| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/RenderingHints.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/Rectangle.html)   [**NEXT CLASS**](http://docs.google.com/java/awt/RenderingHints.Key.html) | [**FRAMES**](http://docs.google.com/index.html?java/awt/RenderingHints.html)    [**NO FRAMES**](http://docs.google.com/RenderingHints.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#3znysh7) | [FIELD](#tyjcwt) | [CONSTR](#3dy6vkm) | [METHOD](#1t3h5sf) | DETAIL: [FIELD](#2s8eyo1) | [CONSTR](#111kx3o) | [METHOD](#4k668n3) |

## **java.awt**

Class RenderingHints

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

**All Implemented Interfaces:** [Cloneable](http://docs.google.com/java/lang/Cloneable.html), [Map](http://docs.google.com/java/util/Map.html)<[Object](http://docs.google.com/java/lang/Object.html),[Object](http://docs.google.com/java/lang/Object.html)>

public class **RenderingHints**extends [Object](http://docs.google.com/java/lang/Object.html)implements [Map](http://docs.google.com/java/util/Map.html)<[Object](http://docs.google.com/java/lang/Object.html),[Object](http://docs.google.com/java/lang/Object.html)>, [Cloneable](http://docs.google.com/java/lang/Cloneable.html)

The RenderingHints class defines and manages collections of keys and associated values which allow an application to provide input into the choice of algorithms used by other classes which perform rendering and image manipulation services. The [Graphics2D](http://docs.google.com/java/awt/Graphics2D.html) class, and classes that implement [BufferedImageOp](http://docs.google.com/java/awt/image/BufferedImageOp.html) and [RasterOp](http://docs.google.com/java/awt/image/RasterOp.html) all provide methods to get and possibly to set individual or groups of RenderingHints keys and their associated values. When those implementations perform any rendering or image manipulation operations they should examine the values of any RenderingHints that were requested by the caller and tailor the algorithms used accordingly and to the best of their ability.

Note that since these keys and values are *hints*, there is no requirement that a given implementation supports all possible choices indicated below or that it can respond to requests to modify its choice of algorithm. The values of the various hint keys may also interact such that while all variants of a given key are supported in one situation, the implementation may be more restricted when the values associated with other keys are modified. For example, some implementations may be able to provide several types of dithering when the antialiasing hint is turned off, but have little control over dithering when antialiasing is on. The full set of supported keys and hints may also vary by destination since runtimes may use different underlying modules to render to the screen, or to [BufferedImage](http://docs.google.com/java/awt/image/BufferedImage.html) objects, or while printing.

Implementations are free to ignore the hints completely, but should try to use an implementation algorithm that is as close as possible to the request. If an implementation supports a given algorithm when any value is used for an associated hint key, then minimally it must do so when the value for that key is the exact value that specifies the algorithm.

The keys used to control the hints are all special values that subclass the associated [RenderingHints.Key](http://docs.google.com/java/awt/RenderingHints.Key.html) class. Many common hints are expressed below as static constants in this class, but the list is not meant to be exhaustive. Other hints may be created by other packages by defining new objects which subclass the Key class and defining the associated values.

| **Nested Class Summary** | |
| --- | --- |
| static class | [**RenderingHints.Key**](http://docs.google.com/java/awt/RenderingHints.Key.html)            Defines the base type of all keys used along with the [RenderingHints](http://docs.google.com/java/awt/RenderingHints.html) class to control various algorithm choices in the rendering and imaging pipelines. |

| **Nested classes/interfaces inherited from interface java.util.**[**Map**](http://docs.google.com/java/util/Map.html) |
| --- |
| [Map.Entry](http://docs.google.com/java/util/Map.Entry.html)<[K](http://docs.google.com/java/util/Map.Entry.html),[V](http://docs.google.com/java/util/Map.Entry.html)> |

