| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/FilePermission.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/java/io/FileOutputStream.html)   [**NEXT CLASS**](http://docs.google.com/java/io/FileReader.html) | [**FRAMES**](http://docs.google.com/index.html?java/io/FilePermission.html)    [**NO FRAMES**](http://docs.google.com/FilePermission.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#1t3h5sf) | [METHOD](#2s8eyo1) |

## **java.io**

Class FilePermission

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 [java.security.Permission](http://docs.google.com/java/security/Permission.html)  
 **java.io.FilePermission**

**All Implemented Interfaces:** [Serializable](http://docs.google.com/java/io/Serializable.html), [Guard](http://docs.google.com/java/security/Guard.html)

public final class **FilePermission**extends [Permission](http://docs.google.com/java/security/Permission.html)implements [Serializable](http://docs.google.com/java/io/Serializable.html)

This class represents access to a file or directory. A FilePermission consists of a pathname and a set of actions valid for that pathname.

Pathname is the pathname of the file or directory granted the specified actions. A pathname that ends in "/\*" (where "/" is the file separator character, File.separatorChar) indicates all the files and directories contained in that directory. A pathname that ends with "/-" indicates (recursively) all files and subdirectories contained in that directory. A pathname consisting of the special token "<<ALL FILES>>" matches **any** file.

Note: A pathname consisting of a single "\*" indicates all the files in the current directory, while a pathname consisting of a single "-" indicates all the files in the current directory and (recursively) all files and subdirectories contained in the current directory.

The actions to be granted are passed to the constructor in a string containing a list of one or more comma-separated keywords. The possible keywords are "read", "write", "execute", and "delete". Their meaning is defined as follows:

read read permission write write permission execute execute permission. Allows Runtime.exec to be called. Corresponds to SecurityManager.checkExec. delete delete permission. Allows File.delete to be called. Corresponds to SecurityManager.checkDelete.

The actions string is converted to lowercase before processing.

Be careful when granting FilePermissions. Think about the implications of granting read and especially write access to various files and directories. The "<<ALL FILES>>" permission with write action is especially dangerous. This grants permission to write to the entire file system. One thing this effectively allows is replacement of the system binary, including the JVM runtime environment.

Please note: Code can always read a file from the same directory it's in (or a subdirectory of that directory); it does not need explicit permission to do so.

**Since:** 1.2 **See Also:**[Permission](http://docs.google.com/java/security/Permission.html), [Permissions](http://docs.google.com/java/security/Permissions.html), [PermissionCollection](http://docs.google.com/java/security/PermissionCollection.html)

| **Constructor Summary** | |
| --- | --- |
| [**FilePermission**](http://docs.google.com/java/io/FilePermission.html#FilePermission(java.lang.String,%20java.lang.String))([String](http://docs.google.com/java/lang/String.html) path, [String](http://docs.google.com/java/lang/String.html) actions)            Creates a new FilePermission object with the specified actions. |

| **Method Summary** | |
| --- | --- |
| boolean | [**equals**](http://docs.google.com/java/io/FilePermission.html#equals(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) obj)            Checks two FilePermission objects for equality. |
| [String](http://docs.google.com/java/lang/String.html) | [**getActions**](http://docs.google.com/java/io/FilePermission.html#getActions())()            Returns the "canonical string representation" of the actions. |
| int | [**hashCode**](http://docs.google.com/java/io/FilePermission.html#hashCode())()            Returns the hash code value for this object. |
| boolean | [**implies**](http://docs.google.com/java/io/FilePermission.html#implies(java.security.Permission))([Permission](http://docs.google.com/java/security/Permission.html) p)            Checks if this FilePermission object "implies" the specified permission. |
| [PermissionCollection](http://docs.google.com/java/security/PermissionCollection.html) | [**newPermissionCollection**](http://docs.google.com/java/io/FilePermission.html#newPermissionCollection())()            Returns a new PermissionCollection object for storing FilePermission objects. |

| **Methods inherited from class java.security.**[**Permission**](http://docs.google.com/java/security/Permission.html) |
| --- |
| [checkGuard](http://docs.google.com/java/security/Permission.html#checkGuard(java.lang.Object)), [getName](http://docs.google.com/java/security/Permission.html#getName()), [toString](http://docs.google.com/java/security/Permission.html#toString()) |

| **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)) |

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

### FilePermission

public **FilePermission**([String](http://docs.google.com/java/lang/String.html) path,  
 [String](http://docs.google.com/java/lang/String.html) actions)

Creates a new FilePermission object with the specified actions. *path* is the pathname of a file or directory, and *actions* contains a comma-separated list of the desired actions granted on the file or directory. Possible actions are "read", "write", "execute", and "delete".

