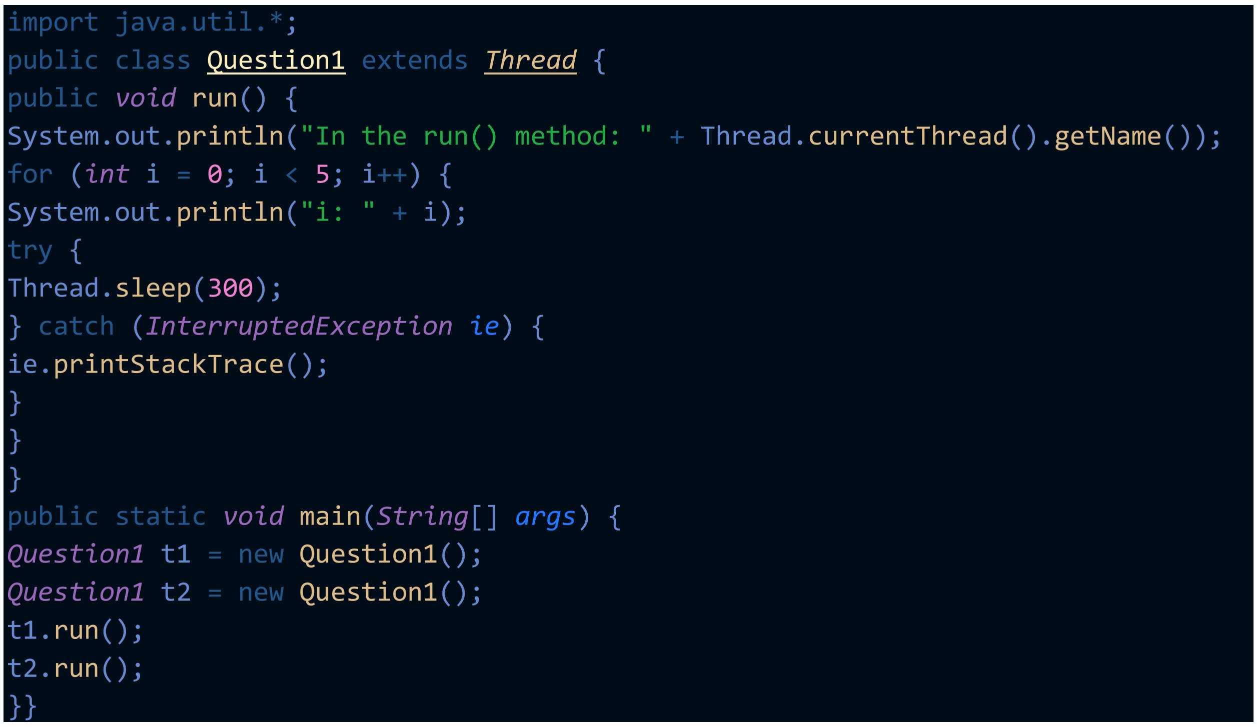
Name : Nikam Khushi Sahebrao
Branch : IT
Subject : Java Programming Lab
PRN : 22620004
Batch : S1

Assignment NO :07

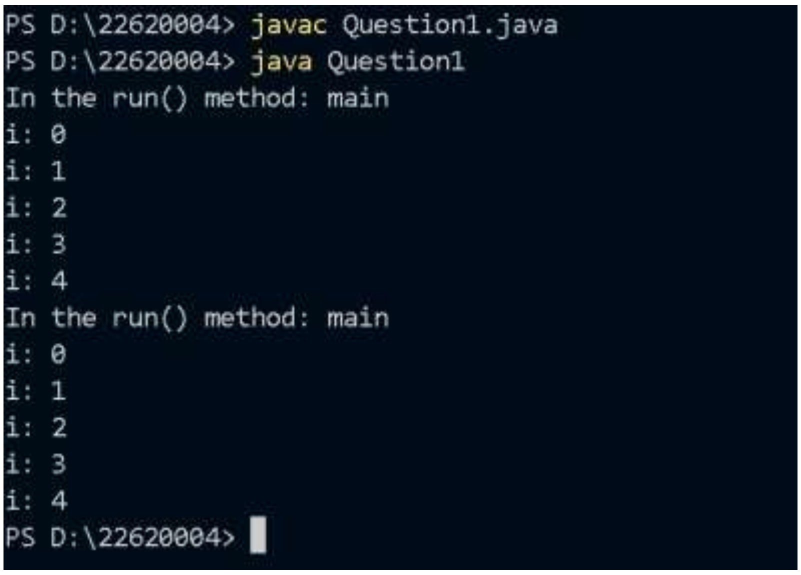
1. Can we call the run() method instead of start()?

