| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/ColorModel.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/ColorConvertOp.html)   [**NEXT CLASS**](http://docs.google.com/java/awt/image/ComponentColorModel.html) | [**FRAMES**](http://docs.google.com/index.html?java/awt/image/ColorModel.html)    [**NO FRAMES**](http://docs.google.com/ColorModel.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | [CONSTR](#tyjcwt) | [METHOD](#3dy6vkm) | DETAIL: [FIELD](#4d34og8) | [CONSTR](#3rdcrjn) | [METHOD](#35nkun2) |

## **java.awt.image**

Class ColorModel

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

**All Implemented Interfaces:** [Transparency](http://docs.google.com/java/awt/Transparency.html) **Direct Known Subclasses:** [ComponentColorModel](http://docs.google.com/java/awt/image/ComponentColorModel.html), [IndexColorModel](http://docs.google.com/java/awt/image/IndexColorModel.html), [PackedColorModel](http://docs.google.com/java/awt/image/PackedColorModel.html)

public abstract class **ColorModel**extends [Object](http://docs.google.com/java/lang/Object.html)implements [Transparency](http://docs.google.com/java/awt/Transparency.html)

The ColorModel abstract class encapsulates the methods for translating a pixel value to color components (for example, red, green, and blue) and an alpha component. In order to render an image to the screen, a printer, or another image, pixel values must be converted to color and alpha components. As arguments to or return values from methods of this class, pixels are represented as 32-bit ints or as arrays of primitive types. The number, order, and interpretation of color components for a ColorModel is specified by its ColorSpace. A ColorModel used with pixel data that does not include alpha information treats all pixels as opaque, which is an alpha value of 1.0.

This ColorModel class supports two representations of pixel values. A pixel value can be a single 32-bit int or an array of primitive types. The Java(tm) Platform 1.0 and 1.1 APIs represented pixels as single byte or single int values. For purposes of the ColorModel class, pixel value arguments were passed as ints. The Java(tm) 2 Platform API introduced additional classes for representing images. With [BufferedImage](http://docs.google.com/java/awt/image/BufferedImage.html) or [RenderedImage](http://docs.google.com/java/awt/image/RenderedImage.html) objects, based on [Raster](http://docs.google.com/java/awt/image/Raster.html) and [SampleModel](http://docs.google.com/java/awt/image/SampleModel.html) classes, pixel values might not be conveniently representable as a single int. Consequently, ColorModel now has methods that accept pixel values represented as arrays of primitive types. The primitive type used by a particular ColorModel object is called its transfer type.

ColorModel objects used with images for which pixel values are not conveniently representable as a single int throw an [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) when methods taking a single int pixel argument are called. Subclasses of ColorModel must specify the conditions under which this occurs. This does not occur with [DirectColorModel](http://docs.google.com/java/awt/image/DirectColorModel.html) or [IndexColorModel](http://docs.google.com/java/awt/image/IndexColorModel.html) objects.

Currently, the transfer types supported by the Java 2D(tm) API are DataBuffer.TYPE\_BYTE, DataBuffer.TYPE\_USHORT, DataBuffer.TYPE\_INT, DataBuffer.TYPE\_SHORT, DataBuffer.TYPE\_FLOAT, and DataBuffer.TYPE\_DOUBLE. Most rendering operations will perform much faster when using ColorModels and images based on the first three of these types. In addition, some image filtering operations are not supported for ColorModels and images based on the latter three types. The transfer type for a particular ColorModel object is specified when the object is created, either explicitly or by default. All subclasses of ColorModel must specify what the possible transfer types are and how the number of elements in the primitive arrays representing pixels is determined.

For BufferedImages, the transfer type of its Raster and of the Raster object's SampleModel (available from the getTransferType methods of these classes) must match that of the ColorModel. The number of elements in an array representing a pixel for the Raster and SampleModel (available from the getNumDataElements methods of these classes) must match that of the ColorModel.

The algorithm used to convert from pixel values to color and alpha components varies by subclass. For example, there is not necessarily a one-to-one correspondence between samples obtained from the SampleModel of a BufferedImage object's Raster and color/alpha components. Even when there is such a correspondence, the number of bits in a sample is not necessarily the same as the number of bits in the corresponding color/alpha component. Each subclass must specify how the translation from pixel values to color/alpha components is done.

Methods in the ColorModel class use two different representations of color and alpha components - a normalized form and an unnormalized form. In the normalized form, each component is a float value between some minimum and maximum values. For the alpha component, the minimum is 0.0 and the maximum is 1.0. For color components the minimum and maximum values for each component can be obtained from the ColorSpace object. These values will often be 0.0 and 1.0 (e.g. normalized component values for the default sRGB color space range from 0.0 to 1.0), but some color spaces have component values with different upper and lower limits. These limits can be obtained using the getMinValue and getMaxValue methods of the ColorSpace class. Normalized color component values are not premultiplied. All ColorModels must support the normalized form.

In the unnormalized form, each component is an unsigned integral value between 0 and 2n - 1, where n is the number of significant bits for a particular component. If pixel values for a particular ColorModel represent color samples premultiplied by the alpha sample, unnormalized color component values are also premultiplied. The unnormalized form is used only with instances of ColorModel whose ColorSpace has minimum component values of 0.0 for all components and maximum values of 1.0 for all components. The unnormalized form for color and alpha components can be a convenient representation for ColorModels whose normalized component values all lie between 0.0 and 1.0. In such cases the integral value 0 maps to 0.0 and the value 2n - 1 maps to 1.0. In other cases, such as when the normalized component values can be either negative or positive, the unnormalized form is not convenient. Such ColorModel objects throw an [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) when methods involving an unnormalized argument are called. Subclasses of ColorModel must specify the conditions under which this occurs.

