MEHER SHRISHTI NIGAM

20BRS1193

JAVA LAB 8

Q1) Assume that a course CSE1007 is offered in two slots D1 and D2 and each slot has only one class number. Write a simple java program using threads to track the student registration of each class and print the total number of students registered for D1 slot and D2 slot. Implement the solution by extending the thread class initially followed by implementation using runnable interface as well.

CODE:

```
public class Registration_Thread_20BRS1193 extends Thread{
    String reg_no;
    String slot;
    static int D1_Students = 0;
    static int D2_Students = 0;

    public void run(String reg_no, int slot_preference){
        this.reg_no = reg_no;
        if(slot_preference == 1)
            this.Slot_D1();
        else if(slot_preference == 2)
            this.Slot_D2();
        this.print();
    }

    public void Slot_D1(){
        this.slot = "D1";
        System.out.println("Slot chosen as D1");
        D1_Students++;
    }
    public void Slot_D2(){
        this.slot = "D2";
        System.out.println("Slot chosen as D2");
        D2_Students++;
    }

    public void print(){
        System.out.println("Reg No:" + this.reg_no);
        System.out.println("Slot:" + this.slot);
        System.out.println("Number of students D1 slot:" +
D1_Students);
        System.out.println("Number of students D2 slot:" +
D2_Students);
        System.out.println();
```

```
    }

    public static void main (String args[]){
        Registration_Thread_20BRS1193 stu1 = new
Registration_Thread_20BRS1193();
        Registration_Thread_20BRS1193 stu2 = new
Registration_Thread_20BRS1193();
        Registration_Thread_20BRS1193 stu3 = new
Registration_Thread_20BRS1193();
        stu2.run("101",1);
        stu1.run("102",2);
        stu3.run("103",2);
    }
}
```

OUTPUT:

```
PS C:\Users\meher\Documents\5th sem courses\Java Programming\lab\LAB8\Q1> javac Registration_Thread_20BRS1193.java
PS C:\Users\meher\Documents\5th sem courses\Java Programming\lab\LAB8\Q1> java Registration_Thread_20BRS1193
Slot chosen as D1
Reg No:101
Slot:D1
Number of students D1 slot:1
Number of students D2 slot:0

Slot chosen as D2
Reg No:102
Slot:D2
Number of students D1 slot:1
Number of students D2 slot:1

Slot chosen as D2
Reg No:103
Slot:D2
Number of students D1 slot:1
Number of students D2 slot:2
```

CODE:

```
import java.util.Scanner;
public class Registration_Runnable_20BRS1193 implements Runnable {
    String reg_no;
    String slot;
    static int D1_Students = 0;
    static int D2_Students = 0;

    public void run(){
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter register no: ");
        this.reg_no = sc.nextLine();
        System.out.println("Enter slot preference: ");
        int slot_preference = sc.nextInt();
        if(slot_preference == 1)
```

```
            this.Slot_D1();
        else if(slot_preference == 2)
            this.Slot_D2();
        this.print();
    }

    public void Slot_D1(){
        this.slot = "D1";
        System.out.println("Slot chosen as D1");
        D1_Students++;
    }
    public void Slot_D2(){
        this.slot = "D2";
        System.out.println("Slot chosen as D2");
        D2_Students++;
    }

    public void print(){
        System.out.println("Reg No:" + this.reg_no);
        System.out.println("Slot:" + this.slot);
        System.out.println("Number of students D1 slot:" +
D1_Students);
        System.out.println("Number of students D2 slot:" +
D2_Students);
        System.out.println();
    }

    public static void main (String args[]){
        Registration_Runnable_20BRS1193 stu1 = new
Registration_Runnable_20BRS1193();
        stu1.run();
        Registration_Runnable_20BRS1193 stu2 = new
Registration_Runnable_20BRS1193();
        stu2.run();
        Registration_Runnable_20BRS1193 stu3 = new
Registration_Runnable_20BRS1193();
        stu3.run();
    }
}
```

OUTPUT:

```
PS C:\Users\meher\Documents\5th sem courses\Java Programming\lab\LAB8\Q1>
                                                    javac Registration_Runnable_20BRS1193.java
PS C:\Users\meher\Documents\5th sem courses\Java Programming\lab\LAB8\Q1> java Registration_Runnable_20BRS1193
Enter register no:
101
Enter slot preference:
1
Slot chosen as D1
Reg No:101
Slot:D1
Number of students D1 slot:1
Number of students D2 slot:0

Enter register no:
102
Enter slot preference:
2
Slot chosen as D2
Reg No:102
Slot:D2
Number of students D1 slot:1
Number of students D2 slot:1

Enter register no:
103
Enter slot preference:
2
Slot chosen as D2
Reg No:103
Slot:D2
Number of students D1 slot:1
Number of students D2 slot:2

PS C:\Users\meher\Documents\5th sem courses\Java Programming\lab\LAB8\Q1>
```

Q2) Write a Java program to print three multiplication tables (2, 3 and 5) using threads. (i.e assign each thread for a table).

