Assignment 8

Objectives

- More practice implementing an interface in Java
- Exposure to the Priority Queue ADT
- Practice implementing the Heap ADT
- Exposure to Comparable<E> interface
- Exposure to inheritance

Introduction

This assignment has two parts:

Part 1 asks you to implement the `PriorityQueue` interface using an array-based Heap data structure that will store `Comparable` objects (objects that implement the Comparable interface). A reference-based List implementation of `PriorityQueue` is provided for you (`LinkedPriorityQueue.java` and `ComparableNode.java`) so you can run the `Part1Tester` and compare running times of the two implementations.

Part 2 asks you to implement a small application modeling a triage center in an emergency room in which you will use your `HeapPriorityQueue`.

Part I

1. Download the files: `A8Tester.java`, `PriorityQueue.java`, `HeapPriorityQueue.java`, `LinkedPriorityQueue.java`, `ComparableNode.java`, `HeapEmptyException.java` and `HeapFullException.java`.
2. Read the comments in `HeapPriorityQueue.java` carefully
3. Compile and run the test program `A8Tester.java` with `LinkedPriorityQueue.java` to understand the behavior of the tester:

```
javac A8Tester.java
java A8Tester linked
```

4. Compile and run the test program `A8Tester.java` with `HeapPriorityQueue.java` and repeat step a and b below

```
javac A8Tester.java
java A8Tester
```

   a. Implement one of the methods in `HeapPriorityQueue.java`
   b. If you see tests 0 through 58 pass you should move on to Part II.

5. Notice the difference in how long the tester takes to run with the linked version versus your heap version!!!

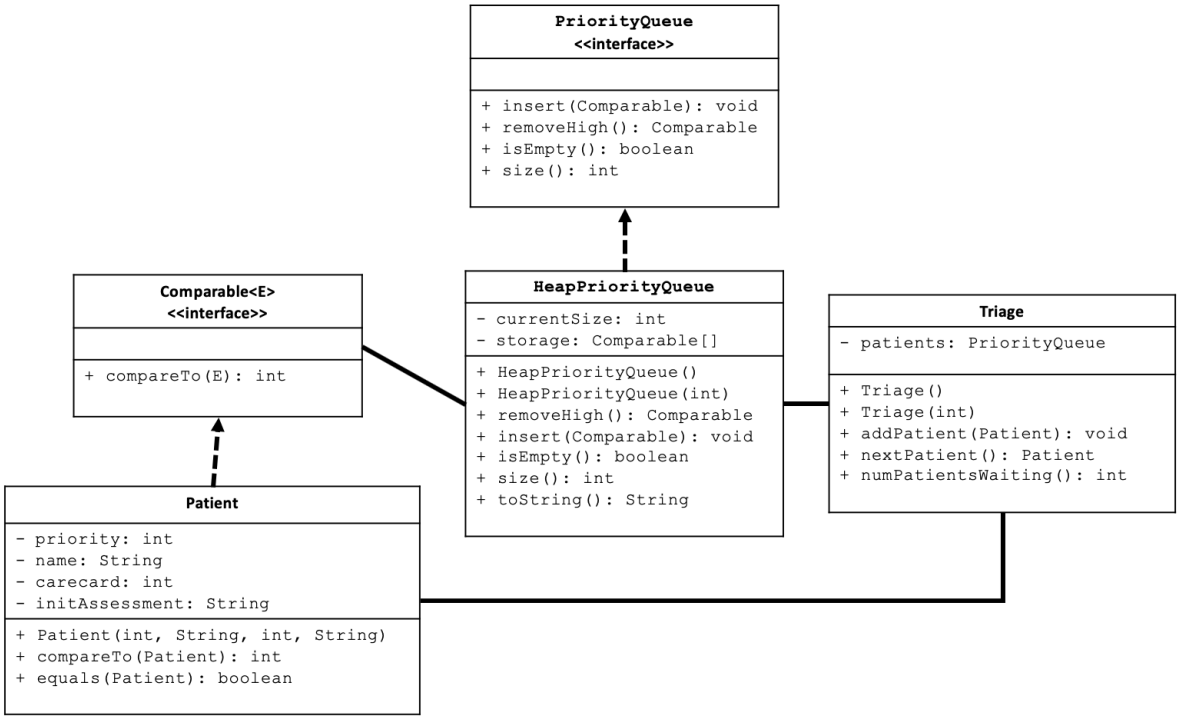
Part II

For this part of the assignment you will be creating an application to support the modelling of a simple triage center in a hospital emergency room. You are asked to write the software that will manage the assignment of patients based on patient assessed priority as doctors become available.

Imagine… you are running a hospital emergency room and patients are coming in continuously. There is a limited number of doctors on staff, for simplicity of explanation let's say there is just one doctor on staff.

- A patient walks in - and he's served immediately.
- Next, a man with the flu comes in and you add him to the priority queue and he waits.
- Next, a woman having signs of a heart attack comes in, you add her to the priority queue with the highest priority.
- Next, a man with a deep cut to his leg comes in, you add him to the priority queue with a high priority.
- The doctor finishes with his current patient and asks for the next patient. The highest priority patient (woman with heart pains) is assigned to the doctor.

The given a tester `A8Tester.java` that will test the functionality of your Triage implementation and mimics a scenario similar to the one the above. The classes involved are represented in the UML diagram below. Read the tester and documentation carefully to help you understand how the classes you will be writing will be used.



CHALLENGE: If you feel like adding more to this application for fun ☺
This is a very naïve implementation of a triage application. Some flaws…

- If the heap gets full, do we turn patients away?
- If two patients with same priority arrive two hours apart, we do not ensure the earlier patient is seen first. What would you change to ensure first come first serve with equal priority?
- If the same patient get a new injury in the waiting room of higher priority, the patient is added to the queue twice – how would you avoid duplication?

Submission

Submit only your `HeapPriorityQueue.java`, `Patient.java` and `Triage.java` and any other files that your classes depend on via conneX.

Please be sure you submit your assignment, not just save a draft. **ALL late** and **incorrect** submissions will be given a **ZERO** grade.

If you submit files that do not compile, or that do not use the correct method names you will receive a **zero grade** for the assignment. It is your responsibility to ensure you follow the specification and submit the correct files.

Your code must **not** be written to specifically pass the test cases in the testers, instead, it must work on all valid inputs. We will change the input values, add extra tests and we will inspect your code for hard-coded solutions.

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. We will use plagiarism detection software on your assignment submissions.