Lab CC 7: Semaphores

Objectives

In this lab, you will create counting semaphore to control access to four resources by five threads

Instructions

Step 1: The Resource Class

1. Implement the counting semaphore as a static variable for the Resource class. We want the semaphore to be independent of any specific instance of a resource.
2. Each resource will have a name and be runnable in its own thread

```
public class Resource extends Thread {

    static Semaphore semaphore = new Semaphore(4);

    String name = null;

    Resource(String n) {
        this.name = n;
    }
```

Step 2: The run() method

1. The first part of the run method prints out the number of available permits then tries to acquire one of the permits.
2. It will block until it gets the permit, and when it does, it prints out the message that it has got one.
3. The thread then emulates doing some work while also checking to see how many permits are available

4. When the work is done, the permit is release in the finally block with the thread printing out the fact its permit is released and the updated number of permits available.

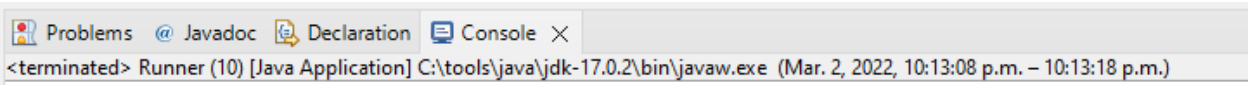
```
@Override
public void run() {
    try {

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

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

        try {
            for (int i = 1; i <= 5; i++) {
                System.out.println(this.name + " : is performing operation "
                        + i + ", available Semaphore permits : "
                        + Resource.semaphore.availablePermits());
                Thread.sleep(1000);
            }

        } finally {
            System.out.println(this.name + " : releasing lock ... ");
            Resource.semaphore.release();
            System.out.println(
                    this.name + " : available Semaphore permits now: "
                            + Resource.semaphore.availablePermits());

        }

    } catch (InterruptedException e) {

        System.out.println(e);

    }
}
```

Part 3: The Runner Class

1. The runner creates five threads sets them running

```
public class Runner {

    public static void main(String[] args) {
        System.out.println("Total available Semaphore permits : "
                + Resource.semaphore.availablePermits());

        Resource t1 = new Resource("Alpha");
        t1.start();
        Resource t2 = new Resource("Beta");
        t2.start();
        Resource t3 = new Resource("Gamma");
        t3.start();
        Resource t4 = new Resource("Delta");
        t4.start();
        Resource t5 = new Resource("Epsilon");
        t5.start();



    }
```

2. The output should look something like that shown on the next page.




```
Total available Semaphore permits : 4
Gamma : acquiring lock ...
Alpha : acquiring lock ...
Delta : acquiring lock ...
Beta : acquiring lock ...
Delta : available Semaphore permits now: 4
Alpha : available Semaphore permits now: 4
Delta : got the permit!
Gamma : available Semaphore permits now: 4
Delta : is performing operation 1, available Semaphore permits : 2
Alpha : got the permit!
Epsilon : acquiring lock ...
Beta : available Semaphore permits now: 4
Epsilon : available Semaphore permits now: 1
Alpha : is performing operation 1, available Semaphore permits : 1
Gamma : got the permit!
Beta : got the permit!
Gamma : is performing operation 1, available Semaphore permits : 0
Beta : is performing operation 1, available Semaphore permits : 0
Gamma : is performing operation 2, available Semaphore permits : 0
Delta : is performing operation 2, available Semaphore permits : 0
Beta : is performing operation 2, available Semaphore permits : 0
Alpha : is performing operation 2, available Semaphore permits : 0
Beta : is performing operation 3, available Semaphore permits : 0
Delta : is performing operation 3, available Semaphore permits : 0
Alpha : is performing operation 3, available Semaphore permits : 0
Gamma : is performing operation 3, available Semaphore permits : 0
Beta : is performing operation 4, available Semaphore permits : 0
Delta : is performing operation 4, available Semaphore permits : 0
Alpha : is performing operation 4, available Semaphore permits : 0
Gamma : is performing operation 4, available Semaphore permits : 0
Beta : is performing operation 5, available Semaphore permits : 0
Delta : is performing operation 5, available Semaphore permits : 0
Alpha : is performing operation 5, available Semaphore permits : 0
Gamma : is performing operation 5, available Semaphore permits : 0
Delta : releasing lock ...
Beta : releasing lock ...
```