**See Also:**[IndexColorModel](http://docs.google.com/java/awt/image/IndexColorModel.html), [ComponentColorModel](http://docs.google.com/java/awt/image/ComponentColorModel.html), [PackedColorModel](http://docs.google.com/java/awt/image/PackedColorModel.html), [DirectColorModel](http://docs.google.com/java/awt/image/DirectColorModel.html), [Image](http://docs.google.com/java/awt/Image.html), [BufferedImage](http://docs.google.com/java/awt/image/BufferedImage.html), [RenderedImage](http://docs.google.com/java/awt/image/RenderedImage.html), [ColorSpace](http://docs.google.com/java/awt/color/ColorSpace.html), [SampleModel](http://docs.google.com/java/awt/image/SampleModel.html), [Raster](http://docs.google.com/java/awt/image/Raster.html), [DataBuffer](http://docs.google.com/java/awt/image/DataBuffer.html)

| **Field Summary** | |
| --- | --- |
| protected  int | [**pixel\_bits**](http://docs.google.com/java/awt/image/ColorModel.html#pixel_bits)            The total number of bits in the pixel. |
| protected  int | [**transferType**](http://docs.google.com/java/awt/image/ColorModel.html#transferType)            Data type of the array used to represent pixel values. |

| **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** | |
| --- | --- |
|  | [**ColorModel**](http://docs.google.com/java/awt/image/ColorModel.html#ColorModel(int))(int bits)            Constructs a ColorModel that translates pixels of the specified number of bits to color/alpha components. |
| protected | [**ColorModel**](http://docs.google.com/java/awt/image/ColorModel.html#ColorModel(int,%20int%5B%5D,%20java.awt.color.ColorSpace,%20boolean,%20boolean,%20int,%20int))(int pixel\_bits, int[] bits, [ColorSpace](http://docs.google.com/java/awt/color/ColorSpace.html) cspace, boolean hasAlpha, boolean isAlphaPremultiplied, int transparency, int transferType)            Constructs a ColorModel that translates pixel values to color/alpha components. |

