| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/IndexColorModel.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/ImagingOpException.html)   [**NEXT CLASS**](http://docs.google.com/java/awt/image/Kernel.html) | [**FRAMES**](http://docs.google.com/index.html?java/awt/image/IndexColorModel.html)    [**NO FRAMES**](http://docs.google.com/IndexColorModel.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3dy6vkm) | [CONSTR](#4d34og8) | [METHOD](#2s8eyo1) | DETAIL: FIELD | [CONSTR](#26in1rg) | [METHOD](#1y810tw) |

## **java.awt.image**

Class IndexColorModel

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

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

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

The IndexColorModel class is a ColorModel class that works with pixel values consisting of a single sample that is an index into a fixed colormap in the default sRGB color space. The colormap specifies red, green, blue, and optional alpha components corresponding to each index. All components are represented in the colormap as 8-bit unsigned integral values. Some constructors allow the caller to specify "holes" in the colormap by indicating which colormap entries are valid and which represent unusable colors via the bits set in a BigInteger object. This color model is similar to an X11 PseudoColor visual.

Some constructors provide a means to specify an alpha component for each pixel in the colormap, while others either provide no such means or, in some cases, a flag to indicate whether the colormap data contains alpha values. If no alpha is supplied to the constructor, an opaque alpha component (alpha = 1.0) is assumed for each entry. An optional transparent pixel value can be supplied that indicates a pixel to be made completely transparent, regardless of any alpha component supplied or assumed for that pixel value. Note that the color components in the colormap of an IndexColorModel objects are never pre-multiplied with the alpha components.

The transparency of an IndexColorModel object is determined by examining the alpha components of the colors in the colormap and choosing the most specific value after considering the optional alpha values and any transparent index specified. The transparency value is Transparency.OPAQUE only if all valid colors in the colormap are opaque and there is no valid transparent pixel. If all valid colors in the colormap are either completely opaque (alpha = 1.0) or completely transparent (alpha = 0.0), which typically occurs when a valid transparent pixel is specified, the value is Transparency.BITMASK. Otherwise, the value is Transparency.TRANSLUCENT, indicating that some valid color has an alpha component that is neither completely transparent nor completely opaque (0.0 < alpha < 1.0).

If an IndexColorModel object has a transparency value of Transparency.OPAQUE, then the hasAlpha and getNumComponents methods (both inherited from ColorModel) return false and 3, respectively. For any other transparency value, hasAlpha returns true and getNumComponents returns 4.

The values used to index into the colormap are taken from the least significant *n* bits of pixel representations where *n* is based on the pixel size specified in the constructor. For pixel sizes smaller than 8 bits, *n* is rounded up to a power of two (3 becomes 4 and 5,6,7 become 8). For pixel sizes between 8 and 16 bits, *n* is equal to the pixel size. Pixel sizes larger than 16 bits are not supported by this class. Higher order bits beyond *n* are ignored in pixel representations. Index values greater than or equal to the map size, but less than 2*n*, are undefined and return 0 for all color and alpha components.

For those methods that use a primitive array pixel representation of type transferType, the array length is always one. The transfer types supported are DataBuffer.TYPE\_BYTE and DataBuffer.TYPE\_USHORT. A single int pixel representation is valid for all objects of this class, since it is always possible to represent pixel values used with this class in a single int. Therefore, methods that use this representation do not throw an IllegalArgumentException due to an invalid pixel value.

Many of the methods in this class are final. The reason for this is that the underlying native graphics code makes assumptions about the layout and operation of this class and those assumptions are reflected in the implementations of the methods here that are marked final. You can subclass this class for other reasons, but you cannot override or modify the behaviour of those methods.

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

| **Field Summary** | |
| --- | --- |

| **Fields inherited from class java.awt.image.**[**ColorModel**](http://docs.google.com/java/awt/image/ColorModel.html) |
| --- |
| [pixel\_bits](http://docs.google.com/java/awt/image/ColorModel.html#pixel_bits), [transferType](http://docs.google.com/java/awt/image/ColorModel.html#transferType) |

| **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** | |
| --- | --- |
| [**IndexColorModel**](http://docs.google.com/java/awt/image/IndexColorModel.html#IndexColorModel(int,%20int,%20byte%5B%5D,%20byte%5B%5D,%20byte%5B%5D))(int bits, int size, byte[] r, byte[] g, byte[] b)            Constructs an IndexColorModel from the specified arrays of red, green, and blue components. |
| [**IndexColorModel**](http://docs.google.com/java/awt/image/IndexColorModel.html#IndexColorModel(int,%20int,%20byte%5B%5D,%20byte%5B%5D,%20byte%5B%5D,%20byte%5B%5D))(int bits, int size, byte[] r, byte[] g, byte[] b, byte[] a)            Constructs an IndexColorModel from the given arrays of red, green, blue and alpha components. |
| [**IndexColorModel**](http://docs.google.com/java/awt/image/IndexColorModel.html#IndexColorModel(int,%20int,%20byte%5B%5D,%20byte%5B%5D,%20byte%5B%5D,%20int))(int bits, int size, byte[] r, byte[] g, byte[] b, int trans)            Constructs an IndexColorModel from the given arrays of red, green, and blue components. |
| [**IndexColorModel**](http://docs.google.com/java/awt/image/IndexColorModel.html#IndexColorModel(int,%20int,%20byte%5B%5D,%20int,%20boolean))(int bits, int size, byte[] cmap, int start, boolean hasalpha)            Constructs an IndexColorModel from a single array of interleaved red, green, blue and optional alpha components. |
| [**IndexColorModel**](http://docs.google.com/java/awt/image/IndexColorModel.html#IndexColorModel(int,%20int,%20byte%5B%5D,%20int,%20boolean,%20int))(int bits, int size, byte[] cmap, int start, boolean hasalpha, int trans)            Constructs an IndexColorModel from a single array of interleaved red, green, blue and optional alpha components. |
| [**IndexColorModel**](http://docs.google.com/java/awt/image/IndexColorModel.html#IndexColorModel(int,%20int,%20int%5B%5D,%20int,%20boolean,%20int,%20int))(int bits, int size, int[] cmap, int start, boolean hasalpha, int trans, int transferType)            Constructs an IndexColorModel from an array of ints where each int is comprised of red, green, blue, and optional alpha components in the default RGB color model format. |
| [**IndexColorModel**](http://docs.google.com/java/awt/image/IndexColorModel.html#IndexColorModel(int,%20int,%20int%5B%5D,%20int,%20int,%20java.math.BigInteger))(int bits, int size, int[] cmap, int start, int transferType, [BigInteger](http://docs.google.com/java/math/BigInteger.html) validBits)            Constructs an IndexColorModel from an int array where each int is comprised of red, green, blue, and alpha components in the default RGB color model format. |

