| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/BufferedImage.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/java/awt/image/BandedSampleModel.html)   [**NEXT CLASS**](http://docs.google.com/java/awt/image/BufferedImageFilter.html) | [**FRAMES**](http://docs.google.com/index.html?java/awt/image/BufferedImage.html)    [**NO FRAMES**](http://docs.google.com/BufferedImage.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | [CONSTR](#3dy6vkm) | [METHOD](#1t3h5sf) | DETAIL: [FIELD](#17dp8vu) | [CONSTR](#2bn6wsx) | [METHOD](#49x2ik5) |

## **java.awt.image**

Class BufferedImage

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 [java.awt.Image](http://docs.google.com/java/awt/Image.html)  
 **java.awt.image.BufferedImage**

**All Implemented Interfaces:** [RenderedImage](http://docs.google.com/java/awt/image/RenderedImage.html), [WritableRenderedImage](http://docs.google.com/java/awt/image/WritableRenderedImage.html), [Transparency](http://docs.google.com/java/awt/Transparency.html)

public class **BufferedImage**extends [Image](http://docs.google.com/java/awt/Image.html)implements [WritableRenderedImage](http://docs.google.com/java/awt/image/WritableRenderedImage.html), [Transparency](http://docs.google.com/java/awt/Transparency.html)

The BufferedImage subclass describes an [Image](http://docs.google.com/java/awt/Image.html) with an accessible buffer of image data. A BufferedImage is comprised of a [ColorModel](http://docs.google.com/java/awt/image/ColorModel.html) and a [Raster](http://docs.google.com/java/awt/image/Raster.html) of image data. The number and types of bands in the [SampleModel](http://docs.google.com/java/awt/image/SampleModel.html) of the Raster must match the number and types required by the ColorModel to represent its color and alpha components. All BufferedImage objects have an upper left corner coordinate of (0, 0). Any Raster used to construct a BufferedImage must therefore have minX=0 and minY=0.

This class relies on the data fetching and setting methods of Raster, and on the color characterization methods of ColorModel.

**See Also:**[ColorModel](http://docs.google.com/java/awt/image/ColorModel.html), [Raster](http://docs.google.com/java/awt/image/Raster.html), [WritableRaster](http://docs.google.com/java/awt/image/WritableRaster.html)

| **Field Summary** | |
| --- | --- |
| static int | [**TYPE\_3BYTE\_BGR**](http://docs.google.com/java/awt/image/BufferedImage.html#TYPE_3BYTE_BGR)            Represents an image with 8-bit RGB color components, corresponding to a Windows-style BGR color model) with the colors Blue, Green, and Red stored in 3 bytes. |
| static int | [**TYPE\_4BYTE\_ABGR**](http://docs.google.com/java/awt/image/BufferedImage.html#TYPE_4BYTE_ABGR)            Represents an image with 8-bit RGBA color components with the colors Blue, Green, and Red stored in 3 bytes and 1 byte of alpha. |
| static int | [**TYPE\_4BYTE\_ABGR\_PRE**](http://docs.google.com/java/awt/image/BufferedImage.html#TYPE_4BYTE_ABGR_PRE)            Represents an image with 8-bit RGBA color components with the colors Blue, Green, and Red stored in 3 bytes and 1 byte of alpha. |
| static int | [**TYPE\_BYTE\_BINARY**](http://docs.google.com/java/awt/image/BufferedImage.html#TYPE_BYTE_BINARY)            Represents an opaque byte-packed 1, 2, or 4 bit image. |
| static int | [**TYPE\_BYTE\_GRAY**](http://docs.google.com/java/awt/image/BufferedImage.html#TYPE_BYTE_GRAY)            Represents a unsigned byte grayscale image, non-indexed. |
| static int | [**TYPE\_BYTE\_INDEXED**](http://docs.google.com/java/awt/image/BufferedImage.html#TYPE_BYTE_INDEXED)            Represents an indexed byte image. |
| static int | [**TYPE\_CUSTOM**](http://docs.google.com/java/awt/image/BufferedImage.html#TYPE_CUSTOM)            Image type is not recognized so it must be a customized image. |
| static int | [**TYPE\_INT\_ARGB**](http://docs.google.com/java/awt/image/BufferedImage.html#TYPE_INT_ARGB)            Represents an image with 8-bit RGBA color components packed into integer pixels. |
| static int | [**TYPE\_INT\_ARGB\_PRE**](http://docs.google.com/java/awt/image/BufferedImage.html#TYPE_INT_ARGB_PRE)            Represents an image with 8-bit RGBA color components packed into integer pixels. |
| static int | [**TYPE\_INT\_BGR**](http://docs.google.com/java/awt/image/BufferedImage.html#TYPE_INT_BGR)            Represents an image with 8-bit RGB color components, corresponding to a Windows- or Solaris- style BGR color model, with the colors Blue, Green, and Red packed into integer pixels. |
| static int | [**TYPE\_INT\_RGB**](http://docs.google.com/java/awt/image/BufferedImage.html#TYPE_INT_RGB)            Represents an image with 8-bit RGB color components packed into integer pixels. |
| static int | [**TYPE\_USHORT\_555\_RGB**](http://docs.google.com/java/awt/image/BufferedImage.html#TYPE_USHORT_555_RGB)            Represents an image with 5-5-5 RGB color components (5-bits red, 5-bits green, 5-bits blue) with no alpha. |
| static int | [**TYPE\_USHORT\_565\_RGB**](http://docs.google.com/java/awt/image/BufferedImage.html#TYPE_USHORT_565_RGB)            Represents an image with 5-6-5 RGB color components (5-bits red, 6-bits green, 5-bits blue) with no alpha. |
| static int | [**TYPE\_USHORT\_GRAY**](http://docs.google.com/java/awt/image/BufferedImage.html#TYPE_USHORT_GRAY)            Represents an unsigned short grayscale image, non-indexed). |

| **Fields inherited from class java.awt.**[**Image**](http://docs.google.com/java/awt/Image.html) |
| --- |
| [accelerationPriority](http://docs.google.com/java/awt/Image.html#accelerationPriority), [SCALE\_AREA\_AVERAGING](http://docs.google.com/java/awt/Image.html#SCALE_AREA_AVERAGING), [SCALE\_DEFAULT](http://docs.google.com/java/awt/Image.html#SCALE_DEFAULT), [SCALE\_FAST](http://docs.google.com/java/awt/Image.html#SCALE_FAST), [SCALE\_REPLICATE](http://docs.google.com/java/awt/Image.html#SCALE_REPLICATE), [SCALE\_SMOOTH](http://docs.google.com/java/awt/Image.html#SCALE_SMOOTH), [UndefinedProperty](http://docs.google.com/java/awt/Image.html#UndefinedProperty) |

| **Fields inherited from interface java.awt.**[**Transparency**](http://docs.google.com/java/awt/Transparency.html) |
| --- |
| [BITMASK](http://docs.google.com/java/awt/Transparency.html#BITMASK), [OPAQUE](http://docs.google.com/java/awt/Transparency.html#OPAQUE), [TRANSLUCENT](http://docs.google.com/java/awt/Transparency.html#TRANSLUCENT) |

| **Constructor Summary** | |
| --- | --- |
| [**BufferedImage**](http://docs.google.com/java/awt/image/BufferedImage.html#BufferedImage(java.awt.image.ColorModel,%20java.awt.image.WritableRaster,%20boolean,%20java.util.Hashtable))([ColorModel](http://docs.google.com/java/awt/image/ColorModel.html) cm, [WritableRaster](http://docs.google.com/java/awt/image/WritableRaster.html) raster, boolean isRasterPremultiplied, [Hashtable](http://docs.google.com/java/util/Hashtable.html)<?,?> properties)            Constructs a new BufferedImage with a specified ColorModel and Raster. |
| [**BufferedImage**](http://docs.google.com/java/awt/image/BufferedImage.html#BufferedImage(int,%20int,%20int))(int width, int height, int imageType)            Constructs a BufferedImage of one of the predefined image types. |
| [**BufferedImage**](http://docs.google.com/java/awt/image/BufferedImage.html#BufferedImage(int,%20int,%20int,%20java.awt.image.IndexColorModel))(int width, int height, int imageType, [IndexColorModel](http://docs.google.com/java/awt/image/IndexColorModel.html) cm)            Constructs a BufferedImage of one of the predefined image types: TYPE\_BYTE\_BINARY or TYPE\_BYTE\_INDEXED. |

