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

## **javax.imageio**

Class ImageIO

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

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

A class containing static convenience methods for locating ImageReaders and ImageWriters, and performing simple encoding and decoding.

| **Method Summary** | |
| --- | --- |
| static [ImageInputStream](http://docs.google.com/javax/imageio/stream/ImageInputStream.html) | [**createImageInputStream**](http://docs.google.com/javax/imageio/ImageIO.html#createImageInputStream(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) input)            Returns an ImageInputStream that will take its input from the given Object. |
| static [ImageOutputStream](http://docs.google.com/javax/imageio/stream/ImageOutputStream.html) | [**createImageOutputStream**](http://docs.google.com/javax/imageio/ImageIO.html#createImageOutputStream(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) output)            Returns an ImageOutputStream that will send its output to the given Object. |
| static [File](http://docs.google.com/java/io/File.html) | [**getCacheDirectory**](http://docs.google.com/javax/imageio/ImageIO.html#getCacheDirectory())()            Returns the current value set by setCacheDirectory, or null if no explicit setting has been made. |
| static [ImageReader](http://docs.google.com/javax/imageio/ImageReader.html) | [**getImageReader**](http://docs.google.com/javax/imageio/ImageIO.html#getImageReader(javax.imageio.ImageWriter))([ImageWriter](http://docs.google.com/javax/imageio/ImageWriter.html) writer)            Returns an ImageReadercorresponding to the given ImageWriter, if there is one, or null if the plug-in for this ImageWriter does not specify a corresponding ImageReader, or if the given ImageWriter is not registered. |
| static [Iterator](http://docs.google.com/java/util/Iterator.html)<[ImageReader](http://docs.google.com/javax/imageio/ImageReader.html)> | [**getImageReaders**](http://docs.google.com/javax/imageio/ImageIO.html#getImageReaders(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) input)            Returns an Iterator containing all currently registered ImageReaders that claim to be able to decode the supplied Object, typically an ImageInputStream. |
| static [Iterator](http://docs.google.com/java/util/Iterator.html)<[ImageReader](http://docs.google.com/javax/imageio/ImageReader.html)> | [**getImageReadersByFormatName**](http://docs.google.com/javax/imageio/ImageIO.html#getImageReadersByFormatName(java.lang.String))([String](http://docs.google.com/java/lang/String.html) formatName)            Returns an Iterator containing all currently registered ImageReaders that claim to be able to decode the named format. |
| static [Iterator](http://docs.google.com/java/util/Iterator.html)<[ImageReader](http://docs.google.com/javax/imageio/ImageReader.html)> | [**getImageReadersByMIMEType**](http://docs.google.com/javax/imageio/ImageIO.html#getImageReadersByMIMEType(java.lang.String))([String](http://docs.google.com/java/lang/String.html) MIMEType)            Returns an Iterator containing all currently registered ImageReaders that claim to be able to decode files with the given MIME type. |
| static [Iterator](http://docs.google.com/java/util/Iterator.html)<[ImageReader](http://docs.google.com/javax/imageio/ImageReader.html)> | [**getImageReadersBySuffix**](http://docs.google.com/javax/imageio/ImageIO.html#getImageReadersBySuffix(java.lang.String))([String](http://docs.google.com/java/lang/String.html) fileSuffix)            Returns an Iterator containing all currently registered ImageReaders that claim to be able to decode files with the given suffix. |
| static [Iterator](http://docs.google.com/java/util/Iterator.html)<[ImageTranscoder](http://docs.google.com/javax/imageio/ImageTranscoder.html)> | [**getImageTranscoders**](http://docs.google.com/javax/imageio/ImageIO.html#getImageTranscoders(javax.imageio.ImageReader,%20javax.imageio.ImageWriter))([ImageReader](http://docs.google.com/javax/imageio/ImageReader.html) reader, [ImageWriter](http://docs.google.com/javax/imageio/ImageWriter.html) writer)            Returns an Iterator containing all currently registered ImageTranscoders that claim to be able to transcode between the metadata of the given ImageReader and ImageWriter. |
