| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/ImageReader.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/ImageIO.html)   [**NEXT CLASS**](http://docs.google.com/javax/imageio/ImageReadParam.html) | [**FRAMES**](http://docs.google.com/index.html?javax/imageio/ImageReader.html)    [**NO FRAMES**](http://docs.google.com/ImageReader.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | [CONSTR](#2et92p0) | [METHOD](#tyjcwt) | DETAIL: [FIELD](#1t3h5sf) | [CONSTR](#3j2qqm3) | [METHOD](#4i7ojhp) |

## **javax.imageio**

Class ImageReader

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

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

An abstract superclass for parsing and decoding of images. This class must be subclassed by classes that read in images in the context of the Java Image I/O framework.

ImageReader objects are normally instantiated by the service provider interface (SPI) class for the specific format. Service provider classes (e.g., instances of ImageReaderSpi) are registered with the IIORegistry, which uses them for format recognition and presentation of available format readers and writers.

When an input source is set (using the setInput method), it may be marked as "seek forward only". This setting means that images contained within the input source will only be read in order, possibly allowing the reader to avoid caching portions of the input containing data associated with images that have been read previously.

**See Also:**[ImageWriter](http://docs.google.com/javax/imageio/ImageWriter.html), [IIORegistry](http://docs.google.com/javax/imageio/spi/IIORegistry.html), [ImageReaderSpi](http://docs.google.com/javax/imageio/spi/ImageReaderSpi.html)

| **Field Summary** | |
| --- | --- |
| protected  [Locale](http://docs.google.com/java/util/Locale.html)[] | [**availableLocales**](http://docs.google.com/javax/imageio/ImageReader.html#availableLocales)            An array of Locales which may be used to localize warning messages, or null if localization is not supported. |
| protected  boolean | [**ignoreMetadata**](http://docs.google.com/javax/imageio/ImageReader.html#ignoreMetadata)            true if the current input source has been marked as allowing metadata to be ignored by setInput. |
| protected  [Object](http://docs.google.com/java/lang/Object.html) | [**input**](http://docs.google.com/javax/imageio/ImageReader.html#input)            The ImageInputStream or other Object by setInput and retrieved by getInput. |
| protected  [Locale](http://docs.google.com/java/util/Locale.html) | [**locale**](http://docs.google.com/javax/imageio/ImageReader.html#locale)            The current Locale to be used for localization, or null if none has been set. |
| protected  int | [**minIndex**](http://docs.google.com/javax/imageio/ImageReader.html#minIndex)            The smallest valid index for reading, initially 0. |
| protected  [ImageReaderSpi](http://docs.google.com/javax/imageio/spi/ImageReaderSpi.html) | [**originatingProvider**](http://docs.google.com/javax/imageio/ImageReader.html#originatingProvider)            The ImageReaderSpi that instantiated this object, or null if its identity is not known or none exists. |
| protected  [List](http://docs.google.com/java/util/List.html)<[IIOReadProgressListener](http://docs.google.com/javax/imageio/event/IIOReadProgressListener.html)> | [**progressListeners**](http://docs.google.com/javax/imageio/ImageReader.html#progressListeners)            A List of currently registered IIOReadProgressListeners, initialized by default to null, which is synonymous with an empty List. |
| protected  boolean | [**seekForwardOnly**](http://docs.google.com/javax/imageio/ImageReader.html#seekForwardOnly)            true if the current input source has been marked as allowing only forward seeking by setInput. |
| protected  [List](http://docs.google.com/java/util/List.html)<[IIOReadUpdateListener](http://docs.google.com/javax/imageio/event/IIOReadUpdateListener.html)> | [**updateListeners**](http://docs.google.com/javax/imageio/ImageReader.html#updateListeners)            A List of currently registered IIOReadUpdateListeners, initialized by default to null, which is synonymous with an empty List. |
| protected  [List](http://docs.google.com/java/util/List.html)<[IIOReadWarningListener](http://docs.google.com/javax/imageio/event/IIOReadWarningListener.html)> | [**warningListeners**](http://docs.google.com/javax/imageio/ImageReader.html#warningListeners)            A List of currently registered IIOReadWarningListeners, initialized by default to null, which is synonymous with an empty List. |
| protected  [List](http://docs.google.com/java/util/List.html)<[Locale](http://docs.google.com/java/util/Locale.html)> | [**warningLocales**](http://docs.google.com/javax/imageio/ImageReader.html#warningLocales)            A List of the Locales associated with each currently registered IIOReadWarningListener, initialized by default to null, which is synonymous with an empty List. |

| **Constructor Summary** | |
| --- | --- |
| protected | [**ImageReader**](http://docs.google.com/javax/imageio/ImageReader.html#ImageReader(javax.imageio.spi.ImageReaderSpi))([ImageReaderSpi](http://docs.google.com/javax/imageio/spi/ImageReaderSpi.html) originatingProvider)            Constructs an ImageReader and sets its originatingProvider field to the supplied value. |

