| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/JavaFileManager.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/tools/JavaCompiler.CompilationTask.html)   [**NEXT CLASS**](http://docs.google.com/javax/tools/JavaFileManager.Location.html) | [**FRAMES**](http://docs.google.com/index.html?javax/tools/JavaFileManager.html)    [**NO FRAMES**](http://docs.google.com/JavaFileManager.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#2et92p0) | FIELD | CONSTR | [METHOD](#tyjcwt) | DETAIL: FIELD | CONSTR | [METHOD](#1t3h5sf) |

## **javax.tools**

Interface JavaFileManager

**All Superinterfaces:** [Closeable](http://docs.google.com/java/io/Closeable.html), [Flushable](http://docs.google.com/java/io/Flushable.html), [OptionChecker](http://docs.google.com/javax/tools/OptionChecker.html) **All Known Subinterfaces:** [StandardJavaFileManager](http://docs.google.com/javax/tools/StandardJavaFileManager.html) **All Known Implementing Classes:** [ForwardingJavaFileManager](http://docs.google.com/javax/tools/ForwardingJavaFileManager.html)

public interface **JavaFileManager**extends [Closeable](http://docs.google.com/java/io/Closeable.html), [Flushable](http://docs.google.com/java/io/Flushable.html), [OptionChecker](http://docs.google.com/javax/tools/OptionChecker.html)

File manager for tools operating on Java™ programming language source and class files. In this context, *file* means an abstraction of regular files and other sources of data.

When constructing new JavaFileObjects, the file manager must determine where to create them. For example, if a file manager manages regular files on a file system, it would most likely have a current/working directory to use as default location when creating or finding files. A number of hints can be provided to a file manager as to where to create files. Any file manager might choose to ignore these hints.

Some methods in this interface use class names. Such class names must be given in the Java Virtual Machine internal form of fully qualified class and interface names. For convenience '.' and '/' are interchangeable. The internal form is defined in chapter four of the [Java Virtual Machine Specification](http://java.sun.com/docs/books/vmspec/2nd-edition/jvms-maintenance.html).

*Discussion:* this means that the names "java/lang.package-info", "java/lang/package-info", "java.lang.package-info", are valid and equivalent. Compare to binary name as defined in the [Java Language Specification (JLS)](http://java.sun.com/docs/books/jls/) section 13.1 "The Form of a Binary".

The case of names is significant. All names should be treated as case-sensitive. For example, some file systems have case-insensitive, case-aware file names. File objects representing such files should take care to preserve case by using [File.getCanonicalFile()](http://docs.google.com/java/io/File.html#getCanonicalFile()) or similar means. If the system is not case-aware, file objects must use other means to preserve case.

*Relative names:* some methods in this interface use relative names. A relative name is a non-null, non-empty sequence of path segments separated by '/'. '.' or '..' are invalid path segments. A valid relative name must match the "path-rootless" rule of [RFC 3986](http://www.ietf.org/rfc/rfc3986.txt), section 3.3. Informally, this should be true:

URI.[create](http://docs.google.com/java/net/URI.html#create(java.lang.String))(relativeName).[normalize](http://docs.google.com/java/net/URI.html#normalize())().[getPath](http://docs.google.com/java/net/URI.html#getPath())().equals(relativeName)

All methods in this interface might throw a SecurityException.

An object of this interface is not required to support multi-threaded access, that is, be synchronized. However, it must support concurrent access to different file objects created by this object.

*Implementation note:* a consequence of this requirement is that a trivial implementation of output to a [JarOutputStream](http://docs.google.com/java/util/jar/JarOutputStream.html) is not a sufficient implementation. That is, rather than creating a JavaFileObject that returns the JarOutputStream directly, the contents must be cached until closed and then written to the JarOutputStream.

Unless explicitly allowed, all methods in this interface might throw a NullPointerException if given a null argument.