| static [ImageWriter](http://docs.google.com/javax/imageio/ImageWriter.html) | [**getImageWriter**](http://docs.google.com/javax/imageio/ImageIO.html#getImageWriter(javax.imageio.ImageReader))([ImageReader](http://docs.google.com/javax/imageio/ImageReader.html) reader)            Returns an ImageWritercorresponding to the given ImageReader, if there is one, or null if the plug-in for this ImageReader does not specify a corresponding ImageWriter, or if the given ImageReader is not registered. |
| static [Iterator](http://docs.google.com/java/util/Iterator.html)<[ImageWriter](http://docs.google.com/javax/imageio/ImageWriter.html)> | [**getImageWriters**](http://docs.google.com/javax/imageio/ImageIO.html#getImageWriters(javax.imageio.ImageTypeSpecifier,%20java.lang.String))([ImageTypeSpecifier](http://docs.google.com/javax/imageio/ImageTypeSpecifier.html) type, [String](http://docs.google.com/java/lang/String.html) formatName)            Returns an Iterator containing all currently registered ImageWriters that claim to be able to encode images of the given layout (specified using an ImageTypeSpecifier) in the given format. |
| static [Iterator](http://docs.google.com/java/util/Iterator.html)<[ImageWriter](http://docs.google.com/javax/imageio/ImageWriter.html)> | [**getImageWritersByFormatName**](http://docs.google.com/javax/imageio/ImageIO.html#getImageWritersByFormatName(java.lang.String))([String](http://docs.google.com/java/lang/String.html) formatName)            Returns an Iterator containing all currently registered ImageWriters that claim to be able to encode the named format. |
| static [Iterator](http://docs.google.com/java/util/Iterator.html)<[ImageWriter](http://docs.google.com/javax/imageio/ImageWriter.html)> | [**getImageWritersByMIMEType**](http://docs.google.com/javax/imageio/ImageIO.html#getImageWritersByMIMEType(java.lang.String))([String](http://docs.google.com/java/lang/String.html) MIMEType)            Returns an Iterator containing all currently registered ImageWriters that claim to be able to encode files with the given MIME type. |
| static [Iterator](http://docs.google.com/java/util/Iterator.html)<[ImageWriter](http://docs.google.com/javax/imageio/ImageWriter.html)> | [**getImageWritersBySuffix**](http://docs.google.com/javax/imageio/ImageIO.html#getImageWritersBySuffix(java.lang.String))([String](http://docs.google.com/java/lang/String.html) fileSuffix)            Returns an Iterator containing all currently registered ImageWriters that claim to be able to encode files with the given suffix. |
| static [String](http://docs.google.com/java/lang/String.html)[] | [**getReaderFileSuffixes**](http://docs.google.com/javax/imageio/ImageIO.html#getReaderFileSuffixes())()            Returns an array of Strings listing all of the file suffixes associated with the formats understood by the current set of registered readers. |
| static [String](http://docs.google.com/java/lang/String.html)[] | [**getReaderFormatNames**](http://docs.google.com/javax/imageio/ImageIO.html#getReaderFormatNames())()            Returns an array of Strings listing all of the informal format names understood by the current set of registered readers. |
| static [String](http://docs.google.com/java/lang/String.html)[] | [**getReaderMIMETypes**](http://docs.google.com/javax/imageio/ImageIO.html#getReaderMIMETypes())()            Returns an array of Strings listing all of the MIME types understood by the current set of registered readers. |
| static boolean | [**getUseCache**](http://docs.google.com/javax/imageio/ImageIO.html#getUseCache())()            Returns the current value set by setUseCache, or true if no explicit setting has been made. |
| static [String](http://docs.google.com/java/lang/String.html)[] | [**getWriterFileSuffixes**](http://docs.google.com/javax/imageio/ImageIO.html#getWriterFileSuffixes())()            Returns an array of Strings listing all of the file suffixes associated with the formats understood by the current set of registered writers. |
| static [String](http://docs.google.com/java/lang/String.html)[] | [**getWriterFormatNames**](http://docs.google.com/javax/imageio/ImageIO.html#getWriterFormatNames())()            Returns an array of Strings listing all of the informal format names understood by the current set of registered writers. |