| **Method Summary** | |
| --- | --- |
| void | [**abort**](http://docs.google.com/javax/imageio/ImageReader.html#abort())()            Requests that any current read operation be aborted. |
| protected  boolean | [**abortRequested**](http://docs.google.com/javax/imageio/ImageReader.html#abortRequested())()            Returns true if a request to abort the current read operation has been made since the reader was instantiated or clearAbortRequest was called. |
| void | [**addIIOReadProgressListener**](http://docs.google.com/javax/imageio/ImageReader.html#addIIOReadProgressListener(javax.imageio.event.IIOReadProgressListener))([IIOReadProgressListener](http://docs.google.com/javax/imageio/event/IIOReadProgressListener.html) listener)            Adds an IIOReadProgressListener to the list of registered progress listeners. |
| void | [**addIIOReadUpdateListener**](http://docs.google.com/javax/imageio/ImageReader.html#addIIOReadUpdateListener(javax.imageio.event.IIOReadUpdateListener))([IIOReadUpdateListener](http://docs.google.com/javax/imageio/event/IIOReadUpdateListener.html) listener)            Adds an IIOReadUpdateListener to the list of registered update listeners. |
| void | [**addIIOReadWarningListener**](http://docs.google.com/javax/imageio/ImageReader.html#addIIOReadWarningListener(javax.imageio.event.IIOReadWarningListener))([IIOReadWarningListener](http://docs.google.com/javax/imageio/event/IIOReadWarningListener.html) listener)            Adds an IIOReadWarningListener to the list of registered warning listeners. |
| boolean | [**canReadRaster**](http://docs.google.com/javax/imageio/ImageReader.html#canReadRaster())()            Returns true if this plug-in supports reading just a [Raster](http://docs.google.com/java/awt/image/Raster.html) of pixel data. |
| protected static void | [**checkReadParamBandSettings**](http://docs.google.com/javax/imageio/ImageReader.html#checkReadParamBandSettings(javax.imageio.ImageReadParam,%20int,%20int))([ImageReadParam](http://docs.google.com/javax/imageio/ImageReadParam.html) param, int numSrcBands, int numDstBands)            A utility method that may be used by readers to test the validity of the source and destination band settings of an ImageReadParam. |
| protected  void | [**clearAbortRequest**](http://docs.google.com/javax/imageio/ImageReader.html#clearAbortRequest())()            Clears any previous abort request. |
| protected static void | [**computeRegions**](http://docs.google.com/javax/imageio/ImageReader.html#computeRegions(javax.imageio.ImageReadParam,%20int,%20int,%20java.awt.image.BufferedImage,%20java.awt.Rectangle,%20java.awt.Rectangle))([ImageReadParam](http://docs.google.com/javax/imageio/ImageReadParam.html) param, int srcWidth, int srcHeight, [BufferedImage](http://docs.google.com/java/awt/image/BufferedImage.html) image, [Rectangle](http://docs.google.com/java/awt/Rectangle.html) srcRegion, [Rectangle](http://docs.google.com/java/awt/Rectangle.html) destRegion)            Computes the source region of interest and the destination region of interest, taking the width and height of the source image, an optional destination image, and an optional ImageReadParam into account. |
| void | [**dispose**](http://docs.google.com/javax/imageio/ImageReader.html#dispose())()            Allows any resources held by this object to be released. |
| float | [**getAspectRatio**](http://docs.google.com/javax/imageio/ImageReader.html#getAspectRatio(int))(int imageIndex)            Returns the aspect ratio of the given image (that is, its width divided by its height) as a float. |
| [Locale](http://docs.google.com/java/util/Locale.html)[] | [**getAvailableLocales**](http://docs.google.com/javax/imageio/ImageReader.html#getAvailableLocales())()            Returns an array of Locales that may be used to localize warning listeners and compression settings. |
| [ImageReadParam](http://docs.google.com/javax/imageio/ImageReadParam.html) | [**getDefaultReadParam**](http://docs.google.com/javax/imageio/ImageReader.html#getDefaultReadParam())()            Returns a default ImageReadParam object appropriate for this format. |
| protected static [BufferedImage](http://docs.google.com/java/awt/image/BufferedImage.html) | [**getDestination**](http://docs.google.com/javax/imageio/ImageReader.html#getDestination(javax.imageio.ImageReadParam,%20java.util.Iterator,%20int,%20int))([ImageReadParam](http://docs.google.com/javax/imageio/ImageReadParam.html) param, [Iterator](http://docs.google.com/java/util/Iterator.html)<[ImageTypeSpecifier](http://docs.google.com/javax/imageio/ImageTypeSpecifier.html)> imageTypes, int width, int height)            Returns the BufferedImage to which decoded pixel data should be written. |
| [String](http://docs.google.com/java/lang/String.html) | [**getFormatName**](http://docs.google.com/javax/imageio/ImageReader.html#getFormatName())()            Returns a String identifying the format of the input source. |
| abstract  int | [**getHeight**](http://docs.google.com/javax/imageio/ImageReader.html#getHeight(int))(int imageIndex)            Returns the height in pixels of the given image within the input source. |
| abstract  [IIOMetadata](http://docs.google.com/javax/imageio/metadata/IIOMetadata.html) | [**getImageMetadata**](http://docs.google.com/javax/imageio/ImageReader.html#getImageMetadata(int))(int imageIndex)            Returns an IIOMetadata object containing metadata associated with the given image, or null if the reader does not support reading metadata, is set to ignore metadata, or if no metadata is available. |
| [IIOMetadata](http://docs.google.com/javax/imageio/metadata/IIOMetadata.html) | [**getImageMetadata**](http://docs.google.com/javax/imageio/ImageReader.html#getImageMetadata(int,%20java.lang.String,%20java.util.Set))(int imageIndex, [String](http://docs.google.com/java/lang/String.html) formatName, [Set](http://docs.google.com/java/util/Set.html)<[String](http://docs.google.com/java/lang/String.html)> nodeNames)            Returns an IIOMetadata object representing the metadata associated with the given image, or null if the reader does not support reading metadata or none is available. |
| abstract  [Iterator](http://docs.google.com/java/util/Iterator.html)<[ImageTypeSpecifier](http://docs.google.com/javax/imageio/ImageTypeSpecifier.html)> | [**getImageTypes**](http://docs.google.com/javax/imageio/ImageReader.html#getImageTypes(int))(int imageIndex)            Returns an Iterator containing possible image types to which the given image may be decoded, in the form of ImageTypeSpecifierss. |
| [Object](http://docs.google.com/java/lang/Object.html) | [**getInput**](http://docs.google.com/javax/imageio/ImageReader.html#getInput())()            Returns the ImageInputStream or other Object previously set as the input source. |
| [Locale](http://docs.google.com/java/util/Locale.html) | [**getLocale**](http://docs.google.com/javax/imageio/ImageReader.html#getLocale())()            Returns the currently set Locale, or null if none has been set. |
| int | [**getMinIndex**](http://docs.google.com/javax/imageio/ImageReader.html#getMinIndex())()            Returns the lowest valid index for reading an image, thumbnail, or image metadata. |
| abstract  int | [**getNumImages**](http://docs.google.com/javax/imageio/ImageReader.html#getNumImages(boolean))(boolean allowSearch)            Returns the number of images, not including thumbnails, available from the current input source. |
| int | [**getNumThumbnails**](http://docs.google.com/javax/imageio/ImageReader.html#getNumThumbnails(int))(int imageIndex)            Returns the number of thumbnail preview images associated with the given image. |
| [ImageReaderSpi](http://docs.google.com/javax/imageio/spi/ImageReaderSpi.html) | [**getOriginatingProvider**](http://docs.google.com/javax/imageio/ImageReader.html#getOriginatingProvider())()            Returns the ImageReaderSpi that was passed in on the constructor. |
| [ImageTypeSpecifier](http://docs.google.com/javax/imageio/ImageTypeSpecifier.html) | [**getRawImageType**](http://docs.google.com/javax/imageio/ImageReader.html#getRawImageType(int))(int imageIndex)            Returns an ImageTypeSpecifier indicating the SampleModel and ColorModel which most closely represents the "raw" internal format of the image. |
| protected static [Rectangle](http://docs.google.com/java/awt/Rectangle.html) | [**getSourceRegion**](http://docs.google.com/javax/imageio/ImageReader.html#getSourceRegion(javax.imageio.ImageReadParam,%20int,%20int))([ImageReadParam](http://docs.google.com/javax/imageio/ImageReadParam.html) param, int srcWidth, int srcHeight)            A utility method that may be used by readers to compute the region of the source image that should be read, taking into account any source region and subsampling offset settings in the supplied ImageReadParam. |
| abstract  [IIOMetadata](http://docs.google.com/javax/imageio/metadata/IIOMetadata.html) | [**getStreamMetadata**](http://docs.google.com/javax/imageio/ImageReader.html#getStreamMetadata())()            Returns an IIOMetadata object representing the metadata associated with the input source as a whole (i.e., not associated with any particular image), or null if the reader does not support reading metadata, is set to ignore metadata, or if no metadata is available. |
| [IIOMetadata](http://docs.google.com/javax/imageio/metadata/IIOMetadata.html) | [**getStreamMetadata**](http://docs.google.com/javax/imageio/ImageReader.html#getStreamMetadata(java.lang.String,%20java.util.Set))([String](http://docs.google.com/java/lang/String.html) formatName, [Set](http://docs.google.com/java/util/Set.html)<[String](http://docs.google.com/java/lang/String.html)> nodeNames)            Returns an IIOMetadata object representing the metadata associated with the input source as a whole (i.e., not associated with any particular image). |
| int | [**getThumbnailHeight**](http://docs.google.com/javax/imageio/ImageReader.html#getThumbnailHeight(int,%20int))(int imageIndex, int thumbnailIndex)            Returns the height of the thumbnail preview image indexed by thumbnailIndex, associated with the image indexed by ImageIndex. |
| int | [**getThumbnailWidth**](http://docs.google.com/javax/imageio/ImageReader.html#getThumbnailWidth(int,%20int))(int imageIndex, int thumbnailIndex)            Returns the width of the thumbnail preview image indexed by thumbnailIndex, associated with the image indexed by ImageIndex. |
| int | [**getTileGridXOffset**](http://docs.google.com/javax/imageio/ImageReader.html#getTileGridXOffset(int))(int imageIndex)            Returns the X coordinate of the upper-left corner of tile (0, 0) in the given image. |
| int | [**getTileGridYOffset**](http://docs.google.com/javax/imageio/ImageReader.html#getTileGridYOffset(int))(int imageIndex)            Returns the Y coordinate of the upper-left corner of tile (0, 0) in the given image. |
| int | [**getTileHeight**](http://docs.google.com/javax/imageio/ImageReader.html#getTileHeight(int))(int imageIndex)            Returns the height of a tile in the given image. |
| int | [**getTileWidth**](http://docs.google.com/javax/imageio/ImageReader.html#getTileWidth(int))(int imageIndex)            Returns the width of a tile in the given image. |
| abstract  int | [**getWidth**](http://docs.google.com/javax/imageio/ImageReader.html#getWidth(int))(int imageIndex)            Returns the width in pixels of the given image within the input source. |
| boolean | [**hasThumbnails**](http://docs.google.com/javax/imageio/ImageReader.html#hasThumbnails(int))(int imageIndex)            Returns true if the given image has thumbnail preview images associated with it. |
| boolean | [**isIgnoringMetadata**](http://docs.google.com/javax/imageio/ImageReader.html#isIgnoringMetadata())()            Returns true if the current input source has been marked as allowing metadata to be ignored by passing true as the ignoreMetadata argument to the setInput method. |
| boolean | [**isImageTiled**](http://docs.google.com/javax/imageio/ImageReader.html#isImageTiled(int))(int imageIndex)            Returns true if the image is organized into *tiles*, that is, equal-sized non-overlapping rectangles. |
| boolean | [**isRandomAccessEasy**](http://docs.google.com/javax/imageio/ImageReader.html#isRandomAccessEasy(int))(int imageIndex)            Returns true if the storage format of the given image places no inherent impediment on random access to pixels. |
| boolean | [**isSeekForwardOnly**](http://docs.google.com/javax/imageio/ImageReader.html#isSeekForwardOnly())()            Returns true if the current input source has been marked as seek forward only by passing true as the seekForwardOnly argument to the setInput method. |
| protected  void | [**processImageComplete**](http://docs.google.com/javax/imageio/ImageReader.html#processImageComplete())()            Broadcasts the completion of an image read to all registered IIOReadProgressListeners by calling their imageComplete method. |
| protected  void | [**processImageProgress**](http://docs.google.com/javax/imageio/ImageReader.html#processImageProgress(float))(float percentageDone)            Broadcasts the current percentage of image completion to all registered IIOReadProgressListeners by calling their imageProgress method. |
| protected  void | [**processImageStarted**](http://docs.google.com/javax/imageio/ImageReader.html#processImageStarted(int))(int imageIndex)            Broadcasts the start of an image read to all registered IIOReadProgressListeners by calling their imageStarted method. |
| protected  void | [**processImageUpdate**](http://docs.google.com/javax/imageio/ImageReader.html#processImageUpdate(java.awt.image.BufferedImage,%20int,%20int,%20int,%20int,%20int,%20int,%20int%5B%5D))([BufferedImage](http://docs.google.com/java/awt/image/BufferedImage.html) theImage, int minX, int minY, int width, int height, int periodX, int periodY, int[] bands)            Broadcasts the update of a set of samples to all registered IIOReadUpdateListeners by calling their imageUpdate method. |
| protected  void | [**processPassComplete**](http://docs.google.com/javax/imageio/ImageReader.html#processPassComplete(java.awt.image.BufferedImage))([BufferedImage](http://docs.google.com/java/awt/image/BufferedImage.html) theImage)            Broadcasts the end of a progressive pass to all registered IIOReadUpdateListeners by calling their passComplete method. |
| protected  void | [**processPassStarted**](http://docs.google.com/javax/imageio/ImageReader.html#processPassStarted(java.awt.image.BufferedImage,%20int,%20int,%20int,%20int,%20int,%20int,%20int,%20int%5B%5D))([BufferedImage](http://docs.google.com/java/awt/image/BufferedImage.html) theImage, int pass, int minPass, int maxPass, int minX, int minY, int periodX, int periodY, int[] bands)            Broadcasts the beginning of a progressive pass to all registered IIOReadUpdateListeners by calling their passStarted method. |
| protected  void | [**processReadAborted**](http://docs.google.com/javax/imageio/ImageReader.html#processReadAborted())()            Broadcasts that the read has been aborted to all registered IIOReadProgressListeners by calling their readAborted method. |
| protected  void | [**processSequenceComplete**](http://docs.google.com/javax/imageio/ImageReader.html#processSequenceComplete())()            Broadcasts the completion of an sequence of image reads to all registered IIOReadProgressListeners by calling their sequenceComplete method. |
| protected  void | [**processSequenceStarted**](http://docs.google.com/javax/imageio/ImageReader.html#processSequenceStarted(int))(int minIndex)            Broadcasts the start of an sequence of image reads to all registered IIOReadProgressListeners by calling their sequenceStarted method. |
| protected  void | [**processThumbnailComplete**](http://docs.google.com/javax/imageio/ImageReader.html#processThumbnailComplete())()            Broadcasts the completion of a thumbnail read to all registered IIOReadProgressListeners by calling their thumbnailComplete method. |
| protected  void | [**processThumbnailPassComplete**](http://docs.google.com/javax/imageio/ImageReader.html#processThumbnailPassComplete(java.awt.image.BufferedImage))([BufferedImage](http://docs.google.com/java/awt/image/BufferedImage.html) theThumbnail)            Broadcasts the end of a thumbnail progressive pass to all registered IIOReadUpdateListeners by calling their thumbnailPassComplete method. |
| protected  void | [**processThumbnailPassStarted**](http://docs.google.com/javax/imageio/ImageReader.html#processThumbnailPassStarted(java.awt.image.BufferedImage,%20int,%20int,%20int,%20int,%20int,%20int,%20int,%20int%5B%5D))([BufferedImage](http://docs.google.com/java/awt/image/BufferedImage.html) theThumbnail, int pass, int minPass, int maxPass, int minX, int minY, int periodX, int periodY, int[] bands)            Broadcasts the beginning of a thumbnail progressive pass to all registered IIOReadUpdateListeners by calling their thumbnailPassStarted method. |
| protected  void | [**processThumbnailProgress**](http://docs.google.com/javax/imageio/ImageReader.html#processThumbnailProgress(float))(float percentageDone)            Broadcasts the current percentage of thumbnail completion to all registered IIOReadProgressListeners by calling their thumbnailProgress method. |
| protected  void | [**processThumbnailStarted**](http://docs.google.com/javax/imageio/ImageReader.html#processThumbnailStarted(int,%20int))(int imageIndex, int thumbnailIndex)            Broadcasts the start of a thumbnail read to all registered IIOReadProgressListeners by calling their thumbnailStarted method. |
| protected  void | [**processThumbnailUpdate**](http://docs.google.com/javax/imageio/ImageReader.html#processThumbnailUpdate(java.awt.image.BufferedImage,%20int,%20int,%20int,%20int,%20int,%20int,%20int%5B%5D))([BufferedImage](http://docs.google.com/java/awt/image/BufferedImage.html) theThumbnail, int minX, int minY, int width, int height, int periodX, int periodY, int[] bands)            Broadcasts the update of a set of samples in a thumbnail image to all registered IIOReadUpdateListeners by calling their thumbnailUpdate method. |
| protected  void | [**processWarningOccurred**](http://docs.google.com/javax/imageio/ImageReader.html#processWarningOccurred(java.lang.String))([String](http://docs.google.com/java/lang/String.html) warning)            Broadcasts a warning message to all registered IIOReadWarningListeners by calling their warningOccurred method. |
| protected  void | [**processWarningOccurred**](http://docs.google.com/javax/imageio/ImageReader.html#processWarningOccurred(java.lang.String,%20java.lang.String))([String](http://docs.google.com/java/lang/String.html) baseName, [String](http://docs.google.com/java/lang/String.html) keyword)            Broadcasts a localized warning message to all registered IIOReadWarningListeners by calling their warningOccurred method with a string taken from a ResourceBundle. |
| [BufferedImage](http://docs.google.com/java/awt/image/BufferedImage.html) | [**read**](http://docs.google.com/javax/imageio/ImageReader.html#read(int))(int imageIndex)            Reads the image indexed by imageIndex and returns it as a complete BufferedImage, using a default ImageReadParam. |
| abstract  [BufferedImage](http://docs.google.com/java/awt/image/BufferedImage.html) | [**read**](http://docs.google.com/javax/imageio/ImageReader.html#read(int,%20javax.imageio.ImageReadParam))(int imageIndex, [ImageReadParam](http://docs.google.com/javax/imageio/ImageReadParam.html) param)            Reads the image indexed by imageIndex and returns it as a complete BufferedImage, using a supplied ImageReadParam. |
| [IIOImage](http://docs.google.com/javax/imageio/IIOImage.html) | [**readAll**](http://docs.google.com/javax/imageio/ImageReader.html#readAll(int,%20javax.imageio.ImageReadParam))(int imageIndex, [ImageReadParam](http://docs.google.com/javax/imageio/ImageReadParam.html) param)            Reads the image indexed by imageIndex and returns an IIOImage containing the image, thumbnails, and associated image metadata, using a supplied ImageReadParam. |
| [Iterator](http://docs.google.com/java/util/Iterator.html)<[IIOImage](http://docs.google.com/javax/imageio/IIOImage.html)> | [**readAll**](http://docs.google.com/javax/imageio/ImageReader.html#readAll(java.util.Iterator))([Iterator](http://docs.google.com/java/util/Iterator.html)<? extends [ImageReadParam](http://docs.google.com/javax/imageio/ImageReadParam.html)> params)            Returns an Iterator containing all the images, thumbnails, and metadata, starting at the index given by getMinIndex, from the input source in the form of IIOImage objects. |
| [RenderedImage](http://docs.google.com/java/awt/image/RenderedImage.html) | [**readAsRenderedImage**](http://docs.google.com/javax/imageio/ImageReader.html#readAsRenderedImage(int,%20javax.imageio.ImageReadParam))(int imageIndex, [ImageReadParam](http://docs.google.com/javax/imageio/ImageReadParam.html) param)            Returns a RenderedImage object that contains the contents of the image indexed by imageIndex. |
| boolean | [**readerSupportsThumbnails**](http://docs.google.com/javax/imageio/ImageReader.html#readerSupportsThumbnails())()            Returns true if the image format understood by this reader supports thumbnail preview images associated with it. |
| [Raster](http://docs.google.com/java/awt/image/Raster.html) | [**readRaster**](http://docs.google.com/javax/imageio/ImageReader.html#readRaster(int,%20javax.imageio.ImageReadParam))(int imageIndex, [ImageReadParam](http://docs.google.com/javax/imageio/ImageReadParam.html) param)            Returns a new Raster object containing the raw pixel data from the image stream, without any color conversion applied. |
| [BufferedImage](http://docs.google.com/java/awt/image/BufferedImage.html) | [**readThumbnail**](http://docs.google.com/javax/imageio/ImageReader.html#readThumbnail(int,%20int))(int imageIndex, int thumbnailIndex)            Returns the thumbnail preview image indexed by thumbnailIndex, associated with the image indexed by ImageIndex as a BufferedImage. |
| [BufferedImage](http://docs.google.com/java/awt/image/BufferedImage.html) | [**readTile**](http://docs.google.com/javax/imageio/ImageReader.html#readTile(int,%20int,%20int))(int imageIndex, int tileX, int tileY)            Reads the tile indicated by the tileX and tileY arguments, returning it as a BufferedImage. |
| [Raster](http://docs.google.com/java/awt/image/Raster.html) | [**readTileRaster**](http://docs.google.com/javax/imageio/ImageReader.html#readTileRaster(int,%20int,%20int))(int imageIndex, int tileX, int tileY)            Returns a new Raster object containing the raw pixel data from the tile, without any color conversion applied. |
| void | [**removeAllIIOReadProgressListeners**](http://docs.google.com/javax/imageio/ImageReader.html#removeAllIIOReadProgressListeners())()            Removes all currently registered IIOReadProgressListener objects. |
| void | [**removeAllIIOReadUpdateListeners**](http://docs.google.com/javax/imageio/ImageReader.html#removeAllIIOReadUpdateListeners())()            Removes all currently registered IIOReadUpdateListener objects. |
| void | [**removeAllIIOReadWarningListeners**](http://docs.google.com/javax/imageio/ImageReader.html#removeAllIIOReadWarningListeners())()            Removes all currently registered IIOReadWarningListener objects. |
| void | [**removeIIOReadProgressListener**](http://docs.google.com/javax/imageio/ImageReader.html#removeIIOReadProgressListener(javax.imageio.event.IIOReadProgressListener))([IIOReadProgressListener](http://docs.google.com/javax/imageio/event/IIOReadProgressListener.html) listener)            Removes an IIOReadProgressListener from the list of registered progress listeners. |
| void | [**removeIIOReadUpdateListener**](http://docs.google.com/javax/imageio/ImageReader.html#removeIIOReadUpdateListener(javax.imageio.event.IIOReadUpdateListener))([IIOReadUpdateListener](http://docs.google.com/javax/imageio/event/IIOReadUpdateListener.html) listener)            Removes an IIOReadUpdateListener from the list of registered update listeners. |
| void | [**removeIIOReadWarningListener**](http://docs.google.com/javax/imageio/ImageReader.html#removeIIOReadWarningListener(javax.imageio.event.IIOReadWarningListener))([IIOReadWarningListener](http://docs.google.com/javax/imageio/event/IIOReadWarningListener.html) listener)            Removes an IIOReadWarningListener from the list of registered error listeners. |
| void | [**reset**](http://docs.google.com/javax/imageio/ImageReader.html#reset())()            Restores the ImageReader to its initial state. |
| void | [**setInput**](http://docs.google.com/javax/imageio/ImageReader.html#setInput(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) input)            Sets the input source to use to the given ImageInputStream or other Object. |
| void | [**setInput**](http://docs.google.com/javax/imageio/ImageReader.html#setInput(java.lang.Object,%20boolean))([Object](http://docs.google.com/java/lang/Object.html) input, boolean seekForwardOnly)            Sets the input source to use to the given ImageInputStream or other Object. |
| void | [**setInput**](http://docs.google.com/javax/imageio/ImageReader.html#setInput(java.lang.Object,%20boolean,%20boolean))([Object](http://docs.google.com/java/lang/Object.html) input, boolean seekForwardOnly, boolean ignoreMetadata)            Sets the input source to use to the given ImageInputStream or other Object. |
| void | [**setLocale**](http://docs.google.com/javax/imageio/ImageReader.html#setLocale(java.util.Locale))([Locale](http://docs.google.com/java/util/Locale.html) locale)            Sets the current Locale of this ImageReader to the given value. |

