Java Thread – Mutex and Semaphore example

By Datsabk (https://www.mkyong.com/author/datsabk/) | December 4, 2016 | Viewed : 3,337 times +237 pv/w

Java multi threads example to show you how to use `Semaphore` and `Mutex` to limit the number of threads to access resources.

1. `Semaphores` – Restrict the number of threads that can access a resource. Example, limit max 10 connections to access a file simultaneously.
2. `Mutex` – Only one thread to access a resource at once. Example, when a client is accessing a file, no one else should have access the same file at the same time.

1. Semaphore

Consider an ATM cubicle with 4 ATMs, `Semaphore` can make sure only 4 people can access simultaneously.

```
SemaphoreTest.java
```

```
package com.mkyong;

import java.util.concurrent.Semaphore;

public class SemaphoreTest {

    // max 4 people
    static Semaphore semaphore = new Semaphore(4);

    static class MyATMThread extends Thread {

        String name = "";

        MyATMThread(String name) {
            this.name = name;
        }

        public void run() {

            try {


                System.out.println(name + " : acquiring lock...");
                System.out.println(name + " : available Semaphore permits now: "
                                + semaphore.availablePermits());

                semaphore.acquire();
                System.out.println(name + " : got the permit!");

                try {

                    for (int i = 1; i <= 5; i++) {

                        System.out.println(name + " : is performing operation " + i
                                + ", available Semaphore permits : "
                                + semaphore.availablePermits());

                        // sleep 1 second
                        Thread.sleep(1000);

                    }

                } finally {

                    // calling release() after a successful acquire()
                    System.out.println(name + " : releasing lock...");
                    semaphore.release();
                    System.out.println(name + " : available Semaphore permits now: "
                                + semaphore.availablePermits());

                }

            } catch (InterruptedException e) {

                e.printStackTrace();

            }

        }

    }

    public static void main(String[] args) {

        System.out.println("Total available Semaphore permits : "
                + semaphore.availablePermits());

        MyATMThread t1 = new MyATMThread("A");
        t1.start();

        MyATMThread t2 = new MyATMThread("B");
        t2.start();

        MyATMThread t3 = new MyATMThread("C");
        t3.start();
        MyATMThread t4 = new MyATMThread("D");
        t4.start();

        MyATMThread t5 = new MyATMThread("E");
        t5.start();

        MyATMThread t6 = new MyATMThread("F");
        t6.start();

    }
}
```

Output may vary, but the flow of locking and releasing should be more or less same.

```
Total available Semaphore permits : 4
A : acquiring lock...
D : acquiring lock...
C : acquiring lock...
B : acquiring lock...
B : available Semaphore permits now: 4
C : available Semaphore permits now: 4
E : acquiring lock...
F : acquiring lock...
F : available Semaphore permits now: 2
F : got the permit!
F : is performing operation 1, available Semaphore permits : 1
D : available Semaphore permits now: 4
A : available Semaphore permits now: 4
D : got the permit!
D : is performing operation 1, available Semaphore permits : 0
E : available Semaphore permits now: 2
C : got the permit!
B : got the permit!
C : is performing operation 1, available Semaphore permits : 0
B : is performing operation 1, available Semaphore permits : 0
F : is performing operation 2, available Semaphore permits : 0
D : is performing operation 2, available Semaphore permits : 0
C : is performing operation 2, available Semaphore permits : 0
B : is performing operation 2, available Semaphore permits : 0
F : is performing operation 3, available Semaphore permits : 0
D : is performing operation 3, available Semaphore permits : 0
C : is performing operation 3, available Semaphore permits : 0
B : is performing operation 3, available Semaphore permits : 0
F : is performing operation 4, available Semaphore permits : 0
D : is performing operation 4, available Semaphore permits : 0
C : is performing operation 4, available Semaphore permits : 0
B : is performing operation 4, available Semaphore permits : 0
D : is performing operation 5, available Semaphore permits : 0
F : is performing operation 5, available Semaphore permits : 0
B : is performing operation 5, available Semaphore permits : 0
C : is performing operation 5, available Semaphore permits : 0


D : releasing lock...
F : releasing lock...
D : available Semaphore permits now: 1
A : got the permit!
A : is performing operation 1, available Semaphore permits : 0
F : available Semaphore permits now: 1
E : got the permit!


E : is performing operation 1, available Semaphore permits : 0
B : releasing lock...
B : available Semaphore permits now: 1
C : releasing lock...
C : available Semaphore permits now: 2
A : is performing operation 2, available Semaphore permits : 2
E : is performing operation 2, available Semaphore permits : 2
A : is performing operation 3, available Semaphore permits : 2
E : is performing operation 3, available Semaphore permits : 2
A : is performing operation 4, available Semaphore permits : 2
E : is performing operation 4, available Semaphore permits : 2
A : is performing operation 5, available Semaphore permits : 2
E : is performing operation 5, available Semaphore permits : 2
A : releasing lock...
A : available Semaphore permits now: 3
E : releasing lock...
E : available Semaphore permits now: 4
```

