

CLOUD SIMULATION

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

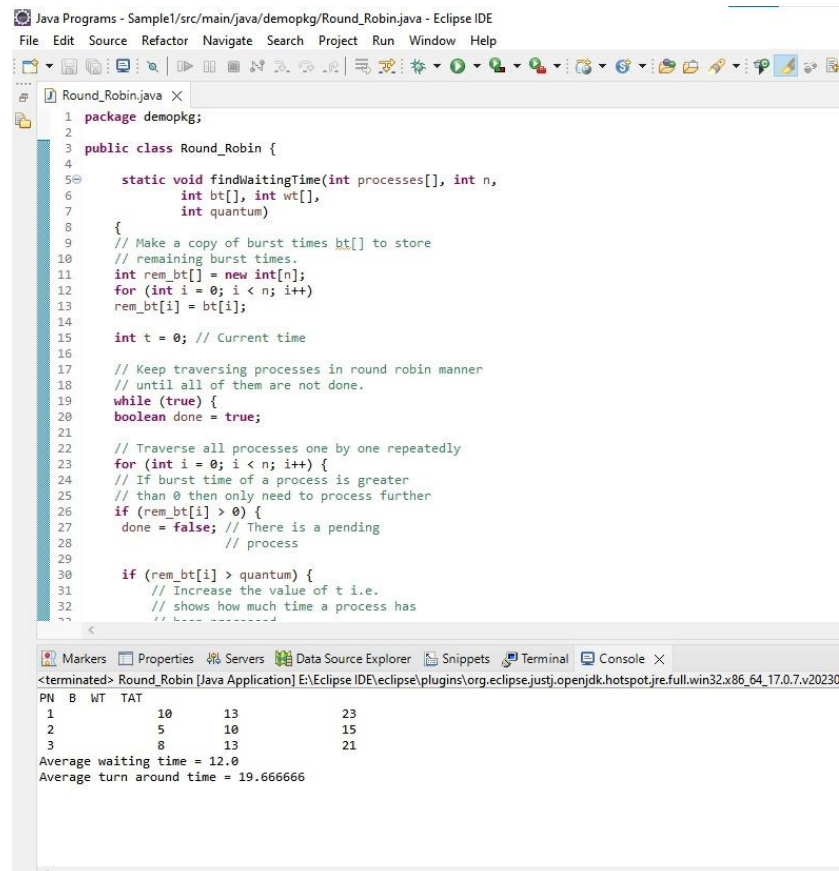
AIM:

Implement Round Robin task scheduling in both Time Shared and Space Shared 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                    }
34                }
35            }
36
37            // If done is true, then all processes are done
38            if (done) break;
39
40            // Increment the time
41            t += quantum;
42
43            // Decrement the remaining burst time of a process
44            rem_bt[i] -= quantum;
45
46            // If the remaining time of a process is less than 0,
47            // it means the process has finished its execution.
48            if (rem_bt[i] < 0) {
49                // Increment the time
50                t += -rem_bt[i];
51                wt[i] = t; // Add the waiting time to the array
52                rem_bt[i] = 0;
53            }
54        }
55    }
56
57    // Driver code
58    public static void main (String[] args) {
59        // Processes and their burst times
60        int processes[] = {1, 2, 3};
61        int n = processes.length;
62        int bt[] = {10, 5, 8};
63        int wt[] = new int[n];
64        int quantum = 3;
65
66        findWaitingTime(processes, n, bt, wt, quantum);
67
68        // Print the array
69        System.out.println("Process\t\tBurst Time\t\tWaiting Time\t\tTurn Around Time");
70        for (int i = 0; i < n; i++)
71            System.out.println(processes[i] + "\t\t" + bt[i] + "\t\t" + wt[i] + "\t\t" + (wt[i] + bt[i]));
72
73        // Average waiting time
74        double avg_wt = 0;
75        for (int i = 0; i < n; i++)
76            avg_wt += wt[i];
77        avg_wt /= n;
78        System.out.println("Average waiting time = " + avg_wt);
79
80        // Average turn around time
81        double avg_tat = 0;
82        for (int i = 0; i < n; i++)
83            avg_tat += (wt[i] + bt[i]);
84        avg_tat /= n;
85        System.out.println("Average turn around time = " + avg_tat);
86    }
87
88 }

```

<terminated> Round_Robin [Java Application] E:\Eclipse IDE\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.