| **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** |
| --- |

### originatingProvider

protected [ImageReaderSpi](http://docs.google.com/javax/imageio/spi/ImageReaderSpi.html) **originatingProvider**

The ImageReaderSpi that instantiated this object, or null if its identity is not known or none exists. By default it is initialized to null.

### input

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

The ImageInputStream or other Object by setInput and retrieved by getInput. By default it is initialized to null.

### seekForwardOnly

protected boolean **seekForwardOnly**

true if the current input source has been marked as allowing only forward seeking by setInput. By default, the value is false.

**See Also:**[minIndex](http://docs.google.com/javax/imageio/ImageReader.html#minIndex), [setInput(java.lang.Object, boolean, boolean)](http://docs.google.com/javax/imageio/ImageReader.html#setInput(java.lang.Object,%20boolean,%20boolean))

### ignoreMetadata

protected boolean **ignoreMetadata**

true if the current input source has been marked as allowing metadata to be ignored by setInput. By default, the value is false.

**See Also:**[setInput(java.lang.Object, boolean, boolean)](http://docs.google.com/javax/imageio/ImageReader.html#setInput(java.lang.Object,%20boolean,%20boolean))

### minIndex

protected int **minIndex**

The smallest valid index for reading, initially 0. When seekForwardOnly is true, various methods may throw an IndexOutOfBoundsException on an attempt to access data associate with an image having a lower index.

**See Also:**[seekForwardOnly](http://docs.google.com/javax/imageio/ImageReader.html#seekForwardOnly), [setInput(java.lang.Object, boolean, boolean)](http://docs.google.com/javax/imageio/ImageReader.html#setInput(java.lang.Object,%20boolean,%20boolean))

### availableLocales

protected [Locale](http://docs.google.com/java/util/Locale.html)[] **availableLocales**

An array of Locales which may be used to localize warning messages, or null if localization is not supported.

### locale

protected [Locale](http://docs.google.com/java/util/Locale.html) **locale**

The current Locale to be used for localization, or null if none has been set.

### warningListeners

protected [List](http://docs.google.com/java/util/List.html)<[IIOReadWarningListener](http://docs.google.com/javax/imageio/event/IIOReadWarningListener.html)> **warningListeners**

A List of currently registered IIOReadWarningListeners, initialized by default to null, which is synonymous with an empty List.

### warningLocales

protected [List](http://docs.google.com/java/util/List.html)<[Locale](http://docs.google.com/java/util/Locale.html)> **warningLocales**

A List of the Locales associated with each currently registered IIOReadWarningListener, initialized by default to null, which is synonymous with an empty List.

### progressListeners

protected [List](http://docs.google.com/java/util/List.html)<[IIOReadProgressListener](http://docs.google.com/javax/imageio/event/IIOReadProgressListener.html)> **progressListeners**

A List of currently registered IIOReadProgressListeners, initialized by default to null, which is synonymous with an empty List.

### updateListeners

protected [List](http://docs.google.com/java/util/List.html)<[IIOReadUpdateListener](http://docs.google.com/javax/imageio/event/IIOReadUpdateListener.html)> **updateListeners**

A List of currently registered IIOReadUpdateListeners, initialized by default to null, which is synonymous with an empty List.

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

### ImageReader

protected **ImageReader**([ImageReaderSpi](http://docs.google.com/javax/imageio/spi/ImageReaderSpi.html) originatingProvider)

Constructs an ImageReader and sets its originatingProvider field to the supplied value.

Subclasses that make use of extensions should provide a constructor with signature (ImageReaderSpi, Object) in order to retrieve the extension object. If the extension object is unsuitable, an IllegalArgumentException should be thrown.

**Parameters:**originatingProvider - the ImageReaderSpi that is invoking this constructor, or null.

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

### getFormatName

public [String](http://docs.google.com/java/lang/String.html) **getFormatName**()  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Returns a String identifying the format of the input source.

The default implementation returns originatingProvider.getFormatNames()[0]. Implementations that may not have an originating service provider, or which desire a different naming policy should override this method.

**Returns:**the format name, as a String. **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if an error occurs reading the information from the input source.

### getOriginatingProvider

public [ImageReaderSpi](http://docs.google.com/javax/imageio/spi/ImageReaderSpi.html) **getOriginatingProvider**()

Returns the ImageReaderSpi that was passed in on the constructor. Note that this value may be null.

**Returns:**an ImageReaderSpi, or null.**See Also:**[ImageReaderSpi](http://docs.google.com/javax/imageio/spi/ImageReaderSpi.html)

### setInput

public void **setInput**([Object](http://docs.google.com/java/lang/Object.html) input,  
 boolean seekForwardOnly,  
 boolean ignoreMetadata)

Sets the input source to use to the given ImageInputStream or other Object. The input source must be set before any of the query or read methods are used. If input is null, any currently set input source will be removed. In any case, the value of minIndex will be initialized to 0.

The seekForwardOnly parameter controls whether the value returned by getMinIndex will be increased as each image (or thumbnail, or image metadata) is read. If seekForwardOnly is true, then a call to read(index) will throw an IndexOutOfBoundsException if index < this.minIndex; otherwise, the value of minIndex will be set to index. If seekForwardOnly is false, the value of minIndex will remain 0 regardless of any read operations.

The ignoreMetadata parameter, if set to true, allows the reader to disregard any metadata encountered during the read. Subsequent calls to the getStreamMetadata and getImageMetadata methods may return null, and an IIOImage returned from readAll may return null from their getMetadata method. Setting this parameter may allow the reader to work more efficiently. The reader may choose to disregard this setting and return metadata normally.