| **Method Summary** | |
| --- | --- |
| [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))([WritableRaster](http://docs.google.com/java/awt/image/WritableRaster.html) raster, boolean isAlphaPremultiplied)            Forces the raster data to match the state specified in the isAlphaPremultiplied variable, assuming the data is currently correctly described by this ColorModel. |
| [SampleModel](http://docs.google.com/java/awt/image/SampleModel.html) | [**createCompatibleSampleModel**](http://docs.google.com/java/awt/image/ColorModel.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/ColorModel.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. |
| boolean | [**equals**](http://docs.google.com/java/awt/image/ColorModel.html#equals(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) obj)            Tests if the specified Object is an instance of ColorModel and if it equals this ColorModel. |
| void | [**finalize**](http://docs.google.com/java/awt/image/ColorModel.html#finalize())()            Disposes of system resources associated with this ColorModel once this ColorModel is no longer referenced. |
| abstract  int | [**getAlpha**](http://docs.google.com/java/awt/image/ColorModel.html#getAlpha(int))(int pixel)            Returns the alpha component for the specified pixel, scaled from 0 to 255. |
| int | [**getAlpha**](http://docs.google.com/java/awt/image/ColorModel.html#getAlpha(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) inData)            Returns the alpha component for the specified pixel, scaled from 0 to 255. |
| [WritableRaster](http://docs.google.com/java/awt/image/WritableRaster.html) | [**getAlphaRaster**](http://docs.google.com/java/awt/image/ColorModel.html#getAlphaRaster(java.awt.image.WritableRaster))([WritableRaster](http://docs.google.com/java/awt/image/WritableRaster.html) raster)            Returns a Raster representing the alpha channel of an image, extracted from the input Raster, provided that pixel values of this ColorModel represent color and alpha information as separate spatial bands (e.g. |
| abstract  int | [**getBlue**](http://docs.google.com/java/awt/image/ColorModel.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. |
| int | [**getBlue**](http://docs.google.com/java/awt/image/ColorModel.html#getBlue(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) inData)            Returns the blue color component for the specified pixel, scaled from 0 to 255 in the default RGB ColorSpace, sRGB. |
| [ColorSpace](http://docs.google.com/java/awt/color/ColorSpace.html) | [**getColorSpace**](http://docs.google.com/java/awt/image/ColorModel.html#getColorSpace())()            Returns the ColorSpace associated with this ColorModel. |
| int[] | [**getComponents**](http://docs.google.com/java/awt/image/ColorModel.html#getComponents(int,%20int%5B%5D,%20int))(int pixel, int[] components, int offset)            Returns an array of unnormalized color/alpha components given a pixel in this ColorModel. |
| int[] | [**getComponents**](http://docs.google.com/java/awt/image/ColorModel.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 given a pixel in this ColorModel. |
| int[] | [**getComponentSize**](http://docs.google.com/java/awt/image/ColorModel.html#getComponentSize())()            Returns an array of the number of bits per color/alpha component. |
| int | [**getComponentSize**](http://docs.google.com/java/awt/image/ColorModel.html#getComponentSize(int))(int componentIdx)            Returns the number of bits for the specified color/alpha component. |
| int | [**getDataElement**](http://docs.google.com/java/awt/image/ColorModel.html#getDataElement(float%5B%5D,%20int))(float[] normComponents, int normOffset)            Returns a pixel value represented as an int in this ColorModel, given an array of normalized color/alpha components. |
| int | [**getDataElement**](http://docs.google.com/java/awt/image/ColorModel.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/ColorModel.html#getDataElements(float%5B%5D,%20int,%20java.lang.Object))(float[] normComponents, int normOffset, [Object](http://docs.google.com/java/lang/Object.html) obj)            Returns a data element array representation of a pixel in this ColorModel, given an array of normalized color/alpha components. |
| [Object](http://docs.google.com/java/lang/Object.html) | [**getDataElements**](http://docs.google.com/java/awt/image/ColorModel.html#getDataElements(int%5B%5D,%20int,%20java.lang.Object))(int[] components, int offset, [Object](http://docs.google.com/java/lang/Object.html) obj)            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/ColorModel.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. |
| abstract  int | [**getGreen**](http://docs.google.com/java/awt/image/ColorModel.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. |
| int | [**getGreen**](http://docs.google.com/java/awt/image/ColorModel.html#getGreen(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) inData)            Returns the green color component for the specified pixel, scaled from 0 to 255 in the default RGB ColorSpace, sRGB. |
| float[] | [**getNormalizedComponents**](http://docs.google.com/java/awt/image/ColorModel.html#getNormalizedComponents(int%5B%5D,%20int,%20float%5B%5D,%20int))(int[] components, int offset, float[] normComponents, int normOffset)            Returns an array of all of the color/alpha components in normalized form, given an unnormalized component array. |
| float[] | [**getNormalizedComponents**](http://docs.google.com/java/awt/image/ColorModel.html#getNormalizedComponents(java.lang.Object,%20float%5B%5D,%20int))([Object](http://docs.google.com/java/lang/Object.html) pixel, float[] normComponents, int normOffset)            Returns an array of all of the color/alpha components in normalized form, given a pixel in this ColorModel. |
| int | [**getNumColorComponents**](http://docs.google.com/java/awt/image/ColorModel.html#getNumColorComponents())()            Returns the number of color components in this ColorModel. |
| int | [**getNumComponents**](http://docs.google.com/java/awt/image/ColorModel.html#getNumComponents())()            Returns the number of components, including alpha, in this ColorModel. |
| int | [**getPixelSize**](http://docs.google.com/java/awt/image/ColorModel.html#getPixelSize())()            Returns the number of bits per pixel described by this ColorModel. |
| abstract  int | [**getRed**](http://docs.google.com/java/awt/image/ColorModel.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. |
| int | [**getRed**](http://docs.google.com/java/awt/image/ColorModel.html#getRed(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) inData)            Returns the red color component for the specified pixel, scaled from 0 to 255 in the default RGB ColorSpace, sRGB. |
| int | [**getRGB**](http://docs.google.com/java/awt/image/ColorModel.html#getRGB(int))(int pixel)            Returns the color/alpha components of the pixel in the default RGB color model format. |
| int | [**getRGB**](http://docs.google.com/java/awt/image/ColorModel.html#getRGB(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) inData)            Returns the color/alpha components for the specified pixel in the default RGB color model format. |
| static [ColorModel](http://docs.google.com/java/awt/image/ColorModel.html) | [**getRGBdefault**](http://docs.google.com/java/awt/image/ColorModel.html#getRGBdefault())()            Returns a DirectColorModel that describes the default format for integer RGB values used in many of the methods in the AWT image interfaces for the convenience of the programmer. |
| int | [**getTransferType**](http://docs.google.com/java/awt/image/ColorModel.html#getTransferType())()            Returns the transfer type of this ColorModel. |
| int | [**getTransparency**](http://docs.google.com/java/awt/image/ColorModel.html#getTransparency())()            Returns the transparency. |
| int[] | [**getUnnormalizedComponents**](http://docs.google.com/java/awt/image/ColorModel.html#getUnnormalizedComponents(float%5B%5D,%20int,%20int%5B%5D,%20int))(float[] normComponents, int normOffset, int[] components, int offset)            Returns an array of all of the color/alpha components in unnormalized form, given a normalized component array. |
| boolean | [**hasAlpha**](http://docs.google.com/java/awt/image/ColorModel.html#hasAlpha())()            Returns whether or not alpha is supported in this ColorModel. |
| int | [**hashCode**](http://docs.google.com/java/awt/image/ColorModel.html#hashCode())()            Returns the hash code for this ColorModel. |
| boolean | [**isAlphaPremultiplied**](http://docs.google.com/java/awt/image/ColorModel.html#isAlphaPremultiplied())()            Returns whether or not the alpha has been premultiplied in the pixel values to be translated by this ColorModel. |
| boolean | [**isCompatibleRaster**](http://docs.google.com/java/awt/image/ColorModel.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 and false if it is not. |
| boolean | [**isCompatibleSampleModel**](http://docs.google.com/java/awt/image/ColorModel.html#isCompatibleSampleModel(java.awt.image.SampleModel))([SampleModel](http://docs.google.com/java/awt/image/SampleModel.html) sm)            Checks if the SampleModel is compatible with this ColorModel. |
| [String](http://docs.google.com/java/lang/String.html) | [**toString**](http://docs.google.com/java/awt/image/ColorModel.html#toString())()            Returns the String representation of the contents of this ColorModelobject. |

| **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)) |

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

### pixel\_bits

protected int **pixel\_bits**

The total number of bits in the pixel.

### transferType

protected int **transferType**

Data type of the array used to represent pixel values.

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

### ColorModel

public **ColorModel**(int bits)

Constructs a ColorModel that translates pixels of the specified number of bits to color/alpha components. The color space is the default RGB ColorSpace, which is sRGB. Pixel values are assumed to include alpha information. If color and alpha information are represented in the pixel value as separate spatial bands, the color bands are assumed not to be premultiplied with the alpha value. The transparency type is java.awt.Transparency.TRANSLUCENT. The transfer type will be the smallest of DataBuffer.TYPE\_BYTE, DataBuffer.TYPE\_USHORT, or DataBuffer.TYPE\_INT that can hold a single pixel (or DataBuffer.TYPE\_UNDEFINED if bits is greater than 32). Since this constructor has no information about the number of bits per color and alpha component, any subclass calling this constructor should override any method that requires this information.