| **Method Summary** | |
| --- | --- |
| void | [**addTileObserver**](http://docs.google.com/java/awt/image/BufferedImage.html#addTileObserver(java.awt.image.TileObserver))([TileObserver](http://docs.google.com/java/awt/image/TileObserver.html) to)            Adds a tile observer. |
| void | [**coerceData**](http://docs.google.com/java/awt/image/BufferedImage.html#coerceData(boolean))(boolean isAlphaPremultiplied)            Forces the data to match the state specified in the isAlphaPremultiplied variable. |
| [WritableRaster](http://docs.google.com/java/awt/image/WritableRaster.html) | [**copyData**](http://docs.google.com/java/awt/image/BufferedImage.html#copyData(java.awt.image.WritableRaster))([WritableRaster](http://docs.google.com/java/awt/image/WritableRaster.html) outRaster)            Computes an arbitrary rectangular region of the BufferedImage and copies it into a specified WritableRaster. |
| [Graphics2D](http://docs.google.com/java/awt/Graphics2D.html) | [**createGraphics**](http://docs.google.com/java/awt/image/BufferedImage.html#createGraphics())()            Creates a Graphics2D, which can be used to draw into this BufferedImage. |
| [WritableRaster](http://docs.google.com/java/awt/image/WritableRaster.html) | [**getAlphaRaster**](http://docs.google.com/java/awt/image/BufferedImage.html#getAlphaRaster())()            Returns a WritableRaster representing the alpha channel for BufferedImage objects with ColorModel objects that support a separate spatial alpha channel, such as ComponentColorModel and DirectColorModel. |
| [ColorModel](http://docs.google.com/java/awt/image/ColorModel.html) | [**getColorModel**](http://docs.google.com/java/awt/image/BufferedImage.html#getColorModel())()            Returns the ColorModel. |
| [Raster](http://docs.google.com/java/awt/image/Raster.html) | [**getData**](http://docs.google.com/java/awt/image/BufferedImage.html#getData())()            Returns the image as one large tile. |
| [Raster](http://docs.google.com/java/awt/image/Raster.html) | [**getData**](http://docs.google.com/java/awt/image/BufferedImage.html#getData(java.awt.Rectangle))([Rectangle](http://docs.google.com/java/awt/Rectangle.html) rect)            Computes and returns an arbitrary region of the BufferedImage. |
| [Graphics](http://docs.google.com/java/awt/Graphics.html) | [**getGraphics**](http://docs.google.com/java/awt/image/BufferedImage.html#getGraphics())()            This method returns a [Graphics2D](http://docs.google.com/java/awt/Graphics2D.html), but is here for backwards compatibility. |
| int | [**getHeight**](http://docs.google.com/java/awt/image/BufferedImage.html#getHeight())()            Returns the height of the BufferedImage. |
| int | [**getHeight**](http://docs.google.com/java/awt/image/BufferedImage.html#getHeight(java.awt.image.ImageObserver))([ImageObserver](http://docs.google.com/java/awt/image/ImageObserver.html) observer)            Returns the height of the BufferedImage. |
| int | [**getMinTileX**](http://docs.google.com/java/awt/image/BufferedImage.html#getMinTileX())()            Returns the minimum tile index in the x direction. |
| int | [**getMinTileY**](http://docs.google.com/java/awt/image/BufferedImage.html#getMinTileY())()            Returns the minimum tile index in the y direction. |
| int | [**getMinX**](http://docs.google.com/java/awt/image/BufferedImage.html#getMinX())()            Returns the minimum x coordinate of this BufferedImage. |
| int | [**getMinY**](http://docs.google.com/java/awt/image/BufferedImage.html#getMinY())()            Returns the minimum y coordinate of this BufferedImage. |
| int | [**getNumXTiles**](http://docs.google.com/java/awt/image/BufferedImage.html#getNumXTiles())()            Returns the number of tiles in the x direction. |
| int | [**getNumYTiles**](http://docs.google.com/java/awt/image/BufferedImage.html#getNumYTiles())()            Returns the number of tiles in the y direction. |
| [Object](http://docs.google.com/java/lang/Object.html) | [**getProperty**](http://docs.google.com/java/awt/image/BufferedImage.html#getProperty(java.lang.String))([String](http://docs.google.com/java/lang/String.html) name)            Returns a property of the image by name. |
| [Object](http://docs.google.com/java/lang/Object.html) | [**getProperty**](http://docs.google.com/java/awt/image/BufferedImage.html#getProperty(java.lang.String,%20java.awt.image.ImageObserver))([String](http://docs.google.com/java/lang/String.html) name, [ImageObserver](http://docs.google.com/java/awt/image/ImageObserver.html) observer)            Returns a property of the image by name. |
| [String](http://docs.google.com/java/lang/String.html)[] | [**getPropertyNames**](http://docs.google.com/java/awt/image/BufferedImage.html#getPropertyNames())()            Returns an array of names recognized by [getProperty(String)](http://docs.google.com/java/awt/image/BufferedImage.html#getProperty(java.lang.String)) or null, if no property names are recognized. |
| [WritableRaster](http://docs.google.com/java/awt/image/WritableRaster.html) | [**getRaster**](http://docs.google.com/java/awt/image/BufferedImage.html#getRaster())()            Returns the [WritableRaster](http://docs.google.com/java/awt/image/WritableRaster.html). |
| int | [**getRGB**](http://docs.google.com/java/awt/image/BufferedImage.html#getRGB(int,%20int))(int x, int y)            Returns an integer pixel in the default RGB color model (TYPE\_INT\_ARGB) and default sRGB colorspace. |
| int[] | [**getRGB**](http://docs.google.com/java/awt/image/BufferedImage.html#getRGB(int,%20int,%20int,%20int,%20int%5B%5D,%20int,%20int))(int startX, int startY, int w, int h, int[] rgbArray, int offset, int scansize)            Returns an array of integer pixels in the default RGB color model (TYPE\_INT\_ARGB) and default sRGB color space, from a portion of the image data. |
| [SampleModel](http://docs.google.com/java/awt/image/SampleModel.html) | [**getSampleModel**](http://docs.google.com/java/awt/image/BufferedImage.html#getSampleModel())()            Returns the SampleModel associated with this BufferedImage. |
| [ImageProducer](http://docs.google.com/java/awt/image/ImageProducer.html) | [**getSource**](http://docs.google.com/java/awt/image/BufferedImage.html#getSource())()            Returns the object that produces the pixels for the image. |
| [Vector](http://docs.google.com/java/util/Vector.html)<[RenderedImage](http://docs.google.com/java/awt/image/RenderedImage.html)> | [**getSources**](http://docs.google.com/java/awt/image/BufferedImage.html#getSources())()            Returns a [Vector](http://docs.google.com/java/util/Vector.html) of [RenderedImage](http://docs.google.com/java/awt/image/RenderedImage.html) objects that are the immediate sources, not the sources of these immediate sources, of image data for this BufferedImage. |
| [BufferedImage](http://docs.google.com/java/awt/image/BufferedImage.html) | [**getSubimage**](http://docs.google.com/java/awt/image/BufferedImage.html#getSubimage(int,%20int,%20int,%20int))(int x, int y, int w, int h)            Returns a subimage defined by a specified rectangular region. |
| [Raster](http://docs.google.com/java/awt/image/Raster.html) | [**getTile**](http://docs.google.com/java/awt/image/BufferedImage.html#getTile(int,%20int))(int tileX, int tileY)            Returns tile (tileX, tileY). |
| int | [**getTileGridXOffset**](http://docs.google.com/java/awt/image/BufferedImage.html#getTileGridXOffset())()            Returns the x offset of the tile grid relative to the origin, For example, the x coordinate of the location of tile (0, 0). |
| int | [**getTileGridYOffset**](http://docs.google.com/java/awt/image/BufferedImage.html#getTileGridYOffset())()            Returns the y offset of the tile grid relative to the origin, For example, the y coordinate of the location of tile (0, 0). |
| int | [**getTileHeight**](http://docs.google.com/java/awt/image/BufferedImage.html#getTileHeight())()            Returns the tile height in pixels. |
| int | [**getTileWidth**](http://docs.google.com/java/awt/image/BufferedImage.html#getTileWidth())()            Returns the tile width in pixels. |
| int | [**getTransparency**](http://docs.google.com/java/awt/image/BufferedImage.html#getTransparency())()            Returns the transparency. |
| int | [**getType**](http://docs.google.com/java/awt/image/BufferedImage.html#getType())()            Returns the image type. |
| int | [**getWidth**](http://docs.google.com/java/awt/image/BufferedImage.html#getWidth())()            Returns the width of the BufferedImage. |
| int | [**getWidth**](http://docs.google.com/java/awt/image/BufferedImage.html#getWidth(java.awt.image.ImageObserver))([ImageObserver](http://docs.google.com/java/awt/image/ImageObserver.html) observer)            Returns the width of the BufferedImage. |
| [WritableRaster](http://docs.google.com/java/awt/image/WritableRaster.html) | [**getWritableTile**](http://docs.google.com/java/awt/image/BufferedImage.html#getWritableTile(int,%20int))(int tileX, int tileY)            Checks out a tile for writing. |
| [Point](http://docs.google.com/java/awt/Point.html)[] | [**getWritableTileIndices**](http://docs.google.com/java/awt/image/BufferedImage.html#getWritableTileIndices())()            Returns an array of [Point](http://docs.google.com/java/awt/Point.html) objects indicating which tiles are checked out for writing. |
| boolean | [**hasTileWriters**](http://docs.google.com/java/awt/image/BufferedImage.html#hasTileWriters())()            Returns whether or not any tile is checked out for writing. |
| boolean | [**isAlphaPremultiplied**](http://docs.google.com/java/awt/image/BufferedImage.html#isAlphaPremultiplied())()            Returns whether or not the alpha has been premultiplied. |
| boolean | [**isTileWritable**](http://docs.google.com/java/awt/image/BufferedImage.html#isTileWritable(int,%20int))(int tileX, int tileY)            Returns whether or not a tile is currently checked out for writing. |
| void | [**releaseWritableTile**](http://docs.google.com/java/awt/image/BufferedImage.html#releaseWritableTile(int,%20int))(int tileX, int tileY)            Relinquishes permission to write to a tile. |
| void | [**removeTileObserver**](http://docs.google.com/java/awt/image/BufferedImage.html#removeTileObserver(java.awt.image.TileObserver))([TileObserver](http://docs.google.com/java/awt/image/TileObserver.html) to)            Removes a tile observer. |
| void | [**setData**](http://docs.google.com/java/awt/image/BufferedImage.html#setData(java.awt.image.Raster))([Raster](http://docs.google.com/java/awt/image/Raster.html) r)            Sets a rectangular region of the image to the contents of the specified Raster r, which is assumed to be in the same coordinate space as the BufferedImage. |
| void | [**setRGB**](http://docs.google.com/java/awt/image/BufferedImage.html#setRGB(int,%20int,%20int))(int x, int y, int rgb)            Sets a pixel in this BufferedImage to the specified RGB value. |
| void | [**setRGB**](http://docs.google.com/java/awt/image/BufferedImage.html#setRGB(int,%20int,%20int,%20int,%20int%5B%5D,%20int,%20int))(int startX, int startY, int w, int h, int[] rgbArray, int offset, int scansize)            Sets an array of integer pixels in the default RGB color model (TYPE\_INT\_ARGB) and default sRGB color space, into a portion of the image data. |
| [String](http://docs.google.com/java/lang/String.html) | [**toString**](http://docs.google.com/java/awt/image/BufferedImage.html#toString())()            Returns a String representation of this BufferedImage object and its values. |