Observe the above output carefully, you will see that there are maximum 4 people (C, B, F, D) to perform an operation at a time, the people A and E are waiting. As soon as one of them release the lock (D and F), A and E will acquire it and resumes immediately.

2. Mutex

`Mutex` is the `Semaphore` with an access count of 1. Consider a situation of using lockers in the bank. Usually the rule is that only one person is allowed to enter the locker room.

```
MutexTest.java
```

```
package com.mkyong;

import java.util.concurrent.Semaphore;

public class SemaphoreTest {

    // max 1 people
    static Semaphore semaphore = new Semaphore(1);

    static class MyLockerThread extends Thread {

        String name = "";

        MyLockerThread(String name) {
            this.name = name;
        }

        public void run() {

            try {

                System.out.println(name + " : acquiring lock...");
                System.out.println(name + " : available Semaphore permits now: "
                                + semaphore.availablePermits());

                semaphore.acquire();
                System.out.println(name + " : got the permit!");

                try {

                    for (int i = 1; i <= 5; i++) {

                        System.out.println(name + " : is performing operation " + i
                                + ", available Semaphore permits : "
                                + semaphore.availablePermits());

                        // sleep 1 second
                        Thread.sleep(1000);

                    }

                } finally {

                    // calling release() after a successful acquire()
                    System.out.println(name + " : releasing lock...");
                    semaphore.release();
                    System.out.println(name + " : available Semaphore permits now: "
                                + semaphore.availablePermits());

                }

            } catch (InterruptedException e) {

                e.printStackTrace();

            }

        }

    }

    public static void main(String[] args) {

        System.out.println("Total available Semaphore permits : "
                + semaphore.availablePermits());

        MyLockerThread t1 = new MyLockerThread("A");
        t1.start();

        MyLockerThread t2 = new MyLockerThread("B");
        t2.start();

        MyLockerThread t3 = new MyLockerThread("C");
        t3.start();

        MyLockerThread t4 = new MyLockerThread("D");
        t4.start();

        MyLockerThread t5 = new MyLockerThread("E");
        t5.start();

        MyLockerThread t6 = new MyLockerThread("F");
        t6.start();

    }
}
```