**Parameters:**bits - the number of bits of a pixel **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the number of bits in bits is less than 1

### ColorModel

protected **ColorModel**(int pixel\_bits,  
 int[] bits,  
 [ColorSpace](http://docs.google.com/java/awt/color/ColorSpace.html) cspace,  
 boolean hasAlpha,  
 boolean isAlphaPremultiplied,  
 int transparency,  
 int transferType)

Constructs a ColorModel that translates pixel values to color/alpha components. Color components will be in the specified ColorSpace. pixel\_bits is the number of bits in the pixel values. The bits array specifies the number of significant bits per color and alpha component. Its length should be the number of components in the ColorSpace if there is no alpha information in the pixel values, or one more than this number if there is alpha information. hasAlpha indicates whether or not alpha information is present. The boolean isAlphaPremultiplied specifies how to interpret pixel values in which color and alpha information are represented as separate spatial bands. If the boolean is true, color samples are assumed to have been multiplied by the alpha sample. The transparency specifies what alpha values can be represented by this color model. The transfer type is the type of primitive array used to represent pixel values. Note that the bits array contains the number of significant bits per color/alpha component after the translation from pixel values. For example, for an IndexColorModel with pixel\_bits equal to 16, the bits array might have four elements with each element set to 8.

**Parameters:**pixel\_bits - the number of bits in the pixel valuesbits - array that specifies the number of significant bits per color and alpha componentcspace - the specified ColorSpacehasAlpha - true if alpha information is present; false otherwiseisAlphaPremultiplied - true if color samples are assumed to be premultiplied by the alpha samples; false otherwisetransparency - what alpha values can be represented by this color modeltransferType - the type of the array used to represent pixel values **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the length of the bit array is less than the number of color or alpha components in this ColorModel, or if the transparency is not a valid value. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the sum of the number of bits in bits is less than 1 or if any of the elements in bits is less than 0.**See Also:**[Transparency](http://docs.google.com/java/awt/Transparency.html)

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

### getRGBdefault

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

Returns a DirectColorModel that describes the default format for integer RGB values used in many of the methods in the AWT image interfaces for the convenience of the programmer. The color space is the default [ColorSpace](http://docs.google.com/java/awt/color/ColorSpace.html), sRGB. The format for the RGB values is an integer with 8 bits each of alpha, red, green, and blue color components ordered correspondingly from the most significant byte to the least significant byte, as in: 0xAARRGGBB. Color components are not premultiplied by the alpha component. This format does not necessarily represent the native or the most efficient ColorModel for a particular device or for all images. It is merely used as a common color model format.

**Returns:**a DirectColorModelobject describing default RGB values.

### hasAlpha

public final boolean **hasAlpha**()

Returns whether or not alpha is supported in this ColorModel.

**Returns:**true if alpha is supported in this ColorModel; false otherwise.

### isAlphaPremultiplied

public final boolean **isAlphaPremultiplied**()

Returns whether or not the alpha has been premultiplied in the pixel values to be translated by this ColorModel. If the boolean is true, this ColorModel is to be used to interpret pixel values in which color and alpha information are represented as separate spatial bands, and color samples are assumed to have been multiplied by the alpha sample.

**Returns:**true if the alpha values are premultiplied in the pixel values to be translated by this ColorModel; false otherwise.

### getTransferType

public final int **getTransferType**()

Returns the transfer type of this ColorModel. The transfer type is the type of primitive array used to represent pixel values as arrays.

**Returns:**the transfer type.**Since:** 1.3

### getPixelSize

public int **getPixelSize**()

Returns the number of bits per pixel described by this ColorModel.

**Returns:**the number of bits per pixel.

### getComponentSize

public int **getComponentSize**(int componentIdx)

Returns the number of bits for the specified color/alpha component. Color components are indexed in the order specified by the ColorSpace. Typically, this order reflects the name of the color space type. For example, for TYPE\_RGB, index 0 corresponds to red, index 1 to green, and index 2 to blue. If this ColorModel supports alpha, the alpha component corresponds to the index following the last color component.

**Parameters:**componentIdx - the index of the color/alpha component **Returns:**the number of bits for the color/alpha component at the specified index. **Throws:** [ArrayIndexOutOfBoundsException](http://docs.google.com/java/lang/ArrayIndexOutOfBoundsException.html) - if componentIdx is greater than the number of components or less than zero [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the number of bits array is null

### getComponentSize

public int[] **getComponentSize**()

Returns an array of the number of bits per color/alpha component. The array contains the color components in the order specified by the ColorSpace, followed by the alpha component, if present.

**Returns:**an array of the number of bits per color/alpha component

### 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 ColorModel.**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)

### getNumComponents

public int **getNumComponents**()

Returns the number of components, including alpha, in this ColorModel. This is equal to the number of color components, optionally plus one, if there is an alpha component.

**Returns:**the number of components in this ColorModel

### getNumColorComponents

public int **getNumColorComponents**()

Returns the number of color components in this ColorModel. This is the number of components returned by [ColorSpace.getNumComponents()](http://docs.google.com/java/awt/color/ColorSpace.html#getNumComponents()).

**Returns:**the number of color components in this ColorModel.**See Also:**[ColorSpace.getNumComponents()](http://docs.google.com/java/awt/color/ColorSpace.html#getNumComponents())

### getRed

public abstract int **getRed**(int pixel)

Returns the red color component for the specified pixel, scaled from 0 to 255 in the default RGB ColorSpace, sRGB. A color conversion is done if necessary. The pixel value is specified as an int. An IllegalArgumentException is thrown if pixel values for this ColorModel are not conveniently representable as a single int. The returned value is not a pre-multiplied value. For example, if the alpha is premultiplied, this method divides it out before returning the value. If the alpha value is 0, the red value is 0.

**Parameters:**pixel - a specified pixel **Returns:**the value of the red component of the specified pixel.

### getGreen

public abstract int **getGreen**(int pixel)

Returns the green color component for the specified pixel, scaled from 0 to 255 in the default RGB ColorSpace, sRGB. A color conversion is done if necessary. The pixel value is specified as an int. An IllegalArgumentException is thrown if pixel values for this ColorModel are not conveniently representable as a single int. The returned value is a non pre-multiplied value. For example, if the alpha is premultiplied, this method divides it out before returning the value. If the alpha value is 0, the green value is 0.

