| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/FileLock.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/nio/channels/FileChannel.MapMode.html)   [**NEXT CLASS**](http://docs.google.com/java/nio/channels/FileLockInterruptionException.html) | [**FRAMES**](http://docs.google.com/index.html?java/nio/channels/FileLock.html)    [**NO FRAMES**](http://docs.google.com/FileLock.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#2et92p0) | [METHOD](#tyjcwt) | DETAIL: FIELD | [CONSTR](#1t3h5sf) | [METHOD](#2s8eyo1) |

## **java.nio.channels**

Class FileLock

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 **java.nio.channels.FileLock**

public abstract class **FileLock**extends [Object](http://docs.google.com/java/lang/Object.html)

A token representing a lock on a region of a file.

A file-lock object is created each time a lock is acquired on a file via one of the [lock](http://docs.google.com/java/nio/channels/FileChannel.html#lock(long,%20long,%20boolean)) or [tryLock](http://docs.google.com/java/nio/channels/FileChannel.html#tryLock(long,%20long,%20boolean)) methods of the [FileChannel](http://docs.google.com/java/nio/channels/FileChannel.html) class.

A file-lock object is initially valid. It remains valid until the lock is released by invoking the [release](http://docs.google.com/java/nio/channels/FileLock.html#release()) method, by closing the channel that was used to acquire it, or by the termination of the Java virtual machine, whichever comes first. The validity of a lock may be tested by invoking its [isValid](http://docs.google.com/java/nio/channels/FileLock.html#isValid()) method.

A file lock is either *exclusive* or *shared*. A shared lock prevents other concurrently-running programs from acquiring an overlapping exclusive lock, but does allow them to acquire overlapping shared locks. An exclusive lock prevents other programs from acquiring an overlapping lock of either type. Once it is released, a lock has no further effect on the locks that may be acquired by other programs.

Whether a lock is exclusive or shared may be determined by invoking its [isShared](http://docs.google.com/java/nio/channels/FileLock.html#isShared()) method. Some platforms do not support shared locks, in which case a request for a shared lock is automatically converted into a request for an exclusive lock.

The locks held on a particular file by a single Java virtual machine do not overlap. The [overlaps](http://docs.google.com/java/nio/channels/FileLock.html#overlaps(long,%20long)) method may be used to test whether a candidate lock range overlaps an existing lock.

A file-lock object records the file channel upon whose file the lock is held, the type and validity of the lock, and the position and size of the locked region. Only the validity of a lock is subject to change over time; all other aspects of a lock's state are immutable.

File locks are held on behalf of the entire Java virtual machine. They are not suitable for controlling access to a file by multiple threads within the same virtual machine.

File-lock objects are safe for use by multiple concurrent threads.

#### Platform dependencies

This file-locking API is intended to map directly to the native locking facility of the underlying operating system. Thus the locks held on a file should be visible to all programs that have access to the file, regardless of the language in which those programs are written.

Whether or not a lock actually prevents another program from accessing the content of the locked region is system-dependent and therefore unspecified. The native file-locking facilities of some systems are merely *advisory*, meaning that programs must cooperatively observe a known locking protocol in order to guarantee data integrity. On other systems native file locks are *mandatory*, meaning that if one program locks a region of a file then other programs are actually prevented from accessing that region in a way that would violate the lock. On yet other systems, whether native file locks are advisory or mandatory is configurable on a per-file basis. To ensure consistent and correct behavior across platforms, it is strongly recommended that the locks provided by this API be used as if they were advisory locks.

On some systems, acquiring a mandatory lock on a region of a file prevents that region from being [mapped into memory](http://docs.google.com/java/nio/channels/FileChannel.html#map(java.nio.channels.FileChannel.MapMode,%20long,%20long)), and vice versa. Programs that combine locking and mapping should be prepared for this combination to fail.

On some systems, closing a channel releases all locks held by the Java virtual machine on the underlying file regardless of whether the locks were acquired via that channel or via another channel open on the same file. It is strongly recommended that, within a program, a unique channel be used to acquire all locks on any given file.

Some network filesystems permit file locking to be used with memory-mapped files only when the locked regions are page-aligned and a whole multiple of the underlying hardware's page size. Some network filesystems do not implement file locks on regions that extend past a certain position, often 230 or 231. In general, great care should be taken when locking files that reside on network filesystems.

