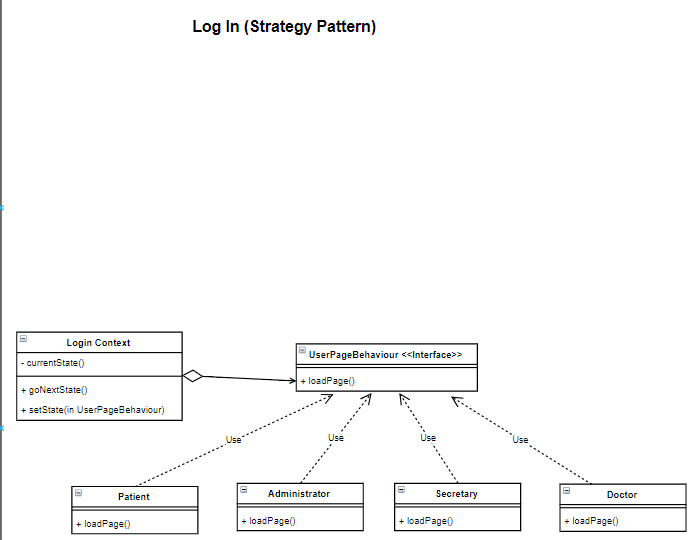
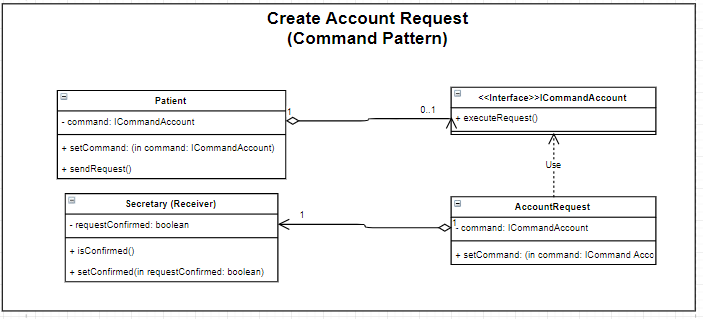
# Reflection

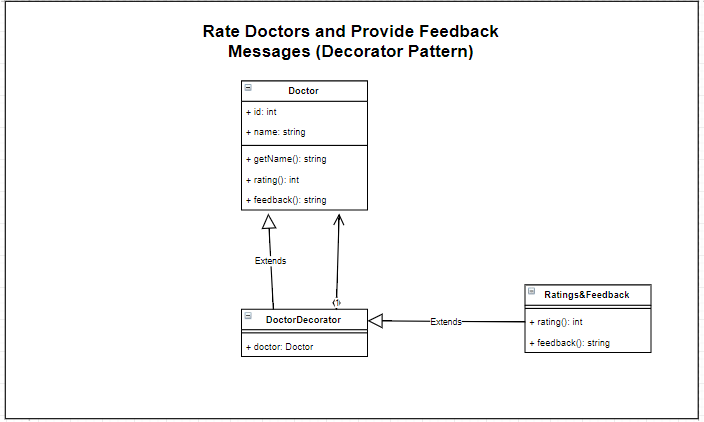
## Repository

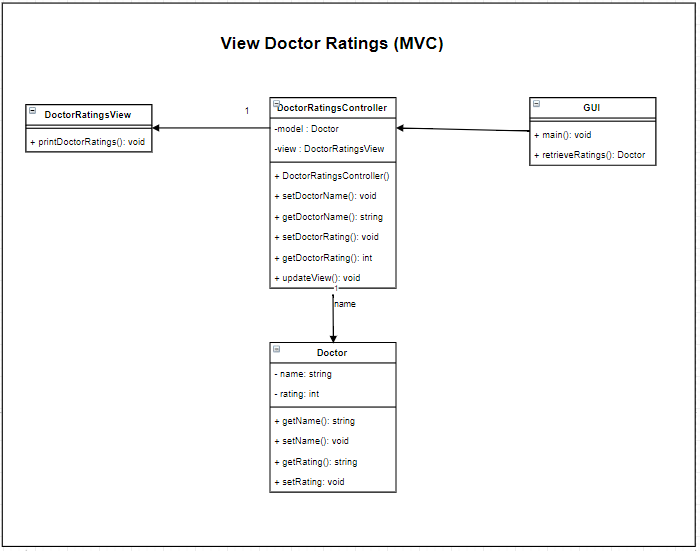
<https://github.com/DanielRichardsWebDesign/SOFT251Assignment>

