Experiment 2.1

Student Name: Sachin Maurya **UID:**21BCS1956
Branch: BE-CSE **Section/Group:**CC_615_B
Date of Performance: 27-02-2024 **Semester:** 6th
Subject Name: Cloud Computing & Distributed Systems Lab
Subject Code: 21CSP-378

1. **Aim:** Simulate a cloud scenario using Mat lab and run a scheduling algorithm.

2. **Objective:** Develop a MATLAB simulation for cloud computing, implementing and evaluating diverse scheduling algorithms. Assess performancemetrics such as task completion time and resource utilization. Analyze results to understand algorithm efficiency, optimize strategies, and contribute insights for effective cloud resource allocation.

3. **Theory:** Cloud computing simulation in MATLAB involves modeling VMs, tasks, and network latency. Various scheduling algorithms like Round Robin are implemented, assessing metrics such as task completion time and resource utilization. The simulation aims to optimize algorithms, providing insights for efficient cloud resource allocation in diverse workloads.

4. Procedure:

Step 1: Write down the java code for executing FCFS scheduling algorithms

```
import java.text.ParseException;

class GFG {

    // Function to find the waiting time for all
    // processes
    static void findWaitingTime(int processes[], int n,
            int bt[], int wt[]) {
        // waiting time for first process is 0
        wt[0] = 0;

        // calculating waiting time
        for (int i = 1; i < n; i++)
            { wt[i] = bt[i - 1] + wt[i -
            1];
```

```
        }
    }

    // Function to calculate turn around time
    static void findTurnAroundTime(int processes[], int n,
             int bt[], int wt[], int tat[]) {
         // calculating turnaround time by adding
        // bt[i] + wt[i]
        for (int i = 0; i < n; i++)
            {tat[i] = bt[i] + wt[i];
        }
    }

    //Function to calculate average time
    static void findavgTime(int processes[], int n, int bt[]) {
        int wt[] = new int[n], tat[] = new int[n];
        int total_wt = 0, total_tat = 0;
        //Function to find waiting time of all processes
        findWaitingTime(processes, n, bt, wt);
        //Function to find turn around time for all processes
        findTurnAroundTime(processes, n, bt, wt, tat);
        //Display processes along with all details
        System.out.printf("Processes Burst time Waiting"
                          +" time Turn around time\n");
        // Calculate total waiting time and total turn
        // around time
        for (int i = 0; i < n; i++)
            { total_wt = total_wt + wt[i];
            total_tat = total_tat + tat[i];
            System.out.printf(" %d ", (i + 1));
            System.out.printf("     %d ", bt[i]);
            System.out.printf("     %d", wt[i]);
            System.out.printf("     %d\n", tat[i]);
        }
        float s = (float)total_wt /(float) n;
        int t = total_tat / n;
        System.out.printf("Average waiting time = %f", s);
        System.out.printf("\n");
        System.out.printf("Average turn around time = %d ", t);
    }
     // Driver code
    public static void main(String[] args) throws ParseException {
        //process id's
        int processes[] = {1, 2, 3};
        int n = processes.length;
          //Burst time of all processes
         int burst_time[] = {10, 5, 8};

        findavgTime(processes, n, burst_time);


    }
}
```

Step 2: Write down the java code for executing SJF scheduling algorithms

```
import java.io.*;
import java.util.*;

class Main {
    public static void main(String[] args)
    {
        Scanner input = new Scanner(System.in);
        int n;
        // Matrix for storing Process Id, Burst
        // Time, Average Waiting Time & Average
        // Turn Around Time.
        int[][] A = new int[100][4];
        int total = 0;
        float avg_wt, avg_tat;
        System.out.println("Enter number of process:");
        n = input.nextInt();
        System.out.println("Enter Burst Time:");
        for (int i = 0; i < n; i++) {
            // User Input Burst Time and alloting
            // Process Id.
            System.out.print("P" + (i + 1) + ": ");
            A[i][1] = input.nextInt();
            A[i][0] = i + 1;
        }
        for (int i = 0; i < n; i++) {
            // Sorting process according to their
            // Burst Time.
            int index = i;
            for (int j = i + 1; j < n; j++) {
                if (A[j][1] < A[index][1])
                    {index = j;
                }
            }
            int temp = A[i][1];
            A[i][1] = A[index][1];
            A[index][1] = temp;
            temp = A[i][0];
            A[i][0] = A[index][0];
            A[index][0] = temp;
        }
        A[0][2] = 0;
        // Calculation of Waiting Times
        for (int i = 1; i < n; i++)
            {A[i][2] = 0;
            for (int j = 0; j < i; j++)
                {A[i][2] += A[j][1];
            }
            total += A[i][2];
        }
        avg_wt = (float)total / n;
        total = 0;
        // Calculation of Turn Around Time and printing the
        // data.
```

```
        System.out.println("P\tBT\tWT\tTAT");
        for (int i = 0; i < n; i++)
            { A[i][3] = A[i][1] +
            A[i][2];
            total += A[i][3];
            System.out.println("P" + A[i][0] + "\t"
                               + A[i][1] + "\t" + A[i][2]
                               + "\t" + A[i][3]);
        }
        avg_tat = (float)total / n;
        System.out.println("Average Waiting Time= "
                           + avg_wt);
        System.out.println("Average Turnaround Time= "
                           + avg_tat);
    }
}
```