Ans : When we call the start() method on a thread it causes the thread to begin execution and run() method of a thread is called by the Java Virtual Machine(JVM). If we call directly the run() method, it will be treated as a normal overridden method of a thread class (or runnable interface) and it will be executed within the context of the current thread, not in a new thread.



```
import java.util.*;
public class Question1 extends Thread {
public void run() {
System.out.println("In the run() method: " + Thread.currentThread().getName());
for (int i = 0; i < 5; i++) {
System.out.println("i: " + i);
try {
Thread.sleep(300);
} catch (InterruptedException ie) {
ie.printStackTrace();
}
}
}
public static void main(String[] args) {
Question1 t1 = new Question1();
Question1 t2 = new Question1();
t1.run();
t2.run();
}}
```

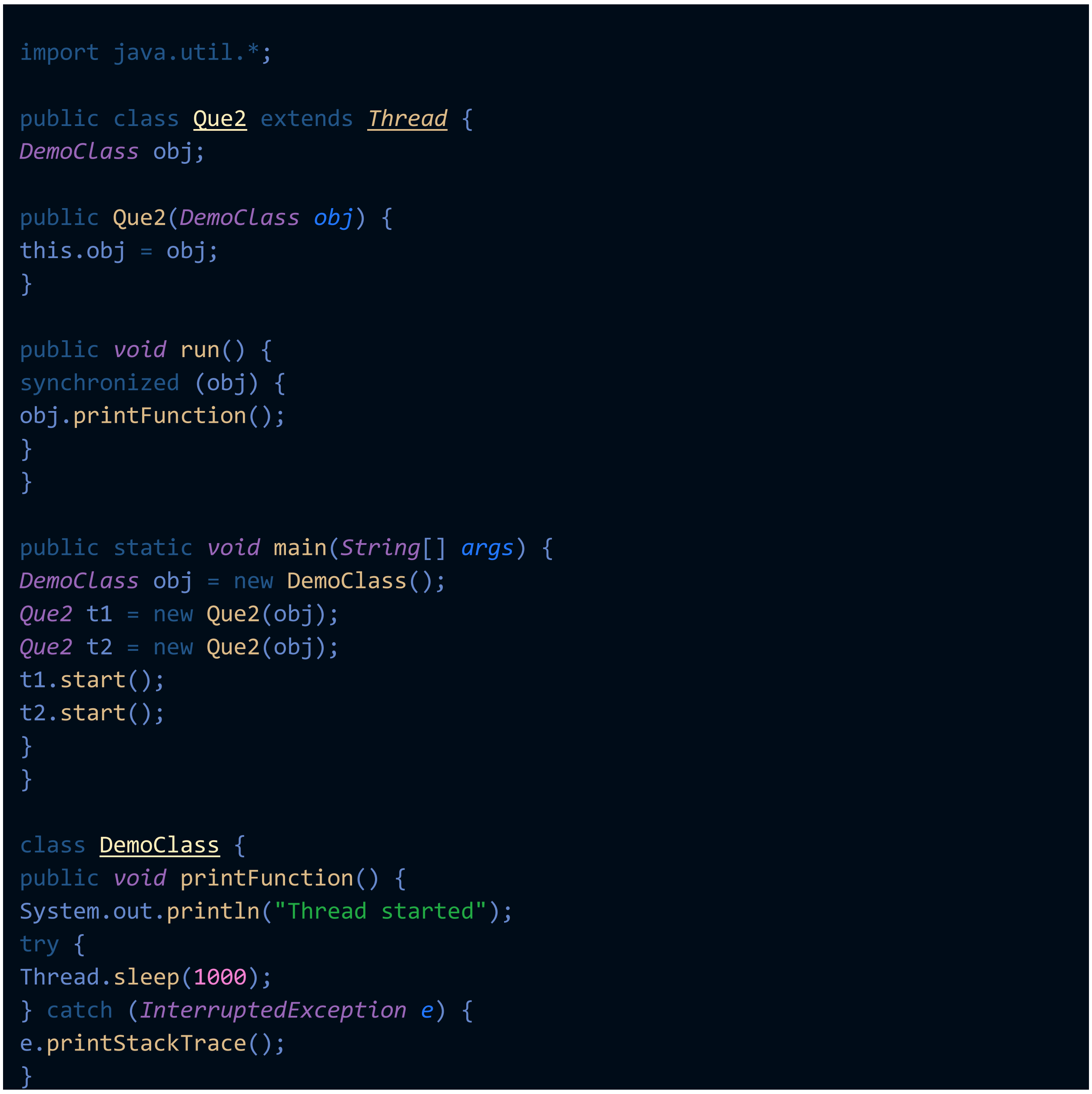
Output :



```
PS D:\22620004> javac Question1.java
PS D:\22620004> java Question1
In the run() method: main
i: 0
i: 1
i: 2
i: 3
i: 4
In the run() method: main
i: 0
i: 1
i: 2
i: 3
i: 4
PS D:\22620004>
```

