Isaac Hartung
August 10, 2015
Phase 4 Report

Data Access Object Pattern:

The data access object pattern is merely implementing a class whose sole responsibility is to be the means of communication with a persistence mechanism. This pattern can greatly simplify and secure database communication. The security benefits of the DAO pattern stem from the principle of information hiding: a user need only call methods on the DAO, he does not need to see how a database is implemented nor manipulate it directly. In our project we implemented DAO's for both a sqlite database and a Json file. This resulted in our server calling relatively simple DAO functions and completely ignoring details such as input sanitation, object serialization, etc.

Abstract Factory Pattern:

The Abstract Factory Pattern allows us to encapsulate a group of similar factory classes without knowing their concrete classes. In our phase 4 we had two DAO factories: one sqlite and one Json. Given the specifications of the project, we had to be able to generate one or the other when launching the server; so we needed a way