Step 3: Write down the java code for executing Round Robin scheduling algorithms

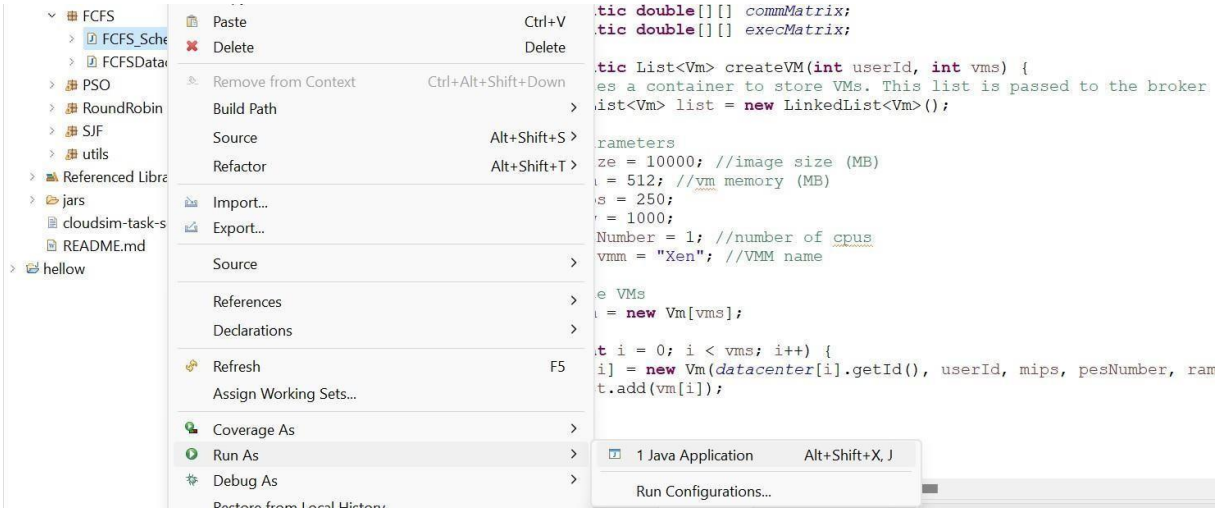
```
public class GFG
{
    // Method to find the waiting time for all
    // processes
    static void findWaitingTime(int processes[], int n,
                  int bt[], int wt[], int quantum)
    {
        // Make a copy of burst times bt[] to store remaining
        // burst times.
        int rem_bt[] = new int[n];
        for (int i = 0 ; i < n ; i++)
            rem_bt[i] =  bt[i];

        int t = 0; // Current time

        // Keep traversing processes in round robin manner
        // until all of them are not done.
        while(true)
        {
            boolean done = true;

            // Traverse all processes one by one repeatedly
            for (int i = 0 ; i < n; i++)
            {
                // If burst time of a process is greater than 0
                // then only need to process further
                if (rem_bt[i] > 0)
                {
                    done = false; // There is a pending process

                    if (rem_bt[i] > quantum)
                    {
                        // Increase the value of t i.e. shows
                        // how much time a process has been processed
                        t += quantum;
```

