Teacher Guide: Hospital Program

Project Goal
The goal of this project is to create a simulation of a hospital database that keeps track of many patients and their information and allows students to diagnose patients based off their symptoms.

Project Overview
This project is meant to be a relaxed project intended to let students review topics they have learned throughout the year, in addition to exposing them to new topics such as HashMaps, file reading, and cosine similarity. The task is to maintain a patient database with a HashMap that allows users to add, delete, or look up patients in the database. They will also have to read in a provided .txt file that contains diseases and their symptoms to load into a disease database. The students will write the main class for user interaction, the DiseaseDatabase class to keep track of all diseases and to calculate probable diseases for patients based off their symptoms, and the PatientProfile class that will represent a patient's profile and store information about them. This project reviews topics such as classes, objects, public vs. private methods, for-loops, substrings, parameters, and more while allowing them to also explore new yet graspable concepts.

Teaching this Project
This is a 3-5 day project. Students can allocate their time as they wish, and should work with a partner(s) or individually. Begin by introducing the project on the first day (15 minutes, using the intro slides), and then let students explore. This project is not intended to be graded on accuracy -- as long as students put in effort into some aspect of the project, they should receive full credit. Complete directions that students can follow are given in the student guide, which should be clear.

Student Guide: Hospital Program

Your goal for this project will be to create a simulation of a hospital database that keeps track of all patients and their information and allows you to diagnose patients based off their symptoms.

Project Guidelines
This project's goal is to allow you the creative freedom to explore any aspect of the project you find interesting. With that being said, you are only required to implement the given methods for a base implementation. However, we encourage you to explore and go beyond the given methods to improve user interaction and customize the program based off of what you find interesting.

Grading
This project should be **stress free**. The goal of this is to let you explore aspects of the project you find interesting. Of course, you should implement at least the basic methods, but after that, you can spend your time on any of the following:

- Research and include more diseases and/or symptoms to the text file of diseases and corresponding symptoms to optimize the accuracy of the cosine similarity algorithm.
- Obtain more information on each patient (gender, height, weight, blood pressure, etc.) and if applicable, implement the information as factors to determine diseases.
- Display more than one possible disease for patients (based off of their probability).

This project will be graded on the effort you put into it. As long as you're working and exploring aspects of the project you're interested in, you'll receive full credit. If you go above and beyond, there might even be an opportunity for extra credit.

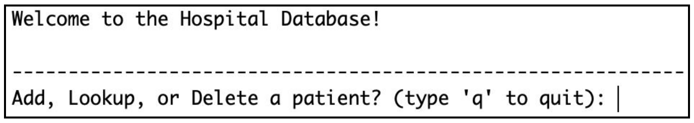
Implementation
Before you start writing code, we recommend that you go through the three classes and the text file to get a better understanding of how they should be implemented and how the files interact with each other.

Although you will write almost all of the implementations for the methods of the classes, we will be giving you some of the starter code to begin with. This will include method headers as well as a few lines of implementation for some of the more difficult methods. However, feel free to (and we encourage you to) create your own private helper methods where needed and declare your own local and/or instance variables.

Going About HospitalProgram.java (main class)
This is the main class where user interaction will occur. The user should be able to add, look up, and delete patients from the hospital's database of patients. The database of patients should be kept track of using a **HashMap**. A HashMap, explained simply, is a data structure that maps keys to values. A HashMap cannot contain duplicate keys; each key can map to at most one value. In our

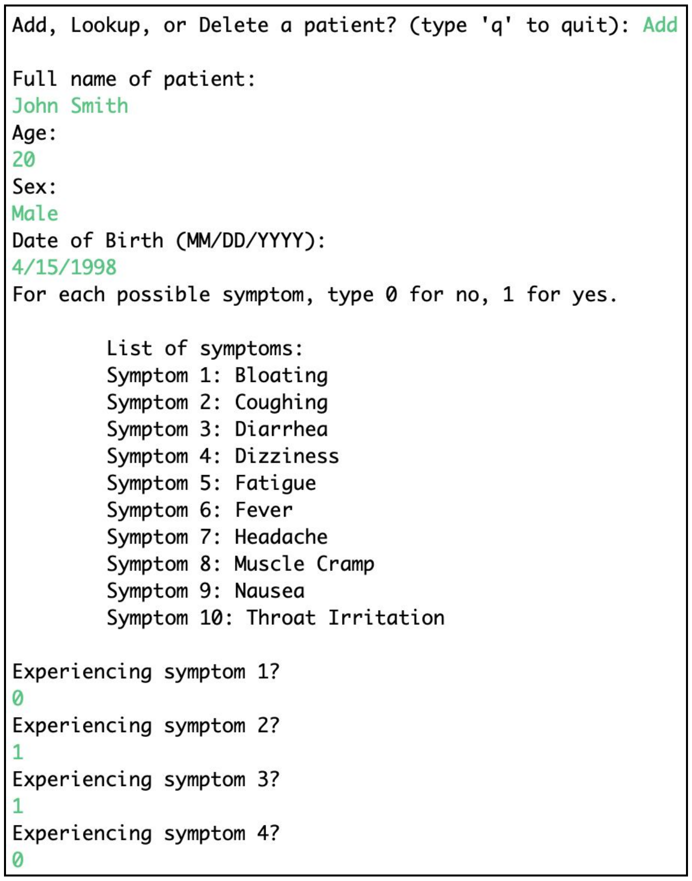
implementation, the key of our patient database HashMap should be the name of the patient, and the value of the key should be a PatientProfile (explained further down below). The documentation for the HashMap class can be found here: https://docs.oracle.com/javase/8/docs/api/java/util/HashMap.html

