Assignment 5

Objectives

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

Introduction

This assignment has two parts:

Part 1 asks you to implement the `PriorityQueue` interface using a Heap data structure that will store `Comparable` objects (objects that implement the Comparable interface). A `LinkedList` implementation is provided for you 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: `Part1Tester.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 `Part1Tester.java` with `LinkedPriorityQueue.java` to understand the behavior of the tester:
   ```
   javac Part1Tester.java
   java Part1Tester linked
   ```
4. Compile and run the test program `Part1Tester.java` with `HeapPriorityQueue.java` and repeat step a and b below
   ```
   javac Part1Tester.java
   java Part1Tester
   ```
   a. If no errors reported by test program, see the grading section of this document
   b. Implement one of the methods in HeapPriorityQueue.java
5. Notice the difference in how long the tester takes to run with the linked version versus your heap version!!!

Part II

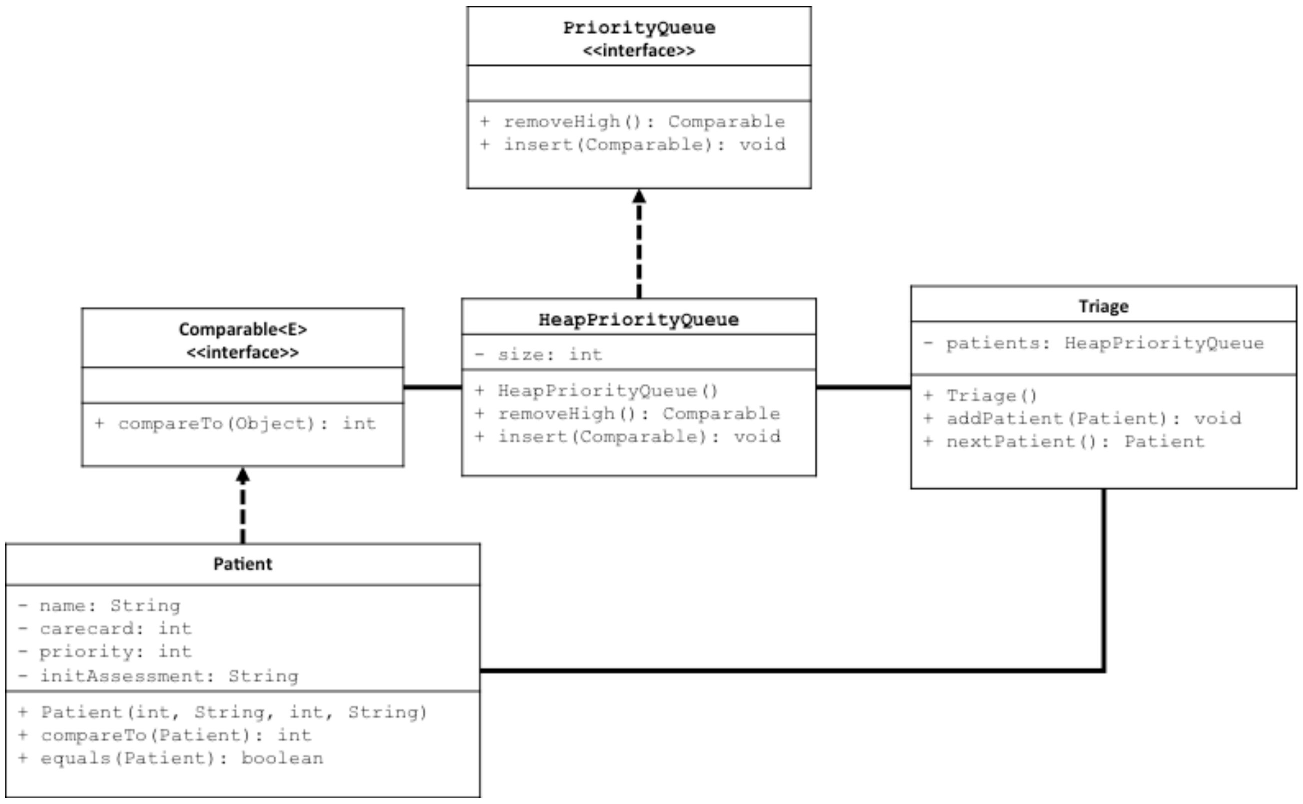
For this part of the assignment you will be creating an application to support the modelling of a simple triage center in a hospital emergency room. You are asked to write the software that will manage the assignment of patients based on patient assessed priority as doctors become available.

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.

You are given a tester `Part2Tester.java` that will test the functionality of your Triage implementation and mimics a scenario similar to the one the above. Read the tester carefully to help you understand how the classes you will be writing will be used.

In order for the `Part2Tester.java` to compile you must provide the implementation for each of the classes described in the UML diagram below. You have the implementation of `PriorityQueue.java` and `HeapPriorityQueue.java` from Part 1, so you will need to implement `Patient.java` and `Triage.java` as described below. You can naively compare Patients on their priority alone. **REMINDER:** It is good practice and may save you time to create your class with empty stubs for methods so that the tester will compile and THEN add code to implement each method one at a time.



PriorityQueue
<<interface>>
+ removeHigh(): Comparable
+ insert(Comparable): void
Comparable<E>
<<interface>>
+ compareTo(Object): int
HeapPriorityQueue
- size: int
+ HeapPriorityQueue()
+ removeHigh(): Comparable
+ insert(Comparable): void
Triage
- patients: HeapPriorityQueue
+ Triage()
+ addPatient(Patient): void
+ nextPatient(): Patient
Patient
- name: String
- carecard: int
- priority: int
- initAssessment: String
+ Patient(int, String, int, String)
+ compareTo(Patient): int
+ equals(Patient): boolean

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**. Submit as many times as you want, your last submission will be the one graded.

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 be using plagiarism detection software on your assignment submissions.

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**
Passes part 1 tests	55
Passes part 2 tests	15
Total	**70**