| | [**Overview**](http://docs.google.com/overview-summary.html) | **Package** | Class | [**Use**](http://docs.google.com/package-use.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 PACKAGE**](http://docs.google.com/java/util/concurrent/atomic/package-summary.html)   [**NEXT PACKAGE**](http://docs.google.com/java/util/jar/package-summary.html) | [**FRAMES**](http://docs.google.com/index.html?java/util/concurrent/locks/package-summary.html)    [**NO FRAMES**](http://docs.google.com/package-summary.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |

## Package java.util.concurrent.locks

Interfaces and classes providing a framework for locking and waiting for conditions that is distinct from built-in synchronization and monitors.

**See:**

[**Description**](#3znysh7)

| **Interface Summary** | |
| --- | --- |
| [**Condition**](http://docs.google.com/java/util/concurrent/locks/Condition.html) | Condition factors out the Object 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())) into distinct objects to give the effect of having multiple wait-sets per object, by combining them with the use of arbitrary [Lock](http://docs.google.com/java/util/concurrent/locks/Lock.html) implementations. |
| [**Lock**](http://docs.google.com/java/util/concurrent/locks/Lock.html) | Lock implementations provide more extensive locking operations than can be obtained using synchronized methods and statements. |
| [**ReadWriteLock**](http://docs.google.com/java/util/concurrent/locks/ReadWriteLock.html) | A ReadWriteLock maintains a pair of associated [locks](http://docs.google.com/java/util/concurrent/locks/Lock.html), one for read-only operations and one for writing. |

| **Class Summary** | |
| --- | --- |
| [**AbstractOwnableSynchronizer**](http://docs.google.com/java/util/concurrent/locks/AbstractOwnableSynchronizer.html) | A synchronizer that may be exclusively owned by a thread. |
| [**AbstractQueuedLongSynchronizer**](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html) | A version of [AbstractQueuedSynchronizer](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedSynchronizer.html) in which synchronization state is maintained as a long. |
| [**AbstractQueuedSynchronizer**](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedSynchronizer.html) | Provides a framework for implementing blocking locks and related synchronizers (semaphores, events, etc) that rely on first-in-first-out (FIFO) wait queues. |
| [**LockSupport**](http://docs.google.com/java/util/concurrent/locks/LockSupport.html) | Basic thread blocking primitives for creating locks and other synchronization classes. |
| [**ReentrantLock**](http://docs.google.com/java/util/concurrent/locks/ReentrantLock.html) | A reentrant mutual exclusion [Lock](http://docs.google.com/java/util/concurrent/locks/Lock.html) with the same basic behavior and semantics as the implicit monitor lock accessed using synchronized methods and statements, but with extended capabilities. |
| [**ReentrantReadWriteLock**](http://docs.google.com/java/util/concurrent/locks/ReentrantReadWriteLock.html) | An implementation of [ReadWriteLock](http://docs.google.com/java/util/concurrent/locks/ReadWriteLock.html) supporting similar semantics to [ReentrantLock](http://docs.google.com/java/util/concurrent/locks/ReentrantLock.html). |
| [**ReentrantReadWriteLock.ReadLock**](http://docs.google.com/java/util/concurrent/locks/ReentrantReadWriteLock.ReadLock.html) | The lock returned by method [ReentrantReadWriteLock.readLock()](http://docs.google.com/java/util/concurrent/locks/ReentrantReadWriteLock.html#readLock()). |
| [**ReentrantReadWriteLock.WriteLock**](http://docs.google.com/java/util/concurrent/locks/ReentrantReadWriteLock.WriteLock.html) | The lock returned by method [ReentrantReadWriteLock.writeLock()](http://docs.google.com/java/util/concurrent/locks/ReentrantReadWriteLock.html#writeLock()). |

## Package java.util.concurrent.locks Description

Interfaces and classes providing a framework for locking and waiting for conditions that is distinct from built-in synchronization and monitors. The framework permits much greater flexibility in the use of locks and conditions, at the expense of more awkward syntax.

The [Lock](http://docs.google.com/java/util/concurrent/locks/Lock.html) interface supports locking disciplines that differ in semantics (reentrant, fair, etc), and that can be used in non-block-structured contexts including hand-over-hand and lock reordering algorithms. The main implementation is [ReentrantLock](http://docs.google.com/java/util/concurrent/locks/ReentrantLock.html).

The [ReadWriteLock](http://docs.google.com/java/util/concurrent/locks/ReadWriteLock.html) interface similarly defines locks that may be shared among readers but are exclusive to writers. Only a single implementation, [ReentrantReadWriteLock](http://docs.google.com/java/util/concurrent/locks/ReentrantReadWriteLock.html), is provided, since it covers most standard usage contexts. But programmers may create their own implementations to cover nonstandard requirements.

The [Condition](http://docs.google.com/java/util/concurrent/locks/Condition.html) interface describes condition variables that may be associated with Locks. These are similar in usage to the implicit monitors accessed using Object.wait, but offer extended capabilities. In particular, multiple Condition objects may be associated with a single Lock. To avoid compatibility issues, the names of Condition methods are different than the corresponding Object versions.

The [AbstractQueuedSynchronizer](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedSynchronizer.html) class serves as a useful superclass for defining locks and other synchronizers that rely on queuing blocked threads. The [AbstractQueuedLongSynchronizer](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html) class provides the same functionality but extends support to 64 bits of synchronization state. Both extend class [AbstractOwnableSynchronizer](http://docs.google.com/java/util/concurrent/locks/AbstractOwnableSynchronizer.html), a simple class that helps record the thread currently holding exclusive synchronization. The [LockSupport](http://docs.google.com/java/util/concurrent/locks/LockSupport.html) class provides lower-level blocking and unblocking support that is useful for those developers implementing their own customized lock classes.

**Since:** 1.5

| | [**Overview**](http://docs.google.com/overview-summary.html) | **Package** | Class | [**Use**](http://docs.google.com/package-use.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 PACKAGE**](http://docs.google.com/java/util/concurrent/atomic/package-summary.html)   [**NEXT PACKAGE**](http://docs.google.com/java/util/jar/package-summary.html) | [**FRAMES**](http://docs.google.com/index.html?java/util/concurrent/locks/package-summary.html)    [**NO FRAMES**](http://docs.google.com/package-summary.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |

[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).