| **Method Summary** | |
| --- | --- |
| [BufferedImage](http://docs.google.com/java/awt/image/BufferedImage.html) | [**convertToIntDiscrete**](http://docs.google.com/java/awt/image/IndexColorModel.html#convertToIntDiscrete(java.awt.image.Raster,%20boolean))([Raster](http://docs.google.com/java/awt/image/Raster.html) raster, boolean forceARGB)            Returns a new BufferedImage of TYPE\_INT\_ARGB or TYPE\_INT\_RGB that has a Raster with pixel data computed by expanding the indices in the source Raster using the color/alpha component arrays of this ColorModel. |
| [SampleModel](http://docs.google.com/java/awt/image/SampleModel.html) | [**createCompatibleSampleModel**](http://docs.google.com/java/awt/image/IndexColorModel.html#createCompatibleSampleModel(int,%20int))(int w, int h)            Creates a SampleModel with the specified width and height that has a data layout compatible with this ColorModel. |
| [WritableRaster](http://docs.google.com/java/awt/image/WritableRaster.html) | [**createCompatibleWritableRaster**](http://docs.google.com/java/awt/image/IndexColorModel.html#createCompatibleWritableRaster(int,%20int))(int w, int h)            Creates a WritableRaster with the specified width and height that has a data layout (SampleModel) compatible with this ColorModel. |
| void | [**finalize**](http://docs.google.com/java/awt/image/IndexColorModel.html#finalize())()            Disposes of system resources associated with this ColorModel once this ColorModel is no longer referenced. |
| int | [**getAlpha**](http://docs.google.com/java/awt/image/IndexColorModel.html#getAlpha(int))(int pixel)            Returns the alpha component for the specified pixel, scaled from 0 to 255. |
| void | [**getAlphas**](http://docs.google.com/java/awt/image/IndexColorModel.html#getAlphas(byte%5B%5D))(byte[] a)            Copies the array of alpha transparency components into the specified array. |
| int | [**getBlue**](http://docs.google.com/java/awt/image/IndexColorModel.html#getBlue(int))(int pixel)            Returns the blue color component for the specified pixel, scaled from 0 to 255 in the default RGB ColorSpace, sRGB. |
| void | [**getBlues**](http://docs.google.com/java/awt/image/IndexColorModel.html#getBlues(byte%5B%5D))(byte[] b)            Copies the array of blue color components into the specified array. |
| int[] | [**getComponents**](http://docs.google.com/java/awt/image/IndexColorModel.html#getComponents(int,%20int%5B%5D,%20int))(int pixel, int[] components, int offset)            Returns an array of unnormalized color/alpha components for a specified pixel in this ColorModel. |
| int[] | [**getComponents**](http://docs.google.com/java/awt/image/IndexColorModel.html#getComponents(java.lang.Object,%20int%5B%5D,%20int))([Object](http://docs.google.com/java/lang/Object.html) pixel, int[] components, int offset)            Returns an array of unnormalized color/alpha components for a specified pixel in this ColorModel. |
| int[] | [**getComponentSize**](http://docs.google.com/java/awt/image/IndexColorModel.html#getComponentSize())()            Returns an array of the number of bits for each color/alpha component. |
| int | [**getDataElement**](http://docs.google.com/java/awt/image/IndexColorModel.html#getDataElement(int%5B%5D,%20int))(int[] components, int offset)            Returns a pixel value represented as an int in this ColorModel given an array of unnormalized color/alpha components. |
| [Object](http://docs.google.com/java/lang/Object.html) | [**getDataElements**](http://docs.google.com/java/awt/image/IndexColorModel.html#getDataElements(int%5B%5D,%20int,%20java.lang.Object))(int[] components, int offset, [Object](http://docs.google.com/java/lang/Object.html) pixel)            Returns a data element array representation of a pixel in this ColorModel given an array of unnormalized color/alpha components. |
| [Object](http://docs.google.com/java/lang/Object.html) | [**getDataElements**](http://docs.google.com/java/awt/image/IndexColorModel.html#getDataElements(int,%20java.lang.Object))(int rgb, [Object](http://docs.google.com/java/lang/Object.html) pixel)            Returns a data element array representation of a pixel in this ColorModel, given an integer pixel representation in the default RGB color model. |
| int | [**getGreen**](http://docs.google.com/java/awt/image/IndexColorModel.html#getGreen(int))(int pixel)            Returns the green color component for the specified pixel, scaled from 0 to 255 in the default RGB ColorSpace, sRGB. |
| void | [**getGreens**](http://docs.google.com/java/awt/image/IndexColorModel.html#getGreens(byte%5B%5D))(byte[] g)            Copies the array of green color components into the specified array. |
| int | [**getMapSize**](http://docs.google.com/java/awt/image/IndexColorModel.html#getMapSize())()            Returns the size of the color/alpha component arrays in this IndexColorModel. |
| int | [**getRed**](http://docs.google.com/java/awt/image/IndexColorModel.html#getRed(int))(int pixel)            Returns the red color component for the specified pixel, scaled from 0 to 255 in the default RGB ColorSpace, sRGB. |
| void | [**getReds**](http://docs.google.com/java/awt/image/IndexColorModel.html#getReds(byte%5B%5D))(byte[] r)            Copies the array of red color components into the specified array. |
| int | [**getRGB**](http://docs.google.com/java/awt/image/IndexColorModel.html#getRGB(int))(int pixel)            Returns the color/alpha components of the pixel in the default RGB color model format. |
| void | [**getRGBs**](http://docs.google.com/java/awt/image/IndexColorModel.html#getRGBs(int%5B%5D))(int[] rgb)            Converts data for each index from the color and alpha component arrays to an int in the default RGB ColorModel format and copies the resulting 32-bit ARGB values into the specified array. |
| int | [**getTransparency**](http://docs.google.com/java/awt/image/IndexColorModel.html#getTransparency())()            Returns the transparency. |
| int | [**getTransparentPixel**](http://docs.google.com/java/awt/image/IndexColorModel.html#getTransparentPixel())()            Returns the index of a transparent pixel in this IndexColorModel or -1 if there is no pixel with an alpha value of 0. |
| [BigInteger](http://docs.google.com/java/math/BigInteger.html) | [**getValidPixels**](http://docs.google.com/java/awt/image/IndexColorModel.html#getValidPixels())()            Returns a BigInteger that indicates the valid/invalid pixels in the colormap. |
| boolean | [**isCompatibleRaster**](http://docs.google.com/java/awt/image/IndexColorModel.html#isCompatibleRaster(java.awt.image.Raster))([Raster](http://docs.google.com/java/awt/image/Raster.html) raster)            Returns true if raster is compatible with this ColorModel or false if it is not compatible with this ColorModel. |
| boolean | [**isCompatibleSampleModel**](http://docs.google.com/java/awt/image/IndexColorModel.html#isCompatibleSampleModel(java.awt.image.SampleModel))([SampleModel](http://docs.google.com/java/awt/image/SampleModel.html) sm)            Checks if the specified SampleModel is compatible with this ColorModel. |
| boolean | [**isValid**](http://docs.google.com/java/awt/image/IndexColorModel.html#isValid())()            Returns whether or not all of the pixels are valid. |
| boolean | [**isValid**](http://docs.google.com/java/awt/image/IndexColorModel.html#isValid(int))(int pixel)            Returns whether or not the pixel is valid. |
| [String](http://docs.google.com/java/lang/String.html) | [**toString**](http://docs.google.com/java/awt/image/IndexColorModel.html#toString())()            Returns the String representation of the contents of this ColorModelobject. |