| **Methods inherited from class java.awt.**[**Image**](http://docs.google.com/java/awt/Image.html) |
| --- |
| [flush](http://docs.google.com/java/awt/Image.html#flush()), [getAccelerationPriority](http://docs.google.com/java/awt/Image.html#getAccelerationPriority()), [getCapabilities](http://docs.google.com/java/awt/Image.html#getCapabilities(java.awt.GraphicsConfiguration)), [getScaledInstance](http://docs.google.com/java/awt/Image.html#getScaledInstance(int,%20int,%20int)), [setAccelerationPriority](http://docs.google.com/java/awt/Image.html#setAccelerationPriority(float)) |

| **Methods inherited from class java.lang.**[**Object**](http://docs.google.com/java/lang/Object.html) |
| --- |
| [clone](http://docs.google.com/java/lang/Object.html#clone()), [equals](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)), [finalize](http://docs.google.com/java/lang/Object.html#finalize()), [getClass](http://docs.google.com/java/lang/Object.html#getClass()), [hashCode](http://docs.google.com/java/lang/Object.html#hashCode()), [notify](http://docs.google.com/java/lang/Object.html#notify()), [notifyAll](http://docs.google.com/java/lang/Object.html#notifyAll()), [wait](http://docs.google.com/java/lang/Object.html#wait()), [wait](http://docs.google.com/java/lang/Object.html#wait(long)), [wait](http://docs.google.com/java/lang/Object.html#wait(long,%20int)) |

| **Field Detail** |
| --- |

### TYPE\_CUSTOM

public static final int **TYPE\_CUSTOM**

Image type is not recognized so it must be a customized image. This type is only used as a return value for the getType() method.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.awt.image.BufferedImage.TYPE_CUSTOM)

### TYPE\_INT\_RGB

public static final int **TYPE\_INT\_RGB**

Represents an image with 8-bit RGB color components packed into integer pixels. The image has a [DirectColorModel](http://docs.google.com/java/awt/image/DirectColorModel.html) without alpha. When data with non-opaque alpha is stored in an image of this type, the color data must be adjusted to a non-premultiplied form and the alpha discarded, as described in the [AlphaComposite](http://docs.google.com/java/awt/AlphaComposite.html) documentation.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.awt.image.BufferedImage.TYPE_INT_RGB)

### TYPE\_INT\_ARGB

public static final int **TYPE\_INT\_ARGB**

Represents an image with 8-bit RGBA color components packed into integer pixels. The image has a DirectColorModel with alpha. The color data in this image is considered not to be premultiplied with alpha. When this type is used as the imageType argument to a BufferedImage constructor, the created image is consistent with images created in the JDK1.1 and earlier releases.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.awt.image.BufferedImage.TYPE_INT_ARGB)

### TYPE\_INT\_ARGB\_PRE

public static final int **TYPE\_INT\_ARGB\_PRE**

Represents an image with 8-bit RGBA color components packed into integer pixels. The image has a DirectColorModel with alpha. The color data in this image is considered to be premultiplied with alpha.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.awt.image.BufferedImage.TYPE_INT_ARGB_PRE)

### TYPE\_INT\_BGR

public static final int **TYPE\_INT\_BGR**

Represents an image with 8-bit RGB color components, corresponding to a Windows- or Solaris- style BGR color model, with the colors Blue, Green, and Red packed into integer pixels. There is no alpha. The image has a [DirectColorModel](http://docs.google.com/java/awt/image/DirectColorModel.html). When data with non-opaque alpha is stored in an image of this type, the color data must be adjusted to a non-premultiplied form and the alpha discarded, as described in the [AlphaComposite](http://docs.google.com/java/awt/AlphaComposite.html) documentation.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.awt.image.BufferedImage.TYPE_INT_BGR)

### TYPE\_3BYTE\_BGR

public static final int **TYPE\_3BYTE\_BGR**

Represents an image with 8-bit RGB color components, corresponding to a Windows-style BGR color model) with the colors Blue, Green, and Red stored in 3 bytes. There is no alpha. The image has a ComponentColorModel. When data with non-opaque alpha is stored in an image of this type, the color data must be adjusted to a non-premultiplied form and the alpha discarded, as described in the [AlphaComposite](http://docs.google.com/java/awt/AlphaComposite.html) documentation.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.awt.image.BufferedImage.TYPE_3BYTE_BGR)

