Design Patterns

For our project we focused more on functionality than form due to time constraints; quick deadlines, large group, program requirements.

We are currently employing a strong MVC design patterns which is used to isolate business logic from presentation. To do this we created Data Transfer Objects(DTOs) which are populated from the CSV parser. These objects are called POJOs in Java and perform a similar function here. They have no other function that to hold the data that is being manipulated by the program. The DTOs are validated and the verification process from the user is performed on them. From here, we created View Objects (VOs) which help to separate the business logic from the presentation. The DTOs are manipulated by separate Populator classes which populate a VO. The VOs are also POJOs and their sole purpose is to hold the data which will be displayed to the user.

We currently use parameters for most of our visualization in the ui, but time permitting an observer pattern will be implemented with a singleton pattern being updated when the ui elements are changed. We didn't directly implement a chain of responsibilities but we would implicitly be using one with a print manager.

Currently we have accessor methods in our DTOs that return the required values (visualization objects) we need for our other class, but we have branched and are working on converting that to a set of Adapter classes. We are not currently using a factory pattern due to limitations with our chosen CSV parsing library. A custom csvparser using some of that library's functionality is currently being created at which point we can refactor our CSVDataAssembler to use a Factory implementation and a Strategy pattern. A Strategy pattern has been used in a non-release branch (merging requires refactoring of other classes).