CSE 2102: Introduction to Software Engineering
Project Assignment #1
Assigned: March 29, 2022, Due: April 29, 2022
Demos: April 26, 2022 & April 27, 2022

Notes:

1. This assignment consists of two parts.
2. Assignment #2a -- Implementation of the MIS, and Assignment #2b – Class Diagrams.
3. Each assignment is worth 100 points.
4. Iteration between these two pieces may be necessary; that is, design of the MIS via class diagrams may inform the implementation and vice versa.
5. Both pieces are due on April 29, 2022.
6. Only one submission per team is required.
7. This assignment will culminate with demos on either April 26, 2022 or April 27, 2022. Demos will be conducted by the TAs, and a sign-up sheet will be circulated about 10 days prior.

Extra Credit

A software system can be successfully implemented if it is approached systematically and steadily on a day to day basis, rather than doing it all at once at the last minute. To encourage you to make systematic progress, we are awarding extra credit to each team to report weekly progress to your TA during the lab hours. A short description, like a "Scrum" for 1-2 minutes that shares your progress with the TA of your group will suffice. If the members of your group belong to different lab sections, you can go to either one. There will be three opportunities for extra credit. Each week, you can earn 2 extra credit points for sharing substantive progress (points will not be awarded for merely showing up or dropping into the lab). Thus, each student can earn anywhere between 2 to 6 extra credit points. Each student must come by (that is, both members of the team must drop in, points will not be awarded by proxy). Extra credit points earned for this assignment will offset the points lost for this assignment only. They cannot be used to offset points in any other component of the course.

Assignment #2a

The first part of the assignment explores a segment of the implementation of the MIS. It focuses on establishing a repository of patient profiles that can be utilized in many different ways within the MIS. You are expected to implement four classes as follows:

The `Patient` class is a data manager class that keeps track of the different characteristics of a patient's profile. The fields in this class include:

- Last name (String)
- First name (String)
- Address (String)
- Phone number (String)
- Date of Birth (String)
- Insurance Type (String) – but can take only two values "Private" or "Government" and must be an enumerated type.
- Co-pay (float) – co-pay associated with a patient's insurance plan.
- Patient Type (String) – but can take only three values "Pediatric", "Adult" or "Geriatric" and must be an enumerated type.

The patient class contains a `MedicalConditions` class that keeps track of the special medical conditions for a patient, i.e. allergies, life threatening or chronic medical conditions and physician contact information. The fields in the `MedicalConditions` class include:

- Name of the physician (String)
- Contact phone number of the physician (String)
- Allergies (String) – but can take one of five values "Food", "Medication", "Seasonal", "None", "Other". Must be an enumerated type.
- Illnesses (String) – but can take one of five values "Diabetes", "CHD", "Asthma", "None", "Other". Must be an enumerated type.

The `Patient` class implements three types of methods. The first method is the constructor that is used to create instances of the patient profiles. There will only be one constructor which will be given values of all the variables in the `Patient` and `MedicalConditions` classes. There will be no other constructors that accept values for a subset of the variables, and then initialize the other variables to their default values. In addition to the constructor, the second category of methods is used to obtain the current values of the instance variables. The names of these methods start with the word "get" followed by the name of the attribute whose value it is supposed to return. For example, the method `getFirstName()` returns the first name of the patient. The third category consists of methods to update the values stored in the various instance variables. The names of these methods follow a standard convention with the word "update" followed by the name of the attribute that it is supposed to update. For example, the method `updateFirstName()` updates the first name of the patient. Similar to the `Patient` class, the `MedicalConditions` class has a constructor, get and update methods. The `Patient` class should also include a method to print the contents of all the variables for one patient at once.

The `PatientDatabase` is a data container class that maintains a collection of `Patient` profiles for all patients that have used MIS in the past or present. A simple data structure such as an array (of size 100) will suffice for this project. The implementation of the data container class should include variables that keep track of the total number of patients. It should allow iterative searches of the database, and it should allow a search of the database to compile and print summary reports. The database should include a constructor method which accepts the name of the file from which the patient profiles will be read. This file name will be provided to the system at startup via its interface. This is the same file in which the patient profiles will be stored upon exiting. Thus, for the MIS system in its current version, the database of patient profiles is implemented using a file. The database container class will offer the following functions with respect to each individual profile:

- Insert a new profile.
- Delete an existing profile.
- Update an existing profile.
- Find and display an existing profile.

For the purposes of this assignment, you can assume that the combination of the last name and date of birth uniquely identifies a patient. Thus, the delete, update and display functions will accept the last name and date of birth as input. The date of birth of a patient is not available for an update, which means it cannot be modified.

In addition to the above actions for the individual patient profiles, additional functions allow summary reports to be compiled. These reports can be generated for each of the following fields individually – physician, patient type, insurance type, allergies, and illnesses. For this assignment, you need not consider a combination of these fields. Names and phone numbers of the patients that match the search criteria are displayed. One could envision many uses of this feature, for example, we may need to reach out to the patients of a given physician to reschedule all the appointments.

The last class is the `PatientProfileInterface` class. It encapsulates the functionality for a menu-driven interface, with each of its methods containing the required code for soliciting input from the user, performing the required action, and displaying the results. This class provides the interface to access the database of patient profiles for queries. The methods defined for `PatientProfileInterface` correspond to the options of a menu-driven user interface.

