| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/ForwardingJavaFileManager.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/ForwardingFileObject.html)   [**NEXT CLASS**](http://docs.google.com/javax/tools/ForwardingJavaFileObject.html) | [**FRAMES**](http://docs.google.com/index.html?javax/tools/ForwardingJavaFileManager.html)    [**NO FRAMES**](http://docs.google.com/ForwardingJavaFileManager.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#tyjcwt) | [CONSTR](#3dy6vkm) | [METHOD](#1t3h5sf) | DETAIL: [FIELD](#2s8eyo1) | [CONSTR](#3rdcrjn) | [METHOD](#35nkun2) |

## **javax.tools**

Class ForwardingJavaFileManager<M extends [**JavaFileManager**](http://docs.google.com/javax/tools/JavaFileManager.html)>

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 **javax.tools.ForwardingJavaFileManager<M>**

**Type Parameters:**M - the kind of file manager forwarded to by this object **All Implemented Interfaces:** [Closeable](http://docs.google.com/java/io/Closeable.html), [Flushable](http://docs.google.com/java/io/Flushable.html), [JavaFileManager](http://docs.google.com/javax/tools/JavaFileManager.html), [OptionChecker](http://docs.google.com/javax/tools/OptionChecker.html)

public class **ForwardingJavaFileManager<M extends** [**JavaFileManager**](http://docs.google.com/javax/tools/JavaFileManager.html)**>**extends [Object](http://docs.google.com/java/lang/Object.html)implements [JavaFileManager](http://docs.google.com/javax/tools/JavaFileManager.html)

Forwards calls to a given file manager. Subclasses of this class might override some of these methods and might also provide additional fields and methods.

**Since:** 1.6

| **Nested Class Summary** | |
| --- | --- |

| **Nested classes/interfaces inherited from interface javax.tools.**[**JavaFileManager**](http://docs.google.com/javax/tools/JavaFileManager.html) |
| --- |
| [JavaFileManager.Location](http://docs.google.com/javax/tools/JavaFileManager.Location.html) |

| **Field Summary** | |
| --- | --- |
| protected  [M](http://docs.google.com/javax/tools/ForwardingJavaFileManager.html) | [**fileManager**](http://docs.google.com/javax/tools/ForwardingJavaFileManager.html#fileManager)            The file manager which all methods are delegated to. |

| **Constructor Summary** | |
| --- | --- |
| protected | [**ForwardingJavaFileManager**](http://docs.google.com/javax/tools/ForwardingJavaFileManager.html#ForwardingJavaFileManager(M))([M](http://docs.google.com/javax/tools/ForwardingJavaFileManager.html) fileManager)            Creates a new instance of ForwardingJavaFileManager. |

| **Method Summary** | |
| --- | --- |
| void | [**close**](http://docs.google.com/javax/tools/ForwardingJavaFileManager.html#close())()            Releases any resources opened by this file manager directly or indirectly. |
| void | [**flush**](http://docs.google.com/javax/tools/ForwardingJavaFileManager.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/ForwardingJavaFileManager.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/ForwardingJavaFileManager.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/ForwardingJavaFileManager.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/ForwardingJavaFileManager.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/ForwardingJavaFileManager.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/ForwardingJavaFileManager.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/ForwardingJavaFileManager.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/ForwardingJavaFileManager.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/ForwardingJavaFileManager.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. |
| int | [**isSupportedOption**](http://docs.google.com/javax/tools/ForwardingJavaFileManager.html#isSupportedOption(java.lang.String))([String](http://docs.google.com/java/lang/String.html) option)            Determines if the given option is supported and if so, the number of arguments the option takes. |
| [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/ForwardingJavaFileManager.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 class java.lang.**[**Object**](http://docs.google.com/java/lang/Object.html) |
| --- |
| [clone](http://docs.google.com/java/lang/Object.html#clone()), [equals](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)), [finalize](http://docs.google.com/java/lang/Object.html#finalize()), [getClass](http://docs.google.com/java/lang/Object.html#getClass()), [hashCode](http://docs.google.com/java/lang/Object.html#hashCode()), [notify](http://docs.google.com/java/lang/Object.html#notify()), [notifyAll](http://docs.google.com/java/lang/Object.html#notifyAll()), [toString](http://docs.google.com/java/lang/Object.html#toString()), [wait](http://docs.google.com/java/lang/Object.html#wait()), [wait](http://docs.google.com/java/lang/Object.html#wait(long)), [wait](http://docs.google.com/java/lang/Object.html#wait(long,%20int)) |

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

### fileManager

protected final [M](http://docs.google.com/javax/tools/ForwardingJavaFileManager.html) extends [JavaFileManager](http://docs.google.com/javax/tools/JavaFileManager.html) **fileManager**

The file manager which all methods are delegated to.