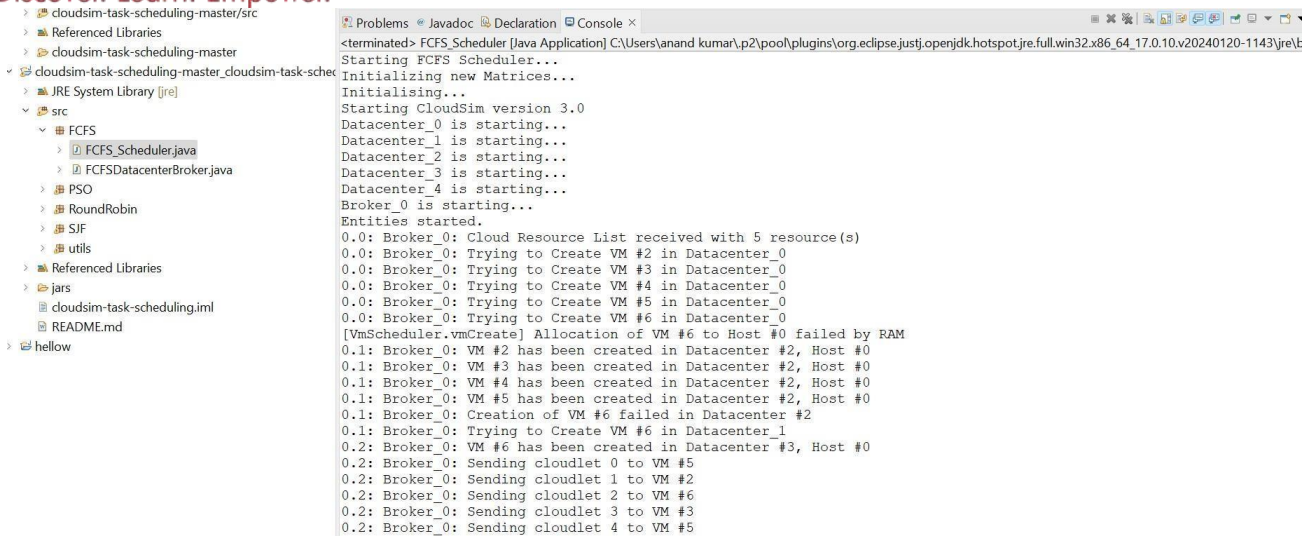
```
                    // Decrease the burst_time of current process
                    // by quantum
                    rem_bt[i] -= quantum;
                }

                // If burst time is smaller than or equal to
                // quantum. Last cycle for this process
                else
                {
                    // Increase the value of t i.e. shows
                    // how much time a process has been processed
                    t = t + rem_bt[i];

                    // Waiting time is current time minus time
                    // used by this process
                    wt[i] = t - bt[i];

                    // As the process gets fully executed
                    // make its remaining burst time = 0
                    rem_bt[i] = 0;
                }
            }
        }

        // If all processes are done
        if (done == true)
          break;
    }
}

// Method to calculate turn around time
static void findTurnAroundTime(int processes[], int n,
                        int bt[], int wt[], int tat[])
{
    // calculating turnaround time by adding
    // bt[i] + wt[i]
    for (int i = 0; i < n ; i++)
        tat[i] = bt[i] + wt[i];
}

// Method to calculate average time
static void findavgTime(int processes[], int n, int bt[],
                                        int quantum)
{
    int wt[] = new int[n], tat[] = new int[n];
    int total_wt = 0, total_tat = 0;

    // Function to find waiting time of all processes
    findWaitingTime(processes, n, bt, wt, quantum);

    // Function to find turn around time for all processes
    findTurnAroundTime(processes, n, bt, wt, tat);
```