- public static void main(String args[])
    - This is the main function of the program. This method should prompt the user to either add, delete or lookup a patient in the database and do so accordingly. The user should be allowed to add, delete or lookup a patient as many times as they want until the want to quit the program by pressing 'q' to quit.



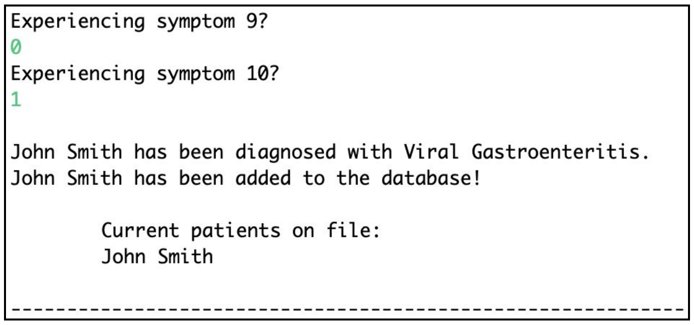
```
Welcome to the Hospital Database!

-------------------------------------------------------------
Add, Lookup, or Delete a patient? (type 'q' to quit): |
```

- private static void add()
    - This function should create a new PatientProfile by prompting the user to input information on the new patient, such as their name, age, sex, date of birth, and symptoms. Symptoms should be inputted as 0s and 1s, as shown below.



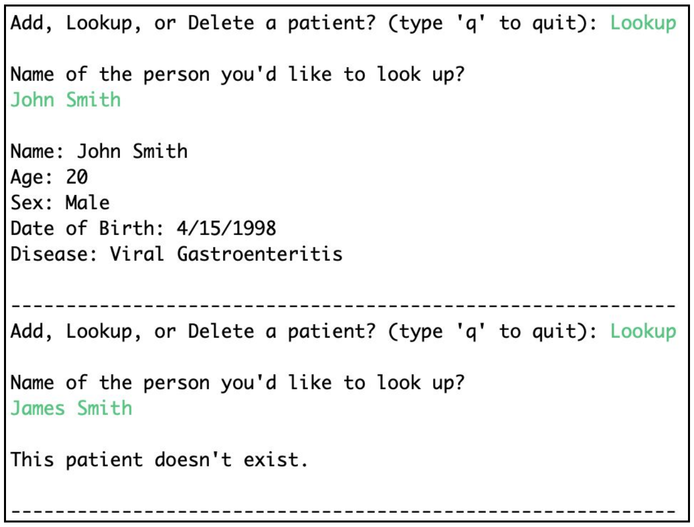
```
Add, Lookup, or Delete a patient? (type 'q' to quit): Add

Full name of patient:
John Smith
Age:
20
Sex:
Male
Date of Birth (MM/DD/YYYY):
4/15/1998
For each possible symptom, type 0 for no, 1 for yes.

        List of symptoms:
        Symptom 1: Bloating
        Symptom 2: Coughing
        Symptom 3: Diarrhea
        Symptom 4: Dizziness
        Symptom 5: Fatigue
        Symptom 6: Fever
        Symptom 7: Headache
        Symptom 8: Muscle Cramp
        Symptom 9: Nausea
        Symptom 10: Throat Irritation

Experiencing symptom 1?
0
Experiencing symptom 2?
1
Experiencing symptom 3?
1
Experiencing symptom 4?
0
```

    - After adding the patient, the function should print out what they have been diagnosed with based on their symptoms (a method implemented later in a different class) as well as the names of the patients currently in the database.



```
Experiencing symptom 9?
0
Experiencing symptom 10?
1

John Smith has been diagnosed with Viral Gastroenteritis.
John Smith has been added to the database!

        Current patients on file:
        John Smith

------------------------------------------------------------
```