### TYPE\_4BYTE\_ABGR

public static final int **TYPE\_4BYTE\_ABGR**

Represents an image with 8-bit RGBA color components with the colors Blue, Green, and Red stored in 3 bytes and 1 byte of alpha. The image has a ComponentColorModel with alpha. The color data in this image is considered not to be premultiplied with alpha. The byte data is interleaved in a single byte array in the order A, B, G, R from lower to higher byte addresses within each pixel.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.awt.image.BufferedImage.TYPE_4BYTE_ABGR)

### TYPE\_4BYTE\_ABGR\_PRE

public static final int **TYPE\_4BYTE\_ABGR\_PRE**

Represents an image with 8-bit RGBA color components with the colors Blue, Green, and Red stored in 3 bytes and 1 byte of alpha. The image has a ComponentColorModel with alpha. The color data in this image is considered to be premultiplied with alpha. The byte data is interleaved in a single byte array in the order A, B, G, R from lower to higher byte addresses within each pixel.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.awt.image.BufferedImage.TYPE_4BYTE_ABGR_PRE)

### TYPE\_USHORT\_565\_RGB

public static final int **TYPE\_USHORT\_565\_RGB**

Represents an image with 5-6-5 RGB color components (5-bits red, 6-bits green, 5-bits blue) with no alpha. This image has a DirectColorModel. When data with non-opaque alpha is stored in an image of this type, the color data must be adjusted to a non-premultiplied form and the alpha discarded, as described in the [AlphaComposite](http://docs.google.com/java/awt/AlphaComposite.html) documentation.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.awt.image.BufferedImage.TYPE_USHORT_565_RGB)

### TYPE\_USHORT\_555\_RGB

public static final int **TYPE\_USHORT\_555\_RGB**

Represents an image with 5-5-5 RGB color components (5-bits red, 5-bits green, 5-bits blue) with no alpha. This image has a DirectColorModel. When data with non-opaque alpha is stored in an image of this type, the color data must be adjusted to a non-premultiplied form and the alpha discarded, as described in the [AlphaComposite](http://docs.google.com/java/awt/AlphaComposite.html) documentation.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.awt.image.BufferedImage.TYPE_USHORT_555_RGB)

### TYPE\_BYTE\_GRAY

public static final int **TYPE\_BYTE\_GRAY**

Represents a unsigned byte grayscale image, non-indexed. This image has a ComponentColorModel with a CS\_GRAY [ColorSpace](http://docs.google.com/java/awt/color/ColorSpace.html). When data with non-opaque alpha is stored in an image of this type, the color data must be adjusted to a non-premultiplied form and the alpha discarded, as described in the [AlphaComposite](http://docs.google.com/java/awt/AlphaComposite.html) documentation.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.awt.image.BufferedImage.TYPE_BYTE_GRAY)

### TYPE\_USHORT\_GRAY

public static final int **TYPE\_USHORT\_GRAY**

Represents an unsigned short grayscale image, non-indexed). This image has a ComponentColorModel with a CS\_GRAY ColorSpace. When data with non-opaque alpha is stored in an image of this type, the color data must be adjusted to a non-premultiplied form and the alpha discarded, as described in the [AlphaComposite](http://docs.google.com/java/awt/AlphaComposite.html) documentation.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.awt.image.BufferedImage.TYPE_USHORT_GRAY)

### TYPE\_BYTE\_BINARY

public static final int **TYPE\_BYTE\_BINARY**

Represents an opaque byte-packed 1, 2, or 4 bit image. The image has an [IndexColorModel](http://docs.google.com/java/awt/image/IndexColorModel.html) without alpha. When this type is used as the imageType argument to the BufferedImage constructor that takes an imageType argument but no ColorModel argument, a 1-bit image is created with an IndexColorModel with two colors in the default sRGB ColorSpace: {0, 0, 0} and {255, 255, 255}.

Images with 2 or 4 bits per pixel may be constructed via the BufferedImage constructor that takes a ColorModel argument by supplying a ColorModel with an appropriate map size.

Images with 8 bits per pixel should use the image types TYPE\_BYTE\_INDEXED or TYPE\_BYTE\_GRAY depending on their ColorModel.

When color data is stored in an image of this type, the closest color in the colormap is determined by the IndexColorModel and the resulting index is stored. Approximation and loss of alpha or color components can result, depending on the colors in the IndexColorModel colormap.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.awt.image.BufferedImage.TYPE_BYTE_BINARY)

### TYPE\_BYTE\_INDEXED

public static final int **TYPE\_BYTE\_INDEXED**

Represents an indexed byte image. When this type is used as the imageType argument to the BufferedImage constructor that takes an imageType argument but no ColorModel argument, an IndexColorModel is created with a 256-color 6/6/6 color cube palette with the rest of the colors from 216-255 populated by grayscale values in the default sRGB ColorSpace.

When color data is stored in an image of this type, the closest color in the colormap is determined by the IndexColorModel and the resulting index is stored. Approximation and loss of alpha or color components can result, depending on the colors in the IndexColorModel colormap.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.awt.image.BufferedImage.TYPE_BYTE_INDEXED)

| **Constructor Detail** |
| --- |

### BufferedImage

public **BufferedImage**(int width,  
 int height,  
 int imageType)

Constructs a BufferedImage of one of the predefined image types. The ColorSpace for the image is the default sRGB space.

**Parameters:**width - width of the created imageheight - height of the created imageimageType - type of the created image**See Also:**[ColorSpace](http://docs.google.com/java/awt/color/ColorSpace.html), [TYPE\_INT\_RGB](http://docs.google.com/java/awt/image/BufferedImage.html#TYPE_INT_RGB), [TYPE\_INT\_ARGB](http://docs.google.com/java/awt/image/BufferedImage.html#TYPE_INT_ARGB), [TYPE\_INT\_ARGB\_PRE](http://docs.google.com/java/awt/image/BufferedImage.html#TYPE_INT_ARGB_PRE), [TYPE\_INT\_BGR](http://docs.google.com/java/awt/image/BufferedImage.html#TYPE_INT_BGR), [TYPE\_3BYTE\_BGR](http://docs.google.com/java/awt/image/BufferedImage.html#TYPE_3BYTE_BGR), [TYPE\_4BYTE\_ABGR](http://docs.google.com/java/awt/image/BufferedImage.html#TYPE_4BYTE_ABGR), [TYPE\_4BYTE\_ABGR\_PRE](http://docs.google.com/java/awt/image/BufferedImage.html#TYPE_4BYTE_ABGR_PRE), [TYPE\_BYTE\_GRAY](http://docs.google.com/java/awt/image/BufferedImage.html#TYPE_BYTE_GRAY), [TYPE\_USHORT\_GRAY](http://docs.google.com/java/awt/image/BufferedImage.html#TYPE_USHORT_GRAY), [TYPE\_BYTE\_BINARY](http://docs.google.com/java/awt/image/BufferedImage.html#TYPE_BYTE_BINARY), [TYPE\_BYTE\_INDEXED](http://docs.google.com/java/awt/image/BufferedImage.html#TYPE_BYTE_INDEXED), [TYPE\_USHORT\_565\_RGB](http://docs.google.com/java/awt/image/BufferedImage.html#TYPE_USHORT_565_RGB), [TYPE\_USHORT\_555\_RGB](http://docs.google.com/java/awt/image/BufferedImage.html#TYPE_USHORT_555_RGB)

### BufferedImage

public **BufferedImage**(int width,  
 int height,  
 int imageType,  
 [IndexColorModel](http://docs.google.com/java/awt/image/IndexColorModel.html) cm)

Constructs a BufferedImage of one of the predefined image types: TYPE\_BYTE\_BINARY or TYPE\_BYTE\_INDEXED.

If the image type is TYPE\_BYTE\_BINARY, the number of entries in the color model is used to determine whether the image should have 1, 2, or 4 bits per pixel. If the color model has 1 or 2 entries, the image will have 1 bit per pixel. If it has 3 or 4 entries, the image with have 2 bits per pixel. If it has between 5 and 16 entries, the image will have 4 bits per pixel. Otherwise, an IllegalArgumentException will be thrown.