**Parameters:**pixel - the specified pixel **Returns:**the value of the green component of the specified pixel.

### getBlue

public abstract int **getBlue**(int pixel)

Returns the blue color component for the specified pixel, scaled from 0 to 255 in the default RGB ColorSpace, sRGB. A color conversion is done if necessary. The pixel value is specified as an int. An IllegalArgumentException is thrown if pixel values for this ColorModel are not conveniently representable as a single int. The returned value is a non pre-multiplied value, for example, if the alpha is premultiplied, this method divides it out before returning the value. If the alpha value is 0, the blue value is 0.

**Parameters:**pixel - the specified pixel **Returns:**the value of the blue component of the specified pixel.

### getAlpha

public abstract 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. An IllegalArgumentException is thrown if pixel values for this ColorModel are not conveniently representable as a single int.

**Parameters:**pixel - the specified pixel **Returns:**the value of alpha component of the specified pixel.

### getRGB

public int **getRGB**(int pixel)

Returns the color/alpha components of the pixel in the default RGB color model format. A color conversion is done if necessary. The pixel value is specified as an int. An IllegalArgumentException thrown if pixel values for this ColorModel are not conveniently representable as a single int. The returned value is in a non pre-multiplied format. For example, if the alpha is premultiplied, this method divides it out of the color components. If the alpha value is 0, the color values are 0.

**Parameters:**pixel - the specified pixel **Returns:**the RGB value of the color/alpha components of the specified pixel.**See Also:**[getRGBdefault()](http://docs.google.com/java/awt/image/ColorModel.html#getRGBdefault())

### getRed

public int **getRed**([Object](http://docs.google.com/java/lang/Object.html) inData)

Returns the red color component for the specified pixel, scaled from 0 to 255 in the default RGB ColorSpace, sRGB. A color conversion is done if necessary. The pixel value is specified by an array of data elements of type transferType passed in as an object reference. The returned value is a non pre-multiplied value. For example, if alpha is premultiplied, this method divides it out before returning the value. If the alpha value is 0, the red value is 0. If inData is not a primitive array of type transferType, a ClassCastException is thrown. An ArrayIndexOutOfBoundsException is thrown if inData is not large enough to hold a pixel value for this ColorModel. If this transferType is not supported, a UnsupportedOperationException will be thrown. Since ColorModel is an abstract class, any instance must be an instance of a subclass. Subclasses inherit the implementation of this method and if they don't override it, this method throws an exception if the subclass uses a transferType other than DataBuffer.TYPE\_BYTE, DataBuffer.TYPE\_USHORT, or DataBuffer.TYPE\_INT.

**Parameters:**inData - an array of pixel values **Returns:**the value of the red component of the specified pixel. **Throws:** [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if inData is not a primitive array of type transferType [ArrayIndexOutOfBoundsException](http://docs.google.com/java/lang/ArrayIndexOutOfBoundsException.html) - if inData is not large enough to hold a pixel value for this ColorModel [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if this tranferType is not supported by this ColorModel

### getGreen

public int **getGreen**([Object](http://docs.google.com/java/lang/Object.html) inData)

Returns the green color component for the specified pixel, scaled from 0 to 255 in the default RGB ColorSpace, sRGB. A color conversion is done if necessary. The pixel value is specified by an array of data elements of type transferType passed in as an object reference. The returned value will be a non pre-multiplied value. For example, if the alpha is premultiplied, this method divides it out before returning the value. If the alpha value is 0, the green value is 0. If inData is not a primitive array of type transferType, a ClassCastException is thrown. An ArrayIndexOutOfBoundsException is thrown if inData is not large enough to hold a pixel value for this ColorModel. If this transferType is not supported, a UnsupportedOperationException will be thrown. Since ColorModel is an abstract class, any instance must be an instance of a subclass. Subclasses inherit the implementation of this method and if they don't override it, this method throws an exception if the subclass uses a transferType other than DataBuffer.TYPE\_BYTE, DataBuffer.TYPE\_USHORT, or DataBuffer.TYPE\_INT.

**Parameters:**inData - an array of pixel values **Returns:**the value of the green component of the specified pixel. **Throws:** ClassCastException - if inData is not a primitive array of type transferType ArrayIndexOutOfBoundsException - if inData is not large enough to hold a pixel value for this ColorModel UnsupportedOperationException - if this tranferType is not supported by this ColorModel

### getBlue

public int **getBlue**([Object](http://docs.google.com/java/lang/Object.html) inData)

Returns the blue color component for the specified pixel, scaled from 0 to 255 in the default RGB ColorSpace, sRGB. A color conversion is done if necessary. The pixel value is specified by an array of data elements of type transferType passed in as an object reference. The returned value is a non pre-multiplied value. For example, if the alpha is premultiplied, this method divides it out before returning the value. If the alpha value is 0, the blue value will be 0. If inData is not a primitive array of type transferType, a ClassCastException is thrown. An ArrayIndexOutOfBoundsException is thrown if inData is not large enough to hold a pixel value for this ColorModel. If this transferType is not supported, a UnsupportedOperationException will be thrown. Since ColorModel is an abstract class, any instance must be an instance of a subclass. Subclasses inherit the implementation of this method and if they don't override it, this method throws an exception if the subclass uses a transferType other than DataBuffer.TYPE\_BYTE, DataBuffer.TYPE\_USHORT, or DataBuffer.TYPE\_INT.

