specified via (in 12.2.5, page 281).

Constructors

- *OctreeColorQuantizer*
`public OctreeColorQuantizer()`

Methods

- *createPalette*
`net.sourceforge.jiu.data.Palette createPalette()`
 - **Description copied from RGBQuantizer (in 12.1.1, page 259)**
Return a Palette object with the list of colors to be used in the quantization process. That palette may be fixed or created specifically for a given input image.
 - **Returns** – Palette object for destination image
- *init*
`public void init() throws net.sourceforge.jiu.ops.MissingParameterException, net.sourceforge.jiu.ops.WrongParameterException`
 - **Description**
Creates an octree and prepares this quantizer so that colors can be mapped to palette index values. If you use (in 12.2.5, page 280) you must not call this method. On the other hand, if you want to do the mapping yourself - maybe if you want to do mapping and dithering interchangeably - call this method first, then do the mapping yourself.
 - **Throws**
 - * `net.sourceforge.jiu.ops.MissingParameterException` – if parameters like the input image are missing
 - * `net.sourceforge.jiu.ops.WrongParameterException` – if parameters exist but are of the wrong type
- *map*
`public int map(int[] origRgb, int[] quantizedRgb)`
 - **Description**
Maps an RGB color `origRgb` to one of the colors in the color map; that color will be written to `quantizedRgb` and its palette index will be returned.
 - **Parameters**
 - * `origRgb` – the color to be mapped to the best-possible counterpart in the palette; the array is indexed by the constants from (in 14.1.15, page 334)
 - * `quantizedRgb` – the resulting color from the palette will be written to this array; it is also indexed by the constants from (in 14.1.15, page 334)
 - **Returns** – index of the found color in the palette
- *process*
`public void process() throws net.sourceforge.jiu.ops.MissingParameterException, net.sourceforge.jiu.ops.WrongParameterException`
 - **Description**
Initializes an octree, reduces it have as many leaves (or less) as the desired palette size and maps the original image to the newly-created palette.

- *setPaletteSize*

```
public void setPaletteSize( int newPaletteSize )
```

12.2.6 Class OctreeNode

A single node in an octree.

See also

- `OctreeColorQuantizer` (in 12.2.5, page 279)

Declaration

```
public class OctreeNode
  extends java.lang.Object
  implements net.sourceforge.jiu.util.ComparatorInterface, net.sourceforge.jiu.data.RGBIndex
```

Constructor summary

`OctreeNode()`

Method summary

`add(OctreeNode, int, int, int, int)` Add a color red-green-blue to the octree, given by its root node.

`compare(Object, Object)`

`copyChildSums()` Adds the sums for red, green and blue values and the pixel count values of all child nodes and stores the results in this node.

`determineRepresentativeColor()`

`getBlue()`

`getChildren()`

`getGreen()`

`getNumChildren()`

`getPaletteIndex()`

`getRed()`

`isLeaf()`

`map(int[], int[])` Returns the index of the best match for origRgb in the palette or -1 if the best match could not be determined.

`setChildren(OctreeNode[])`

`setPaletteIndex(int)`

Constructors

- *OctreeNode*
public **OctreeNode**()

Methods

- *add*
public static boolean **add**(OctreeNode root, int red, int green, int blue, int bitsPerSample)

– **Description**

Add a color red-green-blue to the octree, given by its root node. This methods follows the octree down to the bitsPerSample'th level, creating nodes as necessary. Increases the pixelCount of a leaf node (if the node already exists) or initializes a newly-created leaf.

– **Parameters**

- * **root** – root node of the octree
 - * **red** – the red intensity value of the color to be added
 - * **green** – the green intensity value of the color to be added
 - * **blue** – the blue intensity value of the color to be added
 - * **bitsPerSample** –
-

- *compare*

```
int compare( java.lang.Object o1, java.lang.Object o2 )
```

– **Description copied from net.sourceforge.jiu.util.ComparatorInterface (in 20.1.1, page 516)**

Compares the two argument objects and returns their relation. Returns

- * a value <0 if o1 is smaller than o2,
 - * 0 if o1 is equal to o2 and
 - * a value >0 if o1 is greater than o2.
-

- *copyChildSums*

```
public void copyChildSums( )
```

– **Description**

Adds the sums for red, green and blue values and the pixel count values of all child nodes and stores the results in this node. Does nothing if this is a leaf. Otherwise, recursively calls itself with all non-null child nodes and adds their sums for red, green and blue and the number of pixel values. Then stores these values in this node. They will be used when the octree is pruned to have a certain number of leaves.

- *determineRepresentativeColor*

```
public void determineRepresentativeColor( )
```

- *getBlue*

```
public int getBlue( )
```

- *getChildren*

```
public OctreeNode[] getChildren( )
```

- *getGreen*

```
public int getGreen( )
```

- *getNumChildren*

```
public int getNumChildren( )
```

- *getPaletteIndex*

```
public int getPaletteIndex( )
```

- *getRed*

```
public int getRed( )
```

- *isLeaf*

```
public boolean isLeaf( )
```

- *map*

```
public int map( int[] origRgb, int[] quantizedRgb )
```

- **Description**

Returns the index of the best match for origRgb in the palette or -1 if the best match could not be determined. If there was a best match, quantizedRgb is filled with the quantized color's RGB values.

- *setChildren*

```
public void setChildren( OctreeNode[] newChildren )
```

- *setPaletteIndex*

```
public void setPaletteIndex( int index )
```

12.2.7 Class PopularityQuantizer

Performs the popularity color quantization algorithm that maps an image to the colors occurring most frequently in the input image. The number of colors in the palette can be defined by the user of this operation with (in 12.2.7, page 287).

Supported image types

The input image must implement (in 14.1.12, page 331), the output image must be of type (in 14.1.8, page 325).

Usage example

The following code snippet uses the default settings with a palette of 256 entries.

```
PopularityQuantizer quantizer = new PopularityQuantizer();
quantizer.setInputImage(image);
quantizer.setPaletteSize(256);
quantizer.process();
PixelImage quantizedImage = quantizer.getOutputImage();
```

See also

- ArbitraryPaletteQuantizer (in 12.2.1, page 261)

Declaration

```
public class PopularityQuantizer
  extends net.sourceforge.jiu.ops.ImageToImageOperation (in 19.2.3, page 501)
  implements net.sourceforge.jiu.data.RGBIndex, RGBQuantizer
```

Constructor summary

PopularityQuantizer()

Method summary

createPalette()
getPaletteSize() Returns the number of colors in the destination image.
map(int[], int[])
process()
setMapping(boolean) Specifies whether this operation will map the image to the new palette (true) or not (false).
setPaletteSize(int) Sets the number of colors that this operations is supposed to reduce the original image to.

Constructors

- *PopularityQuantizer*
public PopularityQuantizer()

Methods

- *createPalette*

```
net.sourceforge.jiu.data.Palette createPalette( )
```

- **Description copied from RGBQuantizer (in 12.1.1, page 259)**

Return a Palette object with the list of colors to be used in the quantization process. That palette may be fixed or created specifically for a given input image.

- **Returns** – Palette object for destination image
-

- *getPaletteSize*

```
public int getPaletteSize( )
```

- **Description**

Returns the number of colors in the destination image. If output is paletted, this is also the number of entries in the palette.

- **Returns** – number of colors in the destination

- **See also**

* `PopularityQuantizer.setPaletteSize(int)` (in 12.2.7, page 287)

- *map*

```
int map( int[] origRgb, int[] quantizedRgb )
```

- **Description copied from RGBQuantizer (in 12.1.1, page 259)**

This method maps a triplet of intensity values to its quantized counterpart and returns the palette index of that quantized color. The index values for the two arrays are taken from RGBIndex.

- **Parameters**

* `origRgb` – the three samples red, green and blue for which a good match is searched in the palette
 * `quantizedRgb` – will hold the three samples found to be closest to `origRgb` after the call to this method

- **Returns** – int index in the palette of the match `quantizedRgb`
-

- *process*

```
public void process( ) throws
```

```
net.sourceforge.jiu.ops.MissingParameterException,  
net.sourceforge.jiu.ops.OperationFailedException,  
net.sourceforge.jiu.ops.WrongParameterException
```

- **Description copied from net.sourceforge.jiu.ops.Operation (in 19.2.5, page 508)**

This method does the actual work of the operation. It must be called after all parameters have been given to the operation object.

- **Throws**

* `net.sourceforge.jiu.ops.WrongParameterException` – if at least one of the input parameters was not initialized appropriately (values out of the valid interval, etc.)
 * `net.sourceforge.jiu.ops.MissingParameterException` – if any mandatory parameter was not given to the operation

* net.sourceforge.jiu.ops.OperationFailedException –

- *setMapping*

public void setMapping(boolean newValue)

- **Description**

Specifies whether this operation will map the image to the new palette (true) or not (false). The latter may be interesting if only the palette is required. By default, this operation does map.

- **Parameters**

- * **newValue** – map to new image (true) or just search palette (false)

- *setPaletteSize*

public void setPaletteSize(int newPaletteSize)

- **Description**

Sets the number of colors that this operations is supposed to reduce the original image to.

- **Parameters**

- * **newPaletteSize** – the number of colors

- **Throws**

- * **java.lang.IllegalArgumentException** – if the argument is smaller than 1 or larger than 256

- **See also**

- * **PopularityQuantizer.getPaletteSize()** (in 12.2.7, page 286)

12.2.8 Class RGBColor

Encapsulates a single color from RGB (red, green, blue) color space plus a frequency counter. Each of the three RGB samples is of type int. Also stores a counter of type int.

Declaration

```
public class RGBColor
  extends java.lang.Object
  implements net.sourceforge.jiu.data.RGBIndex
```

Constructor summary

RGBColor(int, int, int) Creates an instance of this class and initializes it to the given intensity values.

RGBColor(int, int, int, int) Creates an instance of this class and initializes it to the given intensity values.

Method summary

compareTo(RGBColor, int) Compares this color to the argument color, using the sortOrder argument (which is one of the three index values defined in (in 14.1.15, page 334).

computeDistance(int, int, int, int, int, int) For two RGB triplets (r1, g1, b1) and (r2, g2, b2) this will return the distance between those colors in RGB color space.

equals(Object) Compares this color with another instance of RGBColor and returns true if all intensity values are equal, false otherwise.

getCounter()

getSample(int)

toString()

Constructors

- *RGBColor*

```
public RGBColor( int red, int green, int blue )
```

- **Description**

Creates an instance of this class and initializes it to the given intensity values. The internal color counter is set to zero.

- *RGBColor*

```
public RGBColor( int red, int green, int blue, int counter )
```

- **Description**

Creates an instance of this class and initializes it to the given intensity values. Also sets the internal color counter to the given parameter.

Methods

- *compareTo*

```
public int compareTo( RGBColor c, int sortOrder )
```

- **Description**

Compares this color to the argument color, using the sortOrder argument (which is one of the three index values defined in (in 14.1.15, page 334). That way, the two sample values for one component (e.g. red if sortOrder == INDEX.RED) are compared.

- **Parameters**

- * **c** – the color to which this color is compared
 - * **sortOrder** – the component used for the comparison

- **Returns** – relation between this color and the argument color

- *computeDistance*

```
public static double computeDistance( int r1, int g1, int b1, int r2, int g2, int b2 )
```

- **Description**

For two RGB triplets (r1, g1, b1) and (r2, g2, b2) this will return the distance between those colors in RGB color space.

- *equals*

```
public boolean equals( java.lang.Object obj )
```

- **Description**

Compares this color with another instance of RGBColor and returns true if all intensity values are equal, false otherwise.

- *getCounter*

```
public int getCounter( )
```

- *getSample*

```
public int getSample( int index )
```

- *toString*

```
public java.lang.String toString( )
```

12.2.9 Class RGBColorComparator

Compares two (in 12.2.8, page 288)objects.

Declaration

```
public class RGBColorComparator
  extends java.lang.Object
  implements net.sourceforge.jiu.util.ComparatorInterface, net.sourceforge.jiu.data.RGBIndex
```

Constructor summary

RGBColorComparator(int)

Method summary

compare(Object, Object)
setSortOrder(int) Sets the internal sort order (it is sorted by one of the three RGB components) to the parameter.

Constructors

- *RGBColorComparator*
 public **RGBColorComparator**(int aSortOrder)

Methods

- *compare*
 int **compare**(java.lang.Object o1, java.lang.Object o2)
 – **Description copied from net.sourceforge.jiu.util.ComparatorInterface (in 20.1.1, page 516)**
 Compares the two argument objects and returns their relation. Returns
 * a value <0 if o1 is smaller than o2,
 * 0 if o1 is equal to o2 and
 * a value >0 if o1 is greater than o2.
- *setSortOrder*
 public void **setSortOrder**(int aSortOrder)
 – **Description**
 Sets the internal sort order (it is sorted by one of the three RGB components) to the parameter.

12.2.10 Class RGBColorList

Holds an array of (in 12.2.8, page 288)objects.

Declaration

```
public class RGBColorList
  extends java.lang.Object
  implements net.sourceforge.jiu.data.RGBIndex
```

Constructor summary

RGBColorList(Histogram3D) Creates a new list and initializes it with the argument histogram.

Method summary

findExtrema(int, int) In a given interval of the list this method searches for the color axis that has the largest distribution of values.

getColor(int) Returns an (in 12.2.8, page 288)object from this list, given by its zero-based index value.

getNumEntries() Returns the number of color objects in this list.

sortByAxis(int, int, int) Sorts an interval of the array of colors by one of the three components (RGB).

sortByCounter(int, int) Sorts an interval of the array of colors by their counters.

Constructors

- *RGBColorList*

```
public RGBColorList( net.sourceforge.jiu.color.data.Histogram3D hist )
```

- **Description**

Creates a new list and initializes it with the argument histogram. All values from the histogram with a counter larger than zero will be added to the list (which will include all colors that appear at least once in the image on which the histogram was created).

- **Parameters**

- * **hist** – the histogram from which the list will be initialized

- **Throws**

- * **java.lang.IllegalArgumentException** – thrown if no histogram entry has a non-zero counter

Methods

- *findExtrema*

```
public int[] findExtrema( int i1, int i2 )
```

- **Description**

In a given interval of the list this method searches for the color axis that has the largest distribution of values. Returns a pair of int values; the first value is the component (0, 1 or 2), the second value is the difference between the minimum and maximum value found in the list. Only checks colors from index i1 to i2 of the list.

- *getColor*

```
public RGBColor getColor( int index )
```

- **Description**

- Returns an (in 12.2.8, page 288) object from this list, given by its zero-based index value.

- **Parameters**

- * **index** – zero-based index into the list; must be smaller than (in 12.2.10, page 292)

- **Returns** – the color object

- *getNumEntries*

```
public int getNumEntries( )
```

- **Description**

- Returns the number of color objects in this list.

- **Returns** – number of colors in the list

- *sortByAxis*

```
public void sortByAxis( int index1, int index2, int axis )
```

- **Description**

- Sorts an interval of the array of colors by one of the three components (RGB).

- **Parameters**

- * **index1** – the index of the first element in the interval

- * **index2** – the index of the last element in the interval

- * **axis** – the color component by which the interval is to be sorted, (in 14.1.15, page 334), (in 14.1.15, page 334) or (in 14.1.15, page 335)

- *sortByCounter*

```
public void sortByCounter( int index1, int index2 )
```

- **Description**

- Sorts an interval of the array of colors by their counters.

- **Parameters**

- * **index1** – the index of the first element in the interval

- * **index2** – the index of the last element in the interval

12.2.11 Class UniformPaletteQuantizer

A color quantizer that maps to a palette which is equidistantly distributed in the RGB color cube. Equidistantly distributed only within each channel.

Declaration

```
public class UniformPaletteQuantizer
extends net.sourceforge.jiu.ops.ImageToImageOperation (in 19.2.3, page 501)
implements net.sourceforge.jiu.data.RGBIndex, RGBQuantizer
```

Constructor summary

UniformPaletteQuantizer(int, int, int)

Method summary

createPalette()
map(int[], int[])
mapToIndex(int, int, int)
process()

Constructors

- *UniformPaletteQuantizer*
public UniformPaletteQuantizer(int redBits, int greenBits, int blueBits)

Methods

- *createPalette*
net.sourceforge.jiu.data.Palette createPalette()
 - **Description copied from RGBQuantizer (in 12.1.1, page 259)**
Return a Palette object with the list of colors to be used in the quantization process. That palette may be fixed or created specifically for a given input image.
 - **Returns** – Palette object for destination image
- *map*
int map(int[] origRgb, int[] quantizedRgb)
 - **Description copied from RGBQuantizer (in 12.1.1, page 259)**
This method maps a triplet of intensity values to its quantized counterpart and returns the palette index of that quantized color. The index values for the two arrays are taken from RGBIndex.
 - **Parameters**
 - * **origRgb** – the three samples red, green and blue for which a good match is searched in the palette
 - * **quantizedRgb** – will hold the three samples found to be closest to origRgb after the call to this method

– **Returns** – int index in the palette of the match quantizedRgb

• *mapToIndex*

public final int mapToIndex(int red, int green, int blue)

• *process*

public void process() throws
net.sourceforge.jiu.ops.MissingParameterException,
net.sourceforge.jiu.ops.OperationFailedException,
net.sourceforge.jiu.ops.WrongParameterException

– **Description copied from net.sourceforge.jiu.ops.Operation (in 19.2.5, page 508)**

This method does the actual work of the operation. It must be called after all parameters have been given to the operation object.

– **Throws**

- * net.sourceforge.jiu.ops.WrongParameterException – if at least one of the input parameters was not initialized appropriately (values out of the valid interval, etc.)
- * net.sourceforge.jiu.ops.MissingParameterException – if any mandatory parameter was not given to the operation
- * net.sourceforge.jiu.ops.OperationFailedException –

Chapter 13

Package

net.sourceforge.jiu.color.reduction

Package Contents

Page

Classes

AutoDetectColorType	296
<i>Detects the minimum (in terms of memory) color type of an image.</i>	
ReduceRGB	299
<i>Reduces the color depth of RGB truecolor images.</i>	
ReduceShadesOfGray	301
<i>Reduces the number of shades of gray of a grayscale image.</i>	
ReduceToBilevelThreshold	303
<i>Reduces a (in 14.1.6, page 321) to a (in 14.1.1, page 311) by setting all values below a certain threshold value to black and everything else to white.</i>	
RGBToGrayConversion	305
<i>Converts RGB color images (both truecolor and paletted) to grayscale images.</i>	

Classes to convert images to a lower color type. Usually, that conversion is lossy (e.g. from RGB truecolor to grayscale, or from shades of gray to black and white).

However, the (in 13.1.1, page 296) class reduces only if an image can be converted to a lower color type without losing information. This is the inverse operation to color promotion, provided by another package.

13.1 Classes

13.1.1 Class AutoDetectColorType

Detects the minimum (in terms of memory) color type of an image. Can convert the original image to that new input type on demand.

Input parameters: image to be examined, boolean that specifies whether conversion will be performed (default is true, conversion is performed). Output parameters: converted image, boolean that expresses whether a conversion was possible.

Supported types for input image: RGB24Image, Gray8Image, Paletted8Image.

BilevelImage is not supported because there is no smaller image type, so bilevel images cannot be reduced.

This operation is not a (in 19.2.3, page 501) because this class need not necessarily produce a new image (with (in 13.1.1, page 298)(false)).

Usage example

This code snippet loads an image and attempts to reduce it to the minimum color type that will hold it.

```
PixelImage image = ImageLoader.load("test.bmp");
AutoDetectColorType op = new AutoDetectColorType();
op.setInputImage(image);
op.process();
if (op.isReducible())
{
    image = op.getOutputImage();
}
```

Declaration

```
public class AutoDetectColorType
extends net.sourceforge.jiu.ops.Operation (in 19.2.5, page 508)
```

Field summary

```
TYPE_BILEVEL
TYPE_GRAY16
TYPE_GRAY8
TYPE_PALETTED8
TYPE_RGB24
TYPE_RGB48
TYPE_UNKNOWN
```

Constructor summary

```
AutoDetectColorType()
```

Method summary

- getOutputImage()** Returns the reduced output image if one was created in (in 13.1.1, page 298).
- getType()** Returns the type of the minimum image type found (one of the TYPE_xyz constants of this class).
- isReducible()** This method can be called after (in 13.1.1, page 298) to find out if the input image in fact can be reduced to a "smaller" image type.
- process()**
- setConversion(boolean)** This method can be used to specify whether the input image is to be converted to the minimum image type if it is clear that such a conversion is possible.
- setInputImage(PixelImage)** This method must be used to specify the mandatory input image.

Fields

- public static final int **TYPE_UNKNOWN**
- public static final int **TYPE_BILEVEL**
- public static final int **TYPE_GRAY16**
- public static final int **TYPE_GRAY8**
- public static final int **TYPE_PALETTE8**
- public static final int **TYPE_RGB24**
- public static final int **TYPE_RGB48**

Constructors

- *AutoDetectColorType*
public **AutoDetectColorType**()

Methods

- *getOutputImage*
public net.sourceforge.jiu.data.PixelImage **getOutputImage**()
 - **Description**
Returns the reduced output image if one was created in (in 13.1.1, page 298).
 - **Returns** – newly-created output image
- *getType*
public int **getType**()
 - **Description**
Returns the type of the minimum image type found (one of the TYPE_xyz constants of this class). Can only be called after a successful call to process.

- *isReducible*

```
public boolean isReducible( )
```

- **Description**

This method can be called after (in 13.1.1, page 298) to find out if the input image in fact can be reduced to a "smaller" image type. If this method returns `true` and if conversion was desired by the user (can be specified via (in 13.1.1, page 298)), the reduced image can be retrieved via (in 13.1.1, page 297).

- **Returns** – if image was found to be reducible in process()

- *process*

```
public void process( ) throws
net.sourceforge.jiu.ops.MissingParameterException,
net.sourceforge.jiu.ops.OperationFailedException,
net.sourceforge.jiu.ops.WrongParameterException
```

- **Description** copied from `net.sourceforge.jiu.ops.Operation` (in 19.2.5, page 508)

This method does the actual work of the operation. It must be called after all parameters have been given to the operation object.

- **Throws**

- * `net.sourceforge.jiu.ops.WrongParameterException` – if at least one of the input parameters was not initialized appropriately (values out of the valid interval, etc.)
 - * `net.sourceforge.jiu.ops.MissingParameterException` – if any mandatory parameter was not given to the operation
 - * `net.sourceforge.jiu.ops.OperationFailedException` –
-

- *setConversion*

```
public void setConversion( boolean convert )
```

- **Description**

This method can be used to specify whether the input image is to be converted to the minimum image type if it is clear that such a conversion is possible. The default value is `true`. If this is set to `false`, it can still be

- **Parameters**

- * `convert` – if true, the conversion will be performed
-

- *setInputImage*

```
public void setInputImage( net.sourceforge.jiu.data.PixelImage image )
```

- **Description**

This method must be used to specify the mandatory input image.

- **Parameters**

- * `image` – `PixelImage` object to be examined

13.1.2 Class ReduceRGB

Reduces the color depth of RGB truecolor images. This class uses a simple approach, it just drops some of the lowest bits and scales the value back to eight or sixteen bits per sample.

Supported image classes

This class works with `Image` (in 14.1.12, page 331) and `ImageIO` (in 14.1.13, page 332).

Usage example

Reduce a 24 bits per pixel RGB image to 15 bits per pixel:

```
RGB24Image inputImage = ...; // initialize
ReduceRGB reduce = new ReduceRGB();
reduce.setBits(5);
reduce.setInputImage(inputImage);
reduce.process();
PixelImage reducedImage = reduce.getOutputImage();
```

See also

- `ReduceShadesOfGray` (in 13.1.3, page 301)

Declaration

```
public class ReduceRGB
  extends net.sourceforge.jiu.ops.ImageToImageOperation (in 19.2.3, page 501)
```

Constructor summary

ReduceRGB()

Method summary

process()
setBits(int) Specifies the number of bits the output image is supposed to have.

Constructors

- *ReduceRGB*
public ReduceRGB()

Methods

- *process*
public void process() throws
`net.sourceforge.jiu.ops.MissingParameterException`,
`net.sourceforge.jiu.ops.OperationFailedException`,
`net.sourceforge.jiu.ops.WrongParameterException`

- **Description** copied from `net.sourceforge.jiu.ops.Operation` (in 19.2.5, page 508)

This method does the actual work of the operation. It must be called after all parameters have been given to the operation object.

- **Throws**

- * `net.sourceforge.jiu.ops.WrongParameterException` – if at least one of the input parameters was not initialized appropriately (values out of the valid interval, etc.)
 - * `net.sourceforge.jiu.ops.MissingParameterException` – if any mandatory parameter was not given to the operation
 - * `net.sourceforge.jiu.ops.OperationFailedException` –
-

- *setBits*

```
public void setBits( int bits )
```

- **Description**

Specifies the number of bits the output image is supposed to have.

- **Parameters**

- * `bits` – number of bits in output image, from 1 to 15

- **Throws**

- * `java.lang.IllegalArgumentException` – if bits is smaller than 1 or larger than 15

13.1.3 Class ReduceShadesOfGray

Reduces the number of shades of gray of a grayscale image. This class uses the most simple possible algorithm. Only the most significant N bits are kept (where N is the number specified with (in 13.1.3, page 302)), the others are dropped and the result is scaled back to either 8 or 16 bits to fit into the two grayscale image types.

Supported image classes

This class works with (in 14.1.4, page 318) and (in 14.1.3, page 317).

Usage example

Reduce a grayscale image to 3 bit ($2^3 = 8$ shades of gray):

```
ReduceShadesOfGray reduce = new ReduceShadesOfGray();
reduce.setBits(3);
reduce.setInputImage(image); // some Gray8Image or Gray16Image
reduce.process();
PixelImage reducedImage = reduce.getOutputImage();
```

Declaration

```
public class ReduceShadesOfGray
extends net.sourceforge.jiu.ops.ImageToImageOperation (in 19.2.3, page 501)
```

Constructor summary

ReduceShadesOfGray()

Method summary

process()
setBits(int) Specifies the number of bits the output image is supposed to have.

Constructors

- *ReduceShadesOfGray*
public ReduceShadesOfGray()

Methods

- *process*
public void process() throws
 net.sourceforge.jiu.ops.MissingParameterException,
 net.sourceforge.jiu.ops.OperationFailedException,
 net.sourceforge.jiu.ops.WrongParameterException

- **Description** copied from `net.sourceforge.jiu.ops.Operation` (in 19.2.5, page 508)

This method does the actual work of the operation. It must be called after all parameters have been given to the operation object.

- **Throws**

- * `net.sourceforge.jiu.ops.WrongParameterException` – if at least one of the input parameters was not initialized appropriately (values out of the valid interval, etc.)
 - * `net.sourceforge.jiu.ops.MissingParameterException` – if any mandatory parameter was not given to the operation
 - * `net.sourceforge.jiu.ops.OperationFailedException` –
-

- *setBits*

```
public void setBits( int bits )
```

- **Description**

Specifies the number of bits the output image is supposed to have.

- **Parameters**

- * `bits` – number of bits in output image, from 1 to 15

- **Throws**

- * `java.lang.IllegalArgumentException` – if bits is smaller than 1 or larger than 15

13.1.4 Class ReduceToBilevelThreshold

Reduces a (in 14.1.6, page 321) to a (in 14.1.1, page 311) by setting all values below a certain threshold value to black and everything else to white.

Default value

If no threshold is specified via (in 13.1.4, page 304), this operation uses a default value of $(\text{(in 14.1.7, page 323)} + 1) / 2$.

Usage example

This example sets all values below 33 percent luminance to black, everything else to white.

```
GrayIntegerImage image = ...;
ReduceToBilevelThreshold red = new ReduceToBilevelThreshold();
red.setInputImage(image);
red.setThreshold(image.getMaxSample(0) / 3);
red.process();
BilevelImage reducedImage= (BilevelImage)red.getOutputStream();
```

Declaration

```
public class ReduceToBilevelThreshold
  extends net.sourceforge.jiu.ops.ImageToImageOperation (in 19.2.3, page 501)
```

Constructor summary

ReduceToBilevelThreshold()

Method summary

getThreshold() Returns the current threshold value, or null if none was specified and the operation's process method was not run yet.
process()
setThreshold(int) Sets a new threshold value.

Constructors

- *ReduceToBilevelThreshold*
public ReduceToBilevelThreshold()

Methods

- *getThreshold*
public java.lang.Integer getThreshold()
 - **Description**
Returns the current threshold value, or null if none was specified and the operation's process method was not run yet.

- **Returns** – threshold value
-

- *process*

public void process() throws
 net.sourceforge.jiu.ops.MissingParameterException,
 net.sourceforge.jiu.ops.OperationFailedException,
 net.sourceforge.jiu.ops.WrongParameterException

- **Description** copied from net.sourceforge.jiu.ops.Operation (in 19.2.5, page 508)

This method does the actual work of the operation. It must be called after all parameters have been given to the operation object.

- **Throws**

- * net.sourceforge.jiu.ops.WrongParameterException – if at least one of the input parameters was not initialized appropriately (values out of the valid interval, etc.)
 - * net.sourceforge.jiu.ops.MissingParameterException – if any mandatory parameter was not given to the operation
 - * net.sourceforge.jiu.ops.OperationFailedException –
-

- *setThreshold*

public void setThreshold(int newThreshold)

- **Description**

Sets a new threshold value.

- **Parameters**

- * newThreshold – the new threshold value to be used for this operation

- **Throws**

- * java.lang.IllegalArgumentException – if the threshold value is negative

13.1.5 Class RGBToGrayConversion

Converts RGB color images (both truecolor and paletted) to grayscale images. The weights to be used with the three base colors red, green and blue can be modified with a call to (in 13.1.5, page 306).

Supported image types

(in 14.1.12, page 331) and (in 14.1.8, page 325) can be used as input image types. A (in 14.1.4, page 318) will be created from them.

Could be optimized to use int multiplication instead of float multiplication.

NOTE: Should be adjusted to support RGB48Image objects once they're available.

Usage example

```
RGBToGrayConversion rgbtogray = new RGBToGrayConversion();
rgbtogray.setInputImage(image);
rgbtogray.process();
PixelImage grayImage = rgbtogray.getOutputImage();
```

Declaration

```
public class RGBToGrayConversion
  extends net.sourceforge.jiu.ops.ImageToImageOperation (in 19.2.3, page 501)
```

Field summary

DEFAULT_BLUE_WEIGHT The default weight for blue samples in the conversion, 0.11f.

DEFAULT_GREEN_WEIGHT The default weight for green samples in the conversion, 0.59f.

DEFAULT_RED_WEIGHT The default weight for red samples in the conversion, 0.3f.

Constructor summary

RGBToGrayConversion()

Method summary

process()

setColorWeights(float, float, float) Sets the weights for the three colors red, green and blue used in the conversion procedure.

Fields

- public static final float **DEFAULT_RED_WEIGHT**
 - The default weight for red samples in the conversion, 0.3f.

- public static final float **DEFAULT_GREEN_WEIGHT**
 - The default weight for green samples in the conversion, 0.59f.
- public static final float **DEFAULT_BLUE_WEIGHT**
 - The default weight for blue samples in the conversion, 0.11f.

Constructors

- *RGBToGrayConversion*
public **RGBToGrayConversion**()

Methods

- *process*
public void **process**() throws
net.sourceforge.jiu.ops.MissingParameterException,
net.sourceforge.jiu.ops.OperationFailedException,
net.sourceforge.jiu.ops.WrongParameterException
 - **Description copied from net.sourceforge.jiu.ops.Operation** (in 19.2.5, page 508)
This method does the actual work of the operation. It must be called after all parameters have been given to the operation object.
 - **Throws**
 - * net.sourceforge.jiu.ops.WrongParameterException – if at least one of the input parameters was not initialized appropriately (values out of the valid interval, etc.)
 - * net.sourceforge.jiu.ops.MissingParameterException – if any mandatory parameter was not given to the operation
 - * net.sourceforge.jiu.ops.OperationFailedException –
- *setColorWeights*
public void **setColorWeights**(float red, float green, float blue)
 - **Description**
Sets the weights for the three colors red, green and blue used in the conversion procedure. For each RGB value (r, g, b) to be converted (whether in a truecolor image or in the palette), the formula is `gray = r * red + g * green + b * blue`. The default values for these weights are (in 13.1.5, page 305), (in 13.1.5, page 306) and (in 13.1.5, page 306). This method lets the user change these values. Each of these arguments must be $\geq 0.0f$ and $\leq 1.0f$. The sum of the three must be $\leq 1.0f$. For any resulting gray value to be spread over the complete scale from 0.0f to 1.0f it is preferable for the sum to be equal to or at least close to 1.0f. However, this is not checked. The smaller the sum of the weights is, the darker the resulting gray image will become.
 - **Parameters**
 - * **red** – weight of the red sample in the conversion, between 0.0f and 1.0f
 - * **green** – weight of the green sample in the conversion, between 0.0f and 1.0f
 - * **blue** – weight of the blue sample in the conversion, between 0.0f and 1.0f

– **Throws**

- * `java.lang.IllegalArgumentException` – if any one of the above mentioned constraints for the arguments is not met

Chapter 14

Package net.sourceforge.jiu.data

Package Contents

Page

Interfaces

BilevelImage	311
<i>An interface for bilevel pixel image data classes.</i>	
ByteChannellImage	314
<i>An extension of the (in 14.1.7, page 322)interface that restricts the image to byte samples.</i>	
Gray16Image	317
<i>Interface for grayscale images using integer samples that are sixteen bits large.</i>	
Gray8Image	318
<i>Interface for grayscale images using integer samples that are eight bits large.</i>	
GrayImage	319
<i>An interface for grayscale images.</i>	
GrayIntegerImage	321
<i>An empty interface for grayscale images which have integer values of up to 32 bits (int or smaller) as samples.</i>	
IntegerImage	322
<i>Extends the (in 14.1.11, page 328)interface to access image data as int values.</i>	
Paletted8Image	325
<i>An interface for classes that store paletted images with eight bit integers for each pixel.</i>	
PalettedImage	326
<i>This interface defines methods for paletted images.</i>	
PalettedIntegerImage	327
<i>An empty interface for a paletted image type that uses integer values as samples.</i>	
PixelImage	328
<i>The base interface for all image data types in JIU.</i>	
RGB24Image	331
<i>An empty interface for RGB truecolor images with integer samples that are each eight bits large (thus, 24 bits per pixel).</i>	
RGB48Image	332
<i>An empty interface for RGB truecolor images with integer samples that are each sixteen bits large (thus, 48 bits per pixel).</i>	
RGBImage	333

<i>An interface for RGB truecolor images.</i>	
RGBIndex	334
<i>This interface provides three <code>int</code> constants as index values for the three channels of an RGB image: red, green and blue.</i>	
RGBIntegerImage	336
<i>An interface for RGB truecolor images that have integer samples.</i>	
ShortChannelImage	337
<i>An extension of the (in 14.1.7, page 322) interface that restricts the image to short samples.</i>	
TransparencyInformation	340
<i>An interface that represents transparency information which may be available for a pixel image.</i>	
Classes	
MemoryBilevelImage	342
<i>An implementation of the (in 14.1.1, page 311) interface that stores image data in a <code>byte</code> array in memory.</i>	
MemoryByteChannelImage	345
<i>An implementation of (in 14.1.2, page 314) that stores image channels as <code>byte[]</code> arrays in memory.</i>	
MemoryGray16Image	350
<i>An implementation of (in 14.1.3, page 317) that keeps the complete image in memory.</i>	
MemoryGray8Image	352
<i>An implementation of (in 14.1.4, page 318) that keeps the complete image in memory.</i>	
MemoryPaletted8Image	354
<i>This class stores a paletted image with one byte per sample in memory.</i>	
MemoryRGB24Image	356
<i>A class to store 24 bit RGB truecolor images in memory.</i>	
MemoryRGB48Image	357
<i>A class to store 48 bit RGB truecolor images in memory.</i>	
MemoryShortChannelImage	358
<i>An implementation of (in 14.1.17, page 337) that stores image channels as <code>short[]</code> arrays in memory.</i>	
Palette	363
<i>This class represents a palette, a list of RGB colors.</i>	

Provides classes to store images and data directly related to them.

Package Specification

The base interface for image data in JIU is (in 14.1.11, page 328). The concept of a pixel image includes the following properties:

- The image data is arranged as a rectangular grid of pixels. The image has a fixed number of columns (width) and rows (height). Width and height can be queried using the method `getWidth` and `getHeight`.
- Each pixel is made up of one or more samples. The exact number of samples per pixel equals the number of channels. Nothing is said about the nature of the samples (e. g., their

type) and no ways to access samples or pixels are provided in the `PixelImage` interface.

- A class implementing `PixelImage` must provide the `createCompatibleImage` method which gets a width and a height value as parameters and must return a new object of the same class with that pixel resolution.
- Helper methods in `PixelImage` include ways to get a textual description of the type and the number of bytes allocated for a specific object implementing `PixelImage`.

The interface (in 14.1.7, page 322) extends the (in 14.1.11, page 328) interface. All sample values belonging to an object of a class implementing `IntegerImage` are supposed to be integer values that can be stored in an `int` value (a signed 32 bit value).

14.1 Interfaces

14.1.1 Interface BilevelImage

An interface for bilevel pixel image data classes. Each pixel in a bilevel image can have two possible values, (in 14.1.1, page 312) and (in 14.1.1, page 312). Those two constants are guaranteed to be 0 and 1, although you should not make any assumptions about what value any of the two constants has. This is the type of image used for faxes. Black and white photos are usually stored as grayscale images.

Apart from implementing (in 14.1.11, page 328)- like all image data classes in JIU - this interface is also an (in 14.1.7, page 322)(each sample is either 0 or 1) and a (in 14.1.5, page 319)(black and white are both grayscale values). The combination of (in 14.1.7, page 322) and (in 14.1.5, page 319) is (in 14.1.6, page 321), which is the direct superinterface of this interface.

Packed bytes

There are methods to get and put packed bytes in this interface. A packed byte is a `byte` value that stores eight horizontally neighbored bilevel pixels in it (pixel and sample can be used interchangeably in the context of bilevel images because there is only one sample per pixel). The most significant bit of such a packed bit is defined to be the leftmost of the eight pixels, the second-most significant bit is the pixel to the right of that leftmost pixel, and so on. The least significant bit is the rightmost pixel. If a bit is set, the corresponding pixel value is supposed to be white, otherwise black.

Usage examples

Here are some code examples that demonstrate how to access image data with this class.

```
BilevelImage image = new MemoryBilevelImage(2000, 500);
// now set all pixels in the first row to white
for (int x = 0; x < image.getWidth(); x++)
{
    image.putWhite(x, 0);
}
// put vertical stripes on the rest
for (int y = 1; y < image.getHeight(); y++)
{
    for (int x = 0; x < image.getWidth(); x++)
    {
        int sample;
        if ((x % 2) == 0)
        {
            sample = BilevelImage.BLACK;
        }
        else
        {
            sample = BilevelImage.WHITE;
        }
        image.putSample(x, y, sample);
    }
}
```



```
}
```

Declaration

```
public interface BilevelImage
implements GrayIntegerImage
```

All known subclasses

MemoryBilevelImage (in 14.2.1, page 342)

All classes known to implement interface

MemoryBilevelImage (in 14.2.1, page 342)

Field summary

BLACK The value for a black pixel.

WHITE The value for a white pixel.

Method summary

getPackedBytes(int, int, int, byte[], int, int) Sets a number of samples in the argument array from this image.

putPackedBytes(int, int, int, byte[], int, int) Sets a number of samples in the image from the argument array data.

Fields

- int **BLACK**
 - The value for a black pixel. To be used with all methods that require `int` arguments for sample values. You can rely on this value being either 0 or 1 (that way you can safely store it in a byte or short).
- int **WHITE**
 - The value for a white pixel. To be used with all methods that require `int` arguments for sample values. You can rely on this value being either 0 or 1 (that way you can safely store it in a byte or short).

Methods

- *getPackedBytes*

```
void getPackedBytes( int x, int y, int numSamples, byte[] dest, int
destOffset, int destBitOffset )
```

 - **Description**

Sets a number of samples in the argument array from this image.
 - **Parameters**
 - * `x` – horizontal position of first sample of this image to read

- * **y** – vertical position of samples to be read from this image
- * **numSamples** – number of samples to be set
- * **dest** – array with packed pixels to which samples are copied
- * **destOffset** – index into dest array of the first byte value to write sample values to
- * **destBitOffset** – index of first bit of **dest[destOffset]** to write a sample to (0 is leftmost, 1 is second-leftmost up to 7, which is the rightmost)

- *putPackedBytes*

```
void putPackedBytes( int x, int y, int numSamples, byte[] src, int
srcOffset, int srcBitOffset )
```

- **Description**

Sets a number of samples in the image from the argument array data.