Subclasses should take care to remove any cached information based on the previous stream, such as header information or partially decoded image data.

Use of a general Object other than an ImageInputStream is intended for readers that interact directly with a capture device or imaging protocol. The set of legal classes is advertised by the reader's service provider's getInputTypes method; most readers will return a single-element array containing only ImageInputStream.class to indicate that they accept only an ImageInputStream.

The default implementation checks the input argument against the list returned by originatingProvider.getInputTypes() and fails if the argument is not an instance of one of the classes in the list. If the originating provider is set to null, the input is accepted only if it is an ImageInputStream.

**Parameters:**input - the ImageInputStream or other Object to use for future decoding.seekForwardOnly - if true, images and metadata may only be read in ascending order from this input source.ignoreMetadata - if true, metadata may be ignored during reads. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if input is not an instance of one of the classes returned by the originating service provider's getInputTypes method, or is not an ImageInputStream.**See Also:**[ImageInputStream](http://docs.google.com/javax/imageio/stream/ImageInputStream.html), [getInput()](http://docs.google.com/javax/imageio/ImageReader.html#getInput()), [ImageReaderSpi.getInputTypes()](http://docs.google.com/javax/imageio/spi/ImageReaderSpi.html#getInputTypes())

### setInput

public void **setInput**([Object](http://docs.google.com/java/lang/Object.html) input,  
 boolean seekForwardOnly)

Sets the input source to use to the given ImageInputStream or other Object. The input source must be set before any of the query or read methods are used. If input is null, any currently set input source will be removed. In any case, the value of minIndex will be initialized to 0.

The seekForwardOnly parameter controls whether the value returned by getMinIndex will be increased as each image (or thumbnail, or image metadata) is read. If seekForwardOnly is true, then a call to read(index) will throw an IndexOutOfBoundsException if index < this.minIndex; otherwise, the value of minIndex will be set to index. If seekForwardOnly is false, the value of minIndex will remain 0 regardless of any read operations.

This method is equivalent to setInput(input, seekForwardOnly, false).

**Parameters:**input - the ImageInputStream or other Object to use for future decoding.seekForwardOnly - if true, images and metadata may only be read in ascending order from this input source. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if input is not an instance of one of the classes returned by the originating service provider's getInputTypes method, or is not an ImageInputStream.**See Also:**[getInput()](http://docs.google.com/javax/imageio/ImageReader.html#getInput())

### setInput

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

Sets the input source to use to the given ImageInputStream or other Object. The input source must be set before any of the query or read methods are used. If input is null, any currently set input source will be removed. In any case, the value of minIndex will be initialized to 0.

This method is equivalent to setInput(input, false, false).

**Parameters:**input - the ImageInputStream or other Object to use for future decoding. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if input is not an instance of one of the classes returned by the originating service provider's getInputTypes method, or is not an ImageInputStream.**See Also:**[getInput()](http://docs.google.com/javax/imageio/ImageReader.html#getInput())

### getInput

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

Returns the ImageInputStream or other Object previously set as the input source. If the input source has not been set, null is returned.

**Returns:**the Object that will be used for future decoding, or null.**See Also:**[ImageInputStream](http://docs.google.com/javax/imageio/stream/ImageInputStream.html), [setInput(java.lang.Object, boolean, boolean)](http://docs.google.com/javax/imageio/ImageReader.html#setInput(java.lang.Object,%20boolean,%20boolean))

### isSeekForwardOnly

public boolean **isSeekForwardOnly**()

Returns true if the current input source has been marked as seek forward only by passing true as the seekForwardOnly argument to the setInput method.

**Returns:**true if the input source is seek forward only.**See Also:**[setInput(java.lang.Object, boolean, boolean)](http://docs.google.com/javax/imageio/ImageReader.html#setInput(java.lang.Object,%20boolean,%20boolean))

### isIgnoringMetadata

public boolean **isIgnoringMetadata**()

Returns true if the current input source has been marked as allowing metadata to be ignored by passing true as the ignoreMetadata argument to the setInput method.

**Returns:**true if the metadata may be ignored.**See Also:**[setInput(java.lang.Object, boolean, boolean)](http://docs.google.com/javax/imageio/ImageReader.html#setInput(java.lang.Object,%20boolean,%20boolean))

### getMinIndex

public int **getMinIndex**()

Returns the lowest valid index for reading an image, thumbnail, or image metadata. If seekForwardOnly() is false, this value will typically remain 0, indicating that random access is possible. Otherwise, it will contain the value of the most recently accessed index, and increase in a monotonic fashion.

**Returns:**the minimum legal index for reading.

### getAvailableLocales

public [Locale](http://docs.google.com/java/util/Locale.html)[] **getAvailableLocales**()

Returns an array of Locales that may be used to localize warning listeners and compression settings. A return value of null indicates that localization is not supported.

The default implementation returns a clone of the availableLocales instance variable if it is non-null, or else returns null.

**Returns:**an array of Locales that may be used as arguments to setLocale, or null.

### setLocale

public void **setLocale**([Locale](http://docs.google.com/java/util/Locale.html) locale)

Sets the current Locale of this ImageReader to the given value. A value of null removes any previous setting, and indicates that the reader should localize as it sees fit.

**Parameters:**locale - the desired Locale, or null. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if locale is non-null but is not one of the values returned by getAvailableLocales.**See Also:**[getLocale()](http://docs.google.com/javax/imageio/ImageReader.html#getLocale())

### getLocale

public [Locale](http://docs.google.com/java/util/Locale.html) **getLocale**()

Returns the currently set Locale, or null if none has been set.

**Returns:**the current Locale, or null.**See Also:**[setLocale(java.util.Locale)](http://docs.google.com/javax/imageio/ImageReader.html#setLocale(java.util.Locale))

### getNumImages

public abstract int **getNumImages**(boolean allowSearch)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Returns the number of images, not including thumbnails, available from the current input source.

Note that some image formats (such as animated GIF) do not specify how many images are present in the stream. Thus determining the number of images will require the entire stream to be scanned and may require memory for buffering. If images are to be processed in order, it may be more efficient to simply call read with increasing indices until an IndexOutOfBoundsException is thrown to indicate that no more images are available. The allowSearch parameter may be set to false to indicate that an exhaustive search is not desired; the return value will be -1 to indicate that a search is necessary. If the input has been specified with seekForwardOnly set to true, this method throws an IllegalStateException if allowSearch is set to true.

**Parameters:**allowSearch - if true, the true number of images will be returned even if a search is required. If false, the reader may return -1 without performing the search. **Returns:**the number of images, as an int, or -1 if allowSearch is false and a search would be required. **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the input source has not been set, or if the input has been specified with seekForwardOnly set to true. [IOException](http://docs.google.com/java/io/IOException.html) - if an error occurs reading the information from the input source.**See Also:**[setInput(java.lang.Object, boolean, boolean)](http://docs.google.com/javax/imageio/ImageReader.html#setInput(java.lang.Object,%20boolean,%20boolean))

### getWidth

public abstract int **getWidth**(int imageIndex)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Returns the width in pixels of the given image within the input source.

If the image can be rendered to a user-specified size, then this method returns the default width.

**Parameters:**imageIndex - the index of the image to be queried. **Returns:**the width of the image, as an int. **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the input source has not been set. [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if the supplied index is out of bounds. [IOException](http://docs.google.com/java/io/IOException.html) - if an error occurs reading the width information from the input source.

### getHeight

public abstract int **getHeight**(int imageIndex)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Returns the height in pixels of the given image within the input source.

If the image can be rendered to a user-specified size, then this method returns the default height.

**Parameters:**imageIndex - the index of the image to be queried. **Returns:**the height of the image, as an int. **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the input source has not been set. [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if the supplied index is out of bounds. [IOException](http://docs.google.com/java/io/IOException.html) - if an error occurs reading the height information from the input source.

### isRandomAccessEasy

public boolean **isRandomAccessEasy**(int imageIndex)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Returns true if the storage format of the given image places no inherent impediment on random access to pixels. For most compressed formats, such as JPEG, this method should return false, as a large section of the image in addition to the region of interest may need to be decoded.

This is merely a hint for programs that wish to be efficient; all readers must be able to read arbitrary regions as specified in an ImageReadParam.

Note that formats that return false from this method may nonetheless allow tiling (*e.g.* Restart Markers in JPEG), and random access will likely be reasonably efficient on tiles. See [isImageTiled](http://docs.google.com/javax/imageio/ImageReader.html#isImageTiled(int)).

A reader for which all images are guaranteed to support easy random access, or are guaranteed not to support easy random access, may return true or false respectively without accessing any image data. In such cases, it is not necessary to throw an exception even if no input source has been set or the image index is out of bounds.

The default implementation returns false.

**Parameters:**imageIndex - the index of the image to be queried. **Returns:**true if reading a region of interest of the given image is likely to be efficient. **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if an input source is required to determine the return value, but none has been set. [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if an image must be accessed to determine the return value, but the supplied index is out of bounds. [IOException](http://docs.google.com/java/io/IOException.html) - if an error occurs during reading.

### getAspectRatio

public float **getAspectRatio**(int imageIndex)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Returns the aspect ratio of the given image (that is, its width divided by its height) as a float. For images that are inherently resizable, this method provides a way to determine the appropriate width given a deired height, or vice versa. For non-resizable images, the true width and height are used.

The default implementation simply returns (float)getWidth(imageIndex)/getHeight(imageIndex).

**Parameters:**imageIndex - the index of the image to be queried. **Returns:**a float indicating the aspect ratio of the given image. **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the input source has not been set. [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if the supplied index is out of bounds. [IOException](http://docs.google.com/java/io/IOException.html) - if an error occurs during reading.

### getRawImageType

public [ImageTypeSpecifier](http://docs.google.com/javax/imageio/ImageTypeSpecifier.html) **getRawImageType**(int imageIndex)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Returns an ImageTypeSpecifier indicating the SampleModel and ColorModel which most closely represents the "raw" internal format of the image. For example, for a JPEG image the raw type might have a YCbCr color space even though the image would conventionally be transformed into an RGB color space prior to display. The returned value should also be included in the list of values returned by getImageTypes.

The default implementation simply returns the first entry from the list provided by getImageType.

**Parameters:**imageIndex - the index of the image to be queried. **Returns:**an ImageTypeSpecifier. **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the input source has not been set. [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if the supplied index is out of bounds. [IOException](http://docs.google.com/java/io/IOException.html) - if an error occurs reading the format information from the input source.

### getImageTypes

public abstract [Iterator](http://docs.google.com/java/util/Iterator.html)<[ImageTypeSpecifier](http://docs.google.com/javax/imageio/ImageTypeSpecifier.html)> **getImageTypes**(int imageIndex)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Returns an Iterator containing possible image types to which the given image may be decoded, in the form of ImageTypeSpecifierss. At least one legal image type will be returned.

The first element of the iterator should be the most "natural" type for decoding the image with as little loss as possible. For example, for a JPEG image the first entry should be an RGB image, even though the image data is stored internally in a YCbCr color space.

**Parameters:**imageIndex - the index of the image to be retrieved. **Returns:**an Iterator containing at least one ImageTypeSpecifier representing suggested image types for decoding the current given image. **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the input source has not been set. [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if the supplied index is out of bounds. [IOException](http://docs.google.com/java/io/IOException.html) - if an error occurs reading the format information from the input source.**See Also:**[ImageReadParam.setDestination(BufferedImage)](http://docs.google.com/javax/imageio/ImageReadParam.html#setDestination(java.awt.image.BufferedImage)), [ImageReadParam.setDestinationType(ImageTypeSpecifier)](http://docs.google.com/javax/imageio/ImageReadParam.html#setDestinationType(javax.imageio.ImageTypeSpecifier))

### getDefaultReadParam

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

Returns a default ImageReadParam object appropriate for this format. All subclasses should define a set of default values for all parameters and return them with this call. This method may be called before the input source is set.