| **Field Summary** | |
| --- | --- |
| static [RenderingHints.Key](http://docs.google.com/java/awt/RenderingHints.Key.html) | [**KEY\_ALPHA\_INTERPOLATION**](http://docs.google.com/java/awt/RenderingHints.html#KEY_ALPHA_INTERPOLATION)            Alpha interpolation hint key. |
| static [RenderingHints.Key](http://docs.google.com/java/awt/RenderingHints.Key.html) | [**KEY\_ANTIALIASING**](http://docs.google.com/java/awt/RenderingHints.html#KEY_ANTIALIASING)            Antialiasing hint key. |
| static [RenderingHints.Key](http://docs.google.com/java/awt/RenderingHints.Key.html) | [**KEY\_COLOR\_RENDERING**](http://docs.google.com/java/awt/RenderingHints.html#KEY_COLOR_RENDERING)            Color rendering hint key. |
| static [RenderingHints.Key](http://docs.google.com/java/awt/RenderingHints.Key.html) | [**KEY\_DITHERING**](http://docs.google.com/java/awt/RenderingHints.html#KEY_DITHERING)            Dithering hint key. |
| static [RenderingHints.Key](http://docs.google.com/java/awt/RenderingHints.Key.html) | [**KEY\_FRACTIONALMETRICS**](http://docs.google.com/java/awt/RenderingHints.html#KEY_FRACTIONALMETRICS)            Font fractional metrics hint key. |
| static [RenderingHints.Key](http://docs.google.com/java/awt/RenderingHints.Key.html) | [**KEY\_INTERPOLATION**](http://docs.google.com/java/awt/RenderingHints.html#KEY_INTERPOLATION)            Interpolation hint key. |
| static [RenderingHints.Key](http://docs.google.com/java/awt/RenderingHints.Key.html) | [**KEY\_RENDERING**](http://docs.google.com/java/awt/RenderingHints.html#KEY_RENDERING)            Rendering hint key. |
| static [RenderingHints.Key](http://docs.google.com/java/awt/RenderingHints.Key.html) | [**KEY\_STROKE\_CONTROL**](http://docs.google.com/java/awt/RenderingHints.html#KEY_STROKE_CONTROL)            Stroke normalization control hint key. |
| static [RenderingHints.Key](http://docs.google.com/java/awt/RenderingHints.Key.html) | [**KEY\_TEXT\_ANTIALIASING**](http://docs.google.com/java/awt/RenderingHints.html#KEY_TEXT_ANTIALIASING)            Text antialiasing hint key. |
| static [RenderingHints.Key](http://docs.google.com/java/awt/RenderingHints.Key.html) | [**KEY\_TEXT\_LCD\_CONTRAST**](http://docs.google.com/java/awt/RenderingHints.html#KEY_TEXT_LCD_CONTRAST)            LCD text contrast rendering hint key. |
| static [Object](http://docs.google.com/java/lang/Object.html) | [**VALUE\_ALPHA\_INTERPOLATION\_DEFAULT**](http://docs.google.com/java/awt/RenderingHints.html#VALUE_ALPHA_INTERPOLATION_DEFAULT)            Alpha interpolation hint value -- alpha blending algorithms are chosen by the implementation for a good tradeoff of performance vs. |
| static [Object](http://docs.google.com/java/lang/Object.html) | [**VALUE\_ALPHA\_INTERPOLATION\_QUALITY**](http://docs.google.com/java/awt/RenderingHints.html#VALUE_ALPHA_INTERPOLATION_QUALITY)            Alpha interpolation hint value -- alpha blending algorithms are chosen with a preference for precision and visual quality. |
| static [Object](http://docs.google.com/java/lang/Object.html) | [**VALUE\_ALPHA\_INTERPOLATION\_SPEED**](http://docs.google.com/java/awt/RenderingHints.html#VALUE_ALPHA_INTERPOLATION_SPEED)            Alpha interpolation hint value -- alpha blending algorithms are chosen with a preference for calculation speed. |
| static [Object](http://docs.google.com/java/lang/Object.html) | [**VALUE\_ANTIALIAS\_DEFAULT**](http://docs.google.com/java/awt/RenderingHints.html#VALUE_ANTIALIAS_DEFAULT)            Antialiasing hint value -- rendering is done with a default antialiasing mode chosen by the implementation. |
| static [Object](http://docs.google.com/java/lang/Object.html) | [**VALUE\_ANTIALIAS\_OFF**](http://docs.google.com/java/awt/RenderingHints.html#VALUE_ANTIALIAS_OFF)            Antialiasing hint value -- rendering is done without antialiasing. |
| static [Object](http://docs.google.com/java/lang/Object.html) | [**VALUE\_ANTIALIAS\_ON**](http://docs.google.com/java/awt/RenderingHints.html#VALUE_ANTIALIAS_ON)            Antialiasing hint value -- rendering is done with antialiasing. |
| static [Object](http://docs.google.com/java/lang/Object.html) | [**VALUE\_COLOR\_RENDER\_DEFAULT**](http://docs.google.com/java/awt/RenderingHints.html#VALUE_COLOR_RENDER_DEFAULT)            Color rendering hint value -- perform color conversion calculations as chosen by the implementation to represent the best available tradeoff between performance and accuracy. |
| static [Object](http://docs.google.com/java/lang/Object.html) | [**VALUE\_COLOR\_RENDER\_QUALITY**](http://docs.google.com/java/awt/RenderingHints.html#VALUE_COLOR_RENDER_QUALITY)            Color rendering hint value -- perform the color conversion calculations with the highest accuracy and visual quality. |
| static [Object](http://docs.google.com/java/lang/Object.html) | [**VALUE\_COLOR\_RENDER\_SPEED**](http://docs.google.com/java/awt/RenderingHints.html#VALUE_COLOR_RENDER_SPEED)            Color rendering hint value -- perform the fastest color conversion to the format of the output device. |
| static [Object](http://docs.google.com/java/lang/Object.html) | [**VALUE\_DITHER\_DEFAULT**](http://docs.google.com/java/awt/RenderingHints.html#VALUE_DITHER_DEFAULT)            Dithering hint value -- use a default for dithering chosen by the implementation. |
| static [Object](http://docs.google.com/java/lang/Object.html) | [**VALUE\_DITHER\_DISABLE**](http://docs.google.com/java/awt/RenderingHints.html#VALUE_DITHER_DISABLE)            Dithering hint value -- do not dither when rendering geometry. |
| static [Object](http://docs.google.com/java/lang/Object.html) | [**VALUE\_DITHER\_ENABLE**](http://docs.google.com/java/awt/RenderingHints.html#VALUE_DITHER_ENABLE)            Dithering hint value -- dither when rendering geometry, if needed. |
| static [Object](http://docs.google.com/java/lang/Object.html) | [**VALUE\_FRACTIONALMETRICS\_DEFAULT**](http://docs.google.com/java/awt/RenderingHints.html#VALUE_FRACTIONALMETRICS_DEFAULT)            Font fractional metrics hint value -- character glyphs are positioned with accuracy chosen by the implementation. |
| static [Object](http://docs.google.com/java/lang/Object.html) | [**VALUE\_FRACTIONALMETRICS\_OFF**](http://docs.google.com/java/awt/RenderingHints.html#VALUE_FRACTIONALMETRICS_OFF)            Font fractional metrics hint value -- character glyphs are positioned with advance widths rounded to pixel boundaries. |
| static [Object](http://docs.google.com/java/lang/Object.html) | [**VALUE\_FRACTIONALMETRICS\_ON**](http://docs.google.com/java/awt/RenderingHints.html#VALUE_FRACTIONALMETRICS_ON)            Font fractional metrics hint value -- character glyphs are positioned with sub-pixel accuracy. |
| static [Object](http://docs.google.com/java/lang/Object.html) | [**VALUE\_INTERPOLATION\_BICUBIC**](http://docs.google.com/java/awt/RenderingHints.html#VALUE_INTERPOLATION_BICUBIC)            Interpolation hint value -- the color samples of 9 nearby integer coordinate samples in the image are interpolated using a cubic function in both X and Y to produce a color sample. |
| static [Object](http://docs.google.com/java/lang/Object.html) | [**VALUE\_INTERPOLATION\_BILINEAR**](http://docs.google.com/java/awt/RenderingHints.html#VALUE_INTERPOLATION_BILINEAR)            Interpolation hint value -- the color samples of the 4 nearest neighboring integer coordinate samples in the image are interpolated linearly to produce a color sample. |
| static [Object](http://docs.google.com/java/lang/Object.html) | [**VALUE\_INTERPOLATION\_NEAREST\_NEIGHBOR**](http://docs.google.com/java/awt/RenderingHints.html#VALUE_INTERPOLATION_NEAREST_NEIGHBOR)            Interpolation hint value -- the color sample of the nearest neighboring integer coordinate sample in the image is used. |
| static [Object](http://docs.google.com/java/lang/Object.html) | [**VALUE\_RENDER\_DEFAULT**](http://docs.google.com/java/awt/RenderingHints.html#VALUE_RENDER_DEFAULT)            Rendering hint value -- rendering algorithms are chosen by the implementation for a good tradeoff of performance vs. |
| static [Object](http://docs.google.com/java/lang/Object.html) | [**VALUE\_RENDER\_QUALITY**](http://docs.google.com/java/awt/RenderingHints.html#VALUE_RENDER_QUALITY)            Rendering hint value -- rendering algorithms are chosen with a preference for output quality. |
| static [Object](http://docs.google.com/java/lang/Object.html) | [**VALUE\_RENDER\_SPEED**](http://docs.google.com/java/awt/RenderingHints.html#VALUE_RENDER_SPEED)            Rendering hint value -- rendering algorithms are chosen with a preference for output speed. |
| static [Object](http://docs.google.com/java/lang/Object.html) | [**VALUE\_STROKE\_DEFAULT**](http://docs.google.com/java/awt/RenderingHints.html#VALUE_STROKE_DEFAULT)            Stroke normalization control hint value -- geometry may be modified or left pure depending on the tradeoffs in a given implementation. |
| static [Object](http://docs.google.com/java/lang/Object.html) | [**VALUE\_STROKE\_NORMALIZE**](http://docs.google.com/java/awt/RenderingHints.html#VALUE_STROKE_NORMALIZE)            Stroke normalization control hint value -- geometry should be normalized to improve uniformity or spacing of lines and overall aesthetics. |
| static [Object](http://docs.google.com/java/lang/Object.html) | [**VALUE\_STROKE\_PURE**](http://docs.google.com/java/awt/RenderingHints.html#VALUE_STROKE_PURE)            Stroke normalization control hint value -- geometry should be left unmodified and rendered with sub-pixel accuracy. |
| static [Object](http://docs.google.com/java/lang/Object.html) | [**VALUE\_TEXT\_ANTIALIAS\_DEFAULT**](http://docs.google.com/java/awt/RenderingHints.html#VALUE_TEXT_ANTIALIAS_DEFAULT)            Text antialiasing hint value -- text rendering is done according to the [KEY\_ANTIALIASING](http://docs.google.com/java/awt/RenderingHints.html#KEY_ANTIALIASING) hint or a default chosen by the implementation. |
| static [Object](http://docs.google.com/java/lang/Object.html) | [**VALUE\_TEXT\_ANTIALIAS\_GASP**](http://docs.google.com/java/awt/RenderingHints.html#VALUE_TEXT_ANTIALIAS_GASP)            Text antialiasing hint value -- text rendering is requested to use information in the font resource which specifies for each point size whether to apply [VALUE\_TEXT\_ANTIALIAS\_ON](http://docs.google.com/java/awt/RenderingHints.html#VALUE_TEXT_ANTIALIAS_ON) or [VALUE\_TEXT\_ANTIALIAS\_OFF](http://docs.google.com/java/awt/RenderingHints.html#VALUE_TEXT_ANTIALIAS_OFF). |
| static [Object](http://docs.google.com/java/lang/Object.html) | [**VALUE\_TEXT\_ANTIALIAS\_LCD\_HBGR**](http://docs.google.com/java/awt/RenderingHints.html#VALUE_TEXT_ANTIALIAS_LCD_HBGR)            Text antialiasing hint value -- request that text be displayed optimised for an LCD display with subpixels in order from display left to right of B,G,R such that the horizontal subpixel resolution is three times that of the full pixel horizontal resolution (HBGR). |
| static [Object](http://docs.google.com/java/lang/Object.html) | [**VALUE\_TEXT\_ANTIALIAS\_LCD\_HRGB**](http://docs.google.com/java/awt/RenderingHints.html#VALUE_TEXT_ANTIALIAS_LCD_HRGB)            Text antialiasing hint value -- request that text be displayed optimised for an LCD display with subpixels in order from display left to right of R,G,B such that the horizontal subpixel resolution is three times that of the full pixel horizontal resolution (HRGB). |
| static [Object](http://docs.google.com/java/lang/Object.html) | [**VALUE\_TEXT\_ANTIALIAS\_LCD\_VBGR**](http://docs.google.com/java/awt/RenderingHints.html#VALUE_TEXT_ANTIALIAS_LCD_VBGR)            Text antialiasing hint value -- request that text be displayed optimised for an LCD display with subpixel organisation from display top to bottom of B,G,R such that the vertical subpixel resolution is three times that of the full pixel vertical resolution (VBGR). |
| static [Object](http://docs.google.com/java/lang/Object.html) | [**VALUE\_TEXT\_ANTIALIAS\_LCD\_VRGB**](http://docs.google.com/java/awt/RenderingHints.html#VALUE_TEXT_ANTIALIAS_LCD_VRGB)            Text antialiasing hint value -- request that text be displayed optimised for an LCD display with subpixel organisation from display top to bottom of R,G,B such that the vertical subpixel resolution is three times that of the full pixel vertical resolution (VRGB). |
| static [Object](http://docs.google.com/java/lang/Object.html) | [**VALUE\_TEXT\_ANTIALIAS\_OFF**](http://docs.google.com/java/awt/RenderingHints.html#VALUE_TEXT_ANTIALIAS_OFF)            Text antialiasing hint value -- text rendering is done without any form of antialiasing. |
| static [Object](http://docs.google.com/java/lang/Object.html) | [**VALUE\_TEXT\_ANTIALIAS\_ON**](http://docs.google.com/java/awt/RenderingHints.html#VALUE_TEXT_ANTIALIAS_ON)            Text antialiasing hint value -- text rendering is done with some form of antialiasing. |