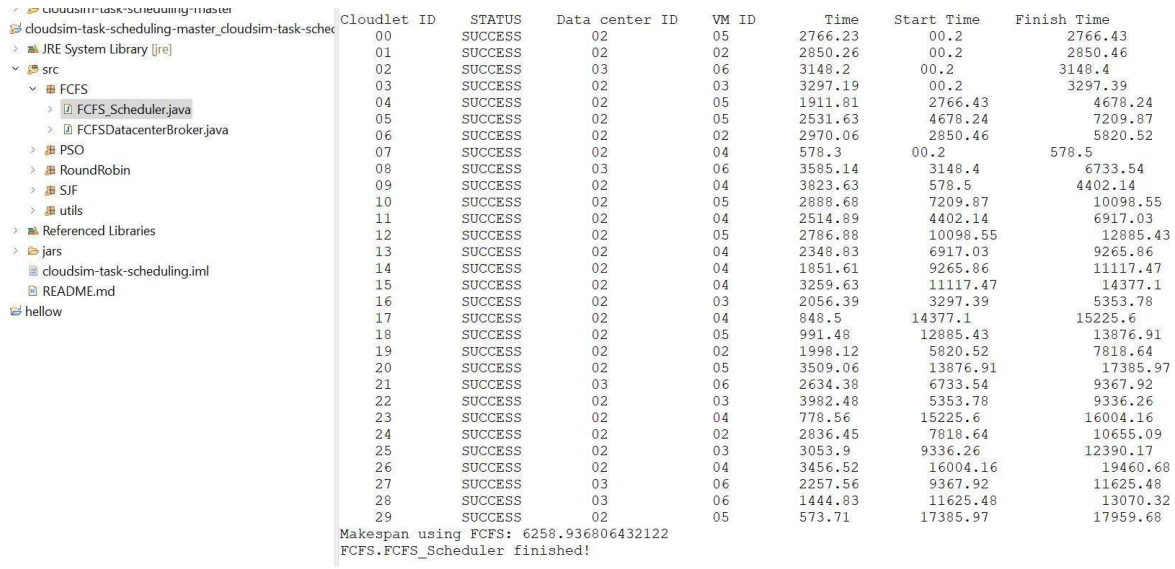
```
        // Display processes along with all details
        System.out.println("PN " + " B " +
                           " WT " + " TAT");

        // Calculate total waiting time and total turn
        // around time
        for (int i=0; i<n; i++)
        {
            total_wt = total_wt + wt[i];
            total_tat = total_tat + tat[i];
            System.out.println(" " + (i+1) + "\t\t" + bt[i] +"\t " +
                               wt[i] +"\t\t " + tat[i]);
        }

        System.out.println("Average waiting time = " +
                           (float)total_wt / (float)n);
        System.out.println("Average turn around time = " +
                           (float)total_tat / (float)n);
    }

    // Driver Method
    public static void main(String[] args)
    {
        // process id's
        int processes[] = { 1, 2, 3};
        int n = processes.length;

        // Burst time of all processes
        int burst_time[] = {10, 5, 8};

        // Time quantum
        int quantum = 2;
        findavgTime(processes, n, burst_time, quantum);
    }
}
```

5. Output:-

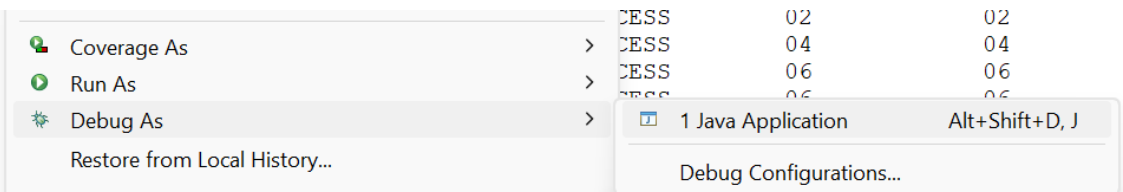
FCFS:

