Name: Jessica Barnitz

Title: Healthcare Portal

Project Description: A healthcare portal for a medical office that schedules upcoming and previous patient appointments and securely maintains the patient's health record.

Project Features Implemented:

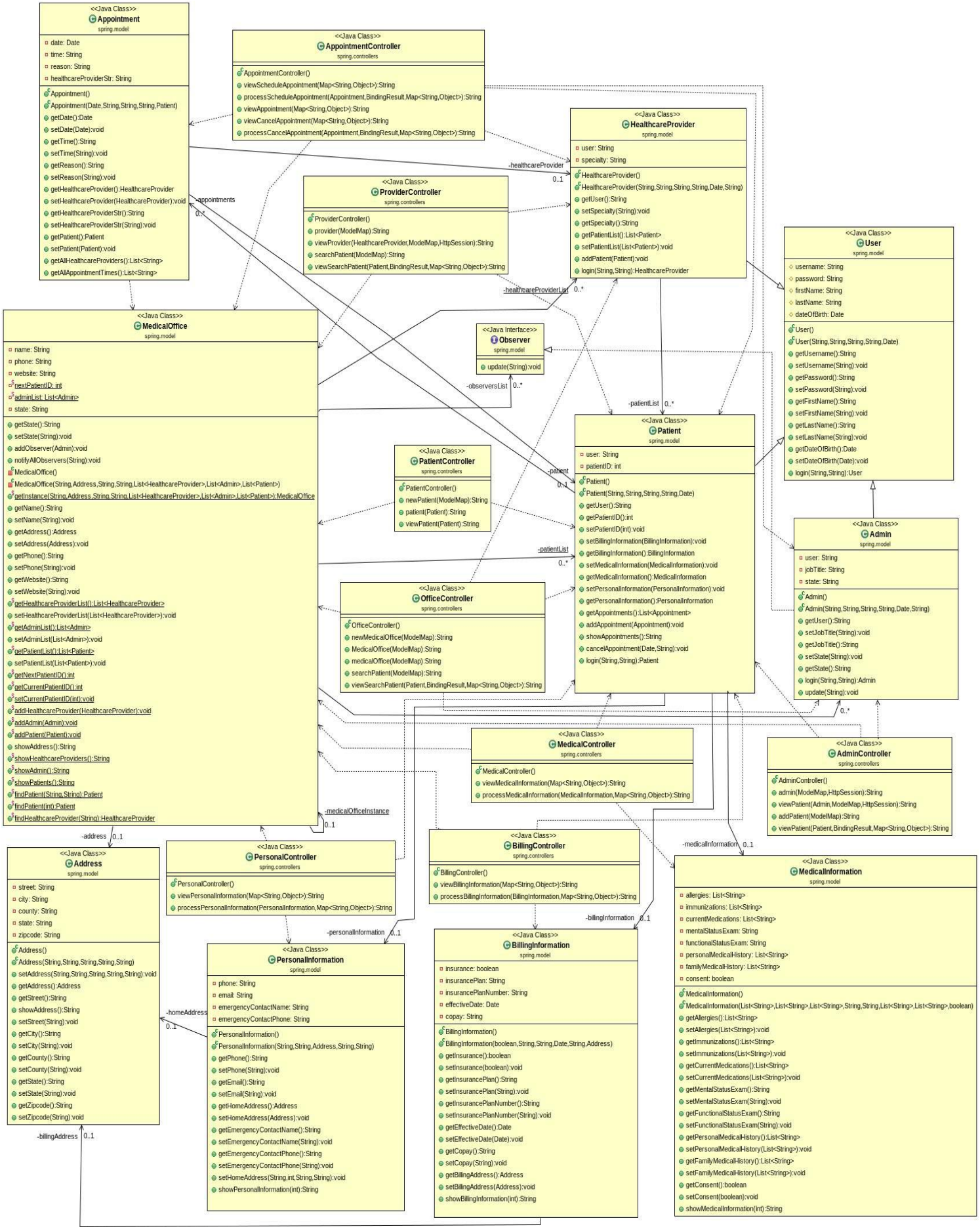
REQ ID	NAME OF REQUIREMENT	RESPONSIBILITIES AS A	I WANT TO	SO THAT
UR-01A	Sign up	Patient	Sign up for the healthcare portal	I can access the healthcare portal
UR-01B	Sign up	Admin	Sign up for the healthcare portal	I can access the healthcare portal with privileged access
UR-01C	Sign up	Healthcare Provider	Sign up for the healthcare portal	I can access the healthcare portal with privileged access
UR-02A	Add patients	Admin	Sign patients up for the healthcare portal	The patient can access the healthcare portal
UR-04A	Request Appointment	Patient	Request an appointment with a certain HCP	I can see my healthcare provider in person on day and time requested
UR-05A	Schedule Appointment	Admin	Schedule an appointment for a patient with a certain HCP	The patient will see their chosen HCP on the day and time requested, HCP schedule permitting
UR-05B	Schedule Appointment	Healthcare Provider	Schedule an appointment for a patient for a follow up visit	The patient will have a follow up visit scheduled prior to leaving the current appointment
UR-06A	Cancel Appointment	Patient	Cancel an existing appointment	I can cancel an existing appointment with my healthcare provider if I am not able to keep the appointment
UR-06B	Cancel Appointment	Admin	Cancel a patients existing appointment with HPC	At the patients request, cancel an existing appointment and free the appointment time in the healthcare provider schedule
UR-08A	Search for Patients	Admin	Search for a patient by name or patient number	I can access their account for updating patient personal information and/or billing information
UR-08B	Search for Patients	Healthcare Provider	Search for a patient by name or patient number	I can access their account to review the patient personal information and/or medical information
UR-09A	Continuum of Care	Patient	Permit or Deny sharing of my medical records	I can determine who accesses my medical information outside of my HPC's medical office