The default implementation constructs and returns a new ImageReadParam object that does not allow source scaling (*i.e.*, it returns new ImageReadParam().

**Returns:**an ImageReadParam object which may be used to control the decoding process using a set of default settings.

### getStreamMetadata

public abstract [IIOMetadata](http://docs.google.com/javax/imageio/metadata/IIOMetadata.html) **getStreamMetadata**()  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Returns an IIOMetadata object representing the metadata associated with the input source as a whole (i.e., not associated with any particular image), or null if the reader does not support reading metadata, is set to ignore metadata, or if no metadata is available.

**Returns:**an IIOMetadata object, or null. **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if an error occurs during reading.

### getStreamMetadata

public [IIOMetadata](http://docs.google.com/javax/imageio/metadata/IIOMetadata.html) **getStreamMetadata**([String](http://docs.google.com/java/lang/String.html) formatName,  
 [Set](http://docs.google.com/java/util/Set.html)<[String](http://docs.google.com/java/lang/String.html)> nodeNames)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Returns an IIOMetadata object representing the metadata associated with the input source as a whole (i.e., not associated with any particular image). If no such data exists, null is returned.

The resuting metadata object is only responsible for returning documents in the format named by formatName. Within any documents that are returned, only nodes whose names are members of nodeNames are required to be returned. In this way, the amount of metadata processing done by the reader may be kept to a minimum, based on what information is actually needed.

If formatName is not the name of a supported metadata format, null is returned.

In all cases, it is legal to return a more capable metadata object than strictly necessary. The format name and node names are merely hints that may be used to reduce the reader's workload.

The default implementation simply returns the result of calling getStreamMetadata(), after checking that the format name is supported. If it is not, null is returned.

**Parameters:**formatName - a metadata format name that may be used to retrieve a document from the returned IIOMetadata object.nodeNames - a Set containing the names of nodes that may be contained in a retrieved document. **Returns:**an IIOMetadata object, or null. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if formatName is null. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if nodeNames is null. [IOException](http://docs.google.com/java/io/IOException.html) - if an error occurs during reading.

### getImageMetadata

public abstract [IIOMetadata](http://docs.google.com/javax/imageio/metadata/IIOMetadata.html) **getImageMetadata**(int imageIndex)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Returns an IIOMetadata object containing metadata associated with the given image, or null if the reader does not support reading metadata, is set to ignore metadata, or if no metadata is available.

**Parameters:**imageIndex - the index of the image whose metadata is to be retrieved. **Returns:**an IIOMetadata object, or null. **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the input source has not been set. [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if the supplied index is out of bounds. [IOException](http://docs.google.com/java/io/IOException.html) - if an error occurs during reading.

### getImageMetadata

public [IIOMetadata](http://docs.google.com/javax/imageio/metadata/IIOMetadata.html) **getImageMetadata**(int imageIndex,  
 [String](http://docs.google.com/java/lang/String.html) formatName,  
 [Set](http://docs.google.com/java/util/Set.html)<[String](http://docs.google.com/java/lang/String.html)> nodeNames)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Returns an IIOMetadata object representing the metadata associated with the given image, or null if the reader does not support reading metadata or none is available.

The resuting metadata object is only responsible for returning documents in the format named by formatName. Within any documents that are returned, only nodes whose names are members of nodeNames are required to be returned. In this way, the amount of metadata processing done by the reader may be kept to a minimum, based on what information is actually needed.

If formatName is not the name of a supported metadata format, null may be returned.

In all cases, it is legal to return a more capable metadata object than strictly necessary. The format name and node names are merely hints that may be used to reduce the reader's workload.

The default implementation simply returns the result of calling getImageMetadata(imageIndex), after checking that the format name is supported. If it is not, null is returned.

**Parameters:**imageIndex - the index of the image whose metadata is to be retrieved.formatName - a metadata format name that may be used to retrieve a document from the returned IIOMetadata object.nodeNames - a Set containing the names of nodes that may be contained in a retrieved document. **Returns:**an IIOMetadata object, or null. **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the input source has not been set. [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if the supplied index is out of bounds. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if formatName is null. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if nodeNames is null. [IOException](http://docs.google.com/java/io/IOException.html) - if an error occurs during reading.

### read

public [BufferedImage](http://docs.google.com/java/awt/image/BufferedImage.html) **read**(int imageIndex)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Reads the image indexed by imageIndex and returns it as a complete BufferedImage, using a default ImageReadParam. This is a convenience method that calls read(imageIndex, null).

The image returned will be formatted according to the first ImageTypeSpecifier returned from getImageTypes.

Any registered IIOReadProgressListener objects will be notified by calling their imageStarted method, followed by calls to their imageProgress method as the read progresses. Finally their imageComplete method will be called. IIOReadUpdateListener objects may be updated at other times during the read as pixels are decoded. Finally, IIOReadWarningListener objects will receive notification of any non-fatal warnings that occur during decoding.

**Parameters:**imageIndex - the index of the image to be retrieved. **Returns:**the desired portion of the image as a BufferedImage. **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the input source has not been set. [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if the supplied index is out of bounds. [IOException](http://docs.google.com/java/io/IOException.html) - if an error occurs during reading.

### read

public abstract [BufferedImage](http://docs.google.com/java/awt/image/BufferedImage.html) **read**(int imageIndex,  
 [ImageReadParam](http://docs.google.com/javax/imageio/ImageReadParam.html) param)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Reads the image indexed by imageIndex and returns it as a complete BufferedImage, using a supplied ImageReadParam.

The actual BufferedImage returned will be chosen using the algorithm defined by the getDestination method.

Any registered IIOReadProgressListener objects will be notified by calling their imageStarted method, followed by calls to their imageProgress method as the read progresses. Finally their imageComplete method will be called. IIOReadUpdateListener objects may be updated at other times during the read as pixels are decoded. Finally, IIOReadWarningListener objects will receive notification of any non-fatal warnings that occur during decoding.

The set of source bands to be read and destination bands to be written is determined by calling getSourceBands and getDestinationBands on the supplied ImageReadParam. If the lengths of the arrays returned by these methods differ, the set of source bands contains an index larger that the largest available source index, or the set of destination bands contains an index larger than the largest legal destination index, an IllegalArgumentException is thrown.

If the supplied ImageReadParam contains optional setting values not supported by this reader (*e.g.* source render size or any format-specific settings), they will be ignored.

**Parameters:**imageIndex - the index of the image to be retrieved.param - an ImageReadParam used to control the reading process, or null. **Returns:**the desired portion of the image as a BufferedImage. **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the input source has not been set. [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if the supplied index is out of bounds. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the set of source and destination bands specified by param.getSourceBands and param.getDestinationBands differ in length or include indices that are out of bounds. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the resulting image would have a width or height less than 1. [IOException](http://docs.google.com/java/io/IOException.html) - if an error occurs during reading.

### readAll

public [IIOImage](http://docs.google.com/javax/imageio/IIOImage.html) **readAll**(int imageIndex,  
 [ImageReadParam](http://docs.google.com/javax/imageio/ImageReadParam.html) param)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Reads the image indexed by imageIndex and returns an IIOImage containing the image, thumbnails, and associated image metadata, using a supplied ImageReadParam.

The actual BufferedImage referenced by the returned IIOImage will be chosen using the algorithm defined by the getDestination method.

Any registered IIOReadProgressListener objects will be notified by calling their imageStarted method, followed by calls to their imageProgress method as the read progresses. Finally their imageComplete method will be called. IIOReadUpdateListener objects may be updated at other times during the read as pixels are decoded. Finally, IIOReadWarningListener objects will receive notification of any non-fatal warnings that occur during decoding.

The set of source bands to be read and destination bands to be written is determined by calling getSourceBands and getDestinationBands on the supplied ImageReadParam. If the lengths of the arrays returned by these methods differ, the set of source bands contains an index larger that the largest available source index, or the set of destination bands contains an index larger than the largest legal destination index, an IllegalArgumentException is thrown.

Thumbnails will be returned in their entirety regardless of the region settings.

If the supplied ImageReadParam contains optional setting values not supported by this reader (*e.g.* source render size or any format-specific settings), those values will be ignored.

**Parameters:**imageIndex - the index of the image to be retrieved.param - an ImageReadParam used to control the reading process, or null. **Returns:**an IIOImage containing the desired portion of the image, a set of thumbnails, and associated image metadata. **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the input source has not been set. [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if the supplied index is out of bounds. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the set of source and destination bands specified by param.getSourceBands and param.getDestinationBands differ in length or include indices that are out of bounds. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the resulting image would have a width or height less than 1. [IOException](http://docs.google.com/java/io/IOException.html) - if an error occurs during reading.

### readAll

public [Iterator](http://docs.google.com/java/util/Iterator.html)<[IIOImage](http://docs.google.com/javax/imageio/IIOImage.html)> **readAll**([Iterator](http://docs.google.com/java/util/Iterator.html)<? extends [ImageReadParam](http://docs.google.com/javax/imageio/ImageReadParam.html)> params)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Returns an Iterator containing all the images, thumbnails, and metadata, starting at the index given by getMinIndex, from the input source in the form of IIOImage objects. An Iterator containing ImageReadParam objects is supplied; one element is consumed for each image read from the input source until no more images are available. If the read param Iterator runs out of elements, but there are still more images available from the input source, default read params are used for the remaining images.

If params is null, a default read param will be used for all images.

The actual BufferedImage referenced by the returned IIOImage will be chosen using the algorithm defined by the getDestination method.

Any registered IIOReadProgressListener objects will be notified by calling their sequenceStarted method once. Then, for each image decoded, there will be a call to imageStarted, followed by calls to imageProgress as the read progresses, and finally to imageComplete. The sequenceComplete method will be called after the last image has been decoded. IIOReadUpdateListener objects may be updated at other times during the read as pixels are decoded. Finally, IIOReadWarningListener objects will receive notification of any non-fatal warnings that occur during decoding.

The set of source bands to be read and destination bands to be written is determined by calling getSourceBands and getDestinationBands on the supplied ImageReadParam. If the lengths of the arrays returned by these methods differ, the set of source bands contains an index larger that the largest available source index, or the set of destination bands contains an index larger than the largest legal destination index, an IllegalArgumentException is thrown.

Thumbnails will be returned in their entirety regardless of the region settings.

If any of the supplied ImageReadParams contain optional setting values not supported by this reader (*e.g.* source render size or any format-specific settings), they will be ignored.

**Parameters:**params - an Iterator containing ImageReadParam objects. **Returns:**an Iterator representing the contents of the input source as IIOImages. **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the input source has not been set. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if any non-null element of params is not an ImageReadParam. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the set of source and destination bands specified by param.getSourceBands and param.getDestinationBands differ in length or include indices that are out of bounds. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if a resulting image would have a width or height less than 1. [IOException](http://docs.google.com/java/io/IOException.html) - if an error occurs during reading.**See Also:**[ImageReadParam](http://docs.google.com/javax/imageio/ImageReadParam.html), [IIOImage](http://docs.google.com/javax/imageio/IIOImage.html)

### canReadRaster

public boolean **canReadRaster**()

Returns true if this plug-in supports reading just a [Raster](http://docs.google.com/java/awt/image/Raster.html) of pixel data. If this method returns false, calls to [readRaster](http://docs.google.com/javax/imageio/ImageReader.html#readRaster(int,%20javax.imageio.ImageReadParam)) or [readTileRaster](http://docs.google.com/javax/imageio/ImageReader.html#readTileRaster(int,%20int,%20int)) will throw an UnsupportedOperationException.

The default implementation returns false.