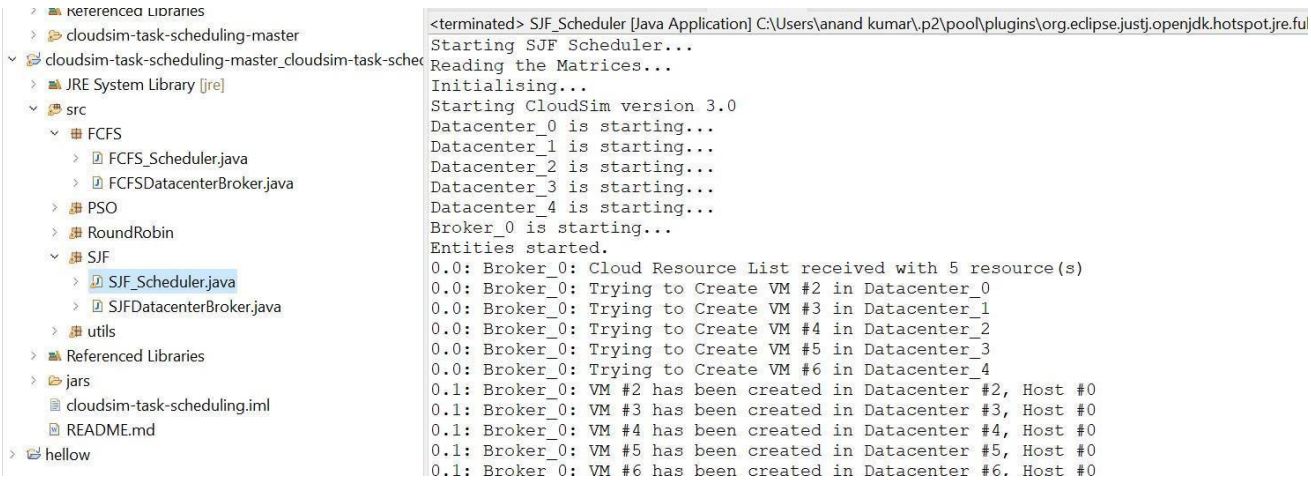


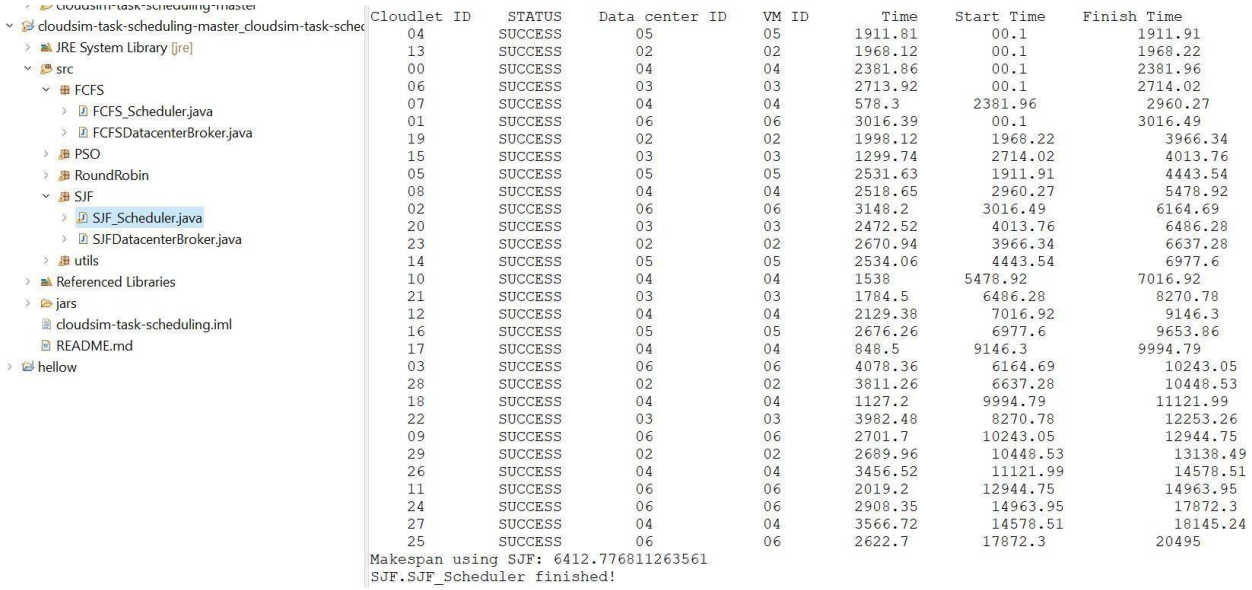

cloudsim-task-scheduling-master/src
Referenced Libraries
cloudsim-task-scheduling-master
cloudsim-task-scheduling-master_cloudsim-task-sched
JRE System Library [jre]
src
FCFS
FCFS_Scheduler.java
FCFSDatacenterBroker.java
PSO
RoundRobin
SJF
utils
Referenced Libraries
jars
cloudsim-task-scheduling.iml
README.md
hellow
Problems
Javadoc
Declaration
Console
<terminated> FCFS_Scheduler [Java Application] C:\Users\anand kumar\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.10.v20240120-1143\jre\b
Starting FCFS Scheduler...
Initializing new Matrices...
Initialising...
Starting CloudSim version 3.0
Datacenter_0 is starting...
Datacenter_1 is starting...
Datacenter_2 is starting...
Datacenter_3 is starting...
Datacenter_4 is starting...
Broker_0 is starting...
Entities started.
0.0: Broker_0: Cloud Resource List received with 5 resource(s)
0.0: Broker_0: Trying to Create VM #2 in Datacenter_0
0.0: Broker_0: Trying to Create VM #3 in Datacenter_0
0.0: Broker_0: Trying to Create VM #4 in Datacenter_0
0.0: Broker_0: Trying to Create VM #5 in Datacenter_0
0.0: Broker_0: Trying to Create VM #6 in Datacenter_0
[VmScheduler.vmCreate] Allocation of VM #6 to Host #0 failed by RAM
0.1: Broker_0: VM #2 has been created in Datacenter #2, Host #0
0.1: Broker_0: VM #3 has been created in Datacenter #2, Host #0
0.1: Broker_0: VM #4 has been created in Datacenter #2, Host #0
0.1: Broker_0: VM #5 has been created in Datacenter #2, Host #0
0.1: Broker_0: Creation of VM #6 failed in Datacenter #2
0.1: Broker_0: Trying to Create VM #6 in Datacenter_1
0.2: Broker_0: VM #6 has been created in Datacenter #3, Host #0
0.2: Broker_0: Sending cloudlet 0 to VM #5
0.2: Broker_0: Sending cloudlet 1 to VM #2
0.2: Broker_0: Sending cloudlet 2 to VM #6
0.2: Broker_0: Sending cloudlet 3 to VM #3
0.2: Broker_0: Sending cloudlet 4 to VM #5


