Scheduling Simulation Assignment

Assignment Objective:

Build a program which schedules simulated processes.

Design:

1. The simulator implements three different CPU scheduling algorithms. The simulator selects a process to run from the ready queue based on the scheduling algorithm chosen at runtime. Since the assignment intends to simulate a CPU scheduler, it does not require any actual process creation or execution. When the CPU scheduler chooses the next process, the simulator will simply print out which the process selected to run at that time. The simulator output is like a Gantt chart.

2. A **Process** is an object in this assignment storing integers which are its ID, arrival time, and CPU burst length. Another object in this assignment is the **Scheduler** which simulates the CPU scheduler for an operating system, the one chosen at runtime. The scheduler contains the ready queue and the only operations the scheduler performs is add and remove. The add operation adds a process into the ready queue into its appropriate spot within the ready queue according to the CPU scheduling algorithm, the one chosen at runtime. The remove operation removes a process from the ready queue according to the CPU scheduling algorithm, the one chosen at runtime. The scheduler implements the CPU scheduling algorithms: First Come First Serve, Shortest Remaining Time First which is the preemptive version of Shortest Job First, and Round Robin.

3. Create a driver class and make the name of the driver class **Assignment2** and it should only contain only one method:
   `public static void main(String args[]).`
   The main method receives, via the command line arguments, the name of the CPU scheduler that your simulator program will use. If Round Robin is the CPU scheduler chosen, then the main method also receives the time quantum value via an additional command line argument. The main method opens the file **assignment2.txt** reading in the entire set of processes and initiates execution of the simulator program. Assume there is only a single processor with only one core. The main method itself should be fairly short.

   The command to launch the program using First Come First Serve scheduling:
   `java Assignment2 FCFS`

   The command to launch the program using Shortest Remaining Time First scheduling:
   `java Assignment2 SRTF`

   The command to launch the program using Round Robin scheduling with a time quantum of 10:
   `java Assignment2 RR 10`

4. The input to the program reads from a plain text file called **assignment2.txt**. This is the statement to use to open the file:

   ```
   FileInputStream fstream = new FileInputStream("assignment2.txt");
   ```

   Assuming you are using Eclipse to create your project, you will store the input file assignment2.txt in the parent directory of your source code (**.java** files) which happens to be the main directory of your project in Eclipse. If you are using some other development environment, then you have to figure out where to store the input file.

   Each line in the file represents a process, 3 integers separated by a space or spaces. The process information includes the process ID, arrival time, and CPU burst length. Arrival time is the time at which the scheduler receives the process and places it in the ready queue. You can assume arrival times of the processes in the input file are in non-decreasing order. Process IDs are unique. Arrival times may be duplicated, which means multiple processes may arrive at the same time. The following table is an example of a three process input file. The text in the top row of the table is just to label the value in each column and would not appear in the input file. Remember, a space or spaces separate the integers on each line.

Process ID	Arrival Time	CPU Burst Length
1	0	10
2	0	20
3	3	5

5. For the output, the example below best describes what the program should produce. The program will not be tested on this sample input but a different sample input.

   Here is the example input:

   ```
   1  0 10
   2  0  9
   3  3  5
   4  7  4
   5 10  6
   6 10  7
   ```

   Here is the output produced for the above example input given the command `java Assignment2 FCFS` to execute the program:

   ```
   Scheduling algorithm: First Come First Serve
   ============================================================
   <system time    0> process    1 is running
   <system time    1> process    1 is running
   <system time    2> process    1 is running
   <system time    3> process    1 is running
   <system time    4> process    1 is running
   <system time    5> process    1 is running
   <system time    6> process    1 is running
   <system time    7> process    1 is running
   <system time    8> process    1 is running
   ```

```
<system time    9> process    1 is running
<system time   10> process    1 is finished....
<system time   10> process    2 is running
<system time   11> process    2 is running
<system time   12> process    2 is running
<system time   13> process    2 is running
<system time   14> process    2 is running
<system time   15> process    2 is running
<system time   16> process    2 is running
<system time   17> process    2 is running
<system time   18> process    2 is running
<system time   19> process    2 is finished....
<system time   19> process    3 is running
<system time   20> process    3 is running
<system time   21> process    3 is running
<system time   22> process    3 is running
<system time   23> process    3 is running
<system time   24> process    3 is finished....
<system time   24> process    4 is running
<system time   25> process    4 is running
<system time   26> process    4 is running
<system time   27> process    4 is running
<system time   28> process    4 is finished....
<system time   28> process    5 is running
<system time   29> process    5 is running
<system time   30> process    5 is running
<system time   31> process    5 is running
<system time   32> process    5 is running
<system time   33> process    5 is running
<system time   34> process    5 is finished....
<system time   34> process    6 is running
<system time   35> process    6 is running
<system time   36> process    6 is running
<system time   37> process    6 is running
<system time   38> process    6 is running
<system time   39> process    6 is running
<system time   40> process    6 is running
<system time   41> process    6 is finished....
<system time   41> All processes finished......
============================================================
Average CPU usage:       100.00%
Average waiting time:     14.17
Average response time:    14.17
Average turnaround time:  21.00
============================================================
```

6. You must declare public each class you create which means you define each class in its own file.

7. You must declare private all the data members in every class you create.

8. It is possible to use inheritance in this assignment, extends one class in another class. But, it has to be the proper use of inheritance. If you simply use extends in one class to allow it access to the private data members of another class and the two classes do not have similar behavior, then that is not the proper use of inheritance. Improper use of inheritance will cause a loss in points.

