Introduction:
In this assignment, you need to implement a priority queue `PQueue` class using a max-heap and to implement a `MaxHeap` class using an array. Each node in the max-heap contains a process. The `Process` class implements `Comparable` interface so that the comparison between nodes in max-heap can be made by calling `compareTo` method. To implement the `compareTo` method, process with larger priority level has a higher priority. In case of tie, process with earlier arrival time should be picked.

This assignment simulates the CPU round robin scheduling. The CPU scheduling works as follows.

1. A system defines a unit time (time slice).
2. A priority queue holds the processes waiting for CPU time.
3. Each process has a priority level, time remaining to finish, and arrival time.
4. The system assigns the next CPU time (time slice) to the highest priority process (if there is a tie, the one arriving first is picked) in the priority queue.
5. Each time a process get a time slice from CPU, its remaining time to finish should be decremented by one.
6. In order to avoid starvation problem, the system will increment the priority for lower priority processes. In this simulation, if any process does not get any time slice for a certain amount of time (timeToIncrementPriority), its priority will be incremented by one until it reaches the highest allowable priority level.

Description:
There are 2 java source files in `~jhyeh/cs321/labs/lab2/files/`. You can copy these two files to your own assignment directory. Please do not modify these two files since my test program relies on the print statements in the provided sources codes.

1. `CPUScheduling.java`: simulates the round robin CPU scheduling algorithm. This class is finished.
2. `Averager.java`: keeps track of the number of processes in the simulation and computes the average turn around time. This class is finished.

In addition to these two java classes, you should implement other classes for the CPU scheduling simulation. Suggested other classes includes, but are not limited to

1. `ProcessGenerator.java` randomly generates processes with a given probability. At beginning of each time unit, whether a process will arrive is decided by the given probability. In addition, while generating a new process, both its priority and the required time units to finish the process are randomly generated within given ranges.
2. Process.java defines a process. You need to implement the compareTo method in this class. Each process has a priority level, time remaining to finish, and arrival time.

3. MaxHeap.java defines a max-heap. Each node in the max-heap contains a process.

4. PQueue.java defines a priority queue using a max-heap.

What you need to do:

1. Create a directory ~/cs321/lab2 for this assignment.

2. Copy CPUScheduling.java and Averager.java to your assignment directory from
   ~/jhyeh/cs321/labs/lab2/files/

3. Write those suggested classes and additional classes if any.

4. To run the simulation, the following command-line arguments need to be provided:
   ```
   java CPUScheduling <maxProcessTime> <maxPriorityLevel> <timeToIncrementPriority> <simulationTime> <processArrivalRate>
   ```
   where

   - maxProcessTime: largest possible time units required to finish a process;
   - maxPriorityLevel: highest possible priority in this simulation. That is, a process can have a priority, ranging from 1, 2, ..., maxPriorityLevel.
   - timeToIncrementPriority: if a process didn’t get any CPU time for this timeToIncrementPriority time units, the process's priority will be increased by one.
   - simulationTime: the total time units for the simulation.
   - processArrivalRate: using this rate to decide whether to generate a new process in each time unit.

   The simulation flow chart for each CPU time unit is given in Figure 1 on the last page.

5. A sample output file sample_result of the simulation
   ```
   java CPUScheduling 5 5 5 100 0.4
   ```
   is in the directory
   ```
   ~/jhyeh/cs321/labs/lab2/files/
   ```

6. You can test the correctness of your program using provided testing programs. Please follow the instruction in a README file, located in
   ```
   ~/jhyeh/cs321/labs/lab2/files/test/README
   ```

Submission:
Submit your program(s) from onyx by copying all of your files (.java files only) to an empty directory (with no subdirectories) and typing the following FROM WITHIN this directory:
```
submit jhyeh cs321 p2
```
Start

Create a new job

Assign the current CPU time to the job A:
1. timeNotProcessed ++
2. if timeNotProcessed >= timeToIncrementPriority
   1. timeRemaining --
   2. timeNotProcessed = 0

Update each job in queue in level-order sequence:
1. Update job A status
2. if timeNotProcessed >= timeToIncrementPriority
   timeNotProcessed = 0;
   if (priorityLevel < maxPriorityLevel)
      priorityLevel++;
      MaxHeapifyUp this job;

Extract the highest priority job A from the queue

New Arrival?

Insert the job to the priority queue

Wait until next CPU time unit

Update job A status:
1. timeRemaining --
2. timeNotProcessed = 0

Y

Create a new job including
assign random values to
1. priorityLevel
2. requiredProcessingTime
and set the variables to
1. timeNotProcessed = 0
2. timeRemaining = requiredProcessingTime
3. arrivalTime = current time unit

N

Job A finished?

Insert job A back to priority queue

Y

N