**Since:** 1.6 **See Also:**[JavaFileObject](http://docs.google.com/javax/tools/JavaFileObject.html), [FileObject](http://docs.google.com/javax/tools/FileObject.html)

| **Nested Class Summary** | |
| --- | --- |
| static interface | [**JavaFileManager.Location**](http://docs.google.com/javax/tools/JavaFileManager.Location.html)            Interface for locations of file objects. |

| **Method Summary** | |
| --- | --- |
| void | [**close**](http://docs.google.com/javax/tools/JavaFileManager.html#close())()            Releases any resources opened by this file manager directly or indirectly. |
| void | [**flush**](http://docs.google.com/javax/tools/JavaFileManager.html#flush())()            Flushes any resources opened for output by this file manager directly or indirectly. |
| [ClassLoader](http://docs.google.com/java/lang/ClassLoader.html) | [**getClassLoader**](http://docs.google.com/javax/tools/JavaFileManager.html#getClassLoader(javax.tools.JavaFileManager.Location))([JavaFileManager.Location](http://docs.google.com/javax/tools/JavaFileManager.Location.html) location)            Gets a class loader for loading plug-ins from the given location. |
| [FileObject](http://docs.google.com/javax/tools/FileObject.html) | [**getFileForInput**](http://docs.google.com/javax/tools/JavaFileManager.html#getFileForInput(javax.tools.JavaFileManager.Location,%20java.lang.String,%20java.lang.String))([JavaFileManager.Location](http://docs.google.com/javax/tools/JavaFileManager.Location.html) location, [String](http://docs.google.com/java/lang/String.html) packageName, [String](http://docs.google.com/java/lang/String.html) relativeName)            Gets a [file object](http://docs.google.com/javax/tools/FileObject.html) for input representing the specified [relative name](http://docs.google.com/JavaFileManager.html#relative_name) in the specified package in the given location. |
| [FileObject](http://docs.google.com/javax/tools/FileObject.html) | [**getFileForOutput**](http://docs.google.com/javax/tools/JavaFileManager.html#getFileForOutput(javax.tools.JavaFileManager.Location,%20java.lang.String,%20java.lang.String,%20javax.tools.FileObject))([JavaFileManager.Location](http://docs.google.com/javax/tools/JavaFileManager.Location.html) location, [String](http://docs.google.com/java/lang/String.html) packageName, [String](http://docs.google.com/java/lang/String.html) relativeName, [FileObject](http://docs.google.com/javax/tools/FileObject.html) sibling)            Gets a [file object](http://docs.google.com/javax/tools/FileObject.html) for output representing the specified [relative name](http://docs.google.com/JavaFileManager.html#relative_name) in the specified package in the given location. |
| [JavaFileObject](http://docs.google.com/javax/tools/JavaFileObject.html) | [**getJavaFileForInput**](http://docs.google.com/javax/tools/JavaFileManager.html#getJavaFileForInput(javax.tools.JavaFileManager.Location,%20java.lang.String,%20javax.tools.JavaFileObject.Kind))([JavaFileManager.Location](http://docs.google.com/javax/tools/JavaFileManager.Location.html) location, [String](http://docs.google.com/java/lang/String.html) className, [JavaFileObject.Kind](http://docs.google.com/javax/tools/JavaFileObject.Kind.html) kind)            Gets a [file object](http://docs.google.com/javax/tools/JavaFileObject.html) for input representing the specified class of the specified kind in the given location. |