| **Methods inherited from class java.awt.image.**[**ColorModel**](http://docs.google.com/java/awt/image/ColorModel.html) |
| --- |
| [coerceData](http://docs.google.com/java/awt/image/ColorModel.html#coerceData(java.awt.image.WritableRaster,%20boolean)), [equals](http://docs.google.com/java/awt/image/ColorModel.html#equals(java.lang.Object)), [getAlpha](http://docs.google.com/java/awt/image/ColorModel.html#getAlpha(java.lang.Object)), [getAlphaRaster](http://docs.google.com/java/awt/image/ColorModel.html#getAlphaRaster(java.awt.image.WritableRaster)), [getBlue](http://docs.google.com/java/awt/image/ColorModel.html#getBlue(java.lang.Object)), [getColorSpace](http://docs.google.com/java/awt/image/ColorModel.html#getColorSpace()), [getComponentSize](http://docs.google.com/java/awt/image/ColorModel.html#getComponentSize(int)), [getDataElement](http://docs.google.com/java/awt/image/ColorModel.html#getDataElement(float%5B%5D,%20int)), [getDataElements](http://docs.google.com/java/awt/image/ColorModel.html#getDataElements(float%5B%5D,%20int,%20java.lang.Object)), [getGreen](http://docs.google.com/java/awt/image/ColorModel.html#getGreen(java.lang.Object)), [getNormalizedComponents](http://docs.google.com/java/awt/image/ColorModel.html#getNormalizedComponents(int%5B%5D,%20int,%20float%5B%5D,%20int)), [getNormalizedComponents](http://docs.google.com/java/awt/image/ColorModel.html#getNormalizedComponents(java.lang.Object,%20float%5B%5D,%20int)), [getNumColorComponents](http://docs.google.com/java/awt/image/ColorModel.html#getNumColorComponents()), [getNumComponents](http://docs.google.com/java/awt/image/ColorModel.html#getNumComponents()), [getPixelSize](http://docs.google.com/java/awt/image/ColorModel.html#getPixelSize()), [getRed](http://docs.google.com/java/awt/image/ColorModel.html#getRed(java.lang.Object)), [getRGB](http://docs.google.com/java/awt/image/ColorModel.html#getRGB(java.lang.Object)), [getRGBdefault](http://docs.google.com/java/awt/image/ColorModel.html#getRGBdefault()), [getTransferType](http://docs.google.com/java/awt/image/ColorModel.html#getTransferType()), [getUnnormalizedComponents](http://docs.google.com/java/awt/image/ColorModel.html#getUnnormalizedComponents(float%5B%5D,%20int,%20int%5B%5D,%20int)), [hasAlpha](http://docs.google.com/java/awt/image/ColorModel.html#hasAlpha()), [hashCode](http://docs.google.com/java/awt/image/ColorModel.html#hashCode()), [isAlphaPremultiplied](http://docs.google.com/java/awt/image/ColorModel.html#isAlphaPremultiplied()) |

| **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()), [getClass](http://docs.google.com/java/lang/Object.html#getClass()), [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)) |