- **Parameters**

- * **x** – horizontal position of first sample to be set
- * **y** – vertical position of samples to be set
- * **numSamples** – number of samples to be set
- * **src** – array with packed pixels to be set
- * **srcOffset** – index into src array of the first byte value to read sample values from
- * **srcBitOffset** – index of first bit of **src[srcOffset]** to read a sample from (0 is leftmost, 1 is second-leftmost up to 7, which is the rightmost)

14.1.2 Interface ByteChannelImage

An extension of the (in 14.1.7, page 322) interface that restricts the image to byte samples. The minimum sample value for all channels is 0, the maximum sample value 255.

Number of channels and resolution must be given to the constructor and cannot be changed after creation.

Each channel of the image is made up of byte values. Note that bytes in Java are signed, they can take values from -128 to 127. If you use (in 14.1.7, page 322)'s `getSample` and `putSample` methods you don't have to deal with this, you always get `int` samples that are in the 0 .. 255 interval.

To manually convert a Java byte value to an `int` value in the range of 0 to 255, do the following:

```
byte b = ...; // initialize byte value
int i = b & 0xff;
// i now is a value between 0 and 255
```

Declaration

```
public interface ByteChannelImage
implements IntegerImage
```

All known subclasses

`RGB24Image` (in 14.1.12, page 331), `Paletted8Image` (in 14.1.8, page 325), `MemoryRGB24Image` (in 14.2.6, page 356), `MemoryPaletted8Image` (in 14.2.5, page 354), `MemoryGray8Image` (in 14.2.4, page 352), `MemoryByteChannelImage` (in 14.2.2, page 345), `Gray8Image` (in 14.1.4, page 318), `BufferedRGB24Image` (in 17.1.4, page 433)

All known subinterfaces

`RGB24Image` (in 14.1.12, page 331), `Paletted8Image` (in 14.1.8, page 325), `Gray8Image` (in 14.1.4, page 318)

All classes known to implement interface

`MemoryByteChannelImage` (in 14.2.2, page 345)

Method summary

- `clear(byte)`** Sets all samples of the first channel to the argument byte value.
- `clear(int, byte)`** Sets all samples of one channel to a new value.
- `getByteSample(int, int)`** Returns a single byte sample from the first channel and the specified position.
- `getByteSample(int, int, int)`** Returns a single byte sample from the image.
- `getByteSamples(int, int, int, int, int, byte[], int)`** Copies samples from this image to a byte array.
- `putByteSample(int, int, byte)`** Sets one byte sample in the first channel (index 0) to a new value.
- `putByteSample(int, int, int, byte)`** Sets one byte sample in one channel to a new value.
- `putByteSamples(int, int, int, int, int, byte[], int)`** Copies a number of samples from the argument array to this image.

Methods

- *clear*

void clear(byte newValue)

- **Description**

Sets all samples of the first channel to the argument byte value. Equal to `clear(0, newValue);`.

- **Parameters**

* **newValue** – all samples in the first channel are set to this value

- **See also**

* `ByteChannelImage.clear(int,byte)` (in 14.1.2, page 315)

* `IntegerImage.clear(int)` (in 14.1.7, page 323)

* `IntegerImage.clear(int,int)` (in 14.1.7, page 323)

- *clear*

void clear(int channelIndex, byte newValue)

- **Description**

Sets all samples of one channel to a new value.

- **Parameters**

* **channelIndex** – zero-based index of the channel to be cleared (must be smaller than (in 14.1.11, page 330)

* **newValue** – all samples in the channel will be set to this value

- *getBytesSample*

byte getBytesSample(int x, int y)

- **Description**

Returns a single byte sample from the first channel and the specified position. A call to this method is the same as `getBytesSample(0, x, y)`.

- **Parameters**

* **x** – horizontal position of the sample to be returned (must be between 0 and (in 14.1.11, page 330)- 1

* **y** – vertical position of the sample to be returned (must be between 0 and (in 14.1.11, page 329)- 1

- **Returns** – the requested byte sample

- *getBytesSample*

byte getBytesSample(int channel, int x, int y)

- **Description**

Returns a single byte sample from the image. When possible, try copying several samples at a time for higher speed ((in 14.1.2, page 316)).

- **Parameters**

* **channel** – the number of the channel of the sample; must be from 0 to (in 14.1.11, page 330)- 1

* **x** – the column of the sample to be returned; must be from 0 to (in 14.1.11, page 330)- 1

* **y** – the row of the sample; must be from 0 to (in 14.1.11, page 329)- 1

- **Returns** – the sample, a single byte value
 - **Throws**
 - * `java.lang.IllegalArgumentException` – if the arguments hurt one of the preconditions above
 - **See also**
 - * `ByteChannelImage.getBytesSamples(int,int,int,int,int,byte[],int)` (in 14.1.2, page 316)
-

- *getBytesSamples*

```
void getBytesSamples( int channelIndex, int x, int y, int w, int h, byte[]
dest, int destOffset )
```

- **Description**

Copies samples from this image to a byte array. Copies `num` samples in row `y` of channel `channel`, starting at horizontal offset `x`. Data will be written to the `dest` array, starting at offset `destOffset`. Data will be copied from one row only, so a maximum of `getWidth()` samples can be copied with a call to this method.
 - **Parameters**
 - * `channelIndex` – the index of the channel to be copied from; must be from 0 to `getNumChannels() - 1`
 - * `x` – the horizontal offset where copying will start; must be from 0 to `getWidth() - 1`
 - * `y` – the row from which will be copied; must be from 0 to `getHeight() - 1`
 - * `w` – the number of columns to be copied
 - * `h` – the number of rows to be copied
 - * `dest` – the array where the data will be copied to; must have a length of at least `destOffset + num`
 - * `destOffset` – the offset into `dest` where this method will start copying data
 - **Throws**
 - * `java.lang.IllegalArgumentException` – if the arguments hurt one of the many preconditions above
-

- *putByteSample*

```
void putByteSample( int x, int y, byte newValue )
```

- **Description**

Sets one byte sample in the first channel (index 0) to a new value. Result is equal to `putByteSample(0, x, y, newValue);`.
-

- *putByteSample*

```
void putByteSample( int channel, int x, int y, byte newValue )
```

- **Description**

Sets one byte sample in one channel to a new value.
-

- *putByteSamples*

```
void putByteSamples( int channel, int x, int y, int w, int h, byte[] src,
int srcOffset )
```

- **Description**

Copies a number of samples from the argument array to this image.

14.1.3 *Interface* **Gray16Image**

Interface for grayscale images using integer samples that are sixteen bits large. Valid sample values must lie in the interval 0 to 65535 (including both of those values). Like all grayscale images, implementations of this class are supposed to have one channel only. Simply merges the two interfaces `GrayIntegerImage` and `ShortChannelImage` without adding methods of its own.

See also

- `ShortChannelImage` (in 14.1.17, page 337)
- `GrayIntegerImage` (in 14.1.6, page 321)

Declaration

```
public interface Gray16Image
implements GrayIntegerImage, ShortChannelImage
```

All known subclasses

`MemoryGray16Image` (in 14.2.3, page 350)

All classes known to implement interface

`MemoryGray16Image` (in 14.2.3, page 350)

14.1.4 *Interface* **Gray8Image**

Interface for grayscale images using integer samples that are eight bits large. Valid sample values must lie in the interval 0 to 255 (including both of those values). Like all grayscale images, implementations of this class are supposed to have one channel only. Simply merges the two interfaces `GrayIntegerImage` and `ByteChannelImage` without adding new methods.

See also

- `ByteChannelImage` (in 14.1.2, page 314)
- `GrayIntegerImage` (in 14.1.6, page 321)

Declaration

```
public interface Gray8Image
implements GrayIntegerImage, ByteChannelImage
```

All known subclasses

`MemoryGray8Image` (in 14.2.4, page 352)

All classes known to implement interface

`MemoryGray8Image` (in 14.2.4, page 352)

14.1.5 Interface **GrayImage**

An interface for grayscale images. Grayscale images have only one channel. Each sample is a shade of gray, an intensity value between black (zero) and white (maximum value). Black and white photos are really grayscale photos. For images that only use black and white, see (in 14.1.1, page 311).

Declaration

```
public interface GrayImage
```

All known subclasses

MemoryGray8Image (in 14.2.4, page 352), MemoryGray16Image (in 14.2.3, page 350), MemoryBilevelImage (in 14.2.1, page 342), GrayIntegerImage (in 14.1.6, page 321), Gray8Image (in 14.1.4, page 318), Gray16Image (in 14.1.3, page 317), BilevelImage (in 14.1.1, page 311)

All known subinterfaces

GrayIntegerImage (in 14.1.6, page 321)

Method summary

- isBlack(int, int)** Returns if the pixel specified by the location in the arguments is black.
- isWhite(int, int)** Returns if the pixel specified by the location in the arguments is white.
- putBlack(int, int)** Sets a pixel to black (minimum intensity value).
- putWhite(int, int)** Sets a pixel to white (maximum intensity value).

Methods

- *isBlack*

```
boolean isBlack( int x, int y )
```

 - **Description**
Returns if the pixel specified by the location in the arguments is black.
 - **Parameters**
 - * **x** – the horizontal location of the pixel
 - * **y** – the vertical location of the pixel
 - **Throws**
 - * `java.lang.IllegalArgumentException` – if any of the parameters are invalid
- *isWhite*

```
boolean isWhite( int x, int y )
```

 - **Description**
Returns if the pixel specified by the location in the arguments is white.
 - **Parameters**
 - * **x** – the horizontal location of the pixel

- * *y* – the vertical location of the pixel
 - **Throws**
 - * `java.lang.IllegalArgumentException` – if any of the parameters are invalid

- *putBlack*
`void putBlack(int x, int y)`
 - **Description**
Sets a pixel to black (minimum intensity value).
 - **Parameters**
 - * *x* – horizontal position of the pixel's location
 - * *y* – vertical position of the pixel's location

- *putWhite*
`void putWhite(int x, int y)`
 - **Description**
Sets a pixel to white (maximum intensity value).
 - **Parameters**
 - * *x* – horizontal position of the pixel's location
 - * *y* – vertical position of the pixel's location

14.1.6 *Interface* **GrayIntegerImage**

An empty interface for grayscale images which have integer values of up to 32 bits (`int` or smaller) as samples. An interface composed of (in 14.1.5, page 319) and (in 14.1.7, page 322). Like all extensions of (in 14.1.5, page 319), this image data class supports only one channel.

See also

- `GrayImage` (in 14.1.5, page 319)
- `IntegerImage` (in 14.1.7, page 322)

Declaration

```
public interface GrayIntegerImage
implements GrayImage, IntegerImage
```

All known subclasses

`MemoryGray8Image` (in 14.2.4, page 352), `MemoryGray16Image` (in 14.2.3, page 350), `MemoryBilevelImage` (in 14.2.1, page 342), `Gray8Image` (in 14.1.4, page 318), `Gray16Image` (in 14.1.3, page 317), `BilevelImage` (in 14.1.1, page 311)

All known subinterfaces

`Gray8Image` (in 14.1.4, page 318), `Gray16Image` (in 14.1.3, page 317), `BilevelImage` (in 14.1.1, page 311)

14.1.7 Interface IntegerImage

Extends the (in 14.1.11, page 328) interface to access image data as `int` values. Image types based on `byte`, `char`, `short` and `int` will work with this interface. `long` will not.

Using this interface provides a nice way of accessing a large variety of image types, but for performance reasons it might be preferable to use one of the class-specific access methods that get or put several values at a time, e.g. `getBytesSamples` in (in 14.1.2, page 314).

Declaration

```
public interface IntegerImage
implements PixelImage
```

All known subclasses

`ShortChannelImage` (in 14.1.17, page 337), `RGBIntegerImage` (in 14.1.16, page 336), `RGB48Image` (in 14.1.13, page 332), `RGB24Image` (in 14.1.12, page 331), `PalettedIntegerImage` (in 14.1.10, page 327), `Paletted8Image` (in 14.1.8, page 325), `MemoryShortChannelImage` (in 14.2.8, page 358), `MemoryRGB48Image` (in 14.2.7, page 357), `MemoryRGB24Image` (in 14.2.6, page 356), `MemoryPaletted8Image` (in 14.2.5, page 354), `MemoryGray8Image` (in 14.2.4, page 352), `MemoryGray16Image` (in 14.2.3, page 350), `MemoryByteChannelImage` (in 14.2.2, page 345), `MemoryBilevelImage` (in 14.2.1, page 342), `GrayIntegerImage` (in 14.1.6, page 321), `Gray8Image` (in 14.1.4, page 318), `Gray16Image` (in 14.1.3, page 317), `ByteChannelImage` (in 14.1.2, page 314), `BilevelImage` (in 14.1.1, page 311), `BufferedRGB24Image` (in 17.1.4, page 433)

All known subinterfaces

`ShortChannelImage` (in 14.1.17, page 337), `RGBIntegerImage` (in 14.1.16, page 336), `PalettedIntegerImage` (in 14.1.10, page 327), `GrayIntegerImage` (in 14.1.6, page 321), `ByteChannelImage` (in 14.1.2, page 314)

Method summary

`clear(int)` Sets all samples in the first channel to the argument value.

`clear(int, int)` Sets all samples of the `channelIndex`'th channel to `newValue`.

`getMaxSample(int)` Returns the maximum value for one of the image's channels.

`getSample(int, int)` Returns one sample of the first channel (index 0).

`getSample(int, int, int)` Returns one sample, specified by its channel index and location.

`getSamples(int, int, int, int, int, int[], int)` Copies a number of samples from this image to an `int[]` object.

`putSample(int, int, int)` This method sets one sample of the first channel (index 0) to a new value.

`putSample(int, int, int, int)` This method sets one sample to a new value.

`putSamples(int, int, int, int, int, int[], int)` Copies a number of samples from an `int[]` array to this image.

Methods

- *clear*

```
void clear( int newValue )
```

- **Description**

Sets all samples in the first channel to the argument value. Equal to `clear(0, newValue);`

- *clear*

```
void clear( int channelIndex, int newValue )
```

- **Description**

Sets all samples of the `channelIndex`'th channel to `newValue`.

- *getMaxSample*

```
int getMaxSample( int channel )
```

- **Description**

Returns the maximum value for one of the image's channels. The minimum value is always 0.

- **Parameters**

* `channel` – zero-based index of the channel, from 0 to (in 14.1.11, page 330)- 1

- **Returns** – maximum allowed sample value

- *getSample*

```
int getSample( int x, int y )
```

- **Description**

Returns one sample of the first channel (index 0). A call to this method must have the same result as the call `getSample(0, x, y);`.

- **Parameters**

* `x` – the horizontal position of the sample, from 0 to (in 14.1.11, page 330)- 1

* `y` – the vertical position of the sample, from 0 to (in 14.1.11, page 329)- 1

- **Returns** – the desired sample

- *getSample*

```
int getSample( int channel, int x, int y )
```

- **Description**

Returns one sample, specified by its channel index and location.

- **Parameters**

* `channel` – the number of the channel, from 0 to (in 14.1.11, page 330)- 1

* `x` – the horizontal position of the sample, from 0 to (in 14.1.11, page 330)- 1

* `y` – the vertical position of the sample, from 0 to (in 14.1.11, page 329)- 1

- **Returns** – the desired sample

- *getSamples*

```
void getSamples( int channelIndex, int x, int y, int w, int h, int[] dest,  
int destOffs )
```

- **Description**

Copies a number of samples from this image to an `int[]` object. A rectangular part of one channel is copied. The channel index is given by - the upper left corner of that rectangle is given by the point `x / y`. Width and height of that rectangle are given by `w` and `h`. Each sample will be stored as one `int` value `dest`, starting at index `destOffs`.

– **Parameters**

- * **channelIndex** – zero-based index of the channel from which data is to be copied (valid values: 0 to (in 14.1.11, page 330)- 1)
 - * **x** – horizontal position of upper left corner of the rectangle to be copied
 - * **y** – vertical position of upper left corner of the rectangle to be copied
 - * **w** – width of rectangle to be copied
 - * **h** – height of rectangle to be copied
 - * **dest** – int array to which the samples will be copied
 - * **destOffs** – int index into the dest array for the position to which the samples will be copied
-

• *putSample*

void putSample(int x, int y, int newValue)

– **Description**

This method sets one sample of the first channel (index 0) to a new value. This call must have the same result as the call `putSample(0, x, y)`. The sample location is given by the spatial coordinates, x and y.

– **Parameters**

- * **x** – the horizontal position of the sample, from 0 to (in 14.1.11, page 330)- 1
 - * **y** – the vertical position of the sample, from 0 to (in 14.1.11, page 329)- 1
 - * **newValue** – the new value of the sample
-

• *putSample*

void putSample(int channel, int x, int y, int newValue)

– **Description**

This method sets one sample to a new value. The sample location is given by the channel index and the spatial coordinates, x and y.

– **Parameters**

- * **channel** – the number of the channel, from 0 to (in 14.1.11, page 330)- 1
 - * **x** – the horizontal position of the sample, from 0 to (in 14.1.11, page 330)- 1
 - * **y** – the vertical position of the sample, from 0 to (in 14.1.11, page 329)- 1
 - * **newValue** – the new value of the sample
-

• *putSamples*

void putSamples(int channel, int x, int y, int w, int h, int[] src, int srcOffset)

– **Description**

Copies a number of samples from an `int[]` array to this image. A rectangular part of one channel is copied - the upper left corner of that rectangle is given by the point x / y. Width and height of that rectangle are given by w and h. Each sample will be stored as one int value src, starting at index srcOffset.

– **Parameters**

- * **channel** – int (from 0 to `getNumChannels()` - 1) to indicate the channel to which data is copied
- * **x** – horizontal position of upper left corner of the rectangle to be copied
- * **y** – vertical position of upper left corner of the rectangle to be copied
- * **w** – width of rectangle to be copied
- * **h** – height of rectangle to be copied
- * **src** – int array from which the samples will be copied
- * **srcOffset** – int index into the src array for the position from which the samples will be copied

14.1.8 *Interface* **Paletted8Image**

An interface for classes that store paletted images with eight bit integers for each pixel.

See also

- **ByteChannelImage** (in 14.1.2, page 314)
- **PalettedIntegerImage** (in 14.1.10, page 327)

Declaration

```
public interface Paletted8Image
implements ByteChannelImage, PalettedIntegerImage
```

All known subclasses

MemoryPaletted8Image (in 14.2.5, page 354)

All classes known to implement interface

MemoryPaletted8Image (in 14.2.5, page 354)

14.1.1.9 Interface **PalettedImage**

This interface defines methods for paletted images. The image data of paletted images are usually integer numbers. Those numbers are index values into a list of colors called the palette or color map. This way, for images with few colors relatively small integers can be used as samples.

Declaration

```
public interface PalettedImage
```

All known subclasses

PalettedIntegerImage (in 14.1.10, page 327), Paletted8Image (in 14.1.8, page 325),
MemoryPaletted8Image (in 14.2.5, page 354)

All known subinterfaces

PalettedIntegerImage (in 14.1.10, page 327)

Method summary

getPalette() Gets the palette of this image.

setPalette(Palette) Sets the palette of this image to the argument palette object.

Methods

- *getPalette*

Palette **getPalette()**

- **Description**

Gets the palette of this image.

- **Returns** – palette object

- *setPalette*

void **setPalette(Palette palette)**

- **Description**

Sets the palette of this image to the argument palette object.

- **Parameters**

- * **palette** – the new palette for this image

14.1.10 Interface PalettedIntegerImage

An empty interface for a paletted image type that uses integer values as samples.

Declaration

```
public interface PalettedIntegerImage  
implements PalettedImage, IntegerImage
```

All known subclasses

Paletted8Image (in 14.1.8, page 325), MemoryPaletted8Image (in 14.2.5, page 354)

All known subinterfaces

Paletted8Image (in 14.1.8, page 325)

14.1.11 Interface PixelImage

The base interface for all image data types in JIU. These image data classes and interfaces share the following properties:

- They are made up of a rectangular grid of pixels (color dots) and have a fixed width (horizontal number of pixels) and height (vertical number of pixels).
- Coordinates to access pixels in the image are defined to run from 0 to WIDTH - 1 in horizontal direction from left to right and in vertical direction from 0 to HEIGHT - 1 (from top to bottom).
- They have one or more channels. A pixel at a given position is made up of the samples (primitive values) of all the channels at that position. A pixel can thus be considered a vector (in the mathematical sense) of one or more samples. Each channel has the same width and height.

Declaration

```
public interface PixelImage
```

All known subclasses

ShortChannellImage (in 14.1.17, page 337), RGBIntegerImage (in 14.1.16, page 336), RGBImage (in 14.1.14, page 333), RGB48Image (in 14.1.13, page 332), RGB24Image (in 14.1.12, page 331), PalettedIntegerImage (in 14.1.10, page 327), Paletted8Image (in 14.1.8, page 325), MemoryShortChannellImage (in 14.2.8, page 358), MemoryRGB48Image (in 14.2.7, page 357), MemoryRGB24Image (in 14.2.6, page 356), MemoryPaletted8Image (in 14.2.5, page 354), MemoryGray8Image (in 14.2.4, page 352), MemoryGray16Image (in 14.2.3, page 350), MemoryByteChannellImage (in 14.2.2, page 345), MemoryBilevelImage (in 14.2.1, page 342), IntegerImage (in 14.1.7, page 322), GrayIntegerImage (in 14.1.6, page 321), Gray8Image (in 14.1.4, page 318), Gray16Image (in 14.1.3, page 317), ByteChannellImage (in 14.1.2, page 314), BilevelImage (in 14.1.1, page 311), BufferedRGB24Image (in 17.1.4, page 433)

All known subinterfaces

RGBImage (in 14.1.14, page 333), IntegerImage (in 14.1.7, page 322)

Method summary

- createCompatibleImage(int, int)** Creates an instance of the same class as this one, with width and height given by the arguments.
- createCopy()** Creates a new image object that will be of the same type as this one, with the same image data, using entirely new resources.
- getAllocatedMemory()** Returns the number of bytes that were dynamically allocated for this image object.
- getBitsPerPixel()** Returns the number of bits per pixel of this image.
- getHeight()** Returns the vertical resolution of the image in pixels.
- getImageType()** If there is a single interface or class that describes the image data type of this class, the object associated with that interface (or class) is returned (or null otherwise).
- getNumChannels()** Returns the number of channels in this image.
- getWidth()** Returns the horizontal resolution of the image in pixels.

Methods

- *createCompatibleImage*

`PixelImage createCompatibleImage(int width, int height)`

- **Description**

Creates an instance of the same class as this one, with width and height given by the arguments.

- **Parameters**

- * **width** – the horizontal resolution of the new image
- * **height** – the vertical resolution of the new image

- **Returns** – the new image

- **Throws**

- * `java.lang.IllegalArgumentException` – if width or height are smaller than one
-

- *createCopy*

`PixelImage createCopy()`

- **Description**

Creates an new image object that will be of the same type as this one, with the same image data, using entirely new resources.

- **Returns** – the new image object

- *getAllocatedMemory*

`long getAllocatedMemory()`

- **Description**

Returns the number of bytes that were dynamically allocated for this image object.

- **Returns** – allocated memory in bytes

- *getBitsPerPixel*

`int getBitsPerPixel()`

- **Description**

Returns the number of bits per pixel of this image. That is the number of bits per sample for all channels of this image. Does not include any transparency channels.

- *getHeight*

`int getHeight()`

- **Description**

Returns the vertical resolution of the image in pixels. Must be one or larger.

- **Returns** – height in pixels

- *getImageType*

`java.lang.Class getImageType()`

- **Description**

If there is a single interface or class that describes the image data type of this class, the object associated with that interface (or class) is returned (or `null` otherwise). This object, if available for two image objects, can be used to find out if they are compatible. Example: (in 14.2.4, page 352) returns `net.sourceforge.jiu.data.Gray8Image.class`.

- *getNumChannels*

int getNumChannels()

- **Description**

Returns the number of channels in this image. Must be one or larger.

- **Returns** – the number of channels

- *getWidth*

int getWidth()

- **Description**

Returns the horizontal resolution of the image in pixels. Must be one or larger.

- **Returns** – width in pixels

14.1.12 *Interface* **RGB24Image**

An empty interface for RGB truecolor images with integer samples that are each eight bits large (thus, 24 bits per pixel).

Declaration

```
public interface RGB24Image
implements ByteChannelImage, RGBIntegerImage
```

All known subclasses

MemoryRGB24Image (in 14.2.6, page 356), BufferedRGB24Image (in 17.1.4, page 433)

All classes known to implement interface

MemoryRGB24Image (in 14.2.6, page 356), BufferedRGB24Image (in 17.1.4, page 433)

14.1.13 *Interface* RGB48Image

An empty interface for RGB truecolor images with integer samples that are each sixteen bits large (thus, 48 bits per pixel).

Declaration

```
public interface RGB48Image
implements ShortChannelImage, RGBIntegerImage
```

All known subclasses

MemoryRGB48Image (in 14.2.7, page 357)

All classes known to implement interface

MemoryRGB48Image (in 14.2.7, page 357)

14.1.14 *Interface* **RGBImage**

An interface for RGB truecolor images. This is an empty interface, useful for labeling an image type as being an RGB image type.

Declaration

```
public interface RGBImage
implements PixelImage, RGBIndex
```

All known subclasses

RGBIntegerImage (in 14.1.16, page 336), RGB48Image (in 14.1.13, page 332), RGB24Image (in 14.1.12, page 331), MemoryRGB48Image (in 14.2.7, page 357), MemoryRGB24Image (in 14.2.6, page 356), BufferedRGB24Image (in 17.1.4, page 433)

All known subinterfaces

RGBIntegerImage (in 14.1.16, page 336)

14.1.15 Interface RGBIndex

This interface provides three `int` constants as index values for the three channels of an RGB image: red, green and blue. The three values are guaranteed to lie in the interval 0 to 2. Furthermore, all three values are different from each other, so that the complete interval from 0 to 2 is used.

Declaration

```
public interface RGBIndex
```

All known subclasses

WebsafePaletteCreator (in 4.2.2, page 172), HueSaturationValue (in 6.1.5, page 206), PCDYCbCrConversion (in 8.1.3, page 227), OrderedDither (in 9.2.5, page 241), ErrorDiffusionDithering (in 9.2.3, page 234), PaletteSerialization (in 10.1.3, page 249), UniformPaletteQuantizer (in 12.2.11, page 293), RGBColorList (in 12.2.10, page 291), RGBColorComparator (in 12.2.9, page 290), RGBColor (in 12.2.8, page 288), PopularityQuantizer (in 12.2.7, page 285), OctreeNode (in 12.2.6, page 282), OctreeColorQuantizer (in 12.2.5, page 279), MedianCutQuantizer (in 12.2.4, page 273), MedianCutNode (in 12.2.3, page 269), ArbitraryPaletteQuantizer (in 12.2.1, page 261), RGBIntegerImage (in 14.1.16, page 336), RGBImage (in 14.1.14, page 333), RGB48Image (in 14.1.13, page 332), RGB24Image (in 14.1.12, page 331), Palette (in 14.2.9, page 363), MemoryRGB48Image (in 14.2.7, page 357), MemoryRGB24Image (in 14.2.6, page 356), BufferedRGB24Image (in 17.1.4, page 433)

All known subinterfaces

RGBImage (in 14.1.14, page 333)

All classes known to implement interface

WebsafePaletteCreator (in 4.2.2, page 172), HueSaturationValue (in 6.1.5, page 206), PCDYCbCrConversion (in 8.1.3, page 227), OrderedDither (in 9.2.5, page 241), ErrorDiffusionDithering (in 9.2.3, page 234), PaletteSerialization (in 10.1.3, page 249), UniformPaletteQuantizer (in 12.2.11, page 293), RGBColorList (in 12.2.10, page 291), RGBColorComparator (in 12.2.9, page 290), RGBColor (in 12.2.8, page 288), PopularityQuantizer (in 12.2.7, page 285), OctreeNode (in 12.2.6, page 282), OctreeColorQuantizer (in 12.2.5, page 279), MedianCutQuantizer (in 12.2.4, page 273), MedianCutNode (in 12.2.3, page 269), ArbitraryPaletteQuantizer (in 12.2.1, page 261), Palette (in 14.2.9, page 363)

Field summary

INDEX_BLUE The index value for the blue channel.
INDEX_GREEN The index value for the green channel.
INDEX_RED The index value for the red channel.

Fields

- `int` **INDEX_RED**
 - The index value for the red channel.
- `int` **INDEX_GREEN**

- The index value for the green channel.
- `int` **INDEX_BLUE**
 - The index value for the blue channel.

14.1.16 *Interface* **RGBIntegerImage**

An interface for RGB truecolor images that have integer samples. A combination of (in 14.1.7, page 322) and (in 14.1.14, page 333).

Declaration

```
public interface RGBIntegerImage
implements IntegerImage, RGBImage
```

All known subclasses

RGB48Image (in 14.1.13, page 332), RGB24Image (in 14.1.12, page 331), MemoryRGB48Image (in 14.2.7, page 357), MemoryRGB24Image (in 14.2.6, page 356), BufferedRGB24Image (in 17.1.4, page 433)

All known subinterfaces

RGB48Image (in 14.1.13, page 332), RGB24Image (in 14.1.12, page 331)

14.1.17 Interface ShortChannelImage

An extension of the (in 14.1.7, page 322) interface that restricts the image to **short** samples. The minimum sample value for all channels is 0, the maximum sample value 65535 ($2^{16} - 1$). Number of channels and resolution must be given to the constructor and cannot be changed after creation.

Each channel of the image is made up of **short** values. Note that shorts in Java are signed, they can take values from -32768 to 32767. If you use (in 14.1.7, page 322)'s `getSample` and `putSample` methods you don't have to deal with this, you always get **int** samples that are in the 0 .. 65535 interval.

To manually convert a Java **short** value to an **int** value in the range of 0 to 65535, do the following:

```
short s = ...; // initialize short value
int i = s & 0xffff;
// i now is a value between 0 and 65535
```

Declaration

```
public interface ShortChannelImage
implements IntegerImage
```

All known subclasses

RGB48Image (in 14.1.13, page 332), MemoryShortChannelImage (in 14.2.8, page 358), MemoryRGB48Image (in 14.2.7, page 357), MemoryGray16Image (in 14.2.3, page 350), Gray16Image (in 14.1.3, page 317)

All known subinterfaces

RGB48Image (in 14.1.13, page 332), Gray16Image (in 14.1.3, page 317)

All classes known to implement interface

MemoryShortChannelImage (in 14.2.8, page 358)

Method summary

- clear(int, short)** Sets all samples of one channel to a new value.
- clear(short)** Sets all samples of the first channel to the argument short value.
- getShortSample(int, int)** Returns a single short sample from the first channel and the specified position.
- getShortSample(int, int, int)** Returns a single short sample from the image.
- getShortSamples(int, int, int, int, int, short[], int)** Copies samples from this image to a short array.
- putShortSample(int, int, int, short)** Sets one short sample in one channel to a new value.
- putShortSample(int, int, short)** Sets one short sample in the first channel (index 0) to a new value.
- putShortSamples(int, int, int, int, int, short[], int)** Copies a number of samples from the argument array to this image.

Methods

- *clear*

`void clear(int channelIndex, short newValue)`

- **Description**

Sets all samples of one channel to a new value.

- **Parameters**

- * `channelIndex` – zero-based index of the channel to be cleared (must be smaller than (in 14.1.11, page 330)
 - * `newValue` – all samples in the channel will be set to this value
-

- *clear*

`void clear(short newValue)`

- **Description**

Sets all samples of the first channel to the argument short value. Equal to `clear(0, newValue);`.

- **Parameters**

- * `newValue` – all samples in the first channel are set to this value

- **See also**

- * `ShortChannelImage.clear(int,short)` (in 14.1.17, page 338)
 - * `IntegerImage.clear(int)` (in 14.1.7, page 323)
 - * `IntegerImage.clear(int,int)` (in 14.1.7, page 323)
-

- *getShortSample*

`short getShortSample(int x, int y)`

- **Description**

Returns a single short sample from the first channel and the specified position. A call to this method is the same as `getShortSample(0, x, y)`.

- **Parameters**

- * `x` – horizontal position of the sample to be returned (must be between 0 and (in 14.1.11, page 330)- 1
- * `y` – vertical position of the sample to be returned (must be between 0 and (in 14.1.11, page 329)- 1

- **Returns** – the requested short sample

- *getShortSample*

`short getShortSample(int channel, int x, int y)`

- **Description**

Returns a single short sample from the image. When possible, try copying several samples at a time for higher speed ((in 14.1.17, page 339)).

- **Parameters**

- * `channel` – the number of the channel of the sample; must be from 0 to (in 14.1.11, page 330)- 1
- * `x` – the column of the sample to be returned; must be from 0 to (in 14.1.11, page 330)- 1
- * `y` – the row of the sample; must be from 0 to (in 14.1.11, page 329)- 1

- **Returns** – the sample, a single short value
 - **Throws**
 - * `java.lang.IllegalArgumentException` – if the arguments hurt one of the preconditions above
 - **See also**
 - * `ShortChannelImage.getShortSamples(int,int,int,int,int,short[],int)` (in 14.1.17, page 339)
-

- *getShortSamples*

```
void getShortSamples( int channelIndex, int x, int y, int w, int h,
short[] dest, int destOffset )
```

- **Description**

Copies samples from this image to a short array. Copies num samples in row y of channel channel, starting at horizontal offset x. Data will be written to the dest array, starting at offset destOffset. Data will be copied from one row only, so a maximum of getWidth() samples can be copied with a call to this method.
 - **Parameters**
 - * `channelIndex` – the index of the channel to be copied from; must be from 0 to `getNumChannels() - 1`
 - * `x` – the horizontal offset where copying will start; must be from 0 to `getWidth() - 1`
 - * `y` – the row from which will be copied; must be from 0 to `getHeight() - 1`
 - * `w` – number of columns to be copied
 - * `h` – number of rows to be copied
 - * `dest` – the array where the data will be copied to; must have a length of at least `destOffset + num`
 - * `destOffset` – the offset into dest where this method will start copying data
 - **Throws**
 - * `java.lang.IllegalArgumentException` – if the arguments hurt one of the many preconditions above
-

- *putShortSample*

```
void putShortSample( int channel, int x, int y, short newValue )
```

- **Description**

Sets one short sample in one channel to a new value.
-

- *putShortSample*

```
void putShortSample( int x, int y, short newValue )
```

- **Description**

Sets one short sample in the first channel (index 0) to a new value. Result is equal to `putShortSample(0, x, y, newValue);`.
-

- *putShortSamples*

```
void putShortSamples( int channel, int x, int y, int w, int h, short[]
src, int srcOffset )
```

- **Description**

Copies a number of samples from the argument array to this image.

14.1.18 Interface TransparencyInformation

An interface that represents transparency information which may be available for a pixel image. Transparency information describes how an image is supposed to be drawn on a pixel background (e.g. another image). That way, irregularly shaped images can easily be handled by excluding those pixels of a rectangular image that are not part of the image.

Declaration

```
public interface TransparencyInformation
```

Method summary

getAlphaChannelImage() Returns an image object that contains an alpha channel.
getTransparencyIndex() If there is a transparency index, this method returns it.
setAlphaChannelImage(IntegerImage) Set a new alpha channel image object.
setTransparencyIndex(Integer) Set a new transparency value.

Methods

- *getAlphaChannelImage*
`IntegerImage getAlphaChannelImage()`
 - **Description**
Returns an image object that contains an alpha channel. The first channel of that image is supposed to be the alpha channel.
 - **Returns** – the alpha channel image object
 - **See also**
 - * `TransparencyInformation.setAlphaChannelImage(IntegerImage)` (in 14.1.18, page 340)
- *getTransparencyIndex*
`java.lang.Integer getTransparencyIndex()`
 - **Description**
If there is a transparency index, this method returns it. Otherwise, the return value is undefined.
 - **Returns** – transparency index
 - **See also**
 - * `TransparencyInformation.setTransparencyIndex(java.lang.Integer)` (in 14.1.18, page 341)
- *setAlphaChannelImage*
`void setAlphaChannelImage(IntegerImage newImage)`
 - **Description**
Set a new alpha channel image object.
 - **See also**
 - * `TransparencyInformation.getAlphaChannelImage()` (in 14.1.18, page 340)

- *setTransparencyIndex*

void setTransparencyIndex(java.lang.Integer newValue)

- **Description**

- Set a new transparency value. Can be null. However, if the value is non-null, it must encapsulate an integer number which is 0 or larger.

- **Parameters**

- * **newValue** – new transparency index

- **Throws**

- * **java.lang.IllegalArgumentException** – if the argument is non-null and contains a negative value

- **See also**

- * **TransparencyInformation.getAlphaChannelImage()** (in 14.1.18, page 340)

14.2 Classes

14.2.1 Class MemoryBilevelImage

An implementation of the (in 14.1.1, page 311) interface that stores image data in a `byte` array in memory. An image of `width` times `height` pixels will require $(width + 7) / 8 * height$ bytes of memory.

Declaration

```
public class MemoryBilevelImage
  extends java.lang.Object
  implements BilevelImage
```

Constructor summary

MemoryBilevelImage(int, int) Create a new MemoryBilevelImage object with the specified resolution.

Method summary

```
clear(int)
clear(int, int)
createCompatibleImage(int, int)
createCopy()
getAllocatedMemory()
getBitsPerPixel()
getHeight()
getImageType()
getMaxSample(int)
getNumChannels()
getPackedBytes(int, int, int, byte[], int, int)
getSample(int, int)
getSample(int, int, int)
getSamples(int, int, int, int, int, int[], int)
getWidth()
isBlack(int, int)
isWhite(int, int)
putBlack(int, int)
putPackedBytes(int, int, int, byte[], int, int)
putSample(int, int, int)
putSample(int, int, int, int)
putSamples(int, int, int, int, int, int[], int)
putWhite(int, int)
```

Constructors

- *MemoryBilevelImage*
 public MemoryBilevelImage(int width, int height)

- **Description**

Create a new MemoryBilevelImage object with the specified resolution.

- **Parameters**

- * **width** – the horizontal resolution of the new image, must be larger than zero
- * **height** – the vertical resolution of the new image, must be larger than zero

- **Throws**

- * `java.lang.IllegalArgumentException` – if any of the two parameters is smaller than one

Methods

- *clear*
`public void clear(int newValue)`
- *clear*
`public void clear(int channelIndex, int newValue)`
- *createCompatibleImage*
`public PixelImage createCompatibleImage(int width, int height)`
- *createCopy*
`public PixelImage createCopy()`
- *getAllocatedMemory*
`public long getAllocatedMemory()`
- *getBitsPerPixel*
`public int getBitsPerPixel()`
- *getHeight*
`public int getHeight()`
- *getImageType*
`public java.lang.Class getImageType()`
- *getMaxSample*
`public int getMaxSample(int channelIndex)`
- *getNumChannels*
`public int getNumChannels()`
- *getPackedBytes*
`void getPackedBytes(int x, int y, int numSamples, byte[] dest, int destOffset, int destBitOffset)`
- **Description copied from BilevelImage (in 14.1.1, page 311)**
Sets a number of samples in the argument array from this image.
- **Parameters**
 - * **x** – horizontal position of first sample of this image to read
 - * **y** – vertical position of samples to be read from this image
 - * **numSamples** – number of samples to be set
 - * **dest** – array with packed pixels to which samples are copied
 - * **destOffset** – index into dest array of the first byte value to write sample values to
 - * **destBitOffset** – index of first bit of `dest[destOffset]` to write a sample to (0 is leftmost, 1 is second-leftmost up to 7, which is the rightmost)

-
- *getSample*
public int getSample(int x, int y)
 - *getSample*
public int getSample(int channelIndex, int x, int y)
 - *getSamples*
public void getSamples(int channelIndex, int x, int y, int w, int h, int[] dest, int destOffset)
 - *getWidth*
public int getWidth()
 - *isBlack*
public boolean isBlack(int x, int y)
 - *isWhite*
public boolean isWhite(int x, int y)
 - *putBlack*
public void putBlack(int x, int y)
 - *putPackedBytes*
void putPackedBytes(int x, int y, int numSamples, byte[] src, int srcOffset, int srcBitOffset)
 - **Description copied from BilevelImage (in 14.1.1, page 311)**
Sets a number of samples in the image from the argument array data.
 - **Parameters**
 - * **x** – horizontal position of first sample to be set
 - * **y** – vertical position of samples to be set
 - * **numSamples** – number of samples to be set
 - * **src** – array with packed pixels to be set
 - * **srcOffset** – index into src array of the first byte value to read sample values from
 - * **srcBitOffset** – index of first bit of src[srcOffset] to read a sample from (0 is leftmost, 1 is second-leftmost up to 7, which is the rightmost)
-
- *putSample*
public void putSample(int x, int y, int newValue)
 - *putSample*
public void putSample(int channelIndex, int x, int y, int newValue)
 - *putSamples*
public void putSamples(int channelIndex, int x, int y, int w, int h, int[] src, int srcOffset)
 - *putWhite*
public void putWhite(int x, int y)

14.2.2 Class MemoryByteChannelImage

An implementation of (in 14.1.2, page 314) that stores image channels as `byte[]` arrays in memory. An image can have an arbitrary number of channels.

