| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/ImageReadParam.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/javax/imageio/ImageReader.html)   [**NEXT CLASS**](http://docs.google.com/javax/imageio/ImageTranscoder.html) | [**FRAMES**](http://docs.google.com/index.html?javax/imageio/ImageReadParam.html)    [**NO FRAMES**](http://docs.google.com/ImageReadParam.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | [CONSTR](#tyjcwt) | [METHOD](#3dy6vkm) | DETAIL: [FIELD](#2s8eyo1) | [CONSTR](#44sinio) | [METHOD](#z337ya) |

## **javax.imageio**

Class ImageReadParam

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 [javax.imageio.IIOParam](http://docs.google.com/javax/imageio/IIOParam.html)  
 **javax.imageio.ImageReadParam**

**Direct Known Subclasses:** [JPEGImageReadParam](http://docs.google.com/javax/imageio/plugins/jpeg/JPEGImageReadParam.html)

public class **ImageReadParam**extends [IIOParam](http://docs.google.com/javax/imageio/IIOParam.html)

A class describing how a stream is to be decoded. Instances of this class or its subclasses are used to supply prescriptive "how-to" information to instances of ImageReader.

An image encoded as part of a file or stream may be thought of extending out in multiple dimensions: the spatial dimensions of width and height, a number of bands, and a number of progressive decoding passes. This class allows a contiguous (hyper)rectangular subarea of the image in all of these dimensions to be selected for decoding. Additionally, the spatial dimensions may be subsampled discontinuously. Finally, color and format conversions may be specified by controlling the ColorModel and SampleModel of the destination image, either by providing a BufferedImage or by using an ImageTypeSpecifier.

An ImageReadParam object is used to specify how an image, or a set of images, will be converted on input from a stream in the context of the Java Image I/O framework. A plug-in for a specific image format will return instances of ImageReadParam from the getDefaultReadParam method of its ImageReader implementation.

The state maintained by an instance of ImageReadParam is independent of any particular image being decoded. When actual decoding takes place, the values set in the read param are combined with the actual properties of the image being decoded from the stream and the destination BufferedImage that will receive the decoded pixel data. For example, the source region set using setSourceRegion will first be intersected with the actual valid source area. The result will be translated by the value returned by getDestinationOffset, and the resulting rectangle intersected with the actual valid destination area to yield the destination area that will be written.

The parameters specified by an ImageReadParam are applied to an image as follows. First, if a rendering size has been set by setSourceRenderSize, the entire decoded image is rendered at the size given by getSourceRenderSize. Otherwise, the image has its natural size given by ImageReader.getWidth and ImageReader.getHeight.

Next, the image is clipped against the source region specified by getSourceXOffset, getSourceYOffset, getSourceWidth, and getSourceHeight.

The resulting region is then subsampled according to the factors given in [IIOParam.setSourceSubsampling](http://docs.google.com/javax/imageio/IIOParam.html#setSourceSubsampling(int,%20int,%20int,%20int)). The first pixel, the number of pixels per row, and the number of rows all depend on the subsampling settings. Call the minimum X and Y coordinates of the resulting rectangle (minX, minY), its width w and its height h.

This rectangle is offset by (getDestinationOffset().x, getDestinationOffset().y) and clipped against the destination bounds. If no destination image has been set, the destination is defined to have a width of getDestinationOffset().x + w, and a height of getDestinationOffset().y + h so that all pixels of the source region may be written to the destination.

Pixels that land, after subsampling, within the destination image, and that are written in one of the progressive passes specified by getSourceMinProgressivePass and getSourceNumProgressivePasses are passed along to the next step.

Finally, the source samples of each pixel are mapped into destination bands according to the algorithm described in the comment for setDestinationBands.

Plug-in writers may extend the functionality of ImageReadParam by providing a subclass that implements additional, plug-in specific interfaces. It is up to the plug-in to document what interfaces are available and how they are to be used. Readers will silently ignore any extended features of an ImageReadParam subclass of which they are not aware. Also, they may ignore any optional features that they normally disable when creating their own ImageReadParam instances via getDefaultReadParam.

