Final Exam

Raad Alnashri|444003328

Task1

```java
/*Raad Mohammad Ali Alnashri
444003328
 */
package org.example.task1;

Edit | Explain | Test | Document | Fix
class MyThread extends Thread { 2 usages
    public void run () {
        System .out . println (" Thread is running . Name : " + getName ());
    }
}
Edit | Explain | Test | Document | Fix
public class Main {
    Edit | Explain | Test | Document | Fix
    public static void main ( String [] args ) {
        MyThread t1 = new MyThread ();
        t1. setName (" MyFirstThread ");
        t1. start ();
    }
}
```

Output:

```
Thread is running . Name :  MyFirstThread
```

Task2

```
Edit | Explain | Test | Document | Fix
class MyTask implements Runnable {  2 usages
    Edit | Explain | Test | Document | Fix
    public void run () {
        System .out . println (" Task is running . Thread : " +
                Thread . currentThread (). getName ());
    }
}
Edit | Explain | Test | Document | Fix
public class Main {
    Edit | Explain | Test | Document | Fix
    public static void main ( String [] args ) {
        MyTask task = new MyTask ();
        Thread t1 = new Thread (task , name: " RunnableThread ");
        t1. start ();
    }
}
```

Output

```
Task is running . Thread :  RunnableThread
```

Task3

```
Edit | Explain | Test | Document | Fix
class MyThread extends Thread {  2 usages
    Edit | Explain | Test | Document | Fix
    public void run () {
        for (int i = 0; i < 3; i++) {
            System .out . println (" Thread running : " + i);
            try {
                Thread . sleep ( millis: 1000) ; // Sleep for 1 second
            } catch ( InterruptedException e) {
                System . out . println (e);
            }
        }
    }
}
Edit | Explain | Test | Document | Fix
public class Main {
    Edit | Explain | Test | Document | Fix
    public static void main ( String [] args ) {
        MyThread thread = new MyThread ();
        thread . start ();
    }
}
```

Output:

```
Thread running : 0
Thread running : 1
Thread running : 2
```

Task 4

```java
class SharedResource {  4 usages
    private int count = 0;  2 usages
    public synchronized void increment () {  1 usage
        count ++;
    }
    public synchronized int getCount () {  1 usage
        return count ;
    }
}
Edit | Explain | Test | Document | Fix
class MyThread extends Thread {  2 usages
    SharedResource resource ;  2 usages
    MyThread ( SharedResource res) {  2 usages
        this . resource = res ;
    }
    Edit | Explain | Test | Document | Fix
    public void run () {
        for (int i = 0; i < 1000; i++) {
            resource . increment ();
        }
    }
}

Edit | Explain | Test | Document | Fix
public class Main {
    Edit | Explain | Test | Document | Fix
    public static void main ( String [] args ) throws InterruptedException {
        SharedResource resource = new SharedResource ();
        Thread t1 = new MyThread ( resource );
        Thread t2 = new MyThread ( resource );
        t1. start ();
        t2. start ();
        t1. join ();
        t2. join ();
        System .out . println (" Final Count : " + resource . getCount ());
    }
}
```

Output

```
Final Count : 2000
```

- Q1: What is the difference between using Thread and Runnable?
  - **Thread**: Extends Thread class. No multiple inheritance possible.
  - **Runnable**: Implements Runnable interface. Allows multiple inheritance and is more flexible.
- Q2: Why is synchronization important in multithreaded programs?
  - Synchronization prevents **data inconsistency** and **race conditions** when multiple threads access shared resources. It ensures **thread safety** by controlling access.
- Q3: What happens if a thread is interrupted while sleeping?
  - The thread throws an **InterruptedException** and immediately exits the sleep() state.
- Q4: How does the JVM manage thread scheduling?
  - JVM uses a **preemptive, priority-based scheduling** mechanism. Threads with higher priority are favored, but actual behavior depends on the OS.