UR-09B	Continuum of Care	Healthcare Provider	Share a patient's medical records with express patient permission	When I refer patients to outside healthcare providers, I can share the patient's medical information for continuum of care

Project Features Not Implemented:

REQ ID	NAME OF REQUIREMENT	RESPONSIBILITIES AS A	I WANT TO	SO THAT
UR-03A	Log in	Patient	Log into the healthcare portal	I can access the patient healthcare portal and the corresponding functionality which includes my personal information, medical information and billing information
UR-03B	Log in	Admin	Log into the healthcare portal	I can access the medical office admin portal and the corresponding functionality which includes access to patient personal information and billing information
UR-03C	Log in	Healthcare Provider	Log into the healthcare portal	I can access the medical office healthcare provider portal and the corresponding functionality which includes access to patient personal information and medical information
UR-07A	Access Medical Office Schedule	Admin	Have full access to the medical office schedule	I can schedule or reschedule a patient who wants to see a certain healthcare provider
UR-07B	Access Medical Office Schedule	Healthcare Provider	Have full access to the medical office schedule	I can determine what patients I have for the day and schedule follow up visits with patients
UR-10A	Appointment	Healthcare Provider	When a patient comes in for an appointment I want to diagnosis symptoms	I can provide patient care and prescribe medications as necessary and have an accurate log of patient encounters

Final Class Diagram:



