| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/ComponentColorModel.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/ColorModel.html)   [**NEXT CLASS**](http://docs.google.com/java/awt/image/ComponentSampleModel.html) | [**FRAMES**](http://docs.google.com/index.html?java/awt/image/ComponentColorModel.html)    [**NO FRAMES**](http://docs.google.com/ComponentColorModel.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#2et92p0) | [CONSTR](#3dy6vkm) | [METHOD](#1t3h5sf) | DETAIL: FIELD | [CONSTR](#17dp8vu) | [METHOD](#lnxbz9) |

## **java.awt.image**

Class ComponentColorModel

[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.ComponentColorModel**

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

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

A ColorModel class that works with pixel values that represent color and alpha information as separate samples and that store each sample in a separate data element. This class can be used with an arbitrary ColorSpace. The number of color samples in the pixel values must be same as the number of color components in the ColorSpace. There may be a single alpha sample.

For those methods that use a primitive array pixel representation of type transferType, the array length is the same as the number of color and alpha samples. Color samples are stored first in the array followed by the alpha sample, if present. The order of the color samples is 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.

The translation from pixel sample values to color/alpha components for display or processing purposes is based on a one-to-one correspondence of samples to components. Depending on the transfer type used to create an instance of ComponentColorModel, the pixel sample values represented by that instance may be signed or unsigned and may be of integral type or float or double (see below for details). The translation from sample values to normalized color/alpha components must follow certain rules. For float and double samples, the translation is an identity, i.e. normalized component values are equal to the corresponding sample values. For integral samples, the translation should be only a simple scale and offset, where the scale and offset constants may be different for each component. The result of applying the scale and offset constants is a set of color/alpha component values, which are guaranteed to fall within a certain range. Typically, the range for a color component will be the range defined by the getMinValue and getMaxValue methods of the ColorSpace class. The range for an alpha component should be 0.0 to 1.0.

Instances of ComponentColorModel created with transfer types DataBuffer.TYPE\_BYTE, DataBuffer.TYPE\_USHORT, and DataBuffer.TYPE\_INT have pixel sample values which are treated as unsigned integral values. The number of bits in a color or alpha sample of a pixel value might not be the same as the number of bits for the corresponding color or alpha sample passed to the ComponentColorModel(ColorSpace, int[], boolean, boolean, int, int) constructor. In that case, this class assumes that the least significant n bits of a sample value hold the component value, where n is the number of significant bits for the component passed to the constructor. It also assumes that any higher-order bits in a sample value are zero. Thus, sample values range from 0 to 2n - 1. This class maps these sample values to normalized color component values such that 0 maps to the value obtained from the ColorSpace's getMinValue method for each component and 2n - 1 maps to the value obtained from getMaxValue. To create a ComponentColorModel with a different color sample mapping requires subclassing this class and overriding the getNormalizedComponents(Object, float[], int) method. The mapping for an alpha sample always maps 0 to 0.0 and 2n - 1 to 1.0.

For instances with unsigned sample values, the unnormalized color/alpha component representation is only supported if two conditions hold. First, sample value value 0 must map to normalized component value 0.0 and sample value 2n - 1 to 1.0. Second the min/max range of all color components of the ColorSpace must be 0.0 to 1.0. In this case, the component representation is the n least significant bits of the corresponding sample. Thus 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 these conditions are not met, any method taking an unnormalized component argument will throw an IllegalArgumentException.

Instances of ComponentColorModel created with transfer types DataBuffer.TYPE\_SHORT, DataBuffer.TYPE\_FLOAT, and DataBuffer.TYPE\_DOUBLE have pixel sample values which are treated as signed short, float, or double values. Such instances do not support the unnormalized color/alpha component representation, so any methods taking such a representation as an argument will throw an IllegalArgumentException when called on one of these instances. The normalized component values of instances of this class have a range which depends on the transfer type as follows: for float samples, the full range of the float data type; for double samples, the full range of the float data type (resulting from casting double to float); for short samples, from approximately -maxVal to +maxVal, where maxVal is the per component maximum value for the ColorSpace (-32767 maps to -maxVal, 0 maps to 0.0, and 32767 maps to +maxVal). A subclass may override the scaling for short sample values to normalized component values by overriding the getNormalizedComponents(Object, float[], int) method. For float and double samples, the normalized component values are taken to be equal to the corresponding sample values, and subclasses should not attempt to add any non-identity scaling for these transfer types.