**Since:** 1.4

| **Constructor Summary** | |
| --- | --- |
| protected | [**FileLock**](http://docs.google.com/java/nio/channels/FileLock.html#FileLock(java.nio.channels.FileChannel,%20long,%20long,%20boolean))([FileChannel](http://docs.google.com/java/nio/channels/FileChannel.html) channel, long position, long size, boolean shared)            Initializes a new instance of this class. |

| **Method Summary** | |
| --- | --- |
| [FileChannel](http://docs.google.com/java/nio/channels/FileChannel.html) | [**channel**](http://docs.google.com/java/nio/channels/FileLock.html#channel())()            Returns the file channel upon whose file this lock is held. |
| boolean | [**isShared**](http://docs.google.com/java/nio/channels/FileLock.html#isShared())()            Tells whether this lock is shared. |
| abstract  boolean | [**isValid**](http://docs.google.com/java/nio/channels/FileLock.html#isValid())()            Tells whether or not this lock is valid. |
| boolean | [**overlaps**](http://docs.google.com/java/nio/channels/FileLock.html#overlaps(long,%20long))(long position, long size)            Tells whether or not this lock overlaps the given lock range. |
| long | [**position**](http://docs.google.com/java/nio/channels/FileLock.html#position())()            Returns the position within the file of the first byte of the locked region. |
| abstract  void | [**release**](http://docs.google.com/java/nio/channels/FileLock.html#release())()            Releases this lock. |
| long | [**size**](http://docs.google.com/java/nio/channels/FileLock.html#size())()            Returns the size of the locked region in bytes. |
| [String](http://docs.google.com/java/lang/String.html) | [**toString**](http://docs.google.com/java/nio/channels/FileLock.html#toString())()            Returns a string describing the range, type, and validity of this lock. |

| **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()), [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** |
| --- |

### FileLock

protected **FileLock**([FileChannel](http://docs.google.com/java/nio/channels/FileChannel.html) channel,  
 long position,  
 long size,  
 boolean shared)

Initializes a new instance of this class.

**Parameters:**channel - The file channel upon whose file this lock is heldposition - The position within the file at which the locked region starts; must be non-negativesize - The size of the locked region; must be non-negative, and the sum position + size must be non-negativeshared - true if this lock is shared, false if it is exclusive **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - If the preconditions on the parameters do not hold

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

### channel

public final [FileChannel](http://docs.google.com/java/nio/channels/FileChannel.html) **channel**()

Returns the file channel upon whose file this lock is held.

**Returns:**The file channel

### position

public final long **position**()

Returns the position within the file of the first byte of the locked region.

A locked region need not be contained within, or even overlap, the actual underlying file, so the value returned by this method may exceed the file's current size.

**Returns:**The position

### size

public final long **size**()

Returns the size of the locked region in bytes.

A locked region need not be contained within, or even overlap, the actual underlying file, so the value returned by this method may exceed the file's current size.

**Returns:**The size of the locked region

### isShared

public final boolean **isShared**()

Tells whether this lock is shared.

**Returns:**true if lock is shared, false if it is exclusive

### overlaps

public final boolean **overlaps**(long position,  
 long size)

Tells whether or not this lock overlaps the given lock range.

**Returns:**true if, and only if, this lock and the given lock range overlap by at least one byte

### isValid

public abstract boolean **isValid**()

Tells whether or not this lock is valid.

A lock object remains valid until it is released or the associated file channel is closed, whichever comes first.

**Returns:**true if, and only if, this lock is valid

### release

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

Releases this lock.

If this lock object is valid then invoking this method releases the lock and renders the object invalid. If this lock object is invalid then invoking this method has no effect.

**Throws:** [ClosedChannelException](http://docs.google.com/java/nio/channels/ClosedChannelException.html) - If the channel that was used to acquire this lock is no longer open [IOException](http://docs.google.com/java/io/IOException.html) - If an I/O error occurs

### toString

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

Returns a string describing the range, type, and validity of this lock.

**Overrides:**[toString](http://docs.google.com/java/lang/Object.html#toString()) in class [Object](http://docs.google.com/java/lang/Object.html) **Returns:**A descriptive string

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/FileLock.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/nio/channels/FileChannel.MapMode.html)   [**NEXT CLASS**](http://docs.google.com/java/nio/channels/FileLockInterruptionException.html) | [**FRAMES**](http://docs.google.com/index.html?java/nio/channels/FileLock.html)    [**NO FRAMES**](http://docs.google.com/FileLock.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#2et92p0) | [METHOD](#tyjcwt) | 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).