Design Patterns

The most significant pattern for this project was **Model-View-Controller**. The Model is the component which stores the data. Our Course, Semester, and Plan classes act as the model since they store the names, credit hours, prerequisites, etc. The Model is independent of the View, which is the Bootstrap user interface and includes the tables for the user to drag-and-drop courses and the arrows between them used to visualize the prerequisites. Lastly, our Executive class is used as a Controller to link the otherwise independent Model and View together, reading data from the Model and responding to user interaction to update the View.

We used multiple **Structural Design Patterns** because they involve composing different objects and classes to realize new functionality. The first structural design pattern we used in our code is the composite, which is used to composite different constructors into a single tree structure to represent the user interface (the graduation plan). Another Structural Design Pattern we used for our code is the decorator which assists us in attaching an additional responsibility to an object. This pattern can allow the constructor to validate the data like monitoring the prerequisite of a course and the maximum number of credits in one semester. Furthermore, an important pattern is the facade which is mostly connected with other patterns and is the method that provides unified interfaces to the set of interfaces in a subsystem, meaning it is responsible for the result of the functionality. We used this pattern with a decorator, this pattern is therefore responsible for giving the result of the validation error. For example, if the user drags the course in the wrong semester it would give an error message to the user.