Instances of ComponentColorModel created with transfer types DataBuffer.TYPE\_SHORT, DataBuffer.TYPE\_FLOAT, and DataBuffer.TYPE\_DOUBLE use all the bits of all sample values. Thus all color/alpha components have 16 bits when using DataBuffer.TYPE\_SHORT, 32 bits when using DataBuffer.TYPE\_FLOAT, and 64 bits when using DataBuffer.TYPE\_DOUBLE. When the ComponentColorModel(ColorSpace, int[], boolean, boolean, int, int) form of constructor is used with one of these transfer types, the bits array argument is ignored.

It is possible to have color/alpha sample values which cannot be reasonably interpreted as component values for rendering. This can happen when ComponentColorModel is subclassed to override the mapping of unsigned sample values to normalized color component values or when signed sample values outside a certain range are used. (As an example, specifying an alpha component as a signed short value outside the range 0 to 32767, normalized range 0.0 to 1.0, can lead to unexpected results.) It is the responsibility of applications to appropriately scale pixel data before rendering such that color components fall within the normalized range of the ColorSpace (obtained using the getMinValue and getMaxValue methods of the ColorSpace class) and the alpha component is between 0.0 and 1.0. If color or alpha component values fall outside these ranges, rendering results are indeterminate.

Methods that use a single int pixel representation throw an IllegalArgumentException, unless the number of components for the ComponentColorModel is one and the component value is unsigned -- in other words, a single color component using a transfer type of DataBuffer.TYPE\_BYTE, DataBuffer.TYPE\_USHORT, or DataBuffer.TYPE\_INT and no alpha.

A ComponentColorModel can be used in conjunction with a ComponentSampleModel, a BandedSampleModel, or a PixelInterleavedSampleModel to construct a BufferedImage.

**See Also:**[ColorModel](http://docs.google.com/java/awt/image/ColorModel.html), [ColorSpace](http://docs.google.com/java/awt/color/ColorSpace.html), [ComponentSampleModel](http://docs.google.com/java/awt/image/ComponentSampleModel.html), [BandedSampleModel](http://docs.google.com/java/awt/image/BandedSampleModel.html), [PixelInterleavedSampleModel](http://docs.google.com/java/awt/image/PixelInterleavedSampleModel.html), [BufferedImage](http://docs.google.com/java/awt/image/BufferedImage.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** | |
| --- | --- |
| [**ComponentColorModel**](http://docs.google.com/java/awt/image/ComponentColorModel.html#ComponentColorModel(java.awt.color.ColorSpace,%20boolean,%20boolean,%20int,%20int))([ColorSpace](http://docs.google.com/java/awt/color/ColorSpace.html) colorSpace, boolean hasAlpha, boolean isAlphaPremultiplied, int transparency, int transferType)            Constructs a ComponentColorModel from the specified parameters. |
| [**ComponentColorModel**](http://docs.google.com/java/awt/image/ComponentColorModel.html#ComponentColorModel(java.awt.color.ColorSpace,%20int%5B%5D,%20boolean,%20boolean,%20int,%20int))([ColorSpace](http://docs.google.com/java/awt/color/ColorSpace.html) colorSpace, int[] bits, boolean hasAlpha, boolean isAlphaPremultiplied, int transparency, int transferType)            Constructs a ComponentColorModel from the specified parameters. |