9. **Tip:** Make your program as modular as possible, not placing all your code in one .java file. You can create as many classes as you need in addition to the class described above. Methods being reasonably small follow the guidance that "A function does one thing and does it well." You will lose a lot of points for code readability if you do not make your program as modular as possible. But do not go overboard on creating classes and methods. Your common sense guides your creation of classes and methods.

10. Do **NOT** use your own packages in your program. If you see the keyword **package** on the top line of any of your .java files, then you created a package. Create every .java file in the **src** folder of your Eclipse project, if you're using Eclipse.

11. Do **NOT** use any graphical user interface code in your program!

12. Do **NOT** type any comments in your program. If you do a good job of programming by following the advice in number 9 above, then it will be easy for me to determine the task of your code.

Grading Criteria:

The assignment is worth a total of 20 points, broken down as follows:

1. If your code does not implement the task described in this assignment, then the grade for the assignment is zero.
2. If your program does not compile successfully then the grade for the assignment is zero.
3. If your program produces runtime errors which prevents the grader from determining if your code works properly then the grade for the assignment is zero.

Important: Make sure your project compiles and runs via the command line using the Java JDK because I will not use any other Java development tool to compile and run your project code.

If the program compiles successfully and executes without significant runtime errors, then the grade computes as follows:
Followed proper submission instructions, 4 points:

1. Was the file submitted a zip file?
2. The zip file has the correct filename.
3. The contents of the zip file are in the correct format.
4. The keyword **package** does not appear at the top of any of the .java files.

Code implementation and Program execution, 12 points:

- The driver file has the correct filename, **Assignment2.java** and contains only the method **main** performing the exact tasks as described in the assignment description.
- The code performs all the tasks as described in the assignment description.
- The code is free from logical errors.
- Program input, the program properly processes the input.
- Program output, the program produces the proper results for the assignment.

Code readability, 4 points:

- Good variable, method, and class names.
- Variables, classes, and methods that have a single small purpose.
- Consistent indentation and formatting style.
- Reduction of the nesting level in code.

Late submission penalty: assignments submitted after the due date are subjected to a 2-point deduction for each day late.

Late submission policy: you **CAN** submit your assignment early, before the due date. You are given plenty of time to complete the assignment well before the due date. Therefore, I do **NOT** accept any reason for not counting late points if you decide to wait until the due date (and the last possible moment) to submit your assignment and something happens to cause you to submit your assignment late. I only use the date submitted, ignoring the time as well as Blackboard's late submission label.

Submission Instructions:

Go to the folder containing the .java files of your assignment and select all (and **ONLY**) the .java files which you created for the assignment in order to place them in a Zip file. The file can **NOT** be a **7z** or **rar** file! Then, follow the directions below for creating a zip file depending on the operating system running on the computer containing your assignment's .java files.

Creating a Zip file in Microsoft Windows (any version):

1. Right-click any of the selected .java files to display a pop-up menu.
2. Click on **Send to**.
3. Click on **Compressed (zipped) Folder**.
4. Rename your Zip file as described below.
5. Follow the directions below to submit your assignment.

Creating a Zip file in Mac OS X:

1. Click **File** on the menu bar.
2. Click on **Compress ? Items** where ? is the number of .java files you selected.
3. Mac OS X creates the file **Archive.zip**.
4. Rename **Archive** as described below.
5. Follow the directions below to submit your assignment.

Save the Zip file with the filename having the following format:
your last name,
followed by an underscore _,
followed by your first name,
followed by an underscore _,
followed by the word **Assignment2**.
For example, if your name is John Doe then the filename would be: **Doe_John_Assignment2**

Once you submit your assignment you will not be able to resubmit it!
Make absolutely sure the assignment you want to submit is the assignment you want graded.
There will be **NO** exceptions to this rule!

You will submit your Zip file via your CUNY Blackboard account.
The only accepted submission method!

Follow these instructions:

Log onto your CUNY Blackboard account.
Click on the CSCI 340 course link in the list of courses you are taking this semester.
Click on **Assignments** tab in the red area on the left side of the webpage.
You will see the **Scheduler Simulation Assignment**.
Click on the assignment.
Upload your Zip file and then click the submit button to submit your assignment.

Due Date: Submit this assignment on or before 11:59 p.m. Wednesday, November 18, 2020.