CODE:

```java
class Q2_20BRS1193 {
    public static void main(String[] args) {
        Table_20BRS1193 ob = new Table_20BRS1193();
        thread1_20BRS1193 t1 = new thread1_20BRS1193(ob);
        thread2_20BRS1193 t2 = new thread2_20BRS1193(ob);
        thread3_20BRS1193 t3 = new thread3_20BRS1193(ob);
        t1.start();
        t2.start();
        t3.start();
    }
}

class Table_20BRS1193 {
    synchronized void table(int n) {
        for (int i = 1; i <= 10; i++) {
            System.out.println(n + " * " + i + " = " + n * i);
        }
    }
}
```

```java
class thread1_20BRS1193 extends Thread {
    Table_20BRS1193 m;

    thread1_20BRS1193(Table_20BRS1193 m) {
        this.m = m;
    }

    public void run() {
        m.table(2);
    }
}

class thread2_20BRS1193 extends Thread {
    Table_20BRS1193 m;

    thread2_20BRS1193(Table_20BRS1193 m) {
        this.m = m;
    }

    public void run() {
        m.table(3);
    }
}

class thread3_20BRS1193 extends Thread {
    Table_20BRS1193 m;

    thread3_20BRS1193(Table_20BRS1193 m) {
        this.m = m;
    }

    public void run() {
        m.table(5);
    }
}
```

OUTPUT:

```
PS C:\Users\meher\Documents\5th sem courses\Java Programming\lab\LAB8\Q2> javac Q2_20BRS1193.java
PS C:\Users\meher\Documents\5th sem courses\Java Programming\lab\LAB8\Q2> java Q2_20BRS1193
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
2 * 6 = 12
2 * 7 = 14
2 * 8 = 16
2 * 9 = 18
2 * 10 = 20
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50
3 * 1 = 3
3 * 2 = 6
3 * 3 = 9
3 * 4 = 12
3 * 5 = 15
3 * 6 = 18
3 * 7 = 21
3 * 8 = 24
3 * 9 = 27
3 * 10 = 30
PS C:\Users\meher\Documents\5th sem courses\Java Programming\lab\LAB8\Q2>
```

Q3) Consider the case of a producer consumer problem. Demonstrate the behaviour when there is no Inter Thread Communication (ITC) and subsequently show how ITC promotes effective communication between two threads.

Below code was provided as a part of module 3 material –

NO ITC ->

CODE:

```
class ConsumeThread_20BRS1193 extends Thread {

    Produce_Consume_20BRS1193 obj;

    ConsumeThread_20BRS1193(Produce_Consume_20BRS1193 obj){
        this.obj = obj;
    }

    public void run() {
        for(int k = 0; k <= 5; k++)
            obj.Consume();
```

```
    }
}
```

```
class NoCommunication_20BRS1193 {
    public static void main(String[] args) {
        Produce_Consume_20BRS1193 obj = new
Produce_Consume_20BRS1193();
        ProduceThread_20BRS1193 P = new ProduceThread_20BRS1193(obj);
        P.start();
        ConsumeThread_20BRS1193 C = new ConsumeThread_20BRS1193(obj);
        C.start();
    }
}
```

```
class Produce_Consume_20BRS1193{
    int i;
    synchronized void Produce(int i) {
        this.i = i; System.out.println("Data Delivered: " +i);
    }
    synchronized int Consume() {
        System.out.println("Data Received: " + i);
        return i;
    }
}


class ProduceThread_20BRS1193 extends Thread{
    Produce_Consume_20BRS1193 obj;
    ProduceThread_20BRS1193(Produce_Consume_20BRS1193 obj){
        this.obj = obj;
    }
    public void run() {
        for(int k = 0; k <= 5; k++)
        obj.Produce(k);
    }
}
```

OUTPUT:

```
PS C:\Users\meher\Documents\5th sem courses\Java Programming\lab\LAB8\Q3\NoITC> java NoCommunication_20BRS1193
Data Delivered: 0
Data Delivered: 1
Data Delivered: 2
Data Delivered: 3
Data Delivered: 4
Data Delivered: 5
Data Received: 5
Data Received: 5
Data Received: 5
Data Received: 5
Data Received: 5
Data Received: 5
PS C:\Users\meher\Documents\5th sem courses\Java Programming\lab\LAB8\Q3\NoITC>
```

We observe that all the data received are 5's, i.e., the last delivered data.