| **Method Summary** | |
| --- | --- |
| [ColorModel](http://docs.google.com/java/awt/image/ColorModel.html) | [**coerceData**](http://docs.google.com/java/awt/image/ComponentColorModel.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/ComponentColorModel.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/ComponentColorModel.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/ComponentColorModel.html#equals(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) obj)            Compares this color model with another for equality. |
| int | [**getAlpha**](http://docs.google.com/java/awt/image/ComponentColorModel.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/ComponentColorModel.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/ComponentColorModel.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. |
| int | [**getBlue**](http://docs.google.com/java/awt/image/ComponentColorModel.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/ComponentColorModel.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. |
| int[] | [**getComponents**](http://docs.google.com/java/awt/image/ComponentColorModel.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/ComponentColorModel.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 | [**getDataElement**](http://docs.google.com/java/awt/image/ComponentColorModel.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/ComponentColorModel.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/ComponentColorModel.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/ComponentColorModel.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/ComponentColorModel.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/ComponentColorModel.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/ComponentColorModel.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/ComponentColorModel.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/ComponentColorModel.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 | [**getRed**](http://docs.google.com/java/awt/image/ComponentColorModel.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/ComponentColorModel.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/ComponentColorModel.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/ComponentColorModel.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. |
| int[] | [**getUnnormalizedComponents**](http://docs.google.com/java/awt/image/ComponentColorModel.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 | [**isCompatibleRaster**](http://docs.google.com/java/awt/image/ComponentColorModel.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; false if it is not. |
| boolean | [**isCompatibleSampleModel**](http://docs.google.com/java/awt/image/ComponentColorModel.html#isCompatibleSampleModel(java.awt.image.SampleModel))([SampleModel](http://docs.google.com/java/awt/image/SampleModel.html) sm)            Checks whether or not the specified SampleModel is compatible with this ColorModel. |

| **Methods inherited from class java.awt.image.**[**ColorModel**](http://docs.google.com/java/awt/image/ColorModel.html) |
| --- |
| [finalize](http://docs.google.com/java/awt/image/ColorModel.html#finalize()), [getColorSpace](http://docs.google.com/java/awt/image/ColorModel.html#getColorSpace()), [getComponentSize](http://docs.google.com/java/awt/image/ColorModel.html#getComponentSize()), [getComponentSize](http://docs.google.com/java/awt/image/ColorModel.html#getComponentSize(int)), [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()), [getRGBdefault](http://docs.google.com/java/awt/image/ColorModel.html#getRGBdefault()), [getTransferType](http://docs.google.com/java/awt/image/ColorModel.html#getTransferType()), [getTransparency](http://docs.google.com/java/awt/image/ColorModel.html#getTransparency()), [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()), [toString](http://docs.google.com/java/awt/image/ColorModel.html#toString()) |

| **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** |
| --- |

### ComponentColorModel

public **ComponentColorModel**([ColorSpace](http://docs.google.com/java/awt/color/ColorSpace.html) colorSpace,  
 int[] bits,  
 boolean hasAlpha,  
 boolean isAlphaPremultiplied,  
 int transparency,  
 int transferType)

Constructs a ComponentColorModel from the specified parameters. Color components will be in the specified ColorSpace. The supported transfer types are DataBuffer.TYPE\_BYTE, DataBuffer.TYPE\_USHORT, DataBuffer.TYPE\_INT, DataBuffer.TYPE\_SHORT, DataBuffer.TYPE\_FLOAT, and DataBuffer.TYPE\_DOUBLE. If not null, the bits array specifies the number of significant bits per color and alpha component and its length should be at least 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. When the transferType is DataBuffer.TYPE\_SHORT, DataBuffer.TYPE\_FLOAT, or DataBuffer.TYPE\_DOUBLE the bits array argument is ignored. hasAlpha indicates whether alpha information is present. If hasAlpha is true, then the boolean isAlphaPremultiplied specifies how to interpret color and alpha samples in pixel values. 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 acceptable transparency values are OPAQUE, BITMASK or TRANSLUCENT. The transferType is the type of primitive array used to represent pixel values.

