

4. Create a class as shown

```
class Counter {  
    int count;  
    void inc() {  
        count = count+1;  
    }  
    int getCount() {  
        return count;  
    }  
}
```

Create three threads that will call the inc() method on the same Counter object. Start them all, and wait for all the threads to terminate. Assign different priority to threads. Justify your output.

```
class Counter implements Runnable {  
    int count;  
    Thread t;  
    private volatile boolean running = true;  
    public Counter(int x ,int p) {  
        t = new Thread(this);  
        count=x;  
        t.setPriority(p);  
  
    }  
    void inc(){  
        count=count+1;  
    }  
    int getCount(){  
        return count;  
    }  
    public void run() {  
  
        while (running) {  
            inc(); }  
        }  
    public void stop() { running = false; }  
    public void start() { t.start(); }
```

```

}
class main_class {
public static void main(String args[])
{

Thread.currentThread().setPriority(
Thread.MAX_PRIORITY);
Counter C1 = new Counter(5,(Thread.NORM_PRIORITY-2));
Counter C2 = new Counter(5,Thread.NORM_PRIORITY);
Counter C3 = new Counter(5,(Thread.NORM_PRIORITY+3));
C1.start();
C2.start();
C3.start();

try {

Thread.sleep(1000);

} catch (InterruptedException e) {

System.out.println("Main thread interrupted."); }

C1.stop();
C2.stop();
C3.stop();

try {

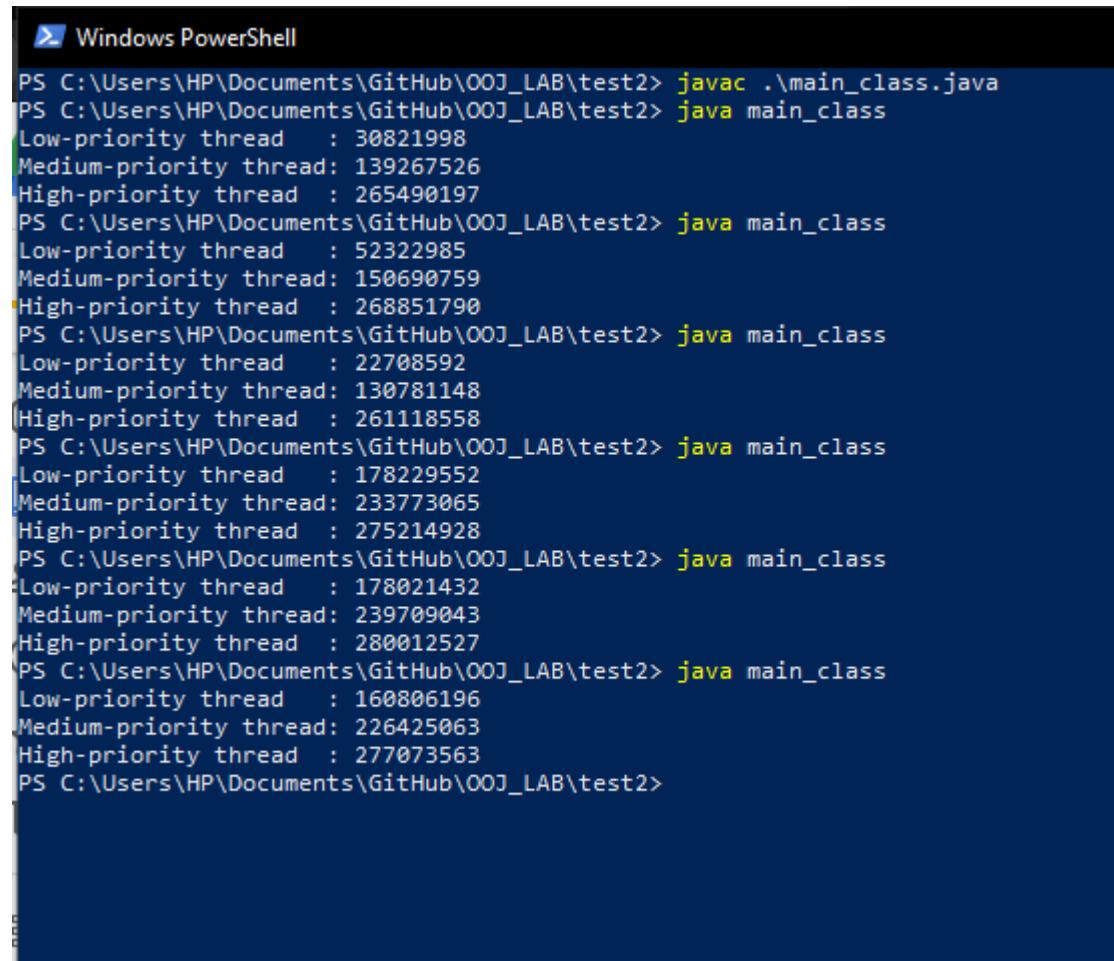
C1.t.join();
C2.t.join();
C3.t.join();
} catch (InterruptedException e) {

System.out.println("InterruptedExceptioncaught"); }
System.out.println("Low-priority thread   : "
+C1.getCount());
System.out.println("Medium-priority thread: "
+C2.getCount());

```

```
System.out.println("High-priority thread  : "  
+C3.getCount());  
}  
}
```

OUTPUT:

A screenshot of a Windows PowerShell terminal window. The title bar says "Windows PowerShell". The terminal shows a series of commands and their outputs. The commands are: `javac .\main_class.java` and `java main_class`. The outputs show thread priorities: Low-priority thread, Medium-priority thread, and High-priority thread, each followed by a numerical value. The values change with each execution of the `java main_class` command. The terminal background is dark blue, and the text is white. The commands are highlighted in yellow.

```
PS C:\Users\HP\Documents\GitHub\OOJ_LAB\test2> javac .\main_class.java  
PS C:\Users\HP\Documents\GitHub\OOJ_LAB\test2> java main_class  
Low-priority thread  : 30821998  
Medium-priority thread: 139267526  
High-priority thread : 265490197  
PS C:\Users\HP\Documents\GitHub\OOJ_LAB\test2> java main_class  
Low-priority thread  : 52322985  
Medium-priority thread: 150690759  
High-priority thread : 268851790  
PS C:\Users\HP\Documents\GitHub\OOJ_LAB\test2> java main_class  
Low-priority thread  : 22708592  
Medium-priority thread: 130781148  
High-priority thread : 261118558  
PS C:\Users\HP\Documents\GitHub\OOJ_LAB\test2> java main_class  
Low-priority thread  : 178229552  
Medium-priority thread: 233773065  
High-priority thread : 275214928  
PS C:\Users\HP\Documents\GitHub\OOJ_LAB\test2> java main_class  
Low-priority thread  : 178021432  
Medium-priority thread: 239709043  
High-priority thread : 280012527  
PS C:\Users\HP\Documents\GitHub\OOJ_LAB\test2> java main_class  
Low-priority thread  : 160806196  
Medium-priority thread: 226425063  
High-priority thread : 277073563  
PS C:\Users\HP\Documents\GitHub\OOJ_LAB\test2>
```