| **Constructor Summary** | |
| --- | --- |
| [**RenderingHints**](http://docs.google.com/java/awt/RenderingHints.html#RenderingHints(java.util.Map))([Map](http://docs.google.com/java/util/Map.html)<[RenderingHints.Key](http://docs.google.com/java/awt/RenderingHints.Key.html),?> init)            Constructs a new object with keys and values initialized from the specified Map object which may be null. |
| [**RenderingHints**](http://docs.google.com/java/awt/RenderingHints.html#RenderingHints(java.awt.RenderingHints.Key,%20java.lang.Object))([RenderingHints.Key](http://docs.google.com/java/awt/RenderingHints.Key.html) key, [Object](http://docs.google.com/java/lang/Object.html) value)            Constructs a new object with the specified key/value pair. |

| **Method Summary** | |
| --- | --- |
| void | [**add**](http://docs.google.com/java/awt/RenderingHints.html#add(java.awt.RenderingHints))([RenderingHints](http://docs.google.com/java/awt/RenderingHints.html) hints)            Adds all of the keys and corresponding values from the specified RenderingHints object to this RenderingHints object. |
| void | [**clear**](http://docs.google.com/java/awt/RenderingHints.html#clear())()            Clears this RenderingHints object of all key/value pairs. |
| [Object](http://docs.google.com/java/lang/Object.html) | [**clone**](http://docs.google.com/java/awt/RenderingHints.html#clone())()            Creates a clone of this RenderingHints object that has the same contents as this RenderingHints object. |
| boolean | [**containsKey**](http://docs.google.com/java/awt/RenderingHints.html#containsKey(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) key)            Returns true if this RenderingHints contains a mapping for the specified key. |
| boolean | [**containsValue**](http://docs.google.com/java/awt/RenderingHints.html#containsValue(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) value)            Returns true if this RenderingHints maps one or more keys to the specified value. |
| [Set](http://docs.google.com/java/util/Set.html)<[Map.Entry](http://docs.google.com/java/util/Map.Entry.html)<[Object](http://docs.google.com/java/lang/Object.html),[Object](http://docs.google.com/java/lang/Object.html)>> | [**entrySet**](http://docs.google.com/java/awt/RenderingHints.html#entrySet())()            Returns a Set view of the mappings contained in this RenderingHints. |
| boolean | [**equals**](http://docs.google.com/java/awt/RenderingHints.html#equals(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) o)            Compares the specified Object with this RenderingHints for equality. |
| [Object](http://docs.google.com/java/lang/Object.html) | [**get**](http://docs.google.com/java/awt/RenderingHints.html#get(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) key)            Returns the value to which the specified key is mapped. |
| int | [**hashCode**](http://docs.google.com/java/awt/RenderingHints.html#hashCode())()            Returns the hash code value for this RenderingHints. |
| boolean | [**isEmpty**](http://docs.google.com/java/awt/RenderingHints.html#isEmpty())()            Returns true if this RenderingHints contains no key-value mappings. |
| [Set](http://docs.google.com/java/util/Set.html)<[Object](http://docs.google.com/java/lang/Object.html)> | [**keySet**](http://docs.google.com/java/awt/RenderingHints.html#keySet())()            Returns a Set view of the Keys contained in this RenderingHints. |
| [Object](http://docs.google.com/java/lang/Object.html) | [**put**](http://docs.google.com/java/awt/RenderingHints.html#put(java.lang.Object,%20java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) key, [Object](http://docs.google.com/java/lang/Object.html) value)            Maps the specified key to the specified value in this RenderingHints object. |
| void | [**putAll**](http://docs.google.com/java/awt/RenderingHints.html#putAll(java.util.Map))([Map](http://docs.google.com/java/util/Map.html)<?,?> m)            Copies all of the mappings from the specified Map to this RenderingHints. |
| [Object](http://docs.google.com/java/lang/Object.html) | [**remove**](http://docs.google.com/java/awt/RenderingHints.html#remove(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) key)            Removes the key and its corresponding value from this RenderingHints object. |
| int | [**size**](http://docs.google.com/java/awt/RenderingHints.html#size())()            Returns the number of key-value mappings in this RenderingHints. |
| [String](http://docs.google.com/java/lang/String.html) | [**toString**](http://docs.google.com/java/awt/RenderingHints.html#toString())()            Returns a rather long string representation of the hashmap which contains the mappings of keys to values for this RenderingHints object. |
| [Collection](http://docs.google.com/java/util/Collection.html)<[Object](http://docs.google.com/java/lang/Object.html)> | [**values**](http://docs.google.com/java/awt/RenderingHints.html#values())()            Returns a Collection view of the values contained in this RenderinHints. |

| **Methods inherited from class java.lang.**[**Object**](http://docs.google.com/java/lang/Object.html) |
| --- |
| [finalize](http://docs.google.com/java/lang/Object.html#finalize()), [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** |
| --- |

### KEY\_ANTIALIASING

public static final [RenderingHints.Key](http://docs.google.com/java/awt/RenderingHints.Key.html) **KEY\_ANTIALIASING**

Antialiasing hint key. The ANTIALIASING hint controls whether or not the geometry rendering methods of a [Graphics2D](http://docs.google.com/java/awt/Graphics2D.html) object will attempt to reduce aliasing artifacts along the edges of shapes.

A typical antialiasing algorithm works by blending the existing colors of the pixels along the boundary of a shape with the requested fill paint according to the estimated partial pixel coverage of the shape.

The allowable values for this hint are

* [VALUE\_ANTIALIAS\_ON](http://docs.google.com/java/awt/RenderingHints.html#VALUE_ANTIALIAS_ON)
* [VALUE\_ANTIALIAS\_OFF](http://docs.google.com/java/awt/RenderingHints.html#VALUE_ANTIALIAS_OFF)
* [VALUE\_ANTIALIAS\_DEFAULT](http://docs.google.com/java/awt/RenderingHints.html#VALUE_ANTIALIAS_DEFAULT)

### VALUE\_ANTIALIAS\_ON

public static final [Object](http://docs.google.com/java/lang/Object.html) **VALUE\_ANTIALIAS\_ON**

Antialiasing hint value -- rendering is done with antialiasing.

**See Also:**[KEY\_ANTIALIASING](http://docs.google.com/java/awt/RenderingHints.html#KEY_ANTIALIASING)

### VALUE\_ANTIALIAS\_OFF

public static final [Object](http://docs.google.com/java/lang/Object.html) **VALUE\_ANTIALIAS\_OFF**

Antialiasing hint value -- rendering is done without antialiasing.

**See Also:**[KEY\_ANTIALIASING](http://docs.google.com/java/awt/RenderingHints.html#KEY_ANTIALIASING)

### VALUE\_ANTIALIAS\_DEFAULT

public static final [Object](http://docs.google.com/java/lang/Object.html) **VALUE\_ANTIALIAS\_DEFAULT**

Antialiasing hint value -- rendering is done with a default antialiasing mode chosen by the implementation.

**See Also:**[KEY\_ANTIALIASING](http://docs.google.com/java/awt/RenderingHints.html#KEY_ANTIALIASING)

### KEY\_RENDERING

public static final [RenderingHints.Key](http://docs.google.com/java/awt/RenderingHints.Key.html) **KEY\_RENDERING**

Rendering hint key. The RENDERING hint is a general hint that provides a high level recommendation as to whether to bias algorithm choices more for speed or quality when evaluating tradeoffs. This hint could be consulted for any rendering or image manipulation operation, but decisions will usually honor other, more specific hints in preference to this hint.

The allowable values for this hint are

* [VALUE\_RENDER\_SPEED](http://docs.google.com/java/awt/RenderingHints.html#VALUE_RENDER_SPEED)
* [VALUE\_RENDER\_QUALITY](http://docs.google.com/java/awt/RenderingHints.html#VALUE_RENDER_QUALITY)
* [VALUE\_RENDER\_DEFAULT](http://docs.google.com/java/awt/RenderingHints.html#VALUE_RENDER_DEFAULT)

### VALUE\_RENDER\_SPEED

public static final [Object](http://docs.google.com/java/lang/Object.html) **VALUE\_RENDER\_SPEED**

Rendering hint value -- rendering algorithms are chosen with a preference for output speed.

**See Also:**[KEY\_RENDERING](http://docs.google.com/java/awt/RenderingHints.html#KEY_RENDERING)

### VALUE\_RENDER\_QUALITY

public static final [Object](http://docs.google.com/java/lang/Object.html) **VALUE\_RENDER\_QUALITY**

Rendering hint value -- rendering algorithms are chosen with a preference for output quality.

**See Also:**[KEY\_RENDERING](http://docs.google.com/java/awt/RenderingHints.html#KEY_RENDERING)

### VALUE\_RENDER\_DEFAULT

public static final [Object](http://docs.google.com/java/lang/Object.html) **VALUE\_RENDER\_DEFAULT**

Rendering hint value -- rendering algorithms are chosen by the implementation for a good tradeoff of performance vs. quality.

