"Sehha" Hospital Reception System

Due on 20/05/2023, midnight, on Blackboard

Project description

1. Introduction

"Sehha" is a Qatari hospital that is dedicated to serve the Qatari citizens, residents or even visitors. It is known for its best services and affordable prices. The hospital has been established 30 years ago and it is due for a major update to replace the traditional paper-based system into a computerized system.

Therefore, and as a software engineer, Sehha's management asked you to upgrade their reception management system.

2. A hospital visit walk-through:

When a patient comes into the hospital, their information is recorded using either their Qatari ID if they are a citizen or a resident, or through a valid visa number.

The patient chooses one or more among these types of services:

Service types	Price	Details	Maximum slots per day
procedure	50 QAR	a procedure such as CT image, blood withdrawal	15
generic	100 QAR	a generic medical checkup	20
specialized	150 QAR	a specialized medical checkup for those who seek specialized advice	10
operation	1000 QAR	an operation on selected surgery fields	5

Table 1: Services description and details

The patient chooses the suitable service and asks to reserve a slot accordingly. The patient then pays the total amount of services prices that are added up to the invoice.

3. Assumptions:

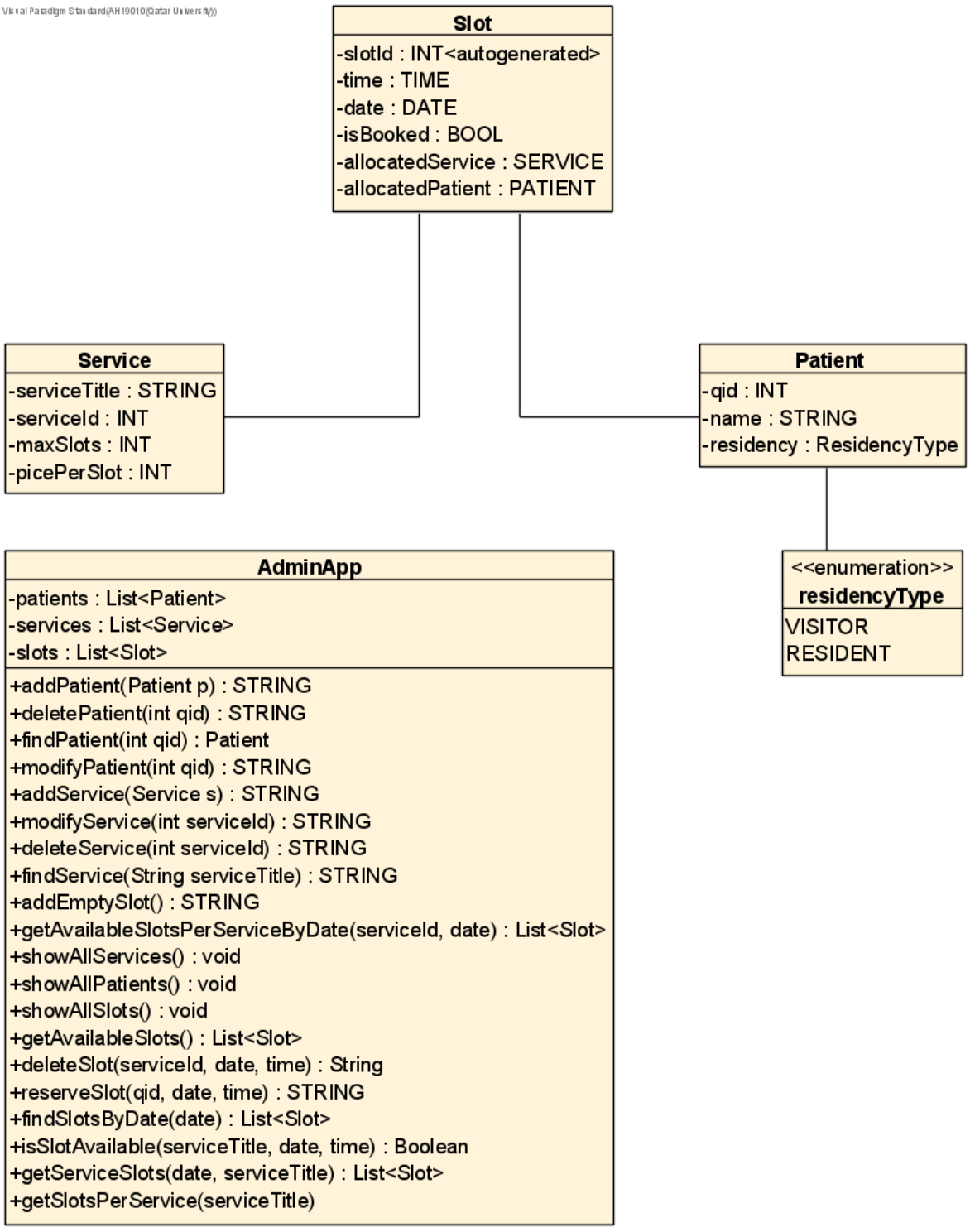
- One patient can reserve for multiple services at different times.
- One service can be done to one patient maximum at a time. Similarly, one patient cannot be allotted two slots at the same time and date even if services are different.
- Working hours for the hospital start from 7 AM to 9 PM. One slot duration in 30 minutes.
- A patient is allowed to reserve 1 or more slots in a day for the same or different services.