This class is abstract because it is merely a data container. It takes a subclass like (in 14.2.4, page 352) to give meaning to the values.

Declaration

```
public abstract class MemoryByteChannelImage
  extends java.lang.Object
  implements ByteChannelImage
```

All known subclasses

MemoryRGB24Image (in 14.2.6, page 356), MemoryPaletted8Image (in 14.2.5, page 354), MemoryGray8Image (in 14.2.4, page 352)

Constructor summary

MemoryByteChannelImage(int, int, int) Create an image of byte channels.

Method summary

checkPositionAndNumber(int, int, int, int, int) Throws an exception if the arguments do not form a valid horizontal sequence of samples.

```
clear(byte)
clear(int)
clear(int, byte)
clear(int, int)
createCompatibleImage(int, int)
createCopy()
getAllocatedMemory()
getBitsPerPixel()
getBytesSample(int, int)
getBytesSample(int, int, int)
getBytesSamples(int, int, int, int, int, byte[], int)
getHeight()
getMaxSample(int)
getNumChannels()
getSample(int, int)
getSample(int, int, int)
getSamples(int, int, int, int, int, int[], int)
getWidth()
putBytesSample(int, int, byte)
putBytesSample(int, int, int, byte)
putBytesSamples(int, int, int, int, int, byte[], int)
putSample(int, int, int)
putSample(int, int, int, int)
putSamples(int, int, int, int, int, int[], int)
```

Constructors

- *MemoryByteChannelImage*

```
public MemoryByteChannelImage( int numChannels, int width, int height
)
```

- **Description**

Create an image of byte channels. Image data will be completely in memory, so memory requirements are `width * height * numChannels` bytes. Note that the data will not be initialized, so you should not assume anything about its content.

- **Parameters**

- * `numChannels` – the number of channels in this image, must be non-zero and positive
- * `width` – the horizontal resolution, must be non-zero and positive
- * `height` – the vertical resolution, must be non-zero and positive

- **Throws**

- * `java.lang.IllegalArgumentException` – if any of the parameters are invalid or if width times height exceeds two GB
- * `OutOfMemoryException` – if there is not enough free memory for the specified resolution

Methods

- *checkPositionAndNumber*

```
protected void checkPositionAndNumber( int channel, int x, int y, int w,
int h )
```

- **Description**

Throws an exception if the arguments do not form a valid horizontal sequence of samples. To be valid, all of the following requirements must be met:

- *clear*

```
void clear( byte newValue )
```

- **Description copied from ByteChannelImage (in 14.1.2, page 314)**

Sets all samples of the first channel to the argument byte value. Equal to `clear(0, newValue);`.

- **Parameters**

- * `newValue` – all samples in the first channel are set to this value

- **See also**

- * `ByteChannelImage.clear(int,byte)` (in 14.1.2, page 315)
- * `IntegerImage.clear(int)` (in 14.1.7, page 323)
- * `IntegerImage.clear(int,int)` (in 14.1.7, page 323)

- *clear*

```
public void clear( int newValue )
```

- *clear*

```
void clear( int channelIndex, byte newValue )
```

- **Description copied from ByteChannellImage (in 14.1.2, page 314)**
Sets all samples of one channel to a new value.

- **Parameters**

- * **channelIndex** – zero-based index of the channel to be cleared (must be smaller than (in 14.1.11, page 330)
- * **newValue** – all samples in the channel will be set to this value

- *clear*

public void clear(int channelIndex, int newValue)

- *createCompatibleImage*

public abstract PixelImage createCompatibleImage(int width, int height)

- *createCopy*

public PixelImage createCopy()

- *getAllocatedMemory*

public long getAllocatedMemory()

- *getBitsPerPixel*

public int getBitsPerPixel()

- *getBytesSample*

byte getByteSample(int x, int y)

- **Description copied from ByteChannellImage (in 14.1.2, page 314)**

Returns a single byte sample from the first channel and the specified position. A call to this method is the same as `getByteSample(0, x, y)`.

- **Parameters**

- * **x** – horizontal position of the sample to be returned (must be between 0 and (in 14.1.11, page 330)- 1
- * **y** – vertical position of the sample to be returned (must be between 0 and (in 14.1.11, page 329)- 1

- **Returns** – the requested byte sample
-

- *getBytesSample*

byte getByteSample(int channel, int x, int y)

- **Description copied from ByteChannellImage (in 14.1.2, page 314)**

Returns a single byte sample from the image. When possible, try copying several samples at a time for higher speed ((in 14.1.2, page 316)).

- **Parameters**

- * **channel** – the number of the channel of the sample; must be from 0 to (in 14.1.11, page 330)- 1
- * **x** – the column of the sample to be returned; must be from 0 to (in 14.1.11, page 330)- 1
- * **y** – the row of the sample; must be from 0 to (in 14.1.11, page 329)- 1

- **Returns** – the sample, a single byte value

- **Throws**

- * **java.lang.IllegalArgumentException** – if the arguments hurt one of the preconditions above

- **See also**

* `ByteChannelImage.getBytesSamples(int,int,int,int,int,byte[],int)` (in 14.1.2, page 316)

- *getBytesSamples*

```
void getBytesSamples( int channelIndex, int x, int y, int w, int h, byte[]
dest, int destOffset )
```

- **Description copied from ByteChannelImage (in 14.1.2, page 314)**

Copies samples from this image to a byte array. Copies `num` samples in row `y` of channel `channel`, starting at horizontal offset `x`. Data will be written to the `dest` array, starting at offset `destOffset`. Data will be copied from one row only, so a maximum of `getWidth()` samples can be copied with a call to this method.

- **Parameters**

- * `channelIndex` – the index of the channel to be copied from; must be from 0 to `getNumChannels() - 1`
- * `x` – the horizontal offset where copying will start; must be from 0 to `getWidth() - 1`
- * `y` – the row from which will be copied; must be from 0 to `getHeight() - 1`
- * `w` – the number of columns to be copied
- * `h` – the number of rows to be copied
- * `dest` – the array where the data will be copied to; must have a length of at least `destOffset + num`
- * `destOffset` – the offset into `dest` where this method will start copying data

- **Throws**

- * `java.lang.IllegalArgumentException` – if the arguments hurt one of the many preconditions above
-

- *getHeight*

```
public final int getHeight( )
```

- *getMaxSample*

```
public int getMaxSample( int channel )
```

- *getNumChannels*

```
public int getNumChannels( )
```

- *getSample*

```
public final int getSample( int x, int y )
```

- *getSample*

```
public final int getSample( int channel, int x, int y )
```

- *getSamples*

```
public void getSamples( int channel, int x, int y, int w, int h, int[]
dest, int destOffs )
```

- *getWidth*

```
public final int getWidth( )
```

- *putByteSample*

```
void putByteSample( int x, int y, byte newValue )
```

- **Description copied from ByteChannelImage (in 14.1.2, page 314)**

Sets one byte sample in the first channel (index 0) to a new value. Result is equal to `putByteSample(0, x, y, newValue);`.

- *putByteSample*
void **putByteSample**(int channel, int x, int y, byte newValue)

– Description copied from ByteChannelImage (in 14.1.2, page 314)
Sets one byte sample in one channel to a new value.

- *putByteSamples*
void **putByteSamples**(int channel, int x, int y, int w, int h, byte[] src, int srcOffset)

– Description copied from ByteChannelImage (in 14.1.2, page 314)
Copies a number of samples from the argument array to this image.

- *putSample*
public final void **putSample**(int x, int y, int newValue)

- *putSample*
public final void **putSample**(int channel, int x, int y, int newValue)

- *putSamples*
public void **putSamples**(int channel, int x, int y, int w, int h, int[] src, int srcOffs)

14.2.3 Class MemoryGray16Image

An implementation of (in 14.1.3, page 317) that keeps the complete image in memory. This class inherits most of its functionality from its parent class (in 14.2.8, page 358), using one `short` channel.

Declaration

```
public class MemoryGray16Image
extends net.sourceforge.jiu.data.MemoryShortChannelImage (in 14.2.8, page 358)
implements Gray16Image
```

Constructor summary

MemoryGray16Image(int, int) Creates a new MemoryGray16Image object with the specified resolution.

Method summary

```
createCompatibleImage(int, int)
getImageType()
isBlack(int, int)
isWhite(int, int)
putBlack(int, int)
putWhite(int, int)
```

Constructors

- *MemoryGray16Image*
public MemoryGray16Image(int width, int height)
 - **Description**
 Creates a new MemoryGray16Image object with the specified resolution. Simply gives 1 (for one channel) and the two resolution arguments to the super constructor (of the parent class (in 14.2.8, page 358)).
 - **Parameters**
 - * **width** – the horizontal resolution, must be larger than zero
 - * **height** – the vertical resolution, must be larger than zero

Methods

- *createCompatibleImage*
public abstract PixelImage createCompatibleImage(int width, int height)
- *getImageType*
public java.lang.Class getImageType()
- *isBlack*
public boolean isBlack(int x, int y)

- *isWhite*
public boolean **isWhite**(int x, int y)
- *putBlack*
public void **putBlack**(int x, int y)
- *putWhite*
public void **putWhite**(int x, int y)

14.2.4 Class MemoryGray8Image

An implementation of (in 14.1.4, page 318) that keeps the complete image in memory. This class inherits most of its functionality from its parent class (in 14.2.2, page 345), using one byte channel.

Declaration

```
public class MemoryGray8Image
extends net.sourceforge.jiu.data.MemoryByteChannelImage (in 14.2.2, page 345)
implements Gray8Image
```

Constructor summary

MemoryGray8Image(int, int) Creates a new MemoryGray8Image object with the specified resolution.

Method summary

```
createCompatibleImage(int, int)
getImageType()
isBlack(int, int)
isWhite(int, int)
putBlack(int, int)
putWhite(int, int)
```

Constructors

- *MemoryGray8Image*
public MemoryGray8Image(int width, int height)
 - **Description**
Creates a new MemoryGray8Image object with the specified resolution. Simply gives 1 (for one channel) and the two resolution arguments to the super constructor (of the parent class (in 14.2.2, page 345)).
 - **Parameters**
 - * **width** – the horizontal resolution, must be non-zero and positive
 - * **height** – the vertical resolution, must be non-zero and positive

Methods

- *createCompatibleImage*
public abstract PixelImage createCompatibleImage(int width, int height)
- *getImageType*
public java.lang.Class getImageType()
- *isBlack*
public boolean isBlack(int x, int y)
- *isWhite*
public boolean isWhite(int x, int y)

- *putBlack*
public void **putBlack**(int x, int y)
- *putWhite*
public void **putWhite**(int x, int y)

14.2.5 Class MemoryPaletted8Image

This class stores a paletted image with one byte per sample in memory.

See also

- `ByteChannelImage` (in 14.1.2, page 314)
- `IntegerImage` (in 14.1.7, page 322)
- `Palette` (in 14.2.9, page 363)

Declaration

```
public class MemoryPaletted8Image
  extends net.sourceforge.jiu.data.MemoryByteChannelImage (in 14.2.2, page 345)
  implements Paletted8Image
```

Constructor summary

MemoryPaletted8Image(int, int) Create an image of byte channels.
MemoryPaletted8Image(int, int, Palette)

Method summary

checkPalette(Palette)
createCompatibleImage(int, int)
getAllocatedMemory()
getImageType()
getMaxSample(int)
getPalette() Returns this image's palette.
getTypeDescription()
setPalette(Palette) Sets this image's palette to a new value.

Constructors

- *MemoryPaletted8Image*
 public **MemoryPaletted8Image(int width, int height)**
 - **Description**
 Create an image of byte channels. Image data will be completely in memory, so memory requirements are `width * height * numChannels` bytes. Note that the data will not be initialized, so you should not assume anything about its content.
 - **Parameters**
 - * `width` – the horizontal resolution, must be non-zero and positive
 - * `height` – the vertical resolution, must be non-zero and positive
 - **Throws**
 - * `java.lang.IllegalArgumentException` – if any of the parameters are smaller than 1
- *MemoryPaletted8Image*
 public **MemoryPaletted8Image(int width, int height, Palette palette)**

Methods

- *checkPalette*
public static void checkPalette(Palette palette)
- *createCompatibleImage*
public abstract PixelImage createCompatibleImage(int width, int height)
- *getAllocatedMemory*
public long getAllocatedMemory()
- *getImageType*
public java.lang.Class getImageType()
- *getMaxSample*
public int getMaxSample(int channel)
- *getPalette*
public Palette getPalette()
 - **Description**
Returns this image's palette.
 - **See also**
 - * MemoryPaletted8Image.setPalette(Palette) (in 14.2.5, page 355)
- *getTypeDescription*
public java.lang.String getTypeDescription()
- *setPalette*
public void setPalette(Palette palette)
 - **Description**
Sets this image's palette to a new value.
 - **See also**
 - * MemoryPaletted8Image.getPalette() (in 14.2.5, page 355)

14.2.6 Class MemoryRGB24Image

A class to store 24 bit RGB truecolor images in memory.

See also

- `RGB24Image` (in 14.1.12, page 331)

Declaration

```
public class MemoryRGB24Image
extends net.sourceforge.jiu.data.MemoryByteChannelImage (in 14.2.2, page 345)
implements RGB24Image
```

Constructor summary

MemoryRGB24Image(int, int) Creates a new object of this class, with width and height as specified by the arguments.

Method summary

createCompatibleImage(int, int)
getImageType()

Constructors

- *MemoryRGB24Image*
public MemoryRGB24Image(int width, int height)
 - **Description**
Creates a new object of this class, with width and height as specified by the arguments.
 - **Parameters**
 - * **width** – the horizontal resolution of the new image in pixels
 - * **height** – the vertical resolution of the new image in pixels

Methods

- *createCompatibleImage*
public abstract PixelImage createCompatibleImage(int width, int height)
- *getImageType*
public java.lang.Class getImageType()

14.2.7 Class MemoryRGB48Image

A class to store 48 bit RGB truecolor images in memory.

See also

- `RGB24Image` (in 14.1.12, page 331)

Declaration

```
public class MemoryRGB48Image
extends net.sourceforge.jiu.data.MemoryShortChannelImage (in 14.2.8, page 358)
implements RGB48Image
```

Constructor summary

MemoryRGB48Image(int, int) Creates a new object of this class, with width and height as specified by the arguments.

Method summary

createCompatibleImage(int, int)
getImageType()

Constructors

- *MemoryRGB48Image*
public MemoryRGB48Image(int width, int height)
 - **Description**
Creates a new object of this class, with width and height as specified by the arguments.
 - **Parameters**
 - * **width** – the horizontal resolution of the new image in pixels
 - * **height** – the vertical resolution of the new image in pixels

Methods

- *createCompatibleImage*
public abstract PixelImage createCompatibleImage(int width, int height)
- *getImageType*
public java.lang.Class getImageType()

14.2.8 Class MemoryShortChannelImage

An implementation of (in 14.1.17, page 337) that stores image channels as `short[]` arrays in memory. An image can have an arbitrary number of channels.

This class is abstract because it is merely a data container. It takes a subclass like (in 14.2.3, page 350) to give meaning to the values.

Declaration

```
public abstract class MemoryShortChannelImage
    extends java.lang.Object
    implements ShortChannelImage
```

All known subclasses

MemoryRGB48Image (in 14.2.7, page 357), MemoryGray16Image (in 14.2.3, page 350)

Constructor summary

MemoryShortChannelImage(int, int, int) Create an image of short channels.

Method summary

checkPositionAndNumber(int, int, int, int, int) Throws an exception if the arguments do not form a valid horizontal sequence of samples.

clear(int)

clear(int, int)

clear(int, short)

clear(short)

createCompatibleImage(int, int)

createCopy()

getAllocatedMemory()

getBitsPerPixel()

getHeight()

getMaxSample(int)

getNumChannels()

getSample(int, int)

getSample(int, int, int)

getSamples(int, int, int, int, int, int[], int)

getShortSample(int, int)

getShortSample(int, int, int)

getShortSamples(int, int, int, int, int, short[], int)

getWidth()

putSample(int, int, int)

putSample(int, int, int, int)

putSamples(int, int, int, int, int, int[], int)

putShortSample(int, int, int, short)

putShortSample(int, int, short)

putShortSamples(int, int, int, int, int, short[], int)

Constructors

- *MemoryShortChannelImage*

```
public MemoryShortChannelImage( int numChannels, int width, int height
                                )
```

 - **Description**
 Create an image of short channels. Image data will be completely in memory, so memory requirements are `width * height * numChannels * 2` bytes.
 - **Parameters**
 - * `numChannels` – the number of channels in this image, must be larger than zero
 - * `width` – the horizontal resolution, must be larger than zero
 - * `height` – the vertical resolution, must be larger than zero

Methods

- *checkPositionAndNumber*

```
protected void checkPositionAndNumber( int channel, int x, int y, int w,
                                        int h )
```

 - **Description**
 Throws an exception if the arguments do not form a valid horizontal sequence of samples. To be valid, all of the following requirements must be met:
- *clear*

```
public void clear( int newValue )
```
- *clear*

```
public void clear( int channelIndex, int newValue )
```
- *clear*

```
void clear( int channelIndex, short newValue )
```

 - **Description copied from ShortChannelImage (in 14.1.17, page 337)**
 Sets all samples of one channel to a new value.
 - **Parameters**
 - * `channelIndex` – zero-based index of the channel to be cleared (must be smaller than (in 14.1.11, page 330))
 - * `newValue` – all samples in the channel will be set to this value
- *clear*

```
void clear( short newValue )
```

 - **Description copied from ShortChannelImage (in 14.1.17, page 337)**
 Sets all samples of the first channel to the argument short value. Equal to `clear(0, newValue);`.
 - **Parameters**
 - * `newValue` – all samples in the first channel are set to this value
 - **See also**
 - * `ShortChannelImage.clear(int,short)` (in 14.1.17, page 338)

- * IntegerImage.clear(int) (in 14.1.7, page 323)
- * IntegerImage.clear(int,int) (in 14.1.7, page 323)

- *createCompatibleImage*

public abstract PixelImage createCompatibleImage(int width, int height)

- *createCopy*

public PixelImage createCopy()

- *getAllocatedMemory*

public long getAllocatedMemory()

- *getBitsPerPixel*

public int getBitsPerPixel()

- *getHeight*

public final int getHeight()

- *getMaxSample*

public int getMaxSample(int channel)

- *getNumChannels*

public int getNumChannels()

- *getSample*

public final int getSample(int x, int y)

- *getSample*

public final int getSample(int channel, int x, int y)

- *getSamples*

public void getSamples(int channel, int x, int y, int w, int h, int[]
dest, int destOffs)

- *getShortSample*

short getShortSample(int x, int y)

- **Description copied from ShortChannelImage (in 14.1.17, page 337)**

Returns a single short sample from the first channel and the specified position. A call to this method is the same as getShortSample(0, x, y).

- **Parameters**

- * x – horizontal position of the sample to be returned (must be between 0 and (in 14.1.11, page 330)– 1
- * y – vertical position of the sample to be returned (must be between 0 and (in 14.1.11, page 329)– 1

- **Returns** – the requested short sample
-

- *getShortSample*

short getShortSample(int channel, int x, int y)

- **Description copied from ShortChannelImage (in 14.1.17, page 337)**

Returns a single short sample from the image. When possible, try copying several samples at a time for higher speed ((in 14.1.17, page 339)).

- **Parameters**

- * **channel** – the number of the channel of the sample; must be from 0 to (in 14.1.11, page 330)– 1
- * **x** – the column of the sample to be returned; must be from 0 to (in 14.1.11, page 330)– 1
- * **y** – the row of the sample; must be from 0 to (in 14.1.11, page 329)– 1
- **Returns** – the sample, a single short value
- **Throws**
 - * `java.lang.IllegalArgumentException` – if the arguments hurt one of the preconditions above
- **See also**
 - * `ShortChannelImage.getShortSamples(int,int,int,int,int,short[],int)` (in 14.1.17, page 339)

- *getShortSamples*

void getShortSamples(int channelIndex, int x, int y, int w, int h, short[] dest, int destOffset)

- **Description copied from ShortChannelImage (in 14.1.17, page 337)**
Copies samples from this image to a short array. Copies num samples in row y of channel channel, starting at horizontal offset x. Data will be written to the dest array, starting at offset destOffset. Data will be copied from one row only, so a maximum of getWidth() samples can be copied with a call to this method.
- **Parameters**
 - * **channelIndex** – the index of the channel to be copied from; must be from 0 to getNumChannels() – 1
 - * **x** – the horizontal offset where copying will start; must be from 0 to getWidth() – 1
 - * **y** – the row from which will be copied; must be from 0 to getHeight() – 1
 - * **w** – number of columns to be copied
 - * **h** – number of rows to be copied
 - * **dest** – the array where the data will be copied to; must have a length of at least destOffset + num
 - * **destOffset** – the offset into dest where this method will start copying data
- **Throws**
 - * `java.lang.IllegalArgumentException` – if the arguments hurt one of the many preconditions above

- *getWidth*

public final int getWidth()

- *putSample*

public final void putSample(int x, int y, int newValue)

- *putSample*

public final void putSample(int channel, int x, int y, int newValue)

- *putSamples*

public void putSamples(int channel, int x, int y, int w, int h, int[] src, int srcOffs)

- *putShortSample*

void putShortSample(int channel, int x, int y, short newValue)

- **Description copied from ShortChannelImage (in 14.1.17, page 337)**
Sets one short sample in one channel to a new value.

- *putShortSample*

void **putShortSample**(int x, int y, short newValue)

- **Description copied from ShortChannelImage (in 14.1.17, page 337)**

Sets one short sample in the first channel (index 0) to a new value. Result is equal to `putShortSample(0, x, y, newValue);`.

- *putShortSamples*

void **putShortSamples**(int channel, int x, int y, int w, int h, short[] src, int srcOffset)

- **Description copied from ShortChannelImage (in 14.1.17, page 337)**

Copies a number of samples from the argument array to this image.

14.2.9 Class Palette

This class represents a palette, a list of RGB colors. An RGB color here has three int values for its red, green and blue intensity. Each intensity value must be larger than or equal to zero and smaller than or equal to the maximum intensity value that can be given to the constructor (in 14.2.9, page 364). This maximum value is typically 255. Note that the number of entries in a palette is restricted only by the element index type `int` so that palettes with more than 256 entries are no problem. When accessing (reading or writing) samples of this palette, use the constants (in 14.1.15, page 334), (in 14.1.15, page 334) and (in 14.1.15, page 335) of this class to define a color channel.

See also

- `PalettedImage` (in 14.1.9, page 326)

Declaration

```
public class Palette
  extends java.lang.Object
  implements RGBIndex
```

Constructor summary

- Palette(int)** Create a palette with the given number of entries and a maximum value of 255.
- Palette(int, int)** Create a palette with the given number of entries and a maximum value for each sample.

Method summary

- clone()** Creates a copy of this palette, allocating a new `Palette` object and copying each RGB triplet to the new palette.
- getAllocatedMemory()** Returns the amount of memory in bytes allocated for this palette.
- getMaxValue()** Returns the maximum value allowed for a sample.
- getNumEntries()** Returns the number of entries in this palette.
- getSample(int, int)** Returns one of the samples of this palette.
- getSamples(int)** Returns all samples of one channel as an `int` array.
- isBlackAndWhite()** Checks if all entries of this palette are either black or white.
- isGray()** Checks if this palette is gray, i.e., checks if all entries are gray.
- put(int, int, int, int)**
- putSample(int, int, int)** Sets one sample of one color entry in the palette to a new value.

Constructors

- *Palette*
 - public Palette(int numEntries)**
 - **Description**
 - Create a palette with the given number of entries and a maximum value of 255.

– **Parameters**

* **numEntries** – the number of entries to be accessible in this palette

• *Palette*

public Palette(int numEntries, int maxValue)

– **Description**

Create a palette with the given number of entries and a maximum value for each sample.

– **Parameters**

* **numEntries** – the number of entries to be accessible in this palette

* **maxValue** – the maximum value to be allowed for each sample

Methods

• *clone*

public java.lang.Object clone()

– **Description**

Creates a copy of this palette, allocating a new Palette object and copying each RGB triplet to the new palette. Then returns the new palette. Thus, a "deep" copy of this Palette object is created, not a "shallow" one.

– **Returns** – newly-created palette

• *getAllocatedMemory*

public long getAllocatedMemory()

– **Description**

Returns the amount of memory in bytes allocated for this palette.

• *getMaxValue*

public int getMaxValue()

– **Description**

Returns the maximum value allowed for a sample.

– **Returns** – the maximum sample value

• *getNumEntries*

public int getNumEntries()

– **Description**

Returns the number of entries in this palette.

– **Returns** – the number of entries in this palette

• *getSample*

public int getSample(int channelIndex, int entryIndex)

– **Description**

Returns one of the samples of this palette.

– **Parameters**

- * **entryIndex** – the index of the color to be addressed, must be from 0 to `getNumEntries() - 1`
 - * **channelIndex** – one of the three channels; must be (in 14.1.15, page 334), (in 14.1.15, page 334) or (in 14.1.15, page 335)
 - **Returns** – the requested sample

- *getSamples*
 public int[] **getSamples**(int **channelIndex**)
 - **Description**
Returns all samples of one channel as an int array.
 - **Parameters**
 - * **channelIndex** – index of the channel, one of the (in 14.1.15, page 334) constants
 - **Returns** – array with samples

- *isBlackAndWhite*
 public boolean **isBlackAndWhite**()
 - **Description**
Checks if all entries of this palette are either black or white. An entry is black if all three intensities (red, green and blue) are 0, it is white if they are all equal to (in 14.2.9, page 364). No particular order of entries (e.g. first color black, second white) is demanded and no specific number of entries (e.g. 2). This means that a palette is black and white if it contains ten entries that are all black.
 - **Returns** – if the palette contains only the colors black and white

- *isGray*
 public boolean **isGray**()
 - **Description**
Checks if this palette is gray, i.e., checks if all entries are gray. This is the case if for all entries red, green and blue have the same intensity.
 - **Returns** – if the palette contains only shades of gray

- *put*
 public void **put**(int **entryIndex**, int **red**, int **green**, int **blue**)

- *putSample*
 public void **putSample**(int **channelIndex**, int **entryIndex**, int **newValue**)
 - **Description**
Sets one sample of one color entry in the palette to a new value.
 - **Parameters**
 - * **channelIndex** –

Chapter 15

Package net.sourceforge.jiu.filters

Package Contents

Page

Classes

AreaFilterOperation	367
<i>Base class for operations that convert images to images and determine an output sample by doing calculations on the input sample at the same position plus some neighboring samples.</i>	
BorderSampleGenerator	371
<i>Abstract base class for classes that fill an <code>int</code> array with samples from a rectangular region of an image's channel by (1) copying <code>int</code> samples from an (in 14.1.7, page 322) object and by (2) generating samples that lie outside of the image.</i>	
ConvolutionKernelData	374
<i>This class encapsulates the information for a specific convolution kernel filter.</i>	
ConvolutionKernelFilter	377
<i>Applies a convolution kernel filter to an image.</i>	
MaximumFilter	381
<i>Filter operation that replaces each sample by the maximum value of itself and its neighbor samples.</i>	
MeanFilter	383
<i>Applies a mean filter that replaces each pixel by the mean of itself and its neighbors.</i>	
MedianFilter	385
<i>Applies a Median filter that replaces each pixel by the median of itself and its neighbors.</i>	
MinimumFilter	387
<i>Filter operation that replaces each sample by the minimum value of itself and its neighbors.</i>	
OilFilter	388
<i>Applies a filter that makes the image look like an oil painting.</i>	
UnsharpMaskKernel	390
<i>An unsharp mask kernel to be used with (in 15.1.4, page 377).</i>	

Various image filters that produce an output image from an input image, mostly reading a pixel and its neighbors in the input image to determine the pixel in the output image.

15.1 Classes

15.1.1 Class AreaFilterOperation

Base class for operations that convert images to images and determine an output sample by doing calculations on the input sample at the same position plus some neighboring samples. Override (in 15.1.1, page 368) and the operation will work.

Declaration

```
public abstract class AreaFilterOperation
extends net.sourceforge.jiu.ops.ImageToImageOperation (in 19.2.3, page 501)
```

All known subclasses

OilFilter (in 15.1.9, page 388), MinimumFilter (in 15.1.8, page 387), MedianFilter (in 15.1.7, page 385), MeanFilter (in 15.1.6, page 383), MaximumFilter (in 15.1.5, page 381)

Constructor summary

AreaFilterOperation()

Method summary

checkAreaHeight(int) Checks if the argument is a valid area height value.
checkAreaWidth(int) Checks if the argument is a valid area width value.
computeSample(int[], int) Determine the resulting sample for an array with the source sample and zero or more of its neighbors.
getAreaHeight() Returns the current area height.
getAreaWidth() Returns the current area width.
process()
setArea(int, int) Sets the area of the window to be used to determine each pixel's mean to the argument width and height.
setAreaHeight(int) Sets the height of the area of the window to be used to determine each pixel's mean to the argument value.
setAreaWidth(int) Sets the width of the area of the window to be used to determine each pixel's mean to the argument value.

Constructors

- *AreaFilterOperation*
public AreaFilterOperation()

Methods

- *checkAreaHeight*
public void checkAreaHeight(int height)

– **Description**

Checks if the argument is a valid area height value. The default implementation requires the argument to be odd and larger than zero. Override this method if your extension of AreaFilterOperation requires different heights.

– **Throws**

* `java.lang.IllegalArgumentException` – if the argument is not valid

• *checkAreaWidth*

`public void checkAreaWidth(int width)`

– **Description**

Checks if the argument is a valid area width value. The default implementation requires the argument to be odd and larger than zero. Override this method if your extension of AreaFilterOperation requires different widths.

– **Throws**

* `java.lang.IllegalArgumentException` – if the argument is not valid

• *computeSample*

`public abstract int computeSample(int[] samples, int numSamples)`

– **Description**

Determine the resulting sample for an array with the source sample and zero or more of its neighbors. This abstract method must be implemented by classes extending this operation. The array will hold `numSamples` samples, which will be stored starting at offset 0.

Normally, `numSamples` is equal to `(in 15.1.1, page 368)times (in 15.1.1, page 368)`. Near the border of the image you may get less samples. Example: the top left sample of an image has only three neighbors (east, south-east and south), so you will only get four samples (three neighbors and the sample itself).

– **Parameters**

* `samples` – the array holding the sample(s)
 * `numSamples` – number of samples in the array

– **Returns** – sample to be written to the output image

• *getAreaHeight*

`public int getAreaHeight()`

– **Description**

Returns the current area height.

– **Returns** – height of area window in pixels

– **See also**

* `AreaFilterOperation.setAreaHeight(int)` (in 15.1.1, page 369)

• *getAreaWidth*

`public int getAreaWidth()`

– **Description**

Returns the current area width.

– **Returns** – width of area window in pixels

– **See also**

* `AreaFilterOperation.setAreaWidth(int)` (in 15.1.1, page 369)

- *process*

public void process() throws
 net.sourceforge.jiu.ops.MissingParameterException,
 net.sourceforge.jiu.ops.OperationFailedException,
 net.sourceforge.jiu.ops.WrongParameterException

- **Description** copied from net.sourceforge.jiu.ops.Operation (in 19.2.5, page 508)

This method does the actual work of the operation. It must be called after all parameters have been given to the operation object.

- **Throws**

- * net.sourceforge.jiu.ops.WrongParameterException – if at least one of the input parameters was not initialized appropriately (values out of the valid interval, etc.)
 - * net.sourceforge.jiu.ops.MissingParameterException – if any mandatory parameter was not given to the operation
 - * net.sourceforge.jiu.ops.OperationFailedException –
-

- *setArea*

public void setArea(int width, int height)

- **Description**

Sets the area of the window to be used to determine each pixel's mean to the argument width and height.

- **Parameters**

- * **width** – width of window, must be 1 or larger
- * **height** – height of window, must be 1 or larger

- **See also**

- * AreaFilterOperation.setAreaHeight(int) (in 15.1.1, page 369)
 - * AreaFilterOperation.setAreaWidth(int) (in 15.1.1, page 369)
-

- *setAreaHeight*

public void setAreaHeight(int height)

- **Description**

Sets the height of the area of the window to be used to determine each pixel's mean to the argument value.

- **Parameters**

- * **height** – height of window, must be odd and 1 or larger

- **See also**

- * AreaFilterOperation.getAreaHeight() (in 15.1.1, page 368)
 - * AreaFilterOperation.setArea(int,int) (in 15.1.1, page 369)
 - * AreaFilterOperation.setAreaWidth(int) (in 15.1.1, page 369)
-

- *setAreaWidth*

public void setAreaWidth(int width)

- **Description**

Sets the width of the area of the window to be used to determine each pixel's mean to the argument value.

– **Parameters**

- * `width` – width of window, must be odd and 1 or larger

– **See also**

- * `AreaFilterOperation.getAreaWidth()` (in 15.1.1, page 368)
- * `AreaFilterOperation.setArea(int,int)` (in 15.1.1, page 369)
- * `AreaFilterOperation.setAreaHeight(int)` (in 15.1.1, page 369)

15.1.2 Class BorderSampleGenerator

Abstract base class for classes that fill an `int` array with samples from a rectangular region of an image's channel by (1) copying `int` samples from an (in 14.1.7, page 322) object and by (2) generating samples that lie outside of the image. To be used by (in 15.1.4, page 377) and other operations that require rectangular parts of an image that may not lie fully inside of the image.

Declaration

```
public abstract class BorderSampleGenerator
extends java.lang.Object
```

Constructor summary

BorderSampleGenerator(IntegerImage, int, int) Initialize width and height of the area to be covered in every call to (in 15.1.2, page 372), also provides the image to be used for data copying.

Method summary

fill(int, int, int[]) Fills the argument array with samples from the current channel of the image given to the constructor, generating samples that lie outside of the image.

getAreaHeight() Returns the number of rows from which data is copied or generated with every call to (in 15.1.2, page 372).

getAreaWidth() Returns the number of columns from which data is copied or generated with every call to (in 15.1.2, page 372).

getChannelIndex() Returns the index of the channel of the image from which data is copied.

getImage() Returns the image from which data is copied.

setChannelIndex(int) Sets the channel from which data is copied in (in 15.1.2, page 372).

Constructors

- *BorderSampleGenerator*

```
public BorderSampleGenerator( net.sourceforge.jiu.data.IntegerImage
integerImage, int areaWidth, int areaHeight )
```

– Description

Initialize width and height of the area to be covered in every call to (in 15.1.2, page 372), also provides the image to be used for data copying. The current channel is set to 0.

– Parameters

- * `integerImage` – the image from which samples will be copied
- * `areaWidth` – number of columns of the area to be covered in (in 15.1.2, page 372)
- * `areaHeight` – number of rows of the area to be covered in (in 15.1.2, page 372)

Methods

- *fill*

```
public abstract void fill( int x, int y, int[] samples )
```

- **Description**

Fills the argument array with samples from the current channel of the image given to the constructor, generating samples that lie outside of the image. The samples are copied (or generated) from the row *y* to row *y* + *areaHeight* - 1, and within each row from column *x* to *x* + *areaWidth* - 1.

The implementation of this method is left to the child classes. There are different ways to generate new samples, and each child class is supposed to implement another way. Obviously, the child classes also must copy samples from the image.

- **Parameters**

- * *x* – leftmost column to be copied or generated
- * *y* – top row to be copied or generated
- * *samples* – array to which samples will be written; must have at least (in 15.1.2, page 372) *times* (in 15.1.2, page 372) *elements*

- *getAreaHeight*

```
public int getAreaHeight( )
```

- **Description**

Returns the number of rows from which data is copied or generated with every call to (in 15.1.2, page 372).

- **Returns** – number or rows of a fill area

- *getAreaWidth*

```
public int getAreaWidth( )
```

- **Description**

Returns the number of columns from which data is copied or generated with every call to (in 15.1.2, page 372).

- **Returns** – number or columns of a fill area

- *getChannelIndex*

```
public int getChannelIndex( )
```

- **Description**

Returns the index of the channel of the image from which data is copied.

- **Returns** – number or rows

- **See also**

- * `BorderSampleGenerator.setChannelIndex(int)` (in 15.1.2, page 373)

- *getImage*

```
public net.sourceforge.jiu.data.IntegerImage getImage( )
```

- **Description**

Returns the image from which data is copied.

- **Returns** – image object

- *setChannelIndex*

```
public void setChannelIndex( int newChannelIndex )
```

- **Description**

Sets the channel from which data is copied in (in 15.1.2, page 372).

- **Returns** – channel index

- **See also**

- * `BorderSampleGenerator.getChannelIndex()` (in 15.1.2, page 372)

15.1.3 Class ConvolutionKernelData

This class encapsulates the information for a specific convolution kernel filter. An object of this class is used in combination with (in 15.1.4, page 377). Several kernel data objects are predefined in that class.

See also

- **ConvolutionKernelFilter** (in 15.1.4, page 377)

Declaration

```
public class ConvolutionKernelData
    extends java.lang.Object
```

All known subclasses

UnsharpMaskKernel (in 15.1.10, page 390)

Constructor summary

ConvolutionKernelData(String, int[], int, int, int, int) Creates a new kernel from the arguments.

Method summary

check() Checks if this kernel's data is valid and throws an `IllegalArgumentException` if anything is wrong.
getBias() Returns this kernel's bias value.
getData() Returns the kernel data.
getDiv() Returns this kernel's div value.
getHeight() Returns this kernel's height, an odd positive number.
getName() Returns this kernel's name.
getWidth() Returns this kernel's width, an odd positive number.
setBias(int) Set new bias value.
setData(int[]) Sets the data array to be used in this kernel.
setDiv(int)
setHeight(int)
setName(String)
setWidth(int)

Constructors

- *ConvolutionKernelData*
public ConvolutionKernelData(java.lang.String name, int[] data, int width, int height, int div, int bias)

– Description

Creates a new kernel from the arguments. Calls the various set methods to actually store these arguments.

Methods

- *check*
`public void check()`
 - **Description**
Checks if this kernel's data is valid and throws an `IllegalArgumentException` if anything is wrong. Otherwise, does nothing.

- *getBias*
`public int getBias()`
 - **Description**
Returns this kernel's bias value. See (in 15.1.4, page 377) for an explanation of this and other kernel properties.
 - **See also**
 - * `ConvolutionKernelData.setBias(int)` (in 15.1.3, page 376)

- *getData*
`public int[] getData()`
 - **Description**
Returns the kernel data. See (in 15.1.4, page 377) for an explanation of this and other kernel properties.
 - **See also**
 - * `ConvolutionKernelData.setData(int[])` (in 15.1.3, page 376)

- *getDiv*
`public int getDiv()`
 - **Description**
Returns this kernel's div value. Must not be 0. See (in 15.1.4, page 377) for an explanation of this and other kernel properties.
 - **See also**
 - * `ConvolutionKernelData.setDiv(int)` (in 15.1.3, page 376)

- *getHeight*
`public int getHeight()`
 - **Description**
Returns this kernel's height, an odd positive number. See (in 15.1.4, page 377) for an explanation of this and other kernel properties.

- *getName*
`public java.lang.String getName()`
 - **Description**
Returns this kernel's name.

- *getWidth*
`public int getWidth()`

– **Description**

Returns this kernel's width, an odd positive number. See (in 15.1.4, page 377) for an explanation of this and other kernel properties.

• *setBias*

`public void setBias(int newBias)`

– **Description**

Set new bias value. See (in 15.1.4, page 377) for an explanation of this and other kernel properties.

