| | [**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.ReadLock.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.html)   [**NEXT CLASS**](http://docs.google.com/java/util/concurrent/locks/ReentrantReadWriteLock.WriteLock.html) | [**FRAMES**](http://docs.google.com/index.html?java/util/concurrent/locks/ReentrantReadWriteLock.ReadLock.html)    [**NO FRAMES**](http://docs.google.com/ReentrantReadWriteLock.ReadLock.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.ReadLock

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

**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.ReadLock**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.readLock()](http://docs.google.com/java/util/concurrent/locks/ReentrantReadWriteLock.html#readLock()).

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

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

| **Method Summary** | |
| --- | --- |
| void | [**lock**](http://docs.google.com/java/util/concurrent/locks/ReentrantReadWriteLock.ReadLock.html#lock())()            Acquires the read lock. |
| void | [**lockInterruptibly**](http://docs.google.com/java/util/concurrent/locks/ReentrantReadWriteLock.ReadLock.html#lockInterruptibly())()            Acquires the read 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.ReadLock.html#newCondition())()            Throws UnsupportedOperationException because ReadLocks do not support conditions. |
| [String](http://docs.google.com/java/lang/String.html) | [**toString**](http://docs.google.com/java/util/concurrent/locks/ReentrantReadWriteLock.ReadLock.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.ReadLock.html#tryLock())()            Acquires the read lock only if the write lock is not held by another thread at the time of invocation. |
| boolean | [**tryLock**](http://docs.google.com/java/util/concurrent/locks/ReentrantReadWriteLock.ReadLock.html#tryLock(long,%20java.util.concurrent.TimeUnit))(long timeout, [TimeUnit](http://docs.google.com/java/util/concurrent/TimeUnit.html) unit)            Acquires the read lock if the write lock 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.ReadLock.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.ReadLock

protected **ReentrantReadWriteLock.ReadLock**([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 read lock.

Acquires the read lock if the write lock is not held by another thread and returns immediately.

If the write lock is held by another thread then the current thread becomes disabled for thread scheduling purposes and lies dormant until the read lock has been acquired.

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

Acquires the read lock if the write lock is not held by another thread and returns immediately.

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

Acquires the read lock if the write lock is not held by another thread and returns immediately with the value true. Even when this lock has been set to use a fair ordering policy, a call to tryLock() *will* immediately acquire the read lock if it is available, whether or not other threads are currently waiting for the read 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.ReadLock.html#tryLock(long,%20java.util.concurrent.TimeUnit))  which is almost equivalent (it also detects interruption).

If the write 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 read lock was acquired

### 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 read lock if the write lock 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 read lock if the write lock is not held by another thread and returns immediately with the value true. 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 lock. This is in contrast to the [tryLock()](http://docs.google.com/java/util/concurrent/locks/ReentrantReadWriteLock.ReadLock.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 write 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 read 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 read lock is acquired then the value true is returned.

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 read 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 read lockunit - the time unit of the timeout argument **Returns:**true if the read lock was 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 number of readers is now zero then the lock is made available for write lock attempts.

**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)

### newCondition

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

Throws UnsupportedOperationException because ReadLocks do not support conditions.

**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:**A new [Condition](http://docs.google.com/java/util/concurrent/locks/Condition.html) instance for this Lock instance **Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - always

### 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 the String "Read locks =" followed by the number of held read locks.

**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

| | [**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.ReadLock.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.html)   [**NEXT CLASS**](http://docs.google.com/java/util/concurrent/locks/ReentrantReadWriteLock.WriteLock.html) | [**FRAMES**](http://docs.google.com/index.html?java/util/concurrent/locks/ReentrantReadWriteLock.ReadLock.html)    [**NO FRAMES**](http://docs.google.com/ReentrantReadWriteLock.ReadLock.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).