Middle East Technical University
Northern Cyprus Campus

CNG 443: Introduction to Object-Oriented Programming Languages and Systems

Assignment 2: Hospital Management Application – Restructuring

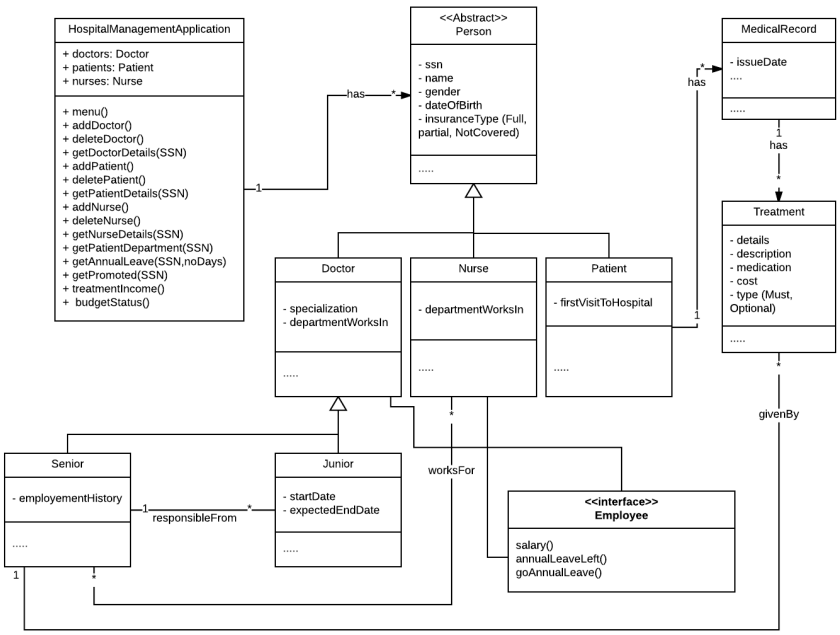
Learning Outcomes

On successful completion of this assignment, a student will:

- Have used an UML class diagram to implement an application.
- Have practiced class hierarchy and the relevant design and implementation decisions.
- Have learnt how to maintain different types of objects.
- Have practiced and used abstract classes, and interfaces.
- Have learnt how to create a package for an application.
- Have learnt how to draw a UML sequence diagram.

Requirements

This assignment is about creating an application to manage a Hospital. This assignment builds on top of the previous assignment (it provides a redesign of the previous version). In overall, the idea is similar to the previous assignment but the class diagram has been significantly changed. The figure below shows a summary class diagram for this application.



The overall requirements are based on this class diagram which are also summarised below:

- A main application called HospitalManagementApplication will be used to maintain information about this hospital. This class will be the entry point to the application and will provide the overall interaction with the application. Since we still haven't yet covered Graphical User Interfaces (GUI) in this course, you **have to** implement this application as a command line application. We also haven't covered how to make objects persistent or how to use and access a database. You **have to** maintain your objects within the application. For further discussion about this, see "Extra requirements" section below. The required methods in HospitalManagementApplication are as follows:

- menu(): this will maintain the menu that will be provided for the interaction with this application.
- addDoctor()/addPatient()/addNurse(): these will add a new doctor, patient or nurse to the application respectively with their given details.
- deleteDoctor()/deletePatient()/deleteNurse(): these will delete a new doctor, patient or nurse to the application respectively with their given details.
- getDoctorDetails(SSN)/getPatientDetails(SSN)/getNurseDetails(SSN): these will display the details of a doctor given their SSN number. Please note that you have to also show their current annual leave days left.
- getPromoted(SSN): This method will convert a junior doctor to a senior doctor. Given the SSN number this has to check if that SSN is a doctor and if it is a junior doctor then this needs to convert that doctor from a junior to a senior status.
- getPatientDept(SSN): This will return the department that a given patient received the latest treatment in.
- getAnnualLeave(SSN,noDays): Employees (doctors and nurses) in this hospital have certain number of days of annual leave. Doctors can have 25 working days and Nurses can have 20 working days. With this, when employees use their annual leaves this will ensure that they are recorded on the system accordingly.
- treatmentIncome(): This hospital works with a specific insurance system. In that system, if the patient can have three types of insurance: full, partial and not covered. If they have full insurance, that means all their treatments are free. If they have partial insurance that means only their "must" treatments are covered. If they have not covered insurance that means their none of their treatments are covered. Based on these insurance system, with this method you need to compute the total income that this hospital had from all the treatments.
- budgetStatus(): Based on the treatmentIncome() you computed then you also need to calculate the total salaries that you have to give and find out the budget status of this hospital.

- In this application, you have Junior and Senior Doctor that are subclasses of Doctors. This will affect their salaries and whether they can give treatment to a patient. Please note that in this hospital, only a senior doctor can give a treatment and a senior doctor would be responsible from a number of other junior students. In this hospital, nurses are also associated with a senior doctor. However, they can be associated with more than one doctor.
- In this hospital application, you keep both patient and employee details. Therefore, to differentiate them you need to have an interface called Employee and your doctor and patient class needs to implement this interface. This will allow you to keep track of salaries and also the annual leave information of your employees.

Extra Requirements:

Some extra requirements are listed below:

- In this course we haven't yet covered how to use a Database or how to make objects persistent, therefore for this application, you need to create a **test class** that populates your application at the beginning. For initial set up of the application, you need to have at least five types of each person. For each patient, you also need to make sure that they visited your hospital at least two times each.
- In this course, we haven't yet covered Graphical User Interfaces (GUI), so please provide a command line interaction.
- For each class, please decide what kind of constructors is required, what would be the access types of methods and fields. For each class, make sure that you have at least two different constructors. Unless it's necessary, do not use public fields. When you use private fields, make sure that you provide accessor and mutator methods.
- Pay attention to the overall design, layout and presentation of your code.
- Make sure that all your methods and fields are commented properly including JavaDoc commands.

- Package all classes into a package called Hospital.jar. With this package one should be able to run your application by just typing "java –jar Hospital.jar".
- Additionally, provide a **UML sequence diagram** for budgetStatus() in your application and also discuss if it is possible to simplify this sequence diagram by making some changes to the given class diagram.

Assessment Criteria

This assignment will be marked as follows:

Aspect	**Marks (Total 100)**
All classes are implemented	10
All class hierarchies are implemented	10
All interfaces are implemented and used	10
For all classes constructors are properly implemented	10
For all classes all required data fields are implemented	10
For all classes all required methods are implemented	10
15 methods	30
Package Structure and Jar for Invoking the application	5
UML Sequence diagram	5

Grading Policy

In order to get full mark, your class should have the following: a constructor with full parameters, at least two constructors with partial parameters, overridden toString method, javadoc. The following grading scheme will also be used for the requested methods.

Fully working	0.2
Appropriate reuse of other code	0.2
Good coding style	0.2
Good Javadoc comments	0.2
Good and neat test results	0.2

Deadline: 30 November, 2017, Thursday.