| static [String](http://docs.google.com/java/lang/String.html)[] | [**getWriterMIMETypes**](http://docs.google.com/javax/imageio/ImageIO.html#getWriterMIMETypes())()            Returns an array of Strings listing all of the MIME types understood by the current set of registered writers. |
| static [BufferedImage](http://docs.google.com/java/awt/image/BufferedImage.html) | [**read**](http://docs.google.com/javax/imageio/ImageIO.html#read(java.io.File))([File](http://docs.google.com/java/io/File.html) input)            Returns a BufferedImage as the result of decoding a supplied File with an ImageReader chosen automatically from among those currently registered. |
| static [BufferedImage](http://docs.google.com/java/awt/image/BufferedImage.html) | [**read**](http://docs.google.com/javax/imageio/ImageIO.html#read(javax.imageio.stream.ImageInputStream))([ImageInputStream](http://docs.google.com/javax/imageio/stream/ImageInputStream.html) stream)            Returns a BufferedImage as the result of decoding a supplied ImageInputStream with an ImageReader chosen automatically from among those currently registered. |
| static [BufferedImage](http://docs.google.com/java/awt/image/BufferedImage.html) | [**read**](http://docs.google.com/javax/imageio/ImageIO.html#read(java.io.InputStream))([InputStream](http://docs.google.com/java/io/InputStream.html) input)            Returns a BufferedImage as the result of decoding a supplied InputStream with an ImageReader chosen automatically from among those currently registered. |
| static [BufferedImage](http://docs.google.com/java/awt/image/BufferedImage.html) | [**read**](http://docs.google.com/javax/imageio/ImageIO.html#read(java.net.URL))([URL](http://docs.google.com/java/net/URL.html) input)            Returns a BufferedImage as the result of decoding a supplied URL with an ImageReader chosen automatically from among those currently registered. |
| static void | [**scanForPlugins**](http://docs.google.com/javax/imageio/ImageIO.html#scanForPlugins())()            Scans for plug-ins on the application class path, loads their service provider classes, and registers a service provider instance for each one found with the IIORegistry. |
| static void | [**setCacheDirectory**](http://docs.google.com/javax/imageio/ImageIO.html#setCacheDirectory(java.io.File))([File](http://docs.google.com/java/io/File.html) cacheDirectory)            Sets the directory where cache files are to be created. |
| static void | [**setUseCache**](http://docs.google.com/javax/imageio/ImageIO.html#setUseCache(boolean))(boolean useCache)            Sets a flag indicating whether a disk-based cache file should be used when creating ImageInputStreams and ImageOutputStreams. |
| static boolean | [**write**](http://docs.google.com/javax/imageio/ImageIO.html#write(java.awt.image.RenderedImage,%20java.lang.String,%20java.io.File))([RenderedImage](http://docs.google.com/java/awt/image/RenderedImage.html) im, [String](http://docs.google.com/java/lang/String.html) formatName, [File](http://docs.google.com/java/io/File.html) output)            Writes an image using an arbitrary ImageWriter that supports the given format to a File. |
| static boolean | [**write**](http://docs.google.com/javax/imageio/ImageIO.html#write(java.awt.image.RenderedImage,%20java.lang.String,%20javax.imageio.stream.ImageOutputStream))([RenderedImage](http://docs.google.com/java/awt/image/RenderedImage.html) im, [String](http://docs.google.com/java/lang/String.html) formatName, [ImageOutputStream](http://docs.google.com/javax/imageio/stream/ImageOutputStream.html) output)            Writes an image using the an arbitrary ImageWriter that supports the given format to an ImageOutputStream. |
| static boolean | [**write**](http://docs.google.com/javax/imageio/ImageIO.html#write(java.awt.image.RenderedImage,%20java.lang.String,%20java.io.OutputStream))([RenderedImage](http://docs.google.com/java/awt/image/RenderedImage.html) im, [String](http://docs.google.com/java/lang/String.html) formatName, [OutputStream](http://docs.google.com/java/io/OutputStream.html) output)            Writes an image using an arbitrary ImageWriter that supports the given format to an OutputStream. |

