

(/)

Semaphores in Java

Last modified: April 27, 2020

by baeldung (<https://www.baeldung.com/author/baeldung/>)

Java (<https://www.baeldung.com/category/java/>) +

Java Concurrency (<https://www.baeldung.com/tag/java-concurrency/>)

I just announced the new *Learn Spring* course, focused on the fundamentals of Spring 5 and Spring Boot 2:

>> CHECK OUT THE COURSE (</ls-course-start>)

1. Overview

In this quick



2. Semaphore

We'll start with `java.util.concurrent.Semaphore`. We can use semaphores to limit the number of concurrent threads accessing a specific resource.

In the following example, we will implement a simple login queue to limit the number of users in the system:

We use cookies to improve your experience with the site. To find out more, you can read the full [Privacy and Cookie Policy](#) ([/privacy-policy](#)).

Ok



```

1  class LoginQueueUsingSemaphore {
2
3      private Semaphore semaphore;
4
5      public LoginQueueUsingSemaphore(int slotLimit) {
6          semaphore = new Semaphore(slotLimit);
7      }
8
9      boolean tryLogin() {
10         return semaphore.tryAcquire();
11     }
12
13     void logout() {
14         semaphore.release();
15     }
16
17     int availableSlots() {
18         return semaphore.availablePermits();
19     }
20
21 }

```

Notice how we used the following methods:

- *tryAcquire()* – return true if a permit is available immediately and acquire it otherwise return false, but *acquire()* acquires a permit and blocking until one is available
- *release()* – *release a permit*
- *availablePermits()* – return number of current permits available

To test our login queue, we will first try to reach the limit and check if the next login attempt will be blocked:

```

1  @Test
2  public void givenLoginQueue_whenReachLimit_thenBlocked() {
3
4
5
6
7      // ... (previous code) ...
8      executorService.shutdown();
9
10     assertEquals(0, loginQueue.availableSlots());
11     assertFalse(loginQueue.tryLogin());
12 }

```



Next, we will see if any slots are available after a logout:

```

1  @Test
2  public void givenLoginQueue_whenLogout_thenSlotsAvailable() {
3      int slots = 10;
4      ExecutorService executorService = Executors.newFixedThreadPool(slots);
5      LoginQueueUsingSemaphore loginQueue = new LoginQueueUsingSemaphore(slots);
6      IntStream.range(0, slots)
7          .forEach(user -> executorService.execute(loginQueue::tryLogin));
8      executorService.shutdown();
9      assertEquals(0, loginQueue.availableSlots());
10     loginQueue.logout();
11
12     assertTrue(loginQueue.availableSlots() > 0);
13     assertTrue(loginQueue.tryLogin());
14 }

```

3. Timed Semaphore

Next, we will discuss Apache Commons *TimedSemaphore*. *TimedSemaphore* allows a number of permits as a simple Semaphore but in a given period of time, after this period the time reset and all permits are released.

Ok

We can use *TimedSemaphore* to build a simple delay queue as follows:

```

1  class DelayQueueUsingTimedSemaphore {
2
3      private TimedSemaphore semaphore;
4
5      DelayQueueUsingTimedSemaphore(long period, int slotLimit) {
6          semaphore = new TimedSemaphore(period, TimeUnit.SECONDS, slotLimit);
7      }
8
9      boolean tryAdd() {
10         return semaphore.tryAcquire();
11     }
12
13     int availableSlots() {
14         return semaphore.getAvailablePermits();
15     }
16
17 }

```

When we use a delay queue with one second as time period and after using all the slots within one second, none should be available:

```

1  public void givenDelayQueue_whenReachLimit_thenBlocked() {
2      int slots = 50;
3      ExecutorService executorService = Executors.newFixedThreadPool(slots);
4      DelayQueueUsingTimedSemaphore delayQueue
5          = new DelayQueueUsingTimedSemaphore(1, slots);
6
7      IntStream.range(0, slots)
8          .forEach(user -> executorService.execute(delayQueue::tryAdd));
9      executorService.shutdown();
10
11      assertEquals(0, delayQueue.availableSlots());
12
13  }

```

But after s

```

1  @Test
2  public void givenDelayQueue_whenTimePass_thenSlotsAvailable() throws InterruptedException {
3      int slots = 50;
4      ExecutorService executorService = Executors.newFixedThreadPool(slots);
5      DelayQueueUsingTimedSemaphore delayQueue = new DelayQueueUsingTimedSemaphore(1, slots);
6      IntStream.range(0, slots)
7          .forEach(user -> executorService.execute(delayQueue::tryAdd));
8      executorService.shutdown();
9
10     assertEquals(0, delayQueue.availableSlots());
11     Thread.sleep(1000);
12     assertTrue(delayQueue.availableSlots() > 0);
13     assertTrue(delayQueue.tryAdd());
14 }

```

4. Semaphore vs. Mutex

Mutex acts similarly to a binary semaphore, we can use it to implement mutual exclusion.

