Update readme.md
Jonas Solhaug Kaad authored 2 months ago

Forked from an inaccessible project.

readme.md 2.73 KiB

Practice Assignment 4

This task is based on multithreading and thread synchronization. Consider a scenario of a vehicle counting system which counts the number of cars entering the SDU parking area. Assuming there are enough parking slots available in the parking area, two vehicles enter the parking area, whereas one vehicle leaves the parking area. All vehicles' arrival and departure can be considered as threads, and the vehicle counting system can be considered as a single object, which is modified by these threads. In this task, you need to avoid race condition, such that when one thread is accessing the state of the object, another thread will wait to access the same object at a time until their turn comes.

In this task, we have provided 2 classes called `ThreadDemo.java` and `VehicleCounting.java` in the package `multithreading`.

Task 1:

Complete the implementation of `VehicleCounting.java` *(Remember to use the correct access modifiers for variables, constructor and methods).*

- Declare a variable `counter` of type `int`.
- Create a one argument constructor to initialize the variable `counter`.
- Create a `getCounter()` method for retrieving the value of variable. The method should return the value of the variable.
- Implement `incrementCounter()` method with the signature `void incrementCounter()`
  - Increment the variable `counter` by 1.

    *(**Hint** : You will need to use some form of implicit/explicit thread synchronization to avoid race condition).*
- Implement `decrementCounter()` method with the signature `void decrementCounter()`
  - Decrement the variable `counter` by 1.

    *(**Hint** : You will need to use some form of implicit/explicit thread synchronization to avoid race condition).*

Task 2:

- Create a class `ArrivalTask.java` with the signature `public class ArrivalTask implements Runnable` in the *multithreading* package.
- Declare a variable `vc` of type `VehicleCounting`.
- Create a one argument constructor to initialize the variable `vc`
- Implement the `run()` method such that it invokes the `incrementCounter()` method of `VehicleCounting`.

Task 3:

- Create a class `DepartureTask.java` with the signature `public class DepartureTask implements Runnable` in the *multithreading* package.
- Declare a variable `vc` of type `VehicleCounting`.
- Create a one argument constructor to initialize the variable `vc`
- Implement the `run()` method such that it invokes the `decrementCounter()` method of `VehicleCounting`.

Task 4:

Time to make sure it all works! A `main()` method is already implemented in `ThreadDemo.java` but the lines of code in the method are commented out.

- Uncomment the code to test your implementation.

Example of correct output

```
No. of parked vehicles before arrival and departure=5
No. of parked vehicles after arrival and departure=6
```

```java
package multithreading;

public class VehicleCounting {
    int counter;

    public VehicleCounting(int counter){
        this.counter = counter;
    }

    public int getCounter() {
        return counter;
    }

    public void incrementCounter(){
        counter++;
    }

    public void decrementCounter(){
        counter--;
    }

}

```

```java
package multithreading;

public class ArrivalTask implements Runnable{

    VehicleCounting vc;

    public ArrivalTask(VehicleCounting vc){
        this.vc = vc;
    }

    @Override
    public void run() {
        vc.incrementCounter();
    }
}

```

File - C:\Users\vivek\Downloads\Practice_Assignment4_VOP\src\main\java\multithreading\DepartureTask.java

```java
package multithreading;

public class DepartureTask implements Runnable{
    VehicleCounting vc;

    public DepartureTask(VehicleCounting vc){
        this.vc = vc;
    }

    @Override
    public void run() {
        vc.decrementCounter();
    }

}

```

File - C:\Users\vivek\Downloads\Practice_Assignment4_VOP\src\main\java\multithreading\ThreadDemo.java

```java
package multithreading;

import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class ThreadDemo {

    public static void main (String [] args) {

        VehicleCounting vc=new VehicleCounting(5);
        System.out.println("No. of parked vehicles before arrival and departure=" + vc.getCounter());

        // Create Tasks
        ArrivalTask p1=new ArrivalTask(vc);
        ArrivalTask p2=new ArrivalTask(vc);
        DepartureTask p3=new DepartureTask(vc);

        ExecutorService executor = Executors.newFixedThreadPool(3);
        executor.execute(p1);
        executor.execute(p2);
        executor.execute(p3);

        executor.shutdown();

        // Wait until all threads are finished
        while (!executor.isTerminated()) {
        }

        System.out.println("No. of parked vehicles after arrival and departure=" + vc.getCounter());
    }


}

```