• *setData*

`public void setData(int[] newData)`

– **Description**

Sets the data array to be used in this kernel. Must have at least getWidth() times getHeight() elements - however, this constraint is not checked in this method (setting width and height may happen later). Call (in 15.1.3, page 375)

– **Parameters**

* `newData` –

• *setDiv*

`public void setDiv(int newDiv)`

• *setHeight*

`public void setHeight(int newHeight)`

• *setName*

`public void setName(java.lang.String newName)`

• *setWidth*

`public void setWidth(int newWidth)`

15.1.4 Class ConvolutionKernelFilter

Applies a convolution kernel filter to an image.

Supported image types

Only image types that store intensity samples are supported. Right now, this only includes (in 14.1.6, page 321) and (in 14.1.16, page 336).

Usage example

Standard approach (set up everything yourself):

```
ConvolutionKernelFilter filter = new ConvolutionKernelFilter();
filter.setKernel(ConvolutionKernelFilter.TYPE_SHARPEN);
filter.setInputImage(image);
filter.process();
PixelImage sharpenedImage = filter.getOutputImage(); Use static convenience method on
image img:
PixelImage filteredImage = ConvolutionKernelFilter.filter(img, ConvolutionKernelFilter.TYPE_BLUR);
```

Credits

The implementation of the filter was created by members of the Java newsgroup **de.comp.lang.java** (at <news://de.comp.lang.java>) and adapted to the JIU framework by Marco Schmidt. As it was done in a contest style where people improved other people's work, and even more people suggested ideas, tested results and discussed the contest it is (1) hard to tell who won the contest and (2) only fair to list all persons involved.

The resulting implementation is significantly faster than the **reference implementation** (at <http://groups.yahoo.com/group/dclj/files/CONTEST/Vorschlag/>). The contest was started by the posting [JPEC#3] Vorschlage to de.comp.lang.java by Marco Schmidt (2001-02-18) and was ended by the posting [JPEC#3] Ergebnisse (2001-03-07). A Usenet archive like **Google Groups** (at <http://groups.google.com>) should be able to provide the postings.

Declaration

```
public class ConvolutionKernelFilter
  extends net.sourceforge.jiu.ops.ImageToImageOperation (in 19.2.3, page 501)
```

Field summary

```
TYPE_BLUR
TYPE_EDGE_DETECTION
TYPE_EMBOSS
TYPE_HORIZONTAL_PREWITT
TYPE_HORIZONTAL_SOBEL
TYPE_LITHOGRAPH
TYPE_PSYCHEDELIC_DISTILLATION
TYPE_SHARPEN
TYPE_VERTICAL_PREWITT
TYPE_VERTICAL_SOBEL
```

Constructor summary

ConvolutionKernelFilter()

Method summary

filter(PixelImage, ConvolutionKernelData)

filter(PixelImage, int) Filters argument image with argument kernel type and returns output image.

process()

setKernel(ConvolutionKernelData) Sets kernel data to be used for filtering.

setKernel(int) Sets one of the predefined kernel types to be used for filtering.

setKernel(int[], int, int, int, int) Sets properties of the kernel to be used in this operation.

Fields

- public static final int **TYPE_BLUR**
- public static final int **TYPE_SHARPEN**
- public static final int **TYPE_EDGE_DETECTION**
- public static final int **TYPE_EMBOSS**
- public static final int **TYPE_PSYCHEDELIC_DISTILLATION**
- public static final int **TYPE_LITHOGRAPH**
- public static final int **TYPE_HORIZONTAL_SOBEL**
- public static final int **TYPE_VERTICAL_SOBEL**
- public static final int **TYPE_HORIZONTAL_PREWITT**
- public static final int **TYPE_VERTICAL_PREWITT**

Constructors

- *ConvolutionKernelFilter*
public **ConvolutionKernelFilter()**

Methods

- *filter*
public static net.sourceforge.jiu.data.PixelImage **filter(**
net.sourceforge.jiu.data.PixelImage input, ConvolutionKernelData data)
- *filter*
public static net.sourceforge.jiu.data.PixelImage **filter(**
net.sourceforge.jiu.data.PixelImage input, int kernelType)

– **Description**

Filters argument image with argument kernel type and returns output image. Static convenience method to do filtering with one line of code:

```
PixelImage blurredImage = ConvolutionKernelFilter.filter(in, ConvolutionKernelFilter.TYPE F
```

• *process*

```
public void process( ) throws
net.sourceforge.jiu.ops.MissingParameterException,
net.sourceforge.jiu.ops.OperationFailedException,
net.sourceforge.jiu.ops.WrongParameterException
```

– **Description copied from net.sourceforge.jiu.ops.Operation (in 19.2.5, page 508)**

This method does the actual work of the operation. It must be called after all parameters have been given to the operation object.

– **Throws**

- * net.sourceforge.jiu.ops.WrongParameterException – if at least one of the input parameters was not initialized appropriately (values out of the valid interval, etc.)
 - * net.sourceforge.jiu.ops.MissingParameterException – if any mandatory parameter was not given to the operation
 - * net.sourceforge.jiu.ops.OperationFailedException –
-

• *setKernel*

```
public void setKernel( ConvolutionKernelData ckd )
```

– **Description**

Sets kernel data to be used for filtering.

– **Parameters**

- * ckd – all information necessary for filtering
-

• *setKernel*

```
public void setKernel( int type )
```

– **Description**

Sets one of the predefined kernel types to be used for filtering.

– **Parameters**

- * type – one of the TYPE_xyz constants of this class

– **Throws**

- * java.lang.IllegalArgumentException – if the argument is not a valid TYPE_xyz constant
-

• *setKernel*

```
public void setKernel( int[] data, int width, int height, int div, int bias
)
```

– **Description**

Sets properties of the kernel to be used in this operation.

– **Parameters**

- * data – the kernel coefficients; this one-dimensional array stores them in order top-to-bottom, left-to-right; the length of this array must be at least width times height

- * **width** – the width of the kernel; must not be even
- * **height** – the height of the kernel; must not be even
- * **div** – the result is divided by this value after the addition of value (so this value must not be zero)
- * **bias** – this value is added to the result before the division

15.1.5 Class MaximumFilter

Filter operation that replaces each sample by the maximum value of itself and its neighbor samples.

Note that this is not the maximum operation that takes two input images and, for each position, takes the maximum sample value and writes it to output.

Usage example

```
MaximumFilter filter = new MaximumFilter();
filter.setArea(7, 5);
filter.setInputImage(image);
filter.process();
PixelImage filteredImage = filter.getOutputImage();
```

See also

- `MinimumFilter` (in 15.1.8, page 387)

Declaration

```
public class MaximumFilter
  extends net.sourceforge.jiu.filters.AreaFilterOperation (in 15.1.1, page 367)
```

Constructor summary

MaximumFilter()

Method summary

computeSample(int[], int)

Constructors

- *MaximumFilter*
public **MaximumFilter**()

Methods

- *computeSample*
public abstract int **computeSample**(int[] samples, int numSamples)
 - **Description copied from AreaFilterOperation (in 15.1.1, page 367)**
Determine the resulting sample for an array with the source sample and zero or more of its neighbors. This abstract method must be implemented by classes extending this operation. The array will hold numSamples samples, which will be stored starting at offset 0.

Normally, `numSamples` is equal to `(in 15.1.1, page 368)times (in 15.1.1, page 368)`. Near the border of the image you may get less samples. Example: the top left sample of an image has only three neighbors (east, south-east and south), so you will only get four samples (three neighbors and the sample itself).

– **Parameters**

- * `samples` – the array holding the sample(s)
- * `numSamples` – number of samples in the array

– **Returns** – sample to be written to the output image

15.1.6 Class MeanFilter

Applies a mean filter that replaces each pixel by the mean of itself and its neighbors. The number of neighbors can be defined by the `setArea` methods. This filter only works with intensity-based image types. More precisely, only (in 14.1.6, page 321) and (in 14.1.16, page 336) will work.

Usage example

```
PixelImage image = ...; // some GrayIntegerImage or RGBIntegerImage
MeanFilter filter = new MeanFilter();
filter.setArea(5, 5);
filter.setInputImage(image);
filter.process();
PixelImage filteredImage = filter.getOutputImage();
```

Declaration

```
public class MeanFilter
  extends net.sourceforge.jiu.filters.AreaFilterOperation (in 15.1.1, page 367)
```

Constructor summary

MeanFilter()

Method summary

computeSample(int[], int)

Constructors

- *MeanFilter*
public **MeanFilter**()

Methods

- *computeSample*
public abstract int **computeSample**(int[] samples, int numSamples)
 - **Description copied from AreaFilterOperation (in 15.1.1, page 367)**
Determine the resulting sample for an array with the source sample and zero or more of its neighbors. This abstract method must be implemented by classes extending this operation. The array will hold `numSamples` samples, which will be stored starting at offset 0.
Normally, `numSamples` is equal to (in 15.1.1, page 368) `times` (in 15.1.1, page 368). Near the border of the image you may get less samples. Example: the top left sample of an image has only three neighbors (east, south-east and south), so you will only get four samples (three neighbors and the sample itself).
 - **Parameters**

- * **samples** – the array holding the sample(s)
- * **numSamples** – number of samples in the array
- **Returns** – sample to be written to the output image

15.1.7 Class MedianFilter

Applies a Median filter that replaces each pixel by the median of itself and its neighbors. The number of neighbors can be defined with the `setArea` methods.

Can be used as despeckle filter, but the image will lose sharpness. The larger the area becomes, the less noise and the less sharpness will remain, and the longer it will take.

Uses (in 20.2.4, page 526) to do the search for the median value.

Usage example

```
PixelImage image = ...; // some GrayIntegerImage or RGBIntegerImage
MedianFilter filter = new MedianFilter();
filter.setArea(5, 5);
filter.setInputImage(image);
filter.process();
PixelImage filteredImage = filter.getOutputImage();
```

Declaration

```
public class MedianFilter
extends net.sourceforge.jiu.filters.AreaFilterOperation (in 15.1.1, page 367)
```

Constructor summary

MedianFilter()

Method summary

computeSample(int[], int)

Constructors

- *MedianFilter*
public **MedianFilter**()

Methods

- *computeSample*
public abstract int **computeSample**(int[] samples, int numSamples)
 - **Description copied from AreaFilterOperation (in 15.1.1, page 367)**
Determine the resulting sample for an array with the source sample and zero or more of its neighbors. This abstract method must be implemented by classes extending this operation. The array will hold `numSamples` samples, which will be stored starting at offset 0.
Normally, `numSamples` is equal to (in 15.1.1, page 368)times (in 15.1.1, page 368). Near the border of the image you may get less samples. Example: the top left sample of an image has only three neighbors (east, south-east and south), so you will only get four samples (three neighbors and the sample itself).

– **Parameters**

- * **samples** – the array holding the sample(s)
- * **numSamples** – number of samples in the array

– **Returns** – sample to be written to the output image

15.1.8 Class MinimumFilter

Filter operation that replaces each sample by the minimum value of itself and its neighbors. See (in 15.1.5, page 381) for a usage example.

Declaration

```
public class MinimumFilter
  extends net.sourceforge.jiu.filters.AreaFilterOperation (in 15.1.1, page 367)
```

Constructor summary

MinimumFilter()

Method summary

computeSample(int[], int)

Constructors

- *MinimumFilter*
public **MinimumFilter**()

Methods

- *computeSample*
public abstract int **computeSample**(int[] samples, int numSamples)
 - **Description copied from AreaFilterOperation (in 15.1.1, page 367)**
Determine the resulting sample for an array with the source sample and zero or more of its neighbors. This abstract method must be implemented by classes extending this operation. The array will hold numSamples samples, which will be stored starting at offset 0.
Normally, numSamples is equal to (in 15.1.1, page 368)times (in 15.1.1, page 368). Near the border of the image you may get less samples. Example: the top left sample of an image has only three neighbors (east, south-east and south), so you will only get four samples (three neighbors and the sample itself).
 - **Parameters**
 - * **samples** – the array holding the sample(s)
 - * **numSamples** – number of samples in the array
 - **Returns** – sample to be written to the output image

15.1.9 Class OilFilter

Applies a filter that makes the image look like an oil painting. This is accomplished by creating a histogram of the neighboring samples for each input sample and storing the value that occurs most often in the output image. If two or more samples occur an equal number of times, the lowest sample value is picked.

Supported image types

Can process both (in 14.1.6, page 321) and (in 14.1.16, page 336). Note that this operation becomes very slow with 16 bits per sample because a lot of runs over a 65536 element array are necessary.

Usage example

```
PixelImage image = ...; // some GrayIntegerImage or RGBIntegerImage
OilFilter filter = new OilFilter();
filter.setArea(5, 5);
filter.setInputImage(image);
filter.process();
PixelImage filteredImage = filter.getOutputImage();
```

Credits

Idea taken from the **Oil class** (at http://www.acme.com/java/software/Acme.JPM.Filters.Oil.html#_top_) of Jef Poskanzer's **ACME package** (at <http://www.acme.com/java/software/>).

Declaration

```
public class OilFilter
  extends net.sourceforge.jiu.filters.AreaFilterOperation (in 15.1.1, page 367)
```

Constructor summary

OilFilter()

Method summary

computeSample(int[], int)
process()

Constructors

- *OilFilter*
public OilFilter()

Methods

- *computeSample*

`public abstract int computeSample(int[] samples, int numSamples)`

- **Description copied from AreaFilterOperation (in 15.1.1, page 367)**

Determine the resulting sample for an array with the source sample and zero or more of its neighbors. This abstract method must be implemented by classes extending this operation. The array will hold `numSamples` samples, which will be stored starting at offset 0.

Normally, `numSamples` is equal to (in 15.1.1, page 368) `times` (in 15.1.1, page 368). Near the border of the image you may get less samples. Example: the top left sample of an image has only three neighbors (east, south-east and south), so you will only get four samples (three neighbors and the sample itself).

- **Parameters**

- * `samples` – the array holding the sample(s)
- * `numSamples` – number of samples in the array

- **Returns** – sample to be written to the output image

- *process*

`public void process() throws`
`net.sourceforge.jiu.ops.MissingParameterException,`
`net.sourceforge.jiu.ops.OperationFailedException,`
`net.sourceforge.jiu.ops.WrongParameterException`

- **Description copied from net.sourceforge.jiu.ops.Operation (in 19.2.5, page 508)**

This method does the actual work of the operation. It must be called after all parameters have been given to the operation object.

- **Throws**

- * `net.sourceforge.jiu.ops.WrongParameterException` – if at least one of the input parameters was not initialized appropriately (values out of the valid interval, etc.)
- * `net.sourceforge.jiu.ops.MissingParameterException` – if any mandatory parameter was not given to the operation
- * `net.sourceforge.jiu.ops.OperationFailedException` –

15.1.10 Class UnsharpMaskKernel

An unsharp mask kernel to be used with (in 15.1.4, page 377).

Declaration

```
public class UnsharpMaskKernel
  extends net.sourceforge.jiu.filters.ConvolutionKernelData (in 15.1.3, page 374)
```

Constructor summary

UnsharpMaskKernel(int) Creates a new unsharp mask kernel.

Constructors

- *UnsharpMaskKernel*
public **UnsharpMaskKernel**(int level)
 - **Description**
Creates a new unsharp mask kernel.
 - **Parameters**
 - * level – adjusts the amount of 'unsharpness', must be from 1 to 50

Chapter 16

Package net.sourceforge.jiu.geometry

Package Contents

Page

Classes

BellFilter	393
<i>A Bell resample filter.</i>	
BoxFilter	395
<i>A box filter (also known as nearest neighbor).</i>	
BSplineFilter	396
<i>A B-spline resample filter.</i>	
Crop	397
<i>Copies a rectangular area of one image to another image that is exactly as large as that rectangular area.</i>	
Flip	399
<i>Flips images (top row becomes bottom row and vice versa, and so on).</i>	
HermiteFilter	400
<i>A Hermite resampling filter.</i>	
Lanczos3Filter	401
<i>The Lanczos 3 resample filter.</i>	
Mirror	402
<i>Mirrors images (leftmost column becomes rightmost column and vice versa, and so on).</i>	
MitchellFilter	404
<i>The Mitchell resample filter.</i>	
Resample	405
<i>Resizes grayscale and truecolor images using filters.</i>	
ResampleFilter	409
<i>Abstract base class for filters to be used with the (in 16.1.10, page 405)operation.</i>	
Rotate180	411
<i>Rotates images by 180 degrees.</i>	
Rotate90Left	413
<i>Rotates images by 90 degrees counter-clockwise (to the left).</i>	
Rotate90Right	414
<i>Rotates images by 90 degrees clockwise (to the right).</i>	
ScaleReplication	416
<i>Changes the pixel resolution of an image by replicating (or dropping) pixels.</i>	
Shear	418

<i>Shears an image by a given angle.</i>	
TriangleFilter	420
<i>A triangle filter (also known as linear or bilinear filter).</i>	

Operations to change the geometry of an image, mirroring it horizontally and vertically, shearing, scaling and rotating it.

16.1 Classes

16.1.1 *Class* BellFilter

A Bell resample filter.

See also

- `Resample` (in 16.1.10, page 405)
- `ResampleFilter` (in 16.1.11, page 409)

Declaration

```
public class BellFilter
extends net.sourceforge.jiu.geometry.ResampleFilter (in 16.1.11, page 409)
```

Constructor summary

`BellFilter()`

Method summary

`apply(float)`
`getName()`
`getRecommendedSamplingRadius()`

Constructors

- *BellFilter*
`public BellFilter()`

Methods

- *apply*
`public abstract float apply(float value)`
 - **Description copied from ResampleFilter (in 16.1.11, page 409)**
Returns the weight of the sample at the distance given by the argument value.
- *getName*
`public abstract java.lang.String getName()`
 - **Description copied from ResampleFilter (in 16.1.11, page 409)**
Return the name of this filter. Should avoid natural language words if possible.
 - **Returns** – String with filter name
- *getRecommendedSamplingRadius*
`public abstract float getRecommendedSamplingRadius()`

- **Description copied from ResampleFilter** (in 16.1.11, page 409)
Returns a recommendation for the sampling radius to be used with this filter. This recommendation value will be the default value for the sampling radius of objects of this class. You can modify it with a call to (in 16.1.11, page 410).
- **Returns** – the recommended sampling radius to be used with this filter

16.1.2 Class BoxFilter

A box filter (also known as nearest neighbor).

See also

- `Resample` (in 16.1.10, page 405)
- `ResampleFilter` (in 16.1.11, page 409)

Declaration

```
public class BoxFilter
  extends net.sourceforge.jiu.geometry.ResampleFilter (in 16.1.11, page 409)
```

Constructor summary

`BoxFilter()`

Method summary

`apply(float)`
`getName()`
`getRecommendedSamplingRadius()`

Constructors

- *BoxFilter*
 public **BoxFilter**()

Methods

- *apply*
 public abstract float **apply**(float value)
 – **Description copied from ResampleFilter (in 16.1.11, page 409)**
 Returns the weight of the sample at the distance given by the argument value.
- *getName*
 public abstract java.lang.String **getName**()
 – **Description copied from ResampleFilter (in 16.1.11, page 409)**
 Return the name of this filter. Should avoid natural language words if possible.
 – **Returns** – String with filter name
- *getRecommendedSamplingRadius*
 public abstract float **getRecommendedSamplingRadius**()
 – **Description copied from ResampleFilter (in 16.1.11, page 409)**
 Returns a recommendation for the sampling radius to be used with this filter. This recommendation value will be the default value for the sampling radius of objects of this class. You can modify it with a call to (in 16.1.11, page 410).
 – **Returns** – the recommended sampling radius to be used with this filter

16.1.3 Class BSplineFilter

A B-spline resample filter.

See also

- `Resample` (in 16.1.10, page 405)
- `ResampleFilter` (in 16.1.11, page 409)

Declaration

```
public class BSplineFilter
extends net.sourceforge.jiu.geometry.ResampleFilter (in 16.1.11, page 409)
```

Constructor summary

`BSplineFilter()`

Method summary

`apply(float)`
`getName()`
`getRecommendedSamplingRadius()`

Constructors

- *BSplineFilter*
`public BSplineFilter()`

Methods

- *apply*
`public abstract float apply(float value)`
 - **Description copied from ResampleFilter** (in 16.1.11, page 409)
Returns the weight of the sample at the distance given by the argument value.
- *getName*
`public abstract java.lang.String getName()`
 - **Description copied from ResampleFilter** (in 16.1.11, page 409)
Return the name of this filter. Should avoid natural language words if possible.
 - **Returns** – String with filter name
- *getRecommendedSamplingRadius*
`public abstract float getRecommendedSamplingRadius()`
 - **Description copied from ResampleFilter** (in 16.1.11, page 409)
Returns a recommendation for the sampling radius to be used with this filter. This recommendation value will be the default value for the sampling radius of objects of this class. You can modify it with a call to (in 16.1.11, page 410).
 - **Returns** – the recommended sampling radius to be used with this filter

16.1.4 Class Crop

Copies a rectangular area of one image to another image that is exactly as large as that rectangular area. Works with all image data classes implementing (in 14.1.7, page 322). Make sure to use zero-based parameters when defining the bounds with (in 16.1.4, page 398)!

Usage example

In this example we assume that the input image is larger than 20 pixels in both directions. Ten pixels will be removed from any of its four borders.

```
PixelImage image = ...; // something implementing IntegerImage
Crop crop = new Crop();
crop.setInputImage(image);
crop.setBounds(10, 10, image.getWidth() - 9, image.getHeight() - 9);
crop.process();
PixelImage croppedImage = crop.getOutputImage();
```

Declaration

```
public class Crop
extends net.sourceforge.jiu.ops.ImageToImageOperation (in 19.2.3, page 501)
```

Constructor summary

Crop()

Method summary

process()

setBounds(int, int, int, int) Specify the rectangular section of the original image that is to be copied to the output image by this operation.

Constructors

- *Crop*
`public Crop()`

Methods

- *process*
`public void process()` throws
`net.sourceforge.jiu.ops.MissingParameterException`,
`net.sourceforge.jiu.ops.OperationFailedException`,
`net.sourceforge.jiu.ops.WrongParameterException`
 - **Description copied from `net.sourceforge.jiu.ops.Operation` (in 19.2.5, page 508)**
This method does the actual work of the operation. It must be called after all parameters have been given to the operation object.

– **Throws**

- * `net.sourceforge.jiu.ops.WrongParameterException` – if at least one of the input parameters was not initialized appropriately (values out of the valid interval, etc.)
- * `net.sourceforge.jiu.ops.MissingParameterException` – if any mandatory parameter was not given to the operation
- * `net.sourceforge.jiu.ops.OperationFailedException` –

- *setBounds*

```
public void setBounds( int x1, int y1, int x2, int y2 ) throws
java.lang.IllegalArgumentException
```

– **Description**

Specify the rectangular section of the original image that is to be copied to the output image by this operation. Note that the arguments are not checked directly against any input image that may have been provided to this Crop object, that checking is done later in (in 16.1.4, page 397). If any of the arguments provided here are outside of the input image's resolution (e.g. `x1 == 100` although the input image's width is only 60), a (in 19.3.3, page 514) will be thrown from within (in 16.1.4, page 397).

Note that the arguments to this method are zero-based, so the first column and row are 0, the second 1, the third 2, and so on. If you have a image that is 200 pixels wide and 100 pixels high, values from 0 to 199 are valid for the x arguments, and values from 0 to 99 are valid for the vertical direction.

– **Parameters**

- * `x1` – horizontal position of upper left corner of the rectangle
- * `y1` – vertical position of upper left corner of the rectangle
- * `x2` – horizontal position of lower right corner of the rectangle
- * `y2` – vertical position of lower right corner of the rectangle

– **Throws**

- * `java.lang.IllegalArgumentException` – if any of the arguments is negative or `x1` larger than `x2` or `y1` larger than `y2`

16.1.5 Class Flip

Flips images (top row becomes bottom row and vice versa, and so on).

Supported image types: (in 14.1.7, page 322).

Usage example:

```
Flip flip = new Flip();
flip.setInputImage(image); // image is some IntegerImage object
flip.process();
PixelImage flippedImage = flip.getOutputImage();
```

Declaration

```
public class Flip
extends net.sourceforge.jiu.ops.ImageToImageOperation (in 19.2.3, page 501)
```

Constructor summary

Flip()

Method summary

process()

Constructors

- *Flip*
public **Flip**()

Methods

- *process*
public void **process**() throws
net.sourceforge.jiu.ops.MissingParameterException,
net.sourceforge.jiu.ops.OperationFailedException,
net.sourceforge.jiu.ops.WrongParameterException
 - **Description copied from net.sourceforge.jiu.ops.Operation** (in 19.2.5, page 508)
This method does the actual work of the operation. It must be called after all parameters have been given to the operation object.
 - **Throws**
 - * net.sourceforge.jiu.ops.WrongParameterException – if at least one of the input parameters was not initialized appropriately (values out of the valid interval, etc.)
 - * net.sourceforge.jiu.ops.MissingParameterException – if any mandatory parameter was not given to the operation
 - * net.sourceforge.jiu.ops.OperationFailedException –

16.1.6 Class HermiteFilter

A Hermite resampling filter.

See also

- `Resample` (in 16.1.10, page 405)
- `ResampleFilter` (in 16.1.11, page 409)

Declaration

```
public class HermiteFilter
extends net.sourceforge.jiu.geometry.ResampleFilter (in 16.1.11, page 409)
```

Constructor summary

`HermiteFilter()`

Method summary

`apply(float)`
`getName()`
`getRecommendedSamplingRadius()`

Constructors

- *HermiteFilter*
`public HermiteFilter()`

Methods

- *apply*
`public abstract float apply(float value)`
 - **Description copied from ResampleFilter (in 16.1.11, page 409)**
Returns the weight of the sample at the distance given by the argument value.
- *getName*
`public abstract java.lang.String getName()`
 - **Description copied from ResampleFilter (in 16.1.11, page 409)**
Return the name of this filter. Should avoid natural language words if possible.
 - **Returns** – String with filter name
- *getRecommendedSamplingRadius*
`public abstract float getRecommendedSamplingRadius()`
 - **Description copied from ResampleFilter (in 16.1.11, page 409)**
Returns a recommendation for the sampling radius to be used with this filter. This recommendation value will be the default value for the sampling radius of objects of this class. You can modify it with a call to (in 16.1.11, page 410).
 - **Returns** – the recommended sampling radius to be used with this filter

16.1.7 Class Lanczos3Filter

The Lanczos 3 resample filter.

See also

- `Resample` (in 16.1.10, page 405)
- `ResampleFilter` (in 16.1.11, page 409)

Declaration

```
public class Lanczos3Filter
  extends net.sourceforge.jiu.geometry.ResampleFilter (in 16.1.11, page 409)
```

Constructor summary

`Lanczos3Filter()`

Method summary

`apply(float)`
`getName()`
`getRecommendedSamplingRadius()`

Constructors

- *Lanczos3Filter*
 public **Lanczos3Filter**()

Methods

- *apply*
 public abstract float **apply**(float value)
 – **Description copied from ResampleFilter** (in 16.1.11, page 409)
 Returns the weight of the sample at the distance given by the argument value.
- *getName*
 public abstract java.lang.String **getName**()
 – **Description copied from ResampleFilter** (in 16.1.11, page 409)
 Return the name of this filter. Should avoid natural language words if possible.
 – **Returns** – String with filter name
- *getRecommendedSamplingRadius*
 public abstract float **getRecommendedSamplingRadius**()
 – **Description copied from ResampleFilter** (in 16.1.11, page 409)
 Returns a recommendation for the sampling radius to be used with this filter. This recommendation value will be the default value for the sampling radius of objects of this class. You can modify it with a call to (in 16.1.11, page 410).
 – **Returns** – the recommended sampling radius to be used with this filter

16.1.8 Class Mirror

Mirrors images (leftmost column becomes rightmost column and vice versa, and so on).
Supported image types: (in 14.1.7, page 322).

Usage example

```
PixelImage image = ...; // something implementing IntegerImage
Mirror mirror = new Mirror();
mirror.setInputImage(image);
mirror.process();
PixelImage mirroredImage = mirror.getOutputStream();
```

Declaration

```
public class Mirror
  extends net.sourceforge.jiu.ops.ImageToImageOperation (in 19.2.3, page 501)
```

Constructor summary

Mirror()

Method summary

process()

Constructors

- *Mirror*
public **Mirror**()

Methods

- *process*
public void **process**() throws
net.sourceforge.jiu.ops.MissingParameterException,
net.sourceforge.jiu.ops.OperationFailedException,
net.sourceforge.jiu.ops.WrongParameterException
 - **Description copied from net.sourceforge.jiu.ops.Operation** (in 19.2.5, page 508)
This method does the actual work of the operation. It must be called after all parameters have been given to the operation object.
 - **Throws**
 - * net.sourceforge.jiu.ops.WrongParameterException – if at least one of the input parameters was not initialized appropriately (values out of the valid interval, etc.)

- * `net.sourceforge.jiu.ops.MissingParameterException` – if any mandatory parameter was not given to the operation
- * `net.sourceforge.jiu.ops.OperationFailedException` –

16.1.9 Class MitchellFilter

The Mitchell resample filter.

See also

- `Resample` (in 16.1.10, page 405)
- `ResampleFilter` (in 16.1.11, page 409)

Declaration

```
public class MitchellFilter
extends net.sourceforge.jiu.geometry.ResampleFilter (in 16.1.11, page 409)
```

Constructor summary

`MitchellFilter()`

Method summary

`apply(float)`
`getName()`
`getRecommendedSamplingRadius()`

Constructors

- *MitchellFilter*
`public MitchellFilter()`

Methods

- *apply*
`public abstract float apply(float value)`
 - **Description copied from ResampleFilter (in 16.1.11, page 409)**
Returns the weight of the sample at the distance given by the argument value.
- *getName*
`public abstract java.lang.String getName()`
 - **Description copied from ResampleFilter (in 16.1.11, page 409)**
Return the name of this filter. Should avoid natural language words if possible.
 - **Returns** – String with filter name
- *getRecommendedSamplingRadius*
`public abstract float getRecommendedSamplingRadius()`
 - **Description copied from ResampleFilter (in 16.1.11, page 409)**
Returns a recommendation for the sampling radius to be used with this filter. This recommendation value will be the default value for the sampling radius of objects of this class. You can modify it with a call to (in 16.1.11, page 410).
 - **Returns** – the recommended sampling radius to be used with this filter

16.1.10 Class Resample

Resizes grayscale and truecolor images using filters. For other image types (including paletted or bilevel images), you might want to use the `ImageToImageOperation` (in 16.1.15, page 416) class or convert the images to grayscale (or RGB truecolor) first and then use this class. Several algorithms for resampling are implemented, they differ in resulting image quality and computational complexity.

Usage example

This will scale image to 150 percent of its original size in both directions, using the Lanczos3 filter type:

```
Resample resample = new Resample();
resample.setInputImage(image);
resample.setSize(image.getWidth() * 3 / 2, image.getHeight() * 3 / 2);
resample.setFilter(Resample.FILTER_TYPE_LANCZOS3);
resample.process();
PixelImage scaledImage = resample.getOutputImage();
```

Known problems

- Scaling down certain images (with stripe or checkers patterns in them) will lead to moire effects in the resulting image. These effects can be somewhat reduced by scaling down in several step (e.g. first from 1600 x 1200 to 800 x 600, then from 800 x 600 to 400 x 300, and so on).
- Scaling down with filter type `Resample.FILTER_TYPE_LANCZOS3` (in 16.1.10, page 406) can lead to errors in the scaled image. No fix known yet. Workaround: Use the class `ImageToImageOperation` (in 16.1.15, page 416).

Origin

This code is a port of Anders Melander's Object Pascal (Delphi) unit `resample.pas` to Java. The Delphi code is an adaptation (with some improvements) of Dale Schumacher's `fzoom` C code. Check out the homepage for the Delphi resample code, a demo application to compare the different filtering algorithms is also provided:

<http://www.melander.dk/delphi/resampler/index.html> (at

<http://www.melander.dk/delphi/resampler/index.html>). You will also find the original C code there.

Theory

The theoretical background for all implementations is Dale Schumacher's article General Filtered Image Rescaling in Graphics Gems III, editor David Kirk, Academic Press, pages 8-16, 1994. The Graphics Gems Repository can be found at <http://www.acm.org/tog/GraphicsGems/> (at <http://www.acm.org/tog/GraphicsGems/>). It also includes information on the books and how to order them.

Declaration

```
public class Resample
  extends net.sourceforge.jiu.ops.ImageToImageOperation (in 19.2.3, page 501)
```

Field summary

FILTER_TYPE_B_SPLINE Constant for the B-Spline filter.
FILTER_TYPE_BELL Constant for the Bell filter.
FILTER_TYPE_BOX Constant for the Box filter (also known as Nearest Neighbor filter).
FILTER_TYPE_HERMITE Constant for the Hermite filter.
FILTER_TYPE_LANCZOS3 Constant for the Lanczos3 filter.
FILTER_TYPE_MITCHELL Constant for the Mitchell filter.
FILTER_TYPE_TRIANGLE Constant for the Triangle filter (also known as Linear filter or Bilinear filter).

Constructor summary

Resample()

Method summary

getFilter() Returns the filter to be used in this operation.
getFilterNames() Returns the names of all predefined filters.
getNumFilters() Returns the number of predefined filters.
process()
setFilter(int) Sets a new filter type, using the default sampling radius of that filter.
setFilter(int, float) Sets a new filter type with a user-defined sampling radius.
setFilter(ResampleFilter) Set a new filter object to be used with this operation.
setSize(int, int) Set the pixel resolution of the output image.

Fields

- public static final int **FILTER_TYPE_BOX**
 - Constant for the Box filter (also known as Nearest Neighbor filter).
- public static final int **FILTER_TYPE_TRIANGLE**
 - Constant for the Triangle filter (also known as Linear filter or Bilinear filter).
- public static final int **FILTER_TYPE_HERMITE**
 - Constant for the Hermite filter.
- public static final int **FILTER_TYPE_BELL**
 - Constant for the Bell filter.
- public static final int **FILTER_TYPE_B_SPLINE**
 - Constant for the B-Spline filter.
- public static final int **FILTER_TYPE_LANCZOS3**
 - Constant for the Lanczos3 filter.
- public static final int **FILTER_TYPE_MITCHELL**
 - Constant for the Mitchell filter.

Constructors

- *Resample*
`public Resample()`

Methods

- *getFilter*
`public ResampleFilter getFilter()`
 - **Description**
Returns the filter to be used in this operation.
 - **Returns** – ResampleFilter object or null if none was defined yet

- *getFilterNames*
`public static java.lang.String[] getFilterNames()`
 - **Description**
Returns the names of all predefined filters. Each FILTER_TYPE_xyz constant can be used as an index into the array that is returned. Names are retrieved by creating an object of each predefined filter class and calling its getName method.
 - **Returns** – String array with filter names

- *getNumFilters*
`public static int getNumFilters()`
 - **Description**
Returns the number of predefined filters.
 - **Returns** – number of filters

- *process*
`public void process() throws
net.sourceforge.jiu.ops.MissingParameterException,
net.sourceforge.jiu.ops.OperationFailedException,
net.sourceforge.jiu.ops.WrongParameterException`
 - **Description** copied from net.sourceforge.jiu.ops.Operation (in 19.2.5, page 508)
This method does the actual work of the operation. It must be called after all parameters have been given to the operation object.
 - **Throws**
 - * net.sourceforge.jiu.ops.WrongParameterException – if at least one of the input parameters was not initialized appropriately (values out of the valid interval, etc.)
 - * net.sourceforge.jiu.ops.MissingParameterException – if any mandatory parameter was not given to the operation
 - * net.sourceforge.jiu.ops.OperationFailedException –

- *setFilter*
`public void setFilter(int filterType)`

- **Description**

Sets a new filter type, using the default sampling radius of that filter.

- **Parameters**

- * **filterType** – the new filter type, one of the FILTER_TYPE_xyz constants of this class

- *setFilter*

public void setFilter(int filterType, float samplingRadius)

- **Description**

Sets a new filter type with a user-defined sampling radius.

- **Parameters**

- * **filterType** – the new filter type, one of the FILTER_TYPE_xyz constants of this class
- * **samplingRadius** – the sampling radius to be used with that filter, must be larger than 0.0f

- *setFilter*

public void setFilter(ResampleFilter newFilter)

- **Description**

Set a new filter object to be used with this operation.

- **Parameters**

- * **newFilter** – a resample filter to be used for scaling

- *setSize*

public void setSize(int width, int height)

- **Description**

Set the pixel resolution of the output image.

- **Parameters**

- * **width** – the horizontal resolution of the output image
- * **height** – the vertical resolution of the output image

16.1.11 Class ResampleFilter

Abstract base class for filters to be used with the (in 16.1.10, page 405) operation.

Declaration

```
public abstract class ResampleFilter
extends java.lang.Object
```

All known subclasses

TriangleFilter (in 16.1.17, page 420), MitchellFilter (in 16.1.9, page 404), Lanczos3Filter (in 16.1.7, page 401), HermiteFilter (in 16.1.6, page 400), BSplineFilter (in 16.1.3, page 396), BoxFilter (in 16.1.2, page 395), BellFilter (in 16.1.1, page 393)

Constructor summary

ResampleFilter() This empty constructor sets the sampling radius to the recommended sampling radius as provided by (in 16.1.11, page 410).

Method summary

apply(float) Returns the weight of the sample at the distance given by the argument value.

getName() Return the name of this filter.

getRecommendedSamplingRadius() Returns a recommendation for the sampling radius to be used with this filter.

getSamplingRadius() Returns the sampling radius of this object.

setSamplingRadius(float) Sets the sampling radius to a new value.

Constructors

- *ResampleFilter*

```
public ResampleFilter( )
```

– Description

This empty constructor sets the sampling radius to the recommended sampling radius as provided by (in 16.1.11, page 410).

Methods

- *apply*

```
public abstract float apply( float value )
```

– Description

Returns the weight of the sample at the distance given by the argument value.

- *getName*

```
public abstract java.lang.String getName( )
```

- **Description**

Return the name of this filter. Should avoid natural language words if possible.

- **Returns** – String with filter name

- *getRecommendedSamplingRadius*

public abstract float **getRecommendedSamplingRadius**()

- **Description**

Returns a recommendation for the sampling radius to be used with this filter. This recommendation value will be the default value for the sampling radius of objects of this class. You can modify it with a call to (in 16.1.11, page 410).

- **Returns** – the recommended sampling radius to be used with this filter

- *getSamplingRadius*

public float **getSamplingRadius**()

- **Description**

Returns the sampling radius of this object.

- **See also**

- * `ResampleFilter.getRecommendedSamplingRadius()` (in 16.1.11, page 410)

- * `ResampleFilter.setSamplingRadius(float)` (in 16.1.11, page 410)

- *setSamplingRadius*

public void **setSamplingRadius**(float **newValue**)

- **Description**

Sets the sampling radius to a new value. Call this method if you do not want to use the default radius as provided by (in 16.1.11, page 410).

- **Parameters**

- * **newValue** – new sampling radius to be used with this object

16.1.12 Class Rotate180

Rotates images by 180 degrees. The result is the same as a a (in 16.1.5, page 399)operation followed by a (in 16.1.8, page 402)operation (or vice versa). Input image must implement (in 14.1.7, page 322).

Usage example

```
Rotate180 rotate = new Rotate180();
rotate.setInputImage(image); // something implementing IntegerImage
rotate.process();
PixelImage rotatedImage = rotate.getOutputImage();
```

Declaration

```
public class Rotate180
  extends net.sourceforge.jiu.ops.ImageToImageOperation (in 19.2.3, page 501)
```

Constructor summary

Rotate180()

Method summary

process()

Constructors

- *Rotate180*
public **Rotate180**()

Methods

- *process*
public void **process**() throws
net.sourceforge.jiu.ops.MissingParameterException,
net.sourceforge.jiu.ops.OperationFailedException,
net.sourceforge.jiu.ops.WrongParameterException
 - **Description** copied from net.sourceforge.jiu.ops.Operation (in 19.2.5, page 508)
This method does the actual work of the operation. It must be called after all parameters have been given to the operation object.
 - **Throws**
 - * net.sourceforge.jiu.ops.WrongParameterException – if at least one of the input parameters was not initialized appropriately (values out of the valid interval, etc.)

- * `net.sourceforge.jiu.ops.MissingParameterException` – if any mandatory parameter was not given to the operation
- * `net.sourceforge.jiu.ops.OperationFailedException` –

16.1.13 Class Rotate90Left

Rotates images by 90 degrees counter-clockwise (to the left). Input image must implement (in 14.1.7, page 322).