**Parameters:**inData - an array of pixel values **Returns:**the value of the blue component of the specified pixel. **Throws:** [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if inData is not a primitive array of type transferType [ArrayIndexOutOfBoundsException](http://docs.google.com/java/lang/ArrayIndexOutOfBoundsException.html) - if inData is not large enough to hold a pixel value for this ColorModel [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if this tranferType is not supported by this ColorModel

### getAlpha

public int **getAlpha**([Object](http://docs.google.com/java/lang/Object.html) inData)

Returns the alpha component for the specified pixel, scaled from 0 to 255. The pixel value is specified by an array of data elements of type transferType passed in as an object reference. If inData is not a primitive array of type transferType, a ClassCastException is thrown. An ArrayIndexOutOfBoundsException is thrown if inData is not large enough to hold a pixel value for this ColorModel. If this transferType is not supported, a UnsupportedOperationException will be thrown. Since ColorModel is an abstract class, any instance must be an instance of a subclass. Subclasses inherit the implementation of this method and if they don't override it, this method throws an exception if the subclass uses a transferType other than DataBuffer.TYPE\_BYTE, DataBuffer.TYPE\_USHORT, or DataBuffer.TYPE\_INT.

**Parameters:**inData - the specified pixel **Returns:**the alpha component of the specified pixel, scaled from 0 to 255. **Throws:** [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if inData is not a primitive array of type transferType [ArrayIndexOutOfBoundsException](http://docs.google.com/java/lang/ArrayIndexOutOfBoundsException.html) - if inData is not large enough to hold a pixel value for this ColorModel [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if this tranferType is not supported by this ColorModel

### getRGB

public int **getRGB**([Object](http://docs.google.com/java/lang/Object.html) inData)

Returns the color/alpha components for the specified pixel in the default RGB color model format. A color conversion is done if necessary. The pixel value is specified by an array of data elements of type transferType passed in as an object reference. If inData is not a primitive array of type transferType, a ClassCastException is thrown. An ArrayIndexOutOfBoundsException is thrown if inData is not large enough to hold a pixel value for this ColorModel. The returned value will be in a non pre-multiplied format, i.e. if the alpha is premultiplied, this method will divide it out of the color components (if the alpha value is 0, the color values will be 0).

**Parameters:**inData - the specified pixel **Returns:**the color and alpha components of the specified pixel.**See Also:**[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 [WritableRaster.setDataElements(int, int, java.lang.Object)](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 will be 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. If this transferType is not supported, a UnsupportedOperationException will be thrown. Since ColorModel is an abstract class, any instance is an instance of a subclass. Subclasses must override this method since the implementation in this abstract class throws an UnsupportedOperationException.

**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 ColorModel. **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 this method is not supported by this ColorModel**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 given a pixel in this ColorModel. The pixel value is specified as an int. An IllegalArgumentException will be thrown if pixel values for this ColorModel are not conveniently representable as a single int or if color component values for this ColorModel are not conveniently representable in the unnormalized form. For example, this method can be used to retrieve the components for a specific pixel value in a DirectColorModel. If the components array is null, a new array will be allocated. The components array will be returned. 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). Since ColorModel is an abstract class, any instance is an instance of a subclass. Subclasses must override this method since the implementation in this abstract class throws an UnsupportedOperationException.

**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. **Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if this method is not supported by this ColorModel

### 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 given a 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 IllegalArgumentException will be thrown if color component values for this ColorModel are not conveniently representable in the unnormalized form. An ArrayIndexOutOfBoundsException is thrown if pixel is not large enough to hold a pixel value for this ColorModel. This method can be used to retrieve the components for a specific pixel value in any ColorModel. If the components array is null, a new array will be allocated. The components array will be returned. 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). Since ColorModel is an abstract class, any instance is an instance of a subclass. Subclasses must override this method since the implementation in this abstract class throws an UnsupportedOperationException.

**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:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if this method is not supported by this ColorModel

### getUnnormalizedComponents

public int[] **getUnnormalizedComponents**(float[] normComponents,  
 int normOffset,  
 int[] components,  
 int offset)

Returns an array of all of the color/alpha components in unnormalized form, given a normalized component array. Unnormalized components are unsigned integral values between 0 and 2n - 1, where n is the number of bits for a particular component. Normalized components are float values between a per component minimum and maximum specified by the ColorSpace object for this ColorModel. An IllegalArgumentException will be thrown if color component values for this ColorModel are not conveniently representable in the unnormalized form. If the components array is null, a new array will be allocated. The components array will be returned. 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). An IllegalArgumentException is thrown if the normComponents array is not large enough to hold all the color and alpha components starting at normOffset.

**Parameters:**normComponents - an array containing normalized componentsnormOffset - the offset into the normComponents array at which to start retrieving normalized componentscomponents - an array that receives the components from normComponentsoffset - the index into components at which to begin storing normalized components from normComponents **Returns:**an array containing unnormalized color and alpha components. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - If the component values for this ColorModel are not conveniently representable in the unnormalized form. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the length of normComponents minus normOffset is less than numComponents [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if the constructor of this ColorModel called the super(bits) constructor, but did not override this method. See the constructor, [ColorModel(int)](http://docs.google.com/java/awt/image/ColorModel.html#ColorModel(int)).

### getNormalizedComponents

public float[] **getNormalizedComponents**(int[] components,  
 int offset,  
 float[] normComponents,  
 int normOffset)

Returns an array of all of the color/alpha components in normalized form, given an unnormalized component array. Unnormalized components are unsigned integral values between 0 and 2n - 1, where n is the number of bits for a particular component. Normalized components are float values between a per component minimum and maximum specified by the ColorSpace object for this ColorModel. An IllegalArgumentException will be thrown if color component values for this ColorModel are not conveniently representable in the unnormalized form. If the normComponents array is null, a new array will be allocated. The normComponents array will be returned. Color/alpha components are stored in the normComponents array starting at normOffset (even if the array is allocated by this method). An ArrayIndexOutOfBoundsException is thrown if the normComponents array is not null and is not large enough to hold all the color and alpha components (starting at normOffset). An IllegalArgumentException is thrown if the components array is not large enough to hold all the color and alpha components starting at offset.

Since ColorModel is an abstract class, any instance is an instance of a subclass. The default implementation of this method in this abstract class assumes that component values for this class are conveniently representable in the unnormalized form. Therefore, subclasses which may have instances which do not support the unnormalized form must override this method.