| **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)) |

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

### scanForPlugins

public static void **scanForPlugins**()

Scans for plug-ins on the application class path, loads their service provider classes, and registers a service provider instance for each one found with the IIORegistry.

This method is needed because the application class path can theoretically change, or additional plug-ins may become available. Rather than re-scanning the classpath on every invocation of the API, the class path is scanned automatically only on the first invocation. Clients can call this method to prompt a re-scan. Thus this method need only be invoked by sophisticated applications which dynamically make new plug-ins available at runtime.

The getResources method of the context ClassLoader is used locate JAR files containing files named META-INF/services/javax.imageio.spi.*classname*, where *classname* is one of ImageReaderSpi, ImageWriterSpi, ImageTranscoderSpi, ImageInputStreamSpi, or ImageOutputStreamSpi, along the application class path.

The contents of the located files indicate the names of actual implementation classes which implement the aforementioned service provider interfaces; the default class loader is then used to load each of these classes and to instantiate an instance of each class, which is then placed into the registry for later retrieval.

The exact set of locations searched depends on the implementation of the Java runtime enviroment.

**See Also:**[ClassLoader.getResources(java.lang.String)](http://docs.google.com/java/lang/ClassLoader.html#getResources(java.lang.String))

### setUseCache

public static void **setUseCache**(boolean useCache)

Sets a flag indicating whether a disk-based cache file should be used when creating ImageInputStreams and ImageOutputStreams.

When reading from a standard InputStream>, it may be necessary to save previously read information in a cache since the underlying stream does not allow data to be re-read. Similarly, when writing to a standard OutputStream, a cache may be used to allow a previously written value to be changed before flushing it to the final destination.

The cache may reside in main memory or on disk. Setting this flag to false disallows the use of disk for future streams, which may be advantageous when working with small images, as the overhead of creating and destroying files is removed.

On startup, the value is set to true.

**Parameters:**useCache - a boolean indicating whether a cache file should be used, in cases where it is optional.**See Also:**[getUseCache()](http://docs.google.com/javax/imageio/ImageIO.html#getUseCache())

### getUseCache

public static boolean **getUseCache**()

Returns the current value set by setUseCache, or true if no explicit setting has been made.

**Returns:**true if a disk-based cache may be used for ImageInputStreams and ImageOutputStreams.**See Also:**[setUseCache(boolean)](http://docs.google.com/javax/imageio/ImageIO.html#setUseCache(boolean))

### setCacheDirectory

public static void **setCacheDirectory**([File](http://docs.google.com/java/io/File.html) cacheDirectory)

Sets the directory where cache files are to be created. A value of null indicates that the system-dependent default temporary-file directory is to be used. If getUseCache returns false, this value is ignored.

**Parameters:**cacheDirectory - a File specifying a directory. **Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if the security manager denies access to the directory. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if cacheDir is non-null but is not a directory.**See Also:**[File.createTempFile(String, String, File)](http://docs.google.com/java/io/File.html#createTempFile(java.lang.String,%20java.lang.String,%20java.io.File)), [getCacheDirectory()](http://docs.google.com/javax/imageio/ImageIO.html#getCacheDirectory())

### getCacheDirectory

public static [File](http://docs.google.com/java/io/File.html) **getCacheDirectory**()

Returns the current value set by setCacheDirectory, or null if no explicit setting has been made.

**Returns:**a File indicating the directory where cache files will be created, or null to indicate the system-dependent default temporary-file directory.**See Also:**[setCacheDirectory(java.io.File)](http://docs.google.com/javax/imageio/ImageIO.html#setCacheDirectory(java.io.File))

### createImageInputStream

public static [ImageInputStream](http://docs.google.com/javax/imageio/stream/ImageInputStream.html) **createImageInputStream**([Object](http://docs.google.com/java/lang/Object.html) input)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Returns an ImageInputStream that will take its input from the given Object. The set of ImageInputStreamSpis registered with the IIORegistry class is queried and the first one that is able to take input from the supplied object is used to create the returned ImageInputStream. If no suitable ImageInputStreamSpi exists, null is returned.

The current cache settings from getUseCacheand getCacheDirectory will be used to control caching.

**Parameters:**input - an Object to be used as an input source, such as a File, readable RandomAccessFile, or InputStream. **Returns:**an ImageInputStream, or null. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if input is null. [IOException](http://docs.google.com/java/io/IOException.html) - if a cache file is needed but cannot be created.**See Also:**[ImageInputStreamSpi](http://docs.google.com/javax/imageio/spi/ImageInputStreamSpi.html)

### createImageOutputStream

public static [ImageOutputStream](http://docs.google.com/javax/imageio/stream/ImageOutputStream.html) **createImageOutputStream**([Object](http://docs.google.com/java/lang/Object.html) output)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Returns an ImageOutputStream that will send its output to the given Object. The set of ImageOutputStreamSpis registered with the IIORegistry class is queried and the first one that is able to send output from the supplied object is used to create the returned ImageOutputStream. If no suitable ImageOutputStreamSpi exists, null is returned.

