Lab No.8 1) 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 < 11; i++)
{
System.out.print(String.format("%d*%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 ThreadTables
{
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();

        }
}
```

2) 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 MatrixTest { 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].j

                oin();

                                sum += rowsum[i].getRowSum();

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

                System.out.println("Total sum = "+sum);
        }
}
```

3) class
Q
{

```
        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
{ Q q;
        Producer(Q q)
        {
                this.q = q;
                new Thread(this, "Producer").start();
        }

        public void run()
        { int i = 0; while(i < 10)
                {
                        q.put(i++);
                }
        }
```

```
} class Consumer implements

Runnable { Q q;

        Consumer(Q q)
        {
                this.q = q;
                new Thread(this, "Consumer").start();
        }

        public void run()
        { while(true)
                {
                        q.get();
                }
        }
}

class PCFixed{ public static void main(String[]
        args)
        {
                Q q = new Q();
                new Producer(q);
                new Consumer(q);
        } }
```