cloudsim-task-scheduling-master_cloudsim-task-sched
JRE System Library [jre]
src
FCFS
FCFS_Scheduler.java
FCFSDatacenterBroker.java
PSO
RoundRobin
SJF
utils
Referenced Libraries
jars
cloudsim-task-scheduling.iml
README.md
hellow
Cloudlet ID STATUS Data center ID VM ID Time Start Time Finish Time
00 SUCCESS 02 05 2766.23 00.2 2766.43
01 SUCCESS 02 02 2850.26 00.2 2850.46
02 SUCCESS 03 06 3148.2 00.2 3148.4
03 SUCCESS 02 03 3297.19 00.2 3297.39
04 SUCCESS 02 05 1911.81 2766.43 4678.24
05 SUCCESS 02 05 2531.63 4678.24 7209.87
06 SUCCESS 02 02 2970.06 2850.46 5820.52
07 SUCCESS 02 04 578.3 00.2 578.5
08 SUCCESS 03 06 3585.14 3148.4 6733.54
09 SUCCESS 02 04 3823.63 578.5 4402.14
10 SUCCESS 02 05 2888.68 7209.87 10098.55
11 SUCCESS 02 04 2514.89 4402.14 6917.03
12 SUCCESS 02 05 2786.88 10098.55 12885.43
13 SUCCESS 02 04 2348.83 6917.03 9265.86
14 SUCCESS 02 04 1851.61 9265.86 11117.47
15 SUCCESS 02 04 3259.63 11117.47 14377.1
16 SUCCESS 02 03 2056.39 3297.39 5353.78
17 SUCCESS 02 04 848.5 14377.1 15225.6
18 SUCCESS 02 05 991.48 12885.43 13876.91
19 SUCCESS 02 02 1998.12 5820.52 7818.64
20 SUCCESS 02 05 3509.06 13876.91 17385.97
21 SUCCESS 03 06 2634.38 6733.54 9367.92
22 SUCCESS 02 03 3982.48 5353.78 9336.26
23 SUCCESS 02 04 778.56 15225.6 16004.16
24 SUCCESS 02 02 2836.45 7818.64 10655.09
25 SUCCESS 02 03 3053.9 9336.26 12390.17
26 SUCCESS 02 04 3456.52 16004.16 19460.68
27 SUCCESS 03 06 2257.56 9367.92 11625.48
28 SUCCESS 03 06 1444.83 11625.48 13070.32
29 SUCCESS 02 05 573.71 17385.97 17959.68
Makespan using FCFS: 6258.936806432122
FCFS.FCFS_Scheduler finished!

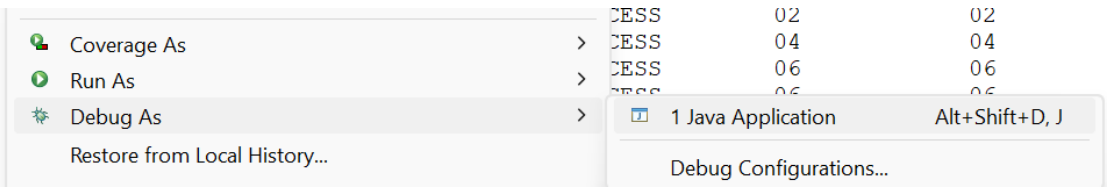
SJF:

