| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/DocFlavor.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/print/Doc.html)   [**NEXT CLASS**](http://docs.google.com/javax/print/DocFlavor.BYTE_ARRAY.html) | [**FRAMES**](http://docs.google.com/index.html?javax/print/DocFlavor.html)    [**NO FRAMES**](http://docs.google.com/DocFlavor.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#3znysh7) | [FIELD](#2et92p0) | [CONSTR](#tyjcwt) | [METHOD](#3dy6vkm) | DETAIL: [FIELD](#4d34og8) | [CONSTR](#17dp8vu) | [METHOD](#26in1rg) |

## **javax.print**

Class DocFlavor

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

**All Implemented Interfaces:** [Serializable](http://docs.google.com/java/io/Serializable.html), [Cloneable](http://docs.google.com/java/lang/Cloneable.html) **Direct Known Subclasses:** [DocFlavor.BYTE\_ARRAY](http://docs.google.com/javax/print/DocFlavor.BYTE_ARRAY.html), [DocFlavor.CHAR\_ARRAY](http://docs.google.com/javax/print/DocFlavor.CHAR_ARRAY.html), [DocFlavor.INPUT\_STREAM](http://docs.google.com/javax/print/DocFlavor.INPUT_STREAM.html), [DocFlavor.READER](http://docs.google.com/javax/print/DocFlavor.READER.html), [DocFlavor.SERVICE\_FORMATTED](http://docs.google.com/javax/print/DocFlavor.SERVICE_FORMATTED.html), [DocFlavor.STRING](http://docs.google.com/javax/print/DocFlavor.STRING.html), [DocFlavor.URL](http://docs.google.com/javax/print/DocFlavor.URL.html)

public class **DocFlavor**extends [Object](http://docs.google.com/java/lang/Object.html)implements [Serializable](http://docs.google.com/java/io/Serializable.html), [Cloneable](http://docs.google.com/java/lang/Cloneable.html)

Class DocFlavor encapsulates an object that specifies the format in which print data is supplied to a [DocPrintJob](http://docs.google.com/javax/print/DocPrintJob.html). "Doc" is a short, easy-to-pronounce term that means "a piece of print data." The print data format, or "doc flavor", consists of two things:

* **MIME type.** This is a Multipurpose Internet Mail Extensions (MIME) media type (as defined in [RFC 2045](http://www.ietf.org/rfc/rfc2045.txt) and [RFC 2046](http://www.ietf.org/rfc/rfc2046.txt)) that specifies how the print data is to be interpreted. The charset of text data should be the IANA MIME-preferred name, or its canonical name if no preferred name is specified. Additionally a few historical names supported by earlier versions of the Java platform may be recognized. See  [character encodings](http://docs.google.com/java/lang/package-summary.html#charenc) for more information on the character encodings supported on the Java platform.
* **Representation class name.** This specifies the fully-qualified name of the class of the object from which the actual print data comes, as returned by the [Class.getName()](http://docs.google.com/java/lang/Class.html#getName()) method. (Thus the class name for byte[] is "[B", for char[] it is "[C".)

A DocPrintJob obtains its print data by means of interface [Doc](http://docs.google.com/javax/print/Doc.html). A Doc object lets the DocPrintJob determine the doc flavor the client can supply. A Doc object also lets the DocPrintJob obtain an instance of the doc flavor's representation class, from which the DocPrintJob then obtains the actual print data.

### Client Formatted Print Data

There are two broad categories of print data, client formatted print data and service formatted print data.

For **client formatted print data**, the client determines or knows the print data format. For example the client may have a JPEG encoded image, a URL for HTML code, or a disk file containing plain text in some encoding, possibly obtained from an external source, and requires a way to describe the data format to the print service.

The doc flavor's representation class is a conduit for the JPS DocPrintJob to obtain a sequence of characters or bytes from the client. The doc flavor's MIME type is one of the standard media types telling how to interpret the sequence of characters or bytes. For a list of standard media types, see the Internet Assigned Numbers Authority's (IANA's) [Media Types Directory](http://www.isi.edu/in-notes/iana/assignments/media-types/). Interface [Doc](http://docs.google.com/javax/print/Doc.html) provides two utility operations, [getReaderForText](http://docs.google.com/javax/print/Doc.html#getReaderForText()) and [getStreamForBytes()](http://docs.google.com/javax/print/Doc.html#getStreamForBytes()), to help a Doc object's client extract client formatted print data.