USING ITC ->

CODE:

```
class Produce_Consume_20BRS1193{
    int i;
    boolean flag = false;
    synchronized void Produce(int i) {
        if(flag)
            try { // Wait till a notification is received from
Thread2.
                wait();
            }
            catch(InterruptedException ie) {
                System.out.println(ie);
            }
        this.i = i;
        System.out.println("Data Delivered: " +i);
        flag = true; // When data production is over.
        notify(); // Notification to Thread2, when data Produced.
    }
    synchronized int Consume() {
    if(!flag)
        try { // Wait till a notification is received from Thread1.
            wait();
        }
        catch(InterruptedException ie){
            System.out.println(ie);
        }
        System.out.println("Data Received: " + i);
        flag = false; // When data is received.
        notify(); //Notification to Thread1, when data Consumed.
        return i;
    }
}
```

```
class ConsumeThread_20BRS1193 extends Thread {
    Produce_Consume_20BRS1193 obj;
    ConsumeThread_20BRS1193(Produce_Consume_20BRS1193 obj){ this.obj =
obj; }
    public void run() { for(int k = 0; k <= 5; k++) obj.Consume(); }
}
```

```
class ProduceThread_20BRS1193 extends Thread {
    Produce_Consume_20BRS1193 obj;
    ProduceThread_20BRS1193(Produce_Consume_20BRS1193 obj){ this.obj =
obj; }
    public void run() { for(int k = 0; k <= 5; k++) obj.Produce(k); }
    }
```

```
class Communication_20BRS1193 {
    public static void main(String[] args) {
    Produce_Consume_20BRS1193 obj = new Produce_Consume_20BRS1193();
    ProduceThread_20BRS1193 P = new ProduceThread_20BRS1193(obj);
    P.start();
    ConsumeThread_20BRS1193 C = new ConsumeThread_20BRS1193(obj);
    C.start();
    }
}
```

OUTPUT:

```
PS C:\Users\meher\Documents\5th sem courses\Java Programming\lab\LAB8\Q3\ITC> java Communication_20BRS1193
Data Delivered: 0
Data Received: 0
Data Delivered: 1
Data Received: 1
Data Delivered: 2
Data Received: 2
Data Delivered: 3
Data Received: 3
Data Delivered: 4
Data Received: 4
Data Delivered: 5
Data Received: 5
PS C:\Users\meher\Documents\5th sem courses\Java Programming\lab\LAB8\Q3\ITC>
```

Q4) As part of a technical week celebration three events namely workshop, poster presentation and hackathon are organized. There are three registration desks put up for these events. The registration cost for workshop, poster presentation and hackathon is Rs.100, Rs.200 and Rs.300 respectively. Using threads track the registration count and registration cost of each event.

CODE:

```
public class TechnicalWeek_20BRS1193 {
    final static int WORKSHOP_COST = 100;
    final static int POSTER_COST = 200;
    final static int HACKATHON_COST = 300;
    static int workshopCount = 0;
    static int posterCount = 0;
    static int hackathonCount = 0;
    int regNo;
    int totalCost;

    public void run(Boolean workshop, Boolean poster, Boolean
hackathon){
        if(workshop)
            this.Workshop();
        if(poster)
            this.Poster();
        if(hackathon)
            this.Hackathon();
        this.print();
    }

    void Workshop(){
        this.totalCost += WORKSHOP_COST;
        workshopCount++;
    }
    void Poster(){
        this.totalCost += POSTER_COST;
        posterCount++;
    }
    void Hackathon(){
        this.totalCost += HACKATHON_COST;
        hackathonCount++;
    }

    public void print(){
        System.out.println("Total Cost to be payed by current
student:" + this.totalCost);
```

```
        System.out.println("Number of students for workshop:" +
workshopCount);
        System.out.println("Number of students for poster
presentation:" + posterCount);
        System.out.println("Number of students for hackathon:" +
hackathonCount);
        int totalFunds = (workshopCount * WORKSHOP_COST) +
(posterCount * POSTER_COST) + (hackathonCount * HACKATHON_COST);
        System.out.println("Total funds:" + totalFunds);
        System.out.println();
    }
    public static void main(String args[]){
        TechnicalWeek_20BRS1193 stu1 = new TechnicalWeek_20BRS1193();
        stu1.run(true, true, false);
        TechnicalWeek_20BRS1193 stu2 = new TechnicalWeek_20BRS1193();
        stu2.run(false, true, true);
        TechnicalWeek_20BRS1193 stu3 = new TechnicalWeek_20BRS1193();
        stu3.run(true, false, false);
        TechnicalWeek_20BRS1193 stu4 = new TechnicalWeek_20BRS1193();
        stu4.run(false, false, false);
        TechnicalWeek_20BRS1193 stu5 = new TechnicalWeek_20BRS1193();
        stu5.run(true, true, true);
    }
}
```