Usage example

```
Rotate90Left rotate = new Rotate90Left();
rotate.setInputImage(image);
rotate.process();
PixelImage rotatedImage = rotate.getOutputImage();
```

Declaration

```
public class Rotate90Left
  extends net.sourceforge.jiu.ops.ImageToImageOperation (in 19.2.3, page 501)
```

Constructor summary

Rotate90Left()

Method summary

process()

Constructors

- *Rotate90Left*
public **Rotate90Left**()

Methods

- *process*
public void **process**() throws
net.sourceforge.jiu.ops.MissingParameterException,
net.sourceforge.jiu.ops.OperationFailedException,
net.sourceforge.jiu.ops.WrongParameterException
 - **Description copied from net.sourceforge.jiu.ops.Operation** (in 19.2.5, page 508)
This method does the actual work of the operation. It must be called after all parameters have been given to the operation object.
 - **Throws**
 - * net.sourceforge.jiu.ops.WrongParameterException – if at least one of the input parameters was not initialized appropriately (values out of the valid interval, etc.)
 - * net.sourceforge.jiu.ops.MissingParameterException – if any mandatory parameter was not given to the operation
 - * net.sourceforge.jiu.ops.OperationFailedException –

16.1.14 Class Rotate90Right

Rotates images by 90 degrees clockwise (to the right). Input image must implement (in 14.1.7, page 322).

Usage example

```
Rotate90Right rotate = new Rotate90Right();
rotate.setInputImage(image);
rotate.process();
PixelImage rotatedImage = rotate.getOutputImage();
```

Declaration

```
public class Rotate90Right
extends net.sourceforge.jiu.ops.ImageToImageOperation (in 19.2.3, page 501)
```

Constructor summary

Rotate90Right()

Method summary

process()

Constructors

- *Rotate90Right*
public Rotate90Right()

Methods

- *process*
public void process() throws
`net.sourceforge.jiu.ops.MissingParameterException`,
`net.sourceforge.jiu.ops.OperationFailedException`,
`net.sourceforge.jiu.ops.WrongParameterException`
 - **Description copied from `net.sourceforge.jiu.ops.Operation` (in 19.2.5, page 508)**
This method does the actual work of the operation. It must be called after all parameters have been given to the operation object.
 - **Throws**
 - * `net.sourceforge.jiu.ops.WrongParameterException` – if at least one of the input parameters was not initialized appropriately (values out of the valid interval, etc.)
 - * `net.sourceforge.jiu.ops.MissingParameterException` – if any mandatory parameter was not given to the operation

```
* net.sourceforge.jiu.ops.OperationFailedException –
```


16.1.15 Class ScaleReplication

Changes the pixel resolution of an image by replicating (or dropping) pixels. A fast but low quality scaling algorithm that works with all kinds of image types. (in 16.1.10, page 405) provides better quality, but is slower and works with intensity-based image data types only.

Usage example

The input image will be scaled to an image that is twice as wide as itself and three times as high.

```
ScaleReplication scale = new ScaleReplication();
scale.setInputImage(image); // something implementing IntegerImage
scale.setSize(image.getWidth() * 2, image.getHeight() * 2);
scale.process();
PixelImage scaledImage = scale.getOutputImage();
```

Declaration

```
public class ScaleReplication
extends net.sourceforge.jiu.ops.ImageToImageOperation (in 19.2.3, page 501)
```

Constructor summary

ScaleReplication()

Method summary

process()
setSize(int, int) Specify the resolution to be used for the image to be created.

Constructors

- *ScaleReplication*
public ScaleReplication()

Methods

- *process*
public void process() throws
 net.sourceforge.jiu.ops.MissingParameterException,
 net.sourceforge.jiu.ops.OperationFailedException,
 net.sourceforge.jiu.ops.WrongParameterException
 - **Description** copied from net.sourceforge.jiu.ops.Operation (in 19.2.5, page 508)
 This method does the actual work of the operation. It must be called after all parameters have been given to the operation object.
 - **Throws**

- * `net.sourceforge.jiu.ops.WrongParameterException` – if at least one of the input parameters was not initialized appropriately (values out of the valid interval, etc.)
 - * `net.sourceforge.jiu.ops.MissingParameterException` – if any mandatory parameter was not given to the operation
 - * `net.sourceforge.jiu.ops.OperationFailedException` –
-

- *setSize*

`public void setSize(int width, int height)`

- **Description**

Specify the resolution to be used for the image to be created.

- **Parameters**

- * `width` – horizontal resolution of the new image
 - * `height` – vertical resolution of the new image

- **Throws**

- * `java.lang.IllegalArgumentException` – if any of the arguments is smaller than 1

16.1.16 Class Shear

Shears an image by a given angle. The angle must be larger than -90 and smaller than 90 degrees. Shearing works with all image types that implement `IntegerImage` (in 14.1.7, page 322).

Usage example

```
Shear shear = new Shear();
shear.setInputImage(image); // some IntegerImage
shear.setAngle(5.0);
shear.process();
PixelImage shearedImage = shear.getOutputImage();
```

This is an adjusted version of Jef Poskanzer's shearing code from his ACME package; see the **API documentation page** (at <http://www.acme.com/java/software/Acme.JPM.Filters.Shear.html>) of ACME's `Shear` class.

Declaration

```
public class Shear
extends net.sourceforge.jiu.ops.ImageToImageOperation (in 19.2.3, page 501)
```

Constructor summary

Shear()

Method summary

computeNewImageWidth(int, int, double) For a given image width and shearing angle this method computes the width of the resulting image.
getAngle() Returns the angle associated with this shearing operation object.
process()
setAngle(double) Sets the angle to be used in the shearing operation to the argument value.

Constructors

- *Shear*
public Shear()

Methods

- *computeNewImageWidth*
public static int computeNewImageWidth(int oldImageWidth, int height, double angle)
 – **Description**
 For a given image width and shearing angle this method computes the width of the resulting image. This method is static so that it can be called easily from a GUI dialog or other objects that want to present the width of a sheared image.

- **Parameters**

- * `oldImageWidth` – horizontal resolution of the image to be sheared
- * `height` – height of the image to be sheared
- * `angle` – the angle to be used in the shearing operation

- **Returns** – width of the sheared image

- *getAngle*

`public double getAngle()`

- **Description**

Returns the angle associated with this shearing operation object.

- **Returns** – shearing angle, between -90 and 90

- **See also**

- * `Shear.setAngle(double)` (in 16.1.16, page 419)
-

- *process*

`public void process()` throws

`net.sourceforge.jiu.ops.MissingParameterException`,
`net.sourceforge.jiu.ops.OperationFailedException`,
`net.sourceforge.jiu.ops.WrongParameterException`

- **Description copied from `net.sourceforge.jiu.ops.Operation` (in 19.2.5, page 508)**

This method does the actual work of the operation. It must be called after all parameters have been given to the operation object.

- **Throws**

- * `net.sourceforge.jiu.ops.WrongParameterException` – if at least one of the input parameters was not initialized appropriately (values out of the valid interval, etc.)
 - * `net.sourceforge.jiu.ops.MissingParameterException` – if any mandatory parameter was not given to the operation
 - * `net.sourceforge.jiu.ops.OperationFailedException` –
-

- *setAngle*

`public void setAngle(double newAngle)`

- **Description**

Sets the angle to be used in the shearing operation to the argument value. The angle must be larger than -90.0 and smaller than 90.0.

- **Parameters**

- * `newAngle` – the angle to be used in this operation

- **Throws**

- * `java.lang.IllegalArgumentException` – if the argument is not in the above mentioned interval

- **See also**

- * `Shear.getAngle()` (in 16.1.16, page 419)

16.1.17 Class *TriangleFilter*

A triangle filter (also known as linear or bilinear filter).

Declaration

```
public class TriangleFilter
extends net.sourceforge.jiu.geometry.ResampleFilter (in 16.1.11, page 409)
```

Constructor summary

TriangleFilter()

Method summary

apply(float)
getName()
getRecommendedSamplingRadius()

Constructors

- *TriangleFilter*
public TriangleFilter()

Methods

- *apply*
public abstract float apply(float value)
 - **Description copied from ResampleFilter (in 16.1.11, page 409)**
Returns the weight of the sample at the distance given by the argument value.
- *getName*
public abstract java.lang.String getName()
 - **Description copied from ResampleFilter (in 16.1.11, page 409)**
Return the name of this filter. Should avoid natural language words if possible.
 - **Returns** – String with filter name
- *getRecommendedSamplingRadius*
public abstract float getRecommendedSamplingRadius()
 - **Description copied from ResampleFilter (in 16.1.11, page 409)**
Returns a recommendation for the sampling radius to be used with this filter. This recommendation value will be the default value for the sampling radius of objects of this class. You can modify it with a call to (in 16.1.11, page 410).
 - **Returns** – the recommended sampling radius to be used with this filter

Chapter 17

Package net.sourceforge.jiu.gui.awt

Package Contents

Page

Classes

AwtInfo	422
<i>Retrieve some information on the current graphical environment.</i>	
AwtMenuWrapper	423
<i>A wrapper around an AWT MenuBar object.</i>	
AwtOperationProcessor	425
<i>Performs operations specified by parent class (in 1.2.10, page 52), uses various AWT dialogs to get parameters from user in a GUI application.</i>	
BufferedRGB24Image	433
<i>A bridge class to use objects (class defined in the standard runtime library, package java.awt.image) as (in 14.1.12, page 331) objects within JIU.</i>	
ImageCanvas	437
<i>An AWT canvas that displays an object.</i>	
ImageCreator	440
<i>A class to create java.awt.Image objects from various JIU image data types and vice versa.</i>	
JiuAwtFrame	443
<i>The frame class for the AWT demo program (in 1.2.6, page 47).</i>	
RGBA	448
<i>This class converts between the 32 bit RGBA int values (used throughout the AWT) and various standard pixel formats like 24 bits RGB, 8 bits gray, 16 bits gray, 1 bit black and white.</i>	
ToolkitLoader	452
<i>This class loads an instance of using 's built-in loading capabilities and converts it to (in 14.1.12, page 331) using (in 17.1.6, page 440).</i>	

Classes to interoperate with Java's first GUI toolkit, the AWT (Abstract Windowing Toolkit). This includes GUI components and conversion from JIU's image types to AWT's image types.

17.1 Classes

17.1.1 Class AwtInfo

Retrieve some information on the current graphical environment.

Declaration

```
public class AwtInfo
extends java.lang.Object
```

Method summary

getAwtInfo(Strings) Returns information on the current AWT settings, regarding the current language by using a (in 1.2.12, page 61)resource.

Methods

- *getAwtInfo*
`public static java.lang.String getAwtInfo(net.sourceforge.jiu.apps.Strings strings)`
 - **Description**
Returns information on the current AWT settings, regarding the current language by using a (in 1.2.12, page 61)resource. Right now, only returns the screen resolution. All textual information is taken from the strings argument.
 - **Parameters**
 - * **strings** – String resources
 - **Returns** – AWT information

17.1.2 Class AwtMenuWrapper

A wrapper around an AWT MenuBar object.

Declaration

```
public class AwtMenuWrapper
  extends net.sourceforge.jiu.apps.MenuWrapper (in 1.2.9, page 50)
```

Constructor summary

AwtMenuWrapper(Strings, ActionListener) Internally creates a MenuBar object and provides methods to update that menu bar.

Method summary

findIndex(Object) Attempts to find one of the menu items in the internal list.
getMenuBar()
setEnabled(int, boolean)
setLabel(int, String)
updateEnabled(Processor)
updateLabels(Strings)

Constructors

- *AwtMenuWrapper*

```
public AwtMenuWrapper( net.sourceforge.jiu.apps.Strings strings,
  java.awt.event.ActionListener actionListener )
```

 - **Description**
Internally creates a MenuBar object and provides methods to update that menu bar.
 - **Parameters**
 - * **strings** – String resource used to initialize menu items
 - * **actionListener** – a listener which will be registered with all menu items

Methods

- *findIndex*

```
public int findIndex( java.lang.Object o )
```

 - **Description**
Attempts to find one of the menu items in the internal list. Returns its index or -1 if it is not one of the items.
- *getMenuBar*

```
public java.awt.MenuBar getMenuBar( )
```
- *setEnabled*

```
public abstract void setEnabled( int index, boolean enabled )
```


- **Description** copied from net.sourceforge.jiu.apps.MenuWrapper (in 1.2.9, page 50)

Sets the enabled status of one of the menu items to either `true` or `false`.

- **Parameters**

- * `index` – menu index of the component whose status is to be reset
 - * `enabled` – boolean with the new value
-

- *setLabel*

```
public abstract void setLabel( int index, java.lang.String text )
```

- **Description** copied from net.sourceforge.jiu.apps.MenuWrapper (in 1.2.9, page 50)

Sets the text of one of the menu elements to a new value. This method is usually called when the language settings have changed and new words have to be assigned.

- **Parameters**

- * `index` – integer index of the menu element
 - * `text` – new text value to be used for this element
-

- *updateEnabled*

```
public void updateEnabled( net.sourceforge.jiu.apps.OperationProcessor op )
```

- *updateLabels*

```
public void updateLabels( net.sourceforge.jiu.apps.Strings strings )
```

17.1.3 Class AwtOperationProcessor

Performs operations specified by parent class (in 1.2.10, page 52), uses various AWT dialogs to get parameters from user in a GUI application.

Declaration

```
public class AwtOperationProcessor
extends net.sourceforge.jiu.apps.OperationProcessor (in 1.2.10, page 52)
```

Constructor summary

AwtOperationProcessor(EditorState, JiuAwtFrame)

Method summary

```
colorAdjustBrightness()
colorAdjustContrast()
colorAdjustGamma()
colorAdjustHueSaturationValue()
colorConvertToMinimumColorType()
colorHistogramCountColorsUsed()
colorHistogramEqualize()
colorHistogramNormalize()
colorHistogramSaveCoOccurrenceFrequencyMatrixAs()
colorHistogramSaveCoOccurrenceMatrixAs()
colorHistogramSaveHistogramAs()
colorHistogramTextureProperties()
colorInvert()
colorPaletteSaveAs()
colorPromotePromoteToGray16()
colorPromotePromoteToGray8()
colorPromotePromoteToPaletted()
colorPromotePromoteToRgb24()
colorPromotePromoteToRgb48()
colorReduceConvertToGrayscale()
colorReduceMapToArbitraryPalette()
colorReduceMedianCut()
colorReduceOctree()
colorReduceReduceNumberOfShadesOfGray()
colorReduceReduceToBilevelThreshold()
colorReduceUniformPalette()
editRedo()
editUndo()
fileClose()
fileExit()
fileOpen()
fileSaveAsBmp()
fileSaveAsGif()
fileSaveAsPalm()
```

```

fileSaveAsPbm()
fileSaveAsPgm()
fileSaveAsPng()
fileSaveAsPpm()
fileSaveAsRas()
filterConvolutionFilter(int)
filtersBlur()
filtersEdgeDetection()
filtersEmboss()
filtersHorizontalPrewitt()
filtersHorizontalSobel()
filtersLithograph()
filtersMaximum()
filtersMean()
filtersMedian()
filtersMinimum()
filtersOil()
filtersPsychedelicDistillation()
filtersSharpen()
filtersVerticalPrewitt()
filtersVerticalSobel()
getUserFileName(String, int, int)
getUserSaveAsFileName(String, int)
helpAbout()
helpSystemInformation()
process(ImageToImageOperation) This method can be called for
    ImageToImageOperation objects.
setImage(PixelImage, boolean)
transformationsCrop()
transformationsFlip()
transformationsMirror()
transformationsRotate180()
transformationsRotate90Left()
transformationsRotate90Right()
transformationsScale()
transformationsShear()
updateFrame(PixelImage)
viewInterpolationTypeBicubic()
viewInterpolationTypeBilinear()
viewInterpolationTypeNearestNeighbor()
viewSetOriginalSize()
viewZoomIn()
viewZoomOut()

```

Constructors

- *AwtOperationProcessor*

```
public AwtOperationProcessor( net.sourceforge.jiu.apps.EditorState
editorState, JiuAwtFrame awtFrame )
```

Methods

- *colorAdjustBrightness*
 public abstract void **colorAdjustBrightness**()
 – Description copied from net.sourceforge.jiu.apps.OperationProcessor (in 1.2.10, page 52)
 Adjust the brightness of the current image.

- *colorAdjustContrast*
 public abstract void **colorAdjustContrast**()
 – Description copied from net.sourceforge.jiu.apps.OperationProcessor (in 1.2.10, page 52)
 Adjust the contrast of the current image.

- *colorAdjustGamma*
 public abstract void **colorAdjustGamma**()
 – Description copied from net.sourceforge.jiu.apps.OperationProcessor (in 1.2.10, page 52)
 Adjust the gamma value of the current image.

- *colorAdjustHueSaturationValue*
 public abstract void **colorAdjustHueSaturationValue**()
 – Description copied from net.sourceforge.jiu.apps.OperationProcessor (in 1.2.10, page 52)
 Adjust hue, saturation and value of the current image.

- *colorConvertToMinimumColorType*
 public abstract void **colorConvertToMinimumColorType**()

- *colorHistogramCountColorsUsed*
 public abstract void **colorHistogramCountColorsUsed**()
 – Description copied from net.sourceforge.jiu.apps.OperationProcessor (in 1.2.10, page 52)
 Count the number of colors used in the current image.

- *colorHistogramEqualize*
 public abstract void **colorHistogramEqualize**()

- *colorHistogramNormalize*
 public abstract void **colorHistogramNormalize**()

- *colorHistogramSaveCoOccurrenceFrequencyMatrixAs*
 public abstract void **colorHistogramSaveCoOccurrenceFrequencyMatrixAs**()

- *colorHistogramSaveCoOccurrenceMatrixAs*
 public abstract void **colorHistogramSaveCoOccurrenceMatrixAs**()

- *colorHistogramSaveHistogramAs*
 public abstract void **colorHistogramSaveHistogramAs**()

- *colorHistogramTextureProperties*
public abstract void colorHistogramTextureProperties()
 - *colorInvert*
public abstract void colorInvert()
 - *colorPaletteSaveAs*
public abstract void colorPaletteSaveAs()
 - *colorPromotePromoteToGray16*
public abstract void colorPromotePromoteToGray16()
 - *colorPromotePromoteToGray8*
public abstract void colorPromotePromoteToGray8()
 - *colorPromotePromoteToPaletted*
public abstract void colorPromotePromoteToPaletted()
 - *colorPromotePromoteToRgb24*
public abstract void colorPromotePromoteToRgb24()
 - *colorPromotePromoteToRgb48*
public abstract void colorPromotePromoteToRgb48()
 - *colorReduceConvertToGrayscale*
public abstract void colorReduceConvertToGrayscale()
 - *colorReduceMapToArbitraryPalette*
public abstract void colorReduceMapToArbitraryPalette()
 - *colorReduceMedianCut*
public abstract void colorReduceMedianCut()
 - *colorReduceOctree*
public abstract void colorReduceOctree()
 - *colorReduceReduceNumberOfShadesOfGray*
public abstract void colorReduceReduceNumberOfShadesOfGray()
 - *colorReduceReduceToBilevelThreshold*
public abstract void colorReduceReduceToBilevelThreshold()
 - *colorReduceUniformPalette*
public abstract void colorReduceUniformPalette()
 - *editRedo*
public abstract void editRedo()
 - *editUndo*
public abstract void editUndo()
 - *fileClose*
public abstract void fileClose()
 - Description copied from net.sourceforge.jiu.apps.OperationProcessor (in 1.2.10, page 52)
If there is an image loaded in the application, remove the image.
-

- *fileExit*
public abstract void fileExit()
 - Description copied from net.sourceforge.jiu.apps.OperationProcessor (in 1.2.10, page 52)
Terminate the application. If changes were not saved, the user should be asked whether these changes should be discarded.

- *fileOpen*
public abstract void fileOpen()
 - Description copied from net.sourceforge.jiu.apps.OperationProcessor (in 1.2.10, page 52)
Load an image in the application.

- *fileSaveAsBmp*
public abstract void fileSaveAsBmp()
 - Description copied from net.sourceforge.jiu.apps.OperationProcessor (in 1.2.10, page 52)
Save the current image as a Windows BMP file.

- *fileSaveAsGif*
public abstract void fileSaveAsGif()
 - Description copied from net.sourceforge.jiu.apps.OperationProcessor (in 1.2.10, page 52)
Save the current image as a GIF file.

- *fileSaveAsPalm*
public abstract void fileSaveAsPalm()
 - Description copied from net.sourceforge.jiu.apps.OperationProcessor (in 1.2.10, page 52)
Save the current image as a Palm image file.

- *fileSaveAsPbm*
public abstract void fileSaveAsPbm()
 - Description copied from net.sourceforge.jiu.apps.OperationProcessor (in 1.2.10, page 52)
Save the current image as a Portable Bitmap file.

- *fileSaveAsPgm*
public abstract void fileSaveAsPgm()
 - Description copied from net.sourceforge.jiu.apps.OperationProcessor (in 1.2.10, page 52)
Save the current image as a Portable Graymap file.

- *fileSaveAsPng*
public abstract void fileSaveAsPng()

- **Description copied from net.sourceforge.jiu.apps.OperationProcessor (in 1.2.10, page 52)**

Save the current image as a Portable Network Graphics file.

- *fileSaveAsPpm*

public abstract void fileSaveAsPpm()

- **Description copied from net.sourceforge.jiu.apps.OperationProcessor (in 1.2.10, page 52)**

Save the current image as a Portable Pixmap file.

- *fileSaveAsRas*

public abstract void fileSaveAsRas()

- **Description copied from net.sourceforge.jiu.apps.OperationProcessor (in 1.2.10, page 52)**

Save the current image as a Sun Raster file.

- *filterConvolutionFilter*

public void filterConvolutionFilter(int type)

- *filtersBlur*

public abstract void filtersBlur()

- *filtersEdgeDetection*

public abstract void filtersEdgeDetection()

- *filtersEmboss*

public abstract void filtersEmboss()

- *filtersHorizontalPrewitt*

public abstract void filtersHorizontalPrewitt()

- *filtersHorizontalSobel*

public abstract void filtersHorizontalSobel()

- *filtersLithograph*

public abstract void filtersLithograph()

- *filtersMaximum*

public abstract void filtersMaximum()

- *filtersMean*

public abstract void filtersMean()

- *filtersMedian*

public abstract void filtersMedian()

- *filtersMinimum*

public abstract void filtersMinimum()

- *filtersOil*

public abstract void filtersOil()

- *filtersPsychedelicDistillation*

public abstract void filtersPsychedelicDistillation()

- *filtersSharpen*

public abstract void **filtersSharpen**()
- *filtersVerticalPrewitt*

public abstract void **filtersVerticalPrewitt**()
- *filtersVerticalSobel*

public abstract void **filtersVerticalSobel**()
- *getUserFileName*

public java.lang.String **getUserFileName**(java.lang.String extension, int titleIndex, int fileDialogType)
- *getUserSaveAsFileName*

public java.lang.String **getUserSaveAsFileName**(java.lang.String extension, int titleIndex)
- *helpAbout*

public abstract void **helpAbout**()
 - **Description copied from net.sourceforge.jiu.apps.OperationProcessor (in 1.2.10, page 52)**
Display information about the application: name, version, feedback email address, website.

- *helpSystemInformation*

public abstract void **helpSystemInformation**()
 - **Description copied from net.sourceforge.jiu.apps.OperationProcessor (in 1.2.10, page 52)**
Display information on the system this application is currently running on.

- *process*

public void **process**(net.sourceforge.jiu.ops.ImageToImageOperation op)
 - **Description**
This method can be called for ImageToImageOperation objects.

- *setImage*

public void **setImage**(net.sourceforge.jiu.data.PixelImage newImage, boolean newModified)
- *transformationsCrop*

public abstract void **transformationsCrop**()
- *transformationsFlip*

public abstract void **transformationsFlip**()
- *transformationsMirror*

public abstract void **transformationsMirror**()
- *transformationsRotate180*

public abstract void **transformationsRotate180**()
- *transformationsRotate90Left*

public abstract void **transformationsRotate90Left**()

- *transformationsRotate90Right*
public abstract void transformationsRotate90Right()
- *transformationsScale*
public abstract void transformationsScale()
- *transformationsShear*
public abstract void transformationsShear()
- *updateFrame*
public void updateFrame(net.sourceforge.jiu.data.PixelImage image)
- *viewInterpolationTypeBicubic*
public abstract void viewInterpolationTypeBicubic()
- *viewInterpolationTypeBilinear*
public abstract void viewInterpolationTypeBilinear()
- *viewInterpolationTypeNearestNeighbor*
public abstract void viewInterpolationTypeNearestNeighbor()
- *viewSetOriginalSize*
public abstract void viewSetOriginalSize()
- *viewZoomIn*
public abstract void viewZoomIn()
- *viewZoomOut*
public abstract void viewZoomOut()

17.1.4 Class BufferedRGB24Image

A bridge class to use `Image` objects (class defined in the standard runtime library, package `java.awt.image`) as (in 14.1.12, page 331) objects within JIU. This class encapsulates a single object. It enables reusing existing `BufferedImage` objects as input or output of JIU operations, removing the necessity for the conversion step from `java.awt.Image` to `net.sourceforge.jiu.data.PixelImage` (or vice versa) and thus reducing memory consumption. The name of this class is a combination of `BufferedImage` (the class of the object that is encapsulated) and `RGB24Image` (the JIU image data interface).

Internally, this class uses `Image`'s `getRGB` and `setRGB` methods to access image data. This approach is slower than working directly on the `BufferedImage`'s data buffers. However, using `getRGB` and `setRGB`, this class will work with all types of `BufferedImage` objects.

Note that while the abstract `java.awt.Image` class existed from the very beginning (version 1.0) of the Java runtime library, `java.awt.image.BufferedImage` has not been added until version 1.2.

Usage example

This code snippet demonstrates to how combine functionality from Java's runtime library with JIU by using this class. Requires Java 1.4 or higher. Obviously, `BufferedRGB24Image` objects can only be used with operations that work on classes implementing `RGB24Image`.

```
import java.awt.image.BufferedImage;
import java.io.File;
import javax.imageio.ImageIO;
import net.sourceforge.jiu.color.Invert;
import net.sourceforge.jiu.data.PixelImage;
import net.sourceforge.jiu.gui.awt.BufferedRGB24Image;
...
BufferedImage bufferedImage = ImageIO.read(new File("image.jpg"));
BufferedRGB24Image image = new BufferedRGB24Image(bufferedImage);
Invert invert = new Invert();
invert.setInputImage(image);
invert.process();
PixelImage outputImage = invert.getOutputImage();
```

If you can be sure that an image object can be input and output image at the same time (as is the case with some operations), you can even work with only one `BufferedRGB24Image` object. `Invert` is one of these operations, so the following would work:

```
Invert invert = new Invert();
invert.setInputImage(image);
invert.setOutputImage(image);
invert.process();
// image now is inverted
```

Declaration

```
public class BufferedRGB24Image
extends java.lang.Object
implements net.sourceforge.jiu.data.RGB24Image
```

Constructor summary

BufferedRGB24Image(BufferedImage) Creates a new BufferedRGB24Image object, storing the argument BufferedImage object internally.

Method summary

clear(byte) Sets all the RGB samples in this image to the argument, keeping the alpha value.

clear(int)

clear(int, byte)

clear(int, int)

createCompatibleImage(int, int)

createCopy()

getAllocatedMemory()

getBitsPerPixel()

getByteSample(int, int)

getByteSample(int, int, int)

getByteSamples(int, int, int, int, byte[], int)

getByteSamples(int, int, int, int, int, byte[], int)

getHeight()

getImageType()

getMaxSample(int)

getNumChannels()

getSample(int, int)

getSample(int, int, int)

getSamples(int, int, int, int, int[], int)

getSamples(int, int, int, int, int, int[], int)

getWidth()

putByteSample(int, int, byte)

putByteSample(int, int, int, byte)

putByteSamples(int, int, int, int, byte[], int)

putByteSamples(int, int, int, int, int, byte[], int)

putSample(int, int, int)

putSample(int, int, int, int)

putSamples(int, int, int, int, int, int[], int)

Constructors

- *BufferedRGB24Image*

```
public BufferedRGB24Image( java.awt.image.BufferedImage bufferedImage )
```

- **Description**

Creates a new BufferedRGB24Image object, storing the argument BufferedImage object internally. All image data access will be delegated to that BufferedImage object's methods.

- **Parameters**

- * **bufferedImage** – the underlying BufferedImage object for this BufferedRGB24Image object

Methods

- *clear*
 public void **clear**(byte **newValue**)
 – **Description**
 Sets all the RGB samples in this image to the argument, keeping the alpha value.
 – **Parameters**
 * **newValue** – all samples in the image will be set to this value

- *clear*
 public void **clear**(int **newValue**)

- *clear*
 public void **clear**(int **channelIndex**, byte **newValue**)

- *clear*
 public void **clear**(int **channelIndex**, int **newValue**)

- *createCompatibleImage*
 public net.sourceforge.jiu.data.PixelImage **createCompatibleImage**(int **width**, int **height**)

- *createCopy*
 public net.sourceforge.jiu.data.PixelImage **createCopy**()

- *getAllocatedMemory*
 public long **getAllocatedMemory**()

- *getBitsPerPixel*
 public int **getBitsPerPixel**()

- *getByteSample*
 public byte **getByteSample**(int **x**, int **y**)

- *getByteSample*
 public byte **getByteSample**(int **channelIndex**, int **x**, int **y**)

- *getByteSamples*
 public void **getByteSamples**(int **x**, int **y**, int **w**, int **h**, byte[] **dest**, int **destOffset**)

- *getByteSamples*
 public void **getByteSamples**(int **channelIndex**, int **x**, int **y**, int **w**, int **h**, byte[] **dest**, int **destOffset**)

- *getHeight*
 public int **getHeight**()

- *getImageType*
 public java.lang.Class **getImageType**()

- *getMaxSample*
 public int **getMaxSample**(int **channel**)

- *getNumChannels*
public int **getNumChannels**()
- *getSample*
public int **getSample**(int x, int y)
- *getSample*
public int **getSample**(int channelIndex, int x, int y)
- *getSamples*
public void **getSamples**(int x, int y, int w, int h, int[] dest, int destOffs)
- *getSamples*
public void **getSamples**(int channelIndex, int x, int y, int w, int h, int[] dest, int destOffs)
- *getWidth*
public int **getWidth**()
- *putByteSample*
public void **putByteSample**(int x, int y, byte newValue)
- *putByteSample*
public void **putByteSample**(int channelIndex, int x, int y, byte newValue)
- *putByteSamples*
public void **putByteSamples**(int x, int y, int w, int h, byte[] src, int srcOffset)
- *putByteSamples*
public void **putByteSamples**(int channelIndex, int x, int y, int w, int h, byte[] src, int srcOffset)
- *putSample*
public void **putSample**(int x, int y, int newValue)
- *putSample*
public void **putSample**(int channelIndex, int x, int y, int newValue)
- *putSamples*
public void **putSamples**(int channelIndex, int x, int y, int w, int h, int[] src, int srcOffset)

17.1.5 Class ImageCanvas

An AWT canvas that displays an `Image` object. Capable to display at arbitrary zooming levels. Does not use rendering hints because they require Java 1.2 or higher (although bilinear and bicubic interpolation usually improve display quality when zooming at the cost of slowing down image drawing).

Declaration

```
public class ImageCanvas
    extends java.awt.Canvas
```

Constructor summary

ImageCanvas(ScrollPane)

Method summary

computeZoomToFitSize()
getPreferredSize()
getZoomFactorX()
getZoomFactorY()
getZoomPercentageX()
getZoomPercentageY()
paint(Graphics) Draws image to upper left corner.
setImage(Image) Specifies a new `Image` object to be displayed in this canvas.
setInterpolation(int) Sets the interpolation type used for drawing to the argument (must be one of the `INTERPOLATION_xyz` constants of `EditorState`), but does not do a redraw.
setOriginalSize() Sets both zoom factors to 1.0.
setZoomFactor(double)
setZoomFactors(double, double)
setZoomToFit(boolean)
update(Graphics) Simply calls `paint` (in 17.1.5, page 438) with the argument.

Serializable Fields

- private `java.awt.Image` **image**
- private `int` **width**
- private `int` **height**
- private `java.lang.Object` **interpolation**
- private `int` **scaledWidth**
- private `int` **scaledHeight**
- private `double` **zoomFactorX**
- private `double` **zoomFactorY**

- private boolean **zoomToFit**
- private java.awt.ScrollPane **myScrollPane**

Constructors

- *ImageCanvas*
public **ImageCanvas**(java.awt.ScrollPane **scrollPane**)

Methods

- *computeZoomToFitSize*
public void **computeZoomToFitSize**()
- *getPreferredSize*
public java.awt.Dimension **getPreferredSize**()
- *getZoomFactorX*
public double **getZoomFactorX**()
- *getZoomFactorY*
public double **getZoomFactorY**()
- *getZoomPercentageX*
public int **getZoomPercentageX**()
- *getZoomPercentageY*
public int **getZoomPercentageY**()
- *paint*
public void **paint**(java.awt.Graphics **g**)

 - **Description**
Draws image to upper left corner.
- *setImage*
public void **setImage**(java.awt.Image **newImage**)

 - **Description**
Specifies a new Image object to be displayed in this canvas.
 - **Parameters**
* **newImage** – the new Image object, potentially null
- *setInterpolation*
public void **setInterpolation**(int **newType**)

 - **Description**
Sets the interpolation type used for drawing to the argument (must be one of the INTERPOLATION_xyz constants of EditorState), but does not do a redraw.
- *setOriginalSize*
public void **setOriginalSize**()

– **Description**

Sets both zoom factors to 1.0.

• *setZoomFactor*

public void setZoomFactor(double newZoomFactor)

• *setZoomFactors*

**public void setZoomFactors(double newZoomFactorX, double
newZoomFactorY)**

• *setZoomToFit*

public void setZoomToFit(boolean newValue)

• *update*

public void update(java.awt.Graphics g)

– **Description**

Simply calls (in 17.1.5, page 438)with the argument.

– **Parameters**

* **g** – Graphics context

17.1.6 Class ImageCreator

A class to create java.awt.Image objects from various JIU image data types and vice versa. java.awt.Image objects can be used with the AWT and Swing GUI environments.

Declaration

```
public class ImageCreator
extends java.lang.Object
```

Field summary

DEFAULT_ALPHA The default transparency value to be used: full opacity.

Method summary

convertImageToRGB24Image(Image) Creates an (in 14.1.12, page 331) from the argument AWT image instance.

convertToAwtImage(BilevelImage, int) Convert a BilevelImage object to an AWT image object.

convertToAwtImage(Gray16Image, int) Creates an AWT Image object from a Gray16Image object and an alpha value.

convertToAwtImage(Gray8Image, int) Creates an AWT Image object from a Gray8Image object and an alpha value.

convertToAwtImage(Palett8Image, int)

convertToAwtImage(PixelImage, int) Creates an instance of from an instance of (in 14.1.12, page 331).

convertToAwtImage(RGB24Image, int)

convertToAwtImage(RGB48Image, int)

createImage(int[], int, int) Creates a object from a pixel array.

Fields

- public static final int **DEFAULT_ALPHA**
 - The default transparency value to be used: full opacity.

Methods

- *convertImageToRGB24Image*

```
public static net.sourceforge.jiu.data.RGB24Image
convertImageToRGB24Image( java.awt.Image image )
```

 - **Description**

Creates an (in 14.1.12, page 331) from the argument AWT image instance.
 - **Parameters**
 - * **image** – AWT image object to be converted to a (in 14.1.12, page 331)
 - **Returns** – a (in 14.1.12, page 331) object holding the image data from the argument image

- *convertToAwtImage*

```
public static java.awt.Image convertToAwtImage(
net.sourceforge.jiu.data.BilevelImage image, int alpha )
```

- **Description**

Convert a BilevelImage object to an AWT image object.

- **Parameters**

- * **image** – the image to be converted
- * **alpha** – the transparency value to be written to each pixel in the resulting image

- **Returns** – newly-created AWT image

- *convertToAwtImage*

```
public static java.awt.Image convertToAwtImage(
net.sourceforge.jiu.data.Gray16Image image, int alpha )
```

- **Description**

Creates an AWT Image object from a Gray16Image object and an alpha value. This is done by allocating a new int array with image.getWidth() times image.getHeight() elements, copying the data to those ints (using transparency information from the top eight bits of the alpha argument) and calling Toolkit.createImage with a MemoryImageSource of those int[] pixels.

- **Parameters**

- * **image** – the grayscale image to be converted
- * **alpha** – the alpha value, bits must only be set in the top eight bits

- **Returns** – AWT image created from the argument input image

- *convertToAwtImage*

```
public static java.awt.Image convertToAwtImage(
net.sourceforge.jiu.data.Gray8Image image, int alpha )
```

- **Description**

Creates an AWT Image object from a Gray8Image object and an alpha value. This is done by allocating a new int array with image.getWidth() times image.getHeight() elements, copying the data to those ints (using transparency information from the top eight bits of the alpha argument) and calling Toolkit.createImage with a MemoryImageSource of those int[] pixels.

- **Parameters**

- * **image** – the grayscale image to be converted
- * **alpha** – the alpha value, bits must only be set in the top eight bits

- **Returns** – AWT image created from the argument input image

- *convertToAwtImage*

```
public static java.awt.Image convertToAwtImage(
net.sourceforge.jiu.data.Paletted8Image image, int alpha )
```

- *convertToAwtImage*

```
public static java.awt.Image convertToAwtImage(
net.sourceforge.jiu.data.PixelImage image, int alpha )
```

– **Description**

Creates an instance of `Image` from an instance of `net.sourceforge.jiu.data.RGB24Image` (in 14.1.12, page 331). This will require `image.getWidth() * image.getHeight() * 4` bytes of free memory. This method checks the type of the argument `image` via `instanceof` and then calls the right `convertToAwtImage` method of this class.

– **Parameters**

* `image` – the `RGB24Image` to be converted

– **Returns** – newly-created AWT image instance

• *convertToAwtImage*

```
public static java.awt.Image convertToAwtImage(
    net.sourceforge.jiu.data.RGB24Image image, int alpha )
```

• *convertToAwtImage*

```
public static java.awt.Image convertToAwtImage(
    net.sourceforge.jiu.data.RGB48Image image, int alpha )
```

• *createImage*

```
public static java.awt.Image createImage( int[] pixels, int width, int
    height )
```

– **Description**

Creates a `Image` object from a pixel array. Internally, a `createImage` object is used to call its `createImage` method with a `createImage` object.

– **Parameters**

* `pixels` – the image pixel data in the typical RGBA 32-bit format, one int per pixel
 * `width` – the horizontal resolution in pixels of the image to be created
 * `height` – the vertical resolution in pixels of the image to be created

17.1.7 Class JiuAwtFrame

The frame class for the AWT demo program (in 1.2.6, page 47).

Declaration

```
public class JiuAwtFrame
  extends java.awt.Frame
  implements      java.awt.event.ActionListener,      java.awt.event.ComponentListener,
  net.sourceforge.jiu.apps.JiuInfo, net.sourceforge.jiu.ops.ProgressListener
```

Field summary

APP_NAME The name of this application, jiuawt, plus the version number taken from (in 1.1.1, page 14).

Constructor summary

JiuAwtFrame(EditorState) Create an object of this class, using the argument editor state.

Method summary

actionPerformed(ActionEvent) Processes event objects that get created when menu items are picked.

componentHidden(ComponentEvent)

componentMoved(ComponentEvent)

componentResized(ComponentEvent)

componentShown(ComponentEvent)

maximize() Maximize the frame on the desktop.

setDefaultCursor() Sets the current cursor to be .

setOriginalSize() If an image is currently loaded,

setProgress(float) Set a new progress status.

setProgress(int, int)

setStatusBar(String)

setWaitCursor()

showError(String) Displays the argument text in a message box with error in the title bar.

showInfo(String, String) Shows a modal dialog with given title bar and message text.

updateCanvas() If there is an image loaded, forces a canvas redraw by calling repaint.

updateImage() Removes the current canvas from the frame (if there is an image loaded) and creates a new canvas for the current image.

updateStatusBar() Creates a description string for the current image and sets the status bar to that text.

updateTitle() Sets the frame's title bar to the application name, plus the file name of the currently loaded image file, plus the current zoom factor, plus an optional asterisk in case the image was modified but not yet saved.

zoomIn() If an image is currently displayed, zoom in one level.

zoomOut() If an image is currently displayed, zoom out one level.