For client formatted print data, the print data representation class is typically one of the following (although other representation classes are permitted):

* Character array (char[]) -- The print data consists of the Unicde characters in the array.
* String -- The print data consists of the Unicode characters in the string.
* Character stream ([java.io.Reader](http://docs.google.com/java/io/Reader.html)) -- The print data consists of the Unicode characters read from the stream up to the end-of-stream.
* Byte array (byte[]) -- The print data consists of the bytes in the array. The bytes are encoded in the character set specified by the doc flavor's MIME type. If the MIME type does not specify a character set, the default character set is US-ASCII.
* Byte stream ([java.io.InputStream](http://docs.google.com/java/io/InputStream.html)) -- The print data consists of the bytes read from the stream up to the end-of-stream. The bytes are encoded in the character set specified by the doc flavor's MIME type. If the MIME type does not specify a character set, the default character set is US-ASCII.
* Uniform Resource Locator ([URL](http://docs.google.com/java/net/URL.html)) -- The print data consists of the bytes read from the URL location. The bytes are encoded in the character set specified by the doc flavor's MIME type. If the MIME type does not specify a character set, the default character set is US-ASCII.  
  When the representation class is a URL, the print service itself accesses and downloads the document directly from its URL address, without involving the client. The service may be some form of network print service which is executing in a different environment. This means you should not use a URL print data flavor to print a document at a restricted URL that the client can see but the printer cannot see. This also means you should not use a URL print data flavor to print a document stored in a local file that is not available at a URL accessible independently of the client. For example, a file that is not served up by an HTTP server or FTP server. To print such documents, let the client open an input stream on the URL or file and use an input stream data flavor.

### Default and Platform Encodings

For byte print data where the doc flavor's MIME type does not include a charset parameter, the Java Print Service instance assumes the US-ASCII character set by default. This is in accordance with [RFC 2046](http://www.ietf.org/rfc/rfc2046.txt), which says the default character set is US-ASCII. Note that US-ASCII is a subset of UTF-8, so in the future this may be widened if a future RFC endorses UTF-8 as the default in a compatible manner.

Also note that this is different than the behaviour of the Java runtime when interpreting a stream of bytes as text data. That assumes the default encoding for the user's locale. Thus, when spooling a file in local encoding to a Java Print Service it is important to correctly specify the encoding. Developers working in the English locales should be particularly conscious of this, as their platform encoding corresponds to the default mime charset. By this coincidence that particular case may work without specifying the encoding of platform data.

Every instance of the Java virtual machine has a default character encoding determined during virtual-machine startup and typically depends upon the locale and charset being used by the underlying operating system. In a distributed environment there is no gurantee that two VM's share the same default encoding. Thus clients which want to stream platform encoded text data from the host platform to a Java Print Service instance must explicitly declare the charset and not rely on defaults.

The preferred form is the official IANA primary name for an encoding. Applications which stream text data should always specify the charset in the mime type, which necessitates obtaining the encoding of the host platform for data (eg files) stored in that platform's encoding. A CharSet which corresponds to this and is suitable for use in a mime-type for a DocFlavor can be obtained from [DocFlavor.hostEncoding](http://docs.google.com/javax/print/DocFlavor.html#hostEncoding) This may not always be the primary IANA name but is guaranteed to be understood by this VM. For common flavors, the pre-defined \*HOST DocFlavors may be used.

See  [character encodings](http://docs.google.com/java/lang/package-summary.html#charenc) for more information on the character encodings supported on the Java platform.

### Recommended DocFlavors

The Java Print Service API does not define any mandatorily supported DocFlavors. However, here are some examples of MIME types that a Java Print Service instance might support for client formatted print data. Nested classes inside class DocFlavor declare predefined static constant DocFlavor objects for these example doc flavors; class DocFlavor's constructor can be used to create an arbitrary doc flavor.