**See Also:**[KEY\_RENDERING](http://docs.google.com/java/awt/RenderingHints.html#KEY_RENDERING)

### KEY\_DITHERING

public static final [RenderingHints.Key](http://docs.google.com/java/awt/RenderingHints.Key.html) **KEY\_DITHERING**

Dithering hint key. The DITHERING hint controls how closely to approximate a color when storing into a destination with limited color resolution.

Some rendering destinations may support a limited number of color choices which may not be able to accurately represent the full spectrum of colors that can result during rendering operations. For such a destination the DITHERING hint controls whether rendering is done with a flat solid fill of a single pixel value which is the closest supported color to what was requested, or whether shapes will be filled with a pattern of colors which combine to better approximate that color.

The allowable values for this hint are

* [VALUE\_DITHER\_DISABLE](http://docs.google.com/java/awt/RenderingHints.html#VALUE_DITHER_DISABLE)
* [VALUE\_DITHER\_ENABLE](http://docs.google.com/java/awt/RenderingHints.html#VALUE_DITHER_ENABLE)
* [VALUE\_DITHER\_DEFAULT](http://docs.google.com/java/awt/RenderingHints.html#VALUE_DITHER_DEFAULT)

### VALUE\_DITHER\_DISABLE

public static final [Object](http://docs.google.com/java/lang/Object.html) **VALUE\_DITHER\_DISABLE**

Dithering hint value -- do not dither when rendering geometry.

**See Also:**[KEY\_DITHERING](http://docs.google.com/java/awt/RenderingHints.html#KEY_DITHERING)

### VALUE\_DITHER\_ENABLE

public static final [Object](http://docs.google.com/java/lang/Object.html) **VALUE\_DITHER\_ENABLE**

Dithering hint value -- dither when rendering geometry, if needed.

**See Also:**[KEY\_DITHERING](http://docs.google.com/java/awt/RenderingHints.html#KEY_DITHERING)

### VALUE\_DITHER\_DEFAULT

public static final [Object](http://docs.google.com/java/lang/Object.html) **VALUE\_DITHER\_DEFAULT**

Dithering hint value -- use a default for dithering chosen by the implementation.

**See Also:**[KEY\_DITHERING](http://docs.google.com/java/awt/RenderingHints.html#KEY_DITHERING)

### KEY\_TEXT\_ANTIALIASING

public static final [RenderingHints.Key](http://docs.google.com/java/awt/RenderingHints.Key.html) **KEY\_TEXT\_ANTIALIASING**

Text antialiasing hint key. The TEXT\_ANTIALIASING hint can control the use of antialiasing algorithms for text independently of the choice used for shape rendering. Often an application may want to use antialiasing for text only and not for other shapes. Additionally, the algorithms for reducing the aliasing artifacts for text are often more sophisticated than those that have been developed for general rendering so this hint key provides additional values which can control the choices of some of those text-specific algorithms. If left in the DEFAULT state, this hint will generally defer to the value of the regular [KEY\_ANTIALIASING](http://docs.google.com/java/awt/RenderingHints.html#KEY_ANTIALIASING) hint key.

The allowable values for this hint are

* [VALUE\_TEXT\_ANTIALIAS\_ON](http://docs.google.com/java/awt/RenderingHints.html#VALUE_TEXT_ANTIALIAS_ON)
* [VALUE\_TEXT\_ANTIALIAS\_OFF](http://docs.google.com/java/awt/RenderingHints.html#VALUE_TEXT_ANTIALIAS_OFF)
* [VALUE\_TEXT\_ANTIALIAS\_DEFAULT](http://docs.google.com/java/awt/RenderingHints.html#VALUE_TEXT_ANTIALIAS_DEFAULT)
* [VALUE\_TEXT\_ANTIALIAS\_GASP](http://docs.google.com/java/awt/RenderingHints.html#VALUE_TEXT_ANTIALIAS_GASP)
* [VALUE\_TEXT\_ANTIALIAS\_LCD\_HRGB](http://docs.google.com/java/awt/RenderingHints.html#VALUE_TEXT_ANTIALIAS_LCD_HRGB)
* [VALUE\_TEXT\_ANTIALIAS\_LCD\_HBGR](http://docs.google.com/java/awt/RenderingHints.html#VALUE_TEXT_ANTIALIAS_LCD_HBGR)
* [VALUE\_TEXT\_ANTIALIAS\_LCD\_VRGB](http://docs.google.com/java/awt/RenderingHints.html#VALUE_TEXT_ANTIALIAS_LCD_VRGB)
* [VALUE\_TEXT\_ANTIALIAS\_LCD\_VBGR](http://docs.google.com/java/awt/RenderingHints.html#VALUE_TEXT_ANTIALIAS_LCD_VBGR)

### VALUE\_TEXT\_ANTIALIAS\_ON

public static final [Object](http://docs.google.com/java/lang/Object.html) **VALUE\_TEXT\_ANTIALIAS\_ON**

Text antialiasing hint value -- text rendering is done with some form of antialiasing.

**See Also:**[KEY\_TEXT\_ANTIALIASING](http://docs.google.com/java/awt/RenderingHints.html#KEY_TEXT_ANTIALIASING)

### VALUE\_TEXT\_ANTIALIAS\_OFF

public static final [Object](http://docs.google.com/java/lang/Object.html) **VALUE\_TEXT\_ANTIALIAS\_OFF**

Text antialiasing hint value -- text rendering is done without any form of antialiasing.

**See Also:**[KEY\_TEXT\_ANTIALIASING](http://docs.google.com/java/awt/RenderingHints.html#KEY_TEXT_ANTIALIASING)

### VALUE\_TEXT\_ANTIALIAS\_DEFAULT

public static final [Object](http://docs.google.com/java/lang/Object.html) **VALUE\_TEXT\_ANTIALIAS\_DEFAULT**

Text antialiasing hint value -- text rendering is done according to the [KEY\_ANTIALIASING](http://docs.google.com/java/awt/RenderingHints.html#KEY_ANTIALIASING) hint or a default chosen by the implementation.

**See Also:**[KEY\_TEXT\_ANTIALIASING](http://docs.google.com/java/awt/RenderingHints.html#KEY_TEXT_ANTIALIASING)

### VALUE\_TEXT\_ANTIALIAS\_GASP

public static final [Object](http://docs.google.com/java/lang/Object.html) **VALUE\_TEXT\_ANTIALIAS\_GASP**

Text antialiasing hint value -- text rendering is requested to use information in the font resource which specifies for each point size whether to apply [VALUE\_TEXT\_ANTIALIAS\_ON](http://docs.google.com/java/awt/RenderingHints.html#VALUE_TEXT_ANTIALIAS_ON) or [VALUE\_TEXT\_ANTIALIAS\_OFF](http://docs.google.com/java/awt/RenderingHints.html#VALUE_TEXT_ANTIALIAS_OFF).

TrueType fonts typically provide this information in the 'gasp' table. In the absence of this information, the behaviour for a particular font and size is determined by implementation defaults.

*Note:*A font designer will typically carefully hint a font for the most common user interface point sizes. Consequently the 'gasp' table will likely specify to use only hinting at those sizes and not "smoothing". So in many cases the resulting text display is equivalent to VALUE\_TEXT\_ANTIALIAS\_OFF. This may be unexpected but is correct.

Logical fonts which are composed of multiple physical fonts will for consistency will use the setting most appropriate for the overall composite font.

**Since:** 1.6 **See Also:**[KEY\_TEXT\_ANTIALIASING](http://docs.google.com/java/awt/RenderingHints.html#KEY_TEXT_ANTIALIASING)

### VALUE\_TEXT\_ANTIALIAS\_LCD\_HRGB

public static final [Object](http://docs.google.com/java/lang/Object.html) **VALUE\_TEXT\_ANTIALIAS\_LCD\_HRGB**

Text antialiasing hint value -- request that text be displayed optimised for an LCD display with subpixels in order from display left to right of R,G,B such that the horizontal subpixel resolution is three times that of the full pixel horizontal resolution (HRGB). This is the most common configuration. Selecting this hint for displays with one of the other LCD subpixel configurations will likely result in unfocused text.

*Notes:*

An implementation when choosing whether to apply any of the LCD text hint values may take into account factors including requiring color depth of the destination to be at least 15 bits per pixel (ie 5 bits per color component), characteristics of a font such as whether embedded bitmaps may produce better results, or when displaying to a non-local networked display device enabling it only if suitable protocols are available, or ignoring the hint if performing very high resolution rendering or the target device is not appropriate: eg when printing.

These hints can equally be applied when rendering to software images, but these images may not then be suitable for general export, as the text will have been rendered appropriately for a specific subpixel organisation. Also lossy images are not a good choice, nor image formats such as GIF which have limited colors. So unless the image is destined solely for rendering on a display device with the same configuration, some other text anti-aliasing hint such as [VALUE\_TEXT\_ANTIALIAS\_ON](http://docs.google.com/java/awt/RenderingHints.html#VALUE_TEXT_ANTIALIAS_ON) may be a better choice.

