| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/ReentrantReadWriteLock.WriteLock.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/util/concurrent/locks/ReentrantReadWriteLock.ReadLock.html)   NEXT CLASS | [**FRAMES**](http://docs.google.com/index.html?java/util/concurrent/locks/ReentrantReadWriteLock.WriteLock.html)    [**NO FRAMES**](http://docs.google.com/ReentrantReadWriteLock.WriteLock.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#3dy6vkm) | [METHOD](#4d34og8) |

## **java.util.concurrent.locks**

Class ReentrantReadWriteLock.WriteLock

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 **java.util.concurrent.locks.ReentrantReadWriteLock.WriteLock**

**All Implemented Interfaces:** [Serializable](http://docs.google.com/java/io/Serializable.html), [Lock](http://docs.google.com/java/util/concurrent/locks/Lock.html) **Enclosing class:**[ReentrantReadWriteLock](http://docs.google.com/java/util/concurrent/locks/ReentrantReadWriteLock.html)

public static class **ReentrantReadWriteLock.WriteLock**extends [Object](http://docs.google.com/java/lang/Object.html)implements [Lock](http://docs.google.com/java/util/concurrent/locks/Lock.html), [Serializable](http://docs.google.com/java/io/Serializable.html)

The lock returned by method [ReentrantReadWriteLock.writeLock()](http://docs.google.com/java/util/concurrent/locks/ReentrantReadWriteLock.html#writeLock()).

**See Also:**[Serialized Form](http://docs.google.com/serialized-form.html#java.util.concurrent.locks.ReentrantReadWriteLock.WriteLock)

| **Constructor Summary** | |
| --- | --- |
| protected | [**ReentrantReadWriteLock.WriteLock**](http://docs.google.com/java/util/concurrent/locks/ReentrantReadWriteLock.WriteLock.html#ReentrantReadWriteLock.WriteLock(java.util.concurrent.locks.ReentrantReadWriteLock))([ReentrantReadWriteLock](http://docs.google.com/java/util/concurrent/locks/ReentrantReadWriteLock.html) lock)            Constructor for use by subclasses |

| **Method Summary** | |
| --- | --- |
| int | [**getHoldCount**](http://docs.google.com/java/util/concurrent/locks/ReentrantReadWriteLock.WriteLock.html#getHoldCount())()            Queries the number of holds on this write lock by the current thread. |
| boolean | [**isHeldByCurrentThread**](http://docs.google.com/java/util/concurrent/locks/ReentrantReadWriteLock.WriteLock.html#isHeldByCurrentThread())()            Queries if this write lock is held by the current thread. |
| void | [**lock**](http://docs.google.com/java/util/concurrent/locks/ReentrantReadWriteLock.WriteLock.html#lock())()            Acquires the write lock. |
| void | [**lockInterruptibly**](http://docs.google.com/java/util/concurrent/locks/ReentrantReadWriteLock.WriteLock.html#lockInterruptibly())()            Acquires the write lock unless the current thread is [interrupted](http://docs.google.com/java/lang/Thread.html#interrupt()). |
| [Condition](http://docs.google.com/java/util/concurrent/locks/Condition.html) | [**newCondition**](http://docs.google.com/java/util/concurrent/locks/ReentrantReadWriteLock.WriteLock.html#newCondition())()            Returns a [Condition](http://docs.google.com/java/util/concurrent/locks/Condition.html) instance for use with this [Lock](http://docs.google.com/java/util/concurrent/locks/Lock.html) instance. |
| [String](http://docs.google.com/java/lang/String.html) | [**toString**](http://docs.google.com/java/util/concurrent/locks/ReentrantReadWriteLock.WriteLock.html#toString())()            Returns a string identifying this lock, as well as its lock state. |
| boolean | [**tryLock**](http://docs.google.com/java/util/concurrent/locks/ReentrantReadWriteLock.WriteLock.html#tryLock())()            Acquires the write lock only if it is not held by another thread at the time of invocation. |
| boolean | [**tryLock**](http://docs.google.com/java/util/concurrent/locks/ReentrantReadWriteLock.WriteLock.html#tryLock(long,%20java.util.concurrent.TimeUnit))(long timeout, [TimeUnit](http://docs.google.com/java/util/concurrent/TimeUnit.html) unit)            Acquires the write lock if it is not held by another thread within the given waiting time and the current thread has not been [interrupted](http://docs.google.com/java/lang/Thread.html#interrupt()). |
| void | [**unlock**](http://docs.google.com/java/util/concurrent/locks/ReentrantReadWriteLock.WriteLock.html#unlock())()            Attempts to release 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** |
| --- |

### ReentrantReadWriteLock.WriteLock

protected **ReentrantReadWriteLock.WriteLock**([ReentrantReadWriteLock](http://docs.google.com/java/util/concurrent/locks/ReentrantReadWriteLock.html) lock)

Constructor for use by subclasses

**Parameters:**lock - the outer lock object **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the lock is null

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

### lock

public void **lock**()

Acquires the write lock.