Preformatted text

| MIME-Type | Description |
| --- | --- |
| "text/plain" | Plain text in the default character set (US-ASCII) |
| "text/plain; charset=*xxx*" | Plain text in character set *xxx* |
| "text/html" | HyperText Markup Language in the default character set (US-ASCII) |
| "text/html; charset=*xxx*" | HyperText Markup Language in character set *xxx* |

* In general, preformatted text print data is provided either in a character oriented representation class (character array, String, Reader) or in a byte oriented representation class (byte array, InputStream, URL).

Preformatted page description language (PDL) documents

| MIME-Type | Description |
| --- | --- |
| "application/pdf" | Portable Document Format document |
| "application/postscript" | PostScript document |
| "application/vnd.hp-PCL" | Printer Control Language document |

* In general, preformatted PDL print data is provided in a byte oriented representation class (byte array, InputStream, URL).

Preformatted images

| MIME-Type | Description |
| --- | --- |
| "image/gif" | Graphics Interchange Format image |
| "image/jpeg" | Joint Photographic Experts Group image |
| "image/png" | Portable Network Graphics image |

* In general, preformatted image print data is provided in a byte oriented representation class (byte array, InputStream, URL).

Preformatted autosense print data

| MIME-Type | Description |
| --- | --- |
| "application/octet-stream" | The print data format is unspecified (just an octet stream) |

### The printer decides how to interpret the print data; the way this "autosensing" works is implementation dependent. In general, preformatted autosense print data is provided in a byte oriented representation class (byte array, InputStream, URL). Service Formatted Print Data For **service formatted print data**, the Java Print Service instance determines the print data format. The doc flavor's representation class denotes an interface whose methods the DocPrintJob invokes to determine the content to be printed -- such as a renderable image interface or a Java printable interface. The doc flavor's MIME type is the special value "application/x-java-jvm-local-objectref" indicating the client will supply a reference to a Java object that implements the interface named as the representation class. This MIME type is just a placeholder; what's important is the print data representation class. For service formatted print data, the print data representation class is typically one of the following (although other representation classes are permitted). Nested classes inside class DocFlavor declare predefined static constant DocFlavor objects for these example doc flavors; class DocFlavor's constructor can be used to create an arbitrary doc flavor.

* + Renderable image object -- The client supplies an object that implements interface [RenderableImage](http://docs.google.com/java/awt/image/renderable/RenderableImage.html). The printer calls methods in that interface to obtain the image to be printed.
  + Printable object -- The client supplies an object that implements interface [Printable](http://docs.google.com/java/awt/print/Printable.html). The printer calls methods in that interface to obtain the pages to be printed, one by one. For each page, the printer supplies a graphics context, and whatever the client draws in that graphics context gets printed.
  + Pageable object -- The client supplies an object that implements interface [Pageable](http://docs.google.com/java/awt/print/Pageable.html). The printer calls methods in that interface to obtain the pages to be printed, one by one. For each page, the printer supplies a graphics context, and whatever the client draws in that graphics context gets printed.

Pre-defined Doc FlavorsA Java Print Service instance is not ***required*** to support the following print data formats and print data representation classes. In fact, a developer using this class should **never** assume that a particular print service supports the document types corresponding to these pre-defined doc flavors. Always query the print service to determine what doc flavors it supports. However, developers who have print services that support these doc flavors are encouraged to refer to the predefined singleton instances created here.

* + Plain text print data provided through a byte stream. Specifically, the following doc flavors are recommended to be supported:  
    ·   ("text/plain", "java.io.InputStream")  
    ·   ("text/plain; charset=us-ascii", "java.io.InputStream")  
    ·   ("text/plain; charset=utf-8", "java.io.InputStream")
  + Renderable image objects. Specifically, the following doc flavor is recommended to be supported:  
    ·   ("application/x-java-jvm-local-objectref", "java.awt.image.renderable.RenderableImage")

A Java Print Service instance is allowed to support any other doc flavors (or none) in addition to the above mandatory ones, at the implementation's choice.Support for the above doc flavors is desirable so a printing client can rely on being able to print on any JPS printer, regardless of which doc flavors the printer supports. If the printer doesn't support the client's preferred doc flavor, the client can at least print plain text, or the client can convert its data to a renderable image and print the image.Furthermore, every Java Print Service instance must fulfill these requirements for processing plain text print data:

* + The character pair carriage return-line feed (CR-LF) means "go to column 1 of the next line."
  + A carriage return (CR) character standing by itself means "go to column 1 of the next line."
  + A line feed (LF) character standing by itself means "go to column 1 of the next line."