| [JavaFileObject](http://docs.google.com/javax/tools/JavaFileObject.html) | [**getJavaFileForOutput**](http://docs.google.com/javax/tools/JavaFileManager.html#getJavaFileForOutput(javax.tools.JavaFileManager.Location,%20java.lang.String,%20javax.tools.JavaFileObject.Kind,%20javax.tools.FileObject))([JavaFileManager.Location](http://docs.google.com/javax/tools/JavaFileManager.Location.html) location, [String](http://docs.google.com/java/lang/String.html) className, [JavaFileObject.Kind](http://docs.google.com/javax/tools/JavaFileObject.Kind.html) kind, [FileObject](http://docs.google.com/javax/tools/FileObject.html) sibling)            Gets a [file object](http://docs.google.com/javax/tools/JavaFileObject.html) for output representing the specified class of the specified kind in the given location. |
| boolean | [**handleOption**](http://docs.google.com/javax/tools/JavaFileManager.html#handleOption(java.lang.String,%20java.util.Iterator))([String](http://docs.google.com/java/lang/String.html) current, [Iterator](http://docs.google.com/java/util/Iterator.html)<[String](http://docs.google.com/java/lang/String.html)> remaining)            Handles one option. |
| boolean | [**hasLocation**](http://docs.google.com/javax/tools/JavaFileManager.html#hasLocation(javax.tools.JavaFileManager.Location))([JavaFileManager.Location](http://docs.google.com/javax/tools/JavaFileManager.Location.html) location)            Determines if a location is known to this file manager. |
| [String](http://docs.google.com/java/lang/String.html) | [**inferBinaryName**](http://docs.google.com/javax/tools/JavaFileManager.html#inferBinaryName(javax.tools.JavaFileManager.Location,%20javax.tools.JavaFileObject))([JavaFileManager.Location](http://docs.google.com/javax/tools/JavaFileManager.Location.html) location, [JavaFileObject](http://docs.google.com/javax/tools/JavaFileObject.html) file)            Infers a binary name of a file object based on a location. |
| boolean | [**isSameFile**](http://docs.google.com/javax/tools/JavaFileManager.html#isSameFile(javax.tools.FileObject,%20javax.tools.FileObject))([FileObject](http://docs.google.com/javax/tools/FileObject.html) a, [FileObject](http://docs.google.com/javax/tools/FileObject.html) b)            Compares two file objects and return true if they represent the same underlying object. |
| [Iterable](http://docs.google.com/java/lang/Iterable.html)<[JavaFileObject](http://docs.google.com/javax/tools/JavaFileObject.html)> | [**list**](http://docs.google.com/javax/tools/JavaFileManager.html#list(javax.tools.JavaFileManager.Location,%20java.lang.String,%20java.util.Set,%20boolean))([JavaFileManager.Location](http://docs.google.com/javax/tools/JavaFileManager.Location.html) location, [String](http://docs.google.com/java/lang/String.html) packageName, [Set](http://docs.google.com/java/util/Set.html)<[JavaFileObject.Kind](http://docs.google.com/javax/tools/JavaFileObject.Kind.html)> kinds, boolean recurse)            Lists all file objects matching the given criteria in the given location. |