Selecting a value which does not match the LCD display in use will likely lead to a degradation in text quality. On display devices (ie CRTs) which do not have the same characteristics as LCD displays, the overall effect may appear similar to standard text anti-aliasing, but the quality may be degraded by color distortion. Analog connected LCD displays may also show little advantage over standard text-antialiasing and be similar to CRTs.

In other words for the best results use an LCD display with a digital display connector and specify the appropriate sub-pixel configuration.

**Since:** 1.6 **See Also:**[KEY\_TEXT\_ANTIALIASING](http://docs.google.com/java/awt/RenderingHints.html#KEY_TEXT_ANTIALIASING)

### VALUE\_TEXT\_ANTIALIAS\_LCD\_HBGR

public static final [Object](http://docs.google.com/java/lang/Object.html) **VALUE\_TEXT\_ANTIALIAS\_LCD\_HBGR**

Text antialiasing hint value -- request that text be displayed optimised for an LCD display with subpixels in order from display left to right of B,G,R such that the horizontal subpixel resolution is three times that of the full pixel horizontal resolution (HBGR). This is a much less common configuration than HRGB. Selecting this hint for displays with one of the other LCD subpixel configurations will likely result in unfocused text. See [VALUE\_TEXT\_ANTIALIAS\_LCD\_HRGB](http://docs.google.com/java/awt/RenderingHints.html#VALUE_TEXT_ANTIALIAS_LCD_HRGB), for more information on when this hint is applied.

**Since:** 1.6 **See Also:**[KEY\_TEXT\_ANTIALIASING](http://docs.google.com/java/awt/RenderingHints.html#KEY_TEXT_ANTIALIASING)

### VALUE\_TEXT\_ANTIALIAS\_LCD\_VRGB

public static final [Object](http://docs.google.com/java/lang/Object.html) **VALUE\_TEXT\_ANTIALIAS\_LCD\_VRGB**

Text antialiasing hint value -- request that text be displayed optimised for an LCD display with subpixel organisation from display top to bottom of R,G,B such that the vertical subpixel resolution is three times that of the full pixel vertical resolution (VRGB). Vertical orientation is very uncommon and probably mainly useful for a physically rotated display. Selecting this hint for displays with one of the other LCD subpixel configurations will likely result in unfocused text. See [VALUE\_TEXT\_ANTIALIAS\_LCD\_HRGB](http://docs.google.com/java/awt/RenderingHints.html#VALUE_TEXT_ANTIALIAS_LCD_HRGB), for more information on when this hint is applied.

**Since:** 1.6 **See Also:**[KEY\_TEXT\_ANTIALIASING](http://docs.google.com/java/awt/RenderingHints.html#KEY_TEXT_ANTIALIASING)

### VALUE\_TEXT\_ANTIALIAS\_LCD\_VBGR

public static final [Object](http://docs.google.com/java/lang/Object.html) **VALUE\_TEXT\_ANTIALIAS\_LCD\_VBGR**

Text antialiasing hint value -- request that text be displayed optimised for an LCD display with subpixel organisation from display top to bottom of B,G,R such that the vertical subpixel resolution is three times that of the full pixel vertical resolution (VBGR). Vertical orientation is very uncommon and probably mainly useful for a physically rotated display. Selecting this hint for displays with one of the other LCD subpixel configurations will likely result in unfocused text. See [VALUE\_TEXT\_ANTIALIAS\_LCD\_HRGB](http://docs.google.com/java/awt/RenderingHints.html#VALUE_TEXT_ANTIALIAS_LCD_HRGB), for more information on when this hint is applied.

**Since:** 1.6 **See Also:**[KEY\_TEXT\_ANTIALIASING](http://docs.google.com/java/awt/RenderingHints.html#KEY_TEXT_ANTIALIASING)

### KEY\_TEXT\_LCD\_CONTRAST

public static final [RenderingHints.Key](http://docs.google.com/java/awt/RenderingHints.Key.html) **KEY\_TEXT\_LCD\_CONTRAST**

LCD text contrast rendering hint key. The value is an Integer object which is used as a text contrast adjustment when used in conjunction with an LCD text anti-aliasing hint such as [VALUE\_TEXT\_ANTIALIAS\_LCD\_HRGB](http://docs.google.com/java/awt/RenderingHints.html#VALUE_TEXT_ANTIALIAS_LCD_HRGB).

* Values should be a positive integer in the range 100 to 250.
* A lower value (eg 100) corresponds to higher contrast text when displaying dark text on a light background.
* A higher value (eg 200) corresponds to lower contrast text when displaying dark text on a light background.
* A typical useful value is in the narrow range 140-180.
* If no value is specified, a system or implementation default value will be applied.

The default value can be expected to be adequate for most purposes, so clients should rarely need to specify a value for this hint unless they have concrete information as to an appropriate value. A higher value does not mean a higher contrast, in fact the opposite is true. The correction is applied in a similar manner to a gamma adjustment for non-linear perceptual luminance response of display systems, but does not indicate a full correction for this.

**Since:** 1.6 **See Also:**[KEY\_TEXT\_ANTIALIASING](http://docs.google.com/java/awt/RenderingHints.html#KEY_TEXT_ANTIALIASING)

### KEY\_FRACTIONALMETRICS

public static final [RenderingHints.Key](http://docs.google.com/java/awt/RenderingHints.Key.html) **KEY\_FRACTIONALMETRICS**

Font fractional metrics hint key. The FRACTIONALMETRICS hint controls whether the positioning of individual character glyphs takes into account the sub-pixel accuracy of the scaled character advances of the font or whether such advance vectors are rounded to an integer number of whole device pixels. This hint only recommends how much accuracy should be used to position the glyphs and does not specify or recommend whether or not the actual rasterization or pixel bounds of the glyph should be modified to match.

Rendering text to a low resolution device like a screen will necessarily involve a number of rounding operations as the high quality and very precise definition of the shape and metrics of the character glyphs must be matched to discrete device pixels. Ideally the positioning of glyphs during text layout would be calculated by scaling the design metrics in the font according to the point size, but then the scaled advance width will not necessarily be an integer number of pixels. If the glyphs are positioned with sub-pixel accuracy according to these scaled design metrics then the rasterization would ideally need to be adjusted for each possible sub-pixel origin.

Unfortunately, scaling each glyph customized to its exact subpixel origin during text layout would be prohibitively expensive so a simplified system based on integer device positions is typically used to lay out the text. The rasterization of the glyph and the scaled advance width are both adjusted together to yield text that looks good at device resolution and has consistent integer pixel distances between glyphs that help the glyphs look uniformly and consistently spaced and readable.

This process of rounding advance widths for rasterized glyphs to integer distances means that the character density and the overall length of a string of text will be different from the theoretical design measurements due to the accumulation of a series of small differences in the adjusted widths of each glyph. The specific differences will be different for each glyph, some being wider and some being narrower than their theoretical design measurements. Thus the overall difference in character density and length will vary by a number of factors including the font, the specific device resolution being targeted, and the glyphs chosen to represent the string being rendered. As a result, rendering the same string at multiple device resolutions can yield widely varying metrics for whole strings.

When FRACTIONALMETRICS are enabled, the true font design metrics are scaled by the point size and used for layout with sub-pixel accuracy. The average density of glyphs and total length of a long string of characters will therefore more closely match the theoretical design of the font, but readability may be affected since individual pairs of characters may not always appear to be consistent distances apart depending on how the sub-pixel accumulation of the glyph origins meshes with the device pixel grid. Enabling this hint may be desirable when text layout is being performed that must be consistent across a wide variety of output resolutions. Specifically, this hint may be desirable in situations where the layout of text is being previewed on a low resolution device like a screen for output that will eventually be rendered on a high resolution printer or typesetting device.

When disabled, the scaled design metrics are rounded or adjusted to integer distances for layout. The distances between any specific pair of glyphs will be more uniform on the device, but the density and total length of long strings may no longer match the theoretical intentions of the font designer. Disabling this hint will typically produce more readable results on low resolution devices like computer monitors.

The allowable values for this key are

* [VALUE\_FRACTIONALMETRICS\_OFF](http://docs.google.com/java/awt/RenderingHints.html#VALUE_FRACTIONALMETRICS_OFF)
* [VALUE\_FRACTIONALMETRICS\_ON](http://docs.google.com/java/awt/RenderingHints.html#VALUE_FRACTIONALMETRICS_ON)
* [VALUE\_FRACTIONALMETRICS\_DEFAULT](http://docs.google.com/java/awt/RenderingHints.html#VALUE_FRACTIONALMETRICS_DEFAULT)

### VALUE\_FRACTIONALMETRICS\_OFF

public static final [Object](http://docs.google.com/java/lang/Object.html) **VALUE\_FRACTIONALMETRICS\_OFF**

Font fractional metrics hint value -- character glyphs are positioned with advance widths rounded to pixel boundaries.