**Parameters:**width - width of the created imageheight - height of the created imageimageType - type of the created imagecm - IndexColorModel of the created image **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the imageType is not TYPE\_BYTE\_BINARY or TYPE\_BYTE\_INDEXED or if the imageType is TYPE\_BYTE\_BINARY and the color map has more than 16 entries.**See Also:**[TYPE\_BYTE\_BINARY](http://docs.google.com/java/awt/image/BufferedImage.html#TYPE_BYTE_BINARY), [TYPE\_BYTE\_INDEXED](http://docs.google.com/java/awt/image/BufferedImage.html#TYPE_BYTE_INDEXED)

### BufferedImage

public **BufferedImage**([ColorModel](http://docs.google.com/java/awt/image/ColorModel.html) cm,  
 [WritableRaster](http://docs.google.com/java/awt/image/WritableRaster.html) raster,  
 boolean isRasterPremultiplied,  
 [Hashtable](http://docs.google.com/java/util/Hashtable.html)<?,?> properties)

Constructs a new BufferedImage with a specified ColorModel and Raster. If the number and types of bands in the SampleModel of the Raster do not match the number and types required by the ColorModel to represent its color and alpha components, a [RasterFormatException](http://docs.google.com/java/awt/image/RasterFormatException.html) is thrown. This method can multiply or divide the color Raster data by alpha to match the alphaPremultiplied state in the ColorModel. Properties for this BufferedImage can be established by passing in a [Hashtable](http://docs.google.com/java/util/Hashtable.html) of String/Object pairs.

**Parameters:**cm - ColorModel for the new imageraster - Raster for the image dataisRasterPremultiplied - if true, the data in the raster has been premultiplied with alpha.properties - Hashtable of String/Object pairs. **Throws:** RasterFormatException - if the number and types of bands in the SampleModel of the Raster do not match the number and types required by the ColorModel to represent its color and alpha components. IllegalArgumentException - if raster is incompatible with cm**See Also:**[ColorModel](http://docs.google.com/java/awt/image/ColorModel.html), [Raster](http://docs.google.com/java/awt/image/Raster.html), [WritableRaster](http://docs.google.com/java/awt/image/WritableRaster.html)

| **Method Detail** |
| --- |

### getType

public int **getType**()

Returns the image type. If it is not one of the known types, TYPE\_CUSTOM is returned.

**Returns:**the image type of this BufferedImage.**See Also:**[TYPE\_INT\_RGB](http://docs.google.com/java/awt/image/BufferedImage.html#TYPE_INT_RGB), [TYPE\_INT\_ARGB](http://docs.google.com/java/awt/image/BufferedImage.html#TYPE_INT_ARGB), [TYPE\_INT\_ARGB\_PRE](http://docs.google.com/java/awt/image/BufferedImage.html#TYPE_INT_ARGB_PRE), [TYPE\_INT\_BGR](http://docs.google.com/java/awt/image/BufferedImage.html#TYPE_INT_BGR), [TYPE\_3BYTE\_BGR](http://docs.google.com/java/awt/image/BufferedImage.html#TYPE_3BYTE_BGR), [TYPE\_4BYTE\_ABGR](http://docs.google.com/java/awt/image/BufferedImage.html#TYPE_4BYTE_ABGR), [TYPE\_4BYTE\_ABGR\_PRE](http://docs.google.com/java/awt/image/BufferedImage.html#TYPE_4BYTE_ABGR_PRE), [TYPE\_BYTE\_GRAY](http://docs.google.com/java/awt/image/BufferedImage.html#TYPE_BYTE_GRAY), [TYPE\_BYTE\_BINARY](http://docs.google.com/java/awt/image/BufferedImage.html#TYPE_BYTE_BINARY), [TYPE\_BYTE\_INDEXED](http://docs.google.com/java/awt/image/BufferedImage.html#TYPE_BYTE_INDEXED), [TYPE\_USHORT\_GRAY](http://docs.google.com/java/awt/image/BufferedImage.html#TYPE_USHORT_GRAY), [TYPE\_USHORT\_565\_RGB](http://docs.google.com/java/awt/image/BufferedImage.html#TYPE_USHORT_565_RGB), [TYPE\_USHORT\_555\_RGB](http://docs.google.com/java/awt/image/BufferedImage.html#TYPE_USHORT_555_RGB), [TYPE\_CUSTOM](http://docs.google.com/java/awt/image/BufferedImage.html#TYPE_CUSTOM)

### getColorModel

public [ColorModel](http://docs.google.com/java/awt/image/ColorModel.html) **getColorModel**()

Returns the ColorModel.

**Specified by:**[getColorModel](http://docs.google.com/java/awt/image/RenderedImage.html#getColorModel()) in interface [RenderedImage](http://docs.google.com/java/awt/image/RenderedImage.html) **Returns:**the ColorModel of this BufferedImage.

### getRaster

public [WritableRaster](http://docs.google.com/java/awt/image/WritableRaster.html) **getRaster**()

Returns the [WritableRaster](http://docs.google.com/java/awt/image/WritableRaster.html).

**Returns:**the WriteableRaster of this BufferedImage.

### getAlphaRaster

public [WritableRaster](http://docs.google.com/java/awt/image/WritableRaster.html) **getAlphaRaster**()

Returns a WritableRaster representing the alpha channel for BufferedImage objects with ColorModel objects that support a separate spatial alpha channel, such as ComponentColorModel and DirectColorModel. Returns null if there is no alpha channel associated with the ColorModel in this image. This method assumes that for all ColorModel objects other than IndexColorModel, if the ColorModel supports alpha, there is a separate alpha channel which is stored as the last band of image data. If the image uses an IndexColorModel that has alpha in the lookup table, this method returns null since there is no spatially discrete alpha channel. This method creates a new WritableRaster, but shares the data array.

**Returns:**a WritableRaster or null if this BufferedImage has no alpha channel associated with its ColorModel.

### getRGB

public int **getRGB**(int x,  
 int y)

Returns an integer pixel in the default RGB color model (TYPE\_INT\_ARGB) and default sRGB colorspace. Color conversion takes place if this default model does not match the image ColorModel. There are only 8-bits of precision for each color component in the returned data when using this method.

An ArrayOutOfBoundsException may be thrown if the coordinates are not in bounds. However, explicit bounds checking is not guaranteed.

**Parameters:**x - the X coordinate of the pixel from which to get the pixel in the default RGB color model and sRGB color spacey - the Y coordinate of the pixel from which to get the pixel in the default RGB color model and sRGB color space **Returns:**an integer pixel in the default RGB color model and default sRGB colorspace.**See Also:**[setRGB(int, int, int)](http://docs.google.com/java/awt/image/BufferedImage.html#setRGB(int,%20int,%20int)), [setRGB(int, int, int, int, int[], int, int)](http://docs.google.com/java/awt/image/BufferedImage.html#setRGB(int,%20int,%20int,%20int,%20int%5B%5D,%20int,%20int))

### getRGB

public int[] **getRGB**(int startX,  
 int startY,  
 int w,  
 int h,  
 int[] rgbArray,  
 int offset,  
 int scansize)

Returns an array of integer pixels in the default RGB color model (TYPE\_INT\_ARGB) and default sRGB color space, from a portion of the image data. Color conversion takes place if the default model does not match the image ColorModel. There are only 8-bits of precision for each color component in the returned data when using this method. With a specified coordinate (x, y) in the image, the ARGB pixel can be accessed in this way:

pixel = rgbArray[offset + (y-startY)\*scansize + (x-startX)];

An ArrayOutOfBoundsException may be thrown if the region is not in bounds. However, explicit bounds checking is not guaranteed.

**Parameters:**startX - the starting X coordinatestartY - the starting Y coordinatew - width of regionh - height of regionrgbArray - if not null, the rgb pixels are written hereoffset - offset into the rgbArrayscansize - scanline stride for the rgbArray **Returns:**array of RGB pixels.**See Also:**[setRGB(int, int, int)](http://docs.google.com/java/awt/image/BufferedImage.html#setRGB(int,%20int,%20int)), [setRGB(int, int, int, int, int[], int, int)](http://docs.google.com/java/awt/image/BufferedImage.html#setRGB(int,%20int,%20int,%20int,%20int%5B%5D,%20int,%20int))

### setRGB

public void **setRGB**(int x,  
 int y,  
 int rgb)

Sets a pixel in this BufferedImage to the specified RGB value. The pixel is assumed to be in the default RGB color model, TYPE\_INT\_ARGB, and default sRGB color space. For images with an IndexColorModel, the index with the nearest color is chosen.

