The hospital management system consists of four different roles. Each role has their own responsibilities and limitations and such roles are Patients, Specialists, Administrators and Front Desk. The scope of those roles is;

|  |  |
| --- | --- |
| **Role** | **Area of Responsibility** |
| Patient | * Signing up and registering on the system with an automatically assigned unique Hospital ID, * Scheduling an appointment for a given day and a specialist, * Attending to the appointment of the selected day. |
| Specialist | * Establishing a new sort of specialty, * Managing appointments; Browsing through appointments, as well as attending, completing and cancelling them. |
| Administrator | * Building reports for a given specialist and a date. |
| Front Desk | * Cancelling unattended late appointments if the patient or the specialist fails to attend. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | When a patient is willing to register, they enter their name in the corresponding text field; the system will automatically assign a unique Hospital ID. Once they proceed to add, and their details will be listed on the list located on the right side. After the registration, when a patients switches to the appointment screen, they first of all enter their unique Hospital ID, and click on “Get Patient” button. Once the button is clicked their booking information will be listed below. | | |
| Accordingly the patient selects a specialist from the list and then they choose a date picker and navigate between dates by using the previous and next day buttons. The aim of the appointment date picker structure is that weekends are excluded, and passed day’s selection is also excluded. Patients neither can select a passed date nor a weekend date. Later on the patient clicks on “Get Available Slots” button to build the available slots. Here is a small hint of a possible algorithm of the available slots; a slot is occupied and not listed, once another patient books for that slot and attends to the appointment, if the patient does not attend, or cancelled by the selected specialist, the slot is however still free. After all once the patient succeeds the completion of criteria selections, they click on “Schedule Appointment” to schedule the appointment with the given details. | | |  | |
|  | For patients to attend to their selected appointment, they enter their unique Hospital Id and Booking code in the corresponding fields and click on attend. Once they proceed that slot will be occupied and excluded on available slots for other patients. The selected appointment status then will be set at “WAITING”. To keep in mind that only daily attendance is allowed. For example when the patient enters details for future date, they’ll not be able to attend it. | | | |
| Specialist operation screen allows adding a new sort of specialist, once the kind detail is entered in the regarding text field, as soon as add button is press, then throughout the life time of the application, the specialist will remain on the system. To keep it in mind that there can only be one kind at a time, same kind of a specialist cannot be inserted in the list. List of available specialists on system will be displayed in the list box. | | | |  |

|  |  |  |
| --- | --- | --- |
|  | In the specialist appointments screen, specialists are able to manage their appointments. First of all, a desired specialist is selected from the Combo box and then “Get Specialist Appointment” button is fired to retrieve corresponding specialist’s appointments. Before giving the details of appointment management, previous appointment listing is not allowed, thus specialists can only view their current and future appointments by using the date picker. | |
| To imply the appointment management logic here, first of all specialist are allowed the cancel current and future appointments whose status codes are in “WAITING” and “UNATTENDED”, as well as they can only attend to one appointment at a time. The attendance part is that first of all a patient has to attend to their appointment, and accordingly their appointment’s status will be set as “WAITING” shown as in the appointment “CFR12”. Moreover specialists consequentially cannot attend to future appointments only daily appointment attendance is allowed. Similarly specialists cannot attend to appointments with “UNATTENDED” status codes. For instance shown in the screen the Cardiac has already attended to the appointment with booking code “CFR11”. For the cardiac to attend to the next appointment they have to choose the appointment whose status is “IN PROGRESS”, and the click on the “Complete Appointment” button. Then the selected appointment’s status will be adjusted to “FINISHED”. The shown sample in the appointment with booking code “CFR10”. Moreover on the contrary an active appointment cannot be cancelled; it logically has to be completed. | | |
| Last of all the remaining screen belongs to administrators. Here on this screen admins are able to produce report of a selected specialist for a given month. In this screen only the appointments whose statuses are in “CANCELLED” and “FINISHED” are listed. Above the list amount of finished and cancelled appointments are individually given in numeric figures. | |  |

Java Swing and AWT libraries are implemented into the application, when the application has been launched there are three objectives being carried out; firstly the dummy data of specialist, patient and appointment is inserted, secondly the application is run in multiple threads configured and managed by Swing utilities, by using this approach the consistency of the application has been provided, lastly there is a front desk implementation which also runs throughout the life time of the application. This sort of implementation runs and sleeps for 5 seconds and the purpose is to cancel late unattended appointments either patient fails to attend or the specialist does. We can think of this like a real world example of a front desk in a hospital which manages appointment flow and management.