**Parameters:**colorSpace - The ColorSpace associated with this color model.bits - The number of significant bits per component. May be null, in which case all bits of all component samples will be significant. Ignored if transferType is one of DataBuffer.TYPE\_SHORT, DataBuffer.TYPE\_FLOAT, or DataBuffer.TYPE\_DOUBLE, in which case all bits of all component samples will be significant.hasAlpha - If true, this color model supports alpha.isAlphaPremultiplied - If true, alpha is premultiplied.transparency - Specifies what alpha values can be represented by this color model.transferType - Specifies the type of primitive array used to represent pixel values. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - If the bits array argument is not null, its length is less than the number of color and alpha components, and transferType is one of DataBuffer.TYPE\_BYTE, DataBuffer.TYPE\_USHORT, or DataBuffer.TYPE\_INT. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - If transferType is not one of DataBuffer.TYPE\_BYTE, DataBuffer.TYPE\_USHORT, DataBuffer.TYPE\_INT, DataBuffer.TYPE\_SHORT, DataBuffer.TYPE\_FLOAT, or DataBuffer.TYPE\_DOUBLE.**See Also:**[ColorSpace](http://docs.google.com/java/awt/color/ColorSpace.html), [Transparency](http://docs.google.com/java/awt/Transparency.html)

### ComponentColorModel

public **ComponentColorModel**([ColorSpace](http://docs.google.com/java/awt/color/ColorSpace.html) colorSpace,  
 boolean hasAlpha,  
 boolean isAlphaPremultiplied,  
 int transparency,  
 int transferType)

Constructs a ComponentColorModel from the specified parameters. Color components will be in the specified ColorSpace. The supported transfer types are DataBuffer.TYPE\_BYTE, DataBuffer.TYPE\_USHORT, DataBuffer.TYPE\_INT, DataBuffer.TYPE\_SHORT, DataBuffer.TYPE\_FLOAT, and DataBuffer.TYPE\_DOUBLE. The number of significant bits per color and alpha component will be 8, 16, 32, 16, 32, or 64, respectively. The number of color components will be the number of components in the ColorSpace. There will be an alpha component if hasAlpha is true. If hasAlpha is true, then the boolean isAlphaPremultiplied specifies how to interpret color and alpha samples in pixel values. 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 acceptable transparency values are OPAQUE, BITMASK or TRANSLUCENT. The transferType is the type of primitive array used to represent pixel values.

**Parameters:**colorSpace - The ColorSpace associated with this color model.hasAlpha - If true, this color model supports alpha.isAlphaPremultiplied - If true, alpha is premultiplied.transparency - Specifies what alpha values can be represented by this color model.transferType - Specifies the type of primitive array used to represent pixel values. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - If transferType is not one of DataBuffer.TYPE\_BYTE, DataBuffer.TYPE\_USHORT, DataBuffer.TYPE\_INT, DataBuffer.TYPE\_SHORT, DataBuffer.TYPE\_FLOAT, or DataBuffer.TYPE\_DOUBLE.**Since:** 1.4 **See Also:**[ColorSpace](http://docs.google.com/java/awt/color/ColorSpace.html), [Transparency](http://docs.google.com/java/awt/Transparency.html)

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

### getRed

public 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. The returned value will be a non pre-multiplied value. If the alpha is premultiplied, this method divides it out before returning the value (if the alpha value is 0, the red value will be 0).

**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 pixel from which you want to get the red color component. **Returns:**The red color component for the specified pixel, as an int. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - If there is more than one component in this ColorModel. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - If the component value for this ColorModel is signed

### getGreen

public 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. The returned value will be a non pre-multiplied value. If the alpha is premultiplied, this method divides it out before returning the value (if the alpha value is 0, the green value will be 0).

**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 pixel from which you want to get the green color component. **Returns:**The green color component for the specified pixel, as an int. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - If there is more than one component in this ColorModel. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - If the component value for this ColorModel is signed

### getBlue

public 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. The returned value will be a non pre-multiplied value. 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).

**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 pixel from which you want to get the blue color component. **Returns:**The blue color component for the specified pixel, as an int. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - If there is more than one component in this ColorModel. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - If the component value for this ColorModel is signed

### getAlpha

public 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.

**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 pixel from which you want to get the alpha component. **Returns:**The alpha component for the specified pixel, as an int. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - If there is more than one component in this ColorModel. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - If the component value for this ColorModel is signed

### 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 returned value will be in a non pre-multiplied format. If the alpha is premultiplied, this method divides it out of the color components (if the alpha value is 0, the color values will be 0).