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

### IndexColorModel

public **IndexColorModel**(int bits,  
 int size,  
 byte[] r,  
 byte[] g,  
 byte[] b)

Constructs an IndexColorModel from the specified arrays of red, green, and blue components. Pixels described by this color model all have alpha components of 255 unnormalized (1.0 normalized), which means they are fully opaque. All of the arrays specifying the color components must have at least the specified number of entries. The ColorSpace is the default sRGB space. Since there is no alpha information in any of the arguments to this constructor, the transparency value is always Transparency.OPAQUE. The transfer type is the smallest of DataBuffer.TYPE\_BYTE or DataBuffer.TYPE\_USHORT that can hold a single pixel.

**Parameters:**bits - the number of bits each pixel occupiessize - the size of the color component arraysr - the array of red color componentsg - the array of green color componentsb - the array of blue color components **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if bits is less than 1 or greater than 16 [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if size is less than 1

### IndexColorModel

public **IndexColorModel**(int bits,  
 int size,  
 byte[] r,  
 byte[] g,  
 byte[] b,  
 int trans)

Constructs an IndexColorModel from the given arrays of red, green, and blue components. Pixels described by this color model all have alpha components of 255 unnormalized (1.0 normalized), which means they are fully opaque, except for the indicated pixel to be made transparent. All of the arrays specifying the color components must have at least the specified number of entries. The ColorSpace is the default sRGB space. The transparency value may be Transparency.OPAQUE or Transparency.BITMASK depending on the arguments, as specified in the [class description](#3znysh7) above. The transfer type is the smallest of DataBuffer.TYPE\_BYTE or DataBuffer.TYPE\_USHORT that can hold a single pixel.

**Parameters:**bits - the number of bits each pixel occupiessize - the size of the color component arraysr - the array of red color componentsg - the array of green color componentsb - the array of blue color componentstrans - the index of the transparent pixel **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if bits is less than 1 or greater than 16 [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if size is less than 1

### IndexColorModel

public **IndexColorModel**(int bits,  
 int size,  
 byte[] r,  
 byte[] g,  
 byte[] b,  
 byte[] a)

Constructs an IndexColorModel from the given arrays of red, green, blue and alpha components. All of the arrays specifying the components must have at least the specified number of entries. The ColorSpace is the default sRGB space. The transparency value may be any of Transparency.OPAQUE, Transparency.BITMASK, or Transparency.TRANSLUCENT depending on the arguments, as specified in the [class description](#3znysh7) above. The transfer type is the smallest of DataBuffer.TYPE\_BYTE or DataBuffer.TYPE\_USHORT that can hold a single pixel.

**Parameters:**bits - the number of bits each pixel occupiessize - the size of the color component arraysr - the array of red color componentsg - the array of green color componentsb - the array of blue color componentsa - the array of alpha value components **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if bits is less than 1 or greater than 16 [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if size is less than 1

### IndexColorModel

public **IndexColorModel**(int bits,  
 int size,  
 byte[] cmap,  
 int start,  
 boolean hasalpha)

Constructs an IndexColorModel from a single array of interleaved red, green, blue and optional alpha components. The array must have enough values in it to fill all of the needed component arrays of the specified size. The ColorSpace is the default sRGB space. The transparency value may be any of Transparency.OPAQUE, Transparency.BITMASK, or Transparency.TRANSLUCENT depending on the arguments, as specified in the [class description](#3znysh7) above. The transfer type is the smallest of DataBuffer.TYPE\_BYTE or DataBuffer.TYPE\_USHORT that can hold a single pixel.

**Parameters:**bits - the number of bits each pixel occupiessize - the size of the color component arrayscmap - the array of color componentsstart - the starting offset of the first color componenthasalpha - indicates whether alpha values are contained in the cmap array **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if bits is less than 1 or greater than 16 [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if size is less than 1

### IndexColorModel

public **IndexColorModel**(int bits,  
 int size,  
 byte[] cmap,  
 int start,  
 boolean hasalpha,  
 int trans)

Constructs an IndexColorModel from a single array of interleaved red, green, blue and optional alpha components. The specified transparent index represents a pixel that is made entirely transparent regardless of any alpha value specified for it. The array must have enough values in it to fill all of the needed component arrays of the specified size. The ColorSpace is the default sRGB space. The transparency value may be any of Transparency.OPAQUE, Transparency.BITMASK, or Transparency.TRANSLUCENT depending on the arguments, as specified in the [class description](#3znysh7) above. The transfer type is the smallest of DataBuffer.TYPE\_BYTE or DataBuffer.TYPE\_USHORT that can hold a single pixel.

**Parameters:**bits - the number of bits each pixel occupiessize - the size of the color component arrayscmap - the array of color componentsstart - the starting offset of the first color componenthasalpha - indicates whether alpha values are contained in the cmap arraytrans - the index of the fully transparent pixel **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if bits is less than 1 or greater than 16 [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if size is less than 1

### IndexColorModel

public **IndexColorModel**(int bits,  
 int size,  
 int[] cmap,  
 int start,  
 boolean hasalpha,  
 int trans,  
 int transferType)

Constructs an IndexColorModel from an array of ints where each int is comprised of red, green, blue, and optional alpha components in the default RGB color model format. The specified transparent index represents a pixel that is made entirely transparent regardless of any alpha value specified for it. The array must have enough values in it to fill all of the needed component arrays of the specified size. The ColorSpace is the default sRGB space. The transparency value may be any of Transparency.OPAQUE, Transparency.BITMASK, or Transparency.TRANSLUCENT depending on the arguments, as specified in the [class description](#3znysh7) above.

