

**Terna Engineering College**  
**Computer Engineering Department**  
**Program: Sem VIII**

**Course: Distributed Computing Lab (CSL802)**

**Faculty: Rohini Patil**

**Experiment No. 8**

**A.1 Aim:** To Implement Load Balancing algorithm.

---

**PART B**  
**(PART B: TO BE COMPLETED BY STUDENTS)**

<b>Roll No.</b> 50	<b>Name:</b> AMEY MAHENDRA THAKUR
<b>Class:</b> BE COMPS B 50	<b>Batch:</b> B3
<b>Date of Experiment:</b> 11-03-2022	<b>Date of Submission:</b> 11-03-2022
<b>Grade:</b>	

**B.1 Software Code written by a student:**

- **LoadBalancing.java**

```
import java.util.*;
public class LoadBalancing {
    public static int l1;
    public static int l2;
    public static int n;
    public static int r1;
    public static void main(String[] args) {
        r1=0;
        Scanner sc=new Scanner(System.in);
        System.out.println("we assume two processor. ");
        System.out.println("Enter the limit of processor one: ");
        l1=sc.nextInt();
        System.out.println("Enter the limit of processor two: ");
        l2=sc.nextInt();
        System.out.println("Enter the number of processes: ");
        n=sc.nextInt();
        n=n+r1;
```

```

Runnable r = new Runnable1();
Thread t = new Thread(r);
Runnable r2 = new Runnable2();
Thread t2 = new Thread(r2);
t.start();
t2.start();
}
}
class Runnable1 implements Runnable{
public void run(){
System.out.println();
int limit1,total1,rem1;
limit1=LoadBalancing.l1;
total1=LoadBalancing.n;
rem1=LoadBalancing.r1;
System.out.println("processor 1 (limit="+limit1+"):");
if(total1==0)
{
System.out.println("No processes are remaining");
}else
{
if(limit1>total1)
{
System.out.println("Underloaded processor");
System.out.println(total1+" processes are executed.");
}
else if(limit1==total1)
{
System.out.println("Normal processor");
System.out.println(total1+" processes are executed.");
System.out.println("No need to forward any process to next processor.");
}
else if(limit1<total1)
{
System.out.println("Overloaded processor");
System.out.println(limit1+" processes are executed.");
rem1=total1-limit1;
System.out.println(+rem1+ " will be forwarded to next processor");
}
}
LoadBalancing.r1=rem1;
}
}

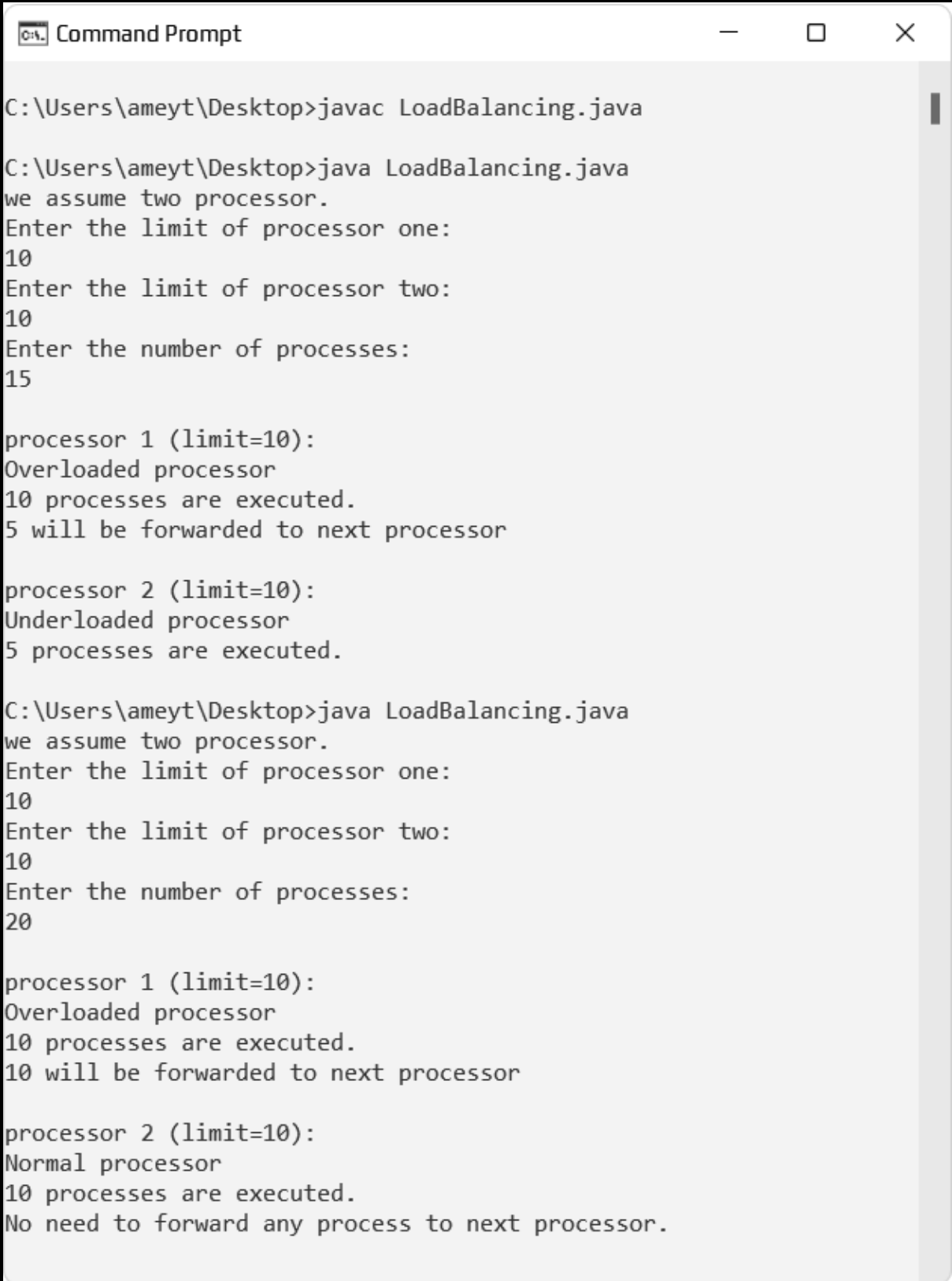
```

```

class Runnable2 implements Runnable{
public void run(){
try{Thread.sleep(5000);}catch(InterruptedException e){System.out.println(e);}
System.out.println();
int limit2,total2,rem2,limitp1;
limit2=LoadBalancing.l2;
total2=LoadBalancing.n;
rem2=LoadBalancing.r1;
limitp1=LoadBalancing.l1;
System.out.println("processor 2 (limit="+limit2+"):");
if(rem2==0)
{
System.out.println("No processes are remaining");
}else
{
if(limit2>rem2)
{
System.out.println("Underloaded processor");
System.out.println(rem2+" processes are executed.");
}
else if(limit2==rem2)
{
System.out.println("Normal processor");
System.out.println(rem2+" processes are executed.");
System.out.println("No need to forward any process to next processor.");
}
else if(limit2<rem2)
{
System.out.println("Overloaded processor");
System.out.println(limit2+" processes are executed.");
rem2=total2-(limitp1+limit2);
System.out.println(+rem2+" will be forwarded to next processor");
}
}
}
}
}