Acquires the write lock if neither the read nor write lock are held by another thread and returns immediately, setting the write lock hold count to one.

If the current thread already holds the write lock then the hold count is incremented by one and the method returns immediately.

If the lock is held by another thread then the current thread becomes disabled for thread scheduling purposes and lies dormant until the write lock has been acquired, at which time the write lock hold count is set to one.

**Specified by:**[lock](http://docs.google.com/java/util/concurrent/locks/Lock.html#lock()) in interface [Lock](http://docs.google.com/java/util/concurrent/locks/Lock.html)

### lockInterruptibly

public void **lockInterruptibly**()  
 throws [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html)

Acquires the write lock unless the current thread is [interrupted](http://docs.google.com/java/lang/Thread.html#interrupt()).

Acquires the write lock if neither the read nor write lock are held by another thread and returns immediately, setting the write lock hold count to one.

If the current thread already holds this lock then the hold count is incremented by one and the method returns immediately.

If the lock is held by another thread then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of two things happens:

* The write lock is acquired by the current thread; or
* Some other thread [interrupts](http://docs.google.com/java/lang/Thread.html#interrupt()) the current thread.

If the write lock is acquired by the current thread then the lock hold count is set to one.

If the current thread:

* has its interrupted status set on entry to this method; or
* is [interrupted](http://docs.google.com/java/lang/Thread.html#interrupt()) while acquiring the write lock,

then [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html) is thrown and the current thread's interrupted status is cleared.

In this implementation, as this method is an explicit interruption point, preference is given to responding to the interrupt over normal or reentrant acquisition of the lock.

**Specified by:**[lockInterruptibly](http://docs.google.com/java/util/concurrent/locks/Lock.html#lockInterruptibly()) in interface [Lock](http://docs.google.com/java/util/concurrent/locks/Lock.html) **Throws:** [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html) - if the current thread is interrupted

### tryLock

public boolean **tryLock**()

Acquires the write lock only if it is not held by another thread at the time of invocation.

Acquires the write lock if neither the read nor write lock are held by another thread and returns immediately with the value true, setting the write lock hold count to one. Even when this lock has been set to use a fair ordering policy, a call to tryLock() *will* immediately acquire the lock if it is available, whether or not other threads are currently waiting for the write lock. This "barging" behavior can be useful in certain circumstances, even though it breaks fairness. If you want to honor the fairness setting for this lock, then use [tryLock(0, TimeUnit.SECONDS)](http://docs.google.com/java/util/concurrent/locks/ReentrantReadWriteLock.WriteLock.html#tryLock(long,%20java.util.concurrent.TimeUnit))  which is almost equivalent (it also detects interruption).

If the current thread already holds this lock then the hold count is incremented by one and the method returns true.

If the lock is held by another thread then this method will return immediately with the value false.

**Specified by:**[tryLock](http://docs.google.com/java/util/concurrent/locks/Lock.html#tryLock()) in interface [Lock](http://docs.google.com/java/util/concurrent/locks/Lock.html) **Returns:**true if the lock was free and was acquired by the current thread, or the write lock was already held by the current thread; and false otherwise.

### tryLock

public boolean **tryLock**(long timeout,  
 [TimeUnit](http://docs.google.com/java/util/concurrent/TimeUnit.html) unit)  
 throws [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html)

Acquires the write lock if it is not held by another thread within the given waiting time and the current thread has not been [interrupted](http://docs.google.com/java/lang/Thread.html#interrupt()).

Acquires the write lock if neither the read nor write lock are held by another thread and returns immediately with the value true, setting the write lock hold count to one. If this lock has been set to use a fair ordering policy then an available lock *will not* be acquired if any other threads are waiting for the write lock. This is in contrast to the [tryLock()](http://docs.google.com/java/util/concurrent/locks/ReentrantReadWriteLock.WriteLock.html#tryLock()) method. If you want a timed tryLock that does permit barging on a fair lock then combine the timed and un-timed forms together:

if (lock.tryLock() || lock.tryLock(timeout, unit) ) { ... }

If the current thread already holds this lock then the hold count is incremented by one and the method returns true.

If the lock is held by another thread then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of three things happens:

* The write lock is acquired by the current thread; or
* Some other thread [interrupts](http://docs.google.com/java/lang/Thread.html#interrupt()) the current thread; or
* The specified waiting time elapses

If the write lock is acquired then the value true is returned and the write lock hold count is set to one.

If the current thread:

* has its interrupted status set on entry to this method; or
* is [interrupted](http://docs.google.com/java/lang/Thread.html#interrupt()) while acquiring the write lock,

then [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html) is thrown and the current thread's interrupted status is cleared.

If the specified waiting time elapses then the value false is returned. If the time is less than or equal to zero, the method will not wait at all.

In this implementation, as this method is an explicit interruption point, preference is given to responding to the interrupt over normal or reentrant acquisition of the lock, and over reporting the elapse of the waiting time.