**Parameters:**components - an array containing unnormalized componentsoffset - the offset into the components array at which to start retrieving unnormalized componentsnormComponents - an array that receives the normalized componentsnormOffset - the index into normComponents at which to begin storing normalized components **Returns:**an array containing normalized color and alpha components. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - If the component values for this ColorModel are not conveniently representable in the unnormalized form. [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if the constructor of this ColorModel called the super(bits) constructor, but did not override this method. See the constructor, [ColorModel(int)](http://docs.google.com/java/awt/image/ColorModel.html#ColorModel(int)). [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if this method is unable to determine the number of bits per component

### 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. This method will throw an IllegalArgumentException if component values for this ColorModel are not conveniently representable as a single int or if color component values for this ColorModel are not conveniently representable in the unnormalized form. An ArrayIndexOutOfBoundsException is thrown if the components array is not large enough to hold all the color and alpha components (starting at offset). Since ColorModel is an abstract class, any instance is an instance of a subclass. Subclasses must override this method since the implementation in this abstract class throws an UnsupportedOperationException.

**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:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if pixel values for this ColorModel are not conveniently representable as a single int [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if component values for this ColorModel are not conveniently representable in the unnormalized form [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 this method is not supported by this ColorModel

### 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) obj)

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. This method will throw an IllegalArgumentException if color component values for this ColorModel are not conveniently representable in the unnormalized form. An ArrayIndexOutOfBoundsException is thrown if the components array is not large enough to hold all the color and alpha components (starting at offset). If the obj variable is null, a new array will be allocated. If obj is not null, it must be a primitive array of type transferType; otherwise, a ClassCastException is thrown. An ArrayIndexOutOfBoundsException is thrown if obj is not large enough to hold a pixel value for this ColorModel. Since ColorModel is an abstract class, any instance is an instance of a subclass. Subclasses must override this method since the implementation in this abstract class throws an UnsupportedOperationException.

**Parameters:**components - an array of unnormalized color and alpha componentsoffset - the index into components at which to begin retrieving color and alpha componentsobj - 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 obj is not a primitive array of type transferType [ArrayIndexOutOfBoundsException](http://docs.google.com/java/lang/ArrayIndexOutOfBoundsException.html) - if obj 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 [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if component values for this ColorModel are not conveniently representable in the unnormalized form [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if this method is not supported by this ColorModel**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))

### getDataElement

public int **getDataElement**(float[] normComponents,  
 int normOffset)

Returns a pixel value represented as an int in this ColorModel, given an array of normalized color/alpha components. This method will throw an IllegalArgumentException if pixel values for this ColorModel are not conveniently representable as a single int. An ArrayIndexOutOfBoundsException is thrown if the normComponents array is not large enough to hold all the color and alpha components (starting at normOffset). Since ColorModel is an abstract class, any instance is an instance of a subclass. The default implementation of this method in this abstract class first converts from the normalized form to the unnormalized form and then calls getDataElement(int[], int). Subclasses which may have instances which do not support the unnormalized form must override this method.

**Parameters:**normComponents - an array of normalized color and alpha componentsnormOffset - the index into normComponents at which to begin retrieving the color and alpha components **Returns:**an int pixel value in this ColorModel corresponding to the specified components. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if pixel values for this ColorModel are not conveniently representable as a single int [ArrayIndexOutOfBoundsException](http://docs.google.com/java/lang/ArrayIndexOutOfBoundsException.html) - if the normComponents array is not large enough to hold all of the color and alpha components starting at normOffset**Since:** 1.4

### getDataElements

public [Object](http://docs.google.com/java/lang/Object.html) **getDataElements**(float[] normComponents,  
 int normOffset,  
 [Object](http://docs.google.com/java/lang/Object.html) obj)

Returns a data element array representation of a pixel in this ColorModel, given an array of normalized color/alpha components. This array can then be passed to the setDataElements method of a WritableRaster object. An ArrayIndexOutOfBoundsException is thrown if the normComponents array is not large enough to hold all the color and alpha components (starting at normOffset). If the obj variable is null, a new array will be allocated. If obj is not null, it must be a primitive array of type transferType; otherwise, a ClassCastException is thrown. An ArrayIndexOutOfBoundsException is thrown if obj is not large enough to hold a pixel value for this ColorModel. Since ColorModel is an abstract class, any instance is an instance of a subclass. The default implementation of this method in this abstract class first converts from the normalized form to the unnormalized form and then calls getDataElement(int[], int, Object). Subclasses which may have instances which do not support the unnormalized form must override this method.

**Parameters:**normComponents - an array of normalized color and alpha componentsnormOffset - the index into normComponents at which to begin retrieving color and alpha componentsobj - a primitive data array to hold the returned pixel **Returns:**an Object which is a primitive data array representation of a pixel **Throws:** [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if obj is not a primitive array of type transferType [ArrayIndexOutOfBoundsException](http://docs.google.com/java/lang/ArrayIndexOutOfBoundsException.html) - if obj is not large enough to hold a pixel value for this ColorModel or the normComponents array is not large enough to hold all of the color and alpha components starting at normOffset**Since:** 1.4 **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))

### getNormalizedComponents

public float[] **getNormalizedComponents**([Object](http://docs.google.com/java/lang/Object.html) pixel,  
 float[] normComponents,  
 int normOffset)

Returns an array of all of the color/alpha components in normalized form, given a 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. Normalized components are float values between a per component minimum and maximum specified by the ColorSpace object for this ColorModel. If the normComponents array is null, a new array will be allocated. The normComponents array will be returned. Color/alpha components are stored in the normComponents array starting at normOffset (even if the array is allocated by this method). An ArrayIndexOutOfBoundsException is thrown if the normComponents array is not null and is not large enough to hold all the color and alpha components (starting at normOffset). Since ColorModel is an abstract class, any instance is an instance of a subclass. The default implementation of this method in this abstract class first retrieves color and alpha components in the unnormalized form using getComponents(Object, int[], int) and then calls getNormalizedComponents(int[], int, float[], int). Subclasses which may have instances which do not support the unnormalized form must override this method.