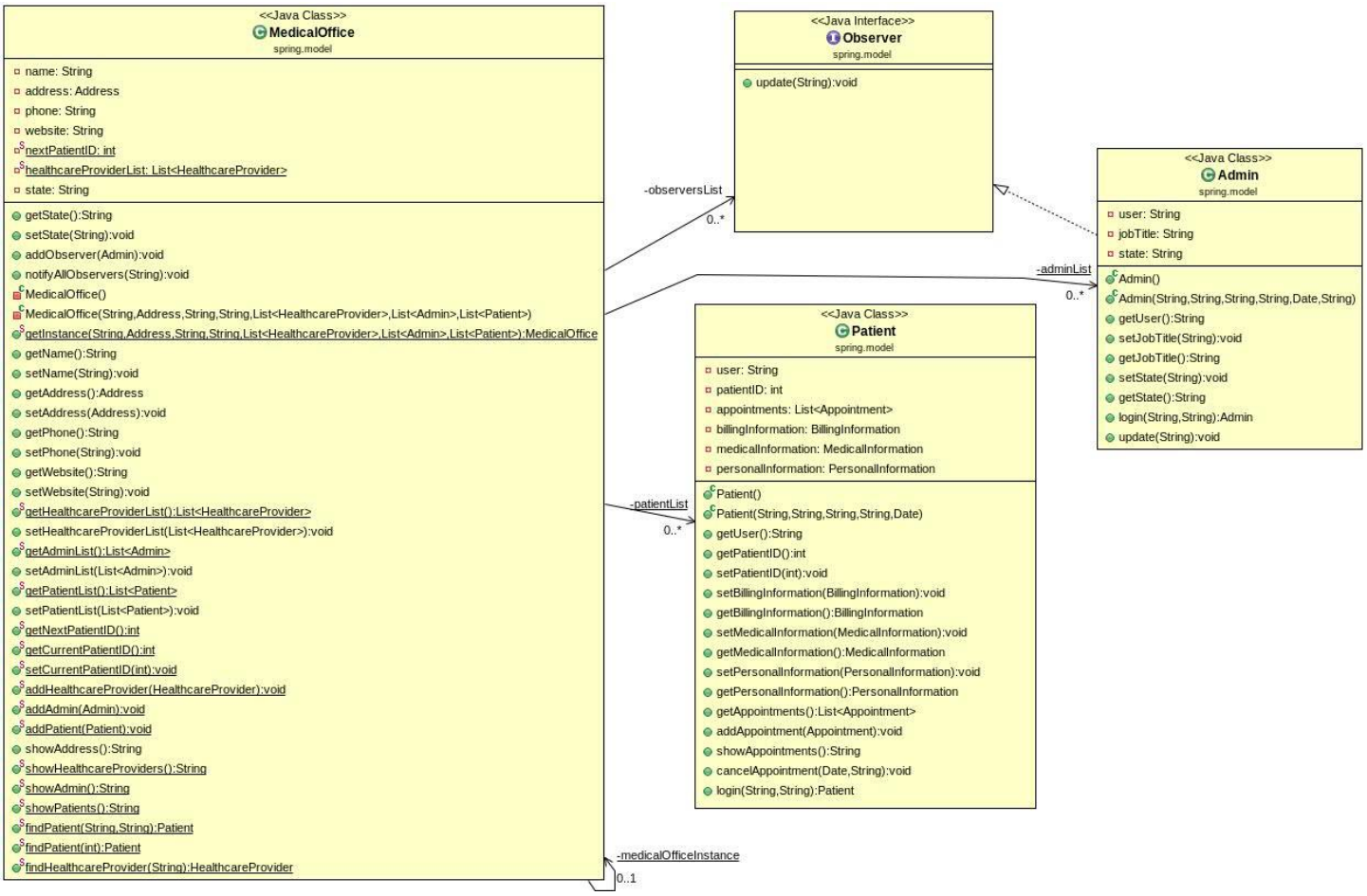
The initial class diagram laid out a great foundation to design a functional object-orientated healthcare portal. By spending time upfront to consider the different user needs and responsibilities the initial class diagram was able to capture the majority of healthcare portal final implementation. Yet, as the healthcare portal began to evolve, additional fields and attributes were implemented to guarantee the functionality of the healthcare portal being able to meet the requirements initially outlined.

Classes that did not appear on the initial class diagram were the controllers, in which the final class diagram shows multiple controllers specific to handling the needs of the model class they are designated for controlling and an Observer class for implementing the Observer design pattern in the healthcare portal.