The client must itself perform all plain text print data formatting not addressed by the above requirements.Design RationaleClass DocFlavor in package javax.print.data is similar to class [DataFlavor](http://docs.google.com/java/awt/datatransfer/DataFlavor.html). Class DataFlavor is not used in the Java Print Service (JPS) API for three reasons which are all rooted in allowing the JPS API to be shared by other print services APIs which may need to run on Java profiles which do not include all of the Java Platform, Standard Edition.

### The JPS API is designed to be used in Java profiles which do not support AWT.

* 1. The implementation of class java.awt.datatransfer.DataFlavor does not guarantee that equivalent data flavors will have the same serialized representation. DocFlavor does, and can be used in services which need this.
  2. The implementation of class java.awt.datatransfer.DataFlavor includes a human presentable name as part of the serialized representation. This is not appropriate as part of a service matching constraint.

Class DocFlavor's serialized representation uses the following canonical form of a MIME type string. Thus, two doc flavors with MIME types that are not identical but that are equivalent (that have the same canonical form) may be considered equal.

* + The media type, media subtype, and parameters are retained, but all comments and whitespace characters are discarded.
  + The media type, media subtype, and parameter names are converted to lowercase.
  + The parameter values retain their original case, except a charset parameter value for a text media type is converted to lowercase.
  + Quote characters surrounding parameter values are removed.
  + Quoting backslash characters inside parameter values are removed.
  + The parameters are arranged in ascending order of parameter name.

Class DocFlavor's serialized representation also contains the fully-qualified class *name* of the representation class (a String object), rather than the representation class itself (a Class object). This allows a client to examine the doc flavors a Java Print Service instance supports without having to load the representation classes, which may be problematic for limited-resource clients.**See Also:**[Serialized Form](http://docs.google.com/serialized-form.html#javax.print.DocFlavor)

| **Nested Class Summary** | |
| --- | --- |
| static class | [**DocFlavor.BYTE\_ARRAY**](http://docs.google.com/javax/print/DocFlavor.BYTE_ARRAY.html)           Class DocFlavor.BYTE\_ARRAY provides predefined static constant DocFlavor objects for example doc flavors using a byte array (byte[]) as the print data representation class. |
| static class | [**DocFlavor.CHAR\_ARRAY**](http://docs.google.com/javax/print/DocFlavor.CHAR_ARRAY.html)           Class DocFlavor.CHAR\_ARRAY provides predefined static constant DocFlavor objects for example doc flavors using a character array (char[]) as the print data representation class. |
| static class | [**DocFlavor.INPUT\_STREAM**](http://docs.google.com/javax/print/DocFlavor.INPUT_STREAM.html)           Class DocFlavor.INPUT\_STREAM provides predefined static constant DocFlavor objects for example doc flavors using a byte stream ([java.io.InputStream](http://docs.google.com/java/io/InputStream.html)) as the print data representation class. |
| static class | [**DocFlavor.READER**](http://docs.google.com/javax/print/DocFlavor.READER.html)           Class DocFlavor.READER provides predefined static constant DocFlavor objects for example doc flavors using a character stream ([java.io.Reader](http://docs.google.com/java/io/Reader.html)) as the print data representation class. |
| static class | [**DocFlavor.SERVICE\_FORMATTED**](http://docs.google.com/javax/print/DocFlavor.SERVICE_FORMATTED.html)           Class DocFlavor.SERVICE\_FORMATTED provides predefined static constant DocFlavor objects for example doc flavors for service formatted print data. |
| static class | [**DocFlavor.STRING**](http://docs.google.com/javax/print/DocFlavor.STRING.html)           Class DocFlavor.STRING provides predefined static constant DocFlavor objects for example doc flavors using a string ([java.lang.String](http://docs.google.com/java/lang/String.html)) as the print data representation class. |
| static class | [**DocFlavor.URL**](http://docs.google.com/javax/print/DocFlavor.URL.html)           Class DocFlavor.URL provides predefined static constant DocFlavor objects. |