The current cache settings from getUseCacheand getCacheDirectory will be used to control caching.

**Parameters:**output - an Object to be used as an output destination, such as a File, writable RandomAccessFile, or OutputStream. **Returns:**an ImageOutputStream, or null. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if output is null. [IOException](http://docs.google.com/java/io/IOException.html) - if a cache file is needed but cannot be created.**See Also:**[ImageOutputStreamSpi](http://docs.google.com/javax/imageio/spi/ImageOutputStreamSpi.html)

### getReaderFormatNames

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

Returns an array of Strings listing all of the informal format names understood by the current set of registered readers.

**Returns:**an array of Strings.

### getReaderMIMETypes

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

Returns an array of Strings listing all of the MIME types understood by the current set of registered readers.

**Returns:**an array of Strings.

### getReaderFileSuffixes

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

Returns an array of Strings listing all of the file suffixes associated with the formats understood by the current set of registered readers.

**Returns:**an array of Strings.**Since:** 1.6

### getImageReaders

public static [Iterator](http://docs.google.com/java/util/Iterator.html)<[ImageReader](http://docs.google.com/javax/imageio/ImageReader.html)> **getImageReaders**([Object](http://docs.google.com/java/lang/Object.html) input)

Returns an Iterator containing all currently registered ImageReaders that claim to be able to decode the supplied Object, typically an ImageInputStream.

The stream position is left at its prior position upon exit from this method.

**Parameters:**input - an ImageInputStream or other Object containing encoded image data. **Returns:**an Iterator containing ImageReaders. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if input is null.**See Also:**[ImageReaderSpi.canDecodeInput(java.lang.Object)](http://docs.google.com/javax/imageio/spi/ImageReaderSpi.html#canDecodeInput(java.lang.Object))

### getImageReadersByFormatName

public static [Iterator](http://docs.google.com/java/util/Iterator.html)<[ImageReader](http://docs.google.com/javax/imageio/ImageReader.html)> **getImageReadersByFormatName**([String](http://docs.google.com/java/lang/String.html) formatName)

Returns an Iterator containing all currently registered ImageReaders that claim to be able to decode the named format.

**Parameters:**formatName - a String containing the informal name of a format (*e.g.*, "jpeg" or "tiff". **Returns:**an Iterator containing ImageReaders. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if formatName is null.**See Also:**[ImageReaderWriterSpi.getFormatNames()](http://docs.google.com/javax/imageio/spi/ImageReaderWriterSpi.html#getFormatNames())

### getImageReadersBySuffix

public static [Iterator](http://docs.google.com/java/util/Iterator.html)<[ImageReader](http://docs.google.com/javax/imageio/ImageReader.html)> **getImageReadersBySuffix**([String](http://docs.google.com/java/lang/String.html) fileSuffix)

Returns an Iterator containing all currently registered ImageReaders that claim to be able to decode files with the given suffix.

**Parameters:**fileSuffix - a String containing a file suffix (*e.g.*, "jpg" or "tiff"). **Returns:**an Iterator containing ImageReaders. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if fileSuffix is null.**See Also:**[ImageReaderWriterSpi.getFileSuffixes()](http://docs.google.com/javax/imageio/spi/ImageReaderWriterSpi.html#getFileSuffixes())

### getImageReadersByMIMEType

public static [Iterator](http://docs.google.com/java/util/Iterator.html)<[ImageReader](http://docs.google.com/javax/imageio/ImageReader.html)> **getImageReadersByMIMEType**([String](http://docs.google.com/java/lang/String.html) MIMEType)

Returns an Iterator containing all currently registered ImageReaders that claim to be able to decode files with the given MIME type.