| **Methods inherited from interface javax.tools.**[**OptionChecker**](http://docs.google.com/javax/tools/OptionChecker.html) |
| --- |
| [isSupportedOption](http://docs.google.com/javax/tools/OptionChecker.html#isSupportedOption(java.lang.String)) |

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

### getClassLoader

[ClassLoader](http://docs.google.com/java/lang/ClassLoader.html) **getClassLoader**([JavaFileManager.Location](http://docs.google.com/javax/tools/JavaFileManager.Location.html) location)

Gets a class loader for loading plug-ins from the given location. For example, to load annotation processors, a compiler will request a class loader for the [ANNOTATION\_PROCESSOR\_PATH](http://docs.google.com/javax/tools/StandardLocation.html#ANNOTATION_PROCESSOR_PATH) location.

**Parameters:**location - a location **Returns:**a class loader for the given location; or null if loading plug-ins from the given location is disabled or if the location is not known **Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if a class loader can not be created in the current security context [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if [close()](http://docs.google.com/javax/tools/JavaFileManager.html#close()) has been called and this file manager cannot be reopened

### list

[Iterable](http://docs.google.com/java/lang/Iterable.html)<[JavaFileObject](http://docs.google.com/javax/tools/JavaFileObject.html)> **list**([JavaFileManager.Location](http://docs.google.com/javax/tools/JavaFileManager.Location.html) location,  
 [String](http://docs.google.com/java/lang/String.html) packageName,  
 [Set](http://docs.google.com/java/util/Set.html)<[JavaFileObject.Kind](http://docs.google.com/javax/tools/JavaFileObject.Kind.html)> kinds,  
 boolean recurse)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Lists all file objects matching the given criteria in the given location. List file objects in "subpackages" if recurse is true.

Note: even if the given location is unknown to this file manager, it may not return null. Also, an unknown location may not cause an exception.

**Parameters:**location - a locationpackageName - a package namekinds - return objects only of these kindsrecurse - if true include "subpackages" **Returns:**an Iterable of file objects matching the given criteria **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurred, or if [close()](http://docs.google.com/javax/tools/JavaFileManager.html#close()) has been called and this file manager cannot be reopened [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if [close()](http://docs.google.com/javax/tools/JavaFileManager.html#close()) has been called and this file manager cannot be reopened

### inferBinaryName

[String](http://docs.google.com/java/lang/String.html) **inferBinaryName**([JavaFileManager.Location](http://docs.google.com/javax/tools/JavaFileManager.Location.html) location,  
 [JavaFileObject](http://docs.google.com/javax/tools/JavaFileObject.html) file)

Infers a binary name of a file object based on a location. The binary name returned might not be a valid JLS binary name.

**Parameters:**location - a locationfile - a file object **Returns:**a binary name or null the file object is not found in the given location **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if [close()](http://docs.google.com/javax/tools/JavaFileManager.html#close()) has been called and this file manager cannot be reopened

### isSameFile

boolean **isSameFile**([FileObject](http://docs.google.com/javax/tools/FileObject.html) a,  
 [FileObject](http://docs.google.com/javax/tools/FileObject.html) b)

Compares two file objects and return true if they represent the same underlying object.

**Parameters:**a - a file objectb - a file object **Returns:**true if the given file objects represent the same underlying object **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if either of the arguments were created with another file manager and this file manager does not support foreign file objects

### handleOption

boolean **handleOption**([String](http://docs.google.com/java/lang/String.html) current,  
 [Iterator](http://docs.google.com/java/util/Iterator.html)<[String](http://docs.google.com/java/lang/String.html)> remaining)

Handles one option. If current is an option to this file manager it will consume any arguments to that option from remaining and return true, otherwise return false.

**Parameters:**current - current optionremaining - remaining options **Returns:**true if this option was handled by this file manager, false otherwise **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if this option to this file manager is used incorrectly [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if [close()](http://docs.google.com/javax/tools/JavaFileManager.html#close()) has been called and this file manager cannot be reopened

### hasLocation

boolean **hasLocation**([JavaFileManager.Location](http://docs.google.com/javax/tools/JavaFileManager.Location.html) location)

Determines if a location is known to this file manager.

**Parameters:**location - a location **Returns:**true if the location is known

### getJavaFileForInput

[JavaFileObject](http://docs.google.com/javax/tools/JavaFileObject.html) **getJavaFileForInput**([JavaFileManager.Location](http://docs.google.com/javax/tools/JavaFileManager.Location.html) location,  
 [String](http://docs.google.com/java/lang/String.html) className,  
 [JavaFileObject.Kind](http://docs.google.com/javax/tools/JavaFileObject.Kind.html) kind)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Gets a [file object](http://docs.google.com/javax/tools/JavaFileObject.html) for input representing the specified class of the specified kind in the given location.