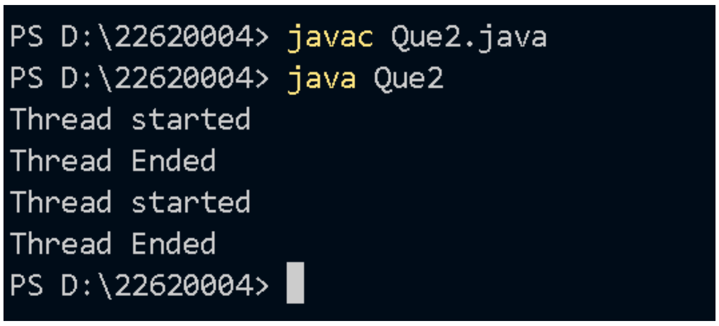
2. Explain the use of word Synchronized?

Ans : The synchronized keyword in java helps us reduce the concurrency when we share the same resource between multiple threads or processes. The fundamental features of the synchronized keyword are: - The synchronized keyword locks the resources to a thread so that no other thread can access it at a time. - The synchronized keyword prevents the program statement from getting reordered. - The synchronized keyword ensures the locking and unlocking of threads before and after getting inside the synchronized block. The thread reads the data from the main memory after receiving the lock. After completing the read operation, it flushes the write operation to release the lock.



```
import java.util.*;

public class Que2 extends Thread {
DemoClass obj;

public Que2(DemoClass obj) {
this.obj = obj;
}

public void run() {
synchronized (obj) {
obj.printFunction();
}
}

public static void main(String[] args) {
DemoClass obj = new DemoClass();
Que2 t1 = new Que2(obj);
Que2 t2 = new Que2(obj);
t1.start();
t2.start();
}
}

class DemoClass {
public void printFunction() {
System.out.println("Thread started");
try {
Thread.sleep(1000);
} catch (InterruptedException e) {
e.printStackTrace();
}
System.out.println("Thread Ended");
}}
```

Output :

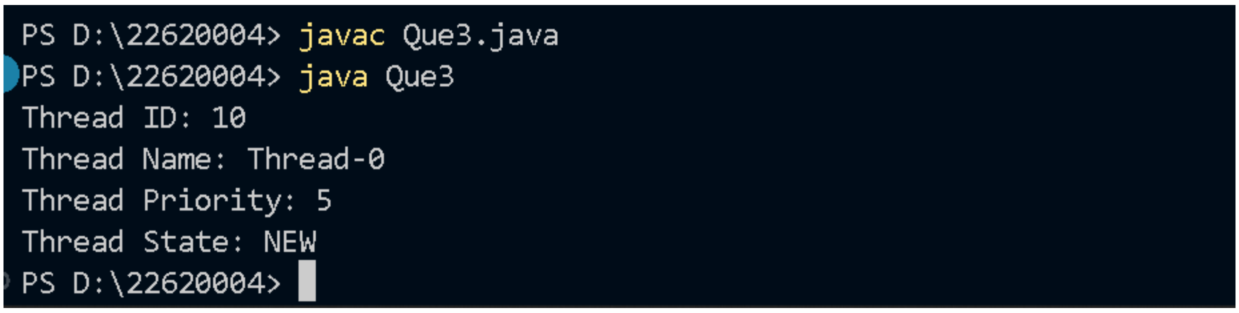


```
PS D:\22620004> javac Que2.java
PS D:\22620004> java Que2
Thread started
Thread Ended
Thread started
Thread Ended
PS D:\22620004>
```