**See Also:**[KEY\_FRACTIONALMETRICS](http://docs.google.com/java/awt/RenderingHints.html#KEY_FRACTIONALMETRICS)

### VALUE\_FRACTIONALMETRICS\_ON

public static final [Object](http://docs.google.com/java/lang/Object.html) **VALUE\_FRACTIONALMETRICS\_ON**

Font fractional metrics hint value -- character glyphs are positioned with sub-pixel accuracy.

**See Also:**[KEY\_FRACTIONALMETRICS](http://docs.google.com/java/awt/RenderingHints.html#KEY_FRACTIONALMETRICS)

### VALUE\_FRACTIONALMETRICS\_DEFAULT

public static final [Object](http://docs.google.com/java/lang/Object.html) **VALUE\_FRACTIONALMETRICS\_DEFAULT**

Font fractional metrics hint value -- character glyphs are positioned with accuracy chosen by the implementation.

**See Also:**[KEY\_FRACTIONALMETRICS](http://docs.google.com/java/awt/RenderingHints.html#KEY_FRACTIONALMETRICS)

### KEY\_INTERPOLATION

public static final [RenderingHints.Key](http://docs.google.com/java/awt/RenderingHints.Key.html) **KEY\_INTERPOLATION**

Interpolation hint key. The INTERPOLATION hint controls how image pixels are filtered or resampled during an image rendering operation.

Implicitly images are defined to provide color samples at integer coordinate locations. When images are rendered upright with no scaling onto a destination, the choice of which image pixels map to which device pixels is obvious and the samples at the integer coordinate locations in the image are transfered to the pixels at the corresponding integer locations on the device pixel grid one for one. When images are rendered in a scaled, rotated, or otherwise transformed coordinate system, then the mapping of device pixel coordinates back to the image can raise the question of what color sample to use for the continuous coordinates that lie between the integer locations of the provided image samples. Interpolation algorithms define functions which provide a color sample for any continuous coordinate in an image based on the color samples at the surrounding integer coordinates.

The allowable values for this hint are

* [VALUE\_INTERPOLATION\_NEAREST\_NEIGHBOR](http://docs.google.com/java/awt/RenderingHints.html#VALUE_INTERPOLATION_NEAREST_NEIGHBOR)
* [VALUE\_INTERPOLATION\_BILINEAR](http://docs.google.com/java/awt/RenderingHints.html#VALUE_INTERPOLATION_BILINEAR)
* [VALUE\_INTERPOLATION\_BICUBIC](http://docs.google.com/java/awt/RenderingHints.html#VALUE_INTERPOLATION_BICUBIC)

### VALUE\_INTERPOLATION\_NEAREST\_NEIGHBOR

public static final [Object](http://docs.google.com/java/lang/Object.html) **VALUE\_INTERPOLATION\_NEAREST\_NEIGHBOR**

Interpolation hint value -- the color sample of the nearest neighboring integer coordinate sample in the image is used. Conceptually the image is viewed as a grid of unit-sized square regions of color centered around the center of each image pixel.

As the image is scaled up, it will look correspondingly blocky. As the image is scaled down, the colors for source pixels will be either used unmodified, or skipped entirely in the output representation.

**See Also:**[KEY\_INTERPOLATION](http://docs.google.com/java/awt/RenderingHints.html#KEY_INTERPOLATION)

### VALUE\_INTERPOLATION\_BILINEAR

public static final [Object](http://docs.google.com/java/lang/Object.html) **VALUE\_INTERPOLATION\_BILINEAR**

Interpolation hint value -- the color samples of the 4 nearest neighboring integer coordinate samples in the image are interpolated linearly to produce a color sample. Conceptually the image is viewed as a set of infinitely small point color samples which have value only at the centers of integer coordinate pixels and the space between those pixel centers is filled with linear ramps of colors that connect adjacent discrete samples in a straight line.

As the image is scaled up, there are no blocky edges between the colors in the image as there are with [NEAREST\_NEIGHBOR](http://docs.google.com/java/awt/RenderingHints.html#VALUE_INTERPOLATION_NEAREST_NEIGHBOR), but the blending may show some subtle discontinuities along the horizontal and vertical edges that line up with the samples caused by a sudden change in the slope of the interpolation from one side of a sample to the other. As the image is scaled down, more image pixels have their color samples represented in the resulting output since each output pixel recieves color information from up to 4 image pixels.

**See Also:**[KEY\_INTERPOLATION](http://docs.google.com/java/awt/RenderingHints.html#KEY_INTERPOLATION)

### VALUE\_INTERPOLATION\_BICUBIC

public static final [Object](http://docs.google.com/java/lang/Object.html) **VALUE\_INTERPOLATION\_BICUBIC**

Interpolation hint value -- the color samples of 9 nearby integer coordinate samples in the image are interpolated using a cubic function in both X and Y to produce a color sample. Conceptually the view of the image is very similar to the view used in the [BILINEAR](http://docs.google.com/java/awt/RenderingHints.html#VALUE_INTERPOLATION_BILINEAR) algorithm except that the ramps of colors that connect between the samples are curved and have better continuity of slope as they cross over between sample boundaries.

As the image is scaled up, there are no blocky edges and the interpolation should appear smoother and with better depictions of any edges in the original image than with BILINEAR. As the image is scaled down, even more of the original color samples from the original image will have their color information carried through and represented.

**See Also:**[KEY\_INTERPOLATION](http://docs.google.com/java/awt/RenderingHints.html#KEY_INTERPOLATION)

### KEY\_ALPHA\_INTERPOLATION

public static final [RenderingHints.Key](http://docs.google.com/java/awt/RenderingHints.Key.html) **KEY\_ALPHA\_INTERPOLATION**

Alpha interpolation hint key. The ALPHA\_INTERPOLATION hint is a general hint that provides a high level recommendation as to whether to bias alpha blending algorithm choices more for speed or quality when evaluating tradeoffs.

This hint could control the choice of alpha blending calculations that sacrifice some precision to use fast lookup tables or lower precision SIMD instructions. This hint could also control whether or not the color and alpha values are converted into a linear color space during the calculations for a more linear visual effect at the expense of additional per-pixel calculations.

The allowable values for this hint are

* [VALUE\_ALPHA\_INTERPOLATION\_SPEED](http://docs.google.com/java/awt/RenderingHints.html#VALUE_ALPHA_INTERPOLATION_SPEED)
* [VALUE\_ALPHA\_INTERPOLATION\_QUALITY](http://docs.google.com/java/awt/RenderingHints.html#VALUE_ALPHA_INTERPOLATION_QUALITY)
* [VALUE\_ALPHA\_INTERPOLATION\_DEFAULT](http://docs.google.com/java/awt/RenderingHints.html#VALUE_ALPHA_INTERPOLATION_DEFAULT)

### VALUE\_ALPHA\_INTERPOLATION\_SPEED

public static final [Object](http://docs.google.com/java/lang/Object.html) **VALUE\_ALPHA\_INTERPOLATION\_SPEED**

Alpha interpolation hint value -- alpha blending algorithms are chosen with a preference for calculation speed.

**See Also:**[KEY\_ALPHA\_INTERPOLATION](http://docs.google.com/java/awt/RenderingHints.html#KEY_ALPHA_INTERPOLATION)

### VALUE\_ALPHA\_INTERPOLATION\_QUALITY

public static final [Object](http://docs.google.com/java/lang/Object.html) **VALUE\_ALPHA\_INTERPOLATION\_QUALITY**

Alpha interpolation hint value -- alpha blending algorithms are chosen with a preference for precision and visual quality.

**See Also:**[KEY\_ALPHA\_INTERPOLATION](http://docs.google.com/java/awt/RenderingHints.html#KEY_ALPHA_INTERPOLATION)

### VALUE\_ALPHA\_INTERPOLATION\_DEFAULT

public static final [Object](http://docs.google.com/java/lang/Object.html) **VALUE\_ALPHA\_INTERPOLATION\_DEFAULT**

Alpha interpolation hint value -- alpha blending algorithms are chosen by the implementation for a good tradeoff of performance vs. quality.

**See Also:**[KEY\_ALPHA\_INTERPOLATION](http://docs.google.com/java/awt/RenderingHints.html#KEY_ALPHA_INTERPOLATION)

### KEY\_COLOR\_RENDERING

public static final [RenderingHints.Key](http://docs.google.com/java/awt/RenderingHints.Key.html) **KEY\_COLOR\_RENDERING**

Color rendering hint key. The COLOR\_RENDERING hint controls the accuracy of approximation and conversion when storing colors into a destination image or surface.

When a rendering or image manipulation operation produces a color value that must be stored into a destination, it must first convert that color into a form suitable for storing into the destination image or surface. Minimally, the color components must be converted to bit representations and ordered in the correct order or an index into a color lookup table must be chosen before the data can be stored into the destination memory. Without this minimal conversion, the data in the destination would likely represent random, incorrect or possibly even unsupported values. Algorithms to quickly convert the results of rendering operations into the color format of most common destinations are well known and fairly optimal to execute.

