| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/AbstractQueuedLongSynchronizer.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/AbstractOwnableSynchronizer.html)   [**NEXT CLASS**](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.ConditionObject.html) | [**FRAMES**](http://docs.google.com/index.html?java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html)    [**NO FRAMES**](http://docs.google.com/AbstractQueuedLongSynchronizer.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#3znysh7) | FIELD | [CONSTR](#2et92p0) | [METHOD](#tyjcwt) | DETAIL: FIELD | [CONSTR](#4d34og8) | [METHOD](#17dp8vu) |

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

Class AbstractQueuedLongSynchronizer

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

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

public abstract class **AbstractQueuedLongSynchronizer**extends [AbstractOwnableSynchronizer](http://docs.google.com/java/util/concurrent/locks/AbstractOwnableSynchronizer.html)implements [Serializable](http://docs.google.com/java/io/Serializable.html)

A version of [AbstractQueuedSynchronizer](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedSynchronizer.html) in which synchronization state is maintained as a long. This class has exactly the same structure, properties, and methods as AbstractQueuedSynchronizer with the exception that all state-related parameters and results are defined as long rather than int. This class may be useful when creating synchronizers such as multilevel locks and barriers that require 64 bits of state.

See [AbstractQueuedSynchronizer](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedSynchronizer.html) for usage notes and examples.

**Since:** 1.6 **See Also:**[Serialized Form](http://docs.google.com/serialized-form.html#java.util.concurrent.locks.AbstractQueuedLongSynchronizer)

| **Nested Class Summary** | |
| --- | --- |
| class | [**AbstractQueuedLongSynchronizer.ConditionObject**](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.ConditionObject.html)            Condition implementation for a [AbstractQueuedLongSynchronizer](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html) serving as the basis of a [Lock](http://docs.google.com/java/util/concurrent/locks/Lock.html) implementation. |

| **Constructor Summary** | |
| --- | --- |
| protected | [**AbstractQueuedLongSynchronizer**](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html#AbstractQueuedLongSynchronizer())()            Creates a new AbstractQueuedLongSynchronizer instance with initial synchronization state of zero. |