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

### ForwardingJavaFileManager

protected **ForwardingJavaFileManager**([M](http://docs.google.com/javax/tools/ForwardingJavaFileManager.html) fileManager)

Creates a new instance of ForwardingJavaFileManager.

**Parameters:**fileManager - delegate to this file manager

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

### getClassLoader

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

**Description copied from interface:** [**JavaFileManager**](http://docs.google.com/javax/tools/JavaFileManager.html#getClassLoader(javax.tools.JavaFileManager.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.

**Specified by:**[getClassLoader](http://docs.google.com/javax/tools/JavaFileManager.html#getClassLoader(javax.tools.JavaFileManager.Location)) in interface [JavaFileManager](http://docs.google.com/javax/tools/JavaFileManager.html) **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 [JavaFileManager.close()](http://docs.google.com/javax/tools/JavaFileManager.html#close()) has been called and this file manager cannot be reopened

### list

public [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)

**Description copied from interface:** [**JavaFileManager**](http://docs.google.com/javax/tools/JavaFileManager.html#list(javax.tools.JavaFileManager.Location,%20java.lang.String,%20java.util.Set,%20boolean)) 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.

**Specified by:**[list](http://docs.google.com/javax/tools/JavaFileManager.html#list(javax.tools.JavaFileManager.Location,%20java.lang.String,%20java.util.Set,%20boolean)) in interface [JavaFileManager](http://docs.google.com/javax/tools/JavaFileManager.html) **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 [JavaFileManager.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 [JavaFileManager.close()](http://docs.google.com/javax/tools/JavaFileManager.html#close()) has been called and this file manager cannot be reopened

### inferBinaryName

public [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)

**Description copied from interface:** [**JavaFileManager**](http://docs.google.com/javax/tools/JavaFileManager.html#inferBinaryName(javax.tools.JavaFileManager.Location,%20javax.tools.JavaFileObject)) Infers a binary name of a file object based on a location. The binary name returned might not be a valid JLS binary name.

**Specified by:**[inferBinaryName](http://docs.google.com/javax/tools/JavaFileManager.html#inferBinaryName(javax.tools.JavaFileManager.Location,%20javax.tools.JavaFileObject)) in interface [JavaFileManager](http://docs.google.com/javax/tools/JavaFileManager.html) **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 [JavaFileManager.close()](http://docs.google.com/javax/tools/JavaFileManager.html#close()) has been called and this file manager cannot be reopened

### isSameFile

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

**Description copied from interface:** [**JavaFileManager**](http://docs.google.com/javax/tools/JavaFileManager.html#isSameFile(javax.tools.FileObject,%20javax.tools.FileObject)) Compares two file objects and return true if they represent the same underlying object.

**Specified by:**[isSameFile](http://docs.google.com/javax/tools/JavaFileManager.html#isSameFile(javax.tools.FileObject,%20javax.tools.FileObject)) in interface [JavaFileManager](http://docs.google.com/javax/tools/JavaFileManager.html) **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

public 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)

**Description copied from interface:** [**JavaFileManager**](http://docs.google.com/javax/tools/JavaFileManager.html#handleOption(java.lang.String,%20java.util.Iterator)) 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.

**Specified by:**[handleOption](http://docs.google.com/javax/tools/JavaFileManager.html#handleOption(java.lang.String,%20java.util.Iterator)) in interface [JavaFileManager](http://docs.google.com/javax/tools/JavaFileManager.html) **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 [JavaFileManager.close()](http://docs.google.com/javax/tools/JavaFileManager.html#close()) has been called and this file manager cannot be reopened

### hasLocation

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

**Description copied from interface:** [**JavaFileManager**](http://docs.google.com/javax/tools/JavaFileManager.html#hasLocation(javax.tools.JavaFileManager.Location)) Determines if a location is known to this file manager.

**Specified by:**[hasLocation](http://docs.google.com/javax/tools/JavaFileManager.html#hasLocation(javax.tools.JavaFileManager.Location)) in interface [JavaFileManager](http://docs.google.com/javax/tools/JavaFileManager.html) **Parameters:**location - a location **Returns:**true if the location is known

### isSupportedOption

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

**Description copied from interface:** [**OptionChecker**](http://docs.google.com/javax/tools/OptionChecker.html#isSupportedOption(java.lang.String)) Determines if the given option is supported and if so, the number of arguments the option takes.

**Specified by:**[isSupportedOption](http://docs.google.com/javax/tools/OptionChecker.html#isSupportedOption(java.lang.String)) in interface [OptionChecker](http://docs.google.com/javax/tools/OptionChecker.html) **Parameters:**option - an option **Returns:**the number of arguments the given option takes or -1 if the option is not supported

### getJavaFileForInput

public [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)

**Description copied from interface:** [**JavaFileManager**](http://docs.google.com/javax/tools/JavaFileManager.html#getJavaFileForInput(javax.tools.JavaFileManager.Location,%20java.lang.String,%20javax.tools.JavaFileObject.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.