Simply performing the most basic color format conversion to store colors into a destination can potentially ignore a difference in the calibration of the [ColorSpace](http://docs.google.com/java/awt/color/ColorSpace.html) of the source and destination or other factors such as the linearity of the gamma correction. Unless the source and destination ColorSpace are identical, to correctly perform a rendering operation with the most care taken for the accuracy of the colors being represented, the source colors should be converted to a device independent ColorSpace and the results then converted back to the destination ColorSpace. Furthermore, if calculations such as the blending of multiple source colors are to be performed during the rendering operation, greater visual clarity can be achieved if the intermediate device independent ColorSpace is chosen to have a linear relationship between the values being calculated and the perception of the human eye to the response curves of the output device.

The allowable values for this hint are

* [VALUE\_COLOR\_RENDER\_SPEED](http://docs.google.com/java/awt/RenderingHints.html#VALUE_COLOR_RENDER_SPEED)
* [VALUE\_COLOR\_RENDER\_QUALITY](http://docs.google.com/java/awt/RenderingHints.html#VALUE_COLOR_RENDER_QUALITY)
* [VALUE\_COLOR\_RENDER\_DEFAULT](http://docs.google.com/java/awt/RenderingHints.html#VALUE_COLOR_RENDER_DEFAULT)

### VALUE\_COLOR\_RENDER\_SPEED

public static final [Object](http://docs.google.com/java/lang/Object.html) **VALUE\_COLOR\_RENDER\_SPEED**

Color rendering hint value -- perform the fastest color conversion to the format of the output device.

**See Also:**[KEY\_COLOR\_RENDERING](http://docs.google.com/java/awt/RenderingHints.html#KEY_COLOR_RENDERING)

### VALUE\_COLOR\_RENDER\_QUALITY

public static final [Object](http://docs.google.com/java/lang/Object.html) **VALUE\_COLOR\_RENDER\_QUALITY**

Color rendering hint value -- perform the color conversion calculations with the highest accuracy and visual quality.

**See Also:**[KEY\_COLOR\_RENDERING](http://docs.google.com/java/awt/RenderingHints.html#KEY_COLOR_RENDERING)

### VALUE\_COLOR\_RENDER\_DEFAULT

public static final [Object](http://docs.google.com/java/lang/Object.html) **VALUE\_COLOR\_RENDER\_DEFAULT**

Color rendering hint value -- perform color conversion calculations as chosen by the implementation to represent the best available tradeoff between performance and accuracy.

**See Also:**[KEY\_COLOR\_RENDERING](http://docs.google.com/java/awt/RenderingHints.html#KEY_COLOR_RENDERING)

### KEY\_STROKE\_CONTROL

public static final [RenderingHints.Key](http://docs.google.com/java/awt/RenderingHints.Key.html) **KEY\_STROKE\_CONTROL**

Stroke normalization control hint key. The STROKE\_CONTROL hint controls whether a rendering implementation should or is allowed to modify the geometry of rendered shapes for various purposes.

Some implementations may be able to use an optimized platform rendering library which may be faster than traditional software rendering algorithms on a given platform, but which may also not support floating point coordinates. Some implementations may also have sophisticated algorithms which perturb the coordinates of a path so that wide lines appear more uniform in width and spacing.

If an implementation performs any type of modification or "normalization" of a path, it should never move the coordinates by more than half a pixel in any direction.

The allowable values for this hint are

* [VALUE\_STROKE\_NORMALIZE](http://docs.google.com/java/awt/RenderingHints.html#VALUE_STROKE_NORMALIZE)
* [VALUE\_STROKE\_PURE](http://docs.google.com/java/awt/RenderingHints.html#VALUE_STROKE_PURE)
* [VALUE\_STROKE\_DEFAULT](http://docs.google.com/java/awt/RenderingHints.html#VALUE_STROKE_DEFAULT)

**Since:** 1.3

### VALUE\_STROKE\_DEFAULT

public static final [Object](http://docs.google.com/java/lang/Object.html) **VALUE\_STROKE\_DEFAULT**

Stroke normalization control hint value -- geometry may be modified or left pure depending on the tradeoffs in a given implementation. Typically this setting allows an implementation to use a fast integer coordinate based platform rendering library, but does not specifically request normalization for uniformity or aesthetics.

**Since:** 1.3 **See Also:**[KEY\_STROKE\_CONTROL](http://docs.google.com/java/awt/RenderingHints.html#KEY_STROKE_CONTROL)

### VALUE\_STROKE\_NORMALIZE

public static final [Object](http://docs.google.com/java/lang/Object.html) **VALUE\_STROKE\_NORMALIZE**

Stroke normalization control hint value -- geometry should be normalized to improve uniformity or spacing of lines and overall aesthetics. Note that different normalization algorithms may be more successful than others for given input paths.

**Since:** 1.3 **See Also:**[KEY\_STROKE\_CONTROL](http://docs.google.com/java/awt/RenderingHints.html#KEY_STROKE_CONTROL)

### VALUE\_STROKE\_PURE

public static final [Object](http://docs.google.com/java/lang/Object.html) **VALUE\_STROKE\_PURE**

Stroke normalization control hint value -- geometry should be left unmodified and rendered with sub-pixel accuracy.

**Since:** 1.3 **See Also:**[KEY\_STROKE\_CONTROL](http://docs.google.com/java/awt/RenderingHints.html#KEY_STROKE_CONTROL)

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

### RenderingHints

public **RenderingHints**([Map](http://docs.google.com/java/util/Map.html)<[RenderingHints.Key](http://docs.google.com/java/awt/RenderingHints.Key.html),?> init)

Constructs a new object with keys and values initialized from the specified Map object which may be null.

**Parameters:**init - a map of key/value pairs to initialize the hints or null if the object should be empty

### RenderingHints

public **RenderingHints**([RenderingHints.Key](http://docs.google.com/java/awt/RenderingHints.Key.html) key,  
 [Object](http://docs.google.com/java/lang/Object.html) value)

Constructs a new object with the specified key/value pair.

**Parameters:**key - the key of the particular hint propertyvalue - the value of the hint property specified with key

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

### size

public int **size**()

Returns the number of key-value mappings in this RenderingHints.

**Specified by:**[size](http://docs.google.com/java/util/Map.html#size()) in interface [Map](http://docs.google.com/java/util/Map.html)<[Object](http://docs.google.com/java/lang/Object.html),[Object](http://docs.google.com/java/lang/Object.html)> **Returns:**the number of key-value mappings in this RenderingHints.

### isEmpty

public boolean **isEmpty**()

Returns true if this RenderingHints contains no key-value mappings.

**Specified by:**[isEmpty](http://docs.google.com/java/util/Map.html#isEmpty()) in interface [Map](http://docs.google.com/java/util/Map.html)<[Object](http://docs.google.com/java/lang/Object.html),[Object](http://docs.google.com/java/lang/Object.html)> **Returns:**true if this RenderingHints contains no key-value mappings.

### containsKey

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

Returns true if this RenderingHints contains a mapping for the specified key.

**Specified by:**[containsKey](http://docs.google.com/java/util/Map.html#containsKey(java.lang.Object)) in interface [Map](http://docs.google.com/java/util/Map.html)<[Object](http://docs.google.com/java/lang/Object.html),[Object](http://docs.google.com/java/lang/Object.html)> **Parameters:**key - key whose presence in this RenderingHints is to be tested. **Returns:**true if this RenderingHints contains a mapping for the specified key. **Throws:** ClassCastException - if the key can not be cast to RenderingHints.Key

### containsValue

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

Returns true if this RenderingHints maps one or more keys to the specified value. More formally, returns true if and only if this RenderingHints contains at least one mapping to a value v such that

(value==null ? v==null : value.equals(v))

. This operation will probably require time linear in the RenderingHints size for most implementations of RenderingHints.

**Specified by:**[containsValue](http://docs.google.com/java/util/Map.html#containsValue(java.lang.Object)) in interface [Map](http://docs.google.com/java/util/Map.html)<[Object](http://docs.google.com/java/lang/Object.html),[Object](http://docs.google.com/java/lang/Object.html)> **Parameters:**value - value whose presence in this RenderingHints is to be tested. **Returns:**true if this RenderingHints maps one or more keys to the specified value.

### get

public [Object](http://docs.google.com/java/lang/Object.html) **get**([Object](http://docs.google.com/java/lang/Object.html) key)

Returns the value to which the specified key is mapped.

**Specified by:**[get](http://docs.google.com/java/util/Map.html#get(java.lang.Object)) in interface [Map](http://docs.google.com/java/util/Map.html)<[Object](http://docs.google.com/java/lang/Object.html),[Object](http://docs.google.com/java/lang/Object.html)> **Parameters:**key - a rendering hint key **Returns:**the value to which the key is mapped in this object or null if the key is not mapped to any value in this object. **Throws:** ClassCastException - if the key can not be cast to RenderingHints.Key**See Also:**[put(Object, Object)](http://docs.google.com/java/awt/RenderingHints.html#put(java.lang.Object,%20java.lang.Object))

### put

public [Object](http://docs.google.com/java/lang/Object.html) **put**([Object](http://docs.google.com/java/lang/Object.html) key,  
 [Object](http://docs.google.com/java/lang/Object.html) value)

Maps the specified key to the specified value in this RenderingHints object. Neither the key nor the value can be null. The value can be retrieved by calling the get method with a key that is equal to the original key.