| **Method Summary** | |
| --- | --- |
| void | [**acquire**](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html#acquire(long))(long arg)            Acquires in exclusive mode, ignoring interrupts. |
| void | [**acquireInterruptibly**](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html#acquireInterruptibly(long))(long arg)            Acquires in exclusive mode, aborting if interrupted. |
| void | [**acquireShared**](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html#acquireShared(long))(long arg)            Acquires in shared mode, ignoring interrupts. |
| void | [**acquireSharedInterruptibly**](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html#acquireSharedInterruptibly(long))(long arg)            Acquires in shared mode, aborting if interrupted. |
| protected  boolean | [**compareAndSetState**](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html#compareAndSetState(long,%20long))(long expect, long update)            Atomically sets synchronization state to the given updated value if the current state value equals the expected value. |
| [Collection](http://docs.google.com/java/util/Collection.html)<[Thread](http://docs.google.com/java/lang/Thread.html)> | [**getExclusiveQueuedThreads**](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html#getExclusiveQueuedThreads())()            Returns a collection containing threads that may be waiting to acquire in exclusive mode. |
| [Thread](http://docs.google.com/java/lang/Thread.html) | [**getFirstQueuedThread**](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html#getFirstQueuedThread())()            Returns the first (longest-waiting) thread in the queue, or null if no threads are currently queued. |
| [Collection](http://docs.google.com/java/util/Collection.html)<[Thread](http://docs.google.com/java/lang/Thread.html)> | [**getQueuedThreads**](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html#getQueuedThreads())()            Returns a collection containing threads that may be waiting to acquire. |
| int | [**getQueueLength**](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html#getQueueLength())()            Returns an estimate of the number of threads waiting to acquire. |
| [Collection](http://docs.google.com/java/util/Collection.html)<[Thread](http://docs.google.com/java/lang/Thread.html)> | [**getSharedQueuedThreads**](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html#getSharedQueuedThreads())()            Returns a collection containing threads that may be waiting to acquire in shared mode. |
| protected  long | [**getState**](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html#getState())()            Returns the current value of synchronization state. |
| [Collection](http://docs.google.com/java/util/Collection.html)<[Thread](http://docs.google.com/java/lang/Thread.html)> | [**getWaitingThreads**](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html#getWaitingThreads(java.util.concurrent.locks.AbstractQueuedLongSynchronizer.ConditionObject))([AbstractQueuedLongSynchronizer.ConditionObject](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.ConditionObject.html) condition)            Returns a collection containing those threads that may be waiting on the given condition associated with this synchronizer. |
| int | [**getWaitQueueLength**](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html#getWaitQueueLength(java.util.concurrent.locks.AbstractQueuedLongSynchronizer.ConditionObject))([AbstractQueuedLongSynchronizer.ConditionObject](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.ConditionObject.html) condition)            Returns an estimate of the number of threads waiting on the given condition associated with this synchronizer. |
| boolean | [**hasContended**](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html#hasContended())()            Queries whether any threads have ever contended to acquire this synchronizer; that is if an acquire method has ever blocked. |
| boolean | [**hasQueuedThreads**](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html#hasQueuedThreads())()            Queries whether any threads are waiting to acquire. |
| boolean | [**hasWaiters**](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html#hasWaiters(java.util.concurrent.locks.AbstractQueuedLongSynchronizer.ConditionObject))([AbstractQueuedLongSynchronizer.ConditionObject](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.ConditionObject.html) condition)            Queries whether any threads are waiting on the given condition associated with this synchronizer. |
| protected  boolean | [**isHeldExclusively**](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html#isHeldExclusively())()            Returns true if synchronization is held exclusively with respect to the current (calling) thread. |
| boolean | [**isQueued**](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html#isQueued(java.lang.Thread))([Thread](http://docs.google.com/java/lang/Thread.html) thread)            Returns true if the given thread is currently queued. |
| boolean | [**owns**](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html#owns(java.util.concurrent.locks.AbstractQueuedLongSynchronizer.ConditionObject))([AbstractQueuedLongSynchronizer.ConditionObject](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.ConditionObject.html) condition)            Queries whether the given ConditionObject uses this synchronizer as its lock. |
| boolean | [**release**](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html#release(long))(long arg)            Releases in exclusive mode. |
| boolean | [**releaseShared**](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html#releaseShared(long))(long arg)            Releases in shared mode. |
| protected  void | [**setState**](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html#setState(long))(long newState)            Sets the value of synchronization state. |
| [String](http://docs.google.com/java/lang/String.html) | [**toString**](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html#toString())()            Returns a string identifying this synchronizer, as well as its state. |
| protected  boolean | [**tryAcquire**](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html#tryAcquire(long))(long arg)            Attempts to acquire in exclusive mode. |
| boolean | [**tryAcquireNanos**](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html#tryAcquireNanos(long,%20long))(long arg, long nanosTimeout)            Attempts to acquire in exclusive mode, aborting if interrupted, and failing if the given timeout elapses. |
| protected  long | [**tryAcquireShared**](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html#tryAcquireShared(long))(long arg)            Attempts to acquire in shared mode. |
| boolean | [**tryAcquireSharedNanos**](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html#tryAcquireSharedNanos(long,%20long))(long arg, long nanosTimeout)            Attempts to acquire in shared mode, aborting if interrupted, and failing if the given timeout elapses. |
| protected  boolean | [**tryRelease**](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html#tryRelease(long))(long arg)            Attempts to set the state to reflect a release in exclusive mode. |
| protected  boolean | [**tryReleaseShared**](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html#tryReleaseShared(long))(long arg)            Attempts to set the state to reflect a release in shared mode. |

| **Methods inherited from class java.util.concurrent.locks.**[**AbstractOwnableSynchronizer**](http://docs.google.com/java/util/concurrent/locks/AbstractOwnableSynchronizer.html) |
| --- |
| [getExclusiveOwnerThread](http://docs.google.com/java/util/concurrent/locks/AbstractOwnableSynchronizer.html#getExclusiveOwnerThread()), [setExclusiveOwnerThread](http://docs.google.com/java/util/concurrent/locks/AbstractOwnableSynchronizer.html#setExclusiveOwnerThread(java.lang.Thread)) |

| **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** |
| --- |

### AbstractQueuedLongSynchronizer

protected **AbstractQueuedLongSynchronizer**()

Creates a new AbstractQueuedLongSynchronizer instance with initial synchronization state of zero.

