CMSC 125 Machine Problem

Due: May 26, 2022. Max of 2 members per group

CPU Scheduling

There are different scheduling algorithms to determine which processes will run in the CPU when there is more than one process in queue. The characteristics which determine which is the best scheduling algorithm are based on the following criteria: minimum waiting time, minimum response time, and minimum turnaround time all the while maximizing CPU utilization.

The different algorithms to simulate are:
Shortest-Job-First (SJF) – also known as shortest-next-CPU-burst algorithm, which gives priority to processes that have the smallest CPU burst. It can either be non-preemptive or preemptive. Preemptive SJF prioritizes the processes with the smallest remaining time as the processes are running.

Priority Scheduling (PS) – from its name, a process is given a rank or a priority value with the lowest set as the highest priority. If all processes arrive at the same time, no pre-emption occurs. But in the case where there are processes arriving at different times, the processes with higher priority will preempt any current running processes.

Round-Robin (RR) – algorithm where the processes are executed following the first-come first-serve basis but can only run within the set time quantum duration. Processes that will have a time burst of more than the time quantum will have to wait for the next round after all the processes are run within the set time quantum.

Project Description:

1. Given a maximum of 10 processes with each having a set burst time, simulate the different CPU scheduling algorithms (without preemption):
   - SJF
   - Priority Scheduling
   - Round Robin*

   Each burst time (and print to screen) will take a time interval of 1 sec.
   To verify the run of each process, print to screen the system time up to the *seconds* unit and the process number. (see example below)
2. Simulate also the condition where there will be additional processes in the queue while the other processes are running (with preemption). For this case, some of the processes will have values greater than 0 under the Arrival Time or varying values of Arrival Time (time in seconds).

* Round Robin algorithm will have to ask the user to input the duration of time quantum in seconds (Maximum of 10 seconds)

3. Summarize the activities of all process after all the processes have finished running which includes:
   - Ghantt chart – time interval of each process (up to seconds)
   - Waiting time for each process
   - Average waiting time and Turnaround time
4. You can create a menu to let the user choose the following:

```
Provide file name containing process information:

Test.txt
```

```
Choose an Algorithm:
A. Shortest-Job First
B. Priority Scheduling
C. Round Robin

C
```

```
Provide time quantum in seconds:

5

* this input for RR only
```

Sample Input - (All process arrives at the same time)

Test.txt :

Process	Burst Time	Arrival Time	Priority
P1	10	0	1
P2	4	0	1
P3	6	0	1

Take Note of the **Max** values
Processes – 10
Burst Time – 25
Arrival Time – 25
Priority - 10

Output: (Note that each printing will take 1 second each)

```
       Round Robin Algorithm
8:15:02        Running Process P1…
8:15:03        Running Process P1…
```

```
8:15:04        Running Process P1…
8:15:05        Running Process P1…
8:15:06        Running Process P1…
8:15:07        Running Process P2…
8:15:08        Running Process P2…
8:15:09        Running Process P2…
8:15:10        Running Process P2…
8:15:11        Running Process P3…
8:15:12        Running Process P3…
8:15:13        Running Process P3…
8:15:14        Running Process P3…
8:15:15        Running Process P3…
8:15:16        Running Process P1…
8:15:17        Running Process P1…
8:15:18        Running Process P1…
8:15:19        Running Process P1…
8:15:20        Running Process P1…
8:15:21        Running Process P3…
```

Hint:

Java java.util.Timer is a utility class that can be used to schedule a thread to be executed at certain time in future. Java Timer class can be used to schedule a task to be run one-time or to be run at regular intervals.
***java.util.TimerTask** is an abstract class that implements Runnable interface*

Summary

	Time Duration	Waiting time	Turnaround time
P1	8:15:02-8:15:06, 8:15:16-8:15:20	9	19
P2	8:15:07-8:15:10	5	4
P3	8:15:11-8:15:15	14	11

5. Input reading:
   a. Input for Priority Scheduling will have to consider the values in all the four columns, namely, Process, Burst Time, Arrival Time and Priority

Process	Burst Time	Arrival Time	Priority
P1	10	0	2
P2	4	2	1
P3	6	10	3

   b. SJF and Round Robin will only consider the first three columns and ignore the Priority column. For as long as the Arrival Time for all processes are the same, then assume that no preemption will occur.

Project Requirements:

All files must be in a zip file (CS125_MP2_<lastname>.zip) and send via Google Classroom

1. Executable file of your code ex. CS125_MP2_<lastname>.jar
2. All source code- .java files in CODE folder
3. Screen shots of the output of each algorithm with and without processes with different arrival time saved in IMAGES folder
4. Documentation includes description of data structure/s used, algorithm, error handling, difficulties encountered and application manual especially for those who will implement a GUI.
5. Submit as much as possible on time. Early submission will incur 1% additional per day(maximum of 5 days)

Grading Criteria:

Documentation	15
Code Construct	10
Functionality	
SJF	20
Priority Scheduling	20
Round Robin	20
Error Handling	15
Bonus	
GUI	10
Early Submission	5
Max Total Points	**115/100**