| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/JPEGImageWriteParam.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/plugins/jpeg/JPEGImageReadParam.html)   [**NEXT CLASS**](http://docs.google.com/javax/imageio/plugins/jpeg/JPEGQTable.html) | [**FRAMES**](http://docs.google.com/index.html?javax/imageio/plugins/jpeg/JPEGImageWriteParam.html)    [**NO FRAMES**](http://docs.google.com/JPEGImageWriteParam.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#2et92p0) | [CONSTR](#3dy6vkm) | [METHOD](#1t3h5sf) | DETAIL: FIELD | [CONSTR](#3rdcrjn) | [METHOD](#lnxbz9) |

## **javax.imageio.plugins.jpeg**

Class JPEGImageWriteParam

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 [javax.imageio.IIOParam](http://docs.google.com/javax/imageio/IIOParam.html)  
 [javax.imageio.ImageWriteParam](http://docs.google.com/javax/imageio/ImageWriteParam.html)  
 **javax.imageio.plugins.jpeg.JPEGImageWriteParam**

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

This class adds the ability to set JPEG quantization and Huffman tables when using the built-in JPEG writer plug-in, and to request that optimized Huffman tables be computed for an image. An instance of this class will be returned from the getDefaultImageWriteParam methods of the built-in JPEG ImageWriter.

The principal purpose of these additions is to allow the specification of tables to use in encoding abbreviated streams. The built-in JPEG writer will also accept an ordinary ImageWriteParam, in which case the writer will construct the necessary tables internally.

In either case, the quality setting in an ImageWriteParam has the same meaning as for the underlying library: 1.00 means a quantization table of all 1's, 0.75 means the "standard", visually lossless quantization table, and 0.00 means aquantization table of all 255's.

While tables for abbreviated streams are often specified by first writing an abbreviated stream containing only the tables, in some applications the tables are fixed ahead of time. This class allows the tables to be specified directly from client code.

Normally, the tables are specified in the IIOMetadata objects passed in to the writer, and any tables included in these objects are written to the stream. If no tables are specified in the metadata, then an abbreviated stream is written. If no tables are included in the metadata and no tables are specified in a JPEGImageWriteParam, then an abbreviated stream is encoded using the "standard" visually lossless tables. This class is necessary for specifying tables when an abbreviated stream must be written without writing any tables to a stream first. In order to use this class, the metadata object passed into the writer must contain no tables, and no stream metadata must be provided. See [JPEGQTable](http://docs.google.com/javax/imageio/plugins/jpeg/JPEGQTable.html) and [JPEGHuffmanTable](http://docs.google.com/javax/imageio/plugins/jpeg/JPEGHuffmanTable.html) for more information on the default tables.

The default JPEGImageWriteParam returned by the getDefaultWriteParam method of the writer contains no tables. Default tables are included in the default IIOMetadata objects returned by the writer.

If the metadata does contain tables, the tables given in a JPEGImageWriteParam are ignored. Furthermore, once a set of tables has been written, only tables in the metadata can override them for subsequent writes, whether to the same stream or a different one. In order to specify new tables using this class, the [reset](http://docs.google.com/javax/imageio/ImageWriter.html#reset()) method of the writer must be called.

For more information about the operation of the built-in JPEG plug-ins, see the [JPEG metadata format specification and usage notes](http://docs.google.com/metadata/doc-files/jpeg_metadata.html).