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

### getState

protected final long **getState**()

Returns the current value of synchronization state. This operation has memory semantics of a volatile read.

**Returns:**current state value

### setState

protected final void **setState**(long newState)

Sets the value of synchronization state. This operation has memory semantics of a volatile write.

**Parameters:**newState - the new state value

### compareAndSetState

protected final boolean **compareAndSetState**(long expect,  
 long update)

Atomically sets synchronization state to the given updated value if the current state value equals the expected value. This operation has memory semantics of a volatile read and write.

**Parameters:**expect - the expected valueupdate - the new value **Returns:**true if successful. False return indicates that the actual value was not equal to the expected value.

### tryAcquire

protected boolean **tryAcquire**(long arg)

Attempts to acquire in exclusive mode. This method should query if the state of the object permits it to be acquired in the exclusive mode, and if so to acquire it.

This method is always invoked by the thread performing acquire. If this method reports failure, the acquire method may queue the thread, if it is not already queued, until it is signalled by a release from some other thread. This can be used to implement method [Lock.tryLock()](http://docs.google.com/java/util/concurrent/locks/Lock.html#tryLock()).

The default implementation throws [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html).

**Parameters:**arg - the acquire argument. This value is always the one passed to an acquire method, or is the value saved on entry to a condition wait. The value is otherwise uninterpreted and can represent anything you like. **Returns:**true if successful. Upon success, this object has been acquired. **Throws:** [IllegalMonitorStateException](http://docs.google.com/java/lang/IllegalMonitorStateException.html) - if acquiring would place this synchronizer in an illegal state. This exception must be thrown in a consistent fashion for synchronization to work correctly. [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if exclusive mode is not supported

### tryRelease

protected boolean **tryRelease**(long arg)

Attempts to set the state to reflect a release in exclusive mode.

This method is always invoked by the thread performing release.

The default implementation throws [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html).

**Parameters:**arg - the release argument. This value is always the one passed to a release method, or the current state value upon entry to a condition wait. The value is otherwise uninterpreted and can represent anything you like. **Returns:**true if this object is now in a fully released state, so that any waiting threads may attempt to acquire; and false otherwise. **Throws:** [IllegalMonitorStateException](http://docs.google.com/java/lang/IllegalMonitorStateException.html) - if releasing would place this synchronizer in an illegal state. This exception must be thrown in a consistent fashion for synchronization to work correctly. [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if exclusive mode is not supported

### tryAcquireShared

protected long **tryAcquireShared**(long arg)

Attempts to acquire in shared mode. This method should query if the state of the object permits it to be acquired in the shared mode, and if so to acquire it.

This method is always invoked by the thread performing acquire. If this method reports failure, the acquire method may queue the thread, if it is not already queued, until it is signalled by a release from some other thread.

The default implementation throws [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html).

**Parameters:**arg - the acquire argument. This value is always the one passed to an acquire method, or is the value saved on entry to a condition wait. The value is otherwise uninterpreted and can represent anything you like. **Returns:**a negative value on failure; zero if acquisition in shared mode succeeded but no subsequent shared-mode acquire can succeed; and a positive value if acquisition in shared mode succeeded and subsequent shared-mode acquires might also succeed, in which case a subsequent waiting thread must check availability. (Support for three different return values enables this method to be used in contexts where acquires only sometimes act exclusively.) Upon success, this object has been acquired. **Throws:** [IllegalMonitorStateException](http://docs.google.com/java/lang/IllegalMonitorStateException.html) - if acquiring would place this synchronizer in an illegal state. This exception must be thrown in a consistent fashion for synchronization to work correctly. [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if shared mode is not supported

### tryReleaseShared

protected boolean **tryReleaseShared**(long arg)

Attempts to set the state to reflect a release in shared mode.

This method is always invoked by the thread performing release.

The default implementation throws [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html).