**Returns:**true if this plug-in supports reading raw Rasters.**See Also:**[readRaster(int, javax.imageio.ImageReadParam)](http://docs.google.com/javax/imageio/ImageReader.html#readRaster(int,%20javax.imageio.ImageReadParam)), [readTileRaster(int, int, int)](http://docs.google.com/javax/imageio/ImageReader.html#readTileRaster(int,%20int,%20int))

### readRaster

public [Raster](http://docs.google.com/java/awt/image/Raster.html) **readRaster**(int imageIndex,  
 [ImageReadParam](http://docs.google.com/javax/imageio/ImageReadParam.html) param)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Returns a new Raster object containing the raw pixel data from the image stream, without any color conversion applied. The application must determine how to interpret the pixel data by other means. Any destination or image-type parameters in the supplied ImageReadParam object are ignored, but all other parameters are used exactly as in the [read](http://docs.google.com/javax/imageio/ImageReader.html#read(int)) method, except that any destination offset is used as a logical rather than a physical offset. The size of the returned Raster will always be that of the source region clipped to the actual image. Logical offsets in the stream itself are ignored.

This method allows formats that normally apply a color conversion, such as JPEG, and formats that do not normally have an associated colorspace, such as remote sensing or medical imaging data, to provide access to raw pixel data.

Any registered readUpdateListeners are ignored, as there is no BufferedImage, but all other listeners are called exactly as they are for the [read](http://docs.google.com/javax/imageio/ImageReader.html#read(int)) method.

If [canReadRaster()](http://docs.google.com/javax/imageio/ImageReader.html#canReadRaster()) returns false, this method throws an UnsupportedOperationException.

If the supplied ImageReadParam contains optional setting values not supported by this reader (*e.g.* source render size or any format-specific settings), they will be ignored.

The default implementation throws an UnsupportedOperationException.

**Parameters:**imageIndex - the index of the image to be read.param - an ImageReadParam used to control the reading process, or null. **Returns:**the desired portion of the image as a Raster. **Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if this plug-in does not support reading raw Rasters. [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the input source has not been set. [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if the supplied index is out of bounds. [IOException](http://docs.google.com/java/io/IOException.html) - if an error occurs during reading.**See Also:**[canReadRaster()](http://docs.google.com/javax/imageio/ImageReader.html#canReadRaster()), [read(int)](http://docs.google.com/javax/imageio/ImageReader.html#read(int)), [Raster](http://docs.google.com/java/awt/image/Raster.html)

### isImageTiled

public boolean **isImageTiled**(int imageIndex)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Returns true if the image is organized into *tiles*, that is, equal-sized non-overlapping rectangles.

A reader plug-in may choose whether or not to expose tiling that is present in the image as it is stored. It may even choose to advertise tiling when none is explicitly present. In general, tiling should only be advertised if there is some advantage (in speed or space) to accessing individual tiles. Regardless of whether the reader advertises tiling, it must be capable of reading an arbitrary rectangular region specified in an ImageReadParam.

A reader for which all images are guaranteed to be tiled, or are guaranteed not to be tiled, may return true or false respectively without accessing any image data. In such cases, it is not necessary to throw an exception even if no input source has been set or the image index is out of bounds.

The default implementation just returns false.

**Parameters:**imageIndex - the index of the image to be queried. **Returns:**true if the image is tiled. **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if an input source is required to determine the return value, but none has been set. [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if an image must be accessed to determine the return value, but the supplied index is out of bounds. [IOException](http://docs.google.com/java/io/IOException.html) - if an error occurs during reading.

### getTileWidth

public int **getTileWidth**(int imageIndex)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Returns the width of a tile in the given image.

The default implementation simply returns getWidth(imageIndex), which is correct for non-tiled images. Readers that support tiling should override this method.

**Parameters:**imageIndex - the index of the image to be queried. **Returns:**the width of a tile. **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the input source has not been set. [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if the supplied index is out of bounds. [IOException](http://docs.google.com/java/io/IOException.html) - if an error occurs during reading.

### getTileHeight

public int **getTileHeight**(int imageIndex)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Returns the height of a tile in the given image.

The default implementation simply returns getHeight(imageIndex), which is correct for non-tiled images. Readers that support tiling should override this method.

**Parameters:**imageIndex - the index of the image to be queried. **Returns:**the height of a tile. **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the input source has not been set. [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if the supplied index is out of bounds. [IOException](http://docs.google.com/java/io/IOException.html) - if an error occurs during reading.

### getTileGridXOffset

public int **getTileGridXOffset**(int imageIndex)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Returns the X coordinate of the upper-left corner of tile (0, 0) in the given image.

A reader for which the tile grid X offset always has the same value (usually 0), may return the value without accessing any image data. In such cases, it is not necessary to throw an exception even if no input source has been set or the image index is out of bounds.

The default implementation simply returns 0, which is correct for non-tiled images and tiled images in most formats. Readers that support tiling with non-(0, 0) offsets should override this method.

**Parameters:**imageIndex - the index of the image to be queried. **Returns:**the X offset of the tile grid. **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if an input source is required to determine the return value, but none has been set. [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if an image must be accessed to determine the return value, but the supplied index is out of bounds. [IOException](http://docs.google.com/java/io/IOException.html) - if an error occurs during reading.

### getTileGridYOffset

public int **getTileGridYOffset**(int imageIndex)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Returns the Y coordinate of the upper-left corner of tile (0, 0) in the given image.

A reader for which the tile grid Y offset always has the same value (usually 0), may return the value without accessing any image data. In such cases, it is not necessary to throw an exception even if no input source has been set or the image index is out of bounds.

The default implementation simply returns 0, which is correct for non-tiled images and tiled images in most formats. Readers that support tiling with non-(0, 0) offsets should override this method.

**Parameters:**imageIndex - the index of the image to be queried. **Returns:**the Y offset of the tile grid. **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if an input source is required to determine the return value, but none has been set. [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if an image must be accessed to determine the return value, but the supplied index is out of bounds. [IOException](http://docs.google.com/java/io/IOException.html) - if an error occurs during reading.

### readTile

public [BufferedImage](http://docs.google.com/java/awt/image/BufferedImage.html) **readTile**(int imageIndex,  
 int tileX,  
 int tileY)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Reads the tile indicated by the tileX and tileY arguments, returning it as a BufferedImage. If the arguments are out of range, an IllegalArgumentException is thrown. If the image is not tiled, the values 0, 0 will return the entire image; any other values will cause an IllegalArgumentException to be thrown.

This method is merely a convenience equivalent to calling read(int, ImageReadParam) with a read param specifiying a source region having offsets of tileX\*getTileWidth(imageIndex), tileY\*getTileHeight(imageIndex) and width and height of getTileWidth(imageIndex), getTileHeight(imageIndex); and subsampling factors of 1 and offsets of 0. To subsample a tile, call read with a read param specifying this region and different subsampling parameters.

The default implementation returns the entire image if tileX and tileY are 0, or throws an IllegalArgumentException otherwise.

**Parameters:**imageIndex - the index of the image to be retrieved.tileX - the column index (starting with 0) of the tile to be retrieved.tileY - the row index (starting with 0) of the tile to be retrieved. **Returns:**the tile as a BufferedImage. **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the input source has not been set. [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if imageIndex is out of bounds. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the tile indices are out of bounds. [IOException](http://docs.google.com/java/io/IOException.html) - if an error occurs during reading.

### readTileRaster

public [Raster](http://docs.google.com/java/awt/image/Raster.html) **readTileRaster**(int imageIndex,  
 int tileX,  
 int tileY)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Returns a new Raster object containing the raw pixel data from the tile, without any color conversion applied. The application must determine how to interpret the pixel data by other means.

If [canReadRaster()](http://docs.google.com/javax/imageio/ImageReader.html#canReadRaster()) returns false, this method throws an UnsupportedOperationException.

The default implementation checks if reading Rasters is supported, and if so calls [readRaster(imageIndex, null)](http://docs.google.com/javax/imageio/ImageReader.html#readRaster(int,%20javax.imageio.ImageReadParam)) if tileX and tileY are 0, or throws an IllegalArgumentException otherwise.

**Parameters:**imageIndex - the index of the image to be retrieved.tileX - the column index (starting with 0) of the tile to be retrieved.tileY - the row index (starting with 0) of the tile to be retrieved. **Returns:**the tile as a Raster. **Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if this plug-in does not support reading raw Rasters. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the tile indices are out of bounds. [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the input source has not been set. [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if imageIndex is out of bounds. [IOException](http://docs.google.com/java/io/IOException.html) - if an error occurs during reading.**See Also:**[readTile(int, int, int)](http://docs.google.com/javax/imageio/ImageReader.html#readTile(int,%20int,%20int)), [readRaster(int, javax.imageio.ImageReadParam)](http://docs.google.com/javax/imageio/ImageReader.html#readRaster(int,%20javax.imageio.ImageReadParam)), [Raster](http://docs.google.com/java/awt/image/Raster.html)

### readAsRenderedImage

public [RenderedImage](http://docs.google.com/java/awt/image/RenderedImage.html) **readAsRenderedImage**(int imageIndex,  
 [ImageReadParam](http://docs.google.com/javax/imageio/ImageReadParam.html) param)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Returns a RenderedImage object that contains the contents of the image indexed by imageIndex. By default, the returned image is simply the BufferedImage returned by read(imageIndex, param).

The semantics of this method may differ from those of the other read methods in several ways. First, any destination image and/or image type set in the ImageReadParam may be ignored. Second, the usual listener calls are not guaranteed to be made, or to be meaningful if they are. This is because the returned image may not be fully populated with pixel data at the time it is returned, or indeed at any time.

If the supplied ImageReadParam contains optional setting values not supported by this reader (*e.g.* source render size or any format-specific settings), they will be ignored.

The default implementation just calls [read(imageIndex, param)](http://docs.google.com/javax/imageio/ImageReader.html#read(int)).

**Parameters:**imageIndex - the index of the image to be retrieved.param - an ImageReadParam used to control the reading process, or null. **Returns:**a RenderedImage object providing a view of the image. **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the input source has not been set. [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if the supplied index is out of bounds. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the set of source and destination bands specified by param.getSourceBands and param.getDestinationBands differ in length or include indices that are out of bounds. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the resulting image would have a width or height less than 1. [IOException](http://docs.google.com/java/io/IOException.html) - if an error occurs during reading.

### readerSupportsThumbnails

public boolean **readerSupportsThumbnails**()

Returns true if the image format understood by this reader supports thumbnail preview images associated with it. The default implementation returns false.

If this method returns false, hasThumbnails and getNumThumbnails will return false and 0, respectively, and readThumbnail will throw an UnsupportedOperationException, regardless of their arguments.

A reader that does not support thumbnails need not implement any of the thumbnail-related methods.

**Returns:**true if thumbnails are supported.

### hasThumbnails

public boolean **hasThumbnails**(int imageIndex)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Returns true if the given image has thumbnail preview images associated with it. If the format does not support thumbnails (readerSupportsThumbnails returns false), false will be returned regardless of whether an input source has been set or whether imageIndex is in bounds.

The default implementation returns true if getNumThumbnails returns a value greater than 0.

**Parameters:**imageIndex - the index of the image being queried. **Returns:**true if the given image has thumbnails. **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the reader supports thumbnails but the input source has not been set. [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if the reader supports thumbnails but imageIndex is out of bounds. [IOException](http://docs.google.com/java/io/IOException.html) - if an error occurs during reading.

### getNumThumbnails

public int **getNumThumbnails**(int imageIndex)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Returns the number of thumbnail preview images associated with the given image. If the format does not support thumbnails, (readerSupportsThumbnails returns false), 0 will be returned regardless of whether an input source has been set or whether imageIndex is in bounds.

The default implementation returns 0 without checking its argument.