**Parameters:**MIMEType - a String containing a file suffix (*e.g.*, "image/jpeg" or "image/x-bmp"). **Returns:**an Iterator containing ImageReaders. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if MIMEType is null.**See Also:**[ImageReaderWriterSpi.getMIMETypes()](http://docs.google.com/javax/imageio/spi/ImageReaderWriterSpi.html#getMIMETypes())

### getWriterFormatNames

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

Returns an array of Strings listing all of the informal format names understood by the current set of registered writers.

**Returns:**an array of Strings.

### getWriterMIMETypes

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

Returns an array of Strings listing all of the MIME types understood by the current set of registered writers.

**Returns:**an array of Strings.

### getWriterFileSuffixes

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

Returns an array of Strings listing all of the file suffixes associated with the formats understood by the current set of registered writers.

**Returns:**an array of Strings.**Since:** 1.6

### getImageWritersByFormatName

public static [Iterator](http://docs.google.com/java/util/Iterator.html)<[ImageWriter](http://docs.google.com/javax/imageio/ImageWriter.html)> **getImageWritersByFormatName**([String](http://docs.google.com/java/lang/String.html) formatName)

Returns an Iterator containing all currently registered ImageWriters that claim to be able to encode the named format.

**Parameters:**formatName - a String containing the informal name of a format (*e.g.*, "jpeg" or "tiff". **Returns:**an Iterator containing ImageWriters. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if formatName is null.**See Also:**[ImageReaderWriterSpi.getFormatNames()](http://docs.google.com/javax/imageio/spi/ImageReaderWriterSpi.html#getFormatNames())

### getImageWritersBySuffix

public static [Iterator](http://docs.google.com/java/util/Iterator.html)<[ImageWriter](http://docs.google.com/javax/imageio/ImageWriter.html)> **getImageWritersBySuffix**([String](http://docs.google.com/java/lang/String.html) fileSuffix)

Returns an Iterator containing all currently registered ImageWriters that claim to be able to encode files with the given suffix.

**Parameters:**fileSuffix - a String containing a file suffix (*e.g.*, "jpg" or "tiff"). **Returns:**an Iterator containing ImageWriters. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if fileSuffix is null.**See Also:**[ImageReaderWriterSpi.getFileSuffixes()](http://docs.google.com/javax/imageio/spi/ImageReaderWriterSpi.html#getFileSuffixes())

### getImageWritersByMIMEType

public static [Iterator](http://docs.google.com/java/util/Iterator.html)<[ImageWriter](http://docs.google.com/javax/imageio/ImageWriter.html)> **getImageWritersByMIMEType**([String](http://docs.google.com/java/lang/String.html) MIMEType)

Returns an Iterator containing all currently registered ImageWriters that claim to be able to encode files with the given MIME type.

**Parameters:**MIMEType - a String containing a file suffix (*e.g.*, "image/jpeg" or "image/x-bmp"). **Returns:**an Iterator containing ImageWriters. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if MIMEType is null.**See Also:**[ImageReaderWriterSpi.getMIMETypes()](http://docs.google.com/javax/imageio/spi/ImageReaderWriterSpi.html#getMIMETypes())

### getImageWriter

public static [ImageWriter](http://docs.google.com/javax/imageio/ImageWriter.html) **getImageWriter**([ImageReader](http://docs.google.com/javax/imageio/ImageReader.html) reader)

Returns an ImageWritercorresponding to the given ImageReader, if there is one, or null if the plug-in for this ImageReader does not specify a corresponding ImageWriter, or if the given ImageReader is not registered. This mechanism may be used to obtain an ImageWriter that will understand the internal structure of non-pixel metadata (as encoded by IIOMetadata objects) generated by the ImageReader. By obtaining this data from the ImageReader and passing it on to the ImageWriter obtained with this method, a client program can read an image, modify it in some way, and write it back out preserving all metadata, without having to understand anything about the structure of the metadata, or even about the image format. Note that this method returns the "preferred" writer, which is the first in the list returned by javax.imageio.spi.ImageReaderSpi.getImageWriterSpiNames().