An ArrayOutOfBoundsException may be thrown if the coordinates are not in bounds. However, explicit bounds checking is not guaranteed.

**Parameters:**x - the X coordinate of the pixel to sety - the Y coordinate of the pixel to setrgb - the RGB value**See Also:**[getRGB(int, int)](http://docs.google.com/java/awt/image/BufferedImage.html#getRGB(int,%20int)), [getRGB(int, int, int, int, int[], int, int)](http://docs.google.com/java/awt/image/BufferedImage.html#getRGB(int,%20int,%20int,%20int,%20int%5B%5D,%20int,%20int))

### setRGB

public void **setRGB**(int startX,  
 int startY,  
 int w,  
 int h,  
 int[] rgbArray,  
 int offset,  
 int scansize)

Sets an array of integer pixels in the default RGB color model (TYPE\_INT\_ARGB) and default sRGB color space, into a portion of the image data. Color conversion takes place if the default model does not match the image ColorModel. There are only 8-bits of precision for each color component in the returned data when using this method. With a specified coordinate (x, y) in the this image, the ARGB pixel can be accessed in this way:

pixel = rgbArray[offset + (y-startY)\*scansize + (x-startX)];

WARNING: No dithering takes place.

An ArrayOutOfBoundsException may be thrown if the region is not in bounds. However, explicit bounds checking is not guaranteed.

**Parameters:**startX - the starting X coordinatestartY - the starting Y coordinatew - width of the regionh - height of the regionrgbArray - the rgb pixelsoffset - offset into the rgbArrayscansize - scanline stride for the rgbArray**See Also:**[getRGB(int, int)](http://docs.google.com/java/awt/image/BufferedImage.html#getRGB(int,%20int)), [getRGB(int, int, int, int, int[], int, int)](http://docs.google.com/java/awt/image/BufferedImage.html#getRGB(int,%20int,%20int,%20int,%20int%5B%5D,%20int,%20int))

### getWidth

public int **getWidth**()

Returns the width of the BufferedImage.

**Specified by:**[getWidth](http://docs.google.com/java/awt/image/RenderedImage.html#getWidth()) in interface [RenderedImage](http://docs.google.com/java/awt/image/RenderedImage.html) **Returns:**the width of this BufferedImage

### getHeight

public int **getHeight**()

Returns the height of the BufferedImage.

**Specified by:**[getHeight](http://docs.google.com/java/awt/image/RenderedImage.html#getHeight()) in interface [RenderedImage](http://docs.google.com/java/awt/image/RenderedImage.html) **Returns:**the height of this BufferedImage

### getWidth

public int **getWidth**([ImageObserver](http://docs.google.com/java/awt/image/ImageObserver.html) observer)

Returns the width of the BufferedImage.

**Specified by:**[getWidth](http://docs.google.com/java/awt/Image.html#getWidth(java.awt.image.ImageObserver)) in class [Image](http://docs.google.com/java/awt/Image.html) **Parameters:**observer - ignored **Returns:**the width of this BufferedImage**See Also:**[Image.getHeight(java.awt.image.ImageObserver)](http://docs.google.com/java/awt/Image.html#getHeight(java.awt.image.ImageObserver)), [ImageObserver](http://docs.google.com/java/awt/image/ImageObserver.html)

### getHeight

public int **getHeight**([ImageObserver](http://docs.google.com/java/awt/image/ImageObserver.html) observer)

Returns the height of the BufferedImage.

**Specified by:**[getHeight](http://docs.google.com/java/awt/Image.html#getHeight(java.awt.image.ImageObserver)) in class [Image](http://docs.google.com/java/awt/Image.html) **Parameters:**observer - ignored **Returns:**the height of this BufferedImage**See Also:**[Image.getWidth(java.awt.image.ImageObserver)](http://docs.google.com/java/awt/Image.html#getWidth(java.awt.image.ImageObserver)), [ImageObserver](http://docs.google.com/java/awt/image/ImageObserver.html)

### getSource

public [ImageProducer](http://docs.google.com/java/awt/image/ImageProducer.html) **getSource**()

Returns the object that produces the pixels for the image.

**Specified by:**[getSource](http://docs.google.com/java/awt/Image.html#getSource()) in class [Image](http://docs.google.com/java/awt/Image.html) **Returns:**the [ImageProducer](http://docs.google.com/java/awt/image/ImageProducer.html) that is used to produce the pixels for this image.**See Also:**[ImageProducer](http://docs.google.com/java/awt/image/ImageProducer.html)

### getProperty

public [Object](http://docs.google.com/java/lang/Object.html) **getProperty**([String](http://docs.google.com/java/lang/String.html) name,  
 [ImageObserver](http://docs.google.com/java/awt/image/ImageObserver.html) observer)

Returns a property of the image by name. Individual property names are defined by the various image formats. If a property is not defined for a particular image, this method returns the UndefinedProperty field. If the properties for this image are not yet known, then this method returns null and the ImageObserver object is notified later. The property name "comment" should be used to store an optional comment that can be presented to the user as a description of the image, its source, or its author.

**Specified by:**[getProperty](http://docs.google.com/java/awt/Image.html#getProperty(java.lang.String,%20java.awt.image.ImageObserver)) in class [Image](http://docs.google.com/java/awt/Image.html) **Parameters:**name - the property nameobserver - the ImageObserver that receives notification regarding image information **Returns:**an [Object](http://docs.google.com/java/lang/Object.html) that is the property referred to by the specified name or null if the properties of this image are not yet known. **Throws:** NullPointerException - if the property name is null.**See Also:**[ImageObserver](http://docs.google.com/java/awt/image/ImageObserver.html), [Image.UndefinedProperty](http://docs.google.com/java/awt/Image.html#UndefinedProperty)

### getProperty

public [Object](http://docs.google.com/java/lang/Object.html) **getProperty**([String](http://docs.google.com/java/lang/String.html) name)

Returns a property of the image by name.

**Specified by:**[getProperty](http://docs.google.com/java/awt/image/RenderedImage.html#getProperty(java.lang.String)) in interface [RenderedImage](http://docs.google.com/java/awt/image/RenderedImage.html) **Parameters:**name - the property name **Returns:**an Object that is the property referred to by the specified name. **Throws:** NullPointerException - if the property name is null.**See Also:**[Image.UndefinedProperty](http://docs.google.com/java/awt/Image.html#UndefinedProperty)

### getGraphics

public [Graphics](http://docs.google.com/java/awt/Graphics.html) **getGraphics**()

This method returns a [Graphics2D](http://docs.google.com/java/awt/Graphics2D.html), but is here for backwards compatibility. [createGraphics](http://docs.google.com/java/awt/image/BufferedImage.html#createGraphics()) is more convenient, since it is declared to return a Graphics2D.

**Specified by:**[getGraphics](http://docs.google.com/java/awt/Image.html#getGraphics()) in class [Image](http://docs.google.com/java/awt/Image.html) **Returns:**a Graphics2D, which can be used to draw into this image.**See Also:**[Graphics](http://docs.google.com/java/awt/Graphics.html), [Component.createImage(int, int)](http://docs.google.com/java/awt/Component.html#createImage(int,%20int))

### createGraphics

public [Graphics2D](http://docs.google.com/java/awt/Graphics2D.html) **createGraphics**()

Creates a Graphics2D, which can be used to draw into this BufferedImage.

**Returns:**a Graphics2D, used for drawing into this image.

### getSubimage

public [BufferedImage](http://docs.google.com/java/awt/image/BufferedImage.html) **getSubimage**(int x,  
 int y,  
 int w,  
 int h)

Returns a subimage defined by a specified rectangular region. The returned BufferedImage shares the same data array as the original image.

**Parameters:**x - the X coordinate of the upper-left corner of the specified rectangular regiony - the Y coordinate of the upper-left corner of the specified rectangular regionw - the width of the specified rectangular regionh - the height of the specified rectangular region **Returns:**a BufferedImage that is the subimage of this BufferedImage. **Throws:** RasterFormatException - if the specified area is not contained within this BufferedImage.

### isAlphaPremultiplied

public boolean **isAlphaPremultiplied**()

Returns whether or not the alpha has been premultiplied. It returns false if there is no alpha.