**Parameters:**arg - the release argument. This value is always the one passed to a release method, or the current state value upon entry to a condition wait. The value is otherwise uninterpreted and can represent anything you like. **Returns:**true if this release of shared mode may permit a waiting acquire (shared or exclusive) to succeed; and false otherwise **Throws:** [IllegalMonitorStateException](http://docs.google.com/java/lang/IllegalMonitorStateException.html) - if releasing would place this synchronizer in an illegal state. This exception must be thrown in a consistent fashion for synchronization to work correctly. [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if shared mode is not supported

### isHeldExclusively

protected boolean **isHeldExclusively**()

Returns true if synchronization is held exclusively with respect to the current (calling) thread. This method is invoked upon each call to a non-waiting [AbstractQueuedLongSynchronizer.ConditionObject](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.ConditionObject.html) method. (Waiting methods instead invoke [release(long)](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html#release(long)).)

The default implementation throws [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html). This method is invoked internally only within [AbstractQueuedLongSynchronizer.ConditionObject](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.ConditionObject.html) methods, so need not be defined if conditions are not used.

**Returns:**true if synchronization is held exclusively; false otherwise **Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if conditions are not supported

### acquire

public final void **acquire**(long arg)

Acquires in exclusive mode, ignoring interrupts. Implemented by invoking at least once [tryAcquire(long)](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html#tryAcquire(long)), returning on success. Otherwise the thread is queued, possibly repeatedly blocking and unblocking, invoking [tryAcquire(long)](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html#tryAcquire(long)) until success. This method can be used to implement method [Lock.lock()](http://docs.google.com/java/util/concurrent/locks/Lock.html#lock()).

**Parameters:**arg - the acquire argument. This value is conveyed to [tryAcquire(long)](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html#tryAcquire(long)) but is otherwise uninterpreted and can represent anything you like.

### acquireInterruptibly

public final void **acquireInterruptibly**(long arg)  
 throws [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html)

Acquires in exclusive mode, aborting if interrupted. Implemented by first checking interrupt status, then invoking at least once [tryAcquire(long)](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html#tryAcquire(long)), returning on success. Otherwise the thread is queued, possibly repeatedly blocking and unblocking, invoking [tryAcquire(long)](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html#tryAcquire(long)) until success or the thread is interrupted. This method can be used to implement method [Lock.lockInterruptibly()](http://docs.google.com/java/util/concurrent/locks/Lock.html#lockInterruptibly()).

**Parameters:**arg - the acquire argument. This value is conveyed to [tryAcquire(long)](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html#tryAcquire(long)) but is otherwise uninterpreted and can represent anything you like. **Throws:** [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html) - if the current thread is interrupted

### tryAcquireNanos

public final boolean **tryAcquireNanos**(long arg,  
 long nanosTimeout)  
 throws [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html)

Attempts to acquire in exclusive mode, aborting if interrupted, and failing if the given timeout elapses. Implemented by first checking interrupt status, then invoking at least once [tryAcquire(long)](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html#tryAcquire(long)), returning on success. Otherwise, the thread is queued, possibly repeatedly blocking and unblocking, invoking [tryAcquire(long)](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html#tryAcquire(long)) until success or the thread is interrupted or the timeout elapses. This method can be used to implement method [Lock.tryLock(long, TimeUnit)](http://docs.google.com/java/util/concurrent/locks/Lock.html#tryLock(long,%20java.util.concurrent.TimeUnit)).

**Parameters:**arg - the acquire argument. This value is conveyed to [tryAcquire(long)](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html#tryAcquire(long)) but is otherwise uninterpreted and can represent anything you like.nanosTimeout - the maximum number of nanoseconds to wait **Returns:**true if acquired; false if timed out **Throws:** [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html) - if the current thread is interrupted

### release

public final boolean **release**(long arg)

Releases in exclusive mode. Implemented by unblocking one or more threads if [tryRelease(long)](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html#tryRelease(long)) returns true. This method can be used to implement method [Lock.unlock()](http://docs.google.com/java/util/concurrent/locks/Lock.html#unlock()).

**Parameters:**arg - the release argument. This value is conveyed to [tryRelease(long)](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html#tryRelease(long)) but is otherwise uninterpreted and can represent anything you like. **Returns:**the value returned from [tryRelease(long)](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html#tryRelease(long))

### acquireShared

public final void **acquireShared**(long arg)

Acquires in shared mode, ignoring interrupts. Implemented by first invoking at least once [tryAcquireShared(long)](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html#tryAcquireShared(long)), returning on success. Otherwise the thread is queued, possibly repeatedly blocking and unblocking, invoking [tryAcquireShared(long)](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html#tryAcquireShared(long)) until success.