Speaking of GUI, there is an approximate MVC design pattern implemented in the application, there are model and view sides in the application, but different than MVC model also controllers are integrated into views. In essence there is a main container Frame in which all other GUIs run in it as Internal frames. Coupled with main frame and internal frames, similarly JMenu provides ability to switch among screens. Each objective is assigned with a corresponding internal frame.

Moreover map interface has been adopted as a scaffold data structure in order for preserving complex relational data e.g. appointment and patient data. Speaking of map interface, however to keep objects nicely sorted, tree map implementation designated. In the sorting regard Tree map requires the corresponding object to implement a sorting interface. In this case the appointment object model implements comparable interface, as a result of that the object class automatically acquired the “compareTo” method to assign a custom comparison algorithm. To sum up all inserted data into tree map implementation is sorted by date values. Besides that providing consistency in statuses, an interface class has been created for this purpose called “StatusCodes”. In this interface those status codes are available;

|  |  |
| --- | --- |
| **Status Code** | **Objective** |
| STATUSCODE\_WAITING | The patient has attended to the appointment and waiting for the specialist attendance. |
| STATUSCODE\_INPROGRESS | The patient is currently diagnosed and the specialist is occupied. |
| STATUSCODE\_FINISHED | The appointment is finished successfully. Both patient and specialist did attend to the appointment completed the session. |
| STATUSCODE\_CANCELLED | The appointment is either cancelled by the front desk because patient or specialist failed to attend or specialist has been canceled it. |
| STATUSCODE\_UNATTENDED | The patient booked for the appointment, but their attendance is expected. |

On the other hand the working hours of the hospital are from 09.00 AM to 05.00 PM and in this case the application has a real time structure. To clarify the regarding structure, the application relies on the system clock to have a consistent appointment management. When a booking transaction is completed by a patient, the appointment date is set by using the system clock and as an illustration in this logic; if an appointment is booked for 09.00AM and the system clock is 10.01 AM, the front desk will automatically cancel it no doubt. By adopting this sort of an approach, patients are constituted not to grasp a passed date value which may be illogical.

Reemphasizing the date picker structure, weekends are excluded. When patient and specialist use date picker they both neither can choose a weekend date for appointment booking nor attend to a weekend appointment which is still unlike. The application will always return a week day date, patients and specialists are disallowed entering a particular date. They need to use the custom made date picker.

Furthermore, many patients can book for the same day and hour of a desired specialist, but if one of them attends to the appointment then the patient who are about to schedule for the corresponding specialist will not be able to see that appointment while booking. Because many can book and fail to attend but earlier only one patient can attend. Similarly specialists can only attend to one appointment at a time. They are disallowed attending to multiple appointments that can be just wrong for the real world logic.

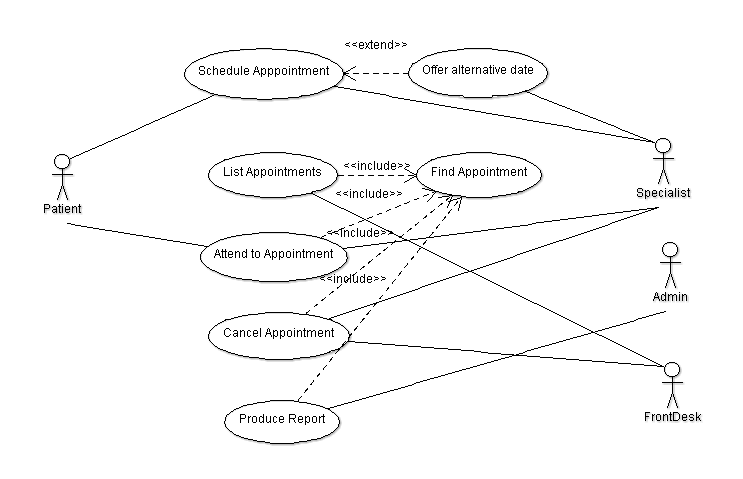
Moreover there is also unit test cases implemented and conducted. JUnit test framework version 4 is used, and only DAO interface is tested. Other than that the database class has been left off excluded apart from the testing, because the collection tests requires other testing components which does not come as a bundle in Netbeans IDE. As a matter of fact there are fifteen test cases prepared for testing purposes.

