Assignment 7 (25 marks)

Focus: *Java Collection Framework*

In this assignment, you are required to write a menu-driven Java program that allows the user to add patients to a priority queue, display the next patient (and remove him/her from the queue), show a list of all patients currently waiting for treatment, and exit the program. The program should simulate the scheduling of patients in a clinic. Use the attached `Patient` class provided along with this assignment.

Your program should schedule patients in the queue according to the emergency of their cases from 1 to 5 (the higher the value, the higher the priority). If two patients have the same emergency value, use their order of arrival to set their priority (the lower the order, the higher the priority). It is up to you to have the `Patient` class implement either `Comparable` or `Comparator`.

Create a class `PatientManager` that has an attribute named `waitingList` of the type `PriorityQueue<Patient>` and a public method, `start()`. When `start` is called, it should display the following menu of choices to the user, and then ask the user to enter a choice from 1 to 4:

```
--------------------------
   (1)  New Patient.
   (2)  Next Patient.
   (3)  Where is Patient in Queue.
   (4)  Exit.
--------------------------
```

Here is a description of each choice:

(1) Ask the user for the patient's name and the emergency from 1 to 10 (1 = low, and 10 = life-and-death). Your program should create an instance of `Patient` using the entered data and add it to the priority queue. Note that your program should not ask the user for the value of the patient's order of arrival. Instead, it should use a counter that is automatically incremented whenever a patient is added to the queue.
(2) Display the name of the next patient in the priority queue and remove him/her from the queue.
(3) Ask the user for the Patient's name and display that Patient's position in the queue.
(4) End the program.

Make sure your `PatientManager` class is robust. It should not crash when a user enters invalid value. Instead, it should display an error message followed by an action depending on the type of error (see the sample run below).

Test your program by instantiating your `PatientManager` class in a `main` method and calling the `start` method.

Note: Add more helper methods and attributes as needed to the `PatientManager` class.

Sample run **(Read carefully to determine the required response in different situations)**

```
-------------------------
   (1)  New Patient.
   (2)  Next Patient.
   (3)  Where is Patient in Queue.
   (4)  Exit.
-------------------------
* Choose an item from the menu: 8
(x) Wrong choice.
* Choose an item from the menu: two
(x) Wrong choice.
* Choose an item from the menu: 2
- No more patients.
* Choose an item from the menu: 3
- No patients in the list.
* Choose an item from the menu: 1
        Enter patient's name: Aballah
        Enter emergency [1 (low) to 10 (life-and-death)]: -1
        (x) Wrong value. Try again: two
        (x) Wrong value. Try again: 3
        Patient added to the waiting list.
* Choose an item from the menu: 1
        Enter patient's name: John
        Enter emergency [1 (low) to 10 (life-and-death)]: 7
        Patient added to the waiting list.
* Choose an item from the menu: 3
        What is the name of the patient? John
        John at position 1 in queue
* Choose an item from the menu: 3
        What is the name of the patient? Sam
        Sam is not in queue
* Choose an item from the menu: 1
        Enter patient's name: Lili
        Enter emergency [1 (low) to 10 (life-and-death)]: 7
        Patient added to the waiting list.
* Choose an item from the menu: 3
        What is the name of the patient? Aballah
        Aballah at position 3 in queue
* Choose an item from the menu: 2
        - John is treated.
* Choose an item from the menu: 2
        - Lili is treated.
* Choose an item from the menu: 2
        - Aballah is treated.
* Choose an item from the menu: 2
- No more patients.
* Choose an item from the menu: 4

Program terminated. Good bye!!
```

Grading

- 15 % for logic explanation
- 70 % for proper code structure and logic
- 15 % for correct syntax and formatting

Submission Instructions

For this assignment, you need to do the following:

1- Create a Java project of which name consists of **your student number followed by the assignment number,** e.g., "1234567_A7".
2- Create one class for each question and write your answer inside that class. Your classes should have the same name as the question number (e.g., Q1)
3- After solving all questions, open Windows Explorer (or any other file explorer).
4- Navigate to your Java project folder (can be found inside your Eclipse workspace folder).
5- Locate the "src" folder for this project (the folder that includes the source code for all questions).
6- Zip the "src" folder and rename the zipped file to match your project name (e.g., 1234567_A2.zip).
7- Submit the zipped file **to Canvas.**

Note that you can resubmit an assignment, but the new submission overwrites the old submission and receives a new timestamp.