**SeniorsFirst**

A certain school gives privileges to its students according to their academic class. One privilege is at the cashiers in the dining room, in which students join the queues, but check out according to the priority of their academic class, namely, Seniors first, then Juniors, then Sophomores, and finally Freshmen. If there are two or more Seniors (for example) in the queue, the one who arrived earliest gets served first.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Se | Ju | So | Fr | Fr |  |  |  |
| Ju | Ju | Fr |  |  |  |  |  |
| Se | Se | Ju | So | Ju | Fr |  |  |
| Se | Ju | Ju | Fr |  |  |  |  |

ArrayList of PriorityQueues of different Customers

For each of the four academic classes, we will count the number who checked out, the average wait time, and the longest wait time. The statistics will be printed to the screen and the priority queues will be printed to file.

**Hints on the Assignment**

SeniorsFirst.java is a modification of Cashiers.java in which the ArrayList of Queue of Customers is replaced with an ArrayList of PriorityQueue of Customer.

You will need to modify the Cashier's Customer class so that it works in a priority queue. How?

1. You will need a new field in Customer to store its academic class. (Alternatively, some students make 4 subclasses to model each kind of Customer.)
2. You will have to figure out to store the academic class so that the seniors have the highest priority, then Juniors, then Sophomores, and lastly Freshmen.
3. The Customer class still uses a 1-arg constructor which records the time of arrival. Both the time spent at the cashier and the academic class are set at random.
4. The compareTo method in Customer compares this customer to another and returns the correct signal. Remember that in priority queues, the highest priority returns the lowest number. In this lab, the priority is Seniors first, then Juniors, then Sophomores, and finally Freshmen. If there are two or more Seniors (for example) in the queue, the one who arrived earliest should get served first.
5. The toString method needs to return the time spent at the cashier and the academic class, for example, a sophomore with lots to check out might show as 4-So.

It will be convenient to store the statistics in three parallel arrays of int.

String[] classes = new String[]{"Senior", "Junior", "Sophomor", "Freshman"};  
 int[] served = new int[]{0,0,0,0};  
 int[] longestWait = new int[]{0,0,0,0};  
 int[] totalWait = new int[]{0,0,0,0};

**Sample Run**

Printed to the screen:

Seniors First Simulation!   
How many cashiers? 4  
How long, in minutes, should the simulation run? 60  
Customer Total Longest Average Wait  
Senior 23 10 4.434782608695652  
Junior 18 40 7.666666666666667  
Sophomor 14 28 13.285714285714286  
Freshman 1 2 2.0

< continued on next page>

The priority queues are printed to a file. Here are the priority queues at minute 59 from the run above. Note: printing a PriorityQueue shows you the order in its heap, which is **not** the same as the actual in priority order processing.

minute 59:

2-Se 3-So 3-So 2-So 6-So 2-Fr 5-So 2-So 4-Fr 3-Fr 4-Fr 4-Fr 6-Fr 6-Fr 2-Fr 2-Fr   
2-Se 1-Ju 1-So 5-So 2-So 2-So 4-So 6-Fr 2-Fr 3-Fr 5-Fr 2-Fr 6-Fr 6-Fr 4-Fr 6-Fr   
3-Ju 3-So 4-So 2-So 3-So 3-Fr 5-So 3-Fr 6-Fr 4-Fr 3-Fr 2-Fr 5-Fr 4-Fr 5-So 6-Fr   
1-Se 2-Ju 2-Ju 3-So 4-Ju 5-So 4-Ju 4-So 5-Fr 6-Fr 4-Fr 6-So 3-So 4-So 2-So 5-Fr

How many seniors are in the cashiers' priority queues? Where do they tend to be? How many freshmen? Where do they tend to be?

**Extension**

Make a class MyPriorityQueue which implements a priority queue. It's your choice as to how you will implement the priority queue.

Test your resource class with SeniorsFirst as the driver. You should not have to make any other changes in SeniorsFirst.