**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 pixel from which you want to get the color/alpha components. **Returns:**The color/alpha components for the specified pixel, as an int. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - If there is more than one component in this ColorModel. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - If the component value for this ColorModel is signed**See Also:**[ColorModel.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 will be a non pre-multiplied value. If the alpha is premultiplied, this method divides it out before returning the value (if the alpha value is 0, the red value will be 0). Since ComponentColorModel 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:**[getRed](http://docs.google.com/java/awt/image/ColorModel.html#getRed(java.lang.Object)) in class [ColorModel](http://docs.google.com/java/awt/image/ColorModel.html) **Parameters:**inData - The pixel from which you want to get the red color component, specified by an array of data elements of type transferType. **Returns:**The red color component for the specified pixel, as an int. **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 the transfer type of this ComponentColorModel is not one of the supported transfer types: DataBuffer.TYPE\_BYTE, DataBuffer.TYPE\_USHORT, DataBuffer.TYPE\_INT, DataBuffer.TYPE\_SHORT, DataBuffer.TYPE\_FLOAT, or DataBuffer.TYPE\_DOUBLE.

### 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 is a non pre-multiplied value. If the alpha is premultiplied, this method divides it out before returning the value (if the alpha value is 0, the green value will be 0). Since ComponentColorModel 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:**[getGreen](http://docs.google.com/java/awt/image/ColorModel.html#getGreen(java.lang.Object)) in class [ColorModel](http://docs.google.com/java/awt/image/ColorModel.html) **Parameters:**inData - The pixel from which you want to get the green color component, specified by an array of data elements of type transferType. **Returns:**The green color component for the specified pixel, as an int. **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 the transfer type of this ComponentColorModel is not one of the supported transfer types: DataBuffer.TYPE\_BYTE, DataBuffer.TYPE\_USHORT, DataBuffer.TYPE\_INT, DataBuffer.TYPE\_SHORT, DataBuffer.TYPE\_FLOAT, or DataBuffer.TYPE\_DOUBLE.

### 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. 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). Since ComponentColorModel 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:**[getBlue](http://docs.google.com/java/awt/image/ColorModel.html#getBlue(java.lang.Object)) in class [ColorModel](http://docs.google.com/java/awt/image/ColorModel.html) **Parameters:**inData - The pixel from which you want to get the blue color component, specified by an array of data elements of type transferType. **Returns:**The blue color component for the specified pixel, as an int. **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 the transfer type of this ComponentColorModel is not one of the supported transfer types: DataBuffer.TYPE\_BYTE, DataBuffer.TYPE\_USHORT, DataBuffer.TYPE\_INT, DataBuffer.TYPE\_SHORT, DataBuffer.TYPE\_FLOAT, or DataBuffer.TYPE\_DOUBLE.

### 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. Since ComponentColorModel 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:**[getAlpha](http://docs.google.com/java/awt/image/ColorModel.html#getAlpha(java.lang.Object)) in class [ColorModel](http://docs.google.com/java/awt/image/ColorModel.html) **Parameters:**inData - The pixel from which you want to get the alpha component, specified by an array of data elements of type transferType. **Returns:**The alpha component for the specified pixel, as an int. **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 the transfer type of this ComponentColorModel is not one of the supported transfer types: DataBuffer.TYPE\_BYTE, DataBuffer.TYPE\_USHORT, DataBuffer.TYPE\_INT, DataBuffer.TYPE\_SHORT, DataBuffer.TYPE\_FLOAT, or DataBuffer.TYPE\_DOUBLE.

