CS102	Spring 2020/21	Project Group	G1A
Assistant:	Selçuk Gülcan		

~ PATIENT MANAGEMENT SYSTEM FOR HOSPITALS ~

Criteria	TA/Grader	Instructor
Presentation		
Overall		

Yusuf Doğan 21702970

Kardelen Ceren 22003017

Efe Can Tepe 21903187

Eylül Badem 22003079

Emre Uğur 22002802

Detailed Design Stage

(version 1.0)

7 April 2021

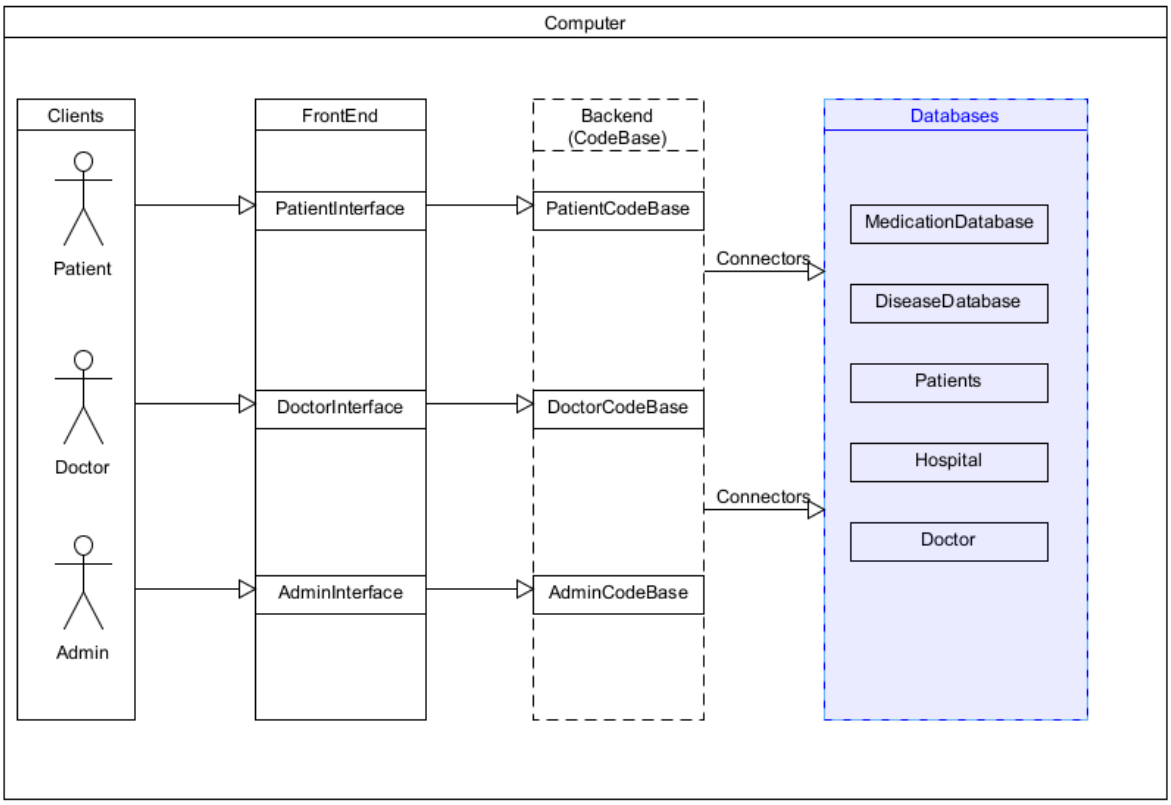
1. Introduction

Our project is to provide hospitals, doctors and patients with a robust management system where users can see and edit many different types of data, create statistics, and schedule appointments.

2. System Overview

2.1 System Architecture

This project will be a desktop application. For database connectivity, JDBC will be used. Swing will be used for the GUI and MySQL for the database. Graphics will be created in the SQL language. It will be assumed that all the users share the same computer as we informed earlier this semester.



The system architecture of the program consists of 4 different parts: Client, Frontend, Backend, Databases.

Model-View-Controller architecture will be respected in the project implementation. Model classes are outlined in the UML diagram below.

The client section represents the user types of the application. Clients can interact with the GUI (frontend) which is linked to model classes.

The frontend section will be a bridge between clients and the codebase.

The frontend section will be a bridge between clients and the codebase. The user will interact with this page. 3 different GUI will be represented according to user types.