Serializable Fields

- private net.sourceforge.jiu.apps.EditorState **editor**
- private AwtMenuWrapper **menuWrapper**
- private AwtOperationProcessor **processor**
- private java.awt.Label **statusBar**
- private java.awt.ScrollPane **scrollPane**
- private ImageCanvas **canvas**

Fields

- public static final java.lang.String **APP_NAME**
 - The name of this application, jiuawt, plus the version number taken from (in 1.1.1, page 14). Example: jiuawt 0.8.0. Will be displayed in the title bar of this frame.

Constructors

- *JiuAwtFrame*
public JiuAwtFrame(net.sourceforge.jiu.apps.EditorState editorState)
 - **Description**
 Create an object of this class, using the argument editor state. String resources to initialize the menu etc. will be taken from the EditorState object's Strings variable
 - **Parameters**
 - * **editorState** – EditorState object used by this frame

Methods

- *actionPerformed*
public void actionPerformed(java.awt.event.ActionEvent e)
 - **Description**
 Processes event objects that get created when menu items are picked. Determines the (in 1.1.2, page 15) value for a given event object and calls the internal (in 17.1.3, page 425) object's process method with the menu value. The operation will then be performed.
 - **Parameters**
 - * **e** – the ActionEvent object
- *componentHidden*
void componentHidden(java.awt.event.ComponentEvent)

- *componentMoved*

void componentMoved(java.awt.event.ComponentEvent)
- *componentResized*

void componentResized(java.awt.event.ComponentEvent)
- *componentShown*

void componentShown(java.awt.event.ComponentEvent)
- *maximize*
public void maximize()
 - **Description**
Maximize the frame on the desktop. There is no such function in the 1.1 AWT (was added in 1.4), so this class determines the screen size and sets the frame to be a little smaller than that (to make up for task bars etc.). So this is just a heuristical approach.

- *setDefaultCursor*
public void setDefaultCursor()
 - **Description**
Sets the current cursor to be .

- *setOriginalSize*
public void setOriginalSize()
 - **Description**
If an image is currently loaded,

- *setProgress*
public void setProgress(float progress)
 - **Description**
Set a new progress status.
 - **Parameters**
 - * **progress** – float from 0.0f to 1.0f, indicating the progress between 0 and 100 percent

- *setProgress*
void setProgress(int zeroBasedIndex, int totalItems)
 - **Description copied from net.sourceforge.jiu.ops.ProgressListener (in 19.1.1, page 493)**
Sets a new progress level. If an operation consists of totalItems steps, which are numbered from 0 to totalItems - 1, this method can be called after the completion of each step.
Example: if there are three steps and the first one is done, the parameters must be 0 and 3, which will indicated 33% completion. Parameters 1 and 3 mean 66%, 2 and 3 100%. If you use 3 and 3, an IllegalArgumentException will be thrown.
Computes (float)(zeroBasedIndex + 1) / (float)totalItems and calls (in 19.1.1, page 493)with that value.
 - **Parameters**
 - * **zeroBasedIndex** – the index of the step that was just completed

- * `totalItems` – the number of steps in this operation
 - **Throws**
 - * `java.lang.IllegalArgumentException` – if the parameters don't match the above criteria
-
- *setStatusBar*
`public void setStatusBar(java.lang.String text)`
 - *setWaitCursor*
`public void setWaitCursor()`
 - *showError*
`public void showError(java.lang.String text)`
 - **Description**
 Displays the argument text in a message box with error in the title bar.
 - **Parameters**
 - * `text` – the error message to be displayed
-
- *showInfo*
`public void showInfo(java.lang.String title, java.lang.String text)`
 - **Description**
 Shows a modal dialog with given title bar and message text.
 - **Parameters**
 - * `title` – will be displayed in the dialog's title bar
 - * `text` – will be displayed in the dialog's center part
-
- *updateCanvas*
`public void updateCanvas()`
 - **Description**
 If there is an image loaded, forces a canvas redraw by calling repaint.
-
- *updateImage*
`public void updateImage()`
 - **Description**
 Removes the current canvas from the frame (if there is an image loaded) and creates a new canvas for the current image.
-
- *updateStatusBar*
`public void updateStatusBar()`
 - **Description**
 Creates a description string for the current image and sets the status bar to that text.
-
- *updateTitle*
`public void updateTitle()`
 - **Description**
 Sets the frame's title bar to the application name, plus the file name of the currently loaded image file, plus the current zoom factor, plus an optional asterisk in case the image was modified but not yet saved.

- *zoomIn*

`public void zoomIn()`

- **Description**

If an image is currently displayed, zoom in one level.

- *zoomOut*

`public void zoomOut()`

- **Description**

If an image is currently displayed, zoom out one level.

17.1.8 Class **RGBA**

This class converts between the 32 bit RGBA int values (used throughout the AWT) and various standard pixel formats like 24 bits RGB, 8 bits gray, 16 bits gray, 1 bit black and white. The conversion is done in a number of static methods. This class is not supposed to be instantiated.

The method names of this class can be interpreted as follows. If they contain **fromXYZ** (where XYZ is a pixel format type like Gray8, RGB24 etc.), a conversion from another pixel format to RGBA is done. If the names contains **toXYZ**, a conversion from RGBA to that pixel format will be performed.

Not all conversions are lossless or well-defined. If 48 bpp RGB truecolor is used as source, only the top eight bits of each 16 bit sample will be used (thus, the procedure is lossy). If RGBA data is to be converted to bilevel (black and white), the conversion is undefined if there are input RGBA pixels that are neither black nor white.

Declaration

```
public class RGBA
extends java.lang.Object
```

Field summary

DEFAULT_ALPHA The default value for the alpha part of RGBA.

Method summary

convertFromGray16(short[], int, int, int[], int, int) Convert a number of 16 bit grayscale pixels to RGBA type int pixels, adding the given alpha value.

convertFromGray8(byte[], int, int, int[], int, int) Convert a number of 8 bit grayscale pixels, shades of gray between 0 (for black) and 255 (for white), given as bytes, to RGBA type int pixels, adding the given alpha value.

convertFromPackedBilevel(byte[], int, int, int[], int, int) Converts pixels from bilevel packed bytes to RGBA format.

convertFromPaletted8(byte[], int, int, int[], int[], int[], int[], int, int) Converts a byte array of palette index values to an array of RGBA values, using palette color data.

convertFromRGB24(byte[], int, byte[], int, byte[], int, int, int[], int, int) Converts 24 bit RGB truecolor data to RGBA int values.

convertFromRGB48(short[], int, short[], int, short[], int, int, int[], int, int) Converts 48 bit RGB truecolor data to RGBA int values, dropping the least significant eight bits of each short sample.

Fields

- public static final int **DEFAULT_ALPHA**
 - The default value for the alpha part of RGBA. The alpha value is eight bits long left-shifted by 24. This default value is no transparency - the underlying image cannot be seen: 0xff000000.

Methods

- *convertFromGray16*

```
public static void convertFromGray16( short[] src, int srcOffset, int
alpha, int[] dest, int destOffset, int num )
```

- **Description**

Convert a number of 16 bit grayscale pixels to RGBA type int pixels, adding the given alpha value. Note that the lower 8 bits of each grayscale value are dropped.

- **Parameters**

- * **src** – array with grayscale pixels
 - * **srcOffset** – index of first entry of src to be converted
 - * **alpha** – transparency value to be used in resulting RGBA array (only top eight bits can be set)
 - * **dest** – array to store resulting RGBA pixels
 - * **destOffset** – index of first entry in dest to be used
 - * **num** – number of pixels to be converted
-

- *convertFromGray8*

```
public static void convertFromGray8( byte[] src, int srcOffset, int alpha,
int[] dest, int destOffset, int num )
```

- **Description**

Convert a number of 8 bit grayscale pixels, shades of gray between 0 (for black) and 255 (for white), given as bytes, to RGBA type int pixels, adding the given alpha value.

- **Parameters**

- * **src** – array with grayscale pixels
 - * **srcOffset** – index of first entry of src to be converted
 - * **alpha** – transparency value to be used in resulting RGBA array (only top eight bits can be set)
 - * **dest** – array to store resulting RGBA pixels
 - * **destOffset** – index of first entry in dest to be used
 - * **num** – number of pixels to be converted
-

- *convertFromPackedBilevel*

```
public static void convertFromPackedBilevel( byte[] src, int srcOffset, int
alpha, int[] dest, int destOffset, int num )
```

- **Description**

Converts pixels from bilevel packed bytes to RGBA format. A byte is supposed to store eight pixels, the most significant bit being the leftmost pixel.

- **Parameters**

- * **src** – the array with the packed bytes
 - * **srcOffset** – the index of the first byte to be converted from src
 - * **alpha** – the alpha value to be used for the destination RGBA values
 - * **dest** – the array where the destination RGBA pixels will be stored
 - * **destOffset** – the index of the first destination pixel in the dest array; that array must be at least destOffset + ((num + 7) / 8) large
 - * **num** – the number of pixels (not bytes) to be converted
-

- *convertFromPaletted8*

```
public static void convertFromPaletted8( byte[] src, int srcOffset, int
alpha, int[] red, int[] green, int[] blue, int[] dest, int destOffset, int
num )
```

- **Description**

Converts a byte array of palette index values to an array of RGBA values, using palette color data.

- **Parameters**

- * **src** – the byte array with the palette index values
 - * **srcOffset** – index of the first entry of src to be used
 - * **alpha** – transparency value to be used (only top eight bits should be set)
 - * **red** – the red palette values
 - * **green** – the green palette values
 - * **blue** – the blue palette values
 - * **dest** – the destination array to store the RGBA values
 - * **destOffset** – the first entry of dest to be used
 - * **num** – the number of pixels to be converted
-

- *convertFromRGB24*

```
public static void convertFromRGB24( byte[] srcRed, int srcRedOffset,
byte[] srcGreen, int srcGreenOffset, byte[] srcBlue, int srcBlueOffset, int
alpha, int[] dest, int destOffset, int num )
```

- **Description**

Converts 24 bit RGB truecolor data to RGBA int values.

- **Parameters**

- * **srcRed** – the red pixel values
 - * **srcRedOffset** – the first entry of srcRed to be used
 - * **srcGreen** – the green pixel values
 - * **srcGreenOffset** – the first entry of srcGreen to be used
 - * **srcBlue** – the blue pixel values
 - * **srcBlueOffset** – the first entry of srcBlue to be used
 - * **alpha** – the transparency value to be used in the destination RGBA array (only top 8 bits should be set)
 - * **dest** – array to store RGBA pixel values
 - * **destOffset** – first entry of dest to be used
 - * **num** – number of pixels to be converted
-

- *convertFromRGB48*

```
public static void convertFromRGB48( short[] srcRed, int srcRedOffset,
short[] srcGreen, int srcGreenOffset, short[] srcBlue, int srcBlueOffset,
int alpha, int[] dest, int destOffset, int num )
```

- **Description**

Converts 48 bit RGB truecolor data to RGBA int values, dropping the least significant eight bits of each short sample.

- **Parameters**

- * **srcRed** – the red pixel values
- * **srcRedOffset** – the first entry of srcRed to be used
- * **srcGreen** – the green pixel values
- * **srcGreenOffset** – the first entry of srcGreen to be used
- * **srcBlue** – the blue pixel values

- * **srcBlueOffset** – the first entry of srcBlue to be used
- * **alpha** – the transparency value to be used in the destination RGBA array (only top 8 bits should be set)
- * **dest** – array to store RGBA pixel values
- * **destOffset** – first entry of dest to be used
- * **num** – number of pixels to be converted

17.1.1.9 Class ToolkitLoader

This class loads an instance of `Image` using `Toolkit`'s built-in loading capabilities and converts it to `Image` (in 14.1.12, page 331) using `ToolkitLoader` (in 17.1.6, page 440).

Supported file formats are JPEG and GIF. PNG is supported since Java 1.3. I have heard that XBM are supposedly loaded as well. I don't know that format and haven't tested this functionality.

In addition, this class can also use JIU's built-in codecs from this class.

Usage examples

Load an image using Java's own `Image` class:

```
RGB24Image rgbImage = ToolkitLoader.loadAsRgb24Image("flower.jpg");
```

This will only load images from files in formats that are supported by `Toolkit` - normally, that only includes JPEG, GIF and since Java 1.3 PNG. A potential problem of this approach is that `Toolkit` always delivers RGB data, even if the image file only contains a black and white image. In order to get an image object of the "real" type, try JIU's `Image` (in 13.1.1, page 296) with `rgbImage` (if you follow the link you will get a usage example for that class as well).

Known issues

If you are using this class to load JPEGs, GIFs or PNGs, an AWT background thread is started (as for a normal AWT GUI application). Before Java 1.4 there was a bug that kept the thread running although an application had reached the end of its execution (by getting to the end of the `main(String[])` method). If you experience this problem, either update to a 1.4+ JDK or follow the advice given at **kguru.com** (at <http://www.jguru.com/faq/view.jsp?EID=467061>) and call `System.exit(0);`.

Declaration

```
public class ToolkitLoader
    extends java.lang.Object
```

Method summary

- load(String)** Loads an image from a file using the AWT's built-in loader.
- loadAsRgb24Image(String)** Loads an image from a file using the AWT's built-in loader and converts the image to a `Image` (in 14.1.12, page 331) object.
- loadViaToolkitOrCodecs(String)** Attempts to load an image from a file given by its name, using both the JIU codecs and the image loading functionality in `java.awt.Toolkit`.
- loadViaToolkitOrCodecs(String, boolean, Vector)** Attempts to load an image from a file given by its name, using both the JIU codecs and the image loading functionality in `java.awt.Toolkit`.

Methods

- *load*

```
public static java.awt.Image load( java.lang.String fileName )
```

– **Description**

Loads an image from a file using the AWT’s built-in loader. Returns that image as an AWT object. This method does nothing more than call `Toolkit.getDefaultImageInputStream`, wait for it using `waitForImage` and return the resulting image.

– **Parameters**

* `fileName` – name of the image file

– **Returns** – the image as AWT image object

• *loadAsRgb24Image*

```
public static net.sourceforge.jiu.data.RGB24Image loadAsRgb24Image(
    java.lang.String fileName )
```

– **Description**

Loads an image from a file using the AWT’s built-in loader and converts the image to a `net.sourceforge.jiu.data.RGB24Image` object. First calls `loadImage` (in 17.1.9, page 452) with the filename, then converts the loaded image using `convertToRGB24` (in 17.1.6, page 440).

– **Parameters**

* `fileName` – name of the file from which the image is to be loaded

– **Returns** – loaded image as `net.sourceforge.jiu.data.RGB24Image` (in 14.1.12, page 331)

• *loadViaToolkitOrCodecs*

```
public static net.sourceforge.jiu.data.PixelImage loadViaToolkitOrCodecs(
    java.lang.String fileName )
```

– **Description**

Attempts to load an image from a file given by its name, using both the JIU codecs and the image loading functionality in `java.awt.Toolkit`. First tries JIU’s codecs, then `java.awt.Toolkit`. Simply calls `loadViaToolkitOrCodecs(fileName, false);`.

– **Parameters**

* `fileName` – name of the image file

– **Returns** – image object or null on failure

• *loadViaToolkitOrCodecs*

```
public static net.sourceforge.jiu.data.PixelImage loadViaToolkitOrCodecs(
    java.lang.String fileName, boolean preferToolkit, java.util.Vector
    progressListeners )
```

– **Description**

Attempts to load an image from a file given by its name, using both the JIU codecs and the image loading functionality in `java.awt.Toolkit`. The second argument determines which method is tried first, `Toolkit` (true) or the JIU codecs (false). Uses `loadImage` (in 17.1.9, page 453) from this class for `Toolkit` loading and `loadImageViaCodecs` (in 2.1.6, page 92) for JIU’s codecs.

– **Parameters**

* `fileName` – name of the image file

– **Returns** – image object or null on failure

Chapter 18

Package

net.sourceforge.jiu.gui.awt.dialogs

Package Contents

Page

Classes

CropDialog	456
<i>A dialog to enter the parameters for an image crop operation.</i>	
Dialogs	459
<i>Convenience class that provides a number of static helper methods to deal with dialogs.</i>	
GammaCorrectionDialog	460
<i>A dialog to enter the parameters for a gamma correction operation.</i>	
HueSaturationValueDialog	462
<i>A dialog to enter the parameters for an hue/saturation/value adjustment operation.</i>	
InfoDialog	464
<i>A modal AWT dialog that displays text in a non-editable text area component (so that it can be selected and easily copied to the system's clipboard).</i>	
IntegerDialog	466
<i>An AWT dialog to select an <code>int</code> value from a given interval.</i>	
MapToArbitraryPaletteDialog	468
<i>A dialog to enter the parameters for an operation to map an RGB truecolor image to any given palette.</i>	
MedianCutDialog	471
<i>A dialog to enter the parameters for a Median Cut color quantization operation.</i>	
OctreeDialog	474
<i>A dialog to enter the parameters for an Octree color quantization operation.</i>	
ReduceGrayscaleDialog	477
<i>A dialog to enter the parameters for a grayscale reduction operation.</i>	
ScaleDialog	480
<i>A dialog to enter the parameters for an image scaling operation.</i>	
ShearDialog	483
<i>An AWT dialog to enter the angle for a shearing operation.</i>	
UniformPaletteQuantizerDialog	485
<i>An AWT dialog to enter the parameters for a uniform palette color quantization operation.</i>	

WindowSizeDialog	488
<i>A dialog to enter values for the width and height of a window (typically for a spatial filter like median or mean).</i>	
YesNoDialog	490
<i>A dialog that asks a question and offers a Yes and a No button (and optionally a Cancel button).</i>	

AWT dialogs that are used in the jiuawt application.

18.1 Classes

18.1.1 *Class* CropDialog

A dialog to enter the parameters for an image crop operation.

Declaration

```
public class CropDialog
extends java.awt.Dialog
implements java.awt.event.ActionListener, java.awt.event.KeyListener
```

Constructor summary

CropDialog(Frame, Strings, int, int)

Method summary

actionPerformed(ActionEvent) Hides (closes) this dialog if the OK button was source of the action event (e.g. if the button was pressed).

getHeight()

getWidth() Returns the width of the to-be-cropped image as given by the current values in the text fields for left column and right column.

getX1()

getX2()

getY1()

getY2()

hasPressedOk()

keyPressed(KeyEvent)

keyReleased(KeyEvent)

keyTyped(KeyEvent)

Serializable Fields

- private java.awt.Button **ok**
- private java.awt.Button **cancel**
- private java.awt.TextField **x1**
- private java.awt.TextField **y1**
- private java.awt.TextField **x2**
- private java.awt.TextField **y2**
- private int **width**
- private int **height**
- private java.awt.Label **newWidth**
- private java.awt.Label **newHeight**
- private boolean **pressedOk**

Constructors

- *CropDialog*

```
public CropDialog( java.awt.Frame owner, net.sourceforge.jiu.apps.Strings
strings, int width, int height )
```

- **Parameters**

- * **owner** – the Frame this dialog will belong to
- * **strings** – Strings resource to be used for text messages
- * **width** – width of the original image, before cropping; this is used to determine the valid values for left and right column, from 0 to width - 1
- * **height** – height of the original image, before cropping; this is used to determine the valid values for top and bottom row, from 0 to height - 1

Methods

- *actionPerformed*

```
public void actionPerformed( java.awt.event.ActionEvent e )
```

- **Description**

Hides (closes) this dialog if the OK button was source of the action event (e.g. if the button was pressed).

- *getHeight*

```
public int getHeight( )
```

- *getWidth*

```
public int getWidth( )
```

- **Description**

Returns the width of the to-be-cropped image as given by the current values in the text fields for left column and right column. Computes the width from those values and returns it or returns -1 if the data in the text fields is not valid for some reason.

- **Returns** – width of cropped image or -1 if information is invalid

- *getX1*

```
public int getX1( )
```

- *getX2*

```
public int getX2( )
```

- *getY1*

```
public int getY1( )
```

- *getY2*

```
public int getY2( )
```

- *hasPressedOk*

```
public boolean hasPressedOk( )
```

- *keyPressed*

```
void keyPressed( java.awt.event.KeyEvent )
```

- *keyReleased*
void keyReleased(java.awt.event.KeyEvent)
- *keyTyped*
void keyTyped(java.awt.event.KeyEvent)

18.1.2 Class Dialogs

Convenience class that provides a number of static helper methods to deal with dialogs.

Declaration

```
public class Dialogs
  extends java.lang.Object
```

Method summary

center(Window) Centers the argument window on screen.

getInteger(Frame, String, String, int, int, int, String, String) Creates a new IntegerDialog, displays it and returns the Integer value specified by the user (or null if the dialog was canceled).

Methods

- *center*

```
public static void center( java.awt.Window window )
```

- **Description**

Centers the argument window on screen.

- *getInteger*

```
public static java.lang.Integer getInteger( java.awt.Frame owner,
  java.lang.String title, java.lang.String message, int minValue, int
  initialValue, int maxValue, java.lang.String okText, java.lang.String
  cancelText )
```

- **Description**

Creates a new IntegerDialog, displays it and returns the Integer value specified by the user (or null if the dialog was canceled).

- **Parameters**

- * **owner** – frame from which the dialog is spawned
- * **title** – text for the title bar of the dialog
- * **message** – message displayed in the dialog
- * **minValue** – minimal allowed integer value to be entered by the user
- * **initialValue** – initial integer value shown in the dialog
- * **maxValue** – maximal allowed integer value to be entered by the user
- * **okText** – the text for the OK button
- * **cancelText** – the text for the cancel button

- **Returns** – the specified integer value or null if the Cancel button was pressed

18.1.3 Class GammaCorrectionDialog

A dialog to enter the parameters for a gamma correction operation.

Declaration

```
public class GammaCorrectionDialog
extends java.awt.Dialog
implements java.awt.event.ActionListener, java.awt.event.KeyListener
```

Constructor summary

GammaCorrectionDialog(Frame, Strings, double, double) Creates a GammaCorrectionDialog.

Method summary

actionPerformed(ActionEvent) Hides (closes) this dialog if the OK button was source of the action event (e.g. if the button was pressed).
getValue()
handleKeys(KeyEvent)
hasPressedOk()
keyPressed(KeyEvent)
keyReleased(KeyEvent)
keyTyped(KeyEvent)

Serializable Fields

- private java.awt.Button **ok**
- private java.awt.Button **cancel**
- private java.awt.TextField **gammaTextField**
- private boolean **pressedOk**
- private java.lang.Double **result**
- private final double **MAX_GAMMA**

Constructors

- *GammaCorrectionDialog*
 public **GammaCorrectionDialog**(java.awt.Frame owner,
 net.sourceforge.jiu.apps.Strings strings, double initialValue, double
 maxGamma)
 - **Description**
 Creates a GammaCorrectionDialog.
 - **Parameters**

- * **owner** – the Frame this dialog will belong to
- * **strings** – Strings resource used for text messages
- * **initialValue** – the value to be set when the dialog pops up
- * **maxGamma** – the maximum allowed gamma value

Methods

- *actionPerformed*

`public void actionPerformed(java.awt.event.ActionEvent e)`

- **Description**

Hides (closes) this dialog if the OK button was source of the action event (e.g. if the button was pressed).

- *getValue*

`public java.lang.Double getValue()`

- *handleKeys*

`public void handleKeys(java.awt.event.KeyEvent e)`

- *hasPressedOk*

`public boolean hasPressedOk()`

- *keyPressed*

`void keyPressed(java.awt.event.KeyEvent)`

- *keyReleased*

`void keyReleased(java.awt.event.KeyEvent)`

- *keyTyped*

`void keyTyped(java.awt.event.KeyEvent)`

18.1.4 Class HueSaturationValueDialog

A dialog to enter the parameters for an hue/saturation/value adjustment operation. Saturation and value are specified as percentage values between -100 and 100, where 0 means no change. Hue can be specified optionally (a Choice component must be checked so that the hue value will be used); it is a value between 0 and 359.

See also

- `net.sourceforge.jiu.color.adjustment.HueSaturationValue` (in 6.1.5, page 206)

Declaration

```
public class HueSaturationValueDialog
extends java.awt.Dialog
implements java.awt.event.ActionListener, java.awt.event.ItemListener, java.awt.event.KeyListener
```

Constructor summary

HueSaturationValueDialog(Frame, Strings, boolean, int, int, int)

Method summary

actionPerformed(ActionEvent) Hides (closes) this dialog if the OK button was source of the action event (e.g. if the button was pressed).

getHue()

getSaturation()

getValue()

hasPressedOk()

isHueSet()

itemStateChanged(ItemEvent)

keyPressed(KeyEvent)

keyReleased(KeyEvent)

keyTyped(KeyEvent)

Serializable Fields

- private java.awt.Button **ok**
- private java.awt.Button **cancel**
- private java.awt.Panel **colorPanel**
- private java.awt.TextField **hue**
- private java.awt.TextField **saturation**
- private java.awt.TextField **value**
- private java.awt.Checkbox **setHue**
- private boolean **pressedOk**

Constructors

- *HueSaturationValueDialog*
`public HueSaturationValueDialog(java.awt.Frame owner,
net.sourceforge.jiu.apps.Strings strings, boolean initialSetHue, int h, int
s, int v)`
 - **Parameters**
 - * **owner** – the Frame this dialog will belong to

Methods

- *actionPerformed*
`public void actionPerformed(java.awt.event.ActionEvent e)`
 - **Description**
Hides (closes) this dialog if the OK button was source of the action event (e.g. if the button was pressed).
- *getHue*
`public int getHue()`
- *getSaturation*
`public int getSaturation()`
- *getValue*
`public int getValue()`
- *hasPressedOk*
`public boolean hasPressedOk()`
- *isHueSet*
`public boolean isHueSet()`
- *itemStateChanged*
`void itemStateChanged(java.awt.event.ItemEvent)`
- *keyPressed*
`void keyPressed(java.awt.event.KeyEvent)`
- *keyReleased*
`void keyReleased(java.awt.event.KeyEvent)`
- *keyTyped*
`void keyTyped(java.awt.event.KeyEvent)`

18.1.5 Class InfoDialog

A modal AWT dialog that displays text in a non-editable text area component (so that it can be selected and easily copied to the system's clipboard). Provides an OK button so that user can remove the dialog.

Declaration

```
public class InfoDialog
extends java.awt.Dialog
implements java.awt.event.ActionListener
```

Constructor summary

InfoDialog(Frame, String, String) Creates an InfoDialog, a modal dialog to display a text message, centered on the desktop.

Method summary

actionPerformed(ActionEvent) Hides (closes) this dialog if the OK button was source of the action event (e.g. if the button was pressed).
center() Centers the dialog on screen.

Serializable Fields

- private java.awt.Button **ok**
- private java.awt.TextArea **textArea**

Constructors

- *InfoDialog*
public InfoDialog(java.awt.Frame owner, java.lang.String title, java.lang.String text)
 - **Description**
Creates an InfoDialog, a modal dialog to display a text message, centered on the desktop.
 - **Parameters**
 - * **owner** – the Frame this dialog will belong to
 - * **title** – the text that will be displayed in the title bar of the dialog
 - * **text** – the message text that will be displayed in the main part of the dialog

Methods

- *actionPerformed*
public void actionPerformed(java.awt.event.ActionEvent e)

- **Description**

Hides (closes) this dialog if the OK button was source of the action event (e.g. if the button was pressed).

- *center*

```
public void center( )
```

- **Description**

Centers the dialog on screen.

18.1.6 Class IntegerDialog

An AWT dialog to select an `int` value from a given interval.

Declaration

```
public class IntegerDialog
extends java.awt.Dialog
implements java.awt.event.ActionListener, java.awt.event.AdjustmentListener,
java.awt.event.KeyListener
```

Constructor summary

IntegerDialog(Frame, String, String, int, int, int, String, String) Creates an IntegerDialog, a modal dialog that lets the user select one `int` value from a given interval.

Method summary

actionPerformed(ActionEvent) Hides (closes) this dialog if the OK button was source of the action event (e.g. if the button was pressed).
adjustmentValueChanged(AdjustmentEvent)
getValue()
handleKeys(KeyEvent)
keyPressed(KeyEvent)
keyReleased(KeyEvent)
keyTyped(KeyEvent)

Serializable Fields

- private java.awt.Button **cancel**
- private int **maxValue**
- private int **minValue**
- private java.awt.Button **ok**
- private boolean **pressedOk**
- private java.lang.Integer **result**
- private java.awt.Scrollbar **scrollbar**
- private java.awt.TextComponent **valueTextField**

Constructors

- *IntegerDialog*
 public **IntegerDialog**(java.awt.Frame owner, java.lang.String title, java.lang.String message, int minValue, int initialValue, int maxValue, java.lang.String okText, java.lang.String cancelText)

– **Description**

Creates an IntegerDialog, a modal dialog that lets the user select one int value from a given interval.

– **Parameters**

- * **owner** – the this dialog will belong to
- * **title** – the text that will be displayed in the title bar of the dialog
- * **message** – the message text that will be displayed in the upper part of the dialog
- * **minValue** – the minimum allowed int value to be selected by the user
- * **initialValue** – the int value that is selected when the dialog first pops up
- * **maxValue** – the maximum allowed int value to be selected by the user
- * **okText** – the value for OK (just "OK" in English programs, may be different for other natural languages)
- * **cancelText** – the value for Cancel (just "Cancel" in English programs, may be different for other natural languages)

Methods

- *actionPerformed*

```
public void actionPerformed( java.awt.event.ActionEvent e )
```

– **Description**

Hides (closes) this dialog if the OK button was source of the action event (e.g. if the button was pressed).

- *adjustmentValueChanged*

```
void adjustmentValueChanged( java.awt.event.AdjustmentEvent )
```

- *getValue*

```
public java.lang.Integer getValue( )
```

- *handleKeys*

```
public void handleKeys( java.awt.event.KeyEvent e )
```

- *keyPressed*

```
void keyPressed( java.awt.event.KeyEvent )
```

- *keyReleased*

```
void keyReleased( java.awt.event.KeyEvent )
```

- *keyTyped*

```
void keyTyped( java.awt.event.KeyEvent )
```

18.1.7 Class MapToArbitraryPaletteDialog

A dialog to enter the parameters for an operation to map an RGB truecolor image to any given palette.

See also

- `net.sourceforge.jiu.color.quantization.ArbitraryPaletteQuantizer` (in 12.2.1, page 261)

Declaration

```
public class MapToArbitraryPaletteDialog
    extends java.awt.Dialog
    implements java.awt.event.ActionListener
```

Field summary

NUM_PALETTE_TYPES
PALETTE_FILE
PALETTE_PALM_16_COLORS
PALETTE_PALM_16_GRAY
PALETTE_PALM_256_COLORS
PALETTE_PALM_4_GRAY
PALETTE_WEBSAFE

Constructor summary

MapToArbitraryPaletteDialog(Frame, Strings)

Method summary

actionPerformed(ActionEvent) Hides (closes) this dialog if the OK button was source of the action event (e.g. if the button was pressed).
getErrorDiffusionType() If the use of error diffusion was selected, this method returns one of the ErrorDiffusionDithering TYPE constants
getPaletteType() Return the palette type (one of the PALETTE_xyz constants of this class) that is currently selected in the dialog.
hasPressedOk() Returns true if the OK button was pressed, false if it was the Cancel button.
useErrorDiffusion() Returns whether the use of one of the error diffusion algorithms is selected in the dialog.

Serializable Fields

- private java.awt.Button **ok**
- private java.awt.Button **cancel**
- private java.awt.Checkbox **checkboxes**

- private java.awt.CheckboxGroup **paletteType**
- private java.awt.Choice **dithering**
- private boolean **pressedOk**

Fields

- public static final int **PALETTE_FILE**
- public static final int **PALETTE_WEBSAFE**
- public static final int **PALETTE_PALM_256_COLORS**
- public static final int **PALETTE_PALM_16_COLORS**
- public static final int **PALETTE_PALM_16_GRAY**
- public static final int **PALETTE_PALM_4_GRAY**
- public static final int **NUM_PALETTE_TYPES**

Constructors

- *MapToArbitraryPaletteDialog*
 public **MapToArbitraryPaletteDialog**(java.awt.Frame owner,
 net.sourceforge.jiu.apps.Strings strings)
 – **Parameters**
 * **owner** – the Frame this dialog will belong to

Methods

- *actionPerformed*
 public void **actionPerformed**(java.awt.event.ActionEvent e)
 – **Description**
 Hides (closes) this dialog if the OK button was source of the action event (e.g. if the button was pressed).
- *getErrorDiffusionType*
 public int **getErrorDiffusionType**()
 – **Description**
 If the use of error diffusion was selected, this method returns one of the ErrorDiffusionDithering TYPE constants
- *getPaletteType*
 public int **getPaletteType**()

– **Description**

Return the palette type (one of the PALETTE_xyz constants of this class) that is currently selected in the dialog.

• *hasPressedOk*

`public boolean hasPressedOk()`

– **Description**

Returns true if the OK button was pressed, false if it was the Cancel button.

• *useErrorDiffusion*

`public boolean useErrorDiffusion()`

– **Description**

Returns whether the use of one of the error diffusion algorithms is selected in the dialog.

18.1.8 Class MedianCutDialog

A dialog to enter the parameters for a Median Cut color quantization operation. It also allows to enter the optional algorithms that can be applied in combination with Median Cut.

Declaration

```
public class MedianCutDialog
  extends java.awt.Dialog
  implements java.awt.event.ActionListener, java.awt.event.ItemListener, java.awt.event.KeyListener
```

Field summary

ERROR_DIFFUSION_STRINGS
ERROR_DIFFUSION_TYPES
METHODS

Constructor summary

MedianCutDialog(Frame, Strings, int, int, boolean, int, double) Creates a modal dialog to enter the parameter.

Method summary

actionPerformed(ActionEvent) Hides (closes) this dialog if the OK button was source of the action event (e.g. if the button was pressed).
center() Centers the dialog on screen.
getErrorDiffusion()
getNumColors()
getNumPasses()
getReprColorMethod()
getTau()
hasPressedOk()
isOutputTypePaletted()
itemStateChanged(ItemEvent)
keyPressed(KeyEvent)
keyReleased(KeyEvent)
keyTyped(KeyEvent)
useContourRemoval()
useErrorDiffusion()

Serializable Fields

- public final int **METHODS**
- public final int **ERROR_DIFFUSION_STRINGS**
- public final int **ERROR_DIFFUSION_TYPES**
- private java.awt.Button **ok**

- private java.awt.Button **cancel**
- private java.awt.TextField **numColorsField**
- private java.awt.Choice **outputColorType**
- private java.awt.Choice **reprColorMethod**
- private java.awt.Choice **algorithms**
- private java.awt.Choice **errorDiffusion**
- private java.awt.TextField **numPassesField**
- private java.awt.TextField **tauField**
- private boolean **pressedOk**

Fields

- public final int **METHODS**
- public final int **ERROR_DIFFUSION_STRINGS**
- public final int **ERROR_DIFFUSION_TYPES**

Constructors

- *MedianCutDialog*

```
public MedianCutDialog( java.awt.Frame owner,
net.sourceforge.jiu.apps.Strings strings, int numColors, int
representativeColorMethod, boolean paletted, int numPasses, double
initialTau )
```

 - **Description**
Creates a modal dialog to enter the parameter.
 - **Parameters**
 - * **owner** – the parent of this modal dialog
 - * **strings** – an object to get String constants in the current language
 - * **numColors** – the number of colors in the resulting image
 - * **representativeColorMethod** – the method to determine the representative color from a set of colors
 - * **paletted** – if true, the output image will be paletted, otherwise truecolor
 - * **numPasses** – number of contour removal iterations
 - * **initialTau** – maximum distance for two colors to be considered similar in contour removal

Methods

- *actionPerformed*
 public void **actionPerformed**(java.awt.event.ActionEvent e)
 – **Description**
 Hides (closes) this dialog if the OK button was source of the action event (e.g. if the button was pressed).

- *center*
 public void **center**()
 – **Description**
 Centers the dialog on screen.

- *getErrorDiffusion*
 public int **getErrorDiffusion**()

- *getNumColors*
 public int **getNumColors**()

- *getNumPasses*
 public int **getNumPasses**()

- *getReprColorMethod*
 public int **getReprColorMethod**()

- *getTau*
 public double **getTau**()

- *hasPressedOk*
 public boolean **hasPressedOk**()

- *isOutputTypePaletted*
 public boolean **isOutputTypePaletted**()

- *itemStateChanged*
 void **itemStateChanged**(java.awt.event.ItemEvent)

- *keyPressed*
 void **keyPressed**(java.awt.event.KeyEvent)

- *keyReleased*
 void **keyReleased**(java.awt.event.KeyEvent)

- *keyTyped*
 void **keyTyped**(java.awt.event.KeyEvent)

- *useContourRemoval*
 public boolean **useContourRemoval**()

- *useErrorDiffusion*
 public boolean **useErrorDiffusion**()

18.1.9 Class OctreeDialog

A dialog to enter the parameters for an Octree color quantization operation. It also allows to enter the optional algorithms that can be applied in combination with Octree.

See also

- `MedianCutDialog` (in 18.1.8, page 471)

Declaration

```
public class OctreeDialog
  extends java.awt.Dialog
  implements java.awt.event.ActionListener, java.awt.event.KeyListener
```

Field summary

DITHERING_STRINGS
DITHERING_TYPES

Constructor summary

OctreeDialog(Frame, Strings, int, boolean) Creates a modal dialog to enter the parameter.

Method summary

actionPerformed(ActionEvent) Hides (closes) this dialog if the OK button was source of the action event (e.g. if the button was pressed).
center() Centers the dialog on screen.
getErrorDiffusion()
getNumColors()
hasPressedOk()
isOutputTypePaletted()
keyPressed(KeyEvent)
keyReleased(KeyEvent)
keyTyped(KeyEvent)
useErrorDiffusion()
useNoDithering()

Serializable Fields

- public final int **DITHERING_STRINGS**
- public final int **DITHERING_TYPES**
- private java.awt.Button **ok**
- private java.awt.Button **cancel**
- private java.awt.TextField **numColorsField**

- private java.awt.Choice **outputColorType**
- private java.awt.Choice **dithering**
- private boolean **pressedOk**

Fields

- public final int **DITHERING_STRINGS**
- public final int **DITHERING_TYPES**

Constructors

- *OctreeDialog*
 public **OctreeDialog**(java.awt.Frame owner, net.sourceforge.jiu.apps.Strings strings, int numColors, boolean paletted)
 - **Description**
Creates a modal dialog to enter the parameter.
 - **Parameters**
 - * **owner** – the parent of this modal dialog
 - * **strings** – an object to get String constants in the current language
 - * **numColors** – the number of colors in the resulting image
 - * **paletted** – if true, the output image will be paletted, otherwise truecolor

Methods

- *actionPerformed*
 public void **actionPerformed**(java.awt.event.ActionEvent e)
 - **Description**
Hides (closes) this dialog if the OK button was source of the action event (e.g. if the button was pressed).
- *center*
 public void **center**()
 - **Description**
Centers the dialog on screen.
- *getErrorDiffusion*
 public int **getErrorDiffusion**()
- *getNumColors*
 public int **getNumColors**()
- *hasPressedOk*
 public boolean **hasPressedOk**()

- *isOutputTypePaletted*
public boolean isOutputTypePaletted()
- *keyPressed*
void keyPressed(java.awt.event.KeyEvent)
- *keyReleased*
void keyReleased(java.awt.event.KeyEvent)
- *keyTyped*
void keyTyped(java.awt.event.KeyEvent)
- *useErrorDiffusion*
public boolean useErrorDiffusion()
- *useNoDithering*
public boolean useNoDithering()

18.1.10 Class ReduceGrayscaleDialog

A dialog to enter the parameters for a grayscale reduction operation.

Declaration

```
public class ReduceGrayscaleDialog
extends java.awt.Dialog
implements java.awt.event.ActionListener, java.awt.event.AdjustmentListener
```

Field summary

DITHERING_METHODS
TYPE_BURKES_ERROR_DIFFUSION
TYPE_DITHERING_NONE
TYPE_FLOYD_STEINBERG_ERROR_DIFFUSION
TYPE_JARVIS_JUDICE_NINKE_ERROR_DIFFUSION
TYPE_ORDERED_DITHERING
TYPE_SIERRA_ERROR_DIFFUSION
TYPE_STEVENSON_ARCE_ERROR_DIFFUSION
TYPE_STUCKI_ERROR_DIFFUSION

Constructor summary

ReduceGrayscaleDialog(Frame, Strings, int, int, int) Creates a modal dialog to enter the parameters.