### 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. The returned value is in a non pre-multiplied format. If the alpha is premultiplied, this method divides it out of the color components (if the alpha value is 0, the color values will be 0). Since ComponentColorModel 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:**[getRGB](http://docs.google.com/java/awt/image/ColorModel.html#getRGB(java.lang.Object)) in class [ColorModel](http://docs.google.com/java/awt/image/ColorModel.html) **Parameters:**inData - The pixel from which you want to get the color/alpha components, specified by an array of data elements of type transferType. **Returns:**The color/alpha components for the specified pixel, as an int. **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 the transfer type of this ComponentColorModel is not one of the supported transfer types: DataBuffer.TYPE\_BYTE, DataBuffer.TYPE\_USHORT, DataBuffer.TYPE\_INT, DataBuffer.TYPE\_SHORT, DataBuffer.TYPE\_FLOAT, or DataBuffer.TYPE\_DOUBLE.**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 method of a WritableRaster object. If the pixel parameter is null, a new array is allocated. Since ComponentColorModel 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 representation of the pixel in the RGB color modelpixel - the specified pixel **Returns:**The data element array representation of a pixel in this ColorModel. **Throws:** [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - If pixel is not null and 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 the transfer type of this ComponentColorModel is not one of the supported transfer types: DataBuffer.TYPE\_BYTE, DataBuffer.TYPE\_USHORT, DataBuffer.TYPE\_INT, DataBuffer.TYPE\_SHORT, DataBuffer.TYPE\_FLOAT, or DataBuffer.TYPE\_DOUBLE.**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. An IllegalArgumentException is thrown if the component value for this ColorModel is not conveniently representable in the unnormalized form. Color/alpha components are stored in the components array starting at offset (even if the array is allocated by this method).

**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 pixel value specified as an integer.components - An integer array in which to store the unnormalized color/alpha components. If the components array is null, a new array is allocated.offset - An offset into the components array. **Returns:**The components array. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - If there is more than one component in this ColorModel. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - If this ColorModel does not support the unnormalized form [ArrayIndexOutOfBoundsException](http://docs.google.com/java/lang/ArrayIndexOutOfBoundsException.html) - If the components array is not null and is not large enough to hold all the color and alpha components (starting at offset).

### 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. An IllegalArgumentException is thrown if the component values for this ColorModel are not conveniently representable in the unnormalized form. Color/alpha components are stored in the components array starting at offset (even if the array is allocated by this method). Since ComponentColorModel can be subclassed, subclasses inherit the implementation of this method and if they don't override it then this method might 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 - A pixel value specified by an array of data elements of type transferType.components - An integer array in which to store the unnormalized color/alpha components. If the components array is null, a new array is allocated.offset - An offset into the components array. **Returns:**The components array. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - If this ComponentColorModel does not support the unnormalized form [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - in some cases iff the transfer type of this ComponentColorModel is not one of the following transfer types: DataBuffer.TYPE\_BYTE, DataBuffer.TYPE\_USHORT, or DataBuffer.TYPE\_INT. [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - If pixel is not a primitive array of type transferType. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - If the components array is not null and is not large enough to hold all the color and alpha components (starting at offset), or if pixel is not large enough to hold a pixel value for 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.

**Overrides:**[getUnnormalizedComponents](http://docs.google.com/java/awt/image/ColorModel.html#getUnnormalizedComponents(float%5B%5D,%20int,%20int%5B%5D,%20int)) in class [ColorModel](http://docs.google.com/java/awt/image/ColorModel.html) **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 this ComponentColorModel does not support the unnormalized form [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the length of normComponents minus normOffset is less than numComponents

### 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.

**Overrides:**[getNormalizedComponents](http://docs.google.com/java/awt/image/ColorModel.html#getNormalizedComponents(int%5B%5D,%20int,%20float%5B%5D,%20int)) in class [ColorModel](http://docs.google.com/java/awt/image/ColorModel.html) **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 this ComponentColorModel does not support the unnormalized form

### 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.

**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/alpha components.offset - An offset into the components array. **Returns:**A pixel value represented as an int. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - If there is more than one component in this ColorModel. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - If this ComponentColorModel does not support the unnormalized form

### 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.

**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/alpha components.offset - The integer offset into the components array.obj - The object in which to store the data element array representation of the pixel. If obj variable is null, a new array is allocated. If obj is not null, it must be a primitive array of type transferType. An ArrayIndexOutOfBoundsException is thrown if obj is not large enough to hold a pixel value for this ColorModel. Since ComponentColorModel 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. **Returns:**The data element array representation of a pixel in this ColorModel. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - If the components array 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 obj is not null and 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. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - If this ComponentColorModel does not support the unnormalized form [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - If the transfer type of this ComponentColorModel is not one of the following transfer types: DataBuffer.TYPE\_BYTE, DataBuffer.TYPE\_USHORT, or DataBuffer.TYPE\_INT.**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).