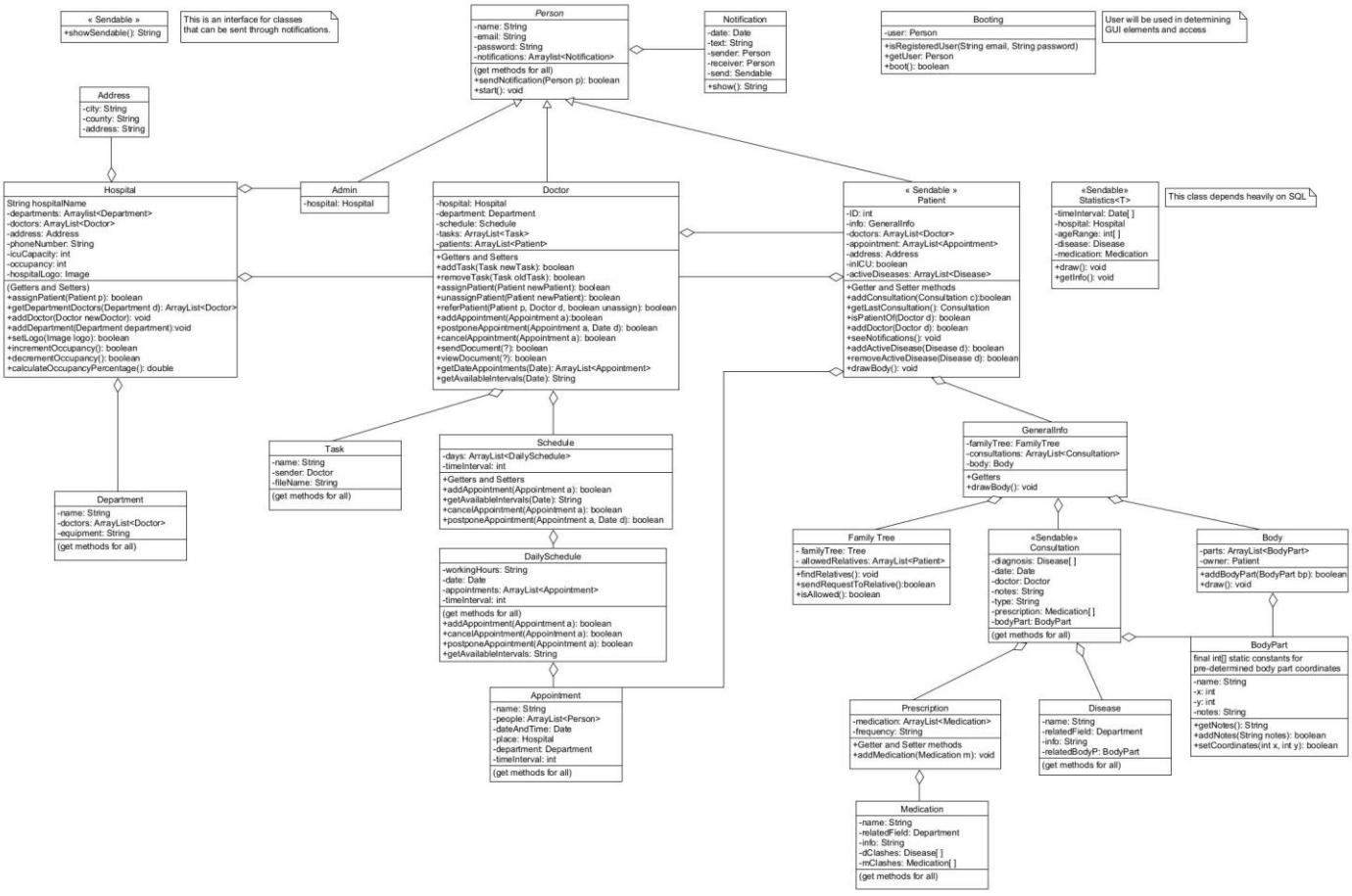
Also, the GUI part will handle some operations which seems effortless for the user but complicated in the code such as sending and receiving information from one user to another. The responsibility of GUI is communicating with the codebase, triggering actions in the codebase and receiving the output to display in the user interface.

The codebase section will be a bridge between GUI and the databases. It will also support some code blocks to maintain a part of the program which users will not be aware of. Dashlines in the codebase section represent how the codebase will behave differently according to the user type. Even though clients of the program share the common codebase, they will be restricted to access some data and use certain functionalities by the GUI.

