

Exp No: 7

Date:

## CLOUD SIMULATION

### **IMPLEMENT ROUND ROBIN TASK SCHEDULING IN BOTH TIME SHARED AND SPACE SHARED CPU ASSIGNMENT**

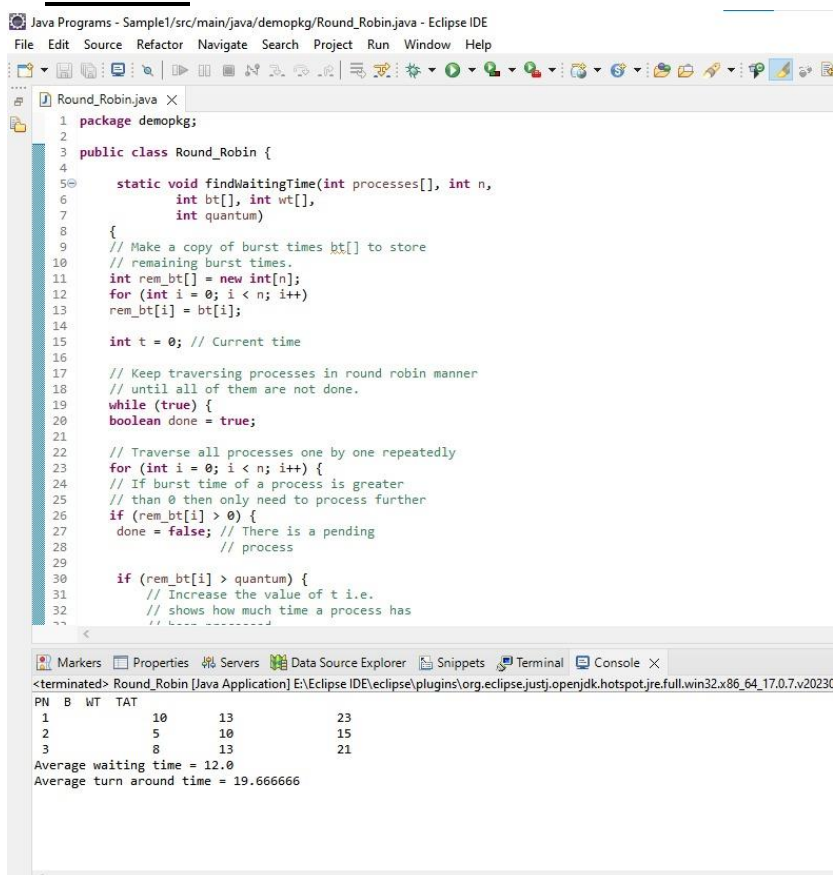
#### **AIM:**

Implement RoundRobin task scheduling in both TimeShared and SpaceShared CPU assignments.

#### **PROCEDURE:**

1. Create a new project by selecting java console line application template and JDK 18.
2. Open project settings from the file menu of the options window.
3. Navigate to project dependencies and select on add external jars and then click on 'Browse' to open the path where you have unzipped the Cloudsim Jars and click on apply.
4. Create a java file with the cloudsim code to implement the Round robin scheduling algorithm.
5. Run the application as a java file to see the output in the console below.

#### **OUTPUT:**



```
1 package demopkg;
2
3 public class Round_Robin {
4
5     static void findWaitingTime(int processes[], int n,
6         int bt[], int wt[],
7         int quantum)
8     {
9         // Make a copy of burst times bt[] to store
10        // remaining burst times.
11        int rem_bt[] = new int[n];
12        for (int i = 0; i < n; i++)
13            rem_bt[i] = bt[i];
14
15        int t = 0; // Current time
16
17        // Keep traversing processes in round robin manner
18        // until all of them are not done.
19        while (true) {
20            boolean done = true;
21
22            // Traverse all processes one by one repeatedly
23            for (int i = 0; i < n; i++) {
24                // If burst time of a process is greater
25                // than 0 then only need to process further
26                if (rem_bt[i] > 0) {
27                    done = false; // There is a pending
28                               // process
29
30                    if (rem_bt[i] > quantum) {
31                        // Increase the value of t i.e.
32                        // shows how much time a process has
33                        // been executed
34                        t += quantum;
35                        rem_bt[i] -= quantum;
36                    }
37                    else {
38                        t += rem_bt[i];
39                        rem_bt[i] = 0;
40                    }
41                }
42            }
43
44            // If all processes are done, then break
45            if (done) break;
46        }
47    }
48
49    // Driver code
50    public static void main (String[] args) {
51        // Processes and their burst times
52        int processes[] = {1, 2, 3};
53        int n = processes.length;
54        int bt[] = {10, 5, 8};
55        int wt[] = new int[n];
56        int quantum = 3;
57
58        findWaitingTime(processes, n, bt, wt, quantum);
59
60        // Print the results
61        System.out.println("Process\t\tBurst Time\t\tWaiting Time\t\tTurn Around Time");
62        for (int i = 0; i < n; i++)
63            System.out.println((i+1) + "\t\t" + bt[i] + "\t\t" + wt[i] + "\t\t" + (bt[i] + wt[i]));
64
65        // Average waiting time
66        double avg_wt = 0;
67        for (int i = 0; i < n; i++)
68            avg_wt += wt[i];
69        avg_wt /= n;
70        System.out.println("Average waiting time = " + avg_wt);
71
72        // Average turn around time
73        double avg_tat = 0;
74        for (int i = 0; i < n; i++)
75            avg_tat += (bt[i] + wt[i]);
76        avg_tat /= n;
77        System.out.println("Average turn around time = " + avg_tat);
78    }
79 }
```

<terminated> Round\_Robin [Java Application] E:\Eclipse IDE\workspace\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86\_64\_17.0.7.v20230

PN	B	WT	TAT
1	10	13	23
2	5	10	15
3	8	13	21

Average waiting time = 12.0  
Average turn around time = 19.666666

#### **RESULT:**

Thus the round robin scheduling algorithm has been successfully implemented using cloud sim.