**Specified by:**[getJavaFileForInput](http://docs.google.com/javax/tools/JavaFileManager.html#getJavaFileForInput(javax.tools.JavaFileManager.Location,%20java.lang.String,%20javax.tools.JavaFileObject.Kind)) in interface [JavaFileManager](http://docs.google.com/javax/tools/JavaFileManager.html) **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 [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if [JavaFileManager.close()](http://docs.google.com/javax/tools/JavaFileManager.html#close()) has been called and this file manager cannot be reopened [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurred, or if [JavaFileManager.close()](http://docs.google.com/javax/tools/JavaFileManager.html#close()) has been called and this file manager cannot be reopened

### getJavaFileForOutput

public [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)

**Description copied from interface:** [**JavaFileManager**](http://docs.google.com/javax/tools/JavaFileManager.html#getJavaFileForOutput(javax.tools.JavaFileManager.Location,%20java.lang.String,%20javax.tools.JavaFileObject.Kind,%20javax.tools.FileObject)) 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.

**Specified by:**[getJavaFileForOutput](http://docs.google.com/javax/tools/JavaFileManager.html#getJavaFileForOutput(javax.tools.JavaFileManager.Location,%20java.lang.String,%20javax.tools.JavaFileObject.Kind,%20javax.tools.FileObject)) in interface [JavaFileManager](http://docs.google.com/javax/tools/JavaFileManager.html) **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 [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - [JavaFileManager.close()](http://docs.google.com/javax/tools/JavaFileManager.html#close()) has been called and this file manager cannot be reopened [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurred, or if [JavaFileManager.close()](http://docs.google.com/javax/tools/JavaFileManager.html#close()) has been called and this file manager cannot be reopened

### getFileForInput

public [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)

**Description copied from interface:** [**JavaFileManager**](http://docs.google.com/javax/tools/JavaFileManager.html#getFileForInput(javax.tools.JavaFileManager.Location,%20java.lang.String,%20java.lang.String)) 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".

**Specified by:**[getFileForInput](http://docs.google.com/javax/tools/JavaFileManager.html#getFileForInput(javax.tools.JavaFileManager.Location,%20java.lang.String,%20java.lang.String)) in interface [JavaFileManager](http://docs.google.com/javax/tools/JavaFileManager.html) **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 [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if [JavaFileManager.close()](http://docs.google.com/javax/tools/JavaFileManager.html#close()) has been called and this file manager cannot be reopened [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurred, or if [JavaFileManager.close()](http://docs.google.com/javax/tools/JavaFileManager.html#close()) has been called and this file manager cannot be reopened

### getFileForOutput

public [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)

**Description copied from interface:** [**JavaFileManager**](http://docs.google.com/javax/tools/JavaFileManager.html#getFileForOutput(javax.tools.JavaFileManager.Location,%20java.lang.String,%20java.lang.String,%20javax.tools.FileObject)) 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.

**Specified by:**[getFileForOutput](http://docs.google.com/javax/tools/JavaFileManager.html#getFileForOutput(javax.tools.JavaFileManager.Location,%20java.lang.String,%20java.lang.String,%20javax.tools.FileObject)) in interface [JavaFileManager](http://docs.google.com/javax/tools/JavaFileManager.html) **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 [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if [JavaFileManager.close()](http://docs.google.com/javax/tools/JavaFileManager.html#close()) has been called and this file manager cannot be reopened [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurred, or if [JavaFileManager.close()](http://docs.google.com/javax/tools/JavaFileManager.html#close()) has been called and this file manager cannot be reopened

### flush

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

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

### close

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

**Description copied from interface:** [**JavaFileManager**](http://docs.google.com/javax/tools/JavaFileManager.html#close()) 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)**Specified by:**[close](http://docs.google.com/javax/tools/JavaFileManager.html#close()) in interface [JavaFileManager](http://docs.google.com/javax/tools/JavaFileManager.html) **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurred**See Also:**[JavaFileManager.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/ForwardingJavaFileManager.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/ForwardingFileObject.html)   [**NEXT CLASS**](http://docs.google.com/javax/tools/ForwardingJavaFileObject.html) | [**FRAMES**](http://docs.google.com/index.html?javax/tools/ForwardingJavaFileManager.html)    [**NO FRAMES**](http://docs.google.com/ForwardingJavaFileManager.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#tyjcwt) | [CONSTR](#3dy6vkm) | [METHOD](#1t3h5sf) | DETAIL: [FIELD](#2s8eyo1) | [CONSTR](#3rdcrjn) | [METHOD](#35nkun2) |

[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).