Note that unless a query method exists for a capability, it must be supported by all ImageReader implementations (*e.g.* source render size is optional, but subsampling must be supported).

**See Also:**[ImageReader](http://docs.google.com/javax/imageio/ImageReader.html), [ImageWriter](http://docs.google.com/javax/imageio/ImageWriter.html), [ImageWriteParam](http://docs.google.com/javax/imageio/ImageWriteParam.html)

| **Field Summary** | |
| --- | --- |
| protected  boolean | [**canSetSourceRenderSize**](http://docs.google.com/javax/imageio/ImageReadParam.html#canSetSourceRenderSize)            true if this ImageReadParam allows the source rendering dimensions to be set. |
| protected  [BufferedImage](http://docs.google.com/java/awt/image/BufferedImage.html) | [**destination**](http://docs.google.com/javax/imageio/ImageReadParam.html#destination)            The current destination BufferedImage, or null if none has been set. |
| protected  int[] | [**destinationBands**](http://docs.google.com/javax/imageio/ImageReadParam.html#destinationBands)            The set of destination bands to be used, as an array of ints. |
| protected  int | [**minProgressivePass**](http://docs.google.com/javax/imageio/ImageReadParam.html#minProgressivePass)            The minimum index of a progressive pass to read from the source. |
| protected  int | [**numProgressivePasses**](http://docs.google.com/javax/imageio/ImageReadParam.html#numProgressivePasses)            The maximum number of progressive passes to read from the source. |
| protected  [Dimension](http://docs.google.com/java/awt/Dimension.html) | [**sourceRenderSize**](http://docs.google.com/javax/imageio/ImageReadParam.html#sourceRenderSize)            The desired rendering width and height of the source, if canSetSourceRenderSize is true, or null. |

| **Fields inherited from class javax.imageio.**[**IIOParam**](http://docs.google.com/javax/imageio/IIOParam.html) |
| --- |
| [controller](http://docs.google.com/javax/imageio/IIOParam.html#controller), [defaultController](http://docs.google.com/javax/imageio/IIOParam.html#defaultController), [destinationOffset](http://docs.google.com/javax/imageio/IIOParam.html#destinationOffset), [destinationType](http://docs.google.com/javax/imageio/IIOParam.html#destinationType), [sourceBands](http://docs.google.com/javax/imageio/IIOParam.html#sourceBands), [sourceRegion](http://docs.google.com/javax/imageio/IIOParam.html#sourceRegion), [sourceXSubsampling](http://docs.google.com/javax/imageio/IIOParam.html#sourceXSubsampling), [sourceYSubsampling](http://docs.google.com/javax/imageio/IIOParam.html#sourceYSubsampling), [subsamplingXOffset](http://docs.google.com/javax/imageio/IIOParam.html#subsamplingXOffset), [subsamplingYOffset](http://docs.google.com/javax/imageio/IIOParam.html#subsamplingYOffset) |

| **Constructor Summary** | |
| --- | --- |
| [**ImageReadParam**](http://docs.google.com/javax/imageio/ImageReadParam.html#ImageReadParam())()            Constructs an ImageReadParam. |