Method summary

actionPerformed(ActionEvent) Hides (closes) this dialog if the OK button was source of the action event (e.g. if the button was pressed).
adjustmentValueChanged(AdjustmentEvent)
center() Centers the dialog on screen.
getDitheringMethod()
getNumBits()
hasPressedOk()

Serializable Fields

- public final int **DITHERING_METHODS**
- private net.sourceforge.jiu.apps.Strings **strings**
- private java.awt.Button **ok**
- private java.awt.Button **cancel**
- private java.awt.Scrollbar **scrollbar**
- private java.awt.Choice **ditheringMethod**
- private java.awt.Label **bitLabel**

- private java.awt.Label **shadesLabel**
- private boolean **pressedOk**

Fields

- public static final int **TYPE_DITHERING_NONE**
- public static final int **TYPE_ORDERED_DITHERING**
- public static final int **TYPE_FLOYD_STEINBERG_ERROR_DIFFUSION**
- public static final int **TYPE_STUCKI_ERROR_DIFFUSION**
- public static final int **TYPE_BURKES_ERROR_DIFFUSION**
- public static final int **TYPE_SIERRA_ERROR_DIFFUSION**
- public static final int **TYPE_JARVIS_JUDICE_NINKE_ERROR_DIFFUSION**
- public static final int **TYPE_STEVENSON_ARCE_ERROR_DIFFUSION**
- public final int **DITHERING_METHODS**

Constructors

- *ReduceGrayscaleDialog*
 public **ReduceGrayscaleDialog**(java.awt.Frame owner,
 net.sourceforge.jiu.apps.Strings strings, int bits, int maxBits, int
 ditheringMethodSelection)
 - **Description**
 Creates a modal dialog to enter the parameters.
 - **Parameters**
 - * **owner** – the parent of this modal dialog
 - * **strings** – an object to get String constants in the current language
 - * **bits** – initial number of bits to be shown in the dialog
 - * **maxBits** – maximum allowed number of bits
 - * **ditheringMethodSelection** – initial selection of dithering method

Methods

- *actionPerformed*
 public void **actionPerformed**(java.awt.event.ActionEvent e)
 - **Description**
 Hides (closes) this dialog if the OK button was source of the action event (e.g. if the button was pressed).
- *adjustmentValueChanged*
 void **adjustmentValueChanged**(java.awt.event.AdjustmentEvent e)

- *center*
`public void center()`

– **Description**
Centers the dialog on screen.

- *getDitheringMethod*
`public int getDitheringMethod()`

- *getNumBits*
`public int getNumBits()`

- *hasPressedOk*
`public boolean hasPressedOk()`

18.1.11 Class ScaleDialog

A dialog to enter the parameters for an image scaling operation.

Declaration

```
public class ScaleDialog
  extends java.awt.Dialog
  implements java.awt.event.ActionListener, java.awt.event.KeyListener
```

Constructor summary

ScaleDialog(Frame, Strings, int, int, boolean, String[], int) Creates an InfoDialog, a modal dialog to display a text message, centered on the desktop.

Method summary

actionPerformed(ActionEvent) Hides (closes) this dialog if the OK button was source of the action event (e.g. if the button was pressed).
center() Centers the dialog on screen.
getHeightValue()
getType()
getWidthValue()
handleKeys(KeyEvent)
hasPressedOk()
keyPressed(KeyEvent)
keyReleased(KeyEvent)
keyTyped(KeyEvent)

Serializable Fields

- private java.awt.Button **ok**
- private java.awt.Button **cancel**
- private java.awt.TextComponent **heightTextField**
- private java.awt.TextComponent **widthTextField**
- private java.awt.Checkbox **maintainAspectRatio**
- private java.awt.Choice **types**
- private boolean **pressedOk**
- private java.lang.String **oldWidthString**
- private java.lang.String **oldHeightString**
- private int **oldWidth**
- private int **oldHeight**
- private int **type**

Constructors

- *ScaleDialog*

```
public ScaleDialog( java.awt.Frame owner, net.sourceforge.jiu.apps.Strings
strings, int width, int height, boolean pickType, java.lang.String[]
typeNameNames, int initialType )
```

- **Description**

Creates an InfoDialog, a modal dialog to display a text message, centered on the desktop.

- **Parameters**

- * **owner** – the Frame this dialog will belong to
- * **strings** – the Strings resource used for text messages
- * **width** – the current width of the image
- * **height** – the current height of the image
- * **pickType** – determines whether there will be a Choice box for picking the type of scaling algorithm
- * **typeNameNames** – names of the image scaling algorithms
- * **initialType** – algorithm type to be initially selected

Methods

- *actionPerformed*

```
public void actionPerformed( java.awt.event.ActionEvent e )
```

- **Description**

Hides (closes) this dialog if the OK button was source of the action event (e.g. if the button was pressed).

- *center*

```
public void center( )
```

- **Description**

Centers the dialog on screen.

- *getHeightValue*

```
public int getHeightValue( )
```

- *getType*

```
public int getType( )
```

- *getWidthValue*

```
public int getWidthValue( )
```

- *handleKeys*

```
public void handleKeys( java.awt.event.KeyEvent e )
```

- *hasPressedOk*

```
public boolean hasPressedOk( )
```

- *keyPressed*

```
void keyPressed( java.awt.event.KeyEvent e )
```

- *keyReleased*
void keyReleased(java.awt.event.KeyEvent)
- *keyTyped*
void keyTyped(java.awt.event.KeyEvent)

18.1.12 Class ShearDialog

An AWT dialog to enter the angle for a shearing operation.

Declaration

```
public class ShearDialog
  extends java.awt.Dialog
  implements java.awt.event.ActionListener, java.awt.event.KeyListener
```

Constructor summary

ShearDialog(Frame, Strings, double, int, int) Creates a ShearDialog.

Method summary

actionPerformed(ActionEvent) Hides (closes) this dialog if the OK button was source of the action event (e.g. if the button was pressed).

getValue()

handleKeys(KeyEvent)

hasPressedOk()

keyPressed(KeyEvent)

keyReleased(KeyEvent)

keyTyped(KeyEvent)

Serializable Fields

- private java.awt.Button **ok**
- private java.awt.Button **cancel**
- private java.awt.TextField **angleTextField**
- private boolean **pressedOk**
- private java.lang.Double **result**
- private java.awt.Label **newWidthLabel**
- private int **imageWidth**
- private int **imageHeight**

Constructors

- *ShearDialog*
 public **ShearDialog**(java.awt.Frame owner, net.sourceforge.jiu.apps.Strings strings, double initialValue, int imageWidth, int imageHeight)
 - **Description**
Creates a ShearDialog.
 - **Parameters**
 - * **owner** – the Frame this dialog will belong to

Methods

- *actionPerformed*
`public void actionPerformed(java.awt.event.ActionEvent e)`
 - **Description**
Hides (closes) this dialog if the OK button was source of the action event (e.g. if the button was pressed).
- *getValue*
`public java.lang.Double getValue()`
- *handleKeys*
`public void handleKeys(java.awt.event.KeyEvent e)`
- *hasPressedOk*
`public boolean hasPressedOk()`
- *keyPressed*
`void keyPressed(java.awt.event.KeyEvent)`
- *keyReleased*
`void keyReleased(java.awt.event.KeyEvent)`
- *keyTyped*
`void keyTyped(java.awt.event.KeyEvent)`

18.1.13 Class UniformPaletteQuantizerDialog

An AWT dialog to enter the parameters for a uniform palette color quantization operation.

Declaration

```
public class UniformPaletteQuantizerDialog
extends java.awt.Dialog
implements java.awt.event.ActionListener, java.awt.event.AdjustmentListener,
java.awt.event.ItemListener
```

Field summary

DITHERING_METHODS
TYPE_BURKES_ERROR_DIFFUSION
TYPE_DITHERING_NONE
TYPE_FLOYD_STEINBERG_ERROR_DIFFUSION
TYPE_JARVIS_JUDICE_NINKE_ERROR_DIFFUSION
TYPE_ORDERED_DITHERING
TYPE_SIERRA_ERROR_DIFFUSION
TYPE_STEVENSON_ARCE_ERROR_DIFFUSION
TYPE_STUCKI_ERROR_DIFFUSION

Constructor summary

UniformPaletteQuantizerDialog(Frame, Strings, int, int, int, int) Creates a modal dialog to enter the parameter.

Method summary

actionPerformed(ActionEvent) Hides (closes) this dialog if the OK button was source of the action event (e.g. if the button was pressed).
adjustmentValueChanged(AdjustmentEvent)
center() Centers the dialog on screen.
getBlueBits()
getDitheringMethod()
getGreenBits()
getRedBits()
hasPressedOk()
isSelectionValid()
itemStateChanged(ItemEvent)

Serializable Fields

- public final int **DITHERING_METHODS**
- private net.sourceforge.jiu.apps.Strings **strings**
- private java.awt.Button **ok**
- private java.awt.Button **cancel**

- private java.awt.Scrollbar **redScrollbar**
- private java.awt.Scrollbar **greenScrollbar**
- private java.awt.Scrollbar **blueScrollbar**
- private java.awt.Choice **ditheringMethod**
- private java.awt.Label **infoLabel1**
- private java.awt.Label **infoLabel2**
- private java.awt.Label **redLabel**
- private java.awt.Label **greenLabel**
- private java.awt.Label **blueLabel**
- private boolean **pressedOk**

Fields

- public static final int **TYPE_DITHERING_NONE**
- public static final int **TYPE_ORDERED_DITHERING**
- public static final int **TYPE_FLOYD_STEINBERG_ERROR_DIFFUSION**
- public static final int **TYPE_STUCKI_ERROR_DIFFUSION**
- public static final int **TYPE_BURKES_ERROR_DIFFUSION**
- public static final int **TYPE_SIERRA_ERROR_DIFFUSION**
- public static final int **TYPE_JARVIS_JUDICE_NINKE_ERROR_DIFFUSION**
- public static final int **TYPE_STEVENSON_ARCE_ERROR_DIFFUSION**
- public final int **DITHERING_METHODS**

Constructors

- *UniformPaletteQuantizerDialog*
 public **UniformPaletteQuantizerDialog**(java.awt.Frame **owner**,
 net.sourceforge.jiu.apps.Strings **strings**, int **redBits**, int **greenBits**, int
blueBits, int **ditheringMethodSelection**)
 - **Description**
 Creates a modal dialog to enter the parameter.
 - **Parameters**
 - * **owner** – the parent of this modal dialog
 - * **strings** – an object to get String constants in the current language
 - * **redBits** – the initial selection of the number of bits for the red channel
 - * **greenBits** – the initial selection of the number of bits for the green channel
 - * **blueBits** – the initial selection of the number of bits for the blue channel
 - * **ditheringMethodSelection** – initial selection for dithering method

Methods

- *actionPerformed*
`public void actionPerformed(java.awt.event.ActionEvent e)`
 - **Description**
Hides (closes) this dialog if the OK button was source of the action event (e.g. if the button was pressed).
- *adjustmentValueChanged*
`void adjustmentValueChanged(java.awt.event.AdjustmentEvent)`
- *center*
`public void center()`
 - **Description**
Centers the dialog on screen.
- *getBlueBits*
`public int getBlueBits()`
- *getDitheringMethod*
`public int getDitheringMethod()`
- *getGreenBits*
`public int getGreenBits()`
- *getRedBits*
`public int getRedBits()`
- *hasPressedOk*
`public boolean hasPressedOk()`
- *isSelectionValid*
`public boolean isSelectionValid()`
- *itemStateChanged*
`void itemStateChanged(java.awt.event.ItemEvent)`

18.1.14 Class WindowSizeDialog

A dialog to enter values for the width and height of a window (typically for a spatial filter like median or mean).

Declaration

```
public class WindowSizeDialog
extends java.awt.Dialog
implements java.awt.event.ActionListener, java.awt.event.KeyListener
```

Constructor summary

WindowSizeDialog(Frame, Strings, int, int, int)

Method summary

actionPerformed(ActionEvent) Hides (closes) this dialog if the OK button was source of the action event (e.g. if the button was pressed).
getHeightValue()
getWidthValue()
hasPressedOk()
keyPressed(KeyEvent)
keyReleased(KeyEvent)
keyTyped(KeyEvent)

Serializable Fields

- private java.awt.Button **ok**
- private java.awt.Button **cancel**
- private java.awt.TextField **width**
- private java.awt.TextField **height**
- private boolean **pressedOk**

Constructors

- *WindowSizeDialog*
public WindowSizeDialog(java.awt.Frame owner,
net.sourceforge.jiu.apps.Strings strings, int titleIndex, int initialWidth,
int initialHeight)
 - **Parameters**
 - * **owner** – the Frame this dialog will belong to

Methods

- *actionPerformed*
`public void actionPerformed(java.awt.event.ActionEvent e)`
 - **Description**
Hides (closes) this dialog if the OK button was source of the action event (e.g. if the button was pressed).
- *getHeightValue*
`public int getHeightValue()`
- *getWidthValue*
`public int getWidthValue()`
- *hasPressedOk*
`public boolean hasPressedOk()`
- *keyPressed*
`void keyPressed(java.awt.event.KeyEvent)`
- *keyReleased*
`void keyReleased(java.awt.event.KeyEvent)`
- *keyTyped*
`void keyTyped(java.awt.event.KeyEvent)`

18.1.15 Class YesNoDialog

A dialog that asks a question and offers a Yes and a No button (and optionally a Cancel button).

Declaration

```
public class YesNoDialog
  extends java.awt.Dialog
  implements java.awt.event.ActionListener
```

Field summary

RESULT_CANCEL Will be returned in (in 18.1.15, page 491)if the CANCEL button was chosen.
RESULT_NO Will be returned in (in 18.1.15, page 491)if the NO button was chosen.
RESULT_YES Will be returned in (in 18.1.15, page 491)if the YES button was chosen.

Constructor summary

YesNoDialog(Frame, Strings, int, int, boolean) Creates a new YesNoDialog object and shows it centered on the screen.

Method summary

actionPerformed(ActionEvent) Hides (closes) this dialog if the OK button was source of the action event (e.g. if the button was pressed).
getResult() Returns one of the RESULT_xyz constants of this class.

Serializable Fields

- private java.awt.Button **yes**
- private java.awt.Button **no**
- private java.awt.Button **cancel**
- private int **result**

Fields

- public static final int **RESULT_YES**
 - Will be returned in (in 18.1.15, page 491)if the YES button was chosen.
- public static final int **RESULT_NO**
 - Will be returned in (in 18.1.15, page 491)if the NO button was chosen.
- public static final int **RESULT_CANCEL**
 - Will be returned in (in 18.1.15, page 491)if the CANCEL button was chosen.

Constructors

- *YesNoDialog*
`public YesNoDialog(java.awt.Frame owner, net.sourceforge.jiu.apps.Strings strings, int titleIndex, int questionIndex, boolean includeCancel)`
 - **Description**
Creates a new YesNoDialog object and shows it centered on the screen.
 - **Parameters**
 - * **owner** – the frame that owns this modal dialog
 - * **strings** – the String resources
 - * **titleIndex** – the index into the String resource of the title text
 - * **questionIndex** – the index into the String resource of the question text
 - * **includeCancel** – determines whether a third button 'Cancel' will be included

Methods

- *actionPerformed*
`public void actionPerformed(java.awt.event.ActionEvent e)`
 - **Description**
Hides (closes) this dialog if the OK button was source of the action event (e.g. if the button was pressed).
-
- *getResult*
`public int getResult()`
 - **Description**
Returns one of the RESULT_xyz constants of this class.
 - **Returns** – the RESULT constant of the button which the user has chosen

Chapter 19

Package net.sourceforge.jiu.ops

Package Contents

Page

Interfaces

ProgressListener	493
<i>This interface must be implemented by classes that want to be notified about progress of an image operation.</i>	

Classes

BatchProcessorOperation	495
<i>Abstract base class to do batch processing on files and complete directory trees.</i>	
ImagesToImageOperation	498
<i>An operation that takes several input images and produces one output image.</i>	
ImageToImageOperation	501
<i>An operation that acesses an input image and produces data for an output image.</i>	
LookupTableOperation	505
<i>An operation that replaces samples with values taken from a lookup table.</i>	
Operation	508
<i>Base class for all operations.</i>	

The operation package, with basic functionality for all JIU classes that actually process images. Includes the base class Operation and some extensions, various exception types and classes for progress notification.

19.1 Interfaces

19.1.1 Interface ProgressListener

This interface must be implemented by classes that want to be notified about progress of an image operation.

Declaration

```
public interface ProgressListener
```

All known subclasses

JiuAwtFrame (in 17.1.7, page 443)

All classes known to implement interface

JiuAwtFrame (in 17.1.7, page 443)

Method summary

setProgress(float) Set the progress level to a new value, which must be between 0.0f and 1.0f (including both of these values).

setProgress(int, int) Sets a new progress level.

Methods

- *setProgress*

```
void setProgress( float progress )
```

- **Description**

Set the progress level to a new value, which must be between 0.0f and 1.0f (including both of these values). You should not call this method with a value lower than any value you've set before. However, this is not checked.

- **Parameters**

- * **progress** – the degree of progress as a value between 0.0f and 1.0f

- **Throws**

- * **java.lang.IllegalArgumentException** – if the float argument is not in the mentioned interval

- *setProgress*

```
void setProgress( int zeroBasedIndex, int totalItems )
```

- **Description**

Sets a new progress level. If an operation consists of totalItems steps, which are numbered from 0 to totalItems - 1, this method can be called after the completion of each step.

Example: if there are three steps and the first one is done, the parameters must be 0 and 3, which will indicated 33% completion. Parameters 1 and 3 mean 66%, 2 and 3 100%. If you use 3 and 3, an IllegalArgumentException will be thrown.

Computes $(\text{float})(\text{zeroBasedIndex} + 1) / (\text{float})\text{totalItems}$ and calls (in 19.1.1, page 493) with that value.

– **Parameters**

- * `zeroBasedIndex` – the index of the step that was just completed
- * `totalItems` – the number of steps in this operation

– **Throws**

- * `java.lang.IllegalArgumentException` – if the parameters don't match the above criteria

19.2 Classes

19.2.1 Class BatchProcessorOperation

Abstract base class to do batch processing on files and complete directory trees. For a non-abstract extension of this operation, you must implement (in 19.2.1, page 497).

Declaration

```
public abstract class BatchProcessorOperation
extends net.sourceforge.jiu.ops.Operation (in 19.2.5, page 508)
```

All known subclasses

ImageLoadTester (in 1.2.5, page 45), ColorIndexer (in 1.2.1, page 33)

Constructor summary

BatchProcessorOperation()

Method summary

addDirectoryTree(String) Adds the argument to the list of directories to be completely processed.

addDirectoryTree(String, String) Adds the first argument to the list of directories to be completely processed, writes all output files to the directory tree specified by the second argument.

addInputFileName(String) Adds a single name to the list of file names to be processed.

addInputFileNames(Vector) Adds a number of file names to the internal list of file names to be processed.

getErrorMessages() Returns a list of error messages collected during the execution of (in 19.2.1, page 497).

getOverwrite() Returns the current overwrite setting.

process() Processes all directory trees and files given to this operation, calling (in 19.2.1, page 497) on each file name.

processFile(String, String, String) Method to be called on each file given to this operation.

setCollectErrorMessages(boolean) Specifies whether error messages are supposed to be collected during the execution of (in 19.2.1, page 497).

setOutputDirectory(String) Specifies the output directory for all single files.

setOverwrite(boolean) Specify whether existing files are to be overwritten.

Constructors

- *BatchProcessorOperation*
public **BatchProcessorOperation**()

Methods

- *addDirectoryTree*

`public void addDirectoryTree(java.lang.String rootDirectoryName)`

- **Description**

Adds the argument to the list of directories to be completely processed.

- **Parameters**

* `rootDirectoryName` – name of the root of the directory tree, can be any valid directory name

- *addDirectoryTree*

`public void addDirectoryTree(java.lang.String rootDirectoryName, java.lang.String outputRootDirectoryName)`

- **Description**

Adds the first argument to the list of directories to be completely processed, writes all output files to the directory tree specified by the second argument.

- **Parameters**

* `rootDirectoryName` – name of the root of the directory tree, can be any valid directory name

* `outputRootDirectoryName` – name of the root of the directory tree, can be any valid directory name

- *addInputFileName*

`public void addInputFileName(java.lang.String fileName)`

- **Description**

Adds a single name to the list of file names to be processed.

- **Parameters**

* `fileName` – name to be added to list

- *addInputFileNames*

`public void addInputFileNames(java.util.Vector fileNameList)`

- **Description**

Adds a number of file names to the internal list of file names to be processed.

- **Parameters**

* `fileNameList` – list of file names, each object in the list must be a String

- *getErrorMessages*

`public java.util.Vector getErrorMessages()`

- **Description**

Returns a list of error messages collected during the execution of (in 19.2.1, page 497).

- **Returns** – list of error messages, each object is a String

- *getOverwrite*

`public boolean getOverwrite()`

– **Description**

Returns the current overwrite setting.

– **Returns** – whether existing files are to be overwritten

• *process*

`public void process()`

– **Description**

Processes all directory trees and files given to this operation, calling (in 19.2.1, page 497) on each file name.

• *processFile*

`public abstract void processFile(java.lang.String inputDirectory,
java.lang.String inputFileName, java.lang.String outputDirectory)`

– **Description**

Method to be called on each file given to this operation. Non-abstract heirs of this class must implement this method to add functionality.

– **Parameters**

- * `inputDirectory` – name of directory where the file to be processed resides
 - * `inputFileName` – name of file to be processed
 - * `outputDirectory` – output directory for that file, need not necessarily be used
-

• *setCollectErrorMessages*

`public void setCollectErrorMessages(boolean collectErrorMessages)`

– **Description**

Specifies whether error messages are supposed to be collected during the execution of (in 19.2.1, page 497).

– **Parameters**

- * `collectErrorMessages` – if true, error messages will be collected, otherwise not

– **See also**

- * `BatchProcessorOperation.getErrorMessages()` (in 19.2.1, page 496)
-

• *setOutputDirectory*

`public void setOutputDirectory(java.lang.String outputDirectoryName)`

– **Description**

Specifies the output directory for all single files. Note that you can specify different output directories when dealing with directory trees.

– **Parameters**

- * `outputDirectoryName` – name of output directory
-

• *setOverwrite*

`public void setOverwrite(boolean newValue)`

– **Description**

Specify whether existing files are to be overwritten.

– **Parameters**

- * `newValue` – if true, files are overwritten, otherwise not

– **See also**

- * `BatchProcessorOperation.getOverwrite()` (in 19.2.1, page 496)

19.2.2 Class ImagesToImageOperation

An operation that takes several input images and produces one output image.

Declaration

```
public abstract class ImagesToImageOperation
  extends net.sourceforge.jiu.ops.Operation (in 19.2.5, page 508)
```

Constructor summary

- ImagesToImageOperation()** Constructs a new ImagesToImageOperation and initializes input images and output image to null.
- ImagesToImageOperation(Vector, PixelImage)** Constructs a new ImagesToImageOperation and initializes input images and output image to the arguments.

Method summary

- addInputImage(PixelImage)** Adds an image to the end of the internal list of input images.
- ensureImagesHaveSameResolution()** Checks if all images have the same resolution as given by their getWidth and getHeight methods.
- ensureOutputImageResolution(int, int)** If an output image has been specified this method will compare its resolution with the argument resolution and throw an exception if the resolutions differ.
- getInputImage(int)** Returns the input image stored in this object.
- getNumInputImages()** Return the number of input images currently stored in this operation.
- getOutputImage()** Returns the output image stored in this object.
- setOutputImage(PixelImage)** Sets the output image stored in this object to the argument.

Constructors

- *ImagesToImageOperation*
 public **ImagesToImageOperation**()
 – **Description**
 Constructs a new ImagesToImageOperation and initializes input images and output image to null.

- *ImagesToImageOperation*
 public **ImagesToImageOperation**(java.util.Vector in,
 net.sourceforge.jiu.data.PixelImage out)
 – **Description**
 Constructs a new ImagesToImageOperation and initializes input images and output image to the arguments.

Methods

- *addInputImage*
 public void **addInputImage**(net.sourceforge.jiu.data.PixelImage in)
 – **Description**
 Adds an image to the end of the internal list of input images.

- *ensureImagesHaveSameResolution*
 public void **ensureImagesHaveSameResolution**() throws
 net.sourceforge.jiu.ops.WrongParameterException
 – **Description**
 Checks if all images have the same resolution as given by their getWidth and getHeight methods. This method will not complain if input and / or output images are not available.
 – **Throws**
 * net.sourceforge.jiu.ops.WrongParameterException – if input and output images exist and their resolutions differ

- *ensureOutputImageResolution*
 public void **ensureOutputImageResolution**(int width, int height) throws
 net.sourceforge.jiu.ops.WrongParameterException
 – **Description**
 If an output image has been specified this method will compare its resolution with the argument resolution and throw an exception if the resolutions differ. If no output image has been specified nothing happens.
 – **Parameters**
 * **width** – the horizontal pixel resolution that the output image must have
 * **height** – the vertical pixel resolution that the output image must have
 – **Throws**
 * net.sourceforge.jiu.ops.WrongParameterException – if the resolutions differ

- *getInputImage*
 public net.sourceforge.jiu.data.PixelImage **getInputImage**(int index)
 – **Description**
 Returns the input image stored in this object.
 – **Returns** – input image, possibly null

- *getNumInputImages*
 public int **getNumInputImages**()
 – **Description**
 Return the number of input images currently stored in this operation.
 – **Returns** – number of images

- *getOutputImage*
 public net.sourceforge.jiu.data.PixelImage **getOutputImage**()

- **Description**

Returns the output image stored in this object.

- **Returns** – output image, possibly null

- *setOutputImage*

```
public void setOutputImage( net.sourceforge.jiu.data.PixelImage out )
```

- **Description**

Sets the output image stored in this object to the argument. Argument can be null.

- **Parameters**

- * **out** – the new output image of this object

19.2.3 Class ImageToImageOperation

An operation that accesses an input image and produces data for an output image. This abstract class only provides methods to get and set those images.

Normally, an operation creates the output image itself. However, an output image can be specified by the user with (in 19.2.3, page 504). This could be done when existing image objects are to be reused.

An operation extending ImageToImageOperation must check if (1) a user-defined output image is available and (2) whether that image matches the required criteria. The criteria depend on the operation - example: for an operation that rotates an image by 180 degrees, an output image must have the same resolution as the input image and be of the same type.

If an output image is not available (case #1), the operation must create the matching output image itself. It should know best what is required. Very generic methods (like rotation of images by 90 degrees) must know relatively little about the image. They can make use of `PixelImage.createCompatibleImage(int, int)` and provide width and height. That way, the operation works for all kinds of images, like `BilevelImage`, `Paletted8Image`, `Gray8Image`, `RGB24Image` etc.

If a user-provided image does not match the required criteria, an appropriate exception (most of the time (in 19.3.3, page 514) will do) with a descriptive error message must be thrown. In the example of the 90-degree rotation, the width of the output image must be equal to the height of the input image and vice versa. The types of input and output must be equal.

However, there are limits to the checks on user-provided output images. As an example, a generic test could not check if a paletted output image has the same palette as the input counterpart because it treats all images based on `IntegerImage` the same.

When performing an image-to-image-operation, the input image can possibly be used as the output image. This can be done

- if input and output are of the same type and resolution and
- if the operation needs only one input pixel to compute the output pixel at any given position.

Mirroring the image horizontally is an example of an operation that can be implemented that way - the operation starts at the top left and at the bottom right pixel, swaps them and proceeds one pixel to the right of the top left pixel (and one to the left of the bottom right pixel).

Declaration

```
public abstract class ImageToImageOperation
extends net.sourceforge.jiu.ops.Operation (in 19.2.5, page 508)
```

All known subclasses

`Invert` (in 4.2.1, page 171), `NormalizeHistogram` (in 6.1.6, page 208), `HueSaturationValue` (in 6.1.5, page 206), `GammaCorrection` (in 6.1.4, page 204), `EqualizeHistogram` (in 6.1.3, page 203), `Contrast` (in 6.1.2, page 201), `Brightness` (in 6.1.1, page 199), `OrderedDither` (in 9.2.5, page 241), `ErrorDiffusionDithering` (in 9.2.3, page 234), `ClusteredDotDither` (in 9.2.1, page 230), `PromotionRGB48` (in 11.1.5, page 256), `PromotionRGB24` (in 11.1.4, page 255), `PromotionPaletted8` (in 11.1.3, page 254), `PromotionGray8` (in 11.1.2, page 253), `PromotionGray16` (in 11.1.1, page 252), `UniformPaletteQuantizer` (in 12.2.11, page 293), `PopularityQuantizer` (in 12.2.7, page 285), `OctreeColorQuantizer` (in 12.2.5, page 279), `MedianCutQuantizer` (in 12.2.4, page 273), `MedianCutContourRemoval` (in 12.2.2, page 264), `ArbitraryPaletteQuantizer` (in 12.2.1, page 261), `RGBToGrayConversion` (in 13.1.5, page 305),

ReduceToBilevelThreshold (in 13.1.4, page 303), ReduceShadesOfGray (in 13.1.3, page 301), ReduceRGB (in 13.1.2, page 299), OilFilter (in 15.1.9, page 388), MinimumFilter (in 15.1.8, page 387), MedianFilter (in 15.1.7, page 385), MeanFilter (in 15.1.6, page 383), MaximumFilter (in 15.1.5, page 381), ConvolutionKernelFilter (in 15.1.4, page 377), AreaFilterOperation (in 15.1.1, page 367), Shear (in 16.1.16, page 418), ScaleReplication (in 16.1.15, page 416), Rotate90Right (in 16.1.14, page 414), Rotate90Left (in 16.1.13, page 413), Rotate180 (in 16.1.12, page 411), Resample (in 16.1.10, page 405), Mirror (in 16.1.8, page 402), Flip (in 16.1.5, page 399), Crop (in 16.1.4, page 397), LookupTableOperation (in 19.2.4, page 505)

Constructor summary

- ImageToImageOperation()** Creates an object of this class and sets both input image and output image to null.
- ImageToImageOperation(PixelImage)** Creates an object of this class and sets the input image to the argument value, output image to null.
- ImageToImageOperation(PixelImage, PixelImage)** Creates an object of this class and sets input image and output image to the argument values.

Method summary

- canInputAndOutputBeEqual()** Returns if input and output image are allowed to be the same object.
- ensureImagesHaveSameResolution()** If both an input and an output image have been specified (both non-null), this method compares their width and height properties and throws an exception if the two images do not have the same resolution.
- ensureInputImageIsAvailable()** If (in 19.2.3, page 504) returns null this method throws a (in 19.3.1, page 512) complaining that an input image is missing.
- ensureOutputImageResolution(int, int)** If an output image has been specified this method will compare its resolution with the argument resolution and throw an exception if the resolutions differ.
- getInputImage()** Returns the input image stored in this object.
- getOutputImage()** Returns the output image stored in this object.
- setCanInputAndOutputBeEqual(boolean)** Specify if input and output image are allowed to be the same object.
- setInputImage(PixelImage)** Sets the input image stored in this object to the argument.
- setOutputImage(PixelImage)** Sets the output image stored in this object to the argument.

Constructors

- *ImageToImageOperation*
 public **ImageToImageOperation**()
 – **Description**
 Creates an object of this class and sets both input image and output image to null.

- *ImageToImageOperation*
 public **ImageToImageOperation**(net.sourceforge.jiu.data.PixelImage in)

– **Description**

Creates an object of this class and sets the input image to the argument value, output image to null.

- *ImageToImageOperation*

```
public ImageToImageOperation( net.sourceforge.jiu.data.PixelImage in,
net.sourceforge.jiu.data.PixelImage out )
```

– **Description**

Creates an object of this class and sets input image and output image to the argument values.

Methods

- *canInputAndOutputBeEqual*

```
public boolean canInputAndOutputBeEqual( )
```

– **Description**

Returns if input and output image are allowed to be the same object.

– **See also**

* `ImageToImageOperation.setCanInputAndOutputBeEqual(boolean)` (in 19.2.3, page 504)

- *ensureImagesHaveSameResolution*

```
public void ensureImagesHaveSameResolution( ) throws
net.sourceforge.jiu.ops.WrongParameterException
```

– **Description**

If both an input and an output image have been specified (both non-null), this method compares their width and height properties and throws an exception if the two images do not have the same resolution.

– **Throws**

* `net.sourceforge.jiu.ops.WrongParameterException` – if input and output images exist and their resolutions differ

- *ensureInputImageIsAvailable*

```
public void ensureInputImageIsAvailable( ) throws
net.sourceforge.jiu.ops.MissingParameterException
```

– **Description**

If (in 19.2.3, page 504) returns null this method throws a (in 19.3.1, page 512) complaining that an input image is missing.

– **Throws**

* `net.sourceforge.jiu.ops.MissingParameterException` – if no input image is available

- *ensureOutputImageResolution*

```
public void ensureOutputImageResolution( int width, int height ) throws
net.sourceforge.jiu.ops.WrongParameterException
```

– **Description**

If an output image has been specified this method will compare its resolution with the argument resolution and throw an exception if the resolutions differ. If no output image has been specified nothing happens.

– **Parameters**

- * **width** – the horizontal pixel resolution that the output image must have
- * **height** – the vertical pixel resolution that the output image must have

– **Throws**

- * `net.sourceforge.jiu.ops.WrongParameterException` – if the resolutions differ
-

• *getInputImage*

```
public net.sourceforge.jiu.data.PixelImage getInputImage( )
```

– **Description**

Returns the input image stored in this object.

– **Returns** – input image, possibly null

• *getOutputImage*

```
public net.sourceforge.jiu.data.PixelImage getOutputImage( )
```

– **Description**

Returns the output image stored in this object.

– **Returns** – output image, possibly null

• *setCanInputAndOutputBeEqual*

```
public void setCanInputAndOutputBeEqual( boolean newValue )
```

– **Description**

Specify if input and output image are allowed to be the same object.

– **See also**

- * `ImageToImageOperation.canInputAndOutputBeEqual()` (in 19.2.3, page 503)
-

• *setInputImage*

```
public void setInputImage( net.sourceforge.jiu.data.PixelImage in )
```

– **Description**

Sets the input image stored in this object to the argument. Argument can be null.

– **Parameters**

- * **in** – the new input image of this object
-

• *setOutputImage*

```
public void setOutputImage( net.sourceforge.jiu.data.PixelImage out )
```

– **Description**

Sets the output image stored in this object to the argument. Argument can be null.

– **Parameters**

- * **out** – the new output image of this object

19.2.4 Class LookupTableOperation

An operation that replaces samples with values taken from a lookup table. Operations where each pixel is treated independently from its neighbors and where a pixel value is always mapped to the same new pixel value can be implemented this way.

Declaration

```
public abstract class LookupTableOperation
extends net.sourceforge.jiu.ops.ImageToImageOperation (in 19.2.3, page 501)
```

All known subclasses

NormalizeHistogram (in 6.1.6, page 208), GammaCorrection (in 6.1.4, page 204), EqualizeHistogram (in 6.1.3, page 203), Contrast (in 6.1.2, page 201), Brightness (in 6.1.1, page 199)

Constructor summary

LookupTableOperation() Creates a LookupTableOperation for one lookup table.

LookupTableOperation(int) Creates an object of this class, calling the super constructor with two null arguments and allocates space for the argument number of lookup tables.

Method summary

getNumTables() Returns the number of tables in this operation.

getTable(int) Returns one of the internal int lookup tables.

prepareImages()

process()

setNumTables(int) Resets the number of tables to be used in this operation to the argument and drops all actual table data initialized so far.

setTable(int, int[]) Provides a new lookup table for one of the channels.

setTables(int[]) Sets the tables for all channels to the argument table.

Constructors

- *LookupTableOperation*

```
public LookupTableOperation( )
```

- **Description**

Creates a LookupTableOperation for one lookup table.

- *LookupTableOperation*

```
public LookupTableOperation( int numTables )
```

- **Description**

Creates an object of this class, calling the super constructor with two null arguments and allocates space for the argument number of lookup tables.

- **Parameters**

* **numTables** – number of tables to be used in this operation

Methods

- *getNumTables*
 public int **getNumTables**()
 – **Description**
 Returns the number of tables in this operation.
 – **Returns** – number of tables

- *getTable*
 public int[] **getTable**(int **channelIndex**)
 – **Description**
 Returns one of the internal int lookup tables.
 – **Parameters**
 * **channelIndex** – the zero-based index of the table to be returned; from 0 to **getNumTables()** - 1
 – **Returns** – the **channelIndex**'th table

- *prepareImages*
 public void **prepareImages**() throws
 net.sourceforge.jiu.ops.MissingParameterException,
 net.sourceforge.jiu.ops.WrongParameterException

- *process*
 public void **process**() throws
 net.sourceforge.jiu.ops.MissingParameterException,
 net.sourceforge.jiu.ops.OperationFailedException,
 net.sourceforge.jiu.ops.WrongParameterException
 – **Description copied from Operation (in 19.2.5, page 508)**
 This method does the actual work of the operation. It must be called after all parameters have been given to the operation object.
 – **Throws**
 * net.sourceforge.jiu.ops.WrongParameterException – if at least one of the input parameters was not initialized appropriately (values out of the valid interval, etc.)
 * net.sourceforge.jiu.ops.MissingParameterException – if any mandatory parameter was not given to the operation
 * net.sourceforge.jiu.ops.OperationFailedException –

- *setNumTables*
 public void **setNumTables**(int **numberOfTables**)
 – **Description**
 Resets the number of tables to be used in this operation to the argument and drops all actual table data initialized so far. After a call to this method, (in 19.2.4, page 506) will return null as long as no new table data is provided via (in 19.2.4, page 507) or (in 19.2.4, page 507).
 – **Parameters**
 * **numberOfTables** – the new number of tables for this operation, must be 1 or larger

- **Throws**

- * `java.lang.IllegalArgumentException` – if the number is zero or smaller

- *setTable*

```
public void setTable( int channelIndex, int[] tableData )
```

- **Description**

- Provides a new lookup table for one of the channels.

- **Parameters**

- * `channelIndex` – the index of the channel for which a table is provided; must be at least 0 and smaller than (in 19.2.4, page 506)

- * `tableData` – the actual table to be used for lookup

- **Throws**

- * `java.lang.IllegalArgumentException` – if the channel index is not in the valid interval (see above)

- *setTables*

```
public void setTables( int[] tableData )
```

- **Description**

- Sets the tables for all channels to the argument table. Useful when the same table can be used for all channels.

- **Parameters**

- * `tableData` – the data that will be used as lookup table for all channels

19.2.5 Class Operation

Base class for all operations.

It supports progress notification. All classes that want to be notified by a new progress level of the operation (defined as value between 0.0f (nothing has been done so far) to 1.0f (operation finished)) must implement the `Operation` interface.

An abortion state is stored in each `Operation` object. It should be queried by a running operation from time to time (via `isAborted()` in 19.2.5, page 509)- if it returns `true`, the operation should terminate and return control to the caller. The abort state can be modified using `setAborted()` in 19.2.5, page 510).

