Assignment 01

1. Write a console-based Hospital Management System. Note that you can use this assignment to demonstrate your concepts of object-oriented programming including classes, objects, composition, inheritance. Major Classes are **Patient:** PatientID, name, address, age, contactNo, date of arrival, **Doctor:** name, empID, address, contactNo, date of birth, date of hiring, specialization, **CheckUp** has patient and doctor details, date and time of checkup, and details of the clinical test performed**, ClinicalTest** has the name of the test and an isPositive attribute which can either be true or false. You can add more classes as you deem appropriate. For example, you can add a super class where you think some of the classes have shared details, and subclasses where a type of a class needs to have more specific details, i.e., InPatient and OutPatients. Think about what instance variables you should define. Perform exception handling and data validation where necessary. You can consider the following details as hints while writing your code.

2. Start by drawing a UML diagram showing the names, instance variables, methods and relationships among the classes. You can use any open source tool to draw the UML diagram.

3. Use proper data types for the instance variables you define. Validate your data where required. The date and time should have user defined data types with a uniform format across the system. The system should use a valid date where the day cannot be less than 1 and does not exceed the maximum number of days in the given month, and the month must be greater than 0 but not greater than 12. You can use 12- or 24-hours format for the time, but the values need to be within the specified limits. For all the classes, printing an object of the class should display the details of that class in a string format.

4. Identify the attributes that can be combined into a common super class, where possible. Write the code for the super class you have identified. For all the classes, use constructors to initialize the instance variables.

5. Write Java code for the doctor class.

6. Implement the patient class. A patient can either be InPatient or OutPatient. An inpatient goes through an OPD while an outpatient is allocated a ward bed or a room.

7. Write code for the ClinicalTest class. By default, a test result is negative.

8. Wrtie Java code for the CheckUp class. Printing an object the CheckUp class should display all the details about the patient, the doctor who performed the check up, the date and time of the check up, and the detail of the clinical test that was performed.

9. Write a client/test class that declares and prints an object of the CheckUp class.

10. Implement a **Hospital** class. A hospital object keeps track of the following information: hospital's name, hospital's phone number, and the doctors. All the instance variables should be private. *What type should you use for the doctors' collection?*

a. Write a constructor for the Hospital class. The constructor takes as parameter the name and the phone number of the hospital. By default, a newly created hospital does not have any doctors (i.e. the collection is empty).

b. Write a **get** method for the name and phone number field. Write a **set** method for the phone number. The name of the hospital cannot be changed once it has been created.

c. Write an **addDoctor** method that receives a Doctor and adds it to the collection.

d. Write a **toString()** method. How can use make use of the toString() method of Doctor when writing the toString() of Hospital?

11. Modify the main method to create a Hospital object. Associate doctors with hospital. Print hospital using **System.out.println()**.

12. The system should also keep track of the number of patients, and the number of doctors.

Deadline: June 23, 2021.