**Parameters:**pixel - the specified pixelnormComponents - an array to receive the normalized componentsnormOffset - the offset into the normComponents array at which to start storing normalized components **Returns:**an array containing normalized 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 normComponents is not large enough to hold all color and alpha components starting at normOffset [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 the constructor of this ColorModel called the super(bits) constructor, but did not override this method. See the constructor, [ColorModel(int)](http://docs.google.com/java/awt/image/ColorModel.html#ColorModel(int)). [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if this method is unable to determine the number of bits per component**Since:** 1.4

### equals

public boolean **equals**([Object](http://docs.google.com/java/lang/Object.html) obj)

Tests if the specified Object is an instance of ColorModel and if it equals this ColorModel.

**Overrides:**[equals](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)) in class [Object](http://docs.google.com/java/lang/Object.html) **Parameters:**obj - the Object to test for equality **Returns:**true if the specified Object is an instance of ColorModel and equals this ColorModel; false otherwise.**See Also:**[Object.hashCode()](http://docs.google.com/java/lang/Object.html#hashCode()), [Hashtable](http://docs.google.com/java/util/Hashtable.html)

### hashCode

public int **hashCode**()

Returns the hash code for this ColorModel.

**Overrides:**[hashCode](http://docs.google.com/java/lang/Object.html#hashCode()) in class [Object](http://docs.google.com/java/lang/Object.html) **Returns:**a hash code for this ColorModel.**See Also:**[Object.equals(java.lang.Object)](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)), [Hashtable](http://docs.google.com/java/util/Hashtable.html)

### getColorSpace

public final [ColorSpace](http://docs.google.com/java/awt/color/ColorSpace.html) **getColorSpace**()

Returns the ColorSpace associated with this ColorModel.

**Returns:**the ColorSpace of this ColorModel.

### coerceData

public [ColorModel](http://docs.google.com/java/awt/image/ColorModel.html) **coerceData**([WritableRaster](http://docs.google.com/java/awt/image/WritableRaster.html) raster,  
 boolean isAlphaPremultiplied)

Forces the raster data to match the state specified in the isAlphaPremultiplied variable, assuming the data is currently correctly described by this ColorModel. It may multiply or divide the color raster data by alpha, or do nothing if the data is in the correct state. If the data needs to be coerced, this method will also return an instance of this ColorModel with the isAlphaPremultiplied flag set appropriately. This method will throw a UnsupportedOperationException if it is not supported by this ColorModel. Since ColorModel is an abstract class, any instance is an instance of a subclass. Subclasses must override this method since the implementation in this abstract class throws an UnsupportedOperationException.

**Parameters:**raster - the WritableRaster dataisAlphaPremultiplied - true if the alpha is premultiplied; false otherwise **Returns:**a ColorModel object that represents the coerced data.

### isCompatibleRaster

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

Returns true if raster is compatible with this ColorModel and false if it is not. Since ColorModel is an abstract class, any instance is an instance of a subclass. Subclasses must override this method since the implementation in this abstract class throws an UnsupportedOperationException.

**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. **Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if this method has not been implemented for this ColorModel

### 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. Since ColorModel is an abstract class, any instance is an instance of a subclass. Subclasses must override this method since the implementation in this abstract class throws an UnsupportedOperationException.

**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 this method is not supported by this ColorModel**See Also:**[WritableRaster](http://docs.google.com/java/awt/image/WritableRaster.html), [SampleModel](http://docs.google.com/java/awt/image/SampleModel.html)

### 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. Since ColorModel is an abstract class, any instance is an instance of a subclass. Subclasses must override this method since the implementation in this abstract class throws an UnsupportedOperationException.

**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:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if this method is not supported by this ColorModel**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 SampleModel is compatible with this ColorModel. Since ColorModel is an abstract class, any instance is an instance of a subclass. Subclasses must override this method since the implementation in this abstract class throws an UnsupportedOperationException.

**Parameters:**sm - the specified SampleModel **Returns:**true if the specified SampleModel is compatible with this ColorModel; false otherwise. **Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if this method is not supported by this ColorModel**See Also:**[SampleModel](http://docs.google.com/java/awt/image/SampleModel.html)

### 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/lang/Object.html#finalize()) in class [Object](http://docs.google.com/java/lang/Object.html)

### getAlphaRaster

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

Returns a Raster representing the alpha channel of an image, extracted from the input Raster, provided that pixel values of this ColorModel represent color and alpha information as separate spatial bands (e.g. [ComponentColorModel](http://docs.google.com/java/awt/image/ComponentColorModel.html) and DirectColorModel). This method assumes that Raster objects associated with such a ColorModel store the alpha band, if present, as the last band of image data. Returns null if there is no separate spatial alpha channel associated with this ColorModel. If this is an IndexColorModel which has alpha in the lookup table, this method will return null since there is no spatially discrete alpha channel. This method will create a new Raster (but will share the data array). Since ColorModel is an abstract class, any instance is an instance of a subclass. Subclasses must override this method to get any behavior other than returning null because the implementation in this abstract class returns null.

**Parameters:**raster - the specified Raster **Returns:**a Raster representing the alpha channel of an image, obtained from the specified Raster.

### 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/lang/Object.html#toString()) in class [Object](http://docs.google.com/java/lang/Object.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/ColorModel.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/ColorConvertOp.html)   [**NEXT CLASS**](http://docs.google.com/java/awt/image/ComponentColorModel.html) | [**FRAMES**](http://docs.google.com/index.html?java/awt/image/ColorModel.html)    [**NO FRAMES**](http://docs.google.com/ColorModel.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | [CONSTR](#tyjcwt) | [METHOD](#3dy6vkm) | DETAIL: [FIELD](#4d34og8) | [CONSTR](#3rdcrjn) | [METHOD](#35nkun2) |

[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).