4. System functional requirements:

The system should be able to perform the below tasks:

1. add, update, delete a patient.
2. add, update, delete a hospital service.
3. add, update, delete a service slot.
4. generate reports for:
    a. all hospital services, their prices, and the frequency of the service reservations (i.e., how many times a service was reserved) in a tabular-like format.
    b. all the registered patients
    c. all slots' status for all services for all patients
5. Data Management: your system must store all data entered though console permanently in files. When running the system, data must be retrieved from files and recreated in memory. When the user exits the system, all changes/new data must be saved permanently.

The below methods in table 3 should be implemented and adapted to realize the above tasks.


Visual Paradigm Standard(AH19010(Qatar University))
Slot
-slotId : INT<autogenerated>
-time : TIME
-date : DATE
-isBooked : BOOL
-allocatedService : SERVICE
-allocatedPatient : PATIENT
Service
-serviceTitle : STRING
-serviceId : INT
-maxSlots : INT
-picePerSlot : INT
Patient
-qid : INT
-name : STRING
-residency : ResidencyType
<<enumeration>>
residencyType
VISITOR
RESIDENT
AdminApp
-patients : List<Patient>
-services : List<Service>
-slots : List<Slot>
+addPatient(Patient p) : STRING
+deletePatient(int qid) : STRING
+findPatient(int qid) : Patient
+modifyPatient(int qid) : STRING
+addService(Service s) : STRING
+modifyService(int serviceId) : STRING
+deleteService(int serviceId) : STRING
+findService(String serviceTitle) : STRING
+addEmptySlot() : STRING
+getAvailableSlotsPerServiceByDate(serviceId, date) : List<Slot>
+showAllServices() : void
+showAllPatients() : void
+showAllSlots() : void
+getAvailableSlots() : List<Slot>
+deleteSlot(serviceId, date, time) : String
+reserveSlot(qid, date, time) : STRING
+findSlotsByDate(date) : List<Slot>
+isSlotAvailable(serviceTitle, date, time) : Boolean
+getServiceSlots(date, serviceTitle) : List<Slot>
+getSlotsPerService(serviceTitle)

Class	General description
Patient	Patient class to hold information about the patient: QID, name, and residencyType
Service	A service that could be one of the following: a generic checkup, specialized checkup, procedure, or surgery. Each service is characterized by an ID, title, price.
Slot	A service slot is a 30 mins timeframe that is allocated for one service to be done for one patient. The slot is characterized by a specific date and time.

Table 2: Main classes high-level descriptions

Below is the description of the different class methods that would be implemented during this project phase:

SehhaAdminApp	
`addPatient (Patient P): String`	a method to add a new patient and returns a String: `"Patient added successfully"`
`deletePatient (int qid): String`	a method to remove a patient. The patient is found using the QID. It returns a String: `"Patient removed successfully"`
`findPatient (int qid): Patient`	a method to find the patient by QID and returns the Patient object

`modifyPatient (int qid): String`	a method to modify any of the patient's information. For instance, once the method is called, it asked the user what to modify and depending on user input, data gets modified.
	
