Project 4

Prof. John P. Baugh – CIS 2353 – Oakland Community College – OR

Objectives

- To understand priority queues
- To understand triage in a hospital emergency room

Instructions

For this assignment you are to write a simulator that will simulate a triage situation, taking input from file.

In the file, you will have the following format:

```
FName LName TriageCode
```

Where *FName* is the patient's first name, *LName* is the patient's last name, and *TriageCode* is a two letter code indicating the injury or illness. To determine the meaning of the code or the severity of the illness or injury in terms of degree of pain or threat to survival, use the following chart (**which includes the priority from 1 to 3, 1 being the most urgent**):

Code	Meaning	Priority
AL	Amputated limb or digit	1
HA	Heart attack	1
ST	Stroke	1
BL	Broken leg	2
SF	Serious Fall injury	2
IW	Infected wound (abscess)	2
KS	Kidney stones	2
OT	Other/Unknown	2
HN	Hangnail	3

An actual file might look something like the following:

```
Ilene Dover SF
Sum-Yung Gai BL
Jacques Strap IW
Wilma Fingaheel HN
Seymour Butts AL
Omai Chestowsh HA
Alotta Payne KS
Mike Rotchertz KS
```

- There are a multitude of ways to implement a priority queue (as we've discussed.) This implementation uses 3 regular queues:
    - You should create a class, `TriageSimulator`, which should maintain **three (3)** regular queues (using your own custom implementation of Queue or another that you find or like) – one for each of the 3 priority levels.
- You must implement the following methods:
    - `add(string lineFromFile)` and doesn't return anything
    - `remove()` which returns the name of the next patient to be seen, and removes him/her from the appropriate queue
    - `isEmpty()` which returns a `boolean`, indicating that **all three** queues are empty
- You must create an instance of your `TriageSimulator` and use it from another class, which will contain the `main` method

To ensure understanding, the actual file shown earlier would be read into memory somewhere in a method called by main (or in main itself) and should simply call the `add( )` method on the lines read from file, one at a time. `TriageSimulator` is responsible for parsing and making sense of the names and codes in the file, placing the names (not the conditions) in the appropriate queue.

If `remove()` is called, it should return the next person who will be seen by a doctor or NP and remove that person from the appropriate queue. Note that this is **transparent to the calling method**. In other words, `main()` or anything in that class should not have to worry about which queue names are stored in.

Regardless of the order that the names were added to the queues, if there is anyone in the priority 1 queue, their name comes first even if they were the last person to come into the emergency room. If priority 1 queue is empty, priority queue 2 is checked, and only when it is empty, priority queue 3 will be checked.

Run your program and make sure all the parts work.

Deliverables

- **Create a zip file** of your .java files or charts and answers to questions (in PDF or Word format) and turn in the zip file. Name the zip file “Capstone” and D2L will take care of putting your name in it.

- You will also need **screen shots of your program working**, pasted inside of a PDF or Word (.doc or .docx) document (you can create PDF from Word documents using the Save As… option)

- Also, make sure your name is in comments on **each** Java file that you turn in. For example:

```
// Isabelle Ringing
// CIS 2353
// Winter 2022
// Prof. John P. Baugh
```