Declaration

```
public abstract class Operation
extends java.lang.Object
```

All known subclasses

`ImageLoadTester` (in 1.2.5, page 45), `ColorIndexer` (in 1.2.1, page 33), `RASCodec` (in 2.1.12, page 122), `PSDCodec` (in 2.1.11, page 120), `PNMCodec` (in 2.1.10, page 115), `PNGCodec` (in 2.1.9, page 109), `PCDCodec` (in 2.1.8, page 105), `PalmCodec` (in 2.1.7, page 96), `ImageCodec` (in 2.1.5, page 79), `IFFCodec` (in 2.1.4, page 76), `GIFCodec` (in 2.1.3, page 71), `BMPCodec` (in 2.1.1, page 66), `TIFFCodec` (in 3.2.1, page 137), `Invert` (in 4.2.1, page 171), `NormalizeHistogram` (in 6.1.6, page 208), `HueSaturationValue` (in 6.1.5, page 206), `GammaCorrection` (in 6.1.4, page 204), `EqualizeHistogram` (in 6.1.3, page 203), `Contrast` (in 6.1.2, page 201), `Brightness` (in 6.1.1, page 199), `TextureAnalysis` (in 7.1.5, page 219), `MeanDifference` (in 7.1.4, page 217), `Histogram3DCreator` (in 7.1.2, page 213), `Histogram1DCreator` (in 7.1.1, page 210), `OrderedDither` (in 9.2.5, page 241), `ErrorDiffusionDithering` (in 9.2.3, page 234), `ClusteredDotDither` (in 9.2.1, page 230), `PromotionRGB48` (in 11.1.5, page 256), `PromotionRGB24` (in 11.1.4, page 255), `PromotionPaletted8` (in 11.1.3, page 254), `PromotionGray8` (in 11.1.2, page 253), `PromotionGray16` (in 11.1.1, page 252), `UniformPaletteQuantizer` (in 12.2.11, page 293), `PopularityQuantizer` (in 12.2.7, page 285), `OctreeColorQuantizer` (in 12.2.5, page 279), `MedianCutQuantizer` (in 12.2.4, page 273), `MedianCutContourRemoval` (in 12.2.2, page 264), `ArbitraryPaletteQuantizer` (in 12.2.1, page 261), `RGBToGrayConversion` (in 13.1.5, page 305), `ReduceToBilevelThreshold` (in 13.1.4, page 303), `ReduceShadesOfGray` (in 13.1.3, page 301), `ReduceRGB` (in 13.1.2, page 299), `AutoDetectColorType` (in 13.1.1, page 296), `OilFilter` (in 15.1.9, page 388), `MinimumFilter` (in 15.1.8, page 387), `MedianFilter` (in 15.1.7, page 385), `MeanFilter` (in 15.1.6, page 383), `MaximumFilter` (in 15.1.5, page 381), `ConvolutionKernelFilter` (in 15.1.4, page 377), `AreaFilterOperation` (in 15.1.1, page 367), `Shear` (in 16.1.16, page 418), `ScaleReplication` (in 16.1.15, page 416), `Rotate90Right` (in 16.1.14, page 414), `Rotate90Left` (in 16.1.13, page 413), `Rotate180` (in 16.1.12, page 411), `Resample` (in 16.1.10, page 405), `Mirror` (in 16.1.8, page 402), `Flip` (in 16.1.5, page 399), `Crop` (in 16.1.4, page 397), `LookupTableOperation` (in 19.2.4, page 505), `ImageToImageOperation` (in 19.2.3, page 501), `ImagesToImageOperation` (in 19.2.2, page 498), `BatchProcessorOperation` (in 19.2.1, page 495)

Constructor summary

Operation() This constructor creates two internal empty lists for progress listeners and parameters.

Method summary

addProgressListener(ProgressListener) Adds the argument progress listener to the internal list of progress listeners.

- addProgressListeners(Vector)** Adds several progress listeners to this operation object.
- getAbort()** Returns the current abort status.
- process()** This method does the actual work of the operation.
- removeProgressListener(ProgressListener)** Removes the argument progress listener from the internal list of progress listeners.
- setAbort(boolean)** Sets a new abort status.
- setProgress(float)** This method will notify all registered progress listeners about a new progress level.
- setProgress(int, int)** This method will notify all registered progress listeners about a new progress level.

Constructors

- *Operation*
public Operation()
 - **Description**
This constructor creates two internal empty lists for progress listeners and parameters.

Methods

- *addProgressListener*
public void addProgressListener(ProgressListener progressListener)
 - **Description**
Adds the argument progress listener to the internal list of progress listeners. Does not check if the argument already exists in that list, so you have to check for duplicates yourself.
 - **Parameters**
 - * **progressListener** – the progress listener to be added
- *addProgressListeners*
public void addProgressListeners(java.util.Vector progressListeners)
 - **Description**
Adds several progress listeners to this operation object.
 - **Parameters**
 - * **progressListeners** – contains zero or more objects implementing ProgressListener; each will be added by calling (in 19.2.5, page 509) on it
- *getAbort*
public boolean getAbort()
 - **Description**
Returns the current abort status. If **true**, a running operation should terminate what it is doing (return from (in 19.2.5, page 510)).
 - **Returns** – abort status
 - **See also**

* `Operation.setAbort(boolean)` (in 19.2.5, page 510)

- *process*

`public void process()` throws
`net.sourceforge.jiu.ops.MissingParameterException`,
`net.sourceforge.jiu.ops.OperationFailedException`,
`net.sourceforge.jiu.ops.WrongParameterException`

- **Description**

This method does the actual work of the operation. It must be called after all parameters have been given to the operation object.

- **Throws**

- * `net.sourceforge.jiu.ops.WrongParameterException` – if at least one of the input parameters was not initialized appropriately (values out of the valid interval, etc.)
 - * `net.sourceforge.jiu.ops.MissingParameterException` – if any mandatory parameter was not given to the operation
 - * `net.sourceforge.jiu.ops.OperationFailedException` –
-

- *removeProgressListener*

`public void removeProgressListener(ProgressListener progressListener)`

- **Description**

Removes the argument progress listener from the internal list of progress listeners.

- **Parameters**

- * `progressListener` – the progress listener to be removed
-

- *setAbort*

`public void setAbort(boolean newAbortStatus)`

- **Description**

Sets a new abort status.

- **Parameters**

- * `newAbortStatus` – the new status

- **See also**

- * `Operation.getAbort()` (in 19.2.5, page 509)
-

- *setProgress*

`public void setProgress(float progress)`

- **Description**

This method will notify all registered progress listeners about a new progress level. The argument must be from 0.0f to 1.0f where 0.0f marks the beginning and 1.0f completion. The progress value should not be smaller than any value that was previously set.

- **Parameters**

- * `progress` – new progress value, from 0.0 to 1.0
-

- *setProgress*

`public void setProgress(int zeroBasedIndex, int totalItems)`

- **Description**

This method will notify all registered progress listeners about a new progress level. Simply checks the arguments and calls `setProgress((float)zeroBasedIndex / (float)totalItems);`.

- **Parameters**

- * `zeroBasedIndex` – the index of the item that was just processed, zero-based
- * `totalItems` – the number of items that will be processed

19.3 Exceptions

19.3.1 *Class* **MissingParameterException**

Exception class to indicate that an operation's parameter is missing (has not been specified by caller and there was no default value that could be used).

Declaration

```
public class MissingParameterException
extends net.sourceforge.jiu.ops.OperationFailedException (in 19.3.2, page 513)
```

Constructor summary

MissingParameterException(String)

Constructors

- *MissingParameterException*
public MissingParameterException(java.lang.String message)

19.3.2 *Class* **OperationFailedException**

Exception class to indicate that an operation failed during the execution of the method (in 19.2.5, page 510). Note that a failure due to missing or wrong parameters must lead to the dedicated exception classes (in 19.3.3, page 514) or (in 19.3.1, page 512) being thrown.

Declaration

```
public class OperationFailedException
    extends java.lang.Exception
```

All known subclasses

WrongFileFormatException (in 2.2.5, page 129), UnsupportedTypeException (in 2.2.4, page 128), UnsupportedCodecModeException (in 2.2.3, page 127), InvalidImageIndexException (in 2.2.2, page 126), InvalidFileStructureException (in 2.2.1, page 125), WrongParameterException (in 19.3.3, page 514), MissingParameterException (in 19.3.1, page 512)

Constructor summary

OperationFailedException(String)

Constructors

- *OperationFailedException*
`public OperationFailedException(java.lang.String message)`

19.3.3 Class **WrongParameterException**

Exception class to indicate that an operation's parameter is of the wrong type, does not fall into a valid interval or a similar mistake.

Declaration

```
public class WrongParameterException
extends net.sourceforge.jiu.ops.OperationFailedException (in 19.3.2, page 513)
```

Constructor summary

WrongParameterException(String)

Constructors

- *WrongParameterException*
public WrongParameterException(java.lang.String message)

Chapter 20

Package net.sourceforge.jiu.util

Package Contents

Page

Interfaces

ComparatorInterface	516
<i>To be able to do sorting in Java 1.1 as defined in java.util.Arrays (which is only available in Java 1.2 and higher), we offer a java.util.Comparator clone under a different name: ComparatorInterface.</i>	

Classes

ArrayConverter	517
<i>Helper class with static methods to convert between byte arrays and primitive types.</i>	
ArrayRotation	522
<i>Provides static methods to rotate (in steps of 90 degrees), flip and mirror array elements.</i>	
ArrayScaling	525
<i>This class currently only scales up an image given as a one-dimensional array of values.</i>	
Median	526
<i>Pick the median value from an array (or an interval of an array).</i>	
SeekableByteArrayOutputStream	528
<i>An extension of that writes data to an internal byte array, resizing it when necessary.</i>	
Sort	532
<i>Provides sorting of an Object array.</i>	
Statistics	533
<i>A number of static methods to compute statistical properties of an array of double values.</i>	
SystemInfo	537
<i>Class to retrieve system information in a human-readable form.</i>	

Various helper classes with functionality not directly related to imaging.

20.1 Interfaces

20.1.1 *Interface* ComparatorInterface

To be able to do sorting in Java 1.1 as defined in java.util.Arrays (which is only available in Java 1.2 and higher), we offer a java.util.Comparator clone under a different name: ComparatorInterface. Sorting will be provided by the (in 20.2.6, page 532)class of this package.

Declaration

```
public interface ComparatorInterface
```

All known subclasses

RGBColorComparator (in 12.2.9, page 290), OctreeNode (in 12.2.6, page 282)

All classes known to implement interface

RGBColorComparator (in 12.2.9, page 290), OctreeNode (in 12.2.6, page 282)

Method summary

compare(Object, Object) Compares the two argument objects and returns their relation.

Methods

- *compare*

```
int compare( java.lang.Object o1, java.lang.Object o2 )
```

- **Description**

Compares the two argument objects and returns their relation. Returns

- * a value <0 if o1 is smaller than o2,
- * 0 if o1 is equal to o2 and
- * a value >0 if o1 is greater than o2.

20.2 Classes

20.2.1 Class ArrayConverter

Helper class with static methods to convert between byte arrays and primitive types. Useful for serialization.

Declaration

```
public class ArrayConverter
extends java.lang.Object
```

Method summary

convertPacked2BitIntensityTo8Bit(byte[], int, byte[], int, int) Converts bytes with two four-bit-intensity samples to 8 byte intensity values, each stored in one byte.

convertPacked4BitIntensityTo8Bit(byte[], int, byte[], int, int) Converts bytes with four two-bit-intensity samples to byte-sized intensity values.

copyPackedBytes(byte[], int, int, byte[], int, int, int) Copies a number of bit values from one byte array to another.

decodePacked1Bit(byte[], int, byte[], int, int)

decodePacked2Bit(byte[], int, byte[], int, int) Decodes bytes with four two-bit samples to single bytes.

decodePacked4Bit(byte[], int, byte[], int, int) Decodes bytes with two four-bit samples to single bytes.

decodePackedRGB565BigEndianToRGB24(byte[], int, byte[], int, byte[], int, byte[], int, int) Convert 16 bit RGB samples stored in big endian (BE) byte order with 5 bits for red and blue and 6 bits for green to 24 bit RGB byte samples.

encodePacked2Bit(byte[], int, byte[], int, int)

encodePacked4Bit(byte[], int, byte[], int, int)

encodeRGB24ToPackedRGB565BigEndian(byte[], int, byte[], int, byte[], int, byte[], int, int) Convert 24 bit RGB pixels to 16 bit pixels stored in big endian (BE) byte order with 5 bits for red and blue and 6 bits for green.

getIntBE(byte[], int) Reads four consecutive bytes from the given array at the given position in big endian order and returns them as an **int**.

getIntLE(byte[], int) Reads four consecutive bytes from the given array at the given position in little endian order and returns them as an **int**.

getShortBE(byte[], int) Reads two consecutive bytes from the given array at the given position in big endian order and returns them as a **short**.

getShortLE(byte[], int) Reads two consecutive bytes from the given array at the given position in little endian order and returns them as a **short**.

setIntBE(byte[], int, int) Writes an int value into four consecutive bytes of a byte array, in big endian (network) byte order.

setIntLE(byte[], int, int) Writes an int value into four consecutive bytes of a byte array, in little endian (Intel) byte order.

setShortBE(byte[], int, short)

setShortLE(byte[], int, short)

Methods

- *convertPacked2BitIntensityTo8Bit*

```
public static void convertPacked2BitIntensityTo8Bit( byte[] src, int
srcOffset, byte[] dest, int destOffset, int numPackedBytes )
```

- **Description**

Converts bytes with two four-bit-intensity samples to 8 byte intensity values, each stored in one byte. Two-bit values can be 0, 1, 2 or 3. These values will be scaled to the full [0;255] range so that 0 remains 0, 1 becomes 85, 2 becomes 170 and 3 becomes 255. A little discussion on how to implement this method was held in the German Java newsgroup **de.comp.lang.java** (at news:de.comp.lang.java). The message I wrote to start the thread has the ID 1ef7du4vfqsd2pskb6jukut6pnhn87htt2@4ax.com. Read the **thread at Google Groups** (at http://groups.google.com/groups?as_umsgid=1ef7du4vfqsd2pskb6jukut6pnhn87htt2@4ax.com).

- **Parameters**

- * **src** – byte array, each byte stores four two-bit intensity values
- * **srcOffset** – index into src
- * **dest** – byte array, each byte stores an eight-bit intensity values
- * **destOffset** – index into dest
- * **numPackedBytes** – number of bytes in src to be decoded

- *convertPacked4BitIntensityTo8Bit*

```
public static void convertPacked4BitIntensityTo8Bit( byte[] src, int
srcOffset, byte[] dest, int destOffset, int numPackedBytes )
```

- **Description**

Converts bytes with four two-bit-intensity samples to byte-sized intensity values. Four-bit values can be from 0 to 15. These values will be scaled to the full [0;255] range so that 0 remains 0, 1 becomes 17, 2 becomes 34, ..., and 15 becomes 255. The most significant four bits in a byte become the left, the least significant four bits the right pixel.

- **Parameters**

- * **src** – byte array, each byte stores two four-bit intensity values
- * **srcOffset** – index into src
- * **dest** – byte array, each byte stores an eight-bit intensity values
- * **destOffset** – index into dest
- * **numPackedBytes** – number of bytes in src to be decoded

- *copyPackedBytes*

```
public static void copyPackedBytes( byte[] src, int srcOffset, int
srcBitOffset, byte[] dest, int destOffset, int destBitOffset, int numSamples
)
```

- **Description**

Copies a number of bit values from one byte array to another.

- **Parameters**

- * **src** – array from which is copied
- * **srcOffset** – index into the src array of the first byte from which is copied
- * **srcBitOffset** – first bit within src[srcOffset] from which is copied (0 is left-most, 1 is second left-most, 7 is right-most)

- * **dest** – array to which is copied
- * **destOffset** – index into the dest array of the first byte to which is copied
- * **destBitOffset** – first bit within dest[destOffset] to which is copied (0 is left-most, 1 is second left-most, 7 is right-most)
- * **numSamples** – number of bits to be copied

- *decodePacked1Bit*

```
public static void decodePacked1Bit( byte[] src, int srcOffset, byte[] dest,
int destOffset, int numPackedBytes )
```

- *decodePacked2Bit*

```
public static void decodePacked2Bit( byte[] src, int srcOffset, byte[] dest,
int destOffset, int numPackedBytes )
```

- **Description**

Decodes bytes with four two-bit samples to single bytes. The two most significant bits of a source byte become the first value, the two least significant bits the fourth value. The method expects numPackedBytes bytes at src[srcOffset] (these will be read and interpreted) and numPackedBytes * 4 at dest[destOffset] (where the decoded byte values will be stored).

- **Parameters**

- * **src** – byte array, each byte stores four two-bit values
 - * **srcOffset** – index into src
 - * **dest** – byte array, each byte stores a single decoded value (from 0 to 3)
 - * **destOffset** – index into dest
 - * **numPackedBytes** – number of bytes in src to be decoded
-

- *decodePacked4Bit*

```
public static void decodePacked4Bit( byte[] src, int srcOffset, byte[] dest,
int destOffset, int numPackedBytes )
```

- **Description**

Decodes bytes with two four-bit samples to single bytes. The four most significant bits of a source byte become the first value, the least significant four bits the second value. The method expects numPackedBytes bytes at src[srcOffset] (these will be read and interpreted) and numPackedBytes * 2 at dest[destOffset] (where the decoded byte values will be stored).

- **Parameters**

- * **src** – byte array, each byte stores two four-bit values
 - * **srcOffset** – index into src
 - * **dest** – byte array, each byte stores a single decoded value
 - * **destOffset** – index into dest
 - * **numPackedBytes** – number of bytes in src to be decoded
-

- *decodePackedRGB565BigEndianToRGB24*

```
public static void decodePackedRGB565BigEndianToRGB24( byte[] src, int
srcOffset, byte[] red, int redOffset, byte[] green, int greenOffset, byte[]
blue, int blueOffset, int numPixels )
```

- **Description**

Convert 16 bit RGB samples stored in big endian (BE) byte order with 5 bits for red and blue and 6 bits for green to 24 bit RGB byte samples.

- *encodePacked2Bit*

```
public static void encodePacked2Bit( byte[] src, int srcOffset, byte[] dest,
int destOffset, int numSamples )
```

- *encodePacked4Bit*

```
public static void encodePacked4Bit( byte[] src, int srcOffset, byte[] dest,
int destOffset, int numSamples )
```

- *encodeRGB24ToPackedRGB565BigEndian*

```
public static void encodeRGB24ToPackedRGB565BigEndian( byte[] red,
int redOffset, byte[] green, int greenOffset, byte[] blue, int blueOffset,
byte[] dest, int destOffset, int numPixels )
```

- **Description**

Convert 24 bit RGB pixels to 16 bit pixels stored in big endian (BE) byte order with 5 bits for red and blue and 6 bits for green.

- *getIntBE*

```
public static int getIntBE( byte[] src, int srcOffset )
```

- **Description**

Reads four consecutive bytes from the given array at the given position in big endian order and returns them as an int.

- **Parameters**

- * **src** – the array from which bytes are read
- * **srcOffset** – the index into the array from which the bytes are read

- **Returns** – int value taken from the array

- *getIntLE*

```
public static int getIntLE( byte[] src, int srcOffset )
```

- **Description**

Reads four consecutive bytes from the given array at the given position in little endian order and returns them as an int.

- **Parameters**

- * **src** – the array from which bytes are read
- * **srcOffset** – the index into the array from which the bytes are read

- **Returns** – short value taken from the array

- *getShortBE*

```
public static short getShortBE( byte[] src, int srcOffset )
```

- **Description**

Reads two consecutive bytes from the given array at the given position in big endian order and returns them as a short.

- **Parameters**

- * **src** – the array from which two bytes are read
- * **srcOffset** – the index into the array from which the two bytes are read

- **Returns** – short value taken from the array

- *getShortLE*

```
public static short getShortLE( byte[] src, int srcOffset )
```

– **Description**

Reads two consecutive bytes from the given array at the given position in little endian order and returns them as a **short**.

– **Parameters**

- * **src** – the array from which two bytes are read
- * **srcOffset** – the index into the array from which the two bytes are read

– **Returns** – short value taken from the array

• *setIntBE*

```
public static void setIntBE( byte[] dest, int destOffset, int newValue )
```

– **Description**

Writes an int value into four consecutive bytes of a byte array, in big endian (network) byte order.

– **Parameters**

- * **dest** – the array to which bytes are written
 - * **destOffset** – index of the array to which the first byte is written
 - * **newValue** – the int value to be written to the array
-

• *setIntLE*

```
public static void setIntLE( byte[] dest, int destOffset, int newValue )
```

– **Description**

Writes an int value into four consecutive bytes of a byte array, in little endian (Intel) byte order.

– **Parameters**

- * **dest** – the array to which bytes are written
 - * **destOffset** – index of the array to which the first byte is written
 - * **newValue** – the int value to be written to the array
-

• *setShortBE*

```
public static void setShortBE( byte[] dest, int destOffset, short newValue )
```

• *setShortLE*

```
public static void setShortLE( byte[] dest, int destOffset, short newValue )
```

20.2.2 Class ArrayRotation

Provides static methods to rotate (in steps of 90 degrees), flip and mirror array elements. The image data is expected to be available as an array of integer values, being stored as rows top-to-bottom. Within each row, the data is laid out from left to right. This class may also be useful for transposing matrices.

The rotation by 90 and 270 degrees in-place (i.e., without using a second array to copy to) is based on ideas and code developed by others. See **Rotation of arrays** (at <http://facweb.cs.depaul.edu/tchristopher/rotates.htm>) by Thomas W. Christopher.

I also got very useful advice from Hans-Bernhard Broeker and others in **comp.graphics.algorithms** (at news:comp.graphics.algorithms). There is a thread titled In-place rotation of pixel images starting Oct 11, 2000.

Note: This class should be adjusted if Java ever supports genericity. Then rotation functionality could be provided for all kinds of arrays.

Declaration

```
public class ArrayRotation
extends java.lang.Object
```

Method summary

checkPixelArray(int[], int, int) This method checks several properties of the arguments.

flip(boolean, int[], int, int) Flips the image given by the arguments.

mirror(boolean, int[], int, int) Mirrors the image given by the arguments.

rotate180(boolean, int[], int, int) Rotates the argument image by 180 degrees.

rotate180(int, int, byte[], int, byte[], int)

rotate90Left(int, int, byte[], int, byte[], int)

rotate90Right(boolean, int[], int, int)

rotate90Right(int, int, byte[], int, byte[], int)

Methods

- *checkPixelArray*
 public static void **checkPixelArray**(int[] pixels, int width, int height)
 – **Description**
 This method checks several properties of the arguments. If any of the properties is not fulfilled, an explaining is thrown. Otherwise, nothing happens. This method is supposed to be called at the beginning of several other methods in this class.
 Properties checked:
 - * pixels is non-null
 - * width and height are larger than zero
 - * number of elements in pixels is at least width times height
- *flip*
 public static final int[] **flip**(boolean inPlace, int[] pixels, int width, int height)

- **Description**

Flips the image given by the arguments. The `inPlace` argument determines if the `pixels` array is modified or not. If `inPlace` is true, no additional array is allocated. Otherwise, an array of width times height items is allocated and the flipped image will be stored in this array.

- **Parameters**

- * `inPlace` – if true all work is done on the `pixels` array; otherwise, a second array is allocated and the `pixels` array remains unmodified
- * `pixels` – the array of pixels that form the image to be flipped
- * `width` – the horizontal resolution of the image; must be larger than 0
- * `height` – the vertical resolution of the image; must be larger than 0

- **Returns** – the flipped image as int array; equals `pixels` if `inPlace` is true

- **Throws**

- * `java.lang.IllegalArgumentException` – if the pixel resolution is invalid or the `pixels` array is not initialized or its length smaller than `width` times `height`

- *mirror*

```
public static int[] mirror( boolean inPlace, int[] pixels, int width, int
height )
```

- **Description**

Mirrors the image given by the arguments. For each row, pixels are swapped, leftmost and rightmost, second-leftmost and second-rightmost, and so on. The `inPlace` argument determines if the `pixels` array is modified or not. If `inPlace` is true, no additional array is used. Otherwise, an array of width times height items is allocated and the mirrored image will be stored in this array.

- **Parameters**

- * `inPlace` – if true all work is done on the `pixels` array; otherwise, a second array is allocated and the `pixels` array remains unmodified
- * `pixels` – the array of pixels that form the image to be flipped
- * `width` – the horizontal resolution of the image; must be larger than 0
- * `height` – the vertical resolution of the image; must be larger than 0

- **Returns** – the flipped image as int array; equals `pixels` if `inPlace` is true

- **Throws**

- * `java.lang.IllegalArgumentException` – if the pixel resolution is invalid or the `pixels` array is not initialized or its length smaller than `width` times `height`

- *rotate180*

```
public static int[] rotate180( boolean inPlace, int[] pixels, int width, int
height )
```

- **Description**

Rotates the argument image by 180 degrees. The resulting image will have exactly the same pixel resolution. Note that this operation is the same as two consecutive 90 degree rotations in the same direction. Another way of implementing a 180 degree rotation is first flipping and then mirroring the original image (or vice versa). If `inPlace` is true, the rotation is done on the argument `pixels` array. Otherwise a new array of sufficient length is allocated and the rotated image will be stored in this new array, not modifying the content of the `pixels` array.

- **Parameters**

- * `inPlace` – determines whether the rotated image is written to the argument array

- * **pixels** – the array of pixels that form the image to be rotated
- * **width** – the horizontal resolution of the image; must be larger than 0
- * **height** – the vertical resolution of the image; must be larger than 0
- **Returns** – the flipped image as int array; equals **pixels** if **inPlace** is true
- **Throws**
 - * **java.lang.IllegalArgumentException** – if the pixel resolution is invalid or the pixels array is not initialized or its length smaller than **width** times **height**

- *rotate180*

```
public static void rotate180( int width, int height, byte[] src, int  
srcOffset, byte[] dest, int destOffset )
```

- *rotate90Left*

```
public static void rotate90Left( int width, int height, byte[] src, int  
srcOffset, byte[] dest, int destOffset )
```

- *rotate90Right*

```
public static int[] rotate90Right( boolean inPlace, int[] pixels, int width,  
int height )
```

- *rotate90Right*

```
public static void rotate90Right( int width, int height, byte[] src, int  
srcOffset, byte[] dest, int destOffset )
```

20.2.3 Class ArrayScaling

This class currently only scales up an image given as a one-dimensional array of values.

Note: This class should be adjusted if Java ever supports genericity. It could then work on all kinds of arrays.

Declaration

```
public class ArrayScaling
extends java.lang.Object
```

Method summary

scaleUp200Percent(byte[], int, int) Scales up the argument image by factor 2 in both directions.

Methods

- *scaleUp200Percent*

```
public static final void scaleUp200Percent( byte[] data, int width, int
height ) throws java.lang.IllegalArgumentException
```

- **Description**

Scales up the argument image by factor 2 in both directions. It is assumed that the first `width` times `height` values of `data` contain an image (or image channel). The pixels (or samples) are assumed to be laid out rows top-to-bottom, within each row left-to-right. It is further assumed that the length of the `data` array is at least 4 times `width` times `height`. This method scales up the image in `data` so that after the call to this method `data` can be treated as an image (a channel) that has a horizontal resolution of `width * 2` and a vertical resolution of `height * 2`.

- **Parameters**

- * `data` – the array of pixels that form the image to be flipped
 - * `width` – the horizontal resolution of the image; must be larger than 0
 - * `height` – the vertical resolution of the image; must be larger than 0

- **Throws**

- * `java.lang.IllegalArgumentException` – if the arguments are invalid

20.2.4 Class Median

Pick the median value from an array (or an interval of an array).

Declaration

```
public class Median
extends java.lang.Object
```

Method summary

find(int[], int, int) Find the median value of the specified interval of the argument array.

swap(int[], int, int) Exchange two elements in the argument array.

Methods

- *find*

```
public static int find( int[] a, int from, int to )
```

- **Description**

Find the median value of the specified interval of the argument array. The interval starts at index **from** and goes to **to**; the values at these positions are included. Note that the array will be modified while searching, so you might want to backup your data. This implementation is a port of the C function from `quickselect.c`, provided at <http://ndevilla.free.fr/median/> (at <http://ndevilla.free.fr/median/>). The page is a good resource for various median value algorithms, including implementations and benchmarks.

The original code on which this class is based was written in C++ by Martin Leese. It was ported to C and optimized by Nicolas Devillard (author of the above mentioned page). The algorithm is from Numerical recipes in C, Second Edition, Cambridge University Press, 1992, Section 8.5, ISBN 0-521-43108-5.

- **Parameters**

- * **a** – the array
- * **from** – the index of the start of the interval in which the median value will be searched
- * **to** – the index of the end of the interval in which the median value will be searched

- **Returns** – the median value

- *swap*

```
public static void swap( int[] a, int i1, int i2 )
```

- **Description**

Exchange two elements in the argument array. A temporary variable is used so that `a[i1]` will hold the value that was previously stored at `a[i2]` and vice versa.

- **Parameters**

- * **a** – the array in which two elements are swapped
- * **i1** – index of the first element
- * **i2** – index of the second element

- **Throws**

* `java.lang.ArrayIndexOutOfBoundsException` – if either `i1` or `i2` are not valid index values into `a` (from 0 to `a.length - 1`)

20.2.5 Class *SeekableByteArrayOutputStream*

An extension of *OutputStream* that writes data to an internal byte array, resizing it when necessary. Similar to *ByteArrayOutputStream*, but also enables seeking and truncating.

Declaration

```
public class SeekableByteArrayOutputStream
    extends java.io.OutputStream
```

Constructor summary

- SeekableByteArrayOutputStream()*** Creates a new object of this class, setting initial capacity and increment size to default values.
- SeekableByteArrayOutputStream(int)*** Creates a new object of this class, setting initial capacity to the argument value.
- SeekableByteArrayOutputStream(int, int)*** Creates a new object of this class, setting initial capacity and increment to the argument values.

Method summary

- close()*** Closes this output stream.
- getPosition()*** Returns the current offset in the output stream.
- getSize()*** Returns the current size of the output stream.
- seek(int)*** Sets the current position in the output stream to the argument.
- toByteArray()*** Allocates a new `byte[]` object, copies (in 20.2.5, page 529) bytes from the internal byte array to that new array and returns the array.
- truncate()*** Removes all bytes after the current position.
- write(byte[])*** Write the complete argument array to this stream.
- write(byte[], int, int)*** Write some bytes from the argument array to this stream.
- write(int)*** Writes the least significant eight bits of the argument `int` to the internal array.
- writeTo(OutputStream)*** Writes the bytes in the internal byte array to the argument output stream.

Constructors

- *SeekableByteArrayOutputStream*
`public SeekableByteArrayOutputStream()`
 - **Description**
 Creates a new object of this class, setting initial capacity and increment size to default values.
- *SeekableByteArrayOutputStream*
`public SeekableByteArrayOutputStream(int initialCapacity)`
 - **Description**
 Creates a new object of this class, setting initial capacity to the argument value. The increment size is set to the initial capacity as well if that value is larger than 0. Otherwise it is set to a default value.

– **Parameters**

- * **initialCapacity** – the number of bytes that are allocated in this constructor (0 or larger)

• *SeekableByteArrayOutputStream*

public SeekableByteArrayOutputStream(int initialCapacity, int increment)

– **Description**

Creates a new object of this class, setting initial capacity and increment to the argument values.

– **Parameters**

- * **initialCapacity** – the number of bytes that are allocated in this constructor (0 or larger)
- * **increment** – the number of bytes by which the internal byte array is increased if it is full (1 or larger)

Methods

• *close*

public void close() throws java.io.IOException

– **Description**

Closes this output stream. After a call to this method, all write attempts will result in an exception.

• *getPosition*

public int getPosition()

– **Description**

Returns the current offset in the output stream. Larger than or equal to 0 and smaller than or equal to (in 20.2.5, page 529).

– **Returns** – current position in the output stream, 0-based

• *getSize*

public int getSize()

– **Description**

Returns the current size of the output stream.

– **Returns** – size of the output stream in bytes (0 or larger)

• *seek*

public void seek(int newOffset) throws java.io.IOException

– **Description**

Sets the current position in the output stream to the argument.

– **Parameters**

- * **newOffset** – new offset into the file, must be ≥ 0 and \leq (in 20.2.5, page 529)

– **Throws**

- * **java.io.IOException** – if the argument is invalid
-

- *toByteArray*

public byte[] toByteArray()

- **Description**

Allocates a new `byte[]` object, copies (in 20.2.5, page 529) bytes from the internal byte array to that new array and returns the array.

- **Returns** – a copy of the `byte[]` data stored internally

- *truncate*

public void truncate()

- **Description**

Removes all bytes after the current position. After a call to this method, (in 20.2.5, page 529) is equal to (in 20.2.5, page 529).

- *write*

public void write(byte[] data) throws java.io.IOException

- **Description**

Write the complete argument array to this stream. Copies the data to the internal byte array. Simply calls `write(data, 0, data.length);`.

- **Parameters**

* `data` – array to be copied to this stream

- *write*

public void write(byte[] src, int srcOffset, int num) throws java.io.IOException

- **Description**

Write some bytes from the argument array to this stream. Copies `num` bytes starting at `src[srcOffset]` to this stream.

- **Parameters**

* `src` – the array from which data is copied

* `srcOffset` – int index into that array pointing to the first byte to be copied

* `num` – number of bytes to be copied

- *write*

public void write(int b) throws java.io.IOException

- **Description**

Writes the least significant eight bits of the argument `int` to the internal array.

- **Parameters**

* `b` – int variable that stores the byte value to be written

- *writeTo*

public void writeTo(java.io.OutputStream out) throws java.io.IOException

- **Description**

Writes the bytes in the internal byte array to the argument output stream. A call to this method has the same effect as

```
byte[] copy = toByteArray();
```

```
out.write(copy, 0, copy.length);
```

However, you with this method you save the allocation of an additional byte array and the copying to that new array.

– **Parameters**

* `out` – the output stream to which this stream's content is copied

– **Throws**

* `java.io.IOException` – if `out` has a problem writing the bytes

20.2.6 Class Sort

Provides sorting of an Object array.

Declaration

```
public class Sort
extends java.lang.Object
```

Method summary

sort(Object[], ComparatorInterface) Sort the complete argument array according to the argument comparator.

sort(Object[], int, int, ComparatorInterface) Sorts some (or all) elements of an Object array according to a specified comparator.

Methods

- *sort*

```
public static void sort( java.lang.Object[] a, ComparatorInterface
comparator )
```

- **Description**

Sort the complete argument array according to the argument comparator. Simply calls `sort(a, 0, a.length - 1, comparator);`

- **Parameters**

- * **a** – array to be sorted
 - * **comparator** – the comparator used to compare to array entries
-

- *sort*

```
public static void sort( java.lang.Object[] a, int from, int to,
ComparatorInterface c )
```

- **Description**

Sorts some (or all) elements of an Object array according to a specified comparator. This method does exactly the same as `java.util.Arrays.sort(Object[], int, int, Comparator)`. Unfortunately, this method is not available in Java 1.1, so it must be provided here.

As for the implementation of this method, it is taken from `Arrays.java` as found in Classpath 0.0.2 (2001-01-06). Go to **www.classpath.org** (at <http://www.classpath.org>) to learn more about the project, which implements the Java core libraries under the GPL.

- **Parameters**

- * **a** – the array which is to be sorted
- * **from** – the index value of the first element of the interval to be sorted
- * **to** – the index value of the last element of the interval to be sorted
- * **c** – the comparator used to query the relation between two objects

20.2.7 Class Statistics

A number of static methods to compute statistical properties of an array of double values. Implements the computation of mean, variance and standard deviation for double values.

Declaration

```
public class Statistics
extends java.lang.Object
```

Method summary

computeMean(double[]) Computes the mean value for the argument array.
computeMean(double[], int, int) Computes the mean value for some elements of the argument array.
computeStandardDeviation(double[]) Computes the standard deviation for the argument array of values.
computeStandardDeviation(double[], double) Computes the standard deviation for the argument array of values.
computeStandardDeviation(double[], int, int) Computes the standard deviation for some of the argument array's values.
computeStandardDeviation(double[], int, int, double) Computes the standard deviation for some of the argument array's values.
computeVariance(double[]) Computes the variance for the argument array.
computeVariance(double[], double) Computes the variance for some of the argument array's values.
computeVariance(double[], int, int) Computes the variance for some of the argument array's values.
computeVariance(double[], int, int, double) Computes the variance for some of the argument array's values.

Methods

- *computeMean*
 public static double **computeMean**(double[] values)
 - **Description**
 Computes the mean value for the argument array. Adds all values and divides them by the number of array elements.
 - **Parameters**
 * values – double array on which the mean is to be determined
 - **Returns** – computed mean value
 - **Throws**
 * java.lang.IllegalArgumentException – if the array has not at least one element
- *computeMean*
 public static double **computeMean**(double[] values, int offset, int number)

– **Description**

Computes the mean value for some elements of the argument array. Adds all values and divides them by the number of array elements.

– **Parameters**

- * **values** – array from which elements are read
- * **offset** – index of the first element to be used
- * **number** – number of elements to be used

– **Returns** – computed mean value

– **Throws**

- * `java.lang.IllegalArgumentException` – if the array has not at least one element
-

• *computeStandardDeviation*

```
public static double computeStandardDeviation( double[] values )
```

– **Description**

Computes the standard deviation for the argument array of values.

– **Parameters**

- * **values** – array from which elements are read

– **Returns** – computed standard deviation

– **Throws**

- * `java.lang.IllegalArgumentException` – if the array has not at least two elements
-

• *computeStandardDeviation*

```
public static double computeStandardDeviation( double[] values, double
mean )
```

– **Description**

Computes the standard deviation for the argument array of values. Reuses the mean value for that argument which must have been computed before.

– **Parameters**

- * **values** – array from which elements are read
- * **mean** – the mean value for the array, possibly computed with a call to `computeMean` (in 20.2.7, page 533).

– **Returns** – computed standard deviation

– **Throws**

- * `java.lang.IllegalArgumentException` – if the array has not at least two elements
-

• *computeStandardDeviation*

```
public static double computeStandardDeviation( double[] values, int offset,
int number )
```

– **Description**

Computes the standard deviation for some of the argument array's values. If you already have computed a mean value using `computeMean` (in 20.2.7, page 533), better call `computeMean` (in 20.2.7, page 535). Otherwise, this method has to compute mean again.

– **Parameters**

- * **values** – array from which elements are read
- * **offset** – first element to be used

- * **number** – number of elements used starting at values[offset]
 - **Returns** – computed standard deviation
 - **Throws**
 - * `java.lang.IllegalArgumentException` – if the array has not at least two elements
-

- *computeStandardDeviation*

```
public static double computeStandardDeviation( double[] values, int offset,
int number, double mean )
```

- **Description**

Computes the standard deviation for some of the argument array's values. Use this version of the method if you already have a mean value, otherwise this method must be computed again.
 - **Parameters**
 - * **values** – array from which elements are read
 - * **offset** – first element to be used
 - * **number** – number of elements used starting at values[offset]
 - * **mean** – value of the elements
 - **Returns** – computed standard deviation
 - **Throws**
 - * `java.lang.IllegalArgumentException` – if the array has not at least two elements
-

- *computeVariance*

```
public static double computeVariance( double[] values )
```

- **Description**

Computes the variance for the argument array.
 - **Parameters**
 - * **values** – array from which elements are read
 - **Returns** – variance for the array elements
 - **Throws**
 - * `java.lang.IllegalArgumentException` – if the array has not at least two elements
-

- *computeVariance*

```
public static double computeVariance( double[] values, double mean )
```

- **Description**

Computes the variance for some of the argument array's values.
 - **Parameters**
 - * **values** – array from which elements are read
 - * **mean** – the mean for the array elements
 - **Returns** – variance for the array elements
 - **Throws**
 - * `java.lang.IllegalArgumentException` – if the array has not at least two elements
-

- *computeVariance*

```
public static double computeVariance( double[] values, int offset, int
number )
```

- **Description**

Computes the variance for some of the argument array's values. If you already have computed a mean value using (in 20.2.7, page 533), better call (in 20.2.7, page 536). Otherwise, this method has to compute mean again.

- **Parameters**

- * **values** – array from which elements are read
 - * **offset** – first element to be used
 - * **number** – number of elements used starting at values[offset]

- **Returns** – computed variance

- **Throws**

- * **java.lang.IllegalArgumentException** – if the array has not at least two elements
-

- *computeVariance*

```
public static double computeVariance( double[] values, int offset, int
number, double mean )
```

- **Description**

Computes the variance for some of the argument array's values. Use this version of the method in case mean has already been computed.

- **Parameters**

- * **values** – array from which elements are read
 - * **offset** – first element to be used
 - * **number** – number of elements used starting at values[offset]
 - * **mean** – the mean for the array elements

- **Returns** – computed variance

- **Throws**

- * **java.lang.IllegalArgumentException** – if the array has not at least two elements

20.2.8 Class SystemInfo

Class to retrieve system information in a human-readable form.

Declaration

```
public class SystemInfo
  extends java.lang.Object
  implements net.sourceforge.jiu.apps.StringIndexConstants
```

Method summary

getMemoryInfo(Strings)

getSystemInfo(Strings) Returns a multiple-line text with information on the Java Virtual Machine, the path settings and the operating system used, regarding the current language by using a (in 1.2.12, page 61)resource.

Methods

- *getMemoryInfo*

```
public static java.lang.String getMemoryInfo(
  net.sourceforge.jiu.apps.Strings strings )
```

- *getSystemInfo*

```
public static java.lang.String getSystemInfo(
  net.sourceforge.jiu.apps.Strings strings )
```

- **Description**

Returns a multiple-line text with information on the Java Virtual Machine, the path settings and the operating system used, regarding the current language by using a (in 1.2.12, page 61)resource.

- **Returns** – system information as String

- **See also**

* `java.lang.System.getProperty(java.lang.String)`