**Parameters:**location - a locationclassName - the name of a classkind - the kind of file, must be one of [SOURCE](http://docs.google.com/javax/tools/JavaFileObject.Kind.html#SOURCE) or [CLASS](http://docs.google.com/javax/tools/JavaFileObject.Kind.html#CLASS) **Returns:**a file object, might return null if the file does not exist **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the location is not known to this file manager and the file manager does not support unknown locations, or if the kind is not valid [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurred, or if [close()](http://docs.google.com/javax/tools/JavaFileManager.html#close()) has been called and this file manager cannot be reopened [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if [close()](http://docs.google.com/javax/tools/JavaFileManager.html#close()) has been called and this file manager cannot be reopened

### getJavaFileForOutput

[JavaFileObject](http://docs.google.com/javax/tools/JavaFileObject.html) **getJavaFileForOutput**([JavaFileManager.Location](http://docs.google.com/javax/tools/JavaFileManager.Location.html) location,  
 [String](http://docs.google.com/java/lang/String.html) className,  
 [JavaFileObject.Kind](http://docs.google.com/javax/tools/JavaFileObject.Kind.html) kind,  
 [FileObject](http://docs.google.com/javax/tools/FileObject.html) sibling)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Gets a [file object](http://docs.google.com/javax/tools/JavaFileObject.html) for output representing the specified class of the specified kind in the given location.

Optionally, this file manager might consider the sibling as a hint for where to place the output. The exact semantics of this hint is unspecified. Sun's compiler, javac, for example, will place class files in the same directories as originating source files unless a class file output directory is provided. To facilitate this behavior, javac might provide the originating source file as sibling when calling this method.

**Parameters:**location - a locationclassName - the name of a classkind - the kind of file, must be one of [SOURCE](http://docs.google.com/javax/tools/JavaFileObject.Kind.html#SOURCE) or [CLASS](http://docs.google.com/javax/tools/JavaFileObject.Kind.html#CLASS)sibling - a file object to be used as hint for placement; might be null **Returns:**a file object for output **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if sibling is not known to this file manager, or if the location is not known to this file manager and the file manager does not support unknown locations, or if the kind is not valid [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurred, or if [close()](http://docs.google.com/javax/tools/JavaFileManager.html#close()) has been called and this file manager cannot be reopened [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - [close()](http://docs.google.com/javax/tools/JavaFileManager.html#close()) has been called and this file manager cannot be reopened

### getFileForInput

[FileObject](http://docs.google.com/javax/tools/FileObject.html) **getFileForInput**([JavaFileManager.Location](http://docs.google.com/javax/tools/JavaFileManager.Location.html) location,  
 [String](http://docs.google.com/java/lang/String.html) packageName,  
 [String](http://docs.google.com/java/lang/String.html) relativeName)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Gets a [file object](http://docs.google.com/javax/tools/FileObject.html) for input representing the specified [relative name](http://docs.google.com/JavaFileManager.html#relative_name) in the specified package in the given location.

If the returned object represents a [source](http://docs.google.com/javax/tools/JavaFileObject.Kind.html#SOURCE) or [class](http://docs.google.com/javax/tools/JavaFileObject.Kind.html#CLASS) file, it must be an instance of [JavaFileObject](http://docs.google.com/javax/tools/JavaFileObject.html).

Informally, the file object returned by this method is located in the concatenation of the location, package name, and relative name. For example, to locate the properties file "resources/compiler.properties" in the package "com.sun.tools.javac" in the [SOURCE\_PATH](http://docs.google.com/javax/tools/StandardLocation.html#SOURCE_PATH) location, this method might be called like so:

getFileForInput(SOURCE\_PATH, "com.sun.tools.javac", "resources/compiler.properties");

If the call was executed on Windows, with SOURCE\_PATH set to "C:\Documents and Settings\UncleBob\src\share\classes", a valid result would be a file object representing the file "C:\Documents and Settings\UncleBob\src\share\classes\com\sun\tools\javac\resources\compiler.properties".