OUTPUT:

```
PS C:\Users\meher\Documents\5th sem courses\Java Programming\lab\LAB8\Q4> java TechnicalWeek_20BRS1193
Total Cost to pay by current student:300
Number of students for workshop:1
Number of students for poster presentation:1
Number of students for hackathon:0
Total funds:300

Total Cost to pay by current student:500
Number of students for workshop:1
Number of students for poster presentation:2
Number of students for hackathon:1
Total funds:800

Total Cost to pay by current student:100
Number of students for workshop:2
Number of students for poster presentation:2
Number of students for hackathon:1
Total funds:900

Total Cost to pay by current student:0
Number of students for workshop:2
Number of students for poster presentation:2
Number of students for hackathon:1
Total funds:900

Total Cost to pay by current student:600
Number of students for workshop:3
Number of students for poster presentation:3
Number of students for hackathon:2
Total funds:1500

PS C:\Users\meher\Documents\5th sem courses\Java Programming\lab\LAB8\Q4>
```

Q5) Assume that a courier delivery office has to deliver 10 parcels to two customers. Write a java program using threads to deliver the odd parcels to customer1 and even parcels to customer2.

CODE:

```
public class DeliveryOffice_20BRS1193 {

    int parcelNumber = 1;

    static int N;

    public void customer1()
    {
        synchronized (this)
        {
            while (parcelNumber < N) {

                while (parcelNumber % 2 == 0) {

                    try {
                        wait();
                    }
```

```
                    catch (
                        InterruptedException e) {
                        e.printStackTrace();
                    }
                }

                System.out.println("Parcel Number: " + parcelNumber +
" is received by customer 1.");
                parcelNumber++;
                notify();
            }
        }
    }

    public void customer2()
    {
        synchronized (this)
        {
            while (parcelNumber < N) {
                while (parcelNumber % 2 == 1) {
                    try {
                        wait();
                    }
                    catch (
                        InterruptedException e) {
                        e.printStackTrace();
                    }
                }

                System.out.println("Parcel Number: " + parcelNumber +
" is received by customer 2.");
                parcelNumber++;
                notify();
            }
        }
    }

    public static void main(String[] args)
    {
        N = 10;
        DeliveryOffice_20BRS1193 parcels = new
DeliveryOffice_20BRS1193();
```

```
        Thread t1 = new Thread(new Runnable() {
            public void run()
            {
                parcels.customer2();
            }
        });

        Thread t2 = new Thread(new Runnable() {
            public void run()
            {
                parcels.customer1();
            }
        });

        t1.start();
        t2.start();
    }
}
```

OUTPUT:

```
PS C:\Users\meher\Documents\5th sem courses\Java Programming\lab\LAB8\Q5> javac DeliveryOffice_20BRS1193.
java
PS C:\Users\meher\Documents\5th sem courses\Java Programming\lab\LAB8\Q5> java DeliveryOffice_20BRS1193
Parcel Number: 1 is received by customer 1.
Parcel Number: 2 is received by customer 2.
Parcel Number: 3 is received by customer 1.
Parcel Number: 4 is received by customer 2.
Parcel Number: 5 is received by customer 1.
Parcel Number: 6 is received by customer 2.
Parcel Number: 7 is received by customer 1.
Parcel Number: 8 is received by customer 2.
Parcel Number: 9 is received by customer 1.
Parcel Number: 10 is received by customer 2.
PS C:\Users\meher\Documents\5th sem courses\Java Programming\lab\LAB8\Q5>
```

Q6) Assume that a restaurant has received 20 orders. The first 5 orders have to be catered to table1, orders 6-10 for table 2, orders 11-15 for table 3 and orders 16-20 for table 4. Implement a solution for this using Java threads.