**Parameters:**arg - the acquire argument. This value is conveyed to [tryAcquireShared(long)](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html#tryAcquireShared(long)) but is otherwise uninterpreted and can represent anything you like.

### acquireSharedInterruptibly

public final void **acquireSharedInterruptibly**(long arg)  
 throws [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html)

Acquires in shared mode, aborting if interrupted. Implemented by first checking interrupt status, then invoking at least once [tryAcquireShared(long)](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html#tryAcquireShared(long)), returning on success. Otherwise the thread is queued, possibly repeatedly blocking and unblocking, invoking [tryAcquireShared(long)](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html#tryAcquireShared(long)) until success or the thread is interrupted.

**Parameters:**arg - the acquire argument. This value is conveyed to [tryAcquireShared(long)](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html#tryAcquireShared(long)) but is otherwise uninterpreted and can represent anything you like. **Throws:** [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html) - if the current thread is interrupted

### tryAcquireSharedNanos

public final boolean **tryAcquireSharedNanos**(long arg,  
 long nanosTimeout)  
 throws [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html)

Attempts to acquire in shared mode, aborting if interrupted, and failing if the given timeout elapses. Implemented by first checking interrupt status, then invoking at least once [tryAcquireShared(long)](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html#tryAcquireShared(long)), returning on success. Otherwise, the thread is queued, possibly repeatedly blocking and unblocking, invoking [tryAcquireShared(long)](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html#tryAcquireShared(long)) until success or the thread is interrupted or the timeout elapses.

**Parameters:**arg - the acquire argument. This value is conveyed to [tryAcquireShared(long)](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html#tryAcquireShared(long)) but is otherwise uninterpreted and can represent anything you like.nanosTimeout - the maximum number of nanoseconds to wait **Returns:**true if acquired; false if timed out **Throws:** [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html) - if the current thread is interrupted

### releaseShared

public final boolean **releaseShared**(long arg)

Releases in shared mode. Implemented by unblocking one or more threads if [tryReleaseShared(long)](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html#tryReleaseShared(long)) returns true.

**Parameters:**arg - the release argument. This value is conveyed to [tryReleaseShared(long)](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html#tryReleaseShared(long)) but is otherwise uninterpreted and can represent anything you like. **Returns:**the value returned from [tryReleaseShared(long)](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html#tryReleaseShared(long))

### hasQueuedThreads

public final boolean **hasQueuedThreads**()

Queries whether any threads are waiting to acquire. Note that because cancellations due to interrupts and timeouts may occur at any time, a true return does not guarantee that any other thread will ever acquire.

In this implementation, this operation returns in constant time.

**Returns:**true if there may be other threads waiting to acquire

### hasContended

public final boolean **hasContended**()

Queries whether any threads have ever contended to acquire this synchronizer; that is if an acquire method has ever blocked.

In this implementation, this operation returns in constant time.

**Returns:**true if there has ever been contention

### getFirstQueuedThread

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

Returns the first (longest-waiting) thread in the queue, or null if no threads are currently queued.

In this implementation, this operation normally returns in constant time, but may iterate upon contention if other threads are concurrently modifying the queue.

**Returns:**the first (longest-waiting) thread in the queue, or null if no threads are currently queued

### isQueued

public final boolean **isQueued**([Thread](http://docs.google.com/java/lang/Thread.html) thread)

Returns true if the given thread is currently queued.

This implementation traverses the queue to determine presence of the given thread.

**Parameters:**thread - the thread **Returns:**true if the given thread is on the queue **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the thread is null

### getQueueLength

public final int **getQueueLength**()

Returns an estimate of the number of threads waiting to acquire. The value is only an estimate because the number of threads may change dynamically while this method traverses internal data structures. This method is designed for use in monitoring system state, not for synchronization control.

**Returns:**the estimated number of threads waiting to acquire

### getQueuedThreads

public final [Collection](http://docs.google.com/java/util/Collection.html)<[Thread](http://docs.google.com/java/lang/Thread.html)> **getQueuedThreads**()

Returns a collection containing threads that may be waiting to acquire. Because the actual set of threads may change dynamically while constructing this result, the returned collection is only a best-effort estimate. The elements of the returned collection are in no particular order. This method is designed to facilitate construction of subclasses that provide more extensive monitoring facilities.

