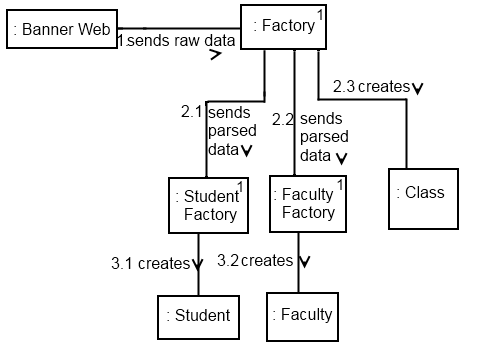
Adapter

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 come 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.

Below is a communication diagram that shows how the Adapter classes are created. The system enters a specific url code to the Schedule Lookup page and the data is sent to the Factory, which then parses the information and sends it to the corresponding classes. An intermediate step is used for the Students and the Faculty as they are both a type of Person, and the data they are fed are similar in structure.



Singleton

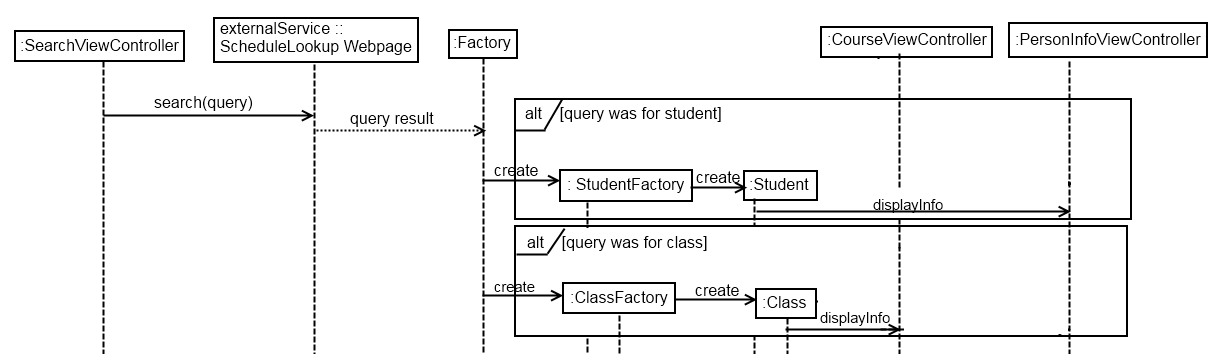
A Singleton class is a class where there is only one instance of it in the entire design model. The Abstract Factory used in the parsing of the web page is a Singleton, since only one instance is needed to parse the information.

As shown in the above communications diagram, there is a 1 in the upper right hand corner of the Factory classes. This indicates that there is only once instance of these classes in the domain model.

Abstract Factory

A Factory is used to receive raw data from the website, parse the information and sent the information to the various model classes, such as Student, Faculty and Classes. This way, only a single class, namely the Factory, has to deal with html code, not the individual model classes. This also keeps the cohesion high in the Factory and model classes. There are also factories corresponding to the classes, such as StudentFactory and ClassFactory that further transforms the parsed data received from the Factory and create the corresponding classes. We decided to create sub-factories instead of just having a single Factory create all the model classes. This was to keep cohesion on the Factory 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.

Below is a sequence diagram showing the sequence for searching for a student or a class. The Abstract Factory named Factory creates the class factories based on the information it receives from the query, and the class factories then create their corresponding classes.

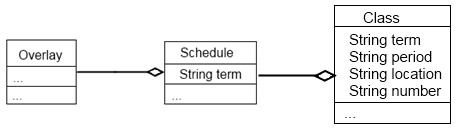


Composite

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.

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 Controller 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 various views the user of the application would see.