**Returns:**true if the alpha has been premultiplied; false otherwise.

### coerceData

public void **coerceData**(boolean isAlphaPremultiplied)

Forces the data to match the state specified in the isAlphaPremultiplied variable. It may multiply or divide the color raster data by alpha, or do nothing if the data is in the correct state.

**Parameters:**isAlphaPremultiplied - true if the alpha has been premultiplied; false otherwise.

### toString

public [String](http://docs.google.com/java/lang/String.html) **toString**()

Returns a String representation of this BufferedImage object and its values.

**Overrides:**[toString](http://docs.google.com/java/lang/Object.html#toString()) in class [Object](http://docs.google.com/java/lang/Object.html) **Returns:**a String representing this BufferedImage.

### getSources

public [Vector](http://docs.google.com/java/util/Vector.html)<[RenderedImage](http://docs.google.com/java/awt/image/RenderedImage.html)> **getSources**()

Returns a [Vector](http://docs.google.com/java/util/Vector.html) of [RenderedImage](http://docs.google.com/java/awt/image/RenderedImage.html) objects that are the immediate sources, not the sources of these immediate sources, of image data for this BufferedImage. This method returns null if the BufferedImage has no information about its immediate sources. It returns an empty Vector if the BufferedImage has no immediate sources.

**Specified by:**[getSources](http://docs.google.com/java/awt/image/RenderedImage.html#getSources()) in interface [RenderedImage](http://docs.google.com/java/awt/image/RenderedImage.html) **Returns:**a Vector containing immediate sources of this BufferedImage object's image date, or null if this BufferedImage has no information about its immediate sources, or an empty Vector if this BufferedImage has no immediate sources.

### getPropertyNames

public [String](http://docs.google.com/java/lang/String.html)[] **getPropertyNames**()

Returns an array of names recognized by [getProperty(String)](http://docs.google.com/java/awt/image/BufferedImage.html#getProperty(java.lang.String)) or null, if no property names are recognized.

**Specified by:**[getPropertyNames](http://docs.google.com/java/awt/image/RenderedImage.html#getPropertyNames()) in interface [RenderedImage](http://docs.google.com/java/awt/image/RenderedImage.html) **Returns:**a String array containing all of the property names that getProperty(String) recognizes; or null if no property names are recognized.

### getMinX

public int **getMinX**()

Returns the minimum x coordinate of this BufferedImage. This is always zero.

**Specified by:**[getMinX](http://docs.google.com/java/awt/image/RenderedImage.html#getMinX()) in interface [RenderedImage](http://docs.google.com/java/awt/image/RenderedImage.html) **Returns:**the minimum x coordinate of this BufferedImage.

### getMinY

public int **getMinY**()

Returns the minimum y coordinate of this BufferedImage. This is always zero.

**Specified by:**[getMinY](http://docs.google.com/java/awt/image/RenderedImage.html#getMinY()) in interface [RenderedImage](http://docs.google.com/java/awt/image/RenderedImage.html) **Returns:**the minimum y coordinate of this BufferedImage.

### getSampleModel

public [SampleModel](http://docs.google.com/java/awt/image/SampleModel.html) **getSampleModel**()

Returns the SampleModel associated with this BufferedImage.

**Specified by:**[getSampleModel](http://docs.google.com/java/awt/image/RenderedImage.html#getSampleModel()) in interface [RenderedImage](http://docs.google.com/java/awt/image/RenderedImage.html) **Returns:**the SampleModel of this BufferedImage.

### getNumXTiles

public int **getNumXTiles**()

Returns the number of tiles in the x direction. This is always one.

**Specified by:**[getNumXTiles](http://docs.google.com/java/awt/image/RenderedImage.html#getNumXTiles()) in interface [RenderedImage](http://docs.google.com/java/awt/image/RenderedImage.html) **Returns:**the number of tiles in the x direction.

### getNumYTiles

public int **getNumYTiles**()

Returns the number of tiles in the y direction. This is always one.

**Specified by:**[getNumYTiles](http://docs.google.com/java/awt/image/RenderedImage.html#getNumYTiles()) in interface [RenderedImage](http://docs.google.com/java/awt/image/RenderedImage.html) **Returns:**the number of tiles in the y direction.

### getMinTileX

public int **getMinTileX**()

Returns the minimum tile index in the x direction. This is always zero.

**Specified by:**[getMinTileX](http://docs.google.com/java/awt/image/RenderedImage.html#getMinTileX()) in interface [RenderedImage](http://docs.google.com/java/awt/image/RenderedImage.html) **Returns:**the minimum tile index in the x direction.

### getMinTileY

public int **getMinTileY**()

Returns the minimum tile index in the y direction. This is always zero.

**Specified by:**[getMinTileY](http://docs.google.com/java/awt/image/RenderedImage.html#getMinTileY()) in interface [RenderedImage](http://docs.google.com/java/awt/image/RenderedImage.html) **Returns:**the mininum tile index in the y direction.

### getTileWidth

public int **getTileWidth**()

Returns the tile width in pixels.

**Specified by:**[getTileWidth](http://docs.google.com/java/awt/image/RenderedImage.html#getTileWidth()) in interface [RenderedImage](http://docs.google.com/java/awt/image/RenderedImage.html) **Returns:**the tile width in pixels.

### getTileHeight

public int **getTileHeight**()

Returns the tile height in pixels.

**Specified by:**[getTileHeight](http://docs.google.com/java/awt/image/RenderedImage.html#getTileHeight()) in interface [RenderedImage](http://docs.google.com/java/awt/image/RenderedImage.html) **Returns:**the tile height in pixels.

### getTileGridXOffset

public int **getTileGridXOffset**()

Returns the x offset of the tile grid relative to the origin, For example, the x coordinate of the location of tile (0, 0). This is always zero.

**Specified by:**[getTileGridXOffset](http://docs.google.com/java/awt/image/RenderedImage.html#getTileGridXOffset()) in interface [RenderedImage](http://docs.google.com/java/awt/image/RenderedImage.html) **Returns:**the x offset of the tile grid.

### getTileGridYOffset

public int **getTileGridYOffset**()

Returns the y offset of the tile grid relative to the origin, For example, the y coordinate of the location of tile (0, 0). This is always zero.

**Specified by:**[getTileGridYOffset](http://docs.google.com/java/awt/image/RenderedImage.html#getTileGridYOffset()) in interface [RenderedImage](http://docs.google.com/java/awt/image/RenderedImage.html) **Returns:**the y offset of the tile grid.

### getTile

public [Raster](http://docs.google.com/java/awt/image/Raster.html) **getTile**(int tileX,  
 int tileY)

Returns tile (tileX, tileY). Note that tileX and tileY are indices into the tile array, not pixel locations. The Raster that is returned is live, which means that it is updated if the image is changed.

**Specified by:**[getTile](http://docs.google.com/java/awt/image/RenderedImage.html#getTile(int,%20int)) in interface [RenderedImage](http://docs.google.com/java/awt/image/RenderedImage.html) **Parameters:**tileX - the x index of the requested tile in the tile arraytileY - the y index of the requested tile in the tile array **Returns:**a Raster that is the tile defined by the arguments tileX and tileY. **Throws:** ArrayIndexOutOfBoundsException - if both tileX and tileY are not equal to 0

### getData

public [Raster](http://docs.google.com/java/awt/image/Raster.html) **getData**()

Returns the image as one large tile. The Raster returned is a copy of the image data is not updated if the image is changed.

**Specified by:**[getData](http://docs.google.com/java/awt/image/RenderedImage.html#getData()) in interface [RenderedImage](http://docs.google.com/java/awt/image/RenderedImage.html) **Returns:**a Raster that is a copy of the image data.**See Also:**[setData(Raster)](http://docs.google.com/java/awt/image/BufferedImage.html#setData(java.awt.image.Raster))

### getData

public [Raster](http://docs.google.com/java/awt/image/Raster.html) **getData**([Rectangle](http://docs.google.com/java/awt/Rectangle.html) rect)

Computes and returns an arbitrary region of the BufferedImage. The Raster returned is a copy of the image data and is not updated if the image is changed.

