Java Week 6: Q4

Due on 2020-10-29, 23:59 IST

Execution of two or more threads occurs in a random order. The keyword 'synchronized' in Java is used to control the execution of thread in a strict sequence. In the following, ***the program is expected to print some numbers***. Use 'synchronized' keyword, so that, ***the program prints the output in the following order:***

----------OUTPUT----------
5
10
15
20
25
100
200
300
400
500

Private Test cases used for evaluation	Input	Expected Output	Actual Output	Status
Test Case 1		5\n 10\n 15\n 20\n 25\n 100\n 200\n 300\n 400\n 500\n	5\n 10\n 15\n 20\n 25\n 100\n 200\n 300\n 400\n 500\n	Passed

The due date for submitting this assignment has passed.
1 out of 1 tests passed.
You scored 100.0/100.

Assignment submitted on 2020-10-21, 15:55 IST

Your last recorded submission was :

```
class Execute{
synchronized void print(int n)
    {
            for(int i=1;i<=5;i++)
            {
                    System.out.println(n*i);
                    try
                    {
                        Thread.sleep(400);
                    }
                    catch(Exception e)
                    {
                        System.out.println(e);
                    }
            }
    }
} // Ending Execute class

class Thread1 extends Thread{
    Execute t;
    Thread1(Execute t){
        this.t=t;
    }
    public void run(){
        t.print(5);
    }
}

class Thread2 extends Thread{
    Execute t;
    Thread2(Execute t){
        this.t=t;
    }
    public void run(){
        t.print(100);
    }
}

public class Question64{
    public static void main(String args[]){
        Execute obj = new Execute();//only one object
        Thread1 t1=new Thread1(obj);
        Thread2 t2=new Thread2(obj);
        t1.start();
        t2.start();
    }
}

```

Sample solutions (Provided by instructor)

```
class Execute{
// Just add 'synchronized' in the method
synchronized void print(int n){
   for(int i=1;i<=5;i++){
     System.out.println(n*i);
     try{
      Thread.sleep(400);
     }catch(Exception e){
        System.out.println(e);
```

```
    }
   }
 }

} // Ending Execute class

class Thread1 extends Thread{
    Execute t;
    Thread1(Execute t){
        this.t=t;
    }
    public void run(){
        t.print(5);
    }
}

class Thread2 extends Thread{
    Execute t;
    Thread2(Execute t){
        this.t=t;
    }
    public void run(){
        t.print(100);
    }
}

public class Question64{
    public static void main(String args[]){
        Execute obj = new Execute();//only one object
        Thread1 t1=new Thread1(obj);
        Thread2 t2=new Thread2(obj);
        t1.start();
        t2.start();
    }
}

```