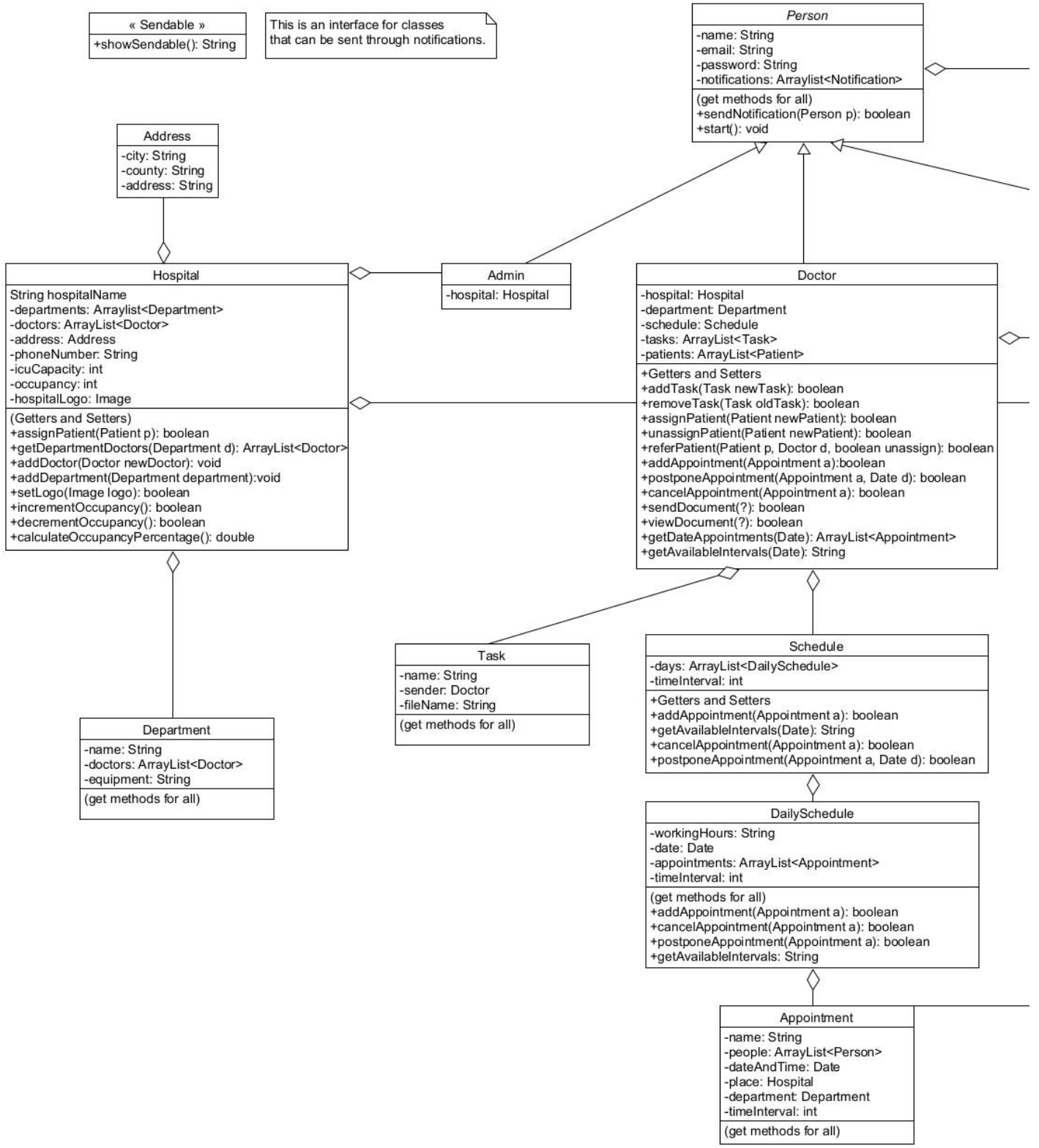
The database section will hold all the information we need. Also, it will create graphs to show the information for the user to understand better.

