Assignment 5

Objectives:

- To create a binary tree using an array.
- To create a priority queue.
- To apply data abstraction: hiding a complex data type inside a simpler ADT.
- To continue exposure to the principles of inheritance, encapsulation and modularity.
- To continue to practice unit testing.

Introduction

A group of patients are sitting in the Emergency Room waiting room at the local hospital, when the attending physician arrives for her shift. The triage nurse has already assessed patients by their main complaint and provides an electronic device whereby the physician can touch the screen and the next patient chart is provided. The order of the list is determined by the patient's priority. You have been selected to write part of the software that stores the patient information and prioritizes the list for the ER physician. Your boss wants the code to be in the form of a priority queue that uses an array-based heap to store the ER_Patient objects (this class has already been completed).

Preparation

1. Refer to Chapter 12 of the textbook for background on the array-based heap and the PriorityQueue ADT. We will be avoiding the use of generic datatypes in this assignment and referencing the provided ER_Patient class. Note that the Heap is the hidden data type we will use as the main instance variable of the PriorityQueue
2. Download all the necessary java files for this assignment into a folder created for this assignment.
3. All the files will compile; run `javac *.java`. During the process of completing the assignment, check to make sure everything still compiles before continuing.
4. Familiarize yourself with the ER_Patient class. It is the `item` that will be stored in the Heap, so you will need to know how to instantiate it and compare the priorities of two patients. You do not need to understand the implementation, but you are welcome to, for learning purposes. You can easily work with the class by reading the documentation; simply run `javadoc ER_Patient.java NoSuchCategoryException.java` and then open the `index.html` file that was created.

Implementation details

You are to write and test the implementation of the Heap and PriorityQueue classes. The following ordered steps are provided to guide you:

1. Complete the Heap class:
    a. A shell has been provided. Fill in the comments above each of the public methods and write the code that makes these methods work.
    b. You are strongly encouraged to add as many additional instance variables and methods as you need. However, you must not change the provided method headers or the instance variable. There are two reasons for this requirement:
        i. You are a programmer on a team; others are expecting their code to plug into your code.
        ii. You are a student in CSC115, doing an exercise that enhances your skills as a programmer. To obtain adequate feedback, the marker(s) are counting on easy access to your implementation.
    c. Note that the array size is not determined or limited. Somewhere in the source code, there must be a provision for increasing the size of the array to accommodate large numbers of patients. It is not sufficient to assume an upper limit on the number of patients in an ER.; it is also not good programming practice. Create a private method that doubles the size of the array to accommodate more patients into a full array.
    d. You must test all of your public Heap methods in the main method. The more rigorously you test, the more robust your code. Do not move to the next step until you are confident that your Heap works as expected.
        i. Note that the ER_Patient main class has a unit tester. It uses a `Thread.sleep()` call to spread a single second between admit times. You may use this technique or choose to create your own times and insert them into one of the constructors. *For the sake of the marker's sanity, DO NOT use Thread.sleep in any method other than main.*
2. Complete the PriorityQueue class using a Heap data structure as the main instance variable.
    a. Comment each of the methods appropriately
    b. Fill in each of the methods. HINT: Let the hidden Heap do all the work for the PriorityQueue.
        i. Note that the default constructor is complete as is.
    c. Test the PriorityQueue:
        i. If the Heap has been thoroughly tested, it is sufficient to insert a few patients and then dequeue and print until the queue is empty. A sorted print out indicates success.

Some tips and considerations

- As in the previous assignment, there are hidden gems to learn from the professional style of the programming; please take advantage of these, by reading the code thoroughly, until you understand it.
- It is a good idea to create private methods that help you test your code. For instance, a method that prints out the contents of the array in the Heap class is a handy tool to mark the progress of the implementation. Because it is private, only the programmer can use it.

The Heap vs the PriorityQueue

A PriorityQueue can use a linked data structure, an un-ordered array, an ordered array, or a tree as its private data structure. We use the Heap in this exercise, because the Heap data structure is so beautifully similar to a PriorityQueue and takes O(log n) for both `insert` and `remove` operations. However, the Heap is a complete binary tree data structure and is not necessarily bound by the PriorityQueue ADT. For instance, it is allowed to have an iterator and it is allowed to have a size method. There are other tree-related methods such as isInternal that are not part of the PriorityQueue ADT. There is also nothing stopping a Heap from deleting an item in the middle of its structure, although there are better data structures for that kind of operation.

The PriorityQueue ADT is much more limiting. For that reason, you want to make sure that the user does not ever have access to the Heap itself. It is desirable to have the PriorityQueue hide any extra functionality of the Heap.

Submission

Submit your Heap.java and PriorityQueue.java using conneX. **Please be sure you submit your assignment, not just save a draft**.

A reminder that it is OK to talk about your assignment with your classmates, and you are encouraged to design solutions together, but each student must implement their own solution.

Grading

If you submit something that does not compile, you will receive a grade of 0 for the assignment. It is your responsibility to make sure you submit the correct files.

Requirement	Marks
You submit something that compiles	expected
Method commenting style	1
All specifications implemented and correct	7
Thorough unit testing	2