| **Field Summary** | |
| --- | --- |

| **Fields inherited from class javax.imageio.**[**ImageWriteParam**](http://docs.google.com/javax/imageio/ImageWriteParam.html) |
| --- |
| [canOffsetTiles](http://docs.google.com/javax/imageio/ImageWriteParam.html#canOffsetTiles), [canWriteCompressed](http://docs.google.com/javax/imageio/ImageWriteParam.html#canWriteCompressed), [canWriteProgressive](http://docs.google.com/javax/imageio/ImageWriteParam.html#canWriteProgressive), [canWriteTiles](http://docs.google.com/javax/imageio/ImageWriteParam.html#canWriteTiles), [compressionMode](http://docs.google.com/javax/imageio/ImageWriteParam.html#compressionMode), [compressionQuality](http://docs.google.com/javax/imageio/ImageWriteParam.html#compressionQuality), [compressionType](http://docs.google.com/javax/imageio/ImageWriteParam.html#compressionType), [compressionTypes](http://docs.google.com/javax/imageio/ImageWriteParam.html#compressionTypes), [locale](http://docs.google.com/javax/imageio/ImageWriteParam.html#locale), [MODE\_COPY\_FROM\_METADATA](http://docs.google.com/javax/imageio/ImageWriteParam.html#MODE_COPY_FROM_METADATA), [MODE\_DEFAULT](http://docs.google.com/javax/imageio/ImageWriteParam.html#MODE_DEFAULT), [MODE\_DISABLED](http://docs.google.com/javax/imageio/ImageWriteParam.html#MODE_DISABLED), [MODE\_EXPLICIT](http://docs.google.com/javax/imageio/ImageWriteParam.html#MODE_EXPLICIT), [preferredTileSizes](http://docs.google.com/javax/imageio/ImageWriteParam.html#preferredTileSizes), [progressiveMode](http://docs.google.com/javax/imageio/ImageWriteParam.html#progressiveMode), [tileGridXOffset](http://docs.google.com/javax/imageio/ImageWriteParam.html#tileGridXOffset), [tileGridYOffset](http://docs.google.com/javax/imageio/ImageWriteParam.html#tileGridYOffset), [tileHeight](http://docs.google.com/javax/imageio/ImageWriteParam.html#tileHeight), [tileWidth](http://docs.google.com/javax/imageio/ImageWriteParam.html#tileWidth), [tilingMode](http://docs.google.com/javax/imageio/ImageWriteParam.html#tilingMode), [tilingSet](http://docs.google.com/javax/imageio/ImageWriteParam.html#tilingSet) |

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

| **Constructor Summary** | |
| --- | --- |
| [**JPEGImageWriteParam**](http://docs.google.com/javax/imageio/plugins/jpeg/JPEGImageWriteParam.html#JPEGImageWriteParam(java.util.Locale))([Locale](http://docs.google.com/java/util/Locale.html) locale)            Constructs a JPEGImageWriteParam. |

| **Method Summary** | |
| --- | --- |
| boolean | [**areTablesSet**](http://docs.google.com/javax/imageio/plugins/jpeg/JPEGImageWriteParam.html#areTablesSet())()            Returns true if tables are currently set. |
| [JPEGHuffmanTable](http://docs.google.com/javax/imageio/plugins/jpeg/JPEGHuffmanTable.html)[] | [**getACHuffmanTables**](http://docs.google.com/javax/imageio/plugins/jpeg/JPEGImageWriteParam.html#getACHuffmanTables())()            Returns a copy of the array of AC Huffman tables set on the most recent call to setEncodeTables, or null if tables are not currently set. |
| [String](http://docs.google.com/java/lang/String.html)[] | [**getCompressionQualityDescriptions**](http://docs.google.com/javax/imageio/plugins/jpeg/JPEGImageWriteParam.html#getCompressionQualityDescriptions())()            Returns an array of Strings that may be used along with getCompressionQualityValues as part of a user interface for setting or displaying the compression quality level. |
| float[] | [**getCompressionQualityValues**](http://docs.google.com/javax/imageio/plugins/jpeg/JPEGImageWriteParam.html#getCompressionQualityValues())()            Returns an array of floats that may be used along with getCompressionQualityDescriptions as part of a user interface for setting or displaying the compression quality level. |
| [JPEGHuffmanTable](http://docs.google.com/javax/imageio/plugins/jpeg/JPEGHuffmanTable.html)[] | [**getDCHuffmanTables**](http://docs.google.com/javax/imageio/plugins/jpeg/JPEGImageWriteParam.html#getDCHuffmanTables())()            Returns a copy of the array of DC Huffman tables set on the most recent call to setEncodeTables, or null if tables are not currently set. |
| boolean | [**getOptimizeHuffmanTables**](http://docs.google.com/javax/imageio/plugins/jpeg/JPEGImageWriteParam.html#getOptimizeHuffmanTables())()            Returns the value passed into the most recent call to setOptimizeHuffmanTables, or false if setOptimizeHuffmanTables has never been called. |
| [JPEGQTable](http://docs.google.com/javax/imageio/plugins/jpeg/JPEGQTable.html)[] | [**getQTables**](http://docs.google.com/javax/imageio/plugins/jpeg/JPEGImageWriteParam.html#getQTables())()            Returns a copy of the array of quantization tables set on the most recent call to setEncodeTables, or null if tables are not currently set. |
| boolean | [**isCompressionLossless**](http://docs.google.com/javax/imageio/plugins/jpeg/JPEGImageWriteParam.html#isCompressionLossless())()            Returns false since the JPEG plug-in only supports lossy compression. |
| void | [**setEncodeTables**](http://docs.google.com/javax/imageio/plugins/jpeg/JPEGImageWriteParam.html#setEncodeTables(javax.imageio.plugins.jpeg.JPEGQTable%5B%5D,%20javax.imageio.plugins.jpeg.JPEGHuffmanTable%5B%5D,%20javax.imageio.plugins.jpeg.JPEGHuffmanTable%5B%5D))([JPEGQTable](http://docs.google.com/javax/imageio/plugins/jpeg/JPEGQTable.html)[] qTables, [JPEGHuffmanTable](http://docs.google.com/javax/imageio/plugins/jpeg/JPEGHuffmanTable.html)[] DCHuffmanTables, [JPEGHuffmanTable](http://docs.google.com/javax/imageio/plugins/jpeg/JPEGHuffmanTable.html)[] ACHuffmanTables)            Sets the quantization and Huffman tables to use in encoding abbreviated streams. |
| void | [**setOptimizeHuffmanTables**](http://docs.google.com/javax/imageio/plugins/jpeg/JPEGImageWriteParam.html#setOptimizeHuffmanTables(boolean))(boolean optimize)            Tells the writer to generate optimized Huffman tables for the image as part of the writing process. |
| void | [**unsetCompression**](http://docs.google.com/javax/imageio/plugins/jpeg/JPEGImageWriteParam.html#unsetCompression())()            Removes any previous compression quality setting. |
| void | [**unsetEncodeTables**](http://docs.google.com/javax/imageio/plugins/jpeg/JPEGImageWriteParam.html#unsetEncodeTables())()            Removes any quantization and Huffman tables that are currently set. |