| **Method Summary** | |
| --- | --- |
| boolean | [**canSetSourceRenderSize**](http://docs.google.com/javax/imageio/ImageReadParam.html#canSetSourceRenderSize())()            Returns true if this reader allows the source image to be rendered at an arbitrary size as part of the decoding process, by means of the setSourceRenderSize method. |
| [BufferedImage](http://docs.google.com/java/awt/image/BufferedImage.html) | [**getDestination**](http://docs.google.com/javax/imageio/ImageReadParam.html#getDestination())()            Returns the BufferedImage currently set by the setDestination method, or null if none is set. |
| int[] | [**getDestinationBands**](http://docs.google.com/javax/imageio/ImageReadParam.html#getDestinationBands())()            Returns the set of band indices where data will be placed. |
| int | [**getSourceMaxProgressivePass**](http://docs.google.com/javax/imageio/ImageReadParam.html#getSourceMaxProgressivePass())()            If getSourceNumProgressivePasses is equal to Integer.MAX\_VALUE, returns Integer.MAX\_VALUE. |
| int | [**getSourceMinProgressivePass**](http://docs.google.com/javax/imageio/ImageReadParam.html#getSourceMinProgressivePass())()            Returns the index of the first progressive pass that will be decoded. |
| int | [**getSourceNumProgressivePasses**](http://docs.google.com/javax/imageio/ImageReadParam.html#getSourceNumProgressivePasses())()            Returns the number of the progressive passes that will be decoded. |
| [Dimension](http://docs.google.com/java/awt/Dimension.html) | [**getSourceRenderSize**](http://docs.google.com/javax/imageio/ImageReadParam.html#getSourceRenderSize())()            Returns the width and height of the source image as it will be rendered during decoding, if they have been set via the setSourceRenderSize method. |
| void | [**setDestination**](http://docs.google.com/javax/imageio/ImageReadParam.html#setDestination(java.awt.image.BufferedImage))([BufferedImage](http://docs.google.com/java/awt/image/BufferedImage.html) destination)            Supplies a BufferedImage to be used as the destination for decoded pixel data. |
| void | [**setDestinationBands**](http://docs.google.com/javax/imageio/ImageReadParam.html#setDestinationBands(int%5B%5D))(int[] destinationBands)            Sets the indices of the destination bands where data will be placed. |
| void | [**setDestinationType**](http://docs.google.com/javax/imageio/ImageReadParam.html#setDestinationType(javax.imageio.ImageTypeSpecifier))([ImageTypeSpecifier](http://docs.google.com/javax/imageio/ImageTypeSpecifier.html) destinationType)            Sets the desired image type for the destination image, using an ImageTypeSpecifier. |
| void | [**setSourceProgressivePasses**](http://docs.google.com/javax/imageio/ImageReadParam.html#setSourceProgressivePasses(int,%20int))(int minPass, int numPasses)            Sets the range of progressive passes that will be decoded. |
| void | [**setSourceRenderSize**](http://docs.google.com/javax/imageio/ImageReadParam.html#setSourceRenderSize(java.awt.Dimension))([Dimension](http://docs.google.com/java/awt/Dimension.html) size)            If the image is able to be rendered at an arbitrary size, sets the source width and height to the supplied values. |

| **Methods inherited from class javax.imageio.**[**IIOParam**](http://docs.google.com/javax/imageio/IIOParam.html) |
| --- |
| [activateController](http://docs.google.com/javax/imageio/IIOParam.html#activateController()), [getController](http://docs.google.com/javax/imageio/IIOParam.html#getController()), [getDefaultController](http://docs.google.com/javax/imageio/IIOParam.html#getDefaultController()), [getDestinationOffset](http://docs.google.com/javax/imageio/IIOParam.html#getDestinationOffset()), [getDestinationType](http://docs.google.com/javax/imageio/IIOParam.html#getDestinationType()), [getSourceBands](http://docs.google.com/javax/imageio/IIOParam.html#getSourceBands()), [getSourceRegion](http://docs.google.com/javax/imageio/IIOParam.html#getSourceRegion()), [getSourceXSubsampling](http://docs.google.com/javax/imageio/IIOParam.html#getSourceXSubsampling()), [getSourceYSubsampling](http://docs.google.com/javax/imageio/IIOParam.html#getSourceYSubsampling()), [getSubsamplingXOffset](http://docs.google.com/javax/imageio/IIOParam.html#getSubsamplingXOffset()), [getSubsamplingYOffset](http://docs.google.com/javax/imageio/IIOParam.html#getSubsamplingYOffset()), [hasController](http://docs.google.com/javax/imageio/IIOParam.html#hasController()), [setController](http://docs.google.com/javax/imageio/IIOParam.html#setController(javax.imageio.IIOParamController)), [setDestinationOffset](http://docs.google.com/javax/imageio/IIOParam.html#setDestinationOffset(java.awt.Point)), [setSourceBands](http://docs.google.com/javax/imageio/IIOParam.html#setSourceBands(int%5B%5D)), [setSourceRegion](http://docs.google.com/javax/imageio/IIOParam.html#setSourceRegion(java.awt.Rectangle)), [setSourceSubsampling](http://docs.google.com/javax/imageio/IIOParam.html#setSourceSubsampling(int,%20int,%20int,%20int)) |

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

### canSetSourceRenderSize

protected boolean **canSetSourceRenderSize**

true if this ImageReadParam allows the source rendering dimensions to be set. By default, the value is false. Subclasses must set this value manually.