**Parameters:**imageIndex - the index of the image being queried. **Returns:**the number of thumbnails associated with the given image. **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the reader supports thumbnails but the input source has not been set. [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if the reader supports thumbnails but imageIndex is out of bounds. [IOException](http://docs.google.com/java/io/IOException.html) - if an error occurs during reading.

### getThumbnailWidth

public int **getThumbnailWidth**(int imageIndex,  
 int thumbnailIndex)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Returns the width of the thumbnail preview image indexed by thumbnailIndex, associated with the image indexed by ImageIndex.

If the reader does not support thumbnails, (readerSupportsThumbnails returns false), an UnsupportedOperationException will be thrown.

The default implementation simply returns readThumbnail(imageindex, thumbnailIndex).getWidth(). Subclasses should therefore override this method if possible in order to avoid forcing the thumbnail to be read.

**Parameters:**imageIndex - the index of the image to be retrieved.thumbnailIndex - the index of the thumbnail to be retrieved. **Returns:**the width of the desired thumbnail as an int. **Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if thumbnails are not supported. [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the input source has not been set. [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if either of the supplied indices are out of bounds. [IOException](http://docs.google.com/java/io/IOException.html) - if an error occurs during reading.

### getThumbnailHeight

public int **getThumbnailHeight**(int imageIndex,  
 int thumbnailIndex)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Returns the height of the thumbnail preview image indexed by thumbnailIndex, associated with the image indexed by ImageIndex.

If the reader does not support thumbnails, (readerSupportsThumbnails returns false), an UnsupportedOperationException will be thrown.

The default implementation simply returns readThumbnail(imageindex, thumbnailIndex).getHeight(). Subclasses should therefore override this method if possible in order to avoid forcing the thumbnail to be read.

**Parameters:**imageIndex - the index of the image to be retrieved.thumbnailIndex - the index of the thumbnail to be retrieved. **Returns:**the height of the desired thumbnail as an int. **Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if thumbnails are not supported. [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the input source has not been set. [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if either of the supplied indices are out of bounds. [IOException](http://docs.google.com/java/io/IOException.html) - if an error occurs during reading.

### readThumbnail

public [BufferedImage](http://docs.google.com/java/awt/image/BufferedImage.html) **readThumbnail**(int imageIndex,  
 int thumbnailIndex)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Returns the thumbnail preview image indexed by thumbnailIndex, associated with the image indexed by ImageIndex as a BufferedImage.

Any registered IIOReadProgressListener objects will be notified by calling their thumbnailStarted, thumbnailProgress, and thumbnailComplete methods.

If the reader does not support thumbnails, (readerSupportsThumbnails returns false), an UnsupportedOperationException will be thrown regardless of whether an input source has been set or whether the indices are in bounds.

The default implementation throws an UnsupportedOperationException.

**Parameters:**imageIndex - the index of the image to be retrieved.thumbnailIndex - the index of the thumbnail to be retrieved. **Returns:**the desired thumbnail as a BufferedImage. **Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if thumbnails are not supported. [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the input source has not been set. [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if either of the supplied indices are out of bounds. [IOException](http://docs.google.com/java/io/IOException.html) - if an error occurs during reading.

### abort

public void **abort**()

Requests that any current read operation be aborted. The contents of the image following the abort will be undefined.

Readers should call clearAbortRequest at the beginning of each read operation, and poll the value of abortRequested regularly during the read.

### abortRequested

protected boolean **abortRequested**()

Returns true if a request to abort the current read operation has been made since the reader was instantiated or clearAbortRequest was called.

**Returns:**true if the current read operation should be aborted.**See Also:**[abort()](http://docs.google.com/javax/imageio/ImageReader.html#abort()), [clearAbortRequest()](http://docs.google.com/javax/imageio/ImageReader.html#clearAbortRequest())

### clearAbortRequest

protected void **clearAbortRequest**()

Clears any previous abort request. After this method has been called, abortRequested will return false.

**See Also:**[abort()](http://docs.google.com/javax/imageio/ImageReader.html#abort()), [abortRequested()](http://docs.google.com/javax/imageio/ImageReader.html#abortRequested())

### addIIOReadWarningListener

public void **addIIOReadWarningListener**([IIOReadWarningListener](http://docs.google.com/javax/imageio/event/IIOReadWarningListener.html) listener)

Adds an IIOReadWarningListener to the list of registered warning listeners. If listener is null, no exception will be thrown and no action will be taken. Messages sent to the given listener will be localized, if possible, to match the current Locale. If no Locale has been set, warning messages may be localized as the reader sees fit.

**Parameters:**listener - an IIOReadWarningListener to be registered.**See Also:**[removeIIOReadWarningListener(javax.imageio.event.IIOReadWarningListener)](http://docs.google.com/javax/imageio/ImageReader.html#removeIIOReadWarningListener(javax.imageio.event.IIOReadWarningListener))

### removeIIOReadWarningListener

public void **removeIIOReadWarningListener**([IIOReadWarningListener](http://docs.google.com/javax/imageio/event/IIOReadWarningListener.html) listener)

Removes an IIOReadWarningListener from the list of registered error listeners. If the listener was not previously registered, or if listener is null, no exception will be thrown and no action will be taken.

**Parameters:**listener - an IIOReadWarningListener to be unregistered.**See Also:**[addIIOReadWarningListener(javax.imageio.event.IIOReadWarningListener)](http://docs.google.com/javax/imageio/ImageReader.html#addIIOReadWarningListener(javax.imageio.event.IIOReadWarningListener))

### removeAllIIOReadWarningListeners

public void **removeAllIIOReadWarningListeners**()

Removes all currently registered IIOReadWarningListener objects.

The default implementation sets the warningListeners and warningLocales instance variables to null.

### addIIOReadProgressListener

public void **addIIOReadProgressListener**([IIOReadProgressListener](http://docs.google.com/javax/imageio/event/IIOReadProgressListener.html) listener)

Adds an IIOReadProgressListener to the list of registered progress listeners. If listener is null, no exception will be thrown and no action will be taken.

**Parameters:**listener - an IIOReadProgressListener to be registered.**See Also:**[removeIIOReadProgressListener(javax.imageio.event.IIOReadProgressListener)](http://docs.google.com/javax/imageio/ImageReader.html#removeIIOReadProgressListener(javax.imageio.event.IIOReadProgressListener))

### removeIIOReadProgressListener

public void **removeIIOReadProgressListener**([IIOReadProgressListener](http://docs.google.com/javax/imageio/event/IIOReadProgressListener.html) listener)

Removes an IIOReadProgressListener from the list of registered progress listeners. If the listener was not previously registered, or if listener is null, no exception will be thrown and no action will be taken.

**Parameters:**listener - an IIOReadProgressListener to be unregistered.**See Also:**[addIIOReadProgressListener(javax.imageio.event.IIOReadProgressListener)](http://docs.google.com/javax/imageio/ImageReader.html#addIIOReadProgressListener(javax.imageio.event.IIOReadProgressListener))

### removeAllIIOReadProgressListeners

public void **removeAllIIOReadProgressListeners**()

Removes all currently registered IIOReadProgressListener objects.

The default implementation sets the progressListeners instance variable to null.

### addIIOReadUpdateListener

public void **addIIOReadUpdateListener**([IIOReadUpdateListener](http://docs.google.com/javax/imageio/event/IIOReadUpdateListener.html) listener)

Adds an IIOReadUpdateListener to the list of registered update listeners. If listener is null, no exception will be thrown and no action will be taken. The listener will receive notification of pixel updates as images and thumbnails are decoded, including the starts and ends of progressive passes.

If no update listeners are present, the reader may choose to perform fewer updates to the pixels of the destination images and/or thumbnails, which may result in more efficient decoding.

For example, in progressive JPEG decoding each pass contains updates to a set of coefficients, which would have to be transformed into pixel values and converted to an RGB color space for each pass if listeners are present. If no listeners are present, the coefficients may simply be accumulated and the final results transformed and color converted one time only.

The final results of decoding will be the same whether or not intermediate updates are performed. Thus if only the final image is desired it may be perferable not to register any IIOReadUpdateListeners. In general, progressive updating is most effective when fetching images over a network connection that is very slow compared to local CPU processing; over a fast connection, progressive updates may actually slow down the presentation of the image.

**Parameters:**listener - an IIOReadUpdateListener to be registered.**See Also:**[removeIIOReadUpdateListener(javax.imageio.event.IIOReadUpdateListener)](http://docs.google.com/javax/imageio/ImageReader.html#removeIIOReadUpdateListener(javax.imageio.event.IIOReadUpdateListener))

### removeIIOReadUpdateListener

public void **removeIIOReadUpdateListener**([IIOReadUpdateListener](http://docs.google.com/javax/imageio/event/IIOReadUpdateListener.html) listener)

Removes an IIOReadUpdateListener from the list of registered update listeners. If the listener was not previously registered, or if listener is null, no exception will be thrown and no action will be taken.

**Parameters:**listener - an IIOReadUpdateListener to be unregistered.**See Also:**[addIIOReadUpdateListener(javax.imageio.event.IIOReadUpdateListener)](http://docs.google.com/javax/imageio/ImageReader.html#addIIOReadUpdateListener(javax.imageio.event.IIOReadUpdateListener))

### removeAllIIOReadUpdateListeners

public void **removeAllIIOReadUpdateListeners**()

Removes all currently registered IIOReadUpdateListener objects.

The default implementation sets the updateListeners instance variable to null.

### processSequenceStarted

protected void **processSequenceStarted**(int minIndex)

Broadcasts the start of an sequence of image reads to all registered IIOReadProgressListeners by calling their sequenceStarted method. Subclasses may use this method as a convenience.

**Parameters:**minIndex - the lowest index being read.

### processSequenceComplete

protected void **processSequenceComplete**()

Broadcasts the completion of an sequence of image reads to all registered IIOReadProgressListeners by calling their sequenceComplete method. Subclasses may use this method as a convenience.

### processImageStarted

protected void **processImageStarted**(int imageIndex)

Broadcasts the start of an image read to all registered IIOReadProgressListeners by calling their imageStarted method. Subclasses may use this method as a convenience.

**Parameters:**imageIndex - the index of the image about to be read.

### processImageProgress

protected void **processImageProgress**(float percentageDone)

Broadcasts the current percentage of image completion to all registered IIOReadProgressListeners by calling their imageProgress method. Subclasses may use this method as a convenience.

**Parameters:**percentageDone - the current percentage of completion, as a float.

### processImageComplete

protected void **processImageComplete**()

Broadcasts the completion of an image read to all registered IIOReadProgressListeners by calling their imageComplete method. Subclasses may use this method as a convenience.

### processThumbnailStarted

protected void **processThumbnailStarted**(int imageIndex,  
 int thumbnailIndex)

Broadcasts the start of a thumbnail read to all registered IIOReadProgressListeners by calling their thumbnailStarted method. Subclasses may use this method as a convenience.

**Parameters:**imageIndex - the index of the image associated with the thumbnail.thumbnailIndex - the index of the thumbnail.

### processThumbnailProgress

protected void **processThumbnailProgress**(float percentageDone)

Broadcasts the current percentage of thumbnail completion to all registered IIOReadProgressListeners by calling their thumbnailProgress method. Subclasses may use this method as a convenience.

**Parameters:**percentageDone - the current percentage of completion, as a float.

### processThumbnailComplete

protected void **processThumbnailComplete**()

Broadcasts the completion of a thumbnail read to all registered IIOReadProgressListeners by calling their thumbnailComplete method. Subclasses may use this method as a convenience.

### processReadAborted

protected void **processReadAborted**()

Broadcasts that the read has been aborted to all registered IIOReadProgressListeners by calling their readAborted method. Subclasses may use this method as a convenience.