**Specified by:**[put](http://docs.google.com/java/util/Map.html#put(K,%20V)) in interface [Map](http://docs.google.com/java/util/Map.html)<[Object](http://docs.google.com/java/lang/Object.html),[Object](http://docs.google.com/java/lang/Object.html)> **Parameters:**key - the rendering hint key.value - the rendering hint value. **Returns:**the previous value of the specified key in this object or null if it did not have one. **Throws:** NullPointerException - if the key is null. ClassCastException - if the key can not be cast to RenderingHints.Key IllegalArgumentException - if the [Key.isCompatibleValue()](http://docs.google.com/java/awt/RenderingHints.Key.html#isCompatibleValue(java.lang.Object)) method of the specified key returns false for the specified value**See Also:**[get(Object)](http://docs.google.com/java/awt/RenderingHints.html#get(java.lang.Object))

### add

public void **add**([RenderingHints](http://docs.google.com/java/awt/RenderingHints.html) hints)

Adds all of the keys and corresponding values from the specified RenderingHints object to this RenderingHints object. Keys that are present in this RenderingHints object, but not in the specified RenderingHints object are not affected.

**Parameters:**hints - the set of key/value pairs to be added to this RenderingHints object

### clear

public void **clear**()

Clears this RenderingHints object of all key/value pairs.

**Specified by:**[clear](http://docs.google.com/java/util/Map.html#clear()) in interface [Map](http://docs.google.com/java/util/Map.html)<[Object](http://docs.google.com/java/lang/Object.html),[Object](http://docs.google.com/java/lang/Object.html)>

### remove

public [Object](http://docs.google.com/java/lang/Object.html) **remove**([Object](http://docs.google.com/java/lang/Object.html) key)

Removes the key and its corresponding value from this RenderingHints object. This method does nothing if the key is not in this RenderingHints object.

**Specified by:**[remove](http://docs.google.com/java/util/Map.html#remove(java.lang.Object)) in interface [Map](http://docs.google.com/java/util/Map.html)<[Object](http://docs.google.com/java/lang/Object.html),[Object](http://docs.google.com/java/lang/Object.html)> **Parameters:**key - the rendering hints key that needs to be removed **Returns:**the value to which the key had previously been mapped in this RenderingHints object, or null if the key did not have a mapping. **Throws:** ClassCastException - if the key can not be cast to RenderingHints.Key

### putAll

public void **putAll**([Map](http://docs.google.com/java/util/Map.html)<?,?> m)

Copies all of the mappings from the specified Map to this RenderingHints. These mappings replace any mappings that this RenderingHints had for any of the keys currently in the specified Map.

**Specified by:**[putAll](http://docs.google.com/java/util/Map.html#putAll(java.util.Map)) in interface [Map](http://docs.google.com/java/util/Map.html)<[Object](http://docs.google.com/java/lang/Object.html),[Object](http://docs.google.com/java/lang/Object.html)> **Parameters:**m - the specified Map **Throws:** ClassCastException - class of a key or value in the specified Map prevents it from being stored in this RenderingHints. IllegalArgumentException - some aspect of a key or value in the specified Map prevents it from being stored in this RenderingHints.

### keySet

public [Set](http://docs.google.com/java/util/Set.html)<[Object](http://docs.google.com/java/lang/Object.html)> **keySet**()

Returns a Set view of the Keys contained in this RenderingHints. The Set is backed by the RenderingHints, so changes to the RenderingHints are reflected in the Set, and vice-versa. If the RenderingHints is modified while an iteration over the Set is in progress, the results of the iteration are undefined. The Set supports element removal, which removes the corresponding mapping from the RenderingHints, via the Iterator.remove, Set.remove, removeAll retainAll, and clear operations. It does not support the add or addAll operations.

**Specified by:**[keySet](http://docs.google.com/java/util/Map.html#keySet()) in interface [Map](http://docs.google.com/java/util/Map.html)<[Object](http://docs.google.com/java/lang/Object.html),[Object](http://docs.google.com/java/lang/Object.html)> **Returns:**a Set view of the keys contained in this RenderingHints.

### values

public [Collection](http://docs.google.com/java/util/Collection.html)<[Object](http://docs.google.com/java/lang/Object.html)> **values**()

Returns a Collection view of the values contained in this RenderinHints. The Collection is backed by the RenderingHints, so changes to the RenderingHints are reflected in the Collection, and vice-versa. If the RenderingHints is modified while an iteration over the Collection is in progress, the results of the iteration are undefined. The Collection supports element removal, which removes the corresponding mapping from the RenderingHints, via the Iterator.remove, Collection.remove, removeAll, retainAll and clear operations. It does not support the add or addAll operations.

**Specified by:**[values](http://docs.google.com/java/util/Map.html#values()) in interface [Map](http://docs.google.com/java/util/Map.html)<[Object](http://docs.google.com/java/lang/Object.html),[Object](http://docs.google.com/java/lang/Object.html)> **Returns:**a Collection view of the values contained in this RenderingHints.

### entrySet

public [Set](http://docs.google.com/java/util/Set.html)<[Map.Entry](http://docs.google.com/java/util/Map.Entry.html)<[Object](http://docs.google.com/java/lang/Object.html),[Object](http://docs.google.com/java/lang/Object.html)>> **entrySet**()

Returns a Set view of the mappings contained in this RenderingHints. Each element in the returned Set is a Map.Entry. The Set is backed by the RenderingHints, so changes to the RenderingHints are reflected in the Set, and vice-versa. If the RenderingHints is modified while while an iteration over the Set is in progress, the results of the iteration are undefined.

The entrySet returned from a RenderingHints object is not modifiable.

**Specified by:**[entrySet](http://docs.google.com/java/util/Map.html#entrySet()) in interface [Map](http://docs.google.com/java/util/Map.html)<[Object](http://docs.google.com/java/lang/Object.html),[Object](http://docs.google.com/java/lang/Object.html)> **Returns:**a Set view of the mappings contained in this RenderingHints.

### equals

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

Compares the specified Object with this RenderingHints for equality. Returns true if the specified object is also a Map and the two Map objects represent the same mappings. More formally, two Map objects t1 and t2 represent the same mappings if t1.keySet().equals(t2.keySet()) and for every key k in t1.keySet(),

(t1.get(k)==null ? t2.get(k)==null : t1.get(k).equals(t2.get(k)))

. This ensures that the equals method works properly across different implementations of the Map interface.

**Specified by:**[equals](http://docs.google.com/java/util/Map.html#equals(java.lang.Object)) in interface [Map](http://docs.google.com/java/util/Map.html)<[Object](http://docs.google.com/java/lang/Object.html),[Object](http://docs.google.com/java/lang/Object.html)>**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:**o - Object to be compared for equality with this RenderingHints. **Returns:**true if the specified Object is equal to this RenderingHints.**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 value for this RenderingHints. The hash code of a RenderingHints is defined to be the sum of the hashCodes of each Entry in the RenderingHints object's entrySet view. This ensures that t1.equals(t2) implies that t1.hashCode()==t2.hashCode() for any two Map objects t1 and t2, as required by the general contract of Object.hashCode.

**Specified by:**[hashCode](http://docs.google.com/java/util/Map.html#hashCode()) in interface [Map](http://docs.google.com/java/util/Map.html)<[Object](http://docs.google.com/java/lang/Object.html),[Object](http://docs.google.com/java/lang/Object.html)>**Overrides:**[hashCode](http://docs.google.com/java/lang/Object.html#hashCode()) in class [Object](http://docs.google.com/java/lang/Object.html) **Returns:**the hash code value for this RenderingHints.**See Also:**[Map.Entry.hashCode()](http://docs.google.com/java/util/Map.Entry.html#hashCode()), [Object.hashCode()](http://docs.google.com/java/lang/Object.html#hashCode()), [Object.equals(Object)](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)), [equals(Object)](http://docs.google.com/java/awt/RenderingHints.html#equals(java.lang.Object))

### clone

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

Creates a clone of this RenderingHints object that has the same contents as this RenderingHints object.

**Overrides:**[clone](http://docs.google.com/java/lang/Object.html#clone()) in class [Object](http://docs.google.com/java/lang/Object.html) **Returns:**a clone of this instance.**See Also:**[Cloneable](http://docs.google.com/java/lang/Cloneable.html)

### toString

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

Returns a rather long string representation of the hashmap which contains the mappings of keys to values for this RenderingHints object.

**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 representation of this 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/RenderingHints.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/Rectangle.html)   [**NEXT CLASS**](http://docs.google.com/java/awt/RenderingHints.Key.html) | [**FRAMES**](http://docs.google.com/index.html?java/awt/RenderingHints.html)    [**NO FRAMES**](http://docs.google.com/RenderingHints.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#3znysh7) | [FIELD](#tyjcwt) | [CONSTR](#3dy6vkm) | [METHOD](#1t3h5sf) | DETAIL: [FIELD](#2s8eyo1) | [CONSTR](#111kx3o) | [METHOD](#4k668n3) |

[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).