| **Methods inherited from class javax.imageio.**[**ImageWriteParam**](http://docs.google.com/javax/imageio/ImageWriteParam.html) |
| --- |
| [canOffsetTiles](http://docs.google.com/javax/imageio/ImageWriteParam.html#canOffsetTiles()), [canWriteCompressed](http://docs.google.com/javax/imageio/ImageWriteParam.html#canWriteCompressed()), [canWriteProgressive](http://docs.google.com/javax/imageio/ImageWriteParam.html#canWriteProgressive()), [canWriteTiles](http://docs.google.com/javax/imageio/ImageWriteParam.html#canWriteTiles()), [getBitRate](http://docs.google.com/javax/imageio/ImageWriteParam.html#getBitRate(float)), [getCompressionMode](http://docs.google.com/javax/imageio/ImageWriteParam.html#getCompressionMode()), [getCompressionQuality](http://docs.google.com/javax/imageio/ImageWriteParam.html#getCompressionQuality()), [getCompressionType](http://docs.google.com/javax/imageio/ImageWriteParam.html#getCompressionType()), [getCompressionTypes](http://docs.google.com/javax/imageio/ImageWriteParam.html#getCompressionTypes()), [getLocale](http://docs.google.com/javax/imageio/ImageWriteParam.html#getLocale()), [getLocalizedCompressionTypeName](http://docs.google.com/javax/imageio/ImageWriteParam.html#getLocalizedCompressionTypeName()), [getPreferredTileSizes](http://docs.google.com/javax/imageio/ImageWriteParam.html#getPreferredTileSizes()), [getProgressiveMode](http://docs.google.com/javax/imageio/ImageWriteParam.html#getProgressiveMode()), [getTileGridXOffset](http://docs.google.com/javax/imageio/ImageWriteParam.html#getTileGridXOffset()), [getTileGridYOffset](http://docs.google.com/javax/imageio/ImageWriteParam.html#getTileGridYOffset()), [getTileHeight](http://docs.google.com/javax/imageio/ImageWriteParam.html#getTileHeight()), [getTileWidth](http://docs.google.com/javax/imageio/ImageWriteParam.html#getTileWidth()), [getTilingMode](http://docs.google.com/javax/imageio/ImageWriteParam.html#getTilingMode()), [setCompressionMode](http://docs.google.com/javax/imageio/ImageWriteParam.html#setCompressionMode(int)), [setCompressionQuality](http://docs.google.com/javax/imageio/ImageWriteParam.html#setCompressionQuality(float)), [setCompressionType](http://docs.google.com/javax/imageio/ImageWriteParam.html#setCompressionType(java.lang.String)), [setProgressiveMode](http://docs.google.com/javax/imageio/ImageWriteParam.html#setProgressiveMode(int)), [setTiling](http://docs.google.com/javax/imageio/ImageWriteParam.html#setTiling(int,%20int,%20int,%20int)), [setTilingMode](http://docs.google.com/javax/imageio/ImageWriteParam.html#setTilingMode(int)), [unsetTiling](http://docs.google.com/javax/imageio/ImageWriteParam.html#unsetTiling()) |