3. Write a program to display thread information?

```
public class Que3 extends Thread {
public void run() {
}
public static void main(String[] args) {
Que3 t1 = new Que3();
System.out.println("Thread ID: " + t1.getId());
System.out.println("Thread Name: " + t1.getName());
System.out.println("Thread Priority: " + t1.getPriority());
System.out.println("Thread State: " + t1.getState());
}
}
```

Output :



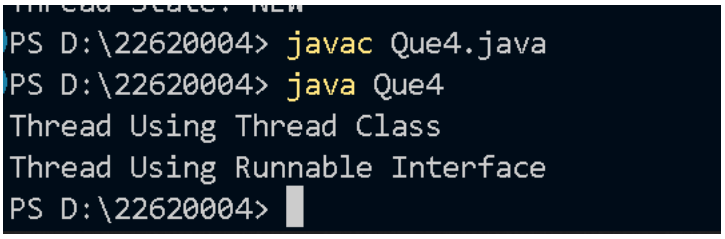
```
PS D:\22620004> javac Que3.java
PS D:\22620004> java Que3
Thread ID: 10
Thread Name: Thread-0
Thread Priority: 5
Thread State: NEW
PS D:\22620004>
```

4 .Create a thread using Thread class and Runnable class.

```
public class Que4 extends Thread {
```

```
public void run() {
System.out.println("Thread Using Thread Class");
}
public static void main(String[] args) {
Que4 t1 = new Que4();
ThreadInterface t2 = new ThreadInterface();
t1.start();
Thread thread = new Thread(t2);
thread.start();
}
}
class ThreadInterface implements Runnable {
@Override
public void run() {
System.out.println("Thread Using Runnable Interface");
}
}
```

Output :



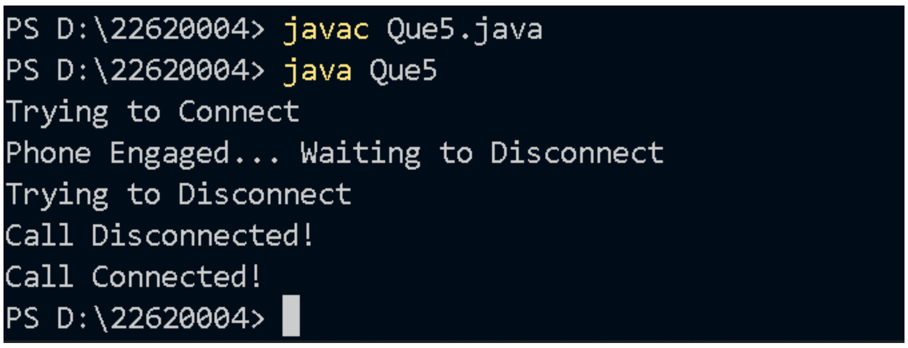
```
PS D:\22620004> javac Que4.java
PS D:\22620004> java Que4
Thread Using Thread Class
Thread Using Runnable Interface
PS D:\22620004>
```

5. Write a program for thread communication and synchronization.

```
class Call {
boolean CallConnected;
public Call() {
CallConnected = true;
}
synchronized void Connect() {
System.out.println("Trying to Connect");
if (CallConnected) {
System.out.println("Phone Engaged... Waiting to Disconnect");
try {
wait();
} catch (Exception e) {
}
}
System.out.println("Call Connected!");
}
synchronized void Disonnect() {
```

```java
System.out.println("Trying to Disconnect");
System.out.println("Call Disconnected! ");
notify();
}
}
public class Que5 {
public static void main(String args[]) {
final Call c1 = new Call();
new Thread() {
public void run() {
c1.Connect();
}
}.start();
new Thread() {
public void run() {
c1.Disonnect();
}
}.start();
}
}
```

Output :



```
PS D:\22620004> javac Que5.java
PS D:\22620004> java Que5
Trying to Connect
Phone Engaged... Waiting to Disconnect
Trying to Disconnect
Call Disconnected!
Call Connected!
PS D:\22620004>
```