ImageReaders that do not support setting of the source render size should set this value to false.

### sourceRenderSize

protected [Dimension](http://docs.google.com/java/awt/Dimension.html) **sourceRenderSize**

The desired rendering width and height of the source, if canSetSourceRenderSize is true, or null.

ImageReaders that do not support setting of the source render size may ignore this value.

### destination

protected [BufferedImage](http://docs.google.com/java/awt/image/BufferedImage.html) **destination**

The current destination BufferedImage, or null if none has been set. By default, the value is null.

### destinationBands

protected int[] **destinationBands**

The set of destination bands to be used, as an array of ints. By default, the value is null, indicating all destination bands should be written in order.

### minProgressivePass

protected int **minProgressivePass**

The minimum index of a progressive pass to read from the source. By default, the value is set to 0, which indicates that passes starting with the first available pass should be decoded.

Subclasses should ensure that this value is non-negative.

### numProgressivePasses

protected int **numProgressivePasses**

The maximum number of progressive passes to read from the source. By default, the value is set to Integer.MAX\_VALUE, which indicates that passes up to and including the last available pass should be decoded.

Subclasses should ensure that this value is positive. Additionally, if the value is not Integer.MAX\_VALUE, then minProgressivePass + numProgressivePasses - 1 should not exceed Integer.MAX\_VALUE.

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

### ImageReadParam

public **ImageReadParam**()

Constructs an ImageReadParam.

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

### setDestinationType

public void **setDestinationType**([ImageTypeSpecifier](http://docs.google.com/javax/imageio/ImageTypeSpecifier.html) destinationType)

**Description copied from class:** [**IIOParam**](http://docs.google.com/javax/imageio/IIOParam.html#setDestinationType(javax.imageio.ImageTypeSpecifier)) Sets the desired image type for the destination image, using an ImageTypeSpecifier.

When reading, if the layout of the destination has been set using this method, each call to an ImageReader read method will return a new BufferedImage using the format specified by the supplied type specifier. As a side effect, any destination BufferedImage set by ImageReadParam.setDestination(BufferedImage) will no longer be set as the destination. In other words, this method may be thought of as calling setDestination((BufferedImage)null).

When writing, the destination type maybe used to determine the color type of the image. The SampleModel information will be ignored, and may be null. For example, a 4-banded image could represent either CMYK or RGBA data. If a destination type is set, its ColorModel will override any ColorModel on the image itself. This is crucial when setSourceBands is used since the image's ColorModel will refer to the entire image rather than to the subset of bands being written.

**Overrides:**[setDestinationType](http://docs.google.com/javax/imageio/IIOParam.html#setDestinationType(javax.imageio.ImageTypeSpecifier)) in class [IIOParam](http://docs.google.com/javax/imageio/IIOParam.html) **Parameters:**destinationType - the ImageTypeSpecifier to be used to determine the destination layout and color type.**See Also:**[IIOParam.getDestinationType()](http://docs.google.com/javax/imageio/IIOParam.html#getDestinationType())

### setDestination

public void **setDestination**([BufferedImage](http://docs.google.com/java/awt/image/BufferedImage.html) destination)

Supplies a BufferedImage to be used as the destination for decoded pixel data. The currently set image will be written to by the read, readAll, and readRaster methods, and a reference to it will be returned by those methods.

Pixel data from the aforementioned methods will be written starting at the offset specified by getDestinationOffset.

If destination is null, a newly-created BufferedImage will be returned by those methods.

At the time of reading, the image is checked to verify that its ColorModel and SampleModel correspond to one of the ImageTypeSpecifiers returned from the ImageReader's getImageTypes method. If it does not, the reader will throw an IIOException.