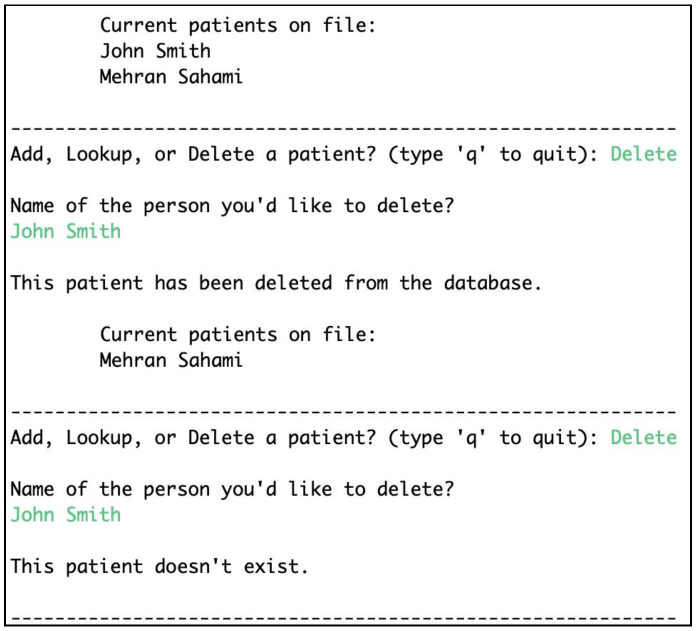
- private static void lookUp()
    - This function should prompt the user for the name of the patient they would like to look up and print out that patient's information. Print an error message if no such patient exists.



```
Add, Lookup, or Delete a patient? (type 'q' to quit): Lookup

Name of the person you'd like to look up?
John Smith

Name: John Smith
Age: 20
Sex: Male
Date of Birth: 4/15/1998
Disease: Viral Gastroenteritis

------------------------------------------------------------
Add, Lookup, or Delete a patient? (type 'q' to quit): Lookup

Name of the person you'd like to look up?
James Smith

This patient doesn't exist.

------------------------------------------------------------
```

- private static void delete()
    - This function should prompt the user for the name of the patient they would like to delete from the database. The function should then print out the updated names of the patients in the database. Print an error message if no such patient exists.

      (Picture on next page)



```
        Current patients on file:
        John Smith
        Mehran Sahami

------------------------------------------------------------
Add, Lookup, or Delete a patient? (type 'q' to quit): Delete

Name of the person you'd like to delete?
John Smith

This patient has been deleted from the database.

        Current patients on file:
        Mehran Sahami

------------------------------------------------------------
Add, Lookup, or Delete a patient? (type 'q' to quit): Delete

Name of the person you'd like to delete?
John Smith

This patient doesn't exist.

------------------------------------------------------------
```

Going About DiseaseDatabase.java (object class)
This is an object class that represents a database of diseases and their symptoms. The main class should be able to utilize this object to determine the disease of a patient.

- public DiseaseDatabase(String filename)
  - This is the constructor that reads in the disease_list.txt file and puts the diseases and corresponding symptoms into another hashmap. The file reading method is given to you, but it is your job to parse through each line and put the name of the disease as the key and its symptoms (as an array of 0s and 1s) as its value.
- public String calcDisease(PatientProfile patient)
  - This method takes in a patient as its parameter and calculates the disease that he/she is most likely to have based on their symptoms and comparing them to the symptoms in the hashmap. This is calculated by the **cosine similarity algorithm**, which is explained in the method below.
- private double cosineSim(int[] list1, int[] list2)
  - This is the implementation of the cosine similarity algorithm between two arrays of 0s and 1s. Cosine similarity is a way to determine how similar two arrays or sets of data are, and below is the formula, with A and B representing the two arrays:

$$similarity(A,B) = \frac{A \cdot B}{\|A\| \times \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}}$$

The symbols of the formula may look complicated if you haven't seen them before. The numerator is the sum of the products of the corresponding entries of the two arrays, and the denominator is the multiplication of the magnitudes (the square root of the sum of each entry squared) of the two arrays. This formula will output a value from -1 and 1, where -1 is perfectly dissimilar and 1 is perfectly similar.

To you are confused and/or you want to further familiarize yourself with it, visit: https://neo4j.com/docs/graph-algorithms/current/algorithms/similarity-cosine/.

Going About PatientProfile.java (object class)

This is an object class that represents a patient's profile. This class should store the necessary information for each patient and allow the main class to access this information.

- public PatientProfile(String name, int age, String sex, String dob, int[] symptoms)
    - This is the constructor that sets all variables to its initial declaration.
- public int[] getSymptoms() and public String getDisease()
    - These are getter methods for the symptoms and disease of the patient.
- public void setDisease(String disease)
    - This is the setter method for the patient's disease, which should change their disease to the one that is passed in as a parameter.
- public String toString()
    - This method should print out the name, age, sex, date of birth, and disease of the patient in a organized, readable manner.

Going About disease_list.txt (text file)

This is the text file containing the list of diseases and their corresponding symptoms.

- Observe the text file carefully. Take note of where certain indexes of characters are and how the file is formatted in terms of the name of the disease and which symptoms it has. To reiterate, each index of the list of 0s and 1s next to each disease represents a specific symptom. A 0 means the disease doesn't have that symptom, and 1 means that it does.