| **Field Summary** | |
| --- | --- |
| static [String](http://docs.google.com/java/lang/String.html) | [**hostEncoding**](http://docs.google.com/javax/print/DocFlavor.html#hostEncoding)           A String representing the host operating system encoding. |

| **Constructor Summary** | |
| --- | --- |
| [**DocFlavor**](http://docs.google.com/javax/print/DocFlavor.html#DocFlavor(java.lang.String,%20java.lang.String))([String](http://docs.google.com/java/lang/String.html) mimeType, [String](http://docs.google.com/java/lang/String.html) className)           Constructs a new doc flavor object from the given MIME type and representation class name. |

| **Method Summary** | |
| --- | --- |
| boolean | [**equals**](http://docs.google.com/javax/print/DocFlavor.html#equals(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) obj)           Determines if this doc flavor object is equal to the given object. |
| [String](http://docs.google.com/java/lang/String.html) | [**getMediaSubtype**](http://docs.google.com/javax/print/DocFlavor.html#getMediaSubtype())()           Returns this doc flavor object's media subtype (from the MIME type). |
| [String](http://docs.google.com/java/lang/String.html) | [**getMediaType**](http://docs.google.com/javax/print/DocFlavor.html#getMediaType())()           Returns this doc flavor object's media type (from the MIME type). |
| [String](http://docs.google.com/java/lang/String.html) | [**getMimeType**](http://docs.google.com/javax/print/DocFlavor.html#getMimeType())()           Returns this doc flavor object's MIME type string based on the canonical form. |
| [String](http://docs.google.com/java/lang/String.html) | [**getParameter**](http://docs.google.com/javax/print/DocFlavor.html#getParameter(java.lang.String))([String](http://docs.google.com/java/lang/String.html) paramName)           Returns a String representing a MIME parameter. |
| [String](http://docs.google.com/java/lang/String.html) | [**getRepresentationClassName**](http://docs.google.com/javax/print/DocFlavor.html#getRepresentationClassName())()           Returns the name of this doc flavor object's representation class. |
| int | [**hashCode**](http://docs.google.com/javax/print/DocFlavor.html#hashCode())()           Returns a hash code for this doc flavor object. |
| [String](http://docs.google.com/java/lang/String.html) | [**toString**](http://docs.google.com/javax/print/DocFlavor.html#toString())()           Converts this DocFlavor to a string. |

| **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()), [finalize](http://docs.google.com/java/lang/Object.html#finalize()), [getClass](http://docs.google.com/java/lang/Object.html#getClass()), [notify](http://docs.google.com/java/lang/Object.html#notify()), [notifyAll](http://docs.google.com/java/lang/Object.html#notifyAll()), [wait](http://docs.google.com/java/lang/Object.html#wait()), [wait](http://docs.google.com/java/lang/Object.html#wait(long)), [wait](http://docs.google.com/java/lang/Object.html#wait(long,%20int)) |

| **Field Detail** |
| --- |

hostEncodingpublic static final [String](http://docs.google.com/java/lang/String.html) **hostEncoding**A String representing the host operating system encoding. This will follow the conventions documented in  [*RFC 2278: IANA Charset Registration Procedures*](http://ietf.org/rfc/rfc2278.txt) except where historical names are returned for compatibility with previous versions of the Java platform. The value returned from method is valid only for the VM which returns it, for use in a DocFlavor. This is the charset for all the "HOST" pre-defined DocFlavors in the executing VM.