**Parameters:**bits - the number of bits each pixel occupiessize - the size of the color component arrayscmap - the array of color componentsstart - the starting offset of the first color componenthasalpha - indicates whether alpha values are contained in the cmap arraytrans - the index of the fully transparent pixeltransferType - the data type of the array used to represent pixel values. The data type must be either DataBuffer.TYPE\_BYTE or DataBuffer.TYPE\_USHORT. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if bits is less than 1 or greater than 16 [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if size is less than 1 [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if transferType is not one of DataBuffer.TYPE\_BYTE or DataBuffer.TYPE\_USHORT

### IndexColorModel

public **IndexColorModel**(int bits,  
 int size,  
 int[] cmap,  
 int start,  
 int transferType,  
 [BigInteger](http://docs.google.com/java/math/BigInteger.html) validBits)

Constructs an IndexColorModel from an int array where each int is comprised of red, green, blue, and alpha components in the default RGB color model format. The array must have enough values in it to fill all of the needed component arrays of the specified size. The ColorSpace is the default sRGB space. The transparency value may be any of Transparency.OPAQUE, Transparency.BITMASK, or Transparency.TRANSLUCENT depending on the arguments, as specified in the [class description](#3znysh7) above. The transfer type must be one of DataBuffer.TYPE\_BYTE DataBuffer.TYPE\_USHORT. The BigInteger object specifies the valid/invalid pixels in the cmap array. A pixel is valid if the BigInteger value at that index is set, and is invalid if the BigInteger bit at that index is not set.

**Parameters:**bits - the number of bits each pixel occupiessize - the size of the color component arraycmap - the array of color componentsstart - the starting offset of the first color componenttransferType - the specified data typevalidBits - a BigInteger object. If a bit is set in the BigInteger, the pixel at that index is valid. If a bit is not set, the pixel at that index is considered invalid. If null, all pixels are valid. Only bits from 0 to the map size are considered. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if bits is less than 1 or greater than 16 [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if size is less than 1 [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if transferType is not one of DataBuffer.TYPE\_BYTE or DataBuffer.TYPE\_USHORT**Since:** 1.3

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

### 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)**Overrides:**[getTransparency](http://docs.google.com/java/awt/image/ColorModel.html#getTransparency()) in class [ColorModel](http://docs.google.com/java/awt/image/ColorModel.html) **Returns:**the transparency of this IndexColorModel**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)

### getComponentSize

public int[] **getComponentSize**()

Returns an array of the number of bits for each color/alpha component. The array contains the color components in the order red, green, blue, followed by the alpha component, if present.

**Overrides:**[getComponentSize](http://docs.google.com/java/awt/image/ColorModel.html#getComponentSize()) in class [ColorModel](http://docs.google.com/java/awt/image/ColorModel.html) **Returns:**an array containing the number of bits of each color and alpha component of this IndexColorModel

### getMapSize

public final int **getMapSize**()

Returns the size of the color/alpha component arrays in this IndexColorModel.

**Returns:**the size of the color and alpha component arrays.

### getTransparentPixel

public final int **getTransparentPixel**()

Returns the index of a transparent pixel in this IndexColorModel or -1 if there is no pixel with an alpha value of 0. If a transparent pixel was explicitly specified in one of the constructors by its index, then that index will be preferred, otherwise, the index of any pixel which happens to be fully transparent may be returned.

**Returns:**the index of a transparent pixel in this IndexColorModel object, or -1 if there is no such pixel

### getReds

public final void **getReds**(byte[] r)

Copies the array of red color components into the specified array. Only the initial entries of the array as specified by [getMapSize](http://docs.google.com/java/awt/image/IndexColorModel.html#getMapSize()) are written.

**Parameters:**r - the specified array into which the elements of the array of red color components are copied

### getGreens

public final void **getGreens**(byte[] g)

Copies the array of green color components into the specified array. Only the initial entries of the array as specified by getMapSize are written.

**Parameters:**g - the specified array into which the elements of the array of green color components are copied

### getBlues

public final void **getBlues**(byte[] b)

Copies the array of blue color components into the specified array. Only the initial entries of the array as specified by getMapSize are written.

**Parameters:**b - the specified array into which the elements of the array of blue color components are copied

### getAlphas

public final void **getAlphas**(byte[] a)

Copies the array of alpha transparency components into the specified array. Only the initial entries of the array as specified by getMapSize are written.

**Parameters:**a - the specified array into which the elements of the array of alpha components are copied

### getRGBs

public final void **getRGBs**(int[] rgb)

Converts data for each index from the color and alpha component arrays to an int in the default RGB ColorModel format and copies the resulting 32-bit ARGB values into the specified array. Only the initial entries of the array as specified by getMapSize are written.

**Parameters:**rgb - the specified array into which the converted ARGB values from this array of color and alpha components are copied.

### getRed

public final int **getRed**(int pixel)

Returns the red color component for the specified pixel, scaled from 0 to 255 in the default RGB ColorSpace, sRGB. The pixel value is specified as an int. Only the lower *n* bits of the pixel value, as specified in the [class description](#2et92p0) above, are used to calculate the returned value. The returned value is a non pre-multiplied value.

**Specified by:**[getRed](http://docs.google.com/java/awt/image/ColorModel.html#getRed(int)) in class [ColorModel](http://docs.google.com/java/awt/image/ColorModel.html) **Parameters:**pixel - the specified pixel **Returns:**the value of the red color component for the specified pixel

### getGreen

public final int **getGreen**(int pixel)

Returns the green color component for the specified pixel, scaled from 0 to 255 in the default RGB ColorSpace, sRGB. The pixel value is specified as an int. Only the lower *n* bits of the pixel value, as specified in the [class description](#2et92p0) above, are used to calculate the returned value. The returned value is a non pre-multiplied value.