**Parameters:**reader - an instance of a registered ImageReader. **Returns:**an ImageWriter, or null. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if reader is null.**See Also:**[getImageReader(ImageWriter)](http://docs.google.com/javax/imageio/ImageIO.html#getImageReader(javax.imageio.ImageWriter)), [ImageReaderSpi.getImageWriterSpiNames()](http://docs.google.com/javax/imageio/spi/ImageReaderSpi.html#getImageWriterSpiNames())

### getImageReader

public static [ImageReader](http://docs.google.com/javax/imageio/ImageReader.html) **getImageReader**([ImageWriter](http://docs.google.com/javax/imageio/ImageWriter.html) writer)

Returns an ImageReadercorresponding to the given ImageWriter, if there is one, or null if the plug-in for this ImageWriter does not specify a corresponding ImageReader, or if the given ImageWriter is not registered. This method is provided principally for symmetry with getImageWriter(ImageReader). Note that this method returns the "preferred" reader, which is the first in the list returned by javax.imageio.spi.ImageWriterSpi.getImageReaderSpiNames().

**Parameters:**writer - an instance of a registered ImageWriter. **Returns:**an ImageReader, or null. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if writer is null.**See Also:**[getImageWriter(ImageReader)](http://docs.google.com/javax/imageio/ImageIO.html#getImageWriter(javax.imageio.ImageReader)), [ImageWriterSpi.getImageReaderSpiNames()](http://docs.google.com/javax/imageio/spi/ImageWriterSpi.html#getImageReaderSpiNames())

### getImageWriters

public static [Iterator](http://docs.google.com/java/util/Iterator.html)<[ImageWriter](http://docs.google.com/javax/imageio/ImageWriter.html)> **getImageWriters**([ImageTypeSpecifier](http://docs.google.com/javax/imageio/ImageTypeSpecifier.html) type,  
 [String](http://docs.google.com/java/lang/String.html) formatName)

Returns an Iterator containing all currently registered ImageWriters that claim to be able to encode images of the given layout (specified using an ImageTypeSpecifier) in the given format.

**Parameters:**type - an ImageTypeSpecifier indicating the layout of the image to be written.formatName - the informal name of the format. **Returns:**an Iterator containing ImageWriters. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if any parameter is null.**See Also:**[ImageWriterSpi.canEncodeImage(ImageTypeSpecifier)](http://docs.google.com/javax/imageio/spi/ImageWriterSpi.html#canEncodeImage(javax.imageio.ImageTypeSpecifier))

### getImageTranscoders

public static [Iterator](http://docs.google.com/java/util/Iterator.html)<[ImageTranscoder](http://docs.google.com/javax/imageio/ImageTranscoder.html)> **getImageTranscoders**([ImageReader](http://docs.google.com/javax/imageio/ImageReader.html) reader,  
 [ImageWriter](http://docs.google.com/javax/imageio/ImageWriter.html) writer)

Returns an Iterator containing all currently registered ImageTranscoders that claim to be able to transcode between the metadata of the given ImageReader and ImageWriter.

**Parameters:**reader - an ImageReader.writer - an ImageWriter. **Returns:**an Iterator containing ImageTranscoders. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if reader or writer is null.

### read

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

Returns a BufferedImage as the result of decoding a supplied File with an ImageReader chosen automatically from among those currently registered. The File is wrapped in an ImageInputStream. If no registered ImageReader claims to be able to read the resulting stream, null is returned.

The current cache settings from getUseCacheand getCacheDirectory will be used to control caching in the ImageInputStream that is created.

Note that there is no read method that takes a filename as a String; use this method instead after creating a File from the filename.

This method does not attempt to locate ImageReaders that can read directly from a File; that may be accomplished using IIORegistry and ImageReaderSpi.

**Parameters:**input - a File to read from. **Returns:**a BufferedImage containing the decoded contents of the input, or null. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if input is null. [IOException](http://docs.google.com/java/io/IOException.html) - if an error occurs during reading.

### read

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

Returns a BufferedImage as the result of decoding a supplied InputStream with an ImageReader chosen automatically from among those currently registered. The InputStream is wrapped in an ImageInputStream. If no registered ImageReader claims to be able to read the resulting stream, null is returned.

The current cache settings from getUseCacheand getCacheDirectory will be used to control caching in the ImageInputStream that is created.

This method does not attempt to locate ImageReaders that can read directly from an InputStream; that may be accomplished using IIORegistry and ImageReaderSpi.

This method *does not* close the provided InputStream after the read operation has completed; it is the responsibility of the caller to close the stream, if desired.