On the contrary there are also design patterns implemented into the application; Data Access Object which is an interface implementation also known as DAO, and Iteration patterns. DAO pattern helped to have different methods for mission specific objectives. Moreover DAO Pattern has two classes one is an interface which only has the method, and the implementation class inherits methods from the parent interface class and method bodies are filled in this implementation class. On the other hand while dealing with collections, iterator pattern has helped to handle such object processes like removing a complete object from a collection as well as searching through the collections.

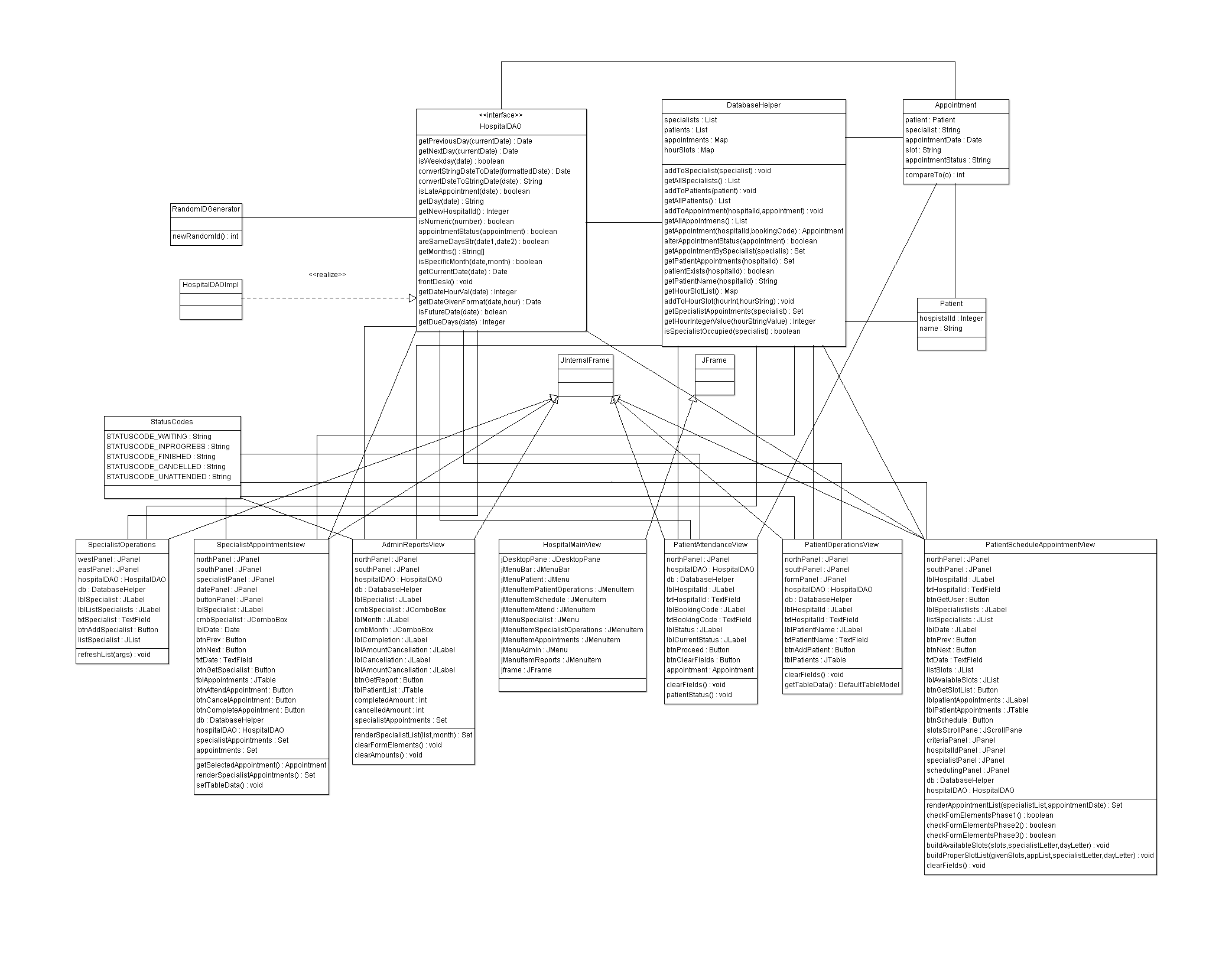
Last of all those objects which mentioned as appointment and patient, classes are considered as such Java beans. Java beans are a part of reusable code. In this case those beans have getter and setter methods which help objects to address and hold data for each mentioned property.

# UML Diagrams

## Use Case Diagram



## Class Diagram



## State Diagram