```

## B.2 Input and Output:



```
Command Prompt

C:\Users\ameyt\Desktop>javac LoadBalancing.java

C:\Users\ameyt\Desktop>java LoadBalancing.java
we assume two processor.
Enter the limit of processor one:
10
Enter the limit of processor two:
10
Enter the number of processes:
15

processor 1 (limit=10):
Overloaded processor
10 processes are executed.
5 will be forwarded to next processor

processor 2 (limit=10):
Underloaded processor
5 processes are executed.

C:\Users\ameyt\Desktop>java LoadBalancing.java
we assume two processor.
Enter the limit of processor one:
10
Enter the limit of processor two:
10
Enter the number of processes:
20

processor 1 (limit=10):
Overloaded processor
10 processes are executed.
10 will be forwarded to next processor

processor 2 (limit=10):
Normal processor
10 processes are executed.
No need to forward any process to next processor.
```

### **B.3 Observations and learning:**

- Load balancing is the practice of spreading the workload across distributed system nodes in order to optimise resource efficiency and task response time while avoiding a situation in which some nodes are substantially loaded while others are idle or performing little work.
- Load balancing solutions are designed to establish a dispersed network in which requests are evenly spread across several servers. Load sharing, on the other hand, includes sending a portion of the traffic to one server and the rest to another.

### **B.4 Conclusion:**

We have successfully implemented a Load Balancing algorithm using java.

### **B.5 Question of Curiosity:**

Q1: Differentiate between static scheduling and dynamic scheduling.

ANS:

- Static Scheduling is the mechanism, where we have already controlled the order/way that the threads/processes are executing in our code (Compile-time). If you have used any control (locks, semaphores, joins, sleeps) over threads in your program (to achieve some goal), then you have intended to use static (compile-time) scheduling.
- Dynamic Scheduling is the mechanism where thread scheduling is done by the operating systems based on any scheduling algorithm implemented at the OS level. So the execution order of threads will be completely dependent on that algorithm unless we have put some control on it (with static scheduling).  
Simply when you are implementing any control over threads with your code to achieve some task, you should make sure that you have used minimal controls and also in the most optimised way.
- Generally, we would never have a computer program that would completely depend on only one of Static or Dynamic Scheduling.

Q2: What is the Threshold policy?

ANS:

- Most of the algorithms use the threshold policy to decide whether the node is lightly-loaded or heavily loaded.
- Threshold value is a limiting value of the workload of a node which can be determined by: Static policy: Predefined threshold value for each node depending on processing capability. Dynamic policy: threshold value is calculated from the average workload and a predefined constant.
- Below threshold value node accepts processes to execute, above threshold value node tries to transfer processes to a lightly-loaded node.

- Single-threshold policy may lead to unstable algorithms because underloaded nodes could turn to be overloaded right after a process migration.
- To reduce instability a double-threshold policy has been proposed which is also known as a high-low policy.

Q3: The technology used to distribute service requests to resources is referred to as:

- A. load performing
- B. load scheduling
- C. load balancing
- D. all of the mentioned

ANS: C. Load Balancing

Q4: Which of the following software can be used to implement load balancing?

- A. Apache mod\_balancer
- B. Apache mod\_proxy\_balancer
- C. F6's BigIP
- D. All of the mentioned

ANS: B. Apache mod\_proxy\_balancer

Q5: Which of the following is a more sophisticated load balancer?

- A. workload managers
- B. workspace managers
- C. rackserve managers
- D. all of the mentioned

ANS: A. Workload Managers