**Specified by:**[tryLock](http://docs.google.com/java/util/concurrent/locks/Lock.html#tryLock(long,%20java.util.concurrent.TimeUnit)) in interface [Lock](http://docs.google.com/java/util/concurrent/locks/Lock.html) **Parameters:**timeout - the time to wait for the write lockunit - the time unit of the timeout argument **Returns:**true if the lock was free and was acquired by the current thread, or the write lock was already held by the current thread; and false if the waiting time elapsed before the lock could be acquired. **Throws:** [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html) - if the current thread is interrupted [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the time unit is null

### unlock

public void **unlock**()

Attempts to release this lock.

If the current thread is the holder of this lock then the hold count is decremented. If the hold count is now zero then the lock is released. If the current thread is not the holder of this lock then [IllegalMonitorStateException](http://docs.google.com/java/lang/IllegalMonitorStateException.html) is thrown.

**Specified by:**[unlock](http://docs.google.com/java/util/concurrent/locks/Lock.html#unlock()) in interface [Lock](http://docs.google.com/java/util/concurrent/locks/Lock.html) **Throws:** [IllegalMonitorStateException](http://docs.google.com/java/lang/IllegalMonitorStateException.html) - if the current thread does not hold this lock.

### newCondition

public [Condition](http://docs.google.com/java/util/concurrent/locks/Condition.html) **newCondition**()

Returns a [Condition](http://docs.google.com/java/util/concurrent/locks/Condition.html) instance for use with this [Lock](http://docs.google.com/java/util/concurrent/locks/Lock.html) instance.

The returned [Condition](http://docs.google.com/java/util/concurrent/locks/Condition.html) instance supports the same usages as do the [Object](http://docs.google.com/java/lang/Object.html) monitor methods ([wait](http://docs.google.com/java/lang/Object.html#wait()), [notify](http://docs.google.com/java/lang/Object.html#notify()), and [notifyAll](http://docs.google.com/java/lang/Object.html#notifyAll())) when used with the built-in monitor lock.

* If this write lock is not held when any [Condition](http://docs.google.com/java/util/concurrent/locks/Condition.html) method is called then an [IllegalMonitorStateException](http://docs.google.com/java/lang/IllegalMonitorStateException.html) is thrown. (Read locks are held independently of write locks, so are not checked or affected. However it is essentially always an error to invoke a condition waiting method when the current thread has also acquired read locks, since other threads that could unblock it will not be able to acquire the write lock.)
* When the condition [waiting](http://docs.google.com/java/util/concurrent/locks/Condition.html#await()) methods are called the write lock is released and, before they return, the write lock is reacquired and the lock hold count restored to what it was when the method was called.
* If a thread is [interrupted](http://docs.google.com/java/lang/Thread.html#interrupt()) while waiting then the wait will terminate, an [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html) will be thrown, and the thread's interrupted status will be cleared.
* Waiting threads are signalled in FIFO order.
* The ordering of lock reacquisition for threads returning from waiting methods is the same as for threads initially acquiring the lock, which is in the default case not specified, but for *fair* locks favors those threads that have been waiting the longest.

**Specified by:**[newCondition](http://docs.google.com/java/util/concurrent/locks/Lock.html#newCondition()) in interface [Lock](http://docs.google.com/java/util/concurrent/locks/Lock.html) **Returns:**the Condition object

### toString

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

Returns a string identifying this lock, as well as its lock state. The state, in brackets includes either the String "Unlocked" or the String "Locked by" followed by the [name](http://docs.google.com/java/lang/Thread.html#getName()) of the owning thread.

**Overrides:**[toString](http://docs.google.com/java/lang/Object.html#toString()) in class [Object](http://docs.google.com/java/lang/Object.html) **Returns:**a string identifying this lock, as well as its lock state

### isHeldByCurrentThread

public boolean **isHeldByCurrentThread**()

Queries if this write lock is held by the current thread. Identical in effect to [ReentrantReadWriteLock.isWriteLockedByCurrentThread()](http://docs.google.com/java/util/concurrent/locks/ReentrantReadWriteLock.html#isWriteLockedByCurrentThread()).

**Returns:**true if the current thread holds this lock and false otherwise**Since:** 1.6

### getHoldCount

public int **getHoldCount**()

Queries the number of holds on this write lock by the current thread. A thread has a hold on a lock for each lock action that is not matched by an unlock action. Identical in effect to [ReentrantReadWriteLock.getWriteHoldCount()](http://docs.google.com/java/util/concurrent/locks/ReentrantReadWriteLock.html#getWriteHoldCount()).

**Returns:**the number of holds on this lock by the current thread, or zero if this lock is not held by the current thread**Since:** 1.6

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/ReentrantReadWriteLock.WriteLock.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/util/concurrent/locks/ReentrantReadWriteLock.ReadLock.html)   NEXT CLASS | [**FRAMES**](http://docs.google.com/index.html?java/util/concurrent/locks/ReentrantReadWriteLock.WriteLock.html)    [**NO FRAMES**](http://docs.google.com/ReentrantReadWriteLock.WriteLock.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#3dy6vkm) | [METHOD](#4d34og8) |

[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).