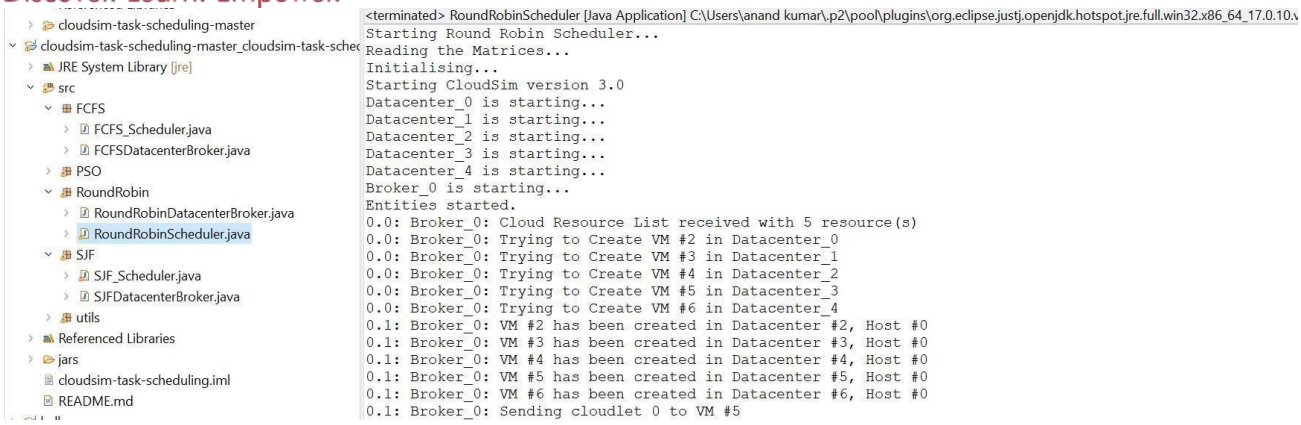

Coverage As
Run As
Debug As
Restore from Local History...
1 Java Application Alt+Shift+D, J
Debug Configurations...