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

DocFlavorpublic **DocFlavor**([String](http://docs.google.com/java/lang/String.html) mimeType,  
 [String](http://docs.google.com/java/lang/String.html) className)Constructs a new doc flavor object from the given MIME type and representation class name. The given MIME type is converted into canonical form and stored internally.**Parameters:**mimeType - MIME media type string.className - Fully-qualified representation class name. **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - (unchecked exception) Thrown if mimeType is null or className is null. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - (unchecked exception) Thrown if mimeType does not obey the syntax for a MIME media type string.

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

getMimeTypepublic [String](http://docs.google.com/java/lang/String.html) **getMimeType**()Returns this doc flavor object's MIME type string based on the canonical form. Each parameter value is enclosed in quotes.**Returns:**the mime typegetMediaTypepublic [String](http://docs.google.com/java/lang/String.html) **getMediaType**()Returns this doc flavor object's media type (from the MIME type).**Returns:**the media typegetMediaSubtypepublic [String](http://docs.google.com/java/lang/String.html) **getMediaSubtype**()Returns this doc flavor object's media subtype (from the MIME type).**Returns:**the media sub-typegetParameterpublic [String](http://docs.google.com/java/lang/String.html) **getParameter**([String](http://docs.google.com/java/lang/String.html) paramName)Returns a String representing a MIME parameter. Mime types may include parameters which are usually optional. The charset for text types is a commonly useful example. This convenience method will return the value of the specified parameter if one was specified in the mime type for this flavor.**Parameters:**paramName - the name of the paramater. This name is internally converted to the canonical lower case format before performing the match. **Returns:**String representing a mime parameter, or null if that parameter is not in the mime type string. **Throws:** throws - NullPointerException if paramName is null.getRepresentationClassNamepublic [String](http://docs.google.com/java/lang/String.html) **getRepresentationClassName**()Returns the name of this doc flavor object's representation class.**Returns:**the name of the representation class.toStringpublic [String](http://docs.google.com/java/lang/String.html) **toString**()Converts this DocFlavor to a string.**Overrides:**[toString](http://docs.google.com/java/lang/Object.html#toString()) in class [Object](http://docs.google.com/java/lang/Object.html) **Returns:**MIME type string based on the canonical form. Each parameter value is enclosed in quotes. A "class=" parameter is appended to the MIME type string to indicate the representation class name.hashCodepublic int **hashCode**()Returns a hash code for this doc flavor object.**Overrides:**[hashCode](http://docs.google.com/java/lang/Object.html#hashCode()) in class [Object](http://docs.google.com/java/lang/Object.html) **Returns:**a hash code value for this object.**See Also:**[Object.equals(java.lang.Object)](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)), [Hashtable](http://docs.google.com/java/util/Hashtable.html)equalspublic boolean **equals**([Object](http://docs.google.com/java/lang/Object.html) obj)Determines if this doc flavor object is equal to the given object. The two are equal if the given object is not null, is an instance of DocFlavor, has a MIME type equivalent to this doc flavor object's MIME type (that is, the MIME types have the same media type, media subtype, and parameters), and has the same representation class name as this doc flavor object. Thus, if two doc flavor objects' MIME types are the same except for comments, they are considered equal. However, two doc flavor objects with MIME types of "text/plain" and "text/plain; charset=US-ASCII" are not considered equal, even though they represent the same media type (because the default character set for plain text is US-ASCII).**Overrides:**[equals](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)) in class [Object](http://docs.google.com/java/lang/Object.html) **Parameters:**obj - Object to test. **Returns:**True if this doc flavor object equals obj, false otherwise.**See Also:**[Object.hashCode()](http://docs.google.com/java/lang/Object.html#hashCode()), [Hashtable](http://docs.google.com/java/util/Hashtable.html)

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/DocFlavor.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/print/Doc.html)   [**NEXT CLASS**](http://docs.google.com/javax/print/DocFlavor.BYTE_ARRAY.html) | [**FRAMES**](http://docs.google.com/index.html?javax/print/DocFlavor.html)    [**NO FRAMES**](http://docs.google.com/DocFlavor.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#3znysh7) | [FIELD](#2et92p0) | [CONSTR](#tyjcwt) | [METHOD](#3dy6vkm) | DETAIL: [FIELD](#4d34og8) | [CONSTR](#17dp8vu) | [METHOD](#26in1rg) |

[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).