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

| **Methods inherited from class java.lang.**[**Object**](http://docs.google.com/java/lang/Object.html) |
| --- |
| [clone](http://docs.google.com/java/lang/Object.html#clone()), [equals](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)), [finalize](http://docs.google.com/java/lang/Object.html#finalize()), [getClass](http://docs.google.com/java/lang/Object.html#getClass()), [hashCode](http://docs.google.com/java/lang/Object.html#hashCode()), [notify](http://docs.google.com/java/lang/Object.html#notify()), [notifyAll](http://docs.google.com/java/lang/Object.html#notifyAll()), [toString](http://docs.google.com/java/lang/Object.html#toString()), [wait](http://docs.google.com/java/lang/Object.html#wait()), [wait](http://docs.google.com/java/lang/Object.html#wait(long)), [wait](http://docs.google.com/java/lang/Object.html#wait(long,%20int)) |

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

### JPEGImageWriteParam

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

Constructs a JPEGImageWriteParam. Tiling is not supported. Progressive encoding is supported. The default progressive mode is MODE\_DISABLED. A single form of compression, named "JPEG", is supported. The default compression quality is 0.75.

**Parameters:**locale - a Locale to be used by the superclass to localize compression type names and quality descriptions, or null.

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

### unsetCompression

public void **unsetCompression**()

Removes any previous compression quality setting.

The default implementation resets the compression quality to 0.75F.

**Overrides:**[unsetCompression](http://docs.google.com/javax/imageio/ImageWriteParam.html#unsetCompression()) in class [ImageWriteParam](http://docs.google.com/javax/imageio/ImageWriteParam.html) **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the compression mode is not MODE\_EXPLICIT.**See Also:**[ImageWriteParam.setCompressionType(java.lang.String)](http://docs.google.com/javax/imageio/ImageWriteParam.html#setCompressionType(java.lang.String)), [ImageWriteParam.setCompressionQuality(float)](http://docs.google.com/javax/imageio/ImageWriteParam.html#setCompressionQuality(float))

### isCompressionLossless

public boolean **isCompressionLossless**()

Returns false since the JPEG plug-in only supports lossy compression.

**Overrides:**[isCompressionLossless](http://docs.google.com/javax/imageio/ImageWriteParam.html#isCompressionLossless()) in class [ImageWriteParam](http://docs.google.com/javax/imageio/ImageWriteParam.html) **Returns:**false. **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the compression mode is not MODE\_EXPLICIT.

### getCompressionQualityDescriptions

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

**Description copied from class:** [**ImageWriteParam**](http://docs.google.com/javax/imageio/ImageWriteParam.html#getCompressionQualityDescriptions()) Returns an array of Strings that may be used along with getCompressionQualityValues as part of a user interface for setting or displaying the compression quality level. The String with index i provides a description of the range of quality levels between getCompressionQualityValues[i] and getCompressionQualityValues[i + 1]. Note that the length of the array returned from getCompressionQualityValues will always be one greater than that returned from getCompressionQualityDescriptions.

As an example, the strings "Good", "Better", and "Best" could be associated with the ranges [0, .33), [.33, .66), and [.66, 1.0]. In this case, getCompressionQualityDescriptions would return { "Good", "Better", "Best" } and getCompressionQualityValues would return { 0.0F, .33F, .66F, 1.0F }.

If no descriptions are available, null is returned. If null is returned from getCompressionQualityValues, this method must also return null.

The descriptions should be localized for the Locale returned by getLocale, if it is non-null.

If there are multiple compression types but none has been set, an IllegalStateException is thrown.

The default implementation checks that compression is supported and that the compression mode is MODE\_EXPLICIT. If so, if getCompressionTypes() is null or getCompressionType() is non-null, it returns null.