**Parameters:**destination - the BufferedImage to be written to, or null.**See Also:**[getDestination()](http://docs.google.com/javax/imageio/ImageReadParam.html#getDestination())

### getDestination

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

Returns the BufferedImage currently set by the setDestination method, or null if none is set.

**Returns:**the BufferedImage to be written to.**See Also:**[setDestination(java.awt.image.BufferedImage)](http://docs.google.com/javax/imageio/ImageReadParam.html#setDestination(java.awt.image.BufferedImage))

### setDestinationBands

public void **setDestinationBands**(int[] destinationBands)

Sets the indices of the destination bands where data will be placed. Duplicate indices are not allowed.

A null value indicates that all destination bands will be used.

Choosing a destination band subset will not affect the number of bands in the output image of a read if no destination image is specified; the created destination image will still have the same number of bands as if this method had never been called. If a different number of bands in the destination image is desired, an image must be supplied using the ImageReadParam.setDestination method.

At the time of reading or writing, an IllegalArgumentException will be thrown by the reader or writer if a value larger than the largest destination band index has been specified, or if the number of source bands and destination bands to be used differ. The ImageReader.checkReadParamBandSettings method may be used to automate this test.

**Parameters:**destinationBands - an array of integer band indices to be used. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if destinationBands contains a negative or duplicate value.**See Also:**[getDestinationBands()](http://docs.google.com/javax/imageio/ImageReadParam.html#getDestinationBands()), [IIOParam.getSourceBands()](http://docs.google.com/javax/imageio/IIOParam.html#getSourceBands()), [ImageReader.checkReadParamBandSettings(javax.imageio.ImageReadParam, int, int)](http://docs.google.com/javax/imageio/ImageReader.html#checkReadParamBandSettings(javax.imageio.ImageReadParam,%20int,%20int))

### getDestinationBands

public int[] **getDestinationBands**()

Returns the set of band indices where data will be placed. If no value has been set, null is returned to indicate that all destination bands will be used.

**Returns:**the indices of the destination bands to be used, or null.**See Also:**[setDestinationBands(int[])](http://docs.google.com/javax/imageio/ImageReadParam.html#setDestinationBands(int%5B%5D))

### canSetSourceRenderSize

public boolean **canSetSourceRenderSize**()

Returns true if this reader allows the source image to be rendered at an arbitrary size as part of the decoding process, by means of the setSourceRenderSize method. If this method returns false, calls to setSourceRenderSize will throw an UnsupportedOperationException.

**Returns:**true if setting source rendering size is supported.**See Also:**[setSourceRenderSize(java.awt.Dimension)](http://docs.google.com/javax/imageio/ImageReadParam.html#setSourceRenderSize(java.awt.Dimension))

### setSourceRenderSize

public void **setSourceRenderSize**([Dimension](http://docs.google.com/java/awt/Dimension.html) size)  
 throws [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html)

If the image is able to be rendered at an arbitrary size, sets the source width and height to the supplied values. Note that the values returned from the getWidth and getHeight methods on ImageReader are not affected by this method; they will continue to return the default size for the image. Similarly, if the image is also tiled the tile width and height are given in terms of the default size.

Typically, the width and height should be chosen such that the ratio of width to height closely approximates the aspect ratio of the image, as returned from ImageReader.getAspectRatio.

If this plug-in does not allow the rendering size to be set, an UnsupportedOperationException will be thrown.

To remove the render size setting, pass in a value of null for size.

**Parameters:**size - a Dimension indicating the desired width and height. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if either the width or the height is negative or 0. [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if image resizing is not supported by this plug-in.**See Also:**[getSourceRenderSize()](http://docs.google.com/javax/imageio/ImageReadParam.html#getSourceRenderSize()), [ImageReader.getWidth(int)](http://docs.google.com/javax/imageio/ImageReader.html#getWidth(int)), [ImageReader.getHeight(int)](http://docs.google.com/javax/imageio/ImageReader.html#getHeight(int)), [ImageReader.getAspectRatio(int)](http://docs.google.com/javax/imageio/ImageReader.html#getAspectRatio(int))

### getSourceRenderSize

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

Returns the width and height of the source image as it will be rendered during decoding, if they have been set via the setSourceRenderSize method. A nullvalue indicates that no setting has been made.