In the following example, we'll use a simple binary semaphore to build a counter:

```
1  class CounterUsingMutex {
2
3      private Semaphore mutex;
4      private int count;
5
6      CounterUsingMutex() {
7          mutex = new Semaphore(1);
8          count = 0;
9      }
10
11     void increase() throws InterruptedException {
12         mutex.acquire();
13         this.count = this.count + 1;
14         Thread.sleep(1000);
15         mutex.release();
16     }
17
18     int getCount() {
19         return this.count;
20     }
21
22
23
24
25
26 }
```

When a lot of threads try to access the counter at once, **they'll simply be blocked in a queue:**

```
1  @Test
2  public void whenMutexAndMultipleThreads_thenBlocked()
3      throws InterruptedException {
4      int count = 5;
5      ExecutorService executorService
6          = Executors.newFixedThreadPool(count);
7      CounterUsingMutex counter = new CounterUsingMutex();
8      IntStream.range(0, count)
9          .forEach(user -> executorService.execute(() -> {
10              try {
11                  counter.increase();
12              } catch (InterruptedException e) {
13                  e.printStackTrace();
14              }
15          }));
16      executorService.shutdown();
17
18      assertTrue(counter.hasQueuedThreads());
19  }
```

When we wait, all threads will access the counter and no threads left in the queue:

```
1  @Test
2  public void givenMutexAndMultipleThreads_ThenDelay_thenCorrectCount()
3      throws InterruptedException {
4      int count = 5;
5      ExecutorService executorService
6          = Executors.newFixedThreadPool(count);
7      CounterUsingMutex counter = new CounterUsingMutex();
8      IntStream.range(0, count)
9          .forEach(user -> executorService.execute(() -> {
10              try {
11                  counter.increase();
12              } catch (InterruptedException e) {
13                  e.printStackTrace();
14              }
15          }));
16      executorService.shutdown();
17
18      assertTrue(counter.hasQueuedThreads());
19      Thread.sleep(5000);
20      assertFalse(counter.hasQueuedThreads());
21      assertEquals(count, counter.getCount());
22  }
```

5. Conclusion

In this article, we explored the basics of semaphores in Java.

As always, the full source code is available over on GitHub

(<https://github.com/eugenp/tutorials/tree/master/core-java-modules/core-java-concurrency-advanced-2>).

I just announced the new *Learn Spring* course, focused on the fundamentals of Spring 5 and Spring Boot 2:

>> CHECK OUT THE COURSE (/ls-course-end)



Learning to "Build your API with Spring"?

Enter your email address

>> Get the eBook

Comments are closed on this article!

CATEGORIES

SPRING ([HTTPS://WWW.BAELDUNG.COM/CATEGORY/SPRING/](https://www.baeldung.com/category/spring/))
REST ([HTTPS://WWW.BAELDUNG.COM/CATEGORY/REST/](https://www.baeldung.com/category/rest/))
JAVA ([HTTPS://WWW.BAELDUNG.COM/CATEGORY/JAVA/](https://www.baeldung.com/category/java/))
SECURITY ([HTTPS://WWW.BAELDUNG.COM/CATEGORY/SECURITY-2/](https://www.baeldung.com/category/security-2/))
PERSISTENCE ([HTTPS://WWW.BAELDUNG.COM/CATEGORY/PERSISTENCE/](https://www.baeldung.com/category/persistence/))
JACKSON ([HTTPS://WWW.BAELDUNG.COM/CATEGORY/JSON/JACKSON/](https://www.baeldung.com/category/json/jackson/))
HTTP CLIENT-SIDE ([HTTPS://WWW.BAELDUNG.COM/CATEGORY/HTTP/](https://www.baeldung.com/category/http/))
KOTLIN ([HTTPS://WWW.BAELDUNG.COM/CATEGORY/KOTLIN/](https://www.baeldung.com/category/kotlin/))

SERIES

JAVA "BACK TO BASICS" TUTORIAL ([/JAVA-TUTORIAL](#))
JACKSON JSON TUTORIAL ([/JACKSON](#))
HTTPCLIENT 4 TUTORIAL ([/HTTPCLIENT-GUIDE](#))
REST WITH SPRING TUTORIAL ([/REST-WITH-SPRING-SERIES](#))
SPRING PERSISTENCE TUTORIAL ([/PERSISTENCE-WITH-SPRING-SERIES](#))
SECURITY WITH SPRING ([/SECURITY-SPRING](#))

We use cookies to improve your experience with the site. To find out more, you can read the full [Privacy and Cookie Policy](#). ([privacy-policy](#))



ABOUT

[ABOUT BAELDUNG \(/ABOUT\)](#)

[THE COURSES \(HTTPS://COURSES.BAELDUNG.COM\)](https://courses.baeldung.com)

[JOBS \(/TAG/ACTIVE-JOB/\)](#)

[THE FULL ARCHIVE \(/FULL_ARCHIVE\)](#)

[WRITE FOR BAELDUNG \(/CONTRIBUTION-GUIDELINES\)](#)

[EDITORS \(/EDITORS\)](#)

[OUR PARTNERS \(/PARTNERS\)](#)

[ADVERTISE ON BAELDUNG \(/ADVERTISE\)](#)

[TERMS OF SERVICE \(/TERMS-OF-SERVICE\)](#)

[PRIVACY POLICY \(/PRIVACY-POLICY\)](#)

[COMPANY INFO \(/BAELDUNG-COMPANY-INFO\)](#)

[CONTACT \(/CONTACT\)](#)

We use cookies to improve your experience with the site. To find out more, you can read the full [Privacy and Cookie Policy.\(/privacy-policy/\)](#)

Ok