A pathname that ends in "/\*" (where "/" is the file separator character, File.separatorChar) indicates all the files and directories contained in that directory. A pathname that ends with "/-" indicates (recursively) all files and subdirectories contained in that directory. The special pathname "<<ALL FILES>>" matches any file.

A pathname consisting of a single "\*" indicates all the files in the current directory, while a pathname consisting of a single "-" indicates all the files in the current directory and (recursively) all files and subdirectories contained in the current directory.

A pathname containing an empty string represents an empty path.

**Parameters:**path - the pathname of the file/directory.actions - the action string. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - If actions is null, empty or contains an action other than the specified possible actions.

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

### implies

public boolean **implies**([Permission](http://docs.google.com/java/security/Permission.html) p)

Checks if this FilePermission object "implies" the specified permission.

More specifically, this method returns true if:

* *p* is an instanceof FilePermission,
* *p*'s actions are a proper subset of this object's actions, and
* *p*'s pathname is implied by this object's pathname. For example, "/tmp/\*" implies "/tmp/foo", since "/tmp/\*" encompasses all files in the "/tmp" directory, including the one named "foo".

**Specified by:**[implies](http://docs.google.com/java/security/Permission.html#implies(java.security.Permission)) in class [Permission](http://docs.google.com/java/security/Permission.html) **Parameters:**p - the permission to check against. **Returns:**true if the specified permission is not null and is implied by this object, false otherwise.

### equals

public boolean **equals**([Object](http://docs.google.com/java/lang/Object.html) obj)

Checks two FilePermission objects for equality. Checks that *obj* is a FilePermission, and has the same pathname and actions as this object.

**Specified by:**[equals](http://docs.google.com/java/security/Permission.html#equals(java.lang.Object)) in class [Permission](http://docs.google.com/java/security/Permission.html) **Parameters:**obj - the object we are testing for equality with this object. **Returns:**true if obj is a FilePermission, and has the same pathname and actions as this FilePermission object, false otherwise.**See Also:**[Object.hashCode()](http://docs.google.com/java/lang/Object.html#hashCode()), [Hashtable](http://docs.google.com/java/util/Hashtable.html)

### hashCode

public int **hashCode**()

Returns the hash code value for this object.

**Specified by:**[hashCode](http://docs.google.com/java/security/Permission.html#hashCode()) in class [Permission](http://docs.google.com/java/security/Permission.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)

### getActions

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

Returns the "canonical string representation" of the actions. That is, this method always returns present actions in the following order: read, write, execute, delete. For example, if this FilePermission object allows both write and read actions, a call to getActions will return the string "read,write".

**Specified by:**[getActions](http://docs.google.com/java/security/Permission.html#getActions()) in class [Permission](http://docs.google.com/java/security/Permission.html) **Returns:**the canonical string representation of the actions.

### newPermissionCollection

public [PermissionCollection](http://docs.google.com/java/security/PermissionCollection.html) **newPermissionCollection**()

Returns a new PermissionCollection object for storing FilePermission objects.

FilePermission objects must be stored in a manner that allows them to be inserted into the collection in any order, but that also enables the PermissionCollection implies method to be implemented in an efficient (and consistent) manner.

For example, if you have two FilePermissions:

1. "/tmp/-", "read"
2. "/tmp/scratch/foo", "write"

and you are calling the implies method with the FilePermission:

"/tmp/scratch/foo", "read,write",

then the implies function must take into account both the "/tmp/-" and "/tmp/scratch/foo" permissions, so the effective permission is "read,write", and implies returns true. The "implies" semantics for FilePermissions are handled properly by the PermissionCollection object returned by this newPermissionCollection method.

**Overrides:**[newPermissionCollection](http://docs.google.com/java/security/Permission.html#newPermissionCollection()) in class [Permission](http://docs.google.com/java/security/Permission.html) **Returns:**a new PermissionCollection object suitable for storing FilePermissions.

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/FilePermission.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/java/io/FileOutputStream.html)   [**NEXT CLASS**](http://docs.google.com/java/io/FileReader.html) | [**FRAMES**](http://docs.google.com/index.html?java/io/FilePermission.html)    [**NO FRAMES**](http://docs.google.com/FilePermission.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#1t3h5sf) | [METHOD](#2s8eyo1) |

[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).