### processPassStarted

protected void **processPassStarted**([BufferedImage](http://docs.google.com/java/awt/image/BufferedImage.html) theImage,  
 int pass,  
 int minPass,  
 int maxPass,  
 int minX,  
 int minY,  
 int periodX,  
 int periodY,  
 int[] bands)

Broadcasts the beginning of a progressive pass to all registered IIOReadUpdateListeners by calling their passStarted method. Subclasses may use this method as a convenience.

**Parameters:**theImage - the BufferedImage being updated.pass - the index of the current pass, starting with 0.minPass - the index of the first pass that will be decoded.maxPass - the index of the last pass that will be decoded.minX - the X coordinate of the upper-left pixel included in the pass.minY - the X coordinate of the upper-left pixel included in the pass.periodX - the horizontal separation between pixels.periodY - the vertical separation between pixels.bands - an array of ints indicating the set of affected bands of the destination.

### processImageUpdate

protected void **processImageUpdate**([BufferedImage](http://docs.google.com/java/awt/image/BufferedImage.html) theImage,  
 int minX,  
 int minY,  
 int width,  
 int height,  
 int periodX,  
 int periodY,  
 int[] bands)

Broadcasts the update of a set of samples to all registered IIOReadUpdateListeners by calling their imageUpdate method. Subclasses may use this method as a convenience.

**Parameters:**theImage - the BufferedImage being updated.minX - the X coordinate of the upper-left pixel included in the pass.minY - the X coordinate of the upper-left pixel included in the pass.width - the total width of the area being updated, including pixels being skipped if periodX > 1.height - the total height of the area being updated, including pixels being skipped if periodY > 1.periodX - the horizontal separation between pixels.periodY - the vertical separation between pixels.bands - an array of ints indicating the set of affected bands of the destination.

### processPassComplete

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

Broadcasts the end of a progressive pass to all registered IIOReadUpdateListeners by calling their passComplete method. Subclasses may use this method as a convenience.

**Parameters:**theImage - the BufferedImage being updated.

### processThumbnailPassStarted

protected void **processThumbnailPassStarted**([BufferedImage](http://docs.google.com/java/awt/image/BufferedImage.html) theThumbnail,  
 int pass,  
 int minPass,  
 int maxPass,  
 int minX,  
 int minY,  
 int periodX,  
 int periodY,  
 int[] bands)

Broadcasts the beginning of a thumbnail progressive pass to all registered IIOReadUpdateListeners by calling their thumbnailPassStarted method. Subclasses may use this method as a convenience.

**Parameters:**theThumbnail - the BufferedImage thumbnail being updated.pass - the index of the current pass, starting with 0.minPass - the index of the first pass that will be decoded.maxPass - the index of the last pass that will be decoded.minX - the X coordinate of the upper-left pixel included in the pass.minY - the X coordinate of the upper-left pixel included in the pass.periodX - the horizontal separation between pixels.periodY - the vertical separation between pixels.bands - an array of ints indicating the set of affected bands of the destination.

### processThumbnailUpdate

protected void **processThumbnailUpdate**([BufferedImage](http://docs.google.com/java/awt/image/BufferedImage.html) theThumbnail,  
 int minX,  
 int minY,  
 int width,  
 int height,  
 int periodX,  
 int periodY,  
 int[] bands)

Broadcasts the update of a set of samples in a thumbnail image to all registered IIOReadUpdateListeners by calling their thumbnailUpdate method. Subclasses may use this method as a convenience.

**Parameters:**theThumbnail - the BufferedImage thumbnail being updated.minX - the X coordinate of the upper-left pixel included in the pass.minY - the X coordinate of the upper-left pixel included in the pass.width - the total width of the area being updated, including pixels being skipped if periodX > 1.height - the total height of the area being updated, including pixels being skipped if periodY > 1.periodX - the horizontal separation between pixels.periodY - the vertical separation between pixels.bands - an array of ints indicating the set of affected bands of the destination.

### processThumbnailPassComplete

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

Broadcasts the end of a thumbnail progressive pass to all registered IIOReadUpdateListeners by calling their thumbnailPassComplete method. Subclasses may use this method as a convenience.

**Parameters:**theThumbnail - the BufferedImage thumbnail being updated.

### processWarningOccurred

protected void **processWarningOccurred**([String](http://docs.google.com/java/lang/String.html) warning)

Broadcasts a warning message to all registered IIOReadWarningListeners by calling their warningOccurred method. Subclasses may use this method as a convenience.

**Parameters:**warning - the warning message to send. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if warning is null.

### processWarningOccurred

protected void **processWarningOccurred**([String](http://docs.google.com/java/lang/String.html) baseName,  
 [String](http://docs.google.com/java/lang/String.html) keyword)

Broadcasts a localized warning message to all registered IIOReadWarningListeners by calling their warningOccurred method with a string taken from a ResourceBundle. Subclasses may use this method as a convenience.

**Parameters:**baseName - the base name of a set of ResourceBundles containing localized warning messages.keyword - the keyword used to index the warning message within the set of ResourceBundles. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if baseName is null. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if keyword is null. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if no appropriate ResourceBundle may be located. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the named resource is not found in the located ResourceBundle. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the object retrieved from the ResourceBundle is not a String.

### reset

public void **reset**()

Restores the ImageReader to its initial state.

The default implementation calls setInput(null, false), setLocale(null), removeAllIIOReadUpdateListeners(), removeAllIIOReadWarningListeners(), removeAllIIOReadProgressListeners(), and clearAbortRequest.

### dispose

public void **dispose**()

Allows any resources held by this object to be released. The result of calling any other method (other than finalize) subsequent to a call to this method is undefined.

It is important for applications to call this method when they know they will no longer be using this ImageReader. Otherwise, the reader may continue to hold on to resources indefinitely.

The default implementation of this method in the superclass does nothing. Subclass implementations should ensure that all resources, especially native resources, are released.

### getSourceRegion

protected static [Rectangle](http://docs.google.com/java/awt/Rectangle.html) **getSourceRegion**([ImageReadParam](http://docs.google.com/javax/imageio/ImageReadParam.html) param,  
 int srcWidth,  
 int srcHeight)

A utility method that may be used by readers to compute the region of the source image that should be read, taking into account any source region and subsampling offset settings in the supplied ImageReadParam. The actual subsampling factors, destination size, and destination offset are *not* taken into consideration, thus further clipping must take place. The [computeRegions](http://docs.google.com/javax/imageio/ImageReader.html#computeRegions(javax.imageio.ImageReadParam,%20int,%20int,%20java.awt.image.BufferedImage,%20java.awt.Rectangle,%20java.awt.Rectangle)) method performs all necessary clipping.

**Parameters:**param - the ImageReadParam being used, or null.srcWidth - the width of the source image.srcHeight - the height of the source image. **Returns:**the source region as a Rectangle.

### computeRegions

protected static void **computeRegions**([ImageReadParam](http://docs.google.com/javax/imageio/ImageReadParam.html) param,  
 int srcWidth,  
 int srcHeight,  
 [BufferedImage](http://docs.google.com/java/awt/image/BufferedImage.html) image,  
 [Rectangle](http://docs.google.com/java/awt/Rectangle.html) srcRegion,  
 [Rectangle](http://docs.google.com/java/awt/Rectangle.html) destRegion)

Computes the source region of interest and the destination region of interest, taking the width and height of the source image, an optional destination image, and an optional ImageReadParam into account. The source region begins with the entire source image. Then that is clipped to the source region specified in the ImageReadParam, if one is specified.

If either of the destination offsets are negative, the source region is clipped so that its top left will coincide with the top left of the destination image, taking subsampling into account. Then the result is clipped to the destination image on the right and bottom, if one is specified, taking subsampling and destination offsets into account.

Similarly, the destination region begins with the source image, is translated to the destination offset given in the ImageReadParam if there is one, and finally is clipped to the destination image, if there is one.

If either the source or destination regions end up having a width or height of 0, an IllegalArgumentException is thrown.

The [getSourceRegion](http://docs.google.com/javax/imageio/ImageReader.html#getSourceRegion(javax.imageio.ImageReadParam,%20int,%20int)) method may be used if only source clipping is desired.

**Parameters:**param - an ImageReadParam, or null.srcWidth - the width of the source image.srcHeight - the height of the source image.image - a BufferedImage that will be the destination image, or null.srcRegion - a Rectangle that will be filled with the source region of interest.destRegion - a Rectangle that will be filled with the destination region of interest. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if srcRegion is null. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if dstRegion is null. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the resulting source or destination region is empty.

### checkReadParamBandSettings

protected static void **checkReadParamBandSettings**([ImageReadParam](http://docs.google.com/javax/imageio/ImageReadParam.html) param,  
 int numSrcBands,  
 int numDstBands)

A utility method that may be used by readers to test the validity of the source and destination band settings of an ImageReadParam. This method may be called as soon as the reader knows both the number of bands of the source image as it exists in the input stream, and the number of bands of the destination image that being written.

The method retrieves the source and destination band setting arrays from param using the getSourceBands and getDestinationBandsmethods (or considers them to be null if param is null). If the source band setting array is null, it is considered to be equal to the array { 0, 1, ..., numSrcBands - 1 }, and similarly for the destination band setting array.

The method then tests that both arrays are equal in length, and that neither array contains a value larger than the largest available band index.

Any failure results in an IllegalArgumentException being thrown; success results in the method returning silently.

**Parameters:**param - the ImageReadParam being used to read the image.numSrcBands - the number of bands of the image as it exists int the input source.numDstBands - the number of bands in the destination image being written. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if param contains an invalid specification of a source and/or destination band subset.

### getDestination

protected static [BufferedImage](http://docs.google.com/java/awt/image/BufferedImage.html) **getDestination**([ImageReadParam](http://docs.google.com/javax/imageio/ImageReadParam.html) param,  
 [Iterator](http://docs.google.com/java/util/Iterator.html)<[ImageTypeSpecifier](http://docs.google.com/javax/imageio/ImageTypeSpecifier.html)> imageTypes,  
 int width,  
 int height)  
 throws [IIOException](http://docs.google.com/javax/imageio/IIOException.html)

Returns the BufferedImage to which decoded pixel data should be written. The image is determined by inspecting the supplied ImageReadParam if it is non-null; if its getDestination method returns a non-null value, that image is simply returned. Otherwise, param.getDestinationType method is called to determine if a particular image type has been specified. If so, the returned ImageTypeSpecifier is used after checking that it is equal to one of those included in imageTypes.

If param is null or the above steps have not yielded an image or an ImageTypeSpecifier, the first value obtained from the imageTypes parameter is used. Typically, the caller will set imageTypes to the value of getImageTypes(imageIndex).

Next, the dimensions of the image are determined by a call to computeRegions. The actual width and height of the image being decoded are passed in as the width and height parameters.

**Parameters:**param - an ImageReadParam to be used to get the destination image or image type, or null.imageTypes - an Iterator of ImageTypeSpecifiers indicating the legal image types, with the default first.width - the true width of the image or tile begin decoded.height - the true width of the image or tile being decoded. **Returns:**the BufferedImage to which decoded pixel data should be written. **Throws:** [IIOException](http://docs.google.com/javax/imageio/IIOException.html) - if the ImageTypeSpecifier specified by param does not match any of the legal ones from imageTypes. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if imageTypes is null or empty, or if an object not of type ImageTypeSpecifier is retrieved from it. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the resulting image would have a width or height less than 1. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the product of width and height is greater than Integer.MAX\_VALUE.

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/ImageReader.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/ImageIO.html)   [**NEXT CLASS**](http://docs.google.com/javax/imageio/ImageReadParam.html) | [**FRAMES**](http://docs.google.com/index.html?javax/imageio/ImageReader.html)    [**NO FRAMES**](http://docs.google.com/ImageReader.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | [CONSTR](#2et92p0) | [METHOD](#tyjcwt) | DETAIL: [FIELD](#1t3h5sf) | [CONSTR](#3j2qqm3) | [METHOD](#4i7ojhp) |

[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).