**Returns:**the rendered width and height of the source image as a Dimension.**See Also:**[setSourceRenderSize(java.awt.Dimension)](http://docs.google.com/javax/imageio/ImageReadParam.html#setSourceRenderSize(java.awt.Dimension))

### setSourceProgressivePasses

public void **setSourceProgressivePasses**(int minPass,  
 int numPasses)

Sets the range of progressive passes that will be decoded. Passes outside of this range will be ignored.

A progressive pass is a re-encoding of the entire image, generally at progressively higher effective resolutions, but requiring greater transmission bandwidth. The most common use of progressive encoding is found in the JPEG format, where successive passes include more detailed representations of the high-frequency image content.

The actual number of passes to be decoded is determined during decoding, based on the number of actual passes available in the stream. Thus if minPass + numPasses - 1 is larger than the index of the last available passes, decoding will end with that pass.

A value of numPasses of Integer.MAX\_VALUE indicates that all passes from minPass forward should be read. Otherwise, the index of the last pass (*i.e.*, minPass + numPasses - 1) must not exceed Integer.MAX\_VALUE.

There is no unsetSourceProgressivePasses method; the same effect may be obtained by calling setSourceProgressivePasses(0, Integer.MAX\_VALUE).

**Parameters:**minPass - the index of the first pass to be decoded.numPasses - the maximum number of passes to be decoded. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if minPass is negative, numPasses is negative or 0, or numPasses is smaller than Integer.MAX\_VALUE but minPass + numPasses - 1 is greater than INTEGER.MAX\_VALUE.**See Also:**[getSourceMinProgressivePass()](http://docs.google.com/javax/imageio/ImageReadParam.html#getSourceMinProgressivePass()), [getSourceMaxProgressivePass()](http://docs.google.com/javax/imageio/ImageReadParam.html#getSourceMaxProgressivePass())

### getSourceMinProgressivePass

public int **getSourceMinProgressivePass**()

Returns the index of the first progressive pass that will be decoded. If no value has been set, 0 will be returned (which is the correct value).

**Returns:**the index of the first pass that will be decoded.**See Also:**[setSourceProgressivePasses(int, int)](http://docs.google.com/javax/imageio/ImageReadParam.html#setSourceProgressivePasses(int,%20int)), [getSourceNumProgressivePasses()](http://docs.google.com/javax/imageio/ImageReadParam.html#getSourceNumProgressivePasses())

### getSourceMaxProgressivePass

public int **getSourceMaxProgressivePass**()

If getSourceNumProgressivePasses is equal to Integer.MAX\_VALUE, returns Integer.MAX\_VALUE. Otherwise, returns getSourceMinProgressivePass() + getSourceNumProgressivePasses() - 1.

**Returns:**the index of the last pass to be read, or Integer.MAX\_VALUE.

### getSourceNumProgressivePasses

public int **getSourceNumProgressivePasses**()

Returns the number of the progressive passes that will be decoded. If no value has been set, Integer.MAX\_VALUE will be returned (which is the correct value).

**Returns:**the number of the passes that will be decoded.**See Also:**[setSourceProgressivePasses(int, int)](http://docs.google.com/javax/imageio/ImageReadParam.html#setSourceProgressivePasses(int,%20int)), [getSourceMinProgressivePass()](http://docs.google.com/javax/imageio/ImageReadParam.html#getSourceMinProgressivePass())

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/ImageReadParam.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/javax/imageio/ImageReader.html)   [**NEXT CLASS**](http://docs.google.com/javax/imageio/ImageTranscoder.html) | [**FRAMES**](http://docs.google.com/index.html?javax/imageio/ImageReadParam.html)    [**NO FRAMES**](http://docs.google.com/ImageReadParam.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | [CONSTR](#tyjcwt) | [METHOD](#3dy6vkm) | DETAIL: [FIELD](#2s8eyo1) | [CONSTR](#44sinio) | [METHOD](#z337ya) |

[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).