Week 6

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.

```
class Multiply extends Thread{
	public Thread t;
	public int n;

	Multiply(int num){
		System.out.println("Thread created for " + num);
		n = num;
	}

	void printTable(){
		System.out.println("Table for " + n);
		for(int i = 1; i <= 10; i++){
			System.out.println(n + " * " + i + " = " + n*i);
		}
	}
	public void run(){
		printTable();
	}
	public void start(){
		// run();
		System.out.print(String.format("Starting thread %d\n", n));
		if(t == null){
			t = new Thread(this, String.format("thread%d", n));
			t.start();
		}
	}
}

class Tables{
	public static void main(String []args){
		Multiply t1 = new Multiply(5);
		t1.start();
		Multiply t2 = new Multiply(7);
		t2.start();
		try {
			t1.join();
		}
		catch(InterruptedException e){
			e.printStackTrace();
		}

	}
```

}

```
student@lplab-Lenovo-Produ
Thread created for 5
Starting thread 5
Thread created for 7
Table for 5
Starting thread 7
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
Table for 7
5 * 7 = 35
7 * 1 = 7
7 * 2 = 14
7 * 3 = 21
7 * 4 = 28
7 * 5 = 35
7 * 6 = 42
5 * 8 = 40
7 * 7 = 49
5 * 9 = 45
7 * 8 = 56
5 * 10 = 50
7 * 9 = 63
7 * 10 = 70
```

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.

```
import java.util.Scanner;

class Matrix{
	int a[][];
	Matrix(int r, int c){
		a = new int[r][c];
	}
```

```
	int[] getRow(int r){
		int arr[] = new int[a[r].length];
		for(int i = 0; i < a[r].length; i++)
			arr[i] = a[r][i];
		return arr;
	}
	void input(int r, int c){
		System.out.println("Enter elements");
		Scanner sc = new Scanner(System.in);
		for(int i = 0; i < r; i++){
			for(int j = 0; j < c; j++){
				a[i][j] = sc.nextInt();
			}
		}
	}
}

class RowSum implements Runnable{
	int a[];
	int sum;

	RowSum(int arr[]){
		a = arr;
		sum = 0;
	}

	public int getSum(){
		return sum;
	}

	public void run(){
		for(int i = 0; i < a.length; i++){
			sum += a[i];
		}
	}
}

class SumTest{
	public static void main(String []args){
		Scanner sc = new Scanner(System.in);
		int r, c;
		System.out.println("Enter rows and columns");
		r = sc.nextInt();
		c = sc.nextInt();
		Matrix mat = new Matrix(r, c);
		mat.input(r, c);

		Thread threads[] = new Thread[r];
		RowSum rowsum[] = new RowSum[r];
		for(int i=0; i<r; i++){
			rowsum[i] = new RowSum(mat.getRow(i));
			threads[i] = new Thread(rowsum[i]);
```

```
                threads[i].start();
            }
            int sum = 0;

            try{
                for(int i=0; i<r; i++){
                    threads[i].join();
                    sum += rowsum[i].getSum();
                }
            }
            catch(InterruptedException e){
                e.printStackTrace();
            }
            System.out.println("Sum = " + sum);
        }
}
```

```
student@lplab-Lenovo-Product:~/Desktop/190905104_OOP/lab6$ java SumTest
Enter rows and columns
2
2
Enter elements
1
2
3
4
Sum = 10
```

3)

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

```
class Q
{
        int n;
        boolean value = false;
        synchronized int get(){
            while(!value){
                try{
                    wait();
                }catch(InterruptedException e){
                    System.out.println(e);
                }
            }
            System.out.println("Consumer got: " + n);
            value = false;
            notify();
            return n;
        }

        synchronized void put(int n){
```

```
		while(value){
			try{
				wait();
			}catch(InterruptedException e){
				System.out.println(e);
			}
		}
		this.n = n;
		value = true;
		System.out.println("Producer produced: " + n);
		notify();
	}
}

class Producer implements Runnable{
	Q q;
	Producer(Q q){
		this.q = q;
		System.out.println("Producer created");
		new Thread(this, "Producer").start();
	}

	public void run(){
		int i = 0;
		while(i < 5){
			q.put(i++);
		}
	}
}


class Consumer implements Runnable{
	Q q;
	Consumer(Q q){
		this.q = q;
		System.out.println("Consumer created");
		new Thread(this, "Consumer").start();
	}
	public void run(){
		while(true){
			q.get();
		}
	}
}


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

}

```
student@lplab-Lenovo-Product:~/Desktop/190905104_OOP/lab6$ java PCTest
Producer created
Producer produced: 0
Consumer created
Consumer got: 0
Producer produced: 1
Consumer got: 1
Producer produced: 2
Consumer got: 2
Producer produced: 3
Consumer got: 3
Producer produced: 4
Consumer got: 4
```