The different options for the menu-driven interface are as follows:

- **Enter a new patient:** All input data required to create an instance of a patient profile as defined by the `Patient` class are solicited from the user. This method presents the user with a menu to elicit the profile information and in turn invokes another method to obtain the medical information. A message is displayed for the user indicating whether the patient profile was successfully created and inserted.
- **Delete a patient:** This option deletes the Patient when the last name and date of birth is supplied. A message is displayed for the user indicating whether the patient profile was successfully deleted.
- **Find and display a Patient**: This option searches for a specific patient when supplied with a last name and date of birth. The entire profile of the patient is displayed.
- **Modify a patient profile**: This in an option that is utilized to allow a user to modify a select subset of an existing patient. Specifically, only the first name, last name, address, phone number, insurance type, co-pay, and patient type are eligible for modification. For example, when a child turns 18, their status should be changed to adult. Additionally, the user should also be able to modify the name and phone number of the physician, allergies and chronic conditions. This option should present the user with a menu of attributes that can be modified. The user will select which attribute should be modified, and

will then be prompted to enter a new value of the attribute. The entire profile of the patient, with the updated value of the chosen attribute, is displayed.

- **Search the database**: This is an option that is utilized by the user to compile all the patients that meet a specific criteria. Specifically, the user should be able to select patients that see a specific doctor, subscribe to a specific type of insurance, are of a given type, have a specific allergy, or a specific chronic illness. This option should present the user with a menu of five attributes (physician, insurance, type, allergy, illness) that are available for compiling summary reports. The user will select the attribute and will then be prompted to enter the value of the attribute. The function will display the names and phone numbers (not the entire profiles) of the patients that match the search criteria.

The interface class will accept the name of the database file at startup from which the stored patient profiles will be read. The interface will prompt the user for a new menu option in a loop. There will be an explicit menu option that will allow the user to exit. Before exiting, the patient data will be written to the database file that was provided at startup. You must implement the GUI using the Swing package in Java.

Important Notes

- Note that you are charged with deciding, as part of your detailed design effort, any implementation details that have not been explicitly provided. You should also provide a justification regarding your implementation choices.
- The access privileges of the methods have not been specified. Note that all methods do not need to be `public`. As a part of the detailed design effort, you are required to identify the access privileges of the methods as well as any other methods that you might write.
- Please document your code adequately. Remember good documentation is absolutely essential for understanding the rationale behind the design decisions embedded in the code and the code itself.
- Please keep an accurate log of the time you spent in thinking, implementing and testing each class, and then testing the integration of the classes.
- In order to test the integration of the classes very little additional test code should be necessary.
- It is strongly suggested that to promote incremental development you work on the different menu options one at a time, designing, implementing, and testing each option before going on to the next one.
- Upon completion of this project, each group will demonstrate their code in the lab. It is imperative that the projects run to your expectation during the demo.

Deliverables

The following deliverables must be submitted on HuskyCT by midnight on April 29, 2022.

- Well-documented classes.
- A written document containing the following information related to the group and project:
  - How the work was distributed among individual team members, and the rationale behind this distribution.
  - The total time spent on the implementation and testing of each class.
  - The total time spent on testing the integration of the classes.
  - Difficulties that you encountered and how you tried to mitigate them.
  - What you would do differently if you were asked to do another assignment of this scope, now that you have gained experience in the implementation of these classes.
- A test plan which describes your rationale in the creation of the test cases, the actual test cases, and any additional code that you might have written to test your classes.

Assignment #2b

The purpose of this assignment is to introduce you to the process of designing a software system using the object-oriented paradigm. In this assignment, you will explore classes, relationships among them, and their representation using UML class diagrams. This assignment has 2 tasks.

Task I: Class Diagrams (50 pts.)

Develop between 10 and 12 classes that you feel can be used to best describe MIS (be sure to choose classes in such a way that a meaningful class diagram can be drawn, as you will be asked to do this in the next task). These classes should be different from those included in the Project Assignment #2a. Each class should have between 4 and 6 attributes and between 4 and 6 methods, though more than 6 may be required to accurately model the full functionality of some classes. These classes should be represented as UML class diagrams.

Task II: Inheritance Hierarchies (50 pts.)

In MIS, inheritance hierarchies can be developed for many of its different features. For example, an inheritance hierarchy could be useful in the identification of different types of patients, or personnel (medical vs. non-medical), or distinguishing between the various types of medical personnel (nurses, doctors, lab technicians, nurse practitioners, therapists). They could also be valuable in keeping track of the different types of appointments (routine physicals, sick visits, follow ups, immunization visits, and so on). Another possibility is to keep track of the different types of medical facilities.

In this assignment, you will build two such inheritance hierarchies (both hierarchies cannot be about the personnel) You must have between 8-10 classes in your final hierarchy, and each should be at least 2 levels deep (3 including the parent class). For each diagram (2 total), you should clearly define what each class in your solution represents by providing a one sentence description. For each inheritance hierarchy, clearly describe the rationale behind organizing the classes in that hierarchy.

Deliverables

The following deliverables must be submitted on HuskyCT by midnight on April 29, 2022.

1. 10 Class diagrams, one for each class that is necessary for MIS to be implemented.
2. 2 inheritance hierarchies.