**Specified by:**[getGreen](http://docs.google.com/java/awt/image/ColorModel.html#getGreen(int)) in class [ColorModel](http://docs.google.com/java/awt/image/ColorModel.html) **Parameters:**pixel - the specified pixel **Returns:**the value of the green color component for the specified pixel

### getBlue

public final int **getBlue**(int pixel)

Returns the blue color component for the specified pixel, scaled from 0 to 255 in the default RGB ColorSpace, sRGB. The pixel value is specified as an int. Only the lower *n* bits of the pixel value, as specified in the [class description](#2et92p0) above, are used to calculate the returned value. The returned value is a non pre-multiplied value.

**Specified by:**[getBlue](http://docs.google.com/java/awt/image/ColorModel.html#getBlue(int)) in class [ColorModel](http://docs.google.com/java/awt/image/ColorModel.html) **Parameters:**pixel - the specified pixel **Returns:**the value of the blue color component for the specified pixel

### getAlpha

public final int **getAlpha**(int pixel)

Returns the alpha component for the specified pixel, scaled from 0 to 255. The pixel value is specified as an int. Only the lower *n* bits of the pixel value, as specified in the [class description](#2et92p0) above, are used to calculate the returned value.

**Specified by:**[getAlpha](http://docs.google.com/java/awt/image/ColorModel.html#getAlpha(int)) in class [ColorModel](http://docs.google.com/java/awt/image/ColorModel.html) **Parameters:**pixel - the specified pixel **Returns:**the value of the alpha component for the specified pixel

### getRGB

public final int **getRGB**(int pixel)

Returns the color/alpha components of the pixel in the default RGB color model format. The pixel value is specified as an int. Only the lower *n* bits of the pixel value, as specified in the [class description](#2et92p0) above, are used to calculate the returned value. The returned value is in a non pre-multiplied format.

**Overrides:**[getRGB](http://docs.google.com/java/awt/image/ColorModel.html#getRGB(int)) in class [ColorModel](http://docs.google.com/java/awt/image/ColorModel.html) **Parameters:**pixel - the specified pixel **Returns:**the color and alpha components of the specified pixel**See Also:**[ColorModel.getRGBdefault()](http://docs.google.com/java/awt/image/ColorModel.html#getRGBdefault())

### getDataElements

public [Object](http://docs.google.com/java/lang/Object.html) **getDataElements**(int rgb,  
 [Object](http://docs.google.com/java/lang/Object.html) pixel)

Returns a data element array representation of a pixel in this ColorModel, given an integer pixel representation in the default RGB color model. This array can then be passed to the [setDataElements](http://docs.google.com/java/awt/image/WritableRaster.html#setDataElements(int,%20int,%20java.lang.Object)) method of a [WritableRaster](http://docs.google.com/java/awt/image/WritableRaster.html) object. If the pixel variable is null, a new array is allocated. If pixel is not null, it must be a primitive array of type transferType; otherwise, a ClassCastException is thrown. An ArrayIndexOutOfBoundsException is thrown if pixel is not large enough to hold a pixel value for this ColorModel. The pixel array is returned.

Since IndexColorModel can be subclassed, subclasses inherit the implementation of this method and if they don't override it then they throw an exception if they use an unsupported transferType.

**Overrides:**[getDataElements](http://docs.google.com/java/awt/image/ColorModel.html#getDataElements(int,%20java.lang.Object)) in class [ColorModel](http://docs.google.com/java/awt/image/ColorModel.html) **Parameters:**rgb - the integer pixel representation in the default RGB color modelpixel - the specified pixel **Returns:**an array representation of the specified pixel in this IndexColorModel. **Throws:** [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if pixel is not a primitive array of type transferType [ArrayIndexOutOfBoundsException](http://docs.google.com/java/lang/ArrayIndexOutOfBoundsException.html) - if pixel is not large enough to hold a pixel value for this ColorModel [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if transferType is invalid**See Also:**[WritableRaster.setDataElements(int, int, java.lang.Object)](http://docs.google.com/java/awt/image/WritableRaster.html#setDataElements(int,%20int,%20java.lang.Object)), [SampleModel.setDataElements(int, int, java.lang.Object, java.awt.image.DataBuffer)](http://docs.google.com/java/awt/image/SampleModel.html#setDataElements(int,%20int,%20java.lang.Object,%20java.awt.image.DataBuffer))

### getComponents

public int[] **getComponents**(int pixel,  
 int[] components,  
 int offset)

Returns an array of unnormalized color/alpha components for a specified pixel in this ColorModel. The pixel value is specified as an int. If the components array is null, a new array is allocated that contains offset + getNumComponents() elements. The components array is returned, with the alpha component included only if hasAlpha returns true. Color/alpha components are stored in the components array starting at offset even if the array is allocated by this method. An ArrayIndexOutOfBoundsException is thrown if the components array is not null and is not large enough to hold all the color and alpha components starting at offset.

