**PRACTICAL NO. 08**

**Name:** Rishabh Jain

**Roll No:** 54

**Batch:** A3

**Subject:** OOPS

**AIM :** Demonstrate the use of multithreading. Consider a website publishes live cricket score. The server thread can change the contents of the website whereas all the client threads read the score. Write the code to demonstrate all the functionalities**.**

**Input:**

**Server.java**

package com.mycompany.threaddemo;

import static java.lang.Thread.sleep; import java.util.ArrayList;

import java.util.Scanner; public class Server implements

Runnable{

Scanner sc = new Scanner(System.in); int score = 0; int choice; int size;

ArrayList<Integer> runs = new ArrayList<>() ; public Server(ArrayList<Integer> runs,int size) { this.runs = runs; this.score = runs.get(0); this.size = size;

}

@Override public void run() {

// runs.add(2);

// runs.add(6);

// runs.add(4);

// runs.add(0);

// runs.add(0); for(int j = 0;j < 6;j++)

{

synchronized (runs)

{

runs.remove(0);

System.out.println("Enter runs Scored : "); choice = sc.nextInt();

System.out.println("Server -> Runs scored in " + (j

+ 1) + " ball is : " + choice); runs.add(choice);

// System.out.println("Total runs in the over are : \n" + total); runs.notifyAll();

} try{

Thread.sleep(2000);

} catch(InterruptedException e) { throw new RuntimeException(e);

}

}

}

}

**Client.java**

package com.mycompany.threaddemo; import java.util.ArrayList; class Client implements Runnable { private final ArrayList<Integer> runs; private final int size;

public Client(ArrayList<Integer> runs, int size)

{ this.runs = runs; this.size = size;

}

@Override public void run() { synchronized (runs) { while (true) { try {

runs.wait();

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

}

int score = runs.get(runs.size() - 1);

System.out.println(Thread.currentThread().getName() + " read -> Scored : " + score);

}

}

}

}

**ThreadDemo.java**

package com.mycompany.threaddemo; import java.util.\*;

public class ThreadDemo{

public static void main(String[] args) throws InterruptedException {

// double runs = 0;

ArrayList<Integer> runs = new

ArrayList<>(); runs.add(1); int

size = 6;

// Client c = new Client(runs,size);

Thread client\_1 = new Thread(new Client(runs,size),"Client 1");

Thread client\_2 = new Thread(new Client(runs,size),"Client 2");

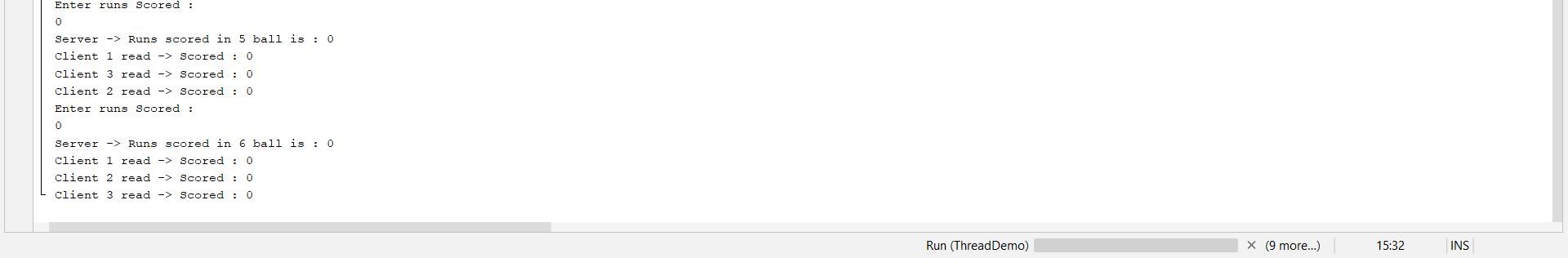
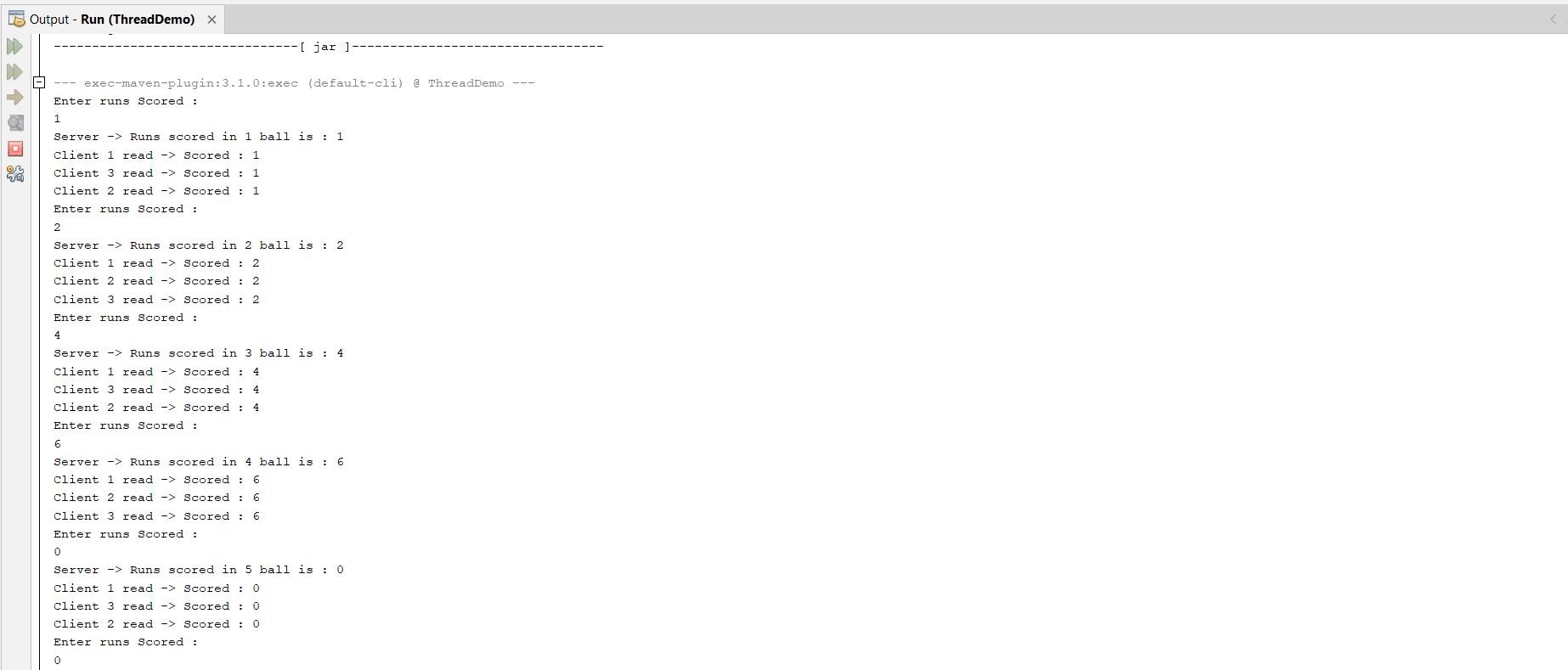
Thread client\_3 = new Thread(new Client(runs,size),"Client 3");