**Returns:**the collection of threads

### getExclusiveQueuedThreads

public final [Collection](http://docs.google.com/java/util/Collection.html)<[Thread](http://docs.google.com/java/lang/Thread.html)> **getExclusiveQueuedThreads**()

Returns a collection containing threads that may be waiting to acquire in exclusive mode. This has the same properties as [getQueuedThreads()](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html#getQueuedThreads()) except that it only returns those threads waiting due to an exclusive acquire.

**Returns:**the collection of threads

### getSharedQueuedThreads

public final [Collection](http://docs.google.com/java/util/Collection.html)<[Thread](http://docs.google.com/java/lang/Thread.html)> **getSharedQueuedThreads**()

Returns a collection containing threads that may be waiting to acquire in shared mode. This has the same properties as [getQueuedThreads()](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html#getQueuedThreads()) except that it only returns those threads waiting due to a shared acquire.

**Returns:**the collection of threads

### toString

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

Returns a string identifying this synchronizer, as well as its state. The state, in brackets, includes the String "State =" followed by the current value of [getState()](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html#getState()), and either "nonempty" or "empty" depending on whether the queue is empty.

**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 synchronizer, as well as its state

### owns

public final boolean **owns**([AbstractQueuedLongSynchronizer.ConditionObject](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.ConditionObject.html) condition)

Queries whether the given ConditionObject uses this synchronizer as its lock.

**Parameters:**condition - the condition **Returns:**true if owned **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the condition is null

### hasWaiters

public final boolean **hasWaiters**([AbstractQueuedLongSynchronizer.ConditionObject](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.ConditionObject.html) condition)

Queries whether any threads are waiting on the given condition associated with this synchronizer. Note that because timeouts and interrupts may occur at any time, a true return does not guarantee that a future signal will awaken any threads. This method is designed primarily for use in monitoring of the system state.

**Parameters:**condition - the condition **Returns:**true if there are any waiting threads **Throws:** [IllegalMonitorStateException](http://docs.google.com/java/lang/IllegalMonitorStateException.html) - if exclusive synchronization is not held [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the given condition is not associated with this synchronizer [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the condition is null

### getWaitQueueLength

public final int **getWaitQueueLength**([AbstractQueuedLongSynchronizer.ConditionObject](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.ConditionObject.html) condition)

Returns an estimate of the number of threads waiting on the given condition associated with this synchronizer. Note that because timeouts and interrupts may occur at any time, the estimate serves only as an upper bound on the actual number of waiters. This method is designed for use in monitoring of the system state, not for synchronization control.

**Parameters:**condition - the condition **Returns:**the estimated number of waiting threads **Throws:** [IllegalMonitorStateException](http://docs.google.com/java/lang/IllegalMonitorStateException.html) - if exclusive synchronization is not held [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the given condition is not associated with this synchronizer [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the condition is null

### getWaitingThreads

public final [Collection](http://docs.google.com/java/util/Collection.html)<[Thread](http://docs.google.com/java/lang/Thread.html)> **getWaitingThreads**([AbstractQueuedLongSynchronizer.ConditionObject](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.ConditionObject.html) condition)

Returns a collection containing those threads that may be waiting on the given condition associated with this synchronizer. Because the actual set of threads may change dynamically while constructing this result, the returned collection is only a best-effort estimate. The elements of the returned collection are in no particular order.

**Parameters:**condition - the condition **Returns:**the collection of threads **Throws:** [IllegalMonitorStateException](http://docs.google.com/java/lang/IllegalMonitorStateException.html) - if exclusive synchronization is not held [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the given condition is not associated with this synchronizer [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the condition is null

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/AbstractQueuedLongSynchronizer.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/AbstractOwnableSynchronizer.html)   [**NEXT CLASS**](http://docs.google.com/java/util/concurrent/locks/AbstractQueuedLongSynchronizer.ConditionObject.html) | [**FRAMES**](http://docs.google.com/index.html?java/util/concurrent/locks/AbstractQueuedLongSynchronizer.html)    [**NO FRAMES**](http://docs.google.com/AbstractQueuedLongSynchronizer.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#3znysh7) | FIELD | [CONSTR](#2et92p0) | [METHOD](#tyjcwt) | DETAIL: FIELD | [CONSTR](#4d34og8) | [METHOD](#17dp8vu) |

[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).