**Overrides:**[getComponents](http://docs.google.com/java/awt/image/ColorModel.html#getComponents(int,%20int%5B%5D,%20int)) in class [ColorModel](http://docs.google.com/java/awt/image/ColorModel.html) **Parameters:**pixel - the specified pixelcomponents - the array to receive the color and alpha components of the specified pixeloffset - the offset into the components array at which to start storing the color and alpha components **Returns:**an array containing the color and alpha components of the specified pixel starting at the specified offset.**See Also:**[ColorModel.hasAlpha()](http://docs.google.com/java/awt/image/ColorModel.html#hasAlpha()), [ColorModel.getNumComponents()](http://docs.google.com/java/awt/image/ColorModel.html#getNumComponents())

### getComponents

public int[] **getComponents**([Object](http://docs.google.com/java/lang/Object.html) pixel,  
 int[] components,  
 int offset)

Returns an array of unnormalized color/alpha components for a specified pixel in this ColorModel. The pixel value is specified by an array of data elements of type transferType passed in as an object reference. If pixel is not a primitive array of type transferType, a ClassCastException is thrown. An ArrayIndexOutOfBoundsException is thrown if pixel is not large enough to hold a pixel value for this ColorModel. If the components array is null, a new array is allocated that contains offset + getNumComponents() elements. The components array is returned, with the alpha component included only if hasAlpha returns true. Color/alpha components are stored in the components array starting at offset even if the array is allocated by this method. An ArrayIndexOutOfBoundsException is also thrown if the components array is not null and is not large enough to hold all the color and alpha components starting at offset.

Since IndexColorModel can be subclassed, subclasses inherit the implementation of this method and if they don't override it then they throw an exception if they use an unsupported transferType.

**Overrides:**[getComponents](http://docs.google.com/java/awt/image/ColorModel.html#getComponents(java.lang.Object,%20int%5B%5D,%20int)) in class [ColorModel](http://docs.google.com/java/awt/image/ColorModel.html) **Parameters:**pixel - the specified pixelcomponents - an array that receives the color and alpha components of the specified pixeloffset - the index into the components array at which to begin storing the color and alpha components of the specified pixel **Returns:**an array containing the color and alpha components of the specified pixel starting at the specified offset. **Throws:** [ArrayIndexOutOfBoundsException](http://docs.google.com/java/lang/ArrayIndexOutOfBoundsException.html) - if pixel is not large enough to hold a pixel value for this ColorModel or if the components array is not null and is not large enough to hold all the color and alpha components starting at offset [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if pixel is not a primitive array of type transferType [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if transferType is not one of the supported transer types**See Also:**[ColorModel.hasAlpha()](http://docs.google.com/java/awt/image/ColorModel.html#hasAlpha()), [ColorModel.getNumComponents()](http://docs.google.com/java/awt/image/ColorModel.html#getNumComponents())

### getDataElement

public int **getDataElement**(int[] components,  
 int offset)

Returns a pixel value represented as an int in this ColorModel given an array of unnormalized color/alpha components. An ArrayIndexOutOfBoundsException is thrown if the components array is not large enough to hold all of the color and alpha components starting at offset. Since ColorModel can be subclassed, subclasses inherit the implementation of this method and if they don't override it then they throw an exception if they use an unsupported transferType.

**Overrides:**[getDataElement](http://docs.google.com/java/awt/image/ColorModel.html#getDataElement(int%5B%5D,%20int)) in class [ColorModel](http://docs.google.com/java/awt/image/ColorModel.html) **Parameters:**components - an array of unnormalized color and alpha componentsoffset - the index into components at which to begin retrieving the color and alpha components **Returns:**an int pixel value in this ColorModel corresponding to the specified components. **Throws:** [ArrayIndexOutOfBoundsException](http://docs.google.com/java/lang/ArrayIndexOutOfBoundsException.html) - if the components array is not large enough to hold all of the color and alpha components starting at offset [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if transferType is invalid

### getDataElements

public [Object](http://docs.google.com/java/lang/Object.html) **getDataElements**(int[] components,  
 int offset,  
 [Object](http://docs.google.com/java/lang/Object.html) pixel)

Returns a data element array representation of a pixel in this ColorModel given an array of unnormalized color/alpha components. This array can then be passed to the setDataElements method of a WritableRaster object. An ArrayIndexOutOfBoundsException is thrown if the components array is not large enough to hold all of the color and alpha components starting at offset. If the pixel variable is null, a new array is allocated. If pixel is not null, it must be a primitive array of type transferType; otherwise, a ClassCastException is thrown. An ArrayIndexOutOfBoundsException is thrown if pixel is not large enough to hold a pixel value for this ColorModel.

Since IndexColorModel can be subclassed, subclasses inherit the implementation of this method and if they don't override it then they throw an exception if they use an unsupported transferType

**Overrides:**[getDataElements](http://docs.google.com/java/awt/image/ColorModel.html#getDataElements(int%5B%5D,%20int,%20java.lang.Object)) in class [ColorModel](http://docs.google.com/java/awt/image/ColorModel.html) **Parameters:**components - an array of unnormalized color and alpha componentsoffset - the index into components at which to begin retrieving color and alpha componentspixel - the Object representing an array of color and alpha components **Returns:**an Object representing an array of color and alpha components. **Throws:** [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if pixel is not a primitive array of type transferType [ArrayIndexOutOfBoundsException](http://docs.google.com/java/lang/ArrayIndexOutOfBoundsException.html) - if pixel is not large enough to hold a pixel value for this ColorModel or the components array is not large enough to hold all of the color and alpha components starting at offset [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if transferType is not one of the supported transer types**See Also:**[WritableRaster.setDataElements(int, int, java.lang.Object)](http://docs.google.com/java/awt/image/WritableRaster.html#setDataElements(int,%20int,%20java.lang.Object)), [SampleModel.setDataElements(int, int, java.lang.Object, java.awt.image.DataBuffer)](http://docs.google.com/java/awt/image/SampleModel.html#setDataElements(int,%20int,%20java.lang.Object,%20java.awt.image.DataBuffer))

### createCompatibleWritableRaster

public [WritableRaster](http://docs.google.com/java/awt/image/WritableRaster.html) **createCompatibleWritableRaster**(int w,  
 int h)

Creates a WritableRaster with the specified width and height that has a data layout (SampleModel) compatible with this ColorModel. This method only works for color models with 16 or fewer bits per pixel.

