COMPX201 & COMPX241 Assignment Four
Priority Queue

Due: Friday 10th June 11:59pm.

Part One:

You must write your own Priority Queue class and all its required operations. Your program should follow the specifications given below.

1. **Priority Queue:** define a class called `PriorityQueue` in a file called `PriorityQueue.java`. This class is to implement a min heap using <u>an array of Nodes</u>. Your solution should support the following methods:
   - `insert(Node n)` - a public method that adds the Node to the Priority Queue. Maintain priority ordering using `upHeap()`.
   - `delete()` - a public method that removes and returns the highest priority Node from the Priority Queue. Maintain priority ordering using `downHeap()`.
   - `upHeap()` - a private method that upheaps the queue to maintain priority order
   - `downHeap()` - a private method that downheaps the queue to maintain priority order
   - `dump()` - a method which prints out the Priority Queue textually as follows.

     Root <p1>; left <p2>; right <p3>;
     Root <p2>, left …
     where p is the priority value (see `Node.toString()`)
2. **The Node:** define a class called `Node` for the nodes in your `PriorityQueue`. Please note, in this case, it must be an external class in a separate file called `Node.java`. It should have the following:
   - A public member variable to hold the priority of the node as an integer.
   - A constructor that takes a priority as an integer, and copies it into the node's member variable.
   - A public `toString()` method which returns the details of a node as a String as follows. *"Priority: 2"*

You may have additional member variables and methods if they are useful to you, but they should be private, except for methods that are used to support part two.

Part Two:

In Part Two, you will gain experience maintaining a Priority Queue of Patients, where a Patient inherits from the Node class. You will implement a Priority Queue of medical patients. To do this you will (1) create a `Patient` class as described below, and (2) create a new class called `MedicalCentre` to process a queue of patients. Please note, you should not need to make any changes to your `PriorityQueue` class to make this work.

The Patient: Create a class called `Patient` for the patients in your `PriorityQueue.` It must be an external class in a separate file called `Patient.java`, which inherits from the `Node` class. It should have the following:

- A public member variable to hold the name of the patient as a String.
- A public member variable to hold the age of the patient as an integer.
- A public member variable to hold the medical severity as a String.
- A public member variable to hold the patient number as an integer.*
- A constructor that takes the patient's priority, name, age, medical severity, and number as arguments and copies the values into the member variables.
- A public `toString()` method which returns the details of a patient as a String as follows. *"Priority: 2, Name: Bob Bobson"*
- A public `toStringLong()` method which returns the full details of a patient as a String as follows. *"Priority: 2, Name: Bob Bobson, Age: 23, Medical severity level: medium, Patient number: 1"*

* The patient number refers to an integer counter that is incremented each time a new patient is created/added into the Priority Queue. I.e. the first patient to be added would hold the integer 1, the second would hold the integer 2, and so on.

MedicalCentre: Create a class called `MedicalCentre` in a file called `MedicalCentre.java`. In this file you are going to write a public method called `processQueue()` which will read a list of patients from the file and use your `PriorityQueue` class to process the queue of patients.

The contents of the input file will be as follows:

- Each line in the file represents the details of an individual patient
- Each line will contain the patient's name, the patient's age, and the severity of their medical ailment, separated by commas
- For example: <patient name>,<patient age>,<medical severity>
- Medical severity can be *low*, *medium*, or *high*

Your program must adhere to the following specifications:

- Your method should read in the file, line by line, and add the patients to your Priority Queue
- The priority of each patient should be calculated as follows.
    - Any patient with a medical severity value of *low* will have priority of 3

    - Any patient with a medical severity value of *medium* who is under the age of 65 will have priority of 2
    - Any patient with a medical severity value of *medium* who is 65 years or older will have priority of 1
    - Any patient with a medical severity value of *high* will have priority of 1
- Once all of the patients have been added to the Priority Queue, your method should remove each patient from the queue in priority order, printing the full details of each patient on a seperate line as you go.

Part Three:

Using JUnit as described in class create two test files, one for each solution to the above parts. You must write two different tests for each of the public `PriorityQueue` operations (at minimum), making sure you think about testing for "edge" cases, like an empty Priority Queue. You also need to write two different tests for your `MedicalCentre` program class.

Assessment:

Completing Part One can earn up to a C+ grade. You must also complete Part Two to earn up to a B+ grade and to be eligible for an A+ you must also complete Part Three. Your solution will be marked on the basis of how well it satisfies the specification, how well-formatted and easy to read your code is and whether each class and public method has at least some comment explaining what it does, what it's for and what any of its arguments are (i.e. documentation).

Your code should compile and run as a console program from the Linux command-line (i.e. no GUI). Students are encouraged to test their code in the lab prior to submitting their solutions.

Submission:

Create an empty directory (i.e. folder) using your student ID number as the directory name. Place copies of your source code in this directory. If you wish to communicate with the marker any additional information then you may include a plain text README file, but nothing else (e.g. no compiled code). Upload this directory through the Moodle submission page for this assignment.