**Overrides:**[getCompressionQualityDescriptions](http://docs.google.com/javax/imageio/ImageWriteParam.html#getCompressionQualityDescriptions()) in class [ImageWriteParam](http://docs.google.com/javax/imageio/ImageWriteParam.html) **Returns:**an array of Strings containing localized descriptions of the compression quality levels.**See Also:**[ImageWriteParam.getCompressionQualityValues()](http://docs.google.com/javax/imageio/ImageWriteParam.html#getCompressionQualityValues())

### getCompressionQualityValues

public float[] **getCompressionQualityValues**()

**Description copied from class:** [**ImageWriteParam**](http://docs.google.com/javax/imageio/ImageWriteParam.html#getCompressionQualityValues()) Returns an array of floats that may be used along with getCompressionQualityDescriptions as part of a user interface for setting or displaying the compression quality level. See [getCompressionQualityDescriptions](http://docs.google.com/javax/imageio/ImageWriteParam.html#getCompressionQualityDescriptions()) for more information.

If no descriptions are available, null is returned. If null is returned from getCompressionQualityDescriptions, this method must also return null.

If there are multiple compression types but none has been set, an IllegalStateException is thrown.

The default implementation checks that compression is supported and that the compression mode is MODE\_EXPLICIT. If so, if getCompressionTypes() is null or getCompressionType() is non-null, it returns null.

**Overrides:**[getCompressionQualityValues](http://docs.google.com/javax/imageio/ImageWriteParam.html#getCompressionQualityValues()) in class [ImageWriteParam](http://docs.google.com/javax/imageio/ImageWriteParam.html) **Returns:**an array of floats indicating the boundaries between the compression quality levels as described by the Strings from getCompressionQualityDescriptions.**See Also:**[ImageWriteParam.getCompressionQualityDescriptions()](http://docs.google.com/javax/imageio/ImageWriteParam.html#getCompressionQualityDescriptions())

### areTablesSet

public boolean **areTablesSet**()

Returns true if tables are currently set.

**Returns:**true if tables are present.

### setEncodeTables

public void **setEncodeTables**([JPEGQTable](http://docs.google.com/javax/imageio/plugins/jpeg/JPEGQTable.html)[] qTables,  
 [JPEGHuffmanTable](http://docs.google.com/javax/imageio/plugins/jpeg/JPEGHuffmanTable.html)[] DCHuffmanTables,  
 [JPEGHuffmanTable](http://docs.google.com/javax/imageio/plugins/jpeg/JPEGHuffmanTable.html)[] ACHuffmanTables)

Sets the quantization and Huffman tables to use in encoding abbreviated streams. There may be a maximum of 4 tables of each type. These tables are ignored if tables are specified in the metadata. All arguments must be non-null. The two arrays of Huffman tables must have the same number of elements. The table specifiers in the frame and scan headers in the metadata are assumed to be equivalent to indices into these arrays. The argument arrays are copied by this method.

**Parameters:**qTables - An array of quantization table objects.DCHuffmanTables - An array of Huffman table objects.ACHuffmanTables - An array of Huffman table objects. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if any of the arguments is null or has more than 4 elements, or if the numbers of DC and AC tables differ.**See Also:**[unsetEncodeTables()](http://docs.google.com/javax/imageio/plugins/jpeg/JPEGImageWriteParam.html#unsetEncodeTables())

### unsetEncodeTables

public void **unsetEncodeTables**()

Removes any quantization and Huffman tables that are currently set.

**See Also:**[setEncodeTables(javax.imageio.plugins.jpeg.JPEGQTable[], javax.imageio.plugins.jpeg.JPEGHuffmanTable[], javax.imageio.plugins.jpeg.JPEGHuffmanTable[])](http://docs.google.com/javax/imageio/plugins/jpeg/JPEGImageWriteParam.html#setEncodeTables(javax.imageio.plugins.jpeg.JPEGQTable%5B%5D,%20javax.imageio.plugins.jpeg.JPEGHuffmanTable%5B%5D,%20javax.imageio.plugins.jpeg.JPEGHuffmanTable%5B%5D))

### getQTables

public [JPEGQTable](http://docs.google.com/javax/imageio/plugins/jpeg/JPEGQTable.html)[] **getQTables**()

Returns a copy of the array of quantization tables set on the most recent call to setEncodeTables, or null if tables are not currently set.

**Returns:**an array of JPEGQTable objects, or null.**See Also:**[setEncodeTables(javax.imageio.plugins.jpeg.JPEGQTable[], javax.imageio.plugins.jpeg.JPEGHuffmanTable[], javax.imageio.plugins.jpeg.JPEGHuffmanTable[])](http://docs.google.com/javax/imageio/plugins/jpeg/JPEGImageWriteParam.html#setEncodeTables(javax.imageio.plugins.jpeg.JPEGQTable%5B%5D,%20javax.imageio.plugins.jpeg.JPEGHuffmanTable%5B%5D,%20javax.imageio.plugins.jpeg.JPEGHuffmanTable%5B%5D))

### getDCHuffmanTables

public [JPEGHuffmanTable](http://docs.google.com/javax/imageio/plugins/jpeg/JPEGHuffmanTable.html)[] **getDCHuffmanTables**()

Returns a copy of the array of DC Huffman tables set on the most recent call to setEncodeTables, or null if tables are not currently set.