**Specified by:**[getData](http://docs.google.com/java/awt/image/RenderedImage.html#getData(java.awt.Rectangle)) in interface [RenderedImage](http://docs.google.com/java/awt/image/RenderedImage.html) **Parameters:**rect - the region of the BufferedImage to be returned. **Returns:**a Raster that is a copy of the image data of the specified region of the BufferedImage**See Also:**[setData(Raster)](http://docs.google.com/java/awt/image/BufferedImage.html#setData(java.awt.image.Raster))

### copyData

public [WritableRaster](http://docs.google.com/java/awt/image/WritableRaster.html) **copyData**([WritableRaster](http://docs.google.com/java/awt/image/WritableRaster.html) outRaster)

Computes an arbitrary rectangular region of the BufferedImage and copies it into a specified WritableRaster. The region to be computed is determined from the bounds of the specified WritableRaster. The specified WritableRaster must have a SampleModel that is compatible with this image. If outRaster is null, an appropriate WritableRaster is created.

**Specified by:**[copyData](http://docs.google.com/java/awt/image/RenderedImage.html#copyData(java.awt.image.WritableRaster)) in interface [RenderedImage](http://docs.google.com/java/awt/image/RenderedImage.html) **Parameters:**outRaster - a WritableRaster to hold the returned part of the image, or null **Returns:**a reference to the supplied or created WritableRaster.

### setData

public void **setData**([Raster](http://docs.google.com/java/awt/image/Raster.html) r)

Sets a rectangular region of the image to the contents of the specified Raster r, which is assumed to be in the same coordinate space as the BufferedImage. The operation is clipped to the bounds of the BufferedImage.

**Specified by:**[setData](http://docs.google.com/java/awt/image/WritableRenderedImage.html#setData(java.awt.image.Raster)) in interface [WritableRenderedImage](http://docs.google.com/java/awt/image/WritableRenderedImage.html) **Parameters:**r - the specified Raster**See Also:**[getData()](http://docs.google.com/java/awt/image/BufferedImage.html#getData()), [getData(Rectangle)](http://docs.google.com/java/awt/image/BufferedImage.html#getData(java.awt.Rectangle))

### addTileObserver

public void **addTileObserver**([TileObserver](http://docs.google.com/java/awt/image/TileObserver.html) to)

Adds a tile observer. If the observer is already present, it receives multiple notifications.

**Specified by:**[addTileObserver](http://docs.google.com/java/awt/image/WritableRenderedImage.html#addTileObserver(java.awt.image.TileObserver)) in interface [WritableRenderedImage](http://docs.google.com/java/awt/image/WritableRenderedImage.html) **Parameters:**to - the specified [TileObserver](http://docs.google.com/java/awt/image/TileObserver.html)

### removeTileObserver

public void **removeTileObserver**([TileObserver](http://docs.google.com/java/awt/image/TileObserver.html) to)

Removes a tile observer. If the observer was not registered, nothing happens. If the observer was registered for multiple notifications, it is now registered for one fewer notification.

**Specified by:**[removeTileObserver](http://docs.google.com/java/awt/image/WritableRenderedImage.html#removeTileObserver(java.awt.image.TileObserver)) in interface [WritableRenderedImage](http://docs.google.com/java/awt/image/WritableRenderedImage.html) **Parameters:**to - the specified TileObserver.

### isTileWritable

public boolean **isTileWritable**(int tileX,  
 int tileY)

Returns whether or not a tile is currently checked out for writing.

**Specified by:**[isTileWritable](http://docs.google.com/java/awt/image/WritableRenderedImage.html#isTileWritable(int,%20int)) in interface [WritableRenderedImage](http://docs.google.com/java/awt/image/WritableRenderedImage.html) **Parameters:**tileX - the x index of the tile.tileY - the y index of the tile. **Returns:**true if the tile specified by the specified indices is checked out for writing; false otherwise. **Throws:** ArrayIndexOutOfBoundsException - if both tileX and tileY are not equal to 0

### getWritableTileIndices

public [Point](http://docs.google.com/java/awt/Point.html)[] **getWritableTileIndices**()

Returns an array of [Point](http://docs.google.com/java/awt/Point.html) objects indicating which tiles are checked out for writing. Returns null if none are checked out.

**Specified by:**[getWritableTileIndices](http://docs.google.com/java/awt/image/WritableRenderedImage.html#getWritableTileIndices()) in interface [WritableRenderedImage](http://docs.google.com/java/awt/image/WritableRenderedImage.html) **Returns:**a Point array that indicates the tiles that are checked out for writing, or null if no tiles are checked out for writing.

### hasTileWriters

public boolean **hasTileWriters**()

Returns whether or not any tile is checked out for writing. Semantically equivalent to

(getWritableTileIndices() != null).

**Specified by:**[hasTileWriters](http://docs.google.com/java/awt/image/WritableRenderedImage.html#hasTileWriters()) in interface [WritableRenderedImage](http://docs.google.com/java/awt/image/WritableRenderedImage.html) **Returns:**true if any tile is checked out for writing; false otherwise.

### getWritableTile

public [WritableRaster](http://docs.google.com/java/awt/image/WritableRaster.html) **getWritableTile**(int tileX,  
 int tileY)

Checks out a tile for writing. All registered TileObservers are notified when a tile goes from having no writers to having one writer.

**Specified by:**[getWritableTile](http://docs.google.com/java/awt/image/WritableRenderedImage.html#getWritableTile(int,%20int)) in interface [WritableRenderedImage](http://docs.google.com/java/awt/image/WritableRenderedImage.html) **Parameters:**tileX - the x index of the tiletileY - the y index of the tile **Returns:**a WritableRaster that is the tile, indicated by the specified indices, to be checked out for writing.

### releaseWritableTile

public void **releaseWritableTile**(int tileX,  
 int tileY)

Relinquishes permission to write to a tile. If the caller continues to write to the tile, the results are undefined. Calls to this method should only appear in matching pairs with calls to [getWritableTile(int, int)](http://docs.google.com/java/awt/image/BufferedImage.html#getWritableTile(int,%20int)). Any other leads to undefined results. All registered TileObservers are notified when a tile goes from having one writer to having no writers.

**Specified by:**[releaseWritableTile](http://docs.google.com/java/awt/image/WritableRenderedImage.html#releaseWritableTile(int,%20int)) in interface [WritableRenderedImage](http://docs.google.com/java/awt/image/WritableRenderedImage.html) **Parameters:**tileX - the x index of the tiletileY - the y index of the tile

### getTransparency

public int **getTransparency**()

Returns the transparency. Returns either OPAQUE, BITMASK, or TRANSLUCENT.

**Specified by:**[getTransparency](http://docs.google.com/java/awt/Transparency.html#getTransparency()) in interface [Transparency](http://docs.google.com/java/awt/Transparency.html) **Returns:**the transparency of this BufferedImage.**Since:** 1.5 **See Also:**[Transparency.OPAQUE](http://docs.google.com/java/awt/Transparency.html#OPAQUE), [Transparency.BITMASK](http://docs.google.com/java/awt/Transparency.html#BITMASK), [Transparency.TRANSLUCENT](http://docs.google.com/java/awt/Transparency.html#TRANSLUCENT)

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/BufferedImage.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/java/awt/image/BandedSampleModel.html)   [**NEXT CLASS**](http://docs.google.com/java/awt/image/BufferedImageFilter.html) | [**FRAMES**](http://docs.google.com/index.html?java/awt/image/BufferedImage.html)    [**NO FRAMES**](http://docs.google.com/BufferedImage.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | [CONSTR](#3dy6vkm) | [METHOD](#1t3h5sf) | DETAIL: [FIELD](#17dp8vu) | [CONSTR](#2bn6wsx) | [METHOD](#49x2ik5) |

[Submit a bug or feature](http://bugs.sun.com/services/bugreport/index.jsp)

For further API reference and developer documentation, see [Java SE Developer Documentation](http://docs.google.com/webnotes/devdocs-vs-specs.html). That documentation contains more detailed, developer-targeted descriptions, with conceptual overviews, definitions of terms, workarounds, and working code examples.

Copyright 2006 Sun Microsystems, Inc. All rights reserved. Use is subject to [license terms](http://docs.google.com/legal/license.html). Also see the [documentation redistribution policy](http://java.sun.com/docs/redist.html).