Lottery Scheduling Assignment

Assignment objective:

Build a program which schedules simulated CPU processes simulating the lottery scheduling algorithm for its CPU Scheduler. Hence, the simulator obtains a process to run from the ready queue of the lottery scheduling algorithm. 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 process was selected to run at that time. The simulator output is like the Gantt chart style.

Overview and Design:

1. The CPU scheduling algorithm implemented for this assignment is lottery scheduling where processes are each assigned some number of lottery tickets, and the scheduler draws a random ticket to select the next process.
    a. How is a process assigned lottery tickets? For this implementation, the scheduling algorithm bases the lottery ticket assignment on the process's priority number. The lottery scheduler assigns to a process no more than a set maximum (M) number of lottery tickets. A process's priority is an integer in the range of (SPN) smallest priority number to (LPN) largest priority number. Remember, the smaller the priority number, the higher the process's priority. Therefore, the lottery scheduler assigns a larger percentage of the maximum number of lottery tickets to a higher priority process. Hence, the formula to compute the number of lottery tickets assigned to a process is: `(1 - (P - SPN) / (LPN - SPN + 1)) * M`
    b. How does the lottery scheduler pick the next process to run when the CPU becomes idle? This implementation of lottery scheduling is non-preemptive. All the scheduler needs are a random number generator, a data structure (a list) to track the processes in the ready queue, and the current total number of tickets allocated to the processes in the ready queue. To make a scheduling decision, pick a random number (the winner) from the total number of tickets and then traverse the list, using a simple counter, to find the winner. See the pseudocode on the next page. Improving the efficiency of this process would result in keeping the list in sorted order, from the highest number of tickets to the lowest. This ensures in general that the fewest number of iterations are taken through the list, especially if there are a few processes that possess most of the tickets.

```
// counter tracks finding the winner.
int counter = 0;

// winner is a random number generated between
// 0 and the total number of tickets allocated.
int winner = getRandomNumber(0, totalNumberTickets);

// Use current: to walk through the list of jobs.
int current = 0;
while (current < numberOfProcessesInReadyQueue)
{
  counter = counter + ReadyQueue(current).numberOfTickets;
  if (counter > winner) break;  // found the winner.
  current = current++;
}
// current is the winner, schedule it.
```

2. The possible objects in this assignment which can become classes in the program code might be the reader of input, a process, the lottery scheduler, the CPU, the calculator of the scheduling criteria averages, and the printer of output.

3. The reader of input can read and create each process keeping them in a list in the order of their arrival time to pass on to the CPU.

4. A process object stores all the information about itself: ID, arrival time, CPU burst length, and priority. A process object should definitely keep track of information needed to compute its own statistics (turnaround time, waiting time, and response time) and to compute those statistics.

5. The lottery scheduler maintains the ready queue and provides a method to add an arriving process to the ready queue and remove the next running process from the ready queue.

6. The CPU object simulates handling arriving processes, obtaining from the lottery scheduler the next process from the ready queue to run assuming the ready is not empty, executing the running process, and maintaining the current clock time which is an integer and initialized to zero at the start of this assignment's program. The CPU object also computes the CPU utilization criteria value. Assume the CPU object represents only a single processor with only one core.

7. The scheduling criteria calculator obtains from each process its turnaround time, waiting time, and response in order to compute the scheduling criteria averages.

8. The printer of output is just a convenient and central location to provide methods that print out the different kinds of output for the assignment.

9. The input to your program reads from a plain text file called **assignment2.txt**. This is the statement 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, you will have to figure out where to store the input file.

   Each line in the file represents a process, 4 integers separated by a space or spaces in the following order: the process ID, arrival time, CPU burst length, and priority. Arrival time is the time at which the scheduler receives the process and places it in the ready queue. 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.

10. For the output, the example below best describes what the program output should resemble.
    This is not an actual output example of this scheduling algorithm.

    ```
    Lottery Scheduling Algorithm
    ===============================================================
    <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    2 is running
    <system time    5> process    2 is running
    <system time    6> process    2 is running
    <system time    7> process    2 is running
    <system time    8> process    3 is running
    <system time    9> process    3 is running
    <system time   10> process    3 is running
    <system time   11> process    3 is running
    <system time   12> process    4 is running
    <system time   13> process    4 is running
    <system time   14> process    4 is running
    <system time   15> process    4 is running
    <system time   16> process    4 is finished....
    <system time   16> process    5 is running
    <system time   17> process    5 is running
    <system time   18> process    5 is running
    <system time   19> process    5 is running
    <system time   20> process    6 is running
    <system time   21> process    6 is running
    <system time   22> process    6 is running
    <system time   23> process    6 is running
    <system time   24> process    1 is running
    <system time   25> process    1 is running
    <system time   26> process    1 is running
    <system time   27> process    1 is running
    ```

```
<system time   28> process    1 is running
<system time   29> process    1 is running
<system time   30> process    1 is finished....
<system time   30> process    2 is running
<system time   31> process    2 is running
<system time   32> process    2 is running
<system time   33> process    2 is running
<system time   34> process    2 is running
<system time   35> process    2 is running
<system time   36> process    2 is running
<system time   37> process    2 is running
<system time   38> process    3 is running
<system time   39> process    3 is finished....
<system time   39> process    5 is running
<system time   40> process    5 is running
<system time   41> process    5 is finished....
<system time   41> process    6 is running
<system time   42> process    6 is running
<system time   43> process    6 is running
<system time   44> process    6 is running
<system time   45> process    6 is running
<system time   46> process    6 is running
<system time   47> process    6 is running
<system time   48> process    6 is running
<system time   49> process    2 is running
<system time   50> process    2 is running
<system time   51> process    2 is running
<system time   52> process    2 is running
<system time   53> process    2 is running
<system time   54> process    2 is running
<system time   55> process    2 is running
<system time   56> process    2 is finished....
<system time   56> process    6 is running
<system time   57> process    6 is running
<system time   58> process    6 is running
<system time   59> process    6 is running
<system time   60> process    6 is running
<system time   61> process    6 is finished....
<system time   61> All processes finished......
============================================================
Average CPU usage:       100.00%
Average waiting time:     25.33
Average response time:     5.00
Average turnaround time:  35.50
============================================================
```

11. 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[]).`
    In the main method, the reader of input produces the list of processes. Then, that list passes to the CPU which begins execution. Once the CPU execution ends, the scheduling criteria calculator computes and prints the criteria averages. Therefore, main method itself should be fairly short.

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

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

14. You cannot use extends in this assignment to extend any class.

15. **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 classes 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 don't 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.

16. 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.

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

18. Do **NOT** type any comments in your program. If you do a good job of programming by following the advice in number 15 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 the **Assignments** tab in the red area on the left side of the webpage.
You will see the **Lottery Scheduling 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. Monday, May 10, 2021.