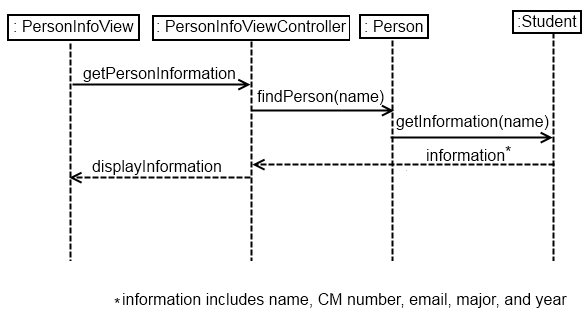
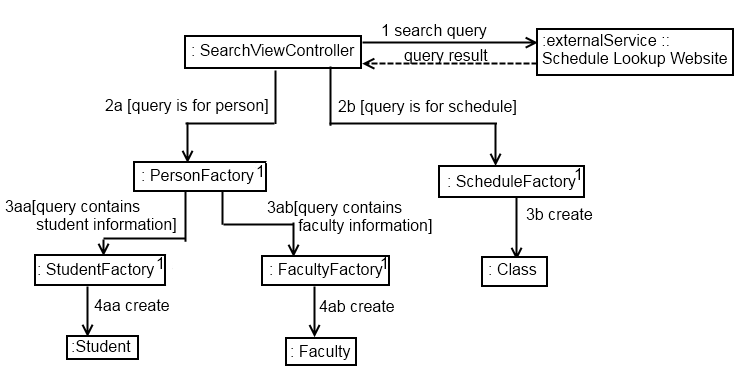
Adapter

Our system scrapes information from the school website and displays this information in a suitable format. These information types are encapsulated in Adapter classes, whose naming scheme comes from the domain model, such as Student, Faculty, and Class. These classes are Adapters because they adapt the raw data brought in from the website into a class with attributes, which can be manipulated easier in the domain.



Abstract Factory

There are various Factories in our design, used to receive and parse raw data from the website, and send the information to the various model classes, such as Student, Faculty and Classes. This way, only a few classes have to deal with html code, and not the individual model classes. This also keeps the cohesion high within the Factory and model classes. The abstract factories are the PersonFactory and the Schedule Factory. The PersonFactory then passes the information to either the StudentFactory or the FacultyFactory, depending on what type the query information is. These sub factories that correspond to classes, such as the StudentFactory and ClassFactory, further transform the parsed data received and create their corresponding classes. We decided to create sub-factories instead of having the high level factories create all the model classes. This was to keep cohesion on the factories high, and instead of having to know how to parse the various results into people or classes or schedules, it sends the information to the specialized factory.



Singleton

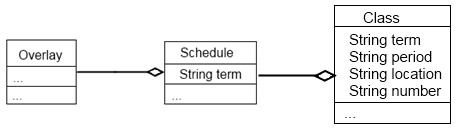
A Singleton class is a class where there is only one instance of it in the entire design model. The various factories used in the parsing of the web page are Singletons, since only one instance is needed to parse the information. This design was chosen instead of having a factory created at every query, because it is uncertain how many searches the user may have in the lifetime of the application. If a new factory was created at every search query, then the application would get cluttered with excess and redundant factories very quickly.

Composite

The overlay is composed of one or more schedules. The schedules are interlaced together so that the class times line up. In essence, an overlay looks like a normal schedule, except with the possibility for multiple classes to appear in a single time slot. Thus, it followed that the overlay should be treated like a single schedule, since like a schedule it is a combination of classes.

Other compositions in the system are the Favorites class - a Favorites list is composed of people - and the Schedule class, which is composed of Courses.

The following class diagram, taken from the domain model, shows the composition relationships between classes.



Observer

Because of the nature of the iOS environment, all of the View classes are Observers, since they are constantly listening for an input from the user, namely a touch on the screen. The response action is dependent on where and how the user touches the screen, such as touching a button, or dragging their finger across the screen in a scrolling manner.

All of the classes in the view architectural layer – ScheduleView, OverlayView, SearchView, FavoritesView, CourseView, SettingsView, and PersonInfoView are observers. They correspond to all of the various views the user of the application would see.