Since IndexColorModel can be subclassed, any subclass that supports greater than 16 bits per pixel must override this method.

**Overrides:**[createCompatibleWritableRaster](http://docs.google.com/java/awt/image/ColorModel.html#createCompatibleWritableRaster(int,%20int)) in class [ColorModel](http://docs.google.com/java/awt/image/ColorModel.html) **Parameters:**w - the width to apply to the new WritableRasterh - the height to apply to the new WritableRaster **Returns:**a WritableRaster object with the specified width and height. **Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if the number of bits in a pixel is greater than 16**See Also:**[WritableRaster](http://docs.google.com/java/awt/image/WritableRaster.html), [SampleModel](http://docs.google.com/java/awt/image/SampleModel.html)

### isCompatibleRaster

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

Returns true if raster is compatible with this ColorModel or false if it is not compatible with this ColorModel.

**Overrides:**[isCompatibleRaster](http://docs.google.com/java/awt/image/ColorModel.html#isCompatibleRaster(java.awt.image.Raster)) in class [ColorModel](http://docs.google.com/java/awt/image/ColorModel.html) **Parameters:**raster - the [Raster](http://docs.google.com/java/awt/image/Raster.html) object to test for compatibility **Returns:**true if raster is compatible with this ColorModel; false otherwise.

### createCompatibleSampleModel

public [SampleModel](http://docs.google.com/java/awt/image/SampleModel.html) **createCompatibleSampleModel**(int w,  
 int h)

Creates a SampleModel with the specified width and height that has a data layout compatible with this ColorModel.

**Overrides:**[createCompatibleSampleModel](http://docs.google.com/java/awt/image/ColorModel.html#createCompatibleSampleModel(int,%20int)) in class [ColorModel](http://docs.google.com/java/awt/image/ColorModel.html) **Parameters:**w - the width to apply to the new SampleModelh - the height to apply to the new SampleModel **Returns:**a SampleModel object with the specified width and height. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if w or h is not greater than 0**See Also:**[SampleModel](http://docs.google.com/java/awt/image/SampleModel.html)

### isCompatibleSampleModel

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

Checks if the specified SampleModel is compatible with this ColorModel. If sm is null, this method returns false.

**Overrides:**[isCompatibleSampleModel](http://docs.google.com/java/awt/image/ColorModel.html#isCompatibleSampleModel(java.awt.image.SampleModel)) in class [ColorModel](http://docs.google.com/java/awt/image/ColorModel.html) **Parameters:**sm - the specified SampleModel, or null **Returns:**true if the specified SampleModel is compatible with this ColorModel; false otherwise.**See Also:**[SampleModel](http://docs.google.com/java/awt/image/SampleModel.html)

### convertToIntDiscrete

public [BufferedImage](http://docs.google.com/java/awt/image/BufferedImage.html) **convertToIntDiscrete**([Raster](http://docs.google.com/java/awt/image/Raster.html) raster,  
 boolean forceARGB)

Returns a new BufferedImage of TYPE\_INT\_ARGB or TYPE\_INT\_RGB that has a Raster with pixel data computed by expanding the indices in the source Raster using the color/alpha component arrays of this ColorModel. Only the lower *n* bits of each index value in the source Raster, as specified in the [class description](#2et92p0) above, are used to compute the color/alpha values in the returned image. If forceARGB is true, a TYPE\_INT\_ARGB image is returned regardless of whether or not this ColorModel has an alpha component array or a transparent pixel.

**Parameters:**raster - the specified RasterforceARGB - if true, the returned BufferedImage is TYPE\_INT\_ARGB; otherwise it is TYPE\_INT\_RGB **Returns:**a BufferedImage created with the specified Raster **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the raster argument is not compatible with this IndexColorModel

### isValid

public boolean **isValid**(int pixel)

Returns whether or not the pixel is valid.

**Parameters:**pixel - the specified pixel value **Returns:**true if pixel is valid; false otherwise.**Since:** 1.3

### isValid

public boolean **isValid**()

Returns whether or not all of the pixels are valid.

**Returns:**true if all pixels are valid; false otherwise.**Since:** 1.3

### getValidPixels

public [BigInteger](http://docs.google.com/java/math/BigInteger.html) **getValidPixels**()

Returns a BigInteger that indicates the valid/invalid pixels in the colormap. A bit is valid if the BigInteger value at that index is set, and is invalid if the BigInteger value at that index is not set. The only valid ranges to query in the BigInteger are between 0 and the map size.

**Returns:**a BigInteger indicating the valid/invalid pixels.**Since:** 1.3

### finalize

public void **finalize**()

Disposes of system resources associated with this ColorModel once this ColorModel is no longer referenced.

**Overrides:**[finalize](http://docs.google.com/java/awt/image/ColorModel.html#finalize()) in class [ColorModel](http://docs.google.com/java/awt/image/ColorModel.html)

### toString

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

Returns the String representation of the contents of this ColorModelobject.

**Overrides:**[toString](http://docs.google.com/java/awt/image/ColorModel.html#toString()) in class [ColorModel](http://docs.google.com/java/awt/image/ColorModel.html) **Returns:**a String representing the contents of this ColorModel object.

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/IndexColorModel.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/ImagingOpException.html)   [**NEXT CLASS**](http://docs.google.com/java/awt/image/Kernel.html) | [**FRAMES**](http://docs.google.com/index.html?java/awt/image/IndexColorModel.html)    [**NO FRAMES**](http://docs.google.com/IndexColorModel.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3dy6vkm) | [CONSTR](#4d34og8) | [METHOD](#2s8eyo1) | DETAIL: FIELD | [CONSTR](#26in1rg) | [METHOD](#1y810tw) |

[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).