	Example:
	`(1) Modify name`
	`(2) modify residencyType`
	
	returns a String: `"Patient updated successfully"` `else "Patient not updated."`

`addService (Service s): String`	a method to add a new service and returns a String: `"Service added successfully"`
`modifyService (int serviceID): String`	a method that modifies the service information in a similar fashion to "modifyPatient" method in Patient class.

	returns a String: "Service updated successfully"
deleteService (int serviceID): String	a method to delete a Service and all of the related slots to it. It returns a String: "Service and all related slots are deleted successfully"
findService (String serviceTitle): Service	a method that finds a service by its title and displays all related information to it: how many slots associated with it, how many booked, etc.
addEmptySlot (): String	a method to add a new slot and returns a String: "Slot added successfully". the isBooked flag is set to false, initially. The Patient and Slot objects can be null as it is an empty slot.
getAvailableSlotsPerServiceByDate (int serviceID, LocalDate date): ArrayList <Slot>	a method that takes a service ID and date and it returns all the available (i.e., isBooked is false) slots of that service on the specified date.
showAllServices(): void showAllPatients(): void showAllSlots(): void	methods that display all slots, all services and all slot with their booking status in a tabular-like format
getAvailableSlots(): ArrayList <Slot>	a method that displays all the available (i.e., isBooked is false) slots for all services
deleteSlot (int serviceID, LocalDate date, LocalTime time): String	a method that deletes a specific slot from the system. It returns a String: "Slot deleted successfully"
reserveSlot (slotId, QID): String	a method that assigns an existing empty slot to a specific patient and a specific service. It returns a String: "Slot for patient <patient name> reserved successfully". If slots are all allocated for one day, the slot shall not be reserved.
findSlotsByDate (date): ArrayList <Slot>	a method that searches for slots for all services in a specific date and returns it in a list
IsSlotAvailable (String serviceTitle, LocalDate date, LocalTime time): Boolean	to know whether a specific slot for a service is available or not
getServiceSlots (LocalDate date, String serviceTitle): ArrayList <Slot>	returns slots with specified date and service title with their respective status (isBooked: true or false)
getSlotsPerService (String serviceTitle): ArrayList<Slot>	a method that returns all the slots and their time, date for a specific service in a tabular-like format

Table 3: Core methods

5. Notes and assumptions:

- Use LocalDate and LocalTime classes for date and time objects.
- If any exception occurs, provide the user with proper error explanation to enter another correct input.
- Do not forget to validate/calculate the appropriate inputs/data before performing changes inside of the methods.
- All getters, getters, parameterized and non-parametrized constructors and `toString()` are not included in Table 3. **However, they are mandatory to be present in your code. They were omitted to simplify the diagram and the table.**
- Feel free to add other methods that you think useful for your system, as long as the system meets the functional requirements.
- Add an **AppTest** class having the main method to test all the methods of the application. The app should display the results to the console.
- You can add more methods or instance variables as you see fit if they serve the purpose of achieving the above tasks.
- Write properly the classes and instances comments/description using JavaDoc feature to be able to generate proper classes documentations.

Submission Instructions:

1. This is a **group** project. It is allowed to have **3 students per group, maximum 4**.
2. Start the project immediately after getting the project description to avoid late submissions and to ensure the project is submitted on time.
3. Late submissions would result in **25% penalty per day!**
4. **The name of the member who created the class should be written as author using Javadoc comments throughout the project**.
5. Date of class creation and the version should be mentioned in the Javadoc comments throughout the project.
6. By the submission date, **each group** should submit the following:
    a. The Zip file of the project code
    b. an MS-document report that contains:
        i. Cover page with members names and QU IDs.
        ii. Source code of the developed Java classes (Font of 12 pts).
        iii. Screenshots of all outputs.
        iv. JavaDoc files. Make sure index.html page is present.
7. Best of luck, my students!