Session - III

Part II

Lab No. 8: Multithreading

Lab Exercises

1. Create a class by extending Thread Class to print a multiplication table of a number supplied as parameter. Create another class Tables which will instantiate two objects of the above class to print multiplication table of 5 and 7.

Code:

```
class Tables extends Thread {

private int num;

private Thread t;

public Tables(){

this.num = 1;

System.out.print(String.format("Created a thread %d\n", this.num));

}

public Tables(int num){

this.num = num;

System.out.print(String.format("Created a thread %d\n", this.num));

}

public void printTables(){

System.out.print(String.format("Printing Tables of %d\n",this.num));

for(int i = 1; i <= 10; i++){

System.out.print(String.format("%dx%d = %d \n", this.num, i, this.num*i));

}
```

```
}
public void run(){
System.out.print(String.format("Running thread %d\n", this.num));
this.printTables();
}
public void start(){
System.out.print(String.format("Starting thread %d\n", this.num));
if(t == null){
t = new Thread(this, String.format("thread%d", this.num));
t.start();
}
}
}
class Tables25{
public static void main(String[] args){
Tables t1 = new Tables(5);
t1.start();try{ t1.join();
}
catch (InterruptedException e){
e.printStackTrace();
}
Tables t2 = new Tables(7);
t2.start();
}
}
```

Test Case:

```
Student@dblab-hp-28:~/190905513$ gedit Tables25.java
Student@dblab-hp-28:~/190905513$ javac Tables25.java
Student@dblab-hp-28:~/190905513$ java Tables25
Created a thread 5
Starting thread 5
Created a thread 7
Running thread 5
Starting thread 7
Printing Tables of 5
5x1 = 5
5x2 = 10
Running thread 7
5x3 = 15
Printing Tables of 7
5x4 = 20
7x1 = 7
5x5 = 25
7x2 = 14
5x6 = 30
7x3 = 21
5x7 = 35
7x4 = 28
5x8 = 40
7x5 = 35
5x9 = 45
7x6 = 42
5x10 = 50
7x7 = 49
7x8 = 56
7x9 = 63
7x10 = 70
Student@dblab-hp-28:~/190905513$
```

2. Write and execute a java program to create and initialize a matrix of integers. Create n threads(by implementing Runnable interface) where n is equal to the number of rows in the matrix. Each of these threads should compute a distinct row sum. The main thread computes the complete sum by looking into the partial sums given by the threads.

Code:

```
import java.util.Scanner;
class Matrix{
private int arr[][];
public Matrix(int n, int m){
arr= new int[n][m];
}
public int[] getRow(int i){
return arr[i];
}
public void input(){
Scanner sc = new Scanner(System.in);
System.out.println("Enter the matrix:");
for(int i=0; i<arr.length; i++){
for(int j=0; j<arr[i].length; j++)
arr[i][j] = sc.nextInt();
}
}
}
class RowSum implements Runnable{
private int arr[];
private int sum;
RowSum(int a[]){
arr = a;
sum = 0;
}
public int getRowSum(){
return sum;
}
public void run(){
System.out.println("Running a new thread");
for (int i:arr)
sum += i;
}
```

```
}
class MatrixDemo {
public static void main(String [] args){
Scanner sc = new Scanner(System.in);
System.out.print("Enter the dimensions of the matrix:");
int n = sc.nextInt();
int m = sc.nextInt();
Matrix matrix = new Matrix(n,m);
matrix.input();
Thread threads[] = new Thread[n];
RowSum rowsum[] = new RowSum[n];
for(int i=0; i<n; i++){
rowsum[i] = new RowSum(matrix.getRow(i));
threads[i]=new Thread(rowsum[i]); threads[i].start();
}
int sum=0;
try{
for(int i=0;i<n; i++){
threads[i].join();
sum += rowsum[i].getRowSum();
}}
catch (InterruptedException e){
e.printStackTrace();
}
System.out.println("Total sum = "+sum);
}}
```

Test Case:

```
Student@dblab-hp-28:~/190905513$ gedit MatrixDemo.java
Student@dblab-hp-28:~/190905513$ javac MatrixDemo.java
Student@dblab-hp-28:~/190905513$ java MatrixDemo
Enter the dimensions of the matrix:2 3
Enter the matrix:
1 2 3
4 5 6
Running a new thread
Running a new thread
Total sum = 21
Student@dblab-hp-28:~/190905513$
```

3. Write and execute a java program to implement a producer and consumer problem using Inter-thread communication.

Code:

```
class T1{
int n;
boolean valueSet = false;
synchronized int get(){
while(!valueSet){
try{
wait();
}catch(InterruptedException e){
System.out.println(e);
}}
System.out.println("Got: " + n);
valueSet = false; notify();
return n;
}
synchronized void put(int n){
while(valueSet){
try{
wait();
}catch(InterruptedException e)
{System.out.println(e);
}}
this.n = n;
valueSet = true;
System.out.println("Put: " +n);
notify();
}}
class Producer implements Runnable
{ T1 q;
Producer(T1 q){
this.q = q;
new Thread(this, "Producer").start();
}
public void run(){
int i = 0; while(i < 6){
q.put(i++);
```

```
}}}
class Consumer implements
Runnable { T1 q;
Consumer(T1 q){
this.q = q;
new Thread(this, "Consumer").start();
}
public void run()
{ while(true)
{
q.get();
}}}
class PCF{ public static void main(String[] args){
T1 q = new T1();
new Producer(q);
new Consumer(q);
} }
```

Test Case:

```
Student@dblab-hp-28:~/190905513$ gedit PCF.java
Student@dblab-hp-28:~/190905513$ javac PCF.java
Student@dblab-hp-28:~/190905513$ java PCF
Put: 0
Got: 0
Put: 1
Got: 1
Put: 2
Got: 2
Put: 3
Got: 3
Put: 4
Got: 4
Put: 5
Got: 5
```