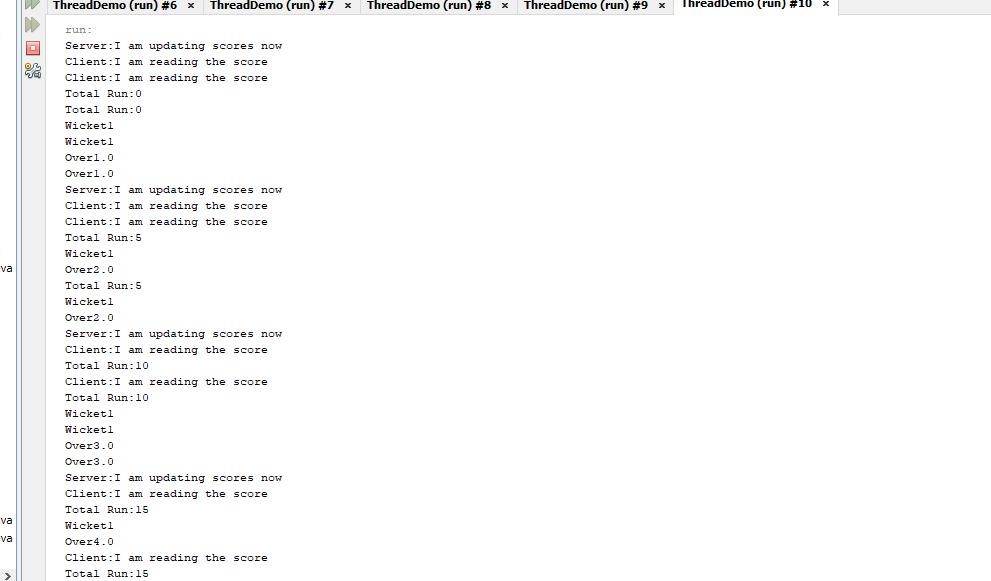
Thread server = new Thread(new Server(runs, size),"Server");

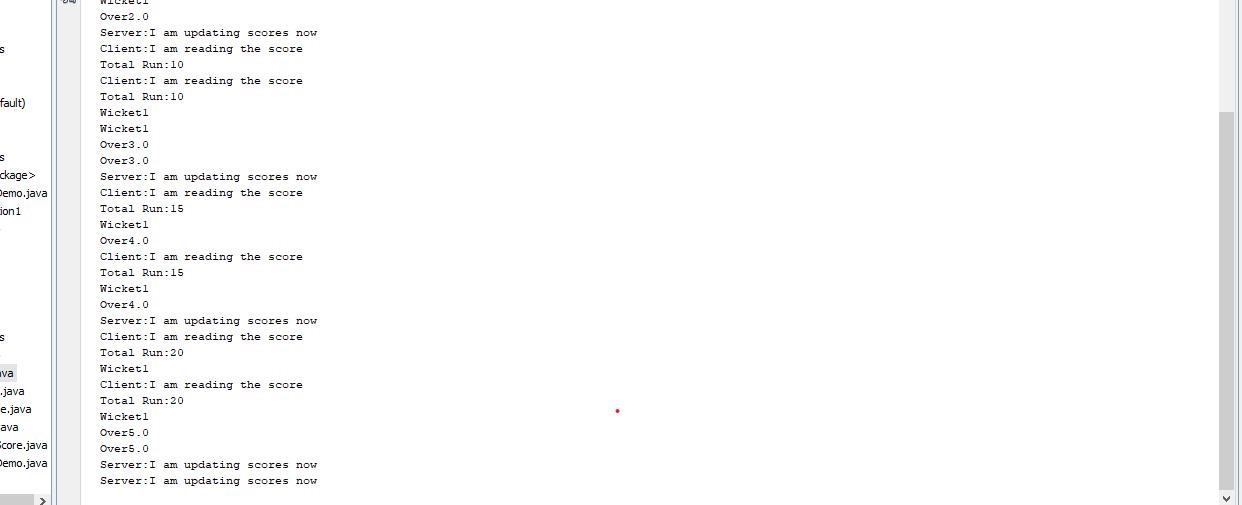
client\_1.start(); client\_2.start(); client\_3.start(); server.start(); }

}

**Output:**







**Result:** Hence ,in this practical I have successfully implemented multi threading