CODE:

```
class Communication_20BRS1193 {
    public static void main(String[] args) {
        Produce_Consume_20BRS1193 obj = new
Produce_Consume_20BRS1193();
        ProduceThread_20BRS1193 P = new ProduceThread_20BRS1193(obj);
        P.start();
```

```
        ConsumeThread_Table1_20BRS1193 C1 = new
ConsumeThread_Table1_20BRS1193(obj);
        C1.start();
        ConsumeThread_Table2_20BRS1193 C2 = new
ConsumeThread_Table2_20BRS1193(obj);
        C2.start();
        ConsumeThread_Table3_20BRS1193 C3 = new
ConsumeThread_Table3_20BRS1193(obj);
        C3.start();
        ConsumeThread_Table4_20BRS1193 C4 = new
ConsumeThread_Table4_20BRS1193(obj);
        C4.start();
    }
}
```

```
class ConsumeThread_Table1_20BRS1193 extends Thread {
    Produce_Consume_20BRS1193 obj;
    ConsumeThread_Table1_20BRS1193(Produce_Consume_20BRS1193 obj){
this.obj = obj; }
    public void run()
    {
        for(int k = 1; k <= 5; k++)
            obj.Consume(k, 1);
    }
}
```

```
class ConsumeThread_Table2_20BRS1193 extends Thread {
    Produce_Consume_20BRS1193 obj;
    ConsumeThread_Table2_20BRS1193(Produce_Consume_20BRS1193 obj){
this.obj = obj; }
    public void run() { for(int k = 6; k <= 10; k++) obj.Consume(k,
2); }
}
```

```
class ConsumeThread_Table3_20BRS1193 extends Thread {
    Produce_Consume_20BRS1193 obj;
    ConsumeThread_Table3_20BRS1193(Produce_Consume_20BRS1193 obj){
this.obj = obj; }
    public void run() { for(int k = 11; k <= 16; k++) obj.Consume(k,
3); }
}
```

```
class ConsumeThread_Table4_20BRS1193 extends Thread {
    Produce_Consume_20BRS1193 obj;
    ConsumeThread_Table4_20BRS1193(Produce_Consume_20BRS1193 obj){
this.obj = obj; }
    public void run() { for(int k = 15; k <= 20; k++) obj.Consume(k,
4); }
}
```

```
class Produce_Consume_20BRS1193{
    int i;
    boolean flag = false;
    synchronized void Produce(int i) {
        if(flag)
            try { // Wait till a notification is received from
Thread2.
                wait();
            }
            catch(InterruptedException ie) {
                System.out.println(ie);
            }
        this.i = i;
        System.out.println("Data Delivered: " + i);
        flag = true; // When data production is over.
        notify(); // Notification to Thread2, when data Produced.
    }
    synchronized int Consume(int x, int tableNo) {
    if(!flag)
        try { // Wait till a notification is received from Thread1.
            wait();
        }
        catch(InterruptedException ie){
            System.out.println(ie);
        }
        System.out.println("Data  "+ x + " Received by table no.: " +
tableNo);
        flag = false; // When data is received.
        notify(); //Notification to Thread1, when data Consumed.
        return i;
    }
}
```

```
class ProduceThread_20BRS1193 extends Thread {
    Produce_Consume_20BRS1193 obj;
    ProduceThread_20BRS1193(Produce_Consume_20BRS1193 obj){ this.obj =
obj; }
    public void run() { for(int k = 1; k <= 20; k++) obj.Produce(k); }
    }
```

OUTPUT:

```
PS C:\Users\meher\Documents\5th sem courses\Java Programming\lab\LAB8\Q6\hmm> javac Communication_20BRS1193.java
PS C:\Users\meher\Documents\5th sem courses\Java Programming\lab\LAB8\Q6\hmm> java Communication_20BRS1193
Data Delivered: 1
Data  15 Received by table no.: 4
Data Delivered: 2
Data  16 Received by table no.: 4
Data  11 Received by table no.: 3
Data  6 Received by table no.: 2
Data  1 Received by table no.: 1
Data Delivered: 3
Data  17 Received by table no.: 4
Data  12 Received by table no.: 3
Data  7 Received by table no.: 2
Data  2 Received by table no.: 1
Data Delivered: 4
Data  18 Received by table no.: 4
Data  13 Received by table no.: 3
Data  8 Received by table no.: 2
Data  3 Received by table no.: 1
Data Delivered: 5
Data  19 Received by table no.: 4
Data  14 Received by table no.: 3
Data  9 Received by table no.: 2
Data  4 Received by table no.: 1
Data Delivered: 6
Data  20 Received by table no.: 4
Data  15 Received by table no.: 3
Data  10 Received by table no.: 2
Data  5 Received by table no.: 1
Data Delivered: 7
Data  16 Received by table no.: 3
Data Delivered: 8
PS C:\Users\meher\Documents\5th sem courses\Java Programming\lab\LAB8\Q6\hmm>
```