Project 2 – Implementing a priority queue

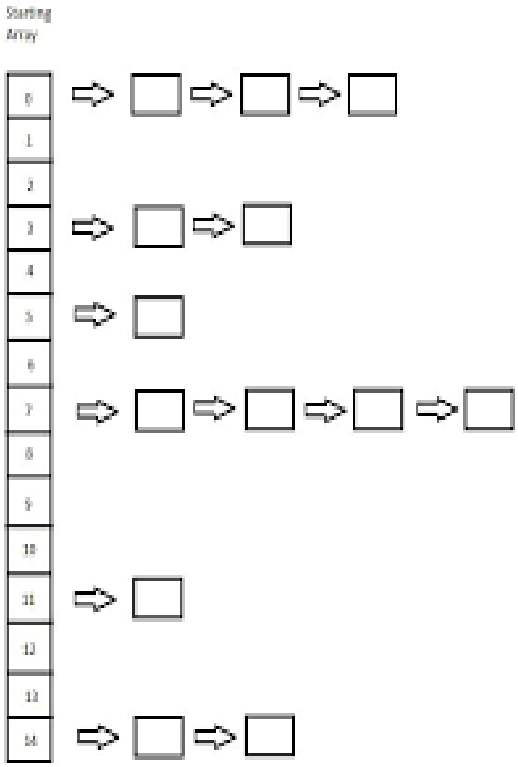
Due: November 6, 2022 11:59 pm **Total point value: 50**

Goals

The goal of this simple project is to get some mild experience with queues. By the end of the project, students will have created a very simple Priority queue system.

Implementation Requirements

The image below shows an array of linked lists that should be used for implementing the priority queue.



Following is a description of **some** specific implementation requirements:

1. The index shows the priority. Lower number has higher priority. 0 has the highest priority.

2. You should implement any class that you need and cannot use the built-in java class like Priority queue or linked list.

3. The information have to be read from a file.

4. The file format is as follow:
    a. In the first line, there is an integer number shows how many process are in the system.

b. The next will have the information of each process including the name of the process, Priority, the start time and the time that process needs to be executed (an imagined processor).

   P0 0 0 3

c. The start time shows when the process insert the system (be inserted in the priority queue)

d. Consider there is just one processor in the system.

e. When the processor is empty, add the next highest priority to the processor from the queue.

f. As soon as a process gets the processor, it will remain there until it is finished.

g. At each period first update the queue then send a process to the processor.

Reaching the rest of the functionality requirements is up to you.

Example

```
5
P0  0  0  3
P1  1  0  4
P2  1  1  1
P3  0  7  2
P4  1  6  2
```

Output

```
T=0
P0 inserts the queue
P1 inserts the queue
P0 starts executing

T=1
P2 inserts the queue

T=3
P0 finish
P1 starts executing

T=6
```

P4 inserts the queue

T=7
P3 inserts the queue
P1 finish
P3 starts executing

T=9
P3 finish
P2 starts executing

T=10
P2 finish
P4 starts executing

T=12
P4 finish

Coding Style

You will be graded on following proper OOP principles and basic coding style. No bad variable names etc.

Comments

Each class and method should have a Javadoc (/** */) style comment, explaining what the class/method does. @param should be used to describe each argument, while @return should be used to describe return types.

Submit: Upload a zip of your .java to brightspace. YOUR CODE MUST COMPILE TO BE GRADED.

Evaluation:

Requirements	Pts	Comment
Correctness	**40/40**	
Code Compiles	10/10	
Queue correctness	20/20	
Output correctness	10/10	
Design	**5/5**	
Documentation and Style	**5/5**	