**Parameters:**input - an InputStream to read from. **Returns:**a BufferedImage containing the decoded contents of the input, or null. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if input is null. [IOException](http://docs.google.com/java/io/IOException.html) - if an error occurs during reading.

### read

public static [BufferedImage](http://docs.google.com/java/awt/image/BufferedImage.html) **read**([URL](http://docs.google.com/java/net/URL.html) input)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Returns a BufferedImage as the result of decoding a supplied URL with an ImageReader chosen automatically from among those currently registered. An InputStream is obtained from the URL, which is wrapped in an ImageInputStream. If no registered ImageReader claims to be able to read the resulting stream, null is returned.

The current cache settings from getUseCacheand getCacheDirectory will be used to control caching in the ImageInputStream that is created.

This method does not attempt to locate ImageReaders that can read directly from a URL; that may be accomplished using IIORegistry and ImageReaderSpi.

**Parameters:**input - a URL to read from. **Returns:**a BufferedImage containing the decoded contents of the input, or null. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if input is null. [IOException](http://docs.google.com/java/io/IOException.html) - if an error occurs during reading.

### read

public static [BufferedImage](http://docs.google.com/java/awt/image/BufferedImage.html) **read**([ImageInputStream](http://docs.google.com/javax/imageio/stream/ImageInputStream.html) stream)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Returns a BufferedImage as the result of decoding a supplied ImageInputStream with an ImageReader chosen automatically from among those currently registered. If no registered ImageReader claims to be able to read the stream, null is returned.

Unlike most other methods in this class, this method *does* close the provided ImageInputStream after the read operation has completed, unless null is returned, in which case this method *does not* close the stream.

**Parameters:**stream - an ImageInputStream to read from. **Returns:**a BufferedImage containing the decoded contents of the input, or null. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if stream is null. [IOException](http://docs.google.com/java/io/IOException.html) - if an error occurs during reading.

### write

public static boolean **write**([RenderedImage](http://docs.google.com/java/awt/image/RenderedImage.html) im,  
 [String](http://docs.google.com/java/lang/String.html) formatName,  
 [ImageOutputStream](http://docs.google.com/javax/imageio/stream/ImageOutputStream.html) output)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Writes an image using the an arbitrary ImageWriter that supports the given format to an ImageOutputStream. The image is written to the ImageOutputStream starting at the current stream pointer, overwriting existing stream data from that point forward, if present.

This method *does not* close the provided ImageOutputStream after the write operation has completed; it is the responsibility of the caller to close the stream, if desired.

**Parameters:**im - a RenderedImage to be written.formatName - a String containg the informal name of the format.output - an ImageOutputStream to be written to. **Returns:**false if no appropriate writer is found. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if any parameter is null. [IOException](http://docs.google.com/java/io/IOException.html) - if an error occurs during writing.

### write

public static boolean **write**([RenderedImage](http://docs.google.com/java/awt/image/RenderedImage.html) im,  
 [String](http://docs.google.com/java/lang/String.html) formatName,  
 [File](http://docs.google.com/java/io/File.html) output)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Writes an image using an arbitrary ImageWriter that supports the given format to a File. If there is already a File present, its contents are discarded.

**Parameters:**im - a RenderedImage to be written.formatName - a String containg the informal name of the format.output - a File to be written to. **Returns:**false if no appropriate writer is found. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if any parameter is null. [IOException](http://docs.google.com/java/io/IOException.html) - if an error occurs during writing.

### write

public static boolean **write**([RenderedImage](http://docs.google.com/java/awt/image/RenderedImage.html) im,  
 [String](http://docs.google.com/java/lang/String.html) formatName,  
 [OutputStream](http://docs.google.com/java/io/OutputStream.html) output)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Writes an image using an arbitrary ImageWriter that supports the given format to an OutputStream.

This method *does not* close the provided OutputStream after the write operation has completed; it is the responsibility of the caller to close the stream, if desired.

The current cache settings from getUseCacheand getCacheDirectory will be used to control caching.

**Parameters:**im - a RenderedImage to be written.formatName - a String containg the informal name of the format.output - an OutputStream to be written to. **Returns:**false if no appropriate writer is found. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if any parameter is null. [IOException](http://docs.google.com/java/io/IOException.html) - if an error occurs during writing.

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

[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).