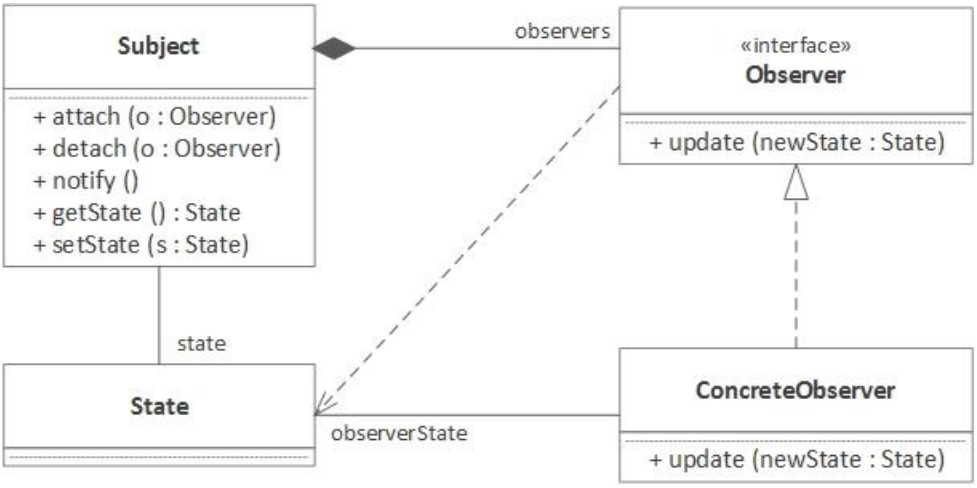
Additionally, two classes from the initial class diagram (PatientInformation and MedicalOfficeSchedule) do not appear on the final class diagram as the initial need for a patient information class was ported over into the patient class and the medical office class as necessary to ensure encapsulation and removing dependencies. The medical office schedule class was removed as it was one of the requirements that was not implemented for the final implementation of the healthcare portal (patients can make, view and cancel appointments and admin are notified of these events, but there is no central screen for accessing this schedule).

Design Pattern Implemented:

- *Show the classes from your class diagram that implemented each design pattern*



- *Observer Design Pattern*



(source: https://dzone.com/articles/the-observer-pattern-using-modern-java)

- *Explain how you implemented the design pattern, why you selected that design pattern*

The observer design pattern was chosen for the healthcare portal as it provides a way to notify the various admin (concrete observer) of real-time changes that occur to the healthcare portal by patients (state) through use of the medical office class (subject). These update notifications occur when a new user signs up for the healthcare portal, when a patient updates personal, medical, or billing information and when a patient schedules or cancels an appointment.

The Observer design pattern was implemented by creating a list of observers of type Admin at the onset of initiating the single instance of the medical office object in the MedicalOffice class. Then as new admin signed up for the healthcare admin portal, they are added to the observers list. There is an observer interface with an abstract method update(String), that lets the Admin determine how to handle the String state that is passed as a parameter to the observers during the method call notifyingAllObservers(String) (*aside: an interface was used as the different users: Admin, Patient, and HealthcareProvider, already extended the User class*). The Admin implementation update(String) blast messages the specific patient information and what state was changed (sign up, billing information update, personal information update, medical information update, or scheduling and canceling an appointment) to the Admin healthcare portal so that necessary steps can be taken when a specific patients state changes. Additionally, by passing in the state of the patient to notify the admin, reduces coupling between the Patient and Admin classes and does not require the observers to query what had changed.

Retrospective:

What have you learned about the process of analysis and design now that you have stepped through the process to create, design and implement a system?

This has been a great journey stepping through the entire process of creating, designing and implementing a system. It illuminates the importance to first plan and design a system in order to best achieve business, user, functional or non-functional requirements. This thorough outlining of the entire system at the onset ensures that the systems vision is maintained, requirements are met, superfluous code is not being introduced and additionally helps navigate handling roadblocks to ensure seamless integration with the entire systems goal. By considering all potential use cases of the system and outlining that in the design process, it becomes much easier to achieve the system requirements while still maintaining enough flexibility to adapt as necessary. Moreover, considering the fundamental principles of object orientated analysis and design as the system was implemented allowed for separation of concerns and a greater increased code reuse. To close, this process has been paramount in my learning and understanding of how to engineer an effective system and how to articulate your system in a succinct manner.