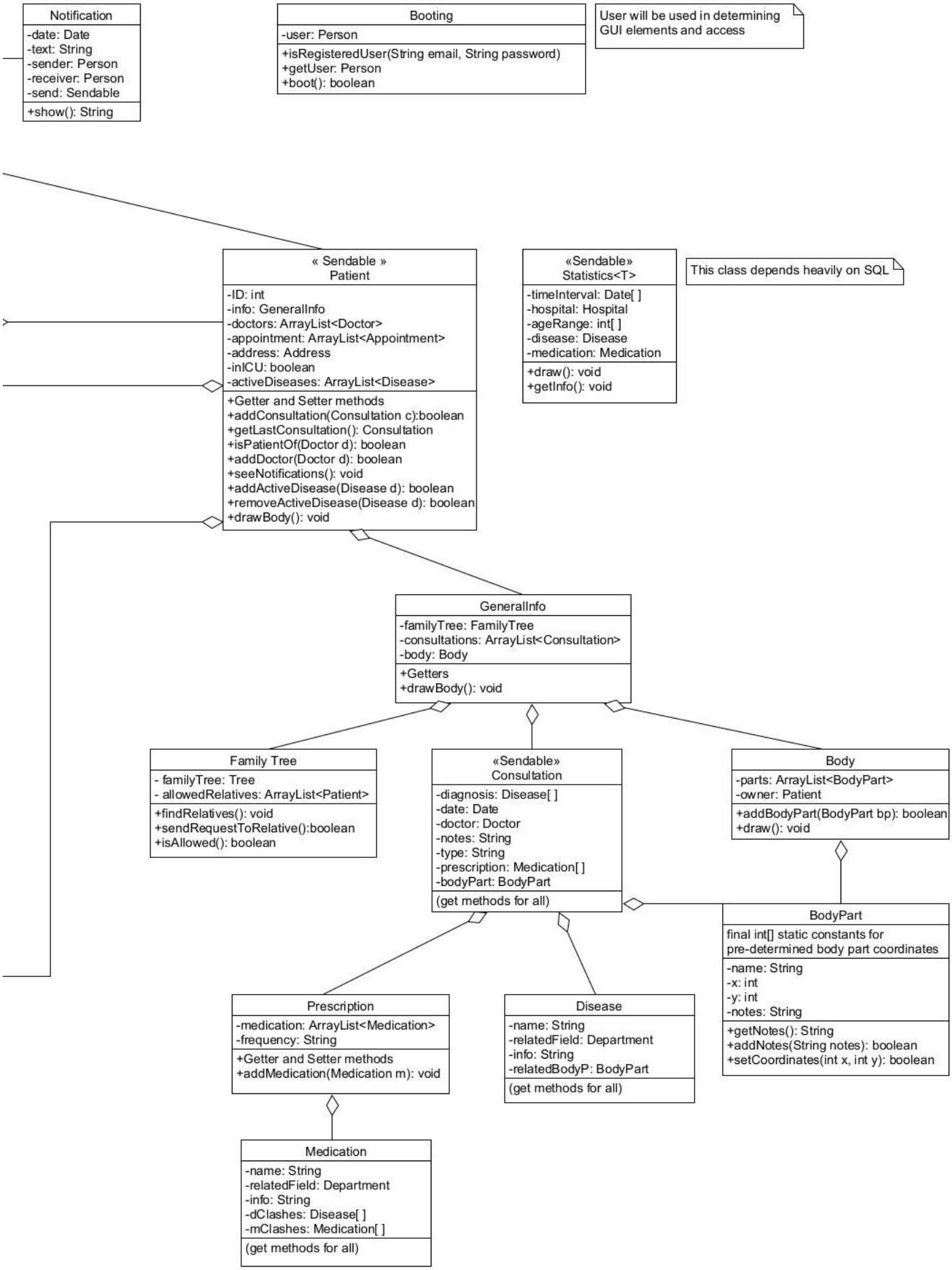
3. Core Design Details

Here’s a general view of the UML diagram of our model classes.



Easier to read pictures can be found below.