**Parameters:**location - a locationpackageName - a package namerelativeName - a relative name **Returns:**a file object, might return null if the file does not exist **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the location is not known to this file manager and the file manager does not support unknown locations, or if relativeName is not valid [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurred, or if [close()](http://docs.google.com/javax/tools/JavaFileManager.html#close()) has been called and this file manager cannot be reopened [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if [close()](http://docs.google.com/javax/tools/JavaFileManager.html#close()) has been called and this file manager cannot be reopened

### getFileForOutput

[FileObject](http://docs.google.com/javax/tools/FileObject.html) **getFileForOutput**([JavaFileManager.Location](http://docs.google.com/javax/tools/JavaFileManager.Location.html) location,  
 [String](http://docs.google.com/java/lang/String.html) packageName,  
 [String](http://docs.google.com/java/lang/String.html) relativeName,  
 [FileObject](http://docs.google.com/javax/tools/FileObject.html) sibling)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Gets a [file object](http://docs.google.com/javax/tools/FileObject.html) for output representing the specified [relative name](http://docs.google.com/JavaFileManager.html#relative_name) in the specified package in the given location.

Optionally, this file manager might consider the sibling as a hint for where to place the output. The exact semantics of this hint is unspecified. Sun's compiler, javac, for example, will place class files in the same directories as originating source files unless a class file output directory is provided. To facilitate this behavior, javac might provide the originating source file as sibling when calling this method.

If the returned object represents a [source](http://docs.google.com/javax/tools/JavaFileObject.Kind.html#SOURCE) or [class](http://docs.google.com/javax/tools/JavaFileObject.Kind.html#CLASS) file, it must be an instance of [JavaFileObject](http://docs.google.com/javax/tools/JavaFileObject.html).

Informally, the file object returned by this method is located in the concatenation of the location, package name, and relative name or next to the sibling argument. See [getFileForInput](http://docs.google.com/javax/tools/JavaFileManager.html#getFileForInput(javax.tools.JavaFileManager.Location,%20java.lang.String,%20java.lang.String)) for an example.

**Parameters:**location - a locationpackageName - a package namerelativeName - a relative namesibling - a file object to be used as hint for placement; might be null **Returns:**a file object **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if sibling is not known to this file manager, or if the location is not known to this file manager and the file manager does not support unknown locations, or if relativeName is not valid [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurred, or if [close()](http://docs.google.com/javax/tools/JavaFileManager.html#close()) has been called and this file manager cannot be reopened [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if [close()](http://docs.google.com/javax/tools/JavaFileManager.html#close()) has been called and this file manager cannot be reopened

### flush

void **flush**()  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Flushes any resources opened for output by this file manager directly or indirectly. Flushing a closed file manager has no effect.

**Specified by:**[flush](http://docs.google.com/java/io/Flushable.html#flush()) in interface [Flushable](http://docs.google.com/java/io/Flushable.html) **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurred**See Also:**[close()](http://docs.google.com/javax/tools/JavaFileManager.html#close())

### close

void **close**()  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Releases any resources opened by this file manager directly or indirectly. This might render this file manager useless and the effect of subsequent calls to methods on this object or any objects obtained through this object is undefined unless explicitly allowed. However, closing a file manager which has already been closed has no effect.

**Specified by:**[close](http://docs.google.com/java/io/Closeable.html#close()) in interface [Closeable](http://docs.google.com/java/io/Closeable.html) **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurred**See Also:**[flush()](http://docs.google.com/javax/tools/JavaFileManager.html#flush())

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/JavaFileManager.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/tools/JavaCompiler.CompilationTask.html)   [**NEXT CLASS**](http://docs.google.com/javax/tools/JavaFileManager.Location.html) | [**FRAMES**](http://docs.google.com/index.html?javax/tools/JavaFileManager.html)    [**NO FRAMES**](http://docs.google.com/JavaFileManager.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#2et92p0) | FIELD | CONSTR | [METHOD](#tyjcwt) | DETAIL: FIELD | CONSTR | [METHOD](#1t3h5sf) |

[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).