**Returns:**an array of JPEGHuffmanTable objects, or null.**See Also:**[setEncodeTables(javax.imageio.plugins.jpeg.JPEGQTable[], javax.imageio.plugins.jpeg.JPEGHuffmanTable[], javax.imageio.plugins.jpeg.JPEGHuffmanTable[])](http://docs.google.com/javax/imageio/plugins/jpeg/JPEGImageWriteParam.html#setEncodeTables(javax.imageio.plugins.jpeg.JPEGQTable%5B%5D,%20javax.imageio.plugins.jpeg.JPEGHuffmanTable%5B%5D,%20javax.imageio.plugins.jpeg.JPEGHuffmanTable%5B%5D))

### getACHuffmanTables

public [JPEGHuffmanTable](http://docs.google.com/javax/imageio/plugins/jpeg/JPEGHuffmanTable.html)[] **getACHuffmanTables**()

Returns a copy of the array of AC Huffman tables set on the most recent call to setEncodeTables, or null if tables are not currently set.

**Returns:**an array of JPEGHuffmanTable objects, or null.**See Also:**[setEncodeTables(javax.imageio.plugins.jpeg.JPEGQTable[], javax.imageio.plugins.jpeg.JPEGHuffmanTable[], javax.imageio.plugins.jpeg.JPEGHuffmanTable[])](http://docs.google.com/javax/imageio/plugins/jpeg/JPEGImageWriteParam.html#setEncodeTables(javax.imageio.plugins.jpeg.JPEGQTable%5B%5D,%20javax.imageio.plugins.jpeg.JPEGHuffmanTable%5B%5D,%20javax.imageio.plugins.jpeg.JPEGHuffmanTable%5B%5D))

### setOptimizeHuffmanTables

public void **setOptimizeHuffmanTables**(boolean optimize)

Tells the writer to generate optimized Huffman tables for the image as part of the writing process. The default is false. If this flag is set to true, it overrides any tables specified in the metadata. Note that this means that any image written with this flag set to true will always contain Huffman tables.

**Parameters:**optimize - A boolean indicating whether to generate optimized Huffman tables when writing.**See Also:**[getOptimizeHuffmanTables()](http://docs.google.com/javax/imageio/plugins/jpeg/JPEGImageWriteParam.html#getOptimizeHuffmanTables())

### getOptimizeHuffmanTables

public boolean **getOptimizeHuffmanTables**()

Returns the value passed into the most recent call to setOptimizeHuffmanTables, or false if setOptimizeHuffmanTables has never been called.

**Returns:**true if the writer will generate optimized Huffman tables.**See Also:**[setOptimizeHuffmanTables(boolean)](http://docs.google.com/javax/imageio/plugins/jpeg/JPEGImageWriteParam.html#setOptimizeHuffmanTables(boolean))

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/JPEGImageWriteParam.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/plugins/jpeg/JPEGImageReadParam.html)   [**NEXT CLASS**](http://docs.google.com/javax/imageio/plugins/jpeg/JPEGQTable.html) | [**FRAMES**](http://docs.google.com/index.html?javax/imageio/plugins/jpeg/JPEGImageWriteParam.html)    [**NO FRAMES**](http://docs.google.com/JPEGImageWriteParam.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#2et92p0) | [CONSTR](#3dy6vkm) | [METHOD](#1t3h5sf) | DETAIL: FIELD | [CONSTR](#3rdcrjn) | [METHOD](#lnxbz9) |

[Submit a bug or feature](http://bugs.sun.com/services/bugreport/index.jsp)

For further API reference and developer documentation, see [Java SE Developer Documentation](http://docs.google.com/webnotes/devdocs-vs-specs.html). That documentation contains more detailed, developer-targeted descriptions, with conceptual overviews, definitions of terms, workarounds, and working code examples.

Copyright 2006 Sun Microsystems, Inc. All rights reserved. Use is subject to [license terms](http://docs.google.com/legal/license.html). Also see the [documentation redistribution policy](http://java.sun.com/docs/redist.html).