Scheduling Algorithms - CPU Scheduling Simulator

Project Descriptions
This project involves implementing two process scheduling algorithms. You will be required to write a Java program to simulate FCFS and RR scheduling policies we discussed in class. The simulator selects a task to run from a ready queue based on the scheduling algorithm. Since the project intends to simulate a CPU scheduler, it does not require any actual process creation or execution. The processes' information is stored in a text file.

When a task(process) is scheduled, the simulator will simply print out what task is selected to run at a time. I have provided a sample run below.

The name of your Java program should be called `CPUScheduler.java`
The selected scheduling algorithms to implement in this project are:

- Round Robin (RR)
- First Come First Serve (FCSF)

The above algorithms are already described in class slides, class video recordings and textbook Chapter 5.

Implementation

The implementation of this project should be completed in Java. Your program will read information about processes from a text file. This supporting text file contains process scheduling information such as the pid, arrival time and CPU burst time. Your program should read in this information, insert the processes into a list, and invoke the scheduler.

Process information
The following example format of how your text file should look like:
P1 0 10
P2 1 8
P3 2 5

- The first column represents a process ID. Process ID uniquely identifies a process.
- The second column represents arrival time. This is the time when the process arrives in the unit of milliseconds
- The third column represents CPU burst. This is the CPU time requested by a time, in the unit of milliseconds

Thus, P1 arrives at 0 and has a CPU burst of 10 milliseconds and so forth.
The program will be run from the command line where you provide the name of the file where the processes are stored.

The simulator first reads task information from the input file and stores all data in a data structure. Then it starts simulating one scheduling algorithm in a time-driven manner. At

each time unit (or slot), it adds any newly arrived task(s) into the ready queue and calls a specific scheduler algorithm in order to select appropriate tasks from the ready queue. When a task is chosen to run, the simulator prints out a message indicating what process ID is chosen to execute for this time slot. If no task is running (i.e. empty ready queue), it prints out an "idle" message. Before advancing to the next time unit, the simulator should update all necessary changes in task and ready queue status.

SAMPLE EXECUTION

Assume that you have created a file `process.txt` with the following data:

```
P0 0 3
P1 1 6
P2 5 4
P3 7 3
```

If you invoke your scheduler (executable `CPUScheduler`) using the command

```
    java CPUScheduler process.txt 2
Note that in the above command, process.txt is the name of the file to
read from and 2 is the time slide for RR
```

then your program should have a behavior similar to the following:

```
------------------------------------------------
            CPU Scheduling Simulation
------------------------------------------------



------------------------------------------------
       First Come First Served Scheduling
------------------------------------------------
[0-3]   P0 running
[3-9]   P1 running
[9-13]  P2 running
[13-16] P3 running

Turnaround times:
        P0 = 3
        P1 = 8
        P2 = 8
        P3 = 9

Wait times:
        P0 = 0
        P1 = 2
        P2 = 4
        P3 = 6

Response times:
```

```
        P0 = 0
        P1 = 2
        P2 = 4
        P3 = 6


Average turnaround time: 7.00
Average wait time: 3.00
Average response time: 3.00



-------------------------------------------------
             Round Robin Scheduling
-------------------------------------------------

[0-2]   P0 running
[2-4]   P1 running
[4-5]   P0 running
[5-7]   P1 running
[7-9]   P2 running
[9-11]  P3 running
[11-13] P1 running
[13-15] P2 running
[15-16] P3 running

Turnaround times:
        P0 = 5
        P1 = 12
        P2 = 10
        P3 = 9

Wait times:
        P0 = 2
        P1 = 6
        P2 = 6
        P3 = 6

Response times:
        P0 = 0
        P1 = 1
        P2 = 2
        P3 = 2

Average turnaround time: 9.00
Average wait time: 5.00
Average Response time: 1.25



-------------------------------------------------
Project done by [Place your full name here]
-------------------------------------------------
```