**Overrides:**[getDataElement](http://docs.google.com/java/awt/image/ColorModel.html#getDataElement(float%5B%5D,%20int)) in class [ColorModel](http://docs.google.com/java/awt/image/ColorModel.html) **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.

**Overrides:**[getDataElements](http://docs.google.com/java/awt/image/ColorModel.html#getDataElements(float%5B%5D,%20int,%20java.lang.Object)) in class [ColorModel](http://docs.google.com/java/awt/image/ColorModel.html) **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).

This method must be overrridden by a subclass if that subclass is designed to translate pixel sample values to color component values in a non-default way. The default translations implemented by this class is described in the class comments. Any subclass implementing a non-default translation must follow the constraints on allowable translations defined there.

**Overrides:**[getNormalizedComponents](http://docs.google.com/java/awt/image/ColorModel.html#getNormalizedComponents(java.lang.Object,%20float%5B%5D,%20int)) in class [ColorModel](http://docs.google.com/java/awt/image/ColorModel.html) **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.**Since:** 1.4

### 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 also returns an instance of this ColorModel with the isAlphaPremultiplied flag set appropriately. 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:**[coerceData](http://docs.google.com/java/awt/image/ColorModel.html#coerceData(java.awt.image.WritableRaster,%20boolean)) in class [ColorModel](http://docs.google.com/java/awt/image/ColorModel.html) **Parameters:**raster - the WritableRaster dataisAlphaPremultiplied - true if the alpha is premultiplied; false otherwise **Returns:**a ColorModel object that represents the coerced data. **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if raster is null and data coercion is required. [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if the transfer type of this ComponentColorModel is not one of the supported transfer types: DataBuffer.TYPE\_BYTE, DataBuffer.TYPE\_USHORT, DataBuffer.TYPE\_INT, DataBuffer.TYPE\_SHORT, DataBuffer.TYPE\_FLOAT, or DataBuffer.TYPE\_DOUBLE.

### isCompatibleRaster

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

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

**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 object to test for compatibility. **Returns:**true if raster is compatible with this ColorModel, false if it is not.

### 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.

**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 of the WritableRaster you want to create.h - The height of the WritableRaster you want to create. **Returns:**A WritableRaster that is compatible with 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.

**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 of the SampleModel you want to create.h - The height of the SampleModel you want to create. **Returns:**A SampleModel that is compatible with 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 whether or not the specified SampleModel is compatible with this ColorModel.

**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 SampleModel to test for compatibility. **Returns:**true if the SampleModel is compatible with this ColorModel, false if it is not.**See Also:**[SampleModel](http://docs.google.com/java/awt/image/SampleModel.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. This method assumes that Raster objects associated with this 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. This method creates a new Raster, but will share the data array.

**Overrides:**[getAlphaRaster](http://docs.google.com/java/awt/image/ColorModel.html#getAlphaRaster(java.awt.image.WritableRaster)) in class [ColorModel](http://docs.google.com/java/awt/image/ColorModel.html) **Parameters:**raster - The WritableRaster from which to extract the alpha channel. **Returns:**A WritableRaster containing the image's alpha channel.

### equals

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

Compares this color model with another for equality.

**Overrides:**[equals](http://docs.google.com/java/awt/image/ColorModel.html#equals(java.lang.Object)) in class [ColorModel](http://docs.google.com/java/awt/image/ColorModel.html) **Parameters:**obj - The object to compare with this color model. **Returns:**true if the color model objects are equal, false if they are not.**See Also:**[Object.hashCode()](http://docs.google.com/java/lang/Object.html#hashCode()), [Hashtable](http://docs.google.com/java/util/Hashtable.html)

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/ComponentColorModel.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/ColorModel.html)   [**NEXT CLASS**](http://docs.google.com/java/awt/image/ComponentSampleModel.html) | [**FRAMES**](http://docs.google.com/index.html?java/awt/image/ComponentColorModel.html)    [**NO FRAMES**](http://docs.google.com/ComponentColorModel.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#2et92p0) | [CONSTR](#3dy6vkm) | [METHOD](#1t3h5sf) | DETAIL: FIELD | [CONSTR](#17dp8vu) | [METHOD](#lnxbz9) |

[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).