Assignment 8

Reminder: Your code is to be designed and written by only you and not to be shared with anyone else. See the Course Outline for details explaining the policies on Academic Integrity. Submissions that violate the Academic Integrity policy will be forwarded directly to the Computer Science Academic Integrity Committee.
All materials provided to you for this work are copyrighted, these and all solutions you create for this work cannot be shared in any form (digital, printed or otherwise). Any violations of this will be investigated and reported to Academic Integrity.

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<E>` interface). A reference-based implementation of `PriorityQueue` is provided for you (`LinkedPriorityQueue` and `ComparableNode`) so you can run the tester 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`.

Submission and Grading

Submit your `HeapPriorityQueue.java` and `Triage.java` and with the completed methods through the assignment link in BrightSpace.

- You **must** name your methods as specified in the given interface and as used in `A8Tester.java` or you will receive a **zero grade** as the tester will not compile.
- If you chose not to complete some of the methods required, you **must at least provide a stub for the incomplete method** in order for our tester to compile.
- If you submit files that do not compile with our tester (ie. an incorrect filename, missing method, etc) you will receive a **zero grade** for the assignment.
- Your code must **not** be written to specifically pass the test cases in the testers, instead, it must work on other inputs. We may change the input values when we run the tests and we will inspect your code for hard-coded solutions.
- **ALL late** and **incorrect** submissions will be given a **ZERO** grade.

Getting Started

Part I

1. Download all java files provided in the Assignment folder on BrightSpace.
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. One at a time, implement 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!!!
   If you have not implemented the `insert` and `removeHigh` methods correctly to take advantage of the O(log n) algorithms they will be very slow and our grader will not be able to grade them.

NOTE: you will see and can ignore compiler warnings on calls to the `compareTo` method that are made in `LinkedPriorityQueue` and `HeapPriorityQueue`.
This is explained in more detail in the documentation at the top of `PriorityQueue.java`

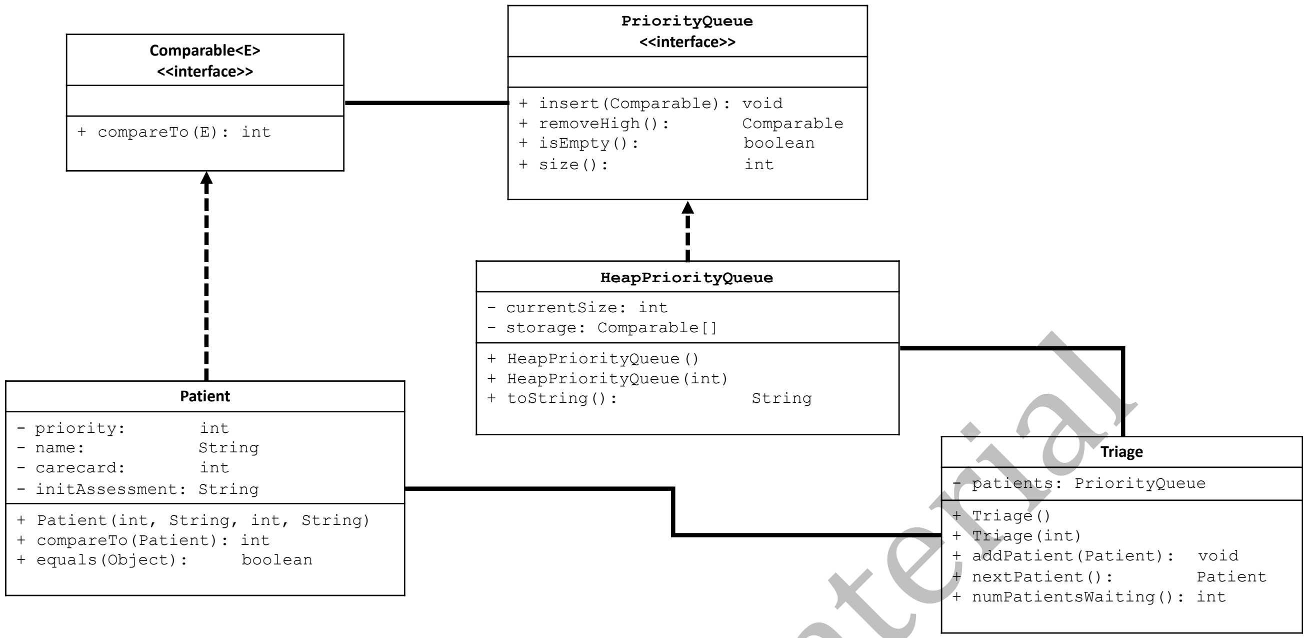
Part II

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 she is 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 gets a new injury in the waiting room of higher priority, the patient is added to the queue twice – how would you avoid duplication?