« Sendable »
+showSendable(): String
This is an interface for classes that can be sent through notifications.
Person
-name: String
-email: String
-password: String
-notifications: Arraylist<Notification>
(get methods for all)
+sendNotification(Person p): boolean
+start(): void
Address
-city: String
-county: String
-address: String
Hospital
String hospitalName
-departments: Arraylist<Department>
-doctors: ArrayList<Doctor>
-address: Address
-phoneNumber: String
-icuCapacity: int
-occupancy: int
-hospitalLogo: Image
(Getters and Setters)
+assignPatient(Patient p): boolean
+getDepartmentDoctors(Department d): ArrayList<Doctor>
+addDoctor(Doctor newDoctor): void
+addDepartment(Department department):void
+setLogo(Image logo): boolean
+incrementOccupancy(): boolean
+decrementOccupancy(): boolean
+calculateOccupancyPercentage(): double
Admin
-hospital: Hospital
Doctor
-hospital: Hospital
-department: Department
-schedule: Schedule
-tasks: ArrayList<Task>
-patients: ArrayList<Patient>
+Getters and Setters
+addTask(Task newTask): boolean
+removeTask(Task oldTask): boolean
+assignPatient(Patient newPatient): boolean
+unassignPatient(Patient newPatient): boolean
+referPatient(Patient p, Doctor d, boolean unassign): boolean
+addAppointment(Appointment a):boolean
+postponeAppointment(Appointment a, Date d): boolean
+cancelAppointment(Appointment a): boolean
+sendDocument(?): boolean
+viewDocument(?): boolean
+getDateAppointments(Date): ArrayList<Appointment>
+getAvailableIntervals(Date): String
Task
-name: String
-sender: Doctor
-fileName: String
(get methods for all)
Schedule
-days: ArrayList<DailySchedule>
-timeInterval: int
+Getters and Setters
+addAppointment(Appointment a): boolean
+getAvailableIntervals(Date): String
+cancelAppointment(Appointment a): boolean
+postponeAppointment(Appointment a, Date d): boolean
Department
-name: String
-doctors: ArrayList<Doctor>
-equipment: String
(get methods for all)
DailySchedule
-workingHours: String
-date: Date
-appointments: ArrayList<Appointment>
-timeInterval: int
(get methods for all)
+addAppointment(Appointment a): boolean
+cancelAppointment(Appointment a): boolean
+postponeAppointment(Appointment a): boolean
+getAvailableIntervals: String
Appointment
-name: String
-people: ArrayList<Person>
-dateAndTime: Date
-place: Hospital
-department: Department
-timeInterval: int
(get methods for all)


Notification
-date: Date
-text: String
-sender: Person
-receiver: Person
-send: Sendable
+show(): String
Booting
-user: Person
+isRegisteredUser(String email, String password)
+getUser: Person
+boot(): boolean
User will be used in determining GUI elements and access
« Sendable »
Patient
-ID: int
-info: GeneralInfo
-doctors: ArrayList<Doctor>
-appointment: ArrayList<Appointment>
-address: Address
-inICU: boolean
-activeDiseases: ArrayList<Disease>
+Getter and Setter methods
+addConsultation(Consultation c):boolean
+getLastConsultation(): Consultation
+isPatientOf(Doctor d): boolean
+addDoctor(Doctor d): boolean
+seeNotifications(): void
+addActiveDisease(Disease d): boolean
+removeActiveDisease(Disease d): boolean
+drawBody(): void
«Sendable»
Statistics<T>
-timeInterval: Date[ ]
-hospital: Hospital
-ageRange: int[ ]
-disease: Disease
-medication: Medication
+draw(): void
+getInfo(): void
This class depends heavily on SQL
GeneralInfo
-familyTree: FamilyTree
-consultations: ArrayList<Consultation>
-body: Body
+Getters
+drawBody(): void
Family Tree
- familyTree: Tree
- allowedRelatives: ArrayList<Patient>
+findRelatives(): void
+sendRequestToRelative():boolean
+isAllowed(): boolean
«Sendable»
Consultation
-diagnosis: Disease[ ]
-date: Date
-doctor: Doctor
-notes: String
-type: String
-prescription: Medication[ ]
-bodyPart: BodyPart
(get methods for all)
Body
-parts: ArrayList<BodyPart>
-owner: Patient
+addBodyPart(BodyPart bp): boolean
+draw(): void
BodyPart
final int[] static constants for pre-determined body part coordinates
-name: String
-x: int
-y: int
-notes: String
+getNotes(): String
+addNotes(String notes): boolean
+setCoordinates(int x, int y): boolean
Prescription
-medication: ArrayList<Medication>
-frequency: String
+Getter and Setter methods
+addMedication(Medication m): void
Disease
-name: String
-relatedField: Department
-info: String
-relatedBodyP: BodyPart
(get methods for all)
Medication
-name: String
-relatedField: Department
-info: String
-dClashes: Disease[ ]
-mClashes: Medication[ ]
(get methods for all)

4. Task Assignment

Yusuf Doğan: Hospital, Department, Booting, FamilyTree

Kardelen Ceren: Patient, Schedule, DailySchedule, Appointment

Efe Can Tepe: Body, Consultation, Prescription, Medication, BodyPart

Eylül Badem: Doctor, GeneralInfo, Person, Task, Disease

Emre Uğur: Statistics, Admin, Notification, Address