Middle East Technical University
Northern Cyprus Campus

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

**Assignment 1: Hospital Management Application**

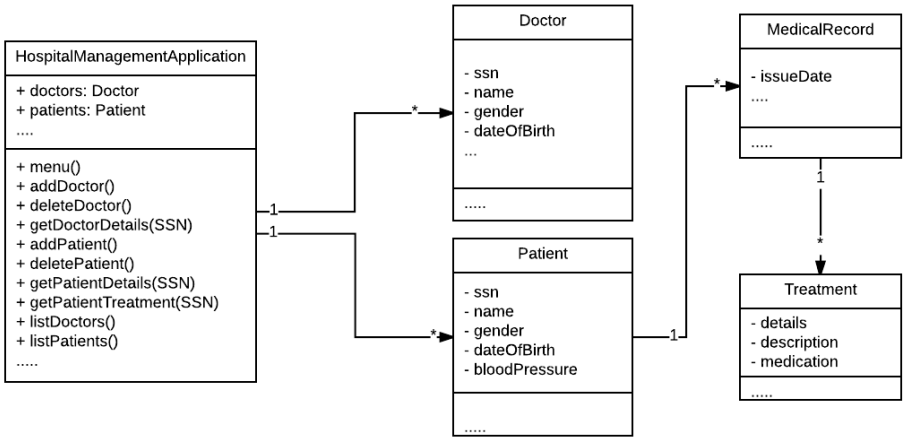
## Learning Outcomes

On successful completion of this assignment, a student will:

- Have written a class suitable for instantiation.
- Have written a Javadoc comments for it.
- Have written code which creates and manipulates instances of this class.
- Have begun to appreciate the usefulness of reusing code, even within one class.
- Have developed a class based on use of an array as a means of storing a collection of objects.
- Have designed and written reusable methods for adding, searching, for and deleting objects to/in/from the collection.
- Have defined and implemented appropriate test program to check the operation of the collection class.
- Have used a UML class diagram to implement an application.

## Requirements

This assignment is about creating a small Java application to manage a small private hospital. It aims to maintain information about patients, doctors and treatments given in this hospital. 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 the doctors and patients in that hospital. HospitalManagementApplication will also have the main method and will provide the overall interaction with the application. Therefore, this class should include the main method where an instance of this class is constructed and the menu of commands is displayed to the user. Since we haven't yet covered Graphical User Interfaces (GUI) in this course, you need to implement this application as a command line application. The required methods are as follows:
    - Create a new doctor.
    - Delete a doctor.
    - Retrieve details of a doctor for a given Social Security Number.
    - Create a new patient.
    - Delete a patient.
    - Retrieve details of a patient for a given Social Security Number.
    - Retrieve the latest medical record for the given patient Social Security Number (SSN).
    - List all doctors.
    - List all patients.
    - Retrieve all the treatments given in the hospital.
- All classes are given with a number of fields but you can add extra fields or methods to perform the requested functionalities above. You also have to make decisions about what kind of primitive types will be used to store these data fields.
- Every time a patient visits the hospital, a new medical record is created to record information about the treatment given at that visit. Therefore, a patient will have a number of MedicalRecords.
- A MedicalRecord holds information about the given treatment. However, treatment information is stored in the Treatment class. Every treatment is always given by one doctor. This is one of the management policies of this hospital.
- Test Class: Since you did not learn how to make your class persistent or use a database, everytime you run your application you will lose data. Therefore, you need to create a class that can be used to populate your application with some mock doctor, patient and treatment data.
- Once you complete your implementation, fully update the UML class diagram and submit it as well. Original UML diagram was created with LucidChart (<https://www.lucidchart.com/>). You can use that or any other tool to create your updated UML diagram.

**Environment**: As a development environment, you can use any IDE you like but you are strongly recommended to use Eclipse (http://www.eclipse.org/).

**Submission**: Store all Java classes in the same folder and if you are using Eclipse make sure that your project is stored in that folder as well. You need to ZIP all the developed code and submit this ZIP file to ODTU Class.

## 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 exercise maintain objects such as doctors and patients in an array. You can use an ArrayList.
- 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. If 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".
- In this course, we haven't yet covered how to make objects persistent so every time you run your application you will lose information about your patients and doctors. Therefore, make sure that you have created a test application that populates your application with some data so that we can test your application.

## Assessment Criteria

This assignment will be marked as follows:

| **Aspect** | **Marks (Total 100)** |
|---|---|
| All classes are implemented | 25 |
| For all classes constructors are properly implemented | 10 |
| All methods are implemented | 25 |
| Command line interaction and menu | 10 |
| Package | 10 |
| Test class | 10 |
| UML Class diagram | 10 |
| **Submitting via GIT (BONUS)**[1] | **5** |

| | |
|---|---|
| Half working | 2 |
| Fully working | 2 |
| Appropriate reuse of other code | 2 |
| Good Javadoc comments | 2 |
| Good and neat test results | 2 |

## Deadline: 8 November 2017, Wednesday.

[1] Further details about this will be given to you later.