SpotTheDoc

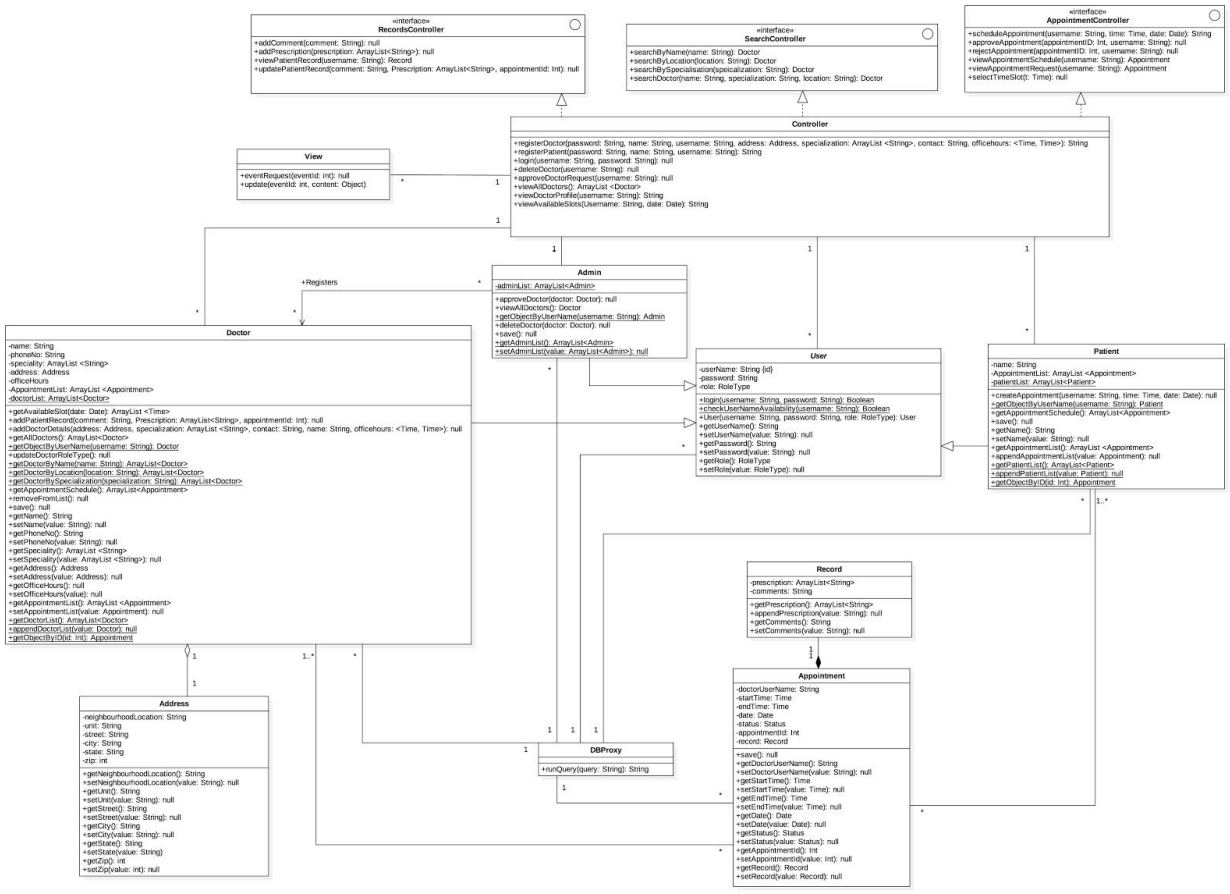
Refactoring

Our design of the class diagram had all of the requirements correctly implemented. Applying the design patterns to the existing design made the system more scalable, flexible and less error prone. We used three design patterns to refactor our system. They are,

1. Facade
2. Observer and
3. Singleton

Class Diagram prior to Refactoring:

Link to old class diagram:
https://raw.githubusercontent.com/prashilbhimani/SpotTheDoc/master/initialClassDiagram.jpg



1. **Facade Design Pattern**

The facade pattern was used to provide a simplified interface to the users of the system. The main controller acts as the interface between the model classes and the sub controllers of the system. The subsystem having the implementation details are the individual controller classes namely

- RecordController
- SearchController and
- AppointmentController

This subsystem formed of the individual controllers is abstracted out so that the users have no direct relationship with it. Hence if the entire subsystem is swapped out, only the main controller has to be modified and all the user classes are left unaffected.

2. **Observer Design Pattern**

Learning the design patterns gave us an easier solution for a problem we faced during the initial design of the system. The problem was to update the appointment schedule for the doctor and the patient whenever a new appointment was created. So when the state of the appointment changed, both doctor and patient should be informed of the change. Thus the observer design pattern was implemented where every doctor and patient is subscribed to the publisher to get notified of any changes occurring to the appointment list.

3. **Singleton Design Pattern**

This pattern was implemented so that there is only one admin for the entire system. So the admin class can be instantiated only once. This ensures that there are no concurrency issues when adding or deleting a doctor from the system.

Steps in refactoring:

1. A static variable and static method was added to the admin class to support singleton pattern using lazy initialization.
2. Subject and Observer Interfaces were added to support the observer design pattern. The doctor and patient class instances are now the observers and the appointment controller class acts as the concrete subject class implementing the update method.
3. The interfaces RecordController, SearchController and AppointmentController were changed from interfaces to classes that implemented the details to manage the recording, searching and appointment scheduling tasks respectively.
4. In order to accommodate these patterns, few classes were modified and operations were added, removed or moved accordingly.

Refactored Class Diagram:

Link to new class diagram:
https://raw.githubusercontent.com/prashilbhimani/SpotTheDoc/master/refactoredClassDiagram.jpg