CSE 2010, HW4
Due Thu Mar 13 at the start of your lab section; Canvas: HW4
Due Thu Mar 13 at the end of your lab section; Canvas: HW4a

To add an element of surprise/excitement (and perhaps more revenue), an online retail/auction site tries to sell multiple items of the same product at random times over a period of time (e.g. 1,000 items at random times over 24 hours). How would you design an efficient system to match items to bids from customers?

The goal of HW4 is to design a system that can efficiently match items with the highest bidding price at different times. The system allows customers to enter bids. Each bid consists of a price and quantity. For simplicity, each customer can have one bid and there is only one product. If two bids have the same price, the earlier bid has a higher priority (assume the timestamp of a bid is unique). Also, to not lose money, the site does not sell an item if the highest bidding price is lower than the minimum acceptable price, which can be updated over time by the retailer (e.g. higher at the beginning).

To manage and find the highest bid efficiently, use a priority queue implemented with a heap. Each entry of the priority queue has: bid price (key), timestamp (secondary key), and customer name (value). Assume at most 100 entries. A tie in the price is broken by the timestamp. Functions/methods include:

- insert(entry)
- removeMax() // return and remove entry with the maximum key
- getMax() // return entry with the maximum key
- isFull()
- isEmpty()

To implement the priority queue, you may modify/rewrite Code Fragment 9.8 on pp. 377-378 (Programs 9.20 and 9.21 on pp. 352-355 in Standish) We will be evaluating your submission on code01.fit.edu; we strongly recommend you to ensure that your submission functions properly on code01.fit.edu.

Input: The command-line argument for HW4.java is the name of the input file, which has:

- EnterBid *time name price quantity*
- UpdateMinimumAcceptablePrice *time price*
- SellOneItem *time*
- DisplayHighestBid *time*

Time is an integer in HHMM format, where HH is 00-23 and MM is 00-59 (leading zeros are optional). Sample input files are on the course website. You may assume names are unique.

Output: Output goes to the standard output (screen), each line corresponds to an action:

- EnterBid *time name price quantity*
- UpdateMinimumAcceptablePrice *time price*
- SellOneItem *time name price* [NoBids / HighestBiddingPriceIsTooLow]
- DisplayHighestBid *time name bidTime price quantity*

Sample output is on the course website.

Extra Credit (10 pts): Separate submission via HW4Extra.java. Consider the customers are also allowed to update bids. Additional possible input action is:

- UpdateBid *time name price quantity*

and output result is:

- UpdateBid *time name price quantity* [customerNotFound]

Although the priority queue is designed to find the highest bid quickly, it is not designed to find a customer quickly [faster than $O(N)$, where N is the number of customers].

1. Design and implement an additional data structure that can help find a customer and update the priority queue faster than $O(N)$.
2. In the comments at the top of your program (or in a separate PDF file):
   (a) explain why your additional data structure can help UpdateBid become faster than $O(N)$ with an analysis of the time complexity of UpdateBid.
   (b) for UpdateBid, discuss the different cases and how the heap (priority queue) needs to be adjusted.
   (c) explain why EnterBid becomes slower than $O(\log N)$ [It's OK for now; it can be can be remedied with data structures to be discussed later in the course.]

Submission: Submit HW4.java that has the main method and other program files. Submissions for HW4 and HW4a have the same guidelines as HW1.

Note the late penalty on the syllabus if you submit after the due date and time as specified at the top of the assignment.

For extra credit, submit HW4Extra.java that has the main method and other program files. HW4a submission is not applicable to extra credit. Late submission for extra credit is not accepted.