Output may vary, but the flow of locking and releasing should be same.

```
Total available Semaphore permits : 1
A : acquiring lock...
B : acquiring lock...
A : available Semaphore permits now: 1
C : acquiring lock...
B : available Semaphore permits now: 1
C : available Semaphore permits now: 0
A : got the permit!
D : acquiring lock...
E : acquiring lock...
A : is performing operation 1, available Semaphore permits : 0
E : available Semaphore permits now: 0
D : available Semaphore permits now: 0
F : acquiring lock...
F : available Semaphore permits now: 0
A : is performing operation 2, available Semaphore permits : 0
A : is performing operation 3, available Semaphore permits : 0
A : is performing operation 4, available Semaphore permits : 0
A : is performing operation 5, available Semaphore permits : 0
A : releasing lock...
A : available Semaphore permits now: 1
B : got the permit!
B : is performing operation 1, available Semaphore permits : 0
B : is performing operation 2, available Semaphore permits : 0
B : is performing operation 3, available Semaphore permits : 0
B : is performing operation 4, available Semaphore permits : 0
B : is performing operation 5, available Semaphore permits : 0
B : releasing lock...
B : available Semaphore permits now: 1
C : got the permit!
C : is performing operation 1, available Semaphore permits : 0
C : is performing operation 2, available Semaphore permits : 0
C : is performing operation 3, available Semaphore permits : 0
C : is performing operation 4, available Semaphore permits : 0
C : is performing operation 5, available Semaphore permits : 0
C : releasing lock...
C : available Semaphore permits now: 1
E : got the permit!
E : is performing operation 1, available Semaphore permits : 0
E : is performing operation 2, available Semaphore permits : 0
E : is performing operation 3, available Semaphore permits : 0
E : is performing operation 4, available Semaphore permits : 0
E : is performing operation 5, available Semaphore permits : 0
E : releasing lock...
E : available Semaphore permits now: 1
D : got the permit!
D : is performing operation 1, available Semaphore permits : 0
D : is performing operation 2, available Semaphore permits : 0
D : is performing operation 3, available Semaphore permits : 0
D : is performing operation 4, available Semaphore permits : 0
D : is performing operation 5, available Semaphore permits : 0
D : releasing lock...
D : available Semaphore permits now: 1
F : got the permit!
F : is performing operation 1, available Semaphore permits : 0
F : is performing operation 2, available Semaphore permits : 0
F : is performing operation 3, available Semaphore permits : 0
F : is performing operation 4, available Semaphore permits : 0
F : is performing operation 5, available Semaphore permits : 0
F : releasing lock...
F : available Semaphore permits now: 1
```

As it can be seen, only one thread executes at a time here. This is the role of a Mutex.

References

1. Semaphore Javadocs (https://docs.oracle.com/javase/7/docs/api/java/util/concurrent/Semaphore.html)
2. What is mutex and semaphore in Java ? What is the main difference? (http://stackoverflow.com/questions/771347/what-is-mutex-and-semaphore-in-java-what-is-the-main-difference)
3. Wikipedia - Semaphore (https://en.wikipedia.org/wiki/Semaphore_(programming))
4. Is there a Mutex in Java? (http://stackoverflow.com/questions/5291041/is-there-a-mutex-in-java)
5. Concurrent Programming with J2SE 5.0 (http://www.oracle.com/technetwork/articles/javase/index-140767.html)
6. Programming Java threads in the real world, Part 2 (http://www.javaworld.com/article/2076797/java-concurrency/programming-java-threads-in-the-real-world--part-2.html)

Tags : java thread (https://www.mkyong.com/tag/java-thread/) mutex (https://www.mkyong.com/tag/mutex/) semaphore (https://www.mkyong.com/tag/semaphore/) thread (https://www.mkyong.com/tag/thread/)

Share this article on

Twitter (https://twitter.com/intent/tweet?text=Java Thread - Mutex and Semaphore example&url=https://www.mkyong.com/java/java-thread-mutex-and-semaphore-example/&via=mkyong) Facebook (https://www.facebook.com/sharer/sharer.php?u=https://www.mkyong.com/java/java-thread-mutex-and-semaphore-example/) Google+ (https://plus.google.com/share?url=https://www.mkyong.com/java/java-thread-mutex-and-semaphore-example/)

About the Author

Datsabk

A technically sound person, oriented towards learning things logically rather than technically. It is my logical approach that has helped me learn and take up any programming language and start with coding. With a responsibility of Java based Web development professionally, I highly believe in imparting knowledge online for any technical subject that I can handle.

Related Posts

218k	Spring and Java Thread example (/spring/spring-and-java-thread-example/)
17k	Review : Java 7 Concurrency Cookbook (/book-review/review-java-7-concurrency-cookbook/)

Popular Posts

1m

JAX-WS Tutorial (/tutorials/jax-ws-tutorials/)

1.5m

How to convert Java object to / from JSON (Jackson) (/java/how-to-convert-java-object-to-from-json-jackson/)

1.5m

Hibernate Tutorial (/tutorials/hibernate-tutorials/)

810k

Java Date and Calendar examples (/java/java-date-and-calendar-examples/)

772k

Hibernate Query examples (HQL) (/hibernate/hibernate-query-examples-hql/)

1.1m