SEMAPHORES JAVA

Constructor and Description

Semaphore (int permits)
Creates a Semaphore with the given number of permits and nonfair fairness setting.
Semaphore (int permits, boolean fair)
Creates a Semaphore with the given number of permits and the given fairness setting.

Methods Modifier and Type Method and Description

void **acquire**()
Acquires a permit from this semaphore, blocking until one is available, or the thread is interrupted.

void **acquire**(int permits)
Acquires the given number of permits from this semaphore, blocking until all are available, or the thread is interrupted.

void **acquireUninterruptibly**()
Acquires a permit from this semaphore, blocking until one is available.

void **acquireUninterruptibly**(int permits)
Acquires the given number of permits from this semaphore, blocking until all are available.

int **availablePermits**()
Returns the current number of permits available in this semaphore.

int **drainPermits**()
Acquires and returns all permits that are immediately available.

protected Collection<Thread> **getQueuedThreads**()
Returns a collection containing threads that may be waiting to acquire.

int **getQueueLength**()
Returns an estimate of the number of threads waiting to acquire.

boolean **hasQueuedThreads**()
Queries whether any threads are waiting to acquire.

boolean **isFair**()
Returns true if this semaphore has fairness set true.

protected void **reducePermits**(int reduction)
Shrinks the number of available permits by the indicated reduction.

void **release**()
Releases a permit, returning it to the semaphore.

void **release**(int permits)
Releases the given number of permits, returning them to the semaphore.

String **toString**()
Returns a string identifying this semaphore, as well as its state.

boolean **tryAcquire**()
Acquires a permit from this semaphore, only if one is available at the time of invocation.

boolean **tryAcquire**(int permits)
Acquires the given number of permits from this semaphore, only if all are available at the time of invocation.

boolean **tryAcquire**(int permits, long timeout, TimeUnit unit)
Acquires the given number of permits from this semaphore, if all become available within the given waiting time and the current thread has not been interrupted.

boolean **tryAcquire**(long timeout, TimeUnit unit)
Acquires a permit from this semaphore, if one becomes available within the given waiting time and the current thread has not been interrupted.

Constructor Detail

public **Semaphore**(int permits)

Creates a Semaphore with the given number of permits and nonfair fairness setting.

Parameters:
permits - the initial number of permits available. This value may be negative, in which case releases must occur before any acquires will be granted.

public **Semaphore**(int permits, boolean fair)

Creates a Semaphore with the given number of permits and the given fairness setting.

Parameters:
permits - the initial number of permits available. This value may be negative, in which case releases must occur before any acquires will be granted.
fair - true if this semaphore will guarantee first-in first-out granting of permits under contention, else false

Method Detail

public void **acquire**()
throws InterruptedException

Acquires a permit from this semaphore, blocking until one is available, or the thread is interrupted.

Acquires a permit, if one is available and returns immediately, reducing the number of available permits by one.
If no permit is available then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of two things happens:
Some other thread invokes the release() method for this semaphore and the current thread is next to be assigned a permit; or

Some other thread interrupts the current thread.

If the current thread:
has its interrupted status set on entry to this method; or
is interrupted while waiting for a permit,
then InterruptedException is thrown and the current thread's interrupted status is cleared.

Throws:
InterruptedException - if the current thread is interrupted

public void **release**()

Releases a permit, returning it to the semaphore.

Releases a permit, increasing the number of available permits by one. If any threads are trying to acquire a permit, then one is selected and given the permit that was just released. That thread is (re)enabled for thread scheduling purposes.

There is no requirement that a thread that releases a permit must have acquired that permit by calling acquire(). Correct usage of a semaphore is established by programming convention in the application.

public boolean **tryAcquire**()

Acquires a permit from this semaphore, only if one is available at the time of invocation.

Acquires a permit, if one is available and returns immediately, with the value true, reducing the number of available permits by one.

If no permit is available then this method will return immediately with the value false.

Even when this semaphore has been set to use a fair ordering policy, a call to tryAcquire() will immediately acquire a permit if one is available, whether or not other threads are currently waiting. This "barging" behavior can be useful in certain circumstances, even though it breaks fairness. If you want to honor the fairness setting, then use tryAcquire(0, TimeUnit.SECONDS) which is almost equivalent (it also detects interruption).

Returns:
true if a permit was acquired and false otherwise

Version with parameter

public void **acquire**(int permits)
throws InterruptedException

Acquires the given number of permits from this semaphore, blocking until all are available, or the thread is interrupted.

Acquires the given number of permits, if they are available, and returns immediately, reducing the number of available permits by the given amount.

If insufficient permits are available then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of two things happens:

Some other thread invokes one of the release methods for this semaphore, the current thread is next to be assigned permits and the number of available permits satisfies this request; or

Some other thread interrupts the current thread.

If the current thread:

has its interrupted status set on entry to this method; or

is interrupted while waiting for a permit,

then InterruptedException is thrown and the current thread's interrupted status is cleared. Any permits that were to be assigned to this thread are instead assigned to other threads trying to acquire permits, as if permits had been made available by a call to release().

Parameters:

permits - the number of permits to acquire

Throws:

InterruptedException - if the current thread is interrupted

IllegalArgumentException - if permits is negative

public void **release**(int permits)

Releases the given number of permits, returning them to the semaphore.

Releases the given number of permits, increasing the number of available permits by that amount. If any threads are trying to acquire permits, then one is selected and given the permits that were just released. If the number of available permits satisfies that thread's request then that thread is (re)enabled for thread scheduling purposes; otherwise the thread will wait until sufficient permits are available. If there are still permits available after this thread's request has been satisfied, then those permits are assigned in turn to other threads trying to acquire permits.

There is no requirement that a thread that releases a permit must have acquired that permit by calling acquire. Correct usage of a semaphore is established by programming convention in the application.

Parameters:

permits - the number of permits to release

Throws:

IllegalArgumentException - if permits is negative