Integration strategy

We used the **Sandwich Integration Strategy** to integrate the various components of our application. We started by building the low-level classes which are used as the base level data storage objects in our application. The Course class is the most basic. It consists of data about an individual course, including the course code code, credit hours, prerequisites, corequisites, and seasons offered, and has no dependencies on any other classes. We then worked our way up from the bottom, building the Major and Semester classes, which each store lists of Course objects, then ultimately Plan classes which utilizes all of the classes built before it. This is the bottom-up part of sandwich integration, as these classes represent our backend and are furthest removed from user interaction. These components were tested initially using simple drivers, such as hard-coded construction of a test plan instead of it being built via user interaction.

Meanwhile, top-down integration was also being performed simultaneously on the frontend with the ArrowRender and Arrow classes, portions of the Executive class, and also the Bootstrap powered HTML. These components were built long before the backend data storage classes described above were ready, so initial testing was performed using stubs: simplified and hard-coded examples of what these future components would be.

Finally, the key to sandwich integration is where the bottom-up and top-down integrations meet in the middle. For us, this occurred in the Executive class, which linked together the high level user interface with the low-level data storage objects. We drew the line between bottom-up and top-down based largely on the role of the classes – whether they were for the high level frontend user interface or low-level backend data storage.