23.59

Bil 212
Fall 2022
Assignment1

Task-Resource Scheduling Algorithm (Job-Resource Scheduler)

In this assignment you are expected to develop a scheduler (scheduler.java) class. The purpose of this class is to schedule the execution time of the input processes by directing them to an appropriate resource (processor core or thread) according to their priority order and start time. Below are the data structures and interface information to be used.

Other Classes

Job Holds arrival time, ID, priority and duration information(int ArrivalTime, int ID, String Priority, int Duration)
You can implement any classes you want except this class.

Data structures to be used:

All processes added with the Add method are first added to a **waitlist** structure according to the start time. The order of addition may not correspond to the order of arrival time. The order in the linked-list is your task.

Transactions held by arrival time should be transferred to 3 different linked-list structures as high (**highPQueue**), **midPQueue** (**midPQueue)** and **lowPQueue** (**lowPQueue**) according to the priority status when the transaction time comes.

It is mandatory to use these data structures. Apart from these, you can use the data structures in the sections seen in the course if you think it is necessary.

Scheduling interface:

public void setResourcesCount(Integer) Sets the total number of resources (cores) the class should use.
public void add(job j) adds job j to the **waitlist** according to the start time. At the end of this operation the waitlist should be sorted by the arrival time field.

public void utilization(Integer) Gets the resource number and displays the throughput value. Throughput is calculated as the ratio of the time a resource is doing work (not idle) to its total time. For example, if a resource worked between time slots 0-2 and 6-7, its throughput is (3+2)/8 = 5/8

public void resourceExplorer(Integer) Gets the resource number, prints the operations (process id, process end time, delay time) executed by the resource to the screen respectively.

public void jobExplorer(job j) prints the id of process j, the source from which it was executed, the start and end time, and the total latency to the screen.
Delay time = start time - arrival time

public void run() starts running the scheduling method. When the run starts, a time simulation also starts. As time flows forward from 0

- All **jobs** executed at arrival time are sent to the relevant queue
- Finished jobs in resources are saved
- New appointments are made to vacant resources.
- Prints the timeline and events on the screen.

Acceptances

setResorcesCount() and add() must be executed before the *run()* method. That is, they will be executed after Scheduling is completely ready to run.

When scheduling, core utilization should be set to the highest level. Resources should not sit idle when there are pending jobs.

Resource assignment priority is set according to the lower number. For example, when both resources 1 and 2 are idle and a job comes in, resource 1 is assigned.

Sample Program and Output:

The example driver code of the designed class is shown below. The process assignments are shown symbolically. **The console output should be as given below. Do not use other words.**

```
J0->0,0,0,H,3
J1->0,1,H,2
J2-> 0,2,M,3
J3->3,3,3,H,3
```

```
Schedular.setResourceCount(2)
Scheduler.add(J0)
Scheduler.add(J1)
Scheduler.add(J2)
Scheduler.add(J3)
Scheduler.run()
System.out.println("------------- ")
Schedular.utilization(1) // when resource 1 has run a total of 6 units and is idle in between
// never happened, 6/6 = 1
System.out.println("------------- ")
Schedular.resourceExplorer(2)
System.out.println("------------- ")
Schedular.jobExplorer(J3)
```

Submission time
30.10.22
23.59

Priority Linked-lists (This is for illustration purposes only, not for output)

```
H                   M
J0->0,0,0,H,3       J2->0,2,M,3
J1->0,1,H,2
J3->3,3,3,H,3
```

Console output:

```
Time       R1              R2
0          J0              J1
1          J0              J1
2          J0              J2
3          J3              J2
4          J3              J2
5          J3
-----------------
R1 yield: 1.00
-----------------
R2: (1,1,0), (2,4,2)
-----------------
islemno    Source     BEGINNING  bitis  Delay
3          R1         3          5      0
```

My submission:

- Add all the classes you wrote as inner classes in *Scheduler.java.* Only *Scheduler.java* file will be loaded on the remote system.
- Add your name, surname and number as a comment at the top of your code.
- Compile with Java 11.
- Make it easier to understand by adding comments inside your code.
- If your code doesn't compile or works incorrectly with the driver code above, make it error-free and submit it.
- Submission restrictions will also be included in the scoring.
- Properly written codes will receive bonus points.