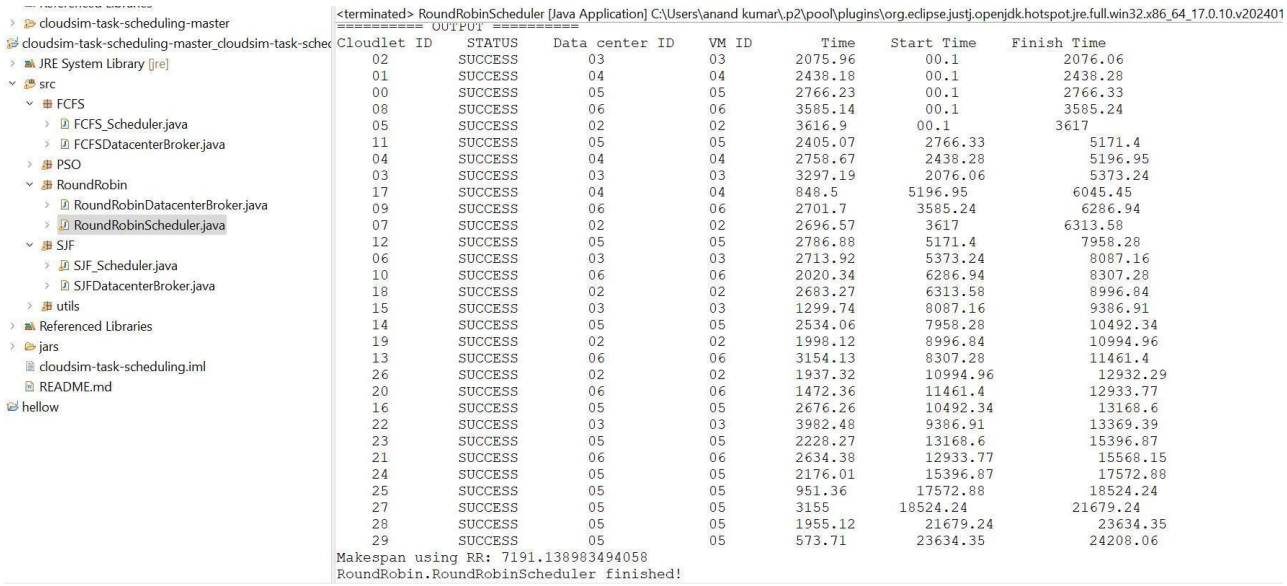

Referenced Libraries
cloudsim-task-scheduling-master
cloudsim-task-scheduling-master_cloudsim-task-schec
JRE System Library [jre]
src
FCFS
FCFS_Scheduler.java
FCFSDatacenterBroker.java
PSO
RoundRobin
SJF
SJF_Scheduler.java
SJFDatacenterBroker.java
utils
Referenced Libraries
jars
cloudsim-task-scheduling.iml
README.md
hellow
<terminated> SJF_Scheduler [Java Application] C:\Users\anand kumar\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.ful
Starting SJF Scheduler...
Reading the Matrices...
Initialising...
Starting CloudSim version 3.0
Datacenter_0 is starting...
Datacenter_1 is starting...
Datacenter_2 is starting...
Datacenter_3 is starting...
Datacenter_4 is starting...
Broker_0 is starting...
Entities started.
0.0: Broker_0: Cloud Resource List received with 5 resource(s)
0.0: Broker_0: Trying to Create VM #2 in Datacenter_0
0.0: Broker_0: Trying to Create VM #3 in Datacenter_1
0.0: Broker_0: Trying to Create VM #4 in Datacenter_2
0.0: Broker_0: Trying to Create VM #5 in Datacenter_3
0.0: Broker_0: Trying to Create VM #6 in Datacenter_4
0.1: Broker_0: VM #2 has been created in Datacenter #2, Host #0
0.1: Broker_0: VM #3 has been created in Datacenter #3, Host #0
0.1: Broker_0: VM #4 has been created in Datacenter #4, Host #0
0.1: Broker_0: VM #5 has been created in Datacenter #5, Host #0
0.1: Broker_0: VM #6 has been created in Datacenter #6, Host #0


cloudsim-task-scheduling-master_cloudsim-task-schec
JRE System Library [jre]
src
FCFS
FCFS_Scheduler.java
FCFSDatacenterBroker.java
PSO
RoundRobin
SJF
SJF_Scheduler.java
SJFDatacenterBroker.java
utils
Referenced Libraries
jars
cloudsim-task-scheduling.iml
README.md
hellow
Cloudlet ID STATUS Data center ID VM ID Time Start Time Finish Time
04 SUCCESS 05 05 1911.81 00.1 1911.91
13 SUCCESS 02 02 1968.12 00.1 1968.22
00 SUCCESS 04 04 2381.86 00.1 2381.96
06 SUCCESS 03 03 2713.92 00.1 2714.02
07 SUCCESS 04 04 578.3 2381.96 2960.27
01 SUCCESS 06 06 3016.39 00.1 3016.49
19 SUCCESS 02 02 1998.12 1968.22 3966.34
15 SUCCESS 03 03 1299.74 2714.02 4013.76
05 SUCCESS 05 05 2531.63 1911.91 4443.54
08 SUCCESS 04 04 2518.65 2960.27 5478.92
02 SUCCESS 06 06 3148.2 3016.49 6164.69
20 SUCCESS 03 03 2472.52 4013.76 6486.28
23 SUCCESS 02 02 2670.94 3966.34 6637.28
14 SUCCESS 05 05 2534.06 4443.54 6977.6
10 SUCCESS 04 04 1538 5478.92 7016.92
21 SUCCESS 03 03 1784.5 6486.28 8270.78
12 SUCCESS 04 04 2129.38 7016.92 9146.3
16 SUCCESS 05 05 2676.26 6977.6 9653.86
17 SUCCESS 04 04 848.5 9146.3 9994.79
03 SUCCESS 06 06 4078.36 6164.69 10243.05
28 SUCCESS 02 02 3811.26 6637.28 10448.53
18 SUCCESS 04 04 1127.2 9994.79 11121.99
22 SUCCESS 03 03 3982.48 8270.78 12253.26
09 SUCCESS 06 06 2701.7 10243.05 12944.75
29 SUCCESS 02 02 2689.96 10448.53 13138.49
26 SUCCESS 04 04 3456.52 11121.99 14578.51
11 SUCCESS 06 06 2019.2 12944.75 14963.95
24 SUCCESS 06 06 2908.35 14963.95 17872.3
27 SUCCESS 04 04 3566.72 14578.51 18145.24
25 SUCCESS 06 06 2622.7 17872.3 20495
Makespan using SJF: 6412.776811263561
SJF.SJF_Scheduler finished!

Round Robin:


Coverage As
Run As
Debug As
Restore from Local History...
1 Java Application Alt+Shift+D, J
Debug Configurations...





6. Learning Outcome:

i). Learned how to install and use Eclipse IDE

ii). Learned how to install Cloud sim IDE and how to use it with eclipse.

iii). Learned how to simulate in Eclipse using cloud sim IDE.