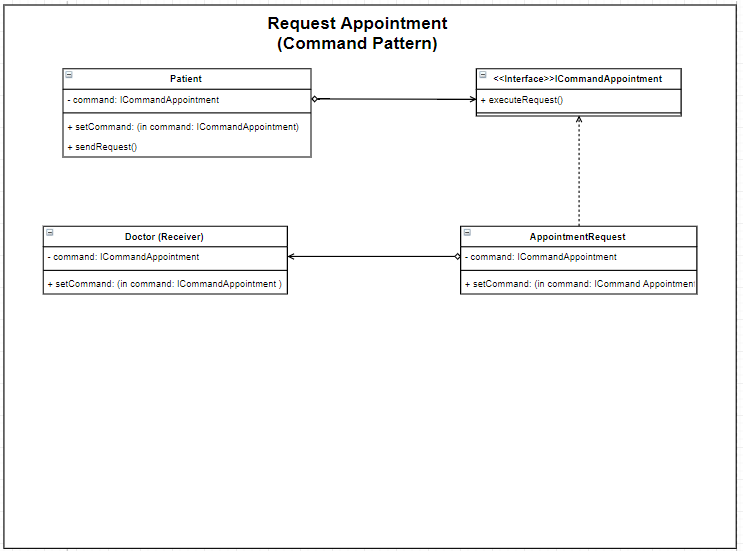
## UML Diagrams

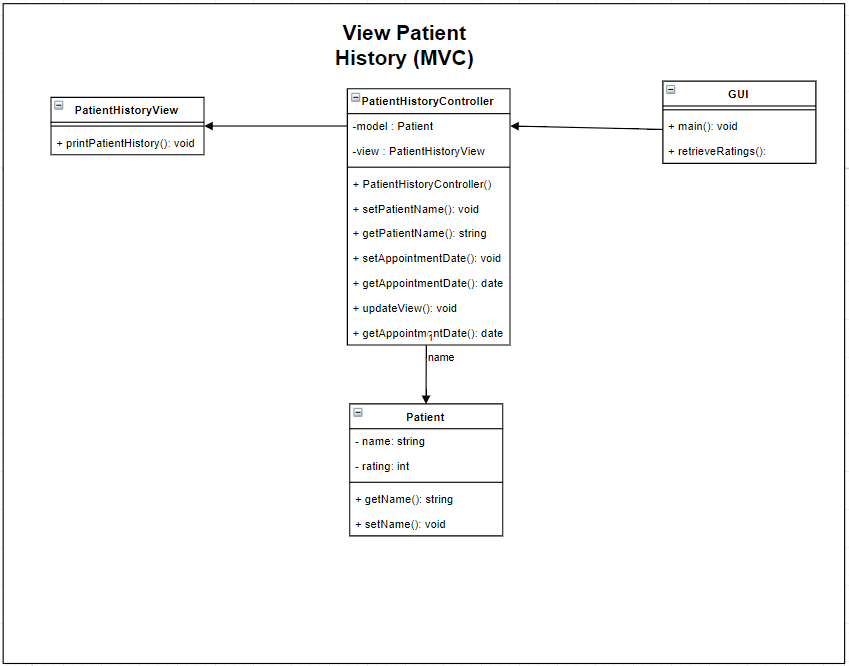


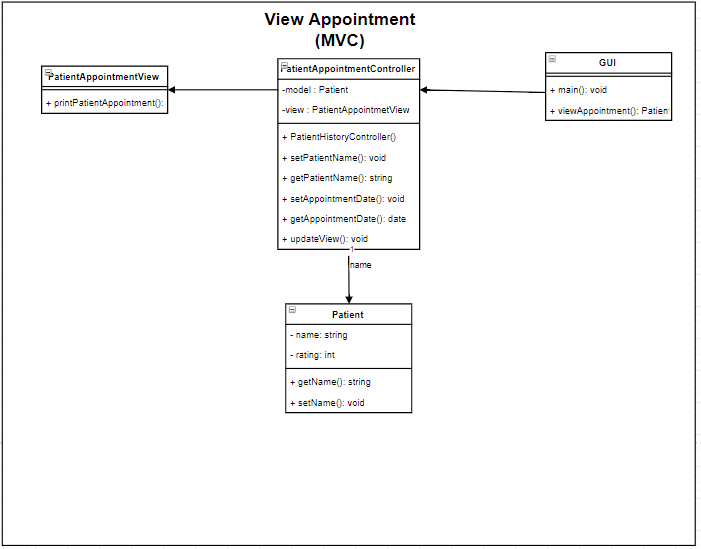


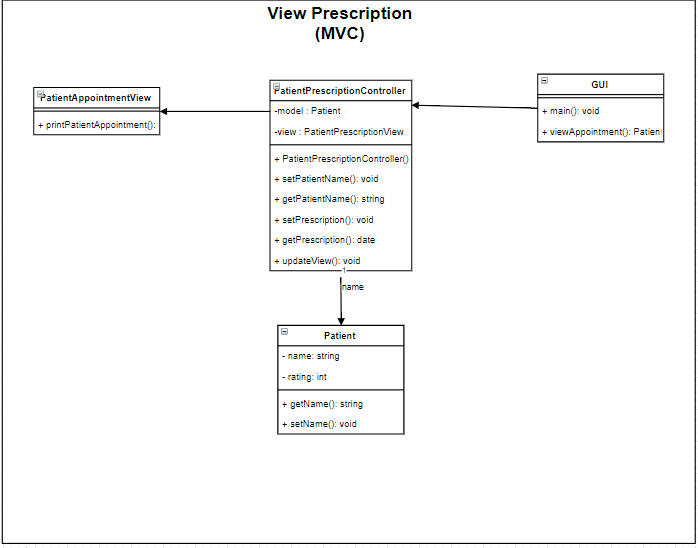


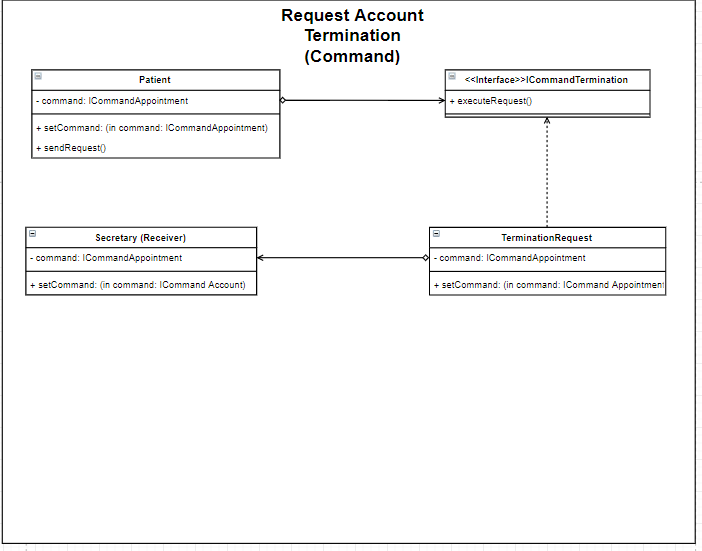


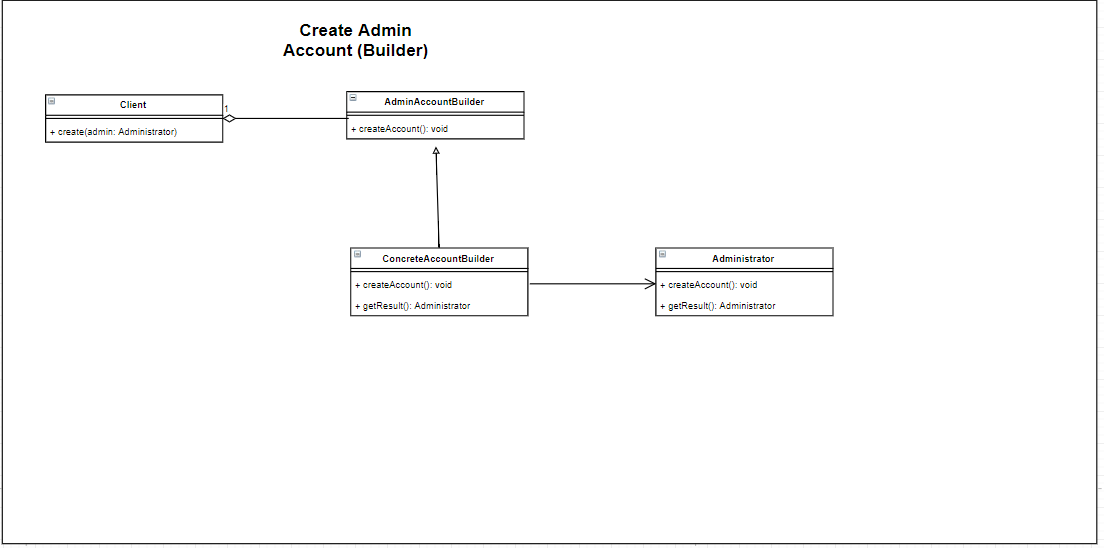


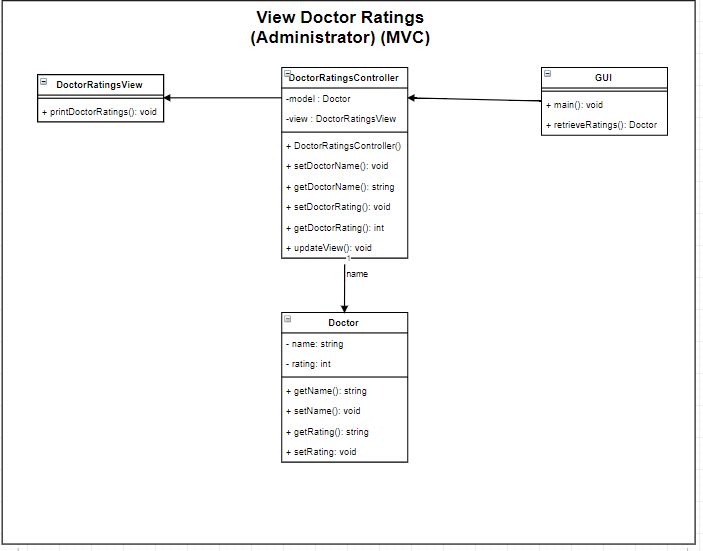


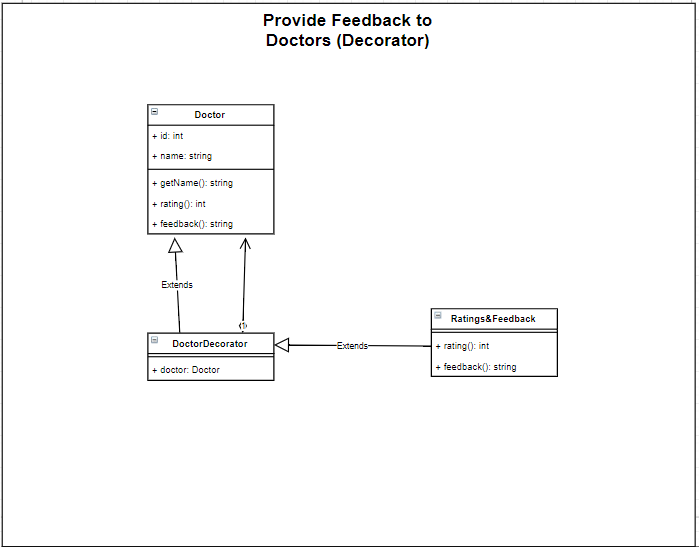


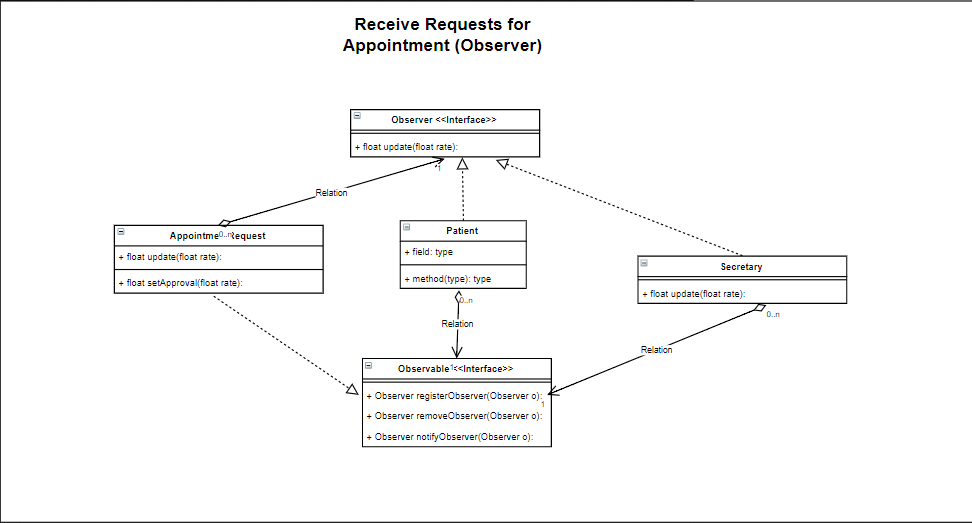


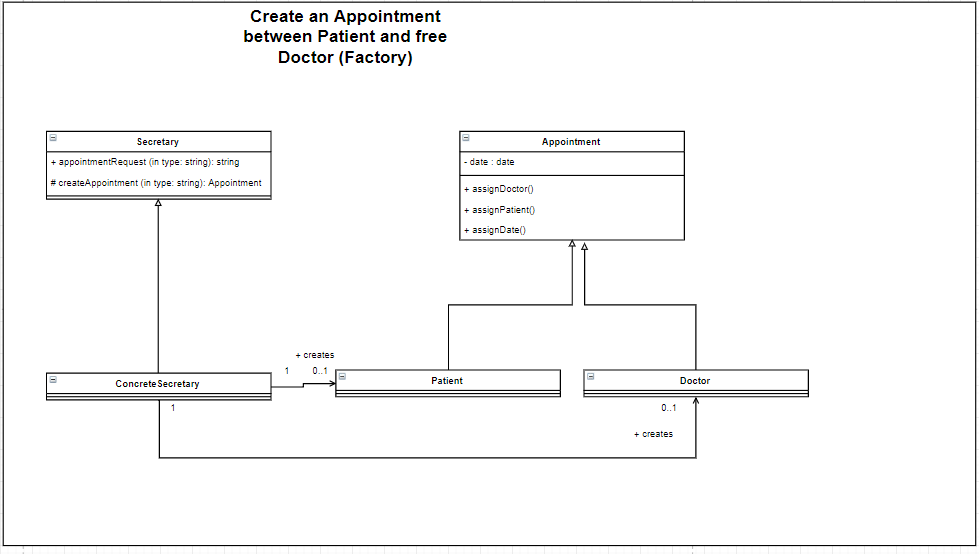


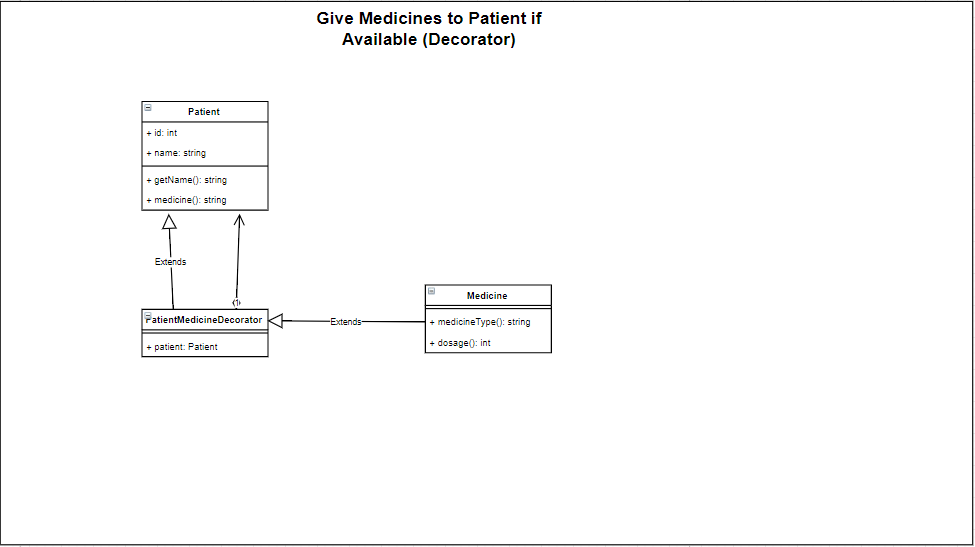


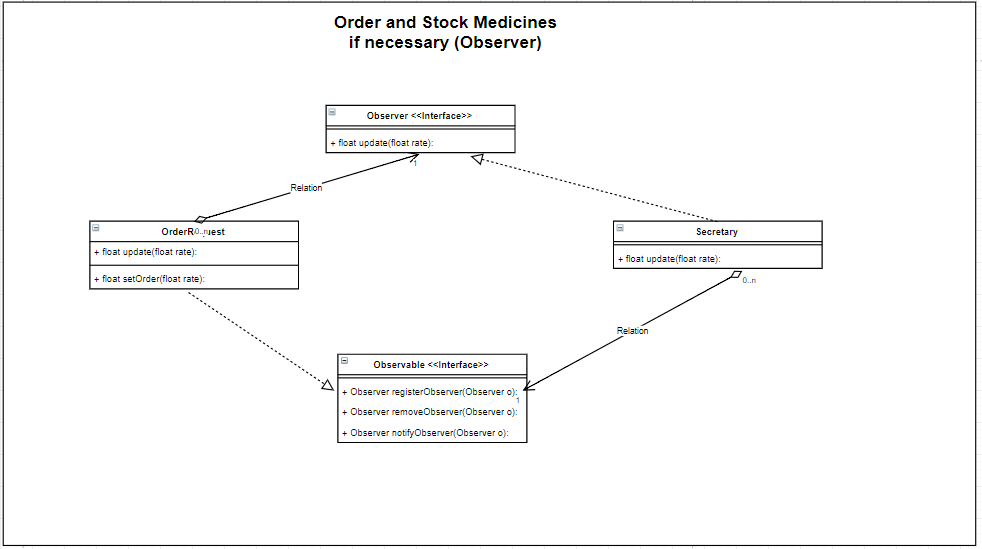


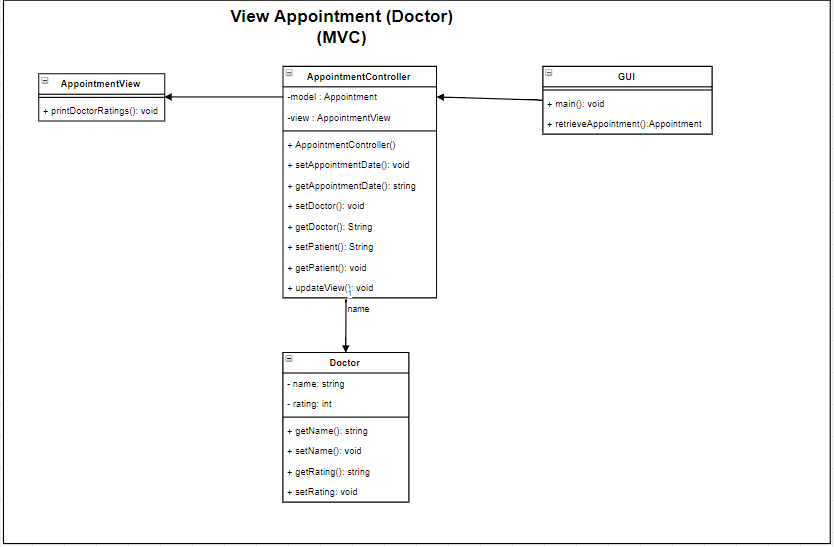


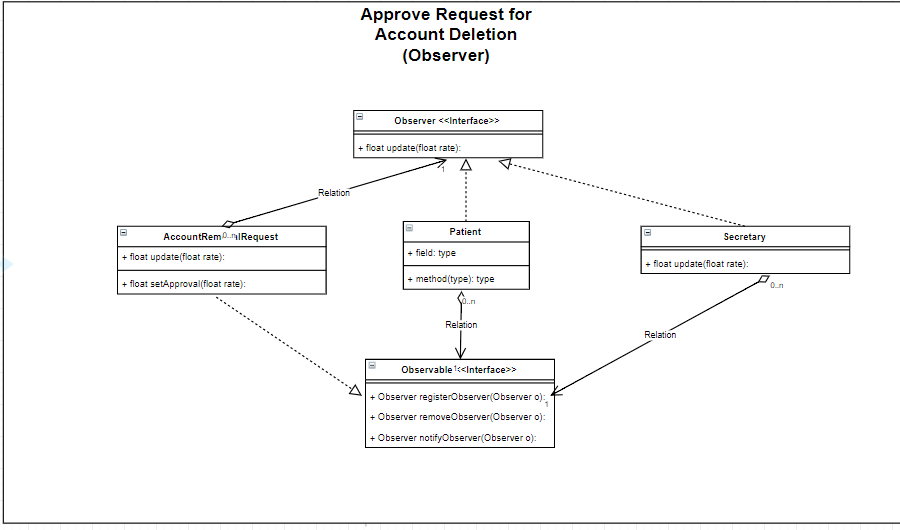


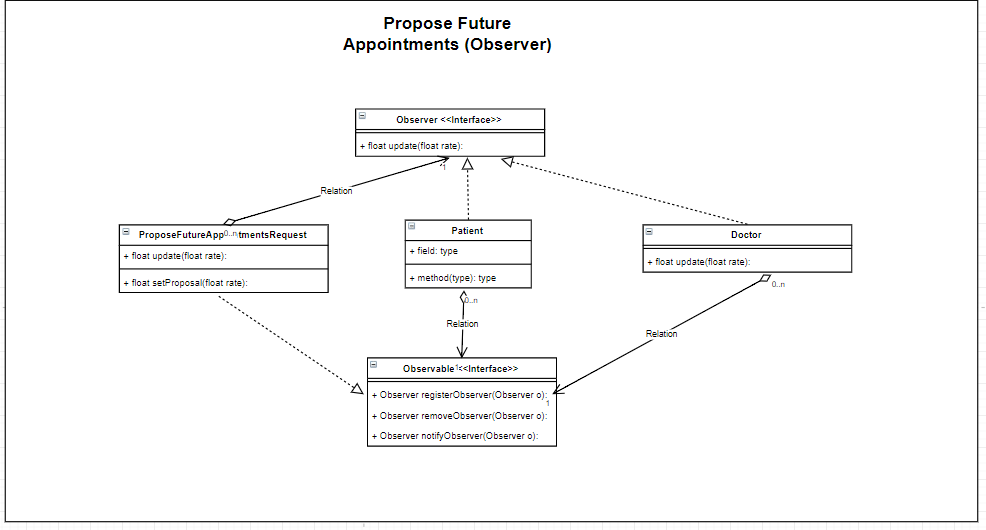


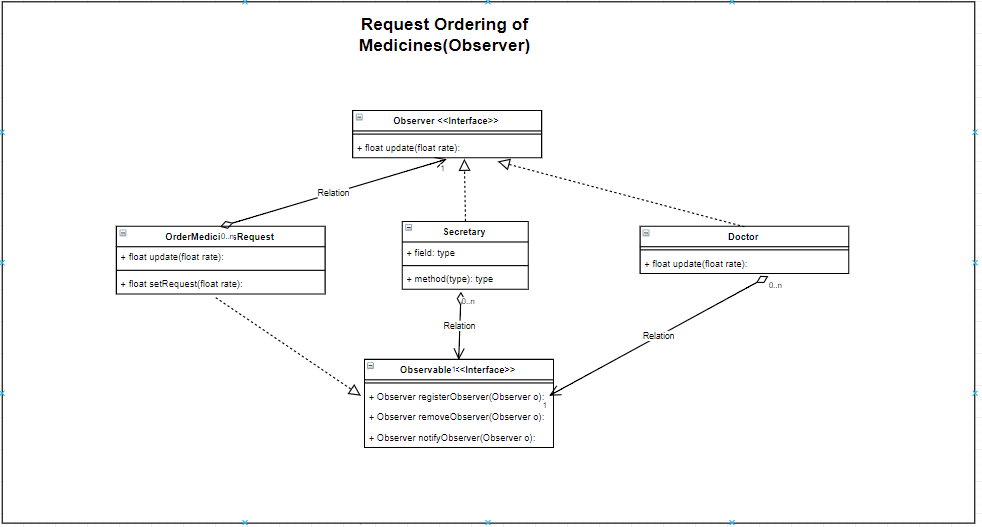












## Design choices I made

I made various different design choices when going about design the Patient Management System.

For the login system, I thought it would be best to implement the Strategy pattern. This is because, depending on which type of user has been logged, the appropriate dashboard page could then be loaded.

For functionality requirements which involved sending requests and messages, I thought it appropriate to use the command pattern, as this pattern deals with heavily with clients and receivers. In the case of requesting a patient account, for example, the patient themselves would be the client, and the receiver of the request would be the secretary. This also applies for requesting an appointment.

When it comes to viewing certain information about specific classes (ratings, appointment details, prescriptions), I felt that incorporating MVC (Model, View, Controller) pattern would be the best option, since it would allow me to access information from a specific model (e.g. Doctor), and print those details in the view (jsp page).

For editing class details, such as for the Doctor’s ratings, I thought it would be best to use decorator pattern, as it would allow me to add custom attributes to the doctor class from a dedicated ratings and feedback class.

For creation purposes, such as creating an appointment or user, such as for creating the administrator account, I thought it would be best to use the builder pattern, as it would allow for the creation of complex classes, such as the administrator account.

When it comes to updating certain users or ordering items, such as when the secretary had to order medicines, or approving account creation or deletion, I decided to use the observer pattern, as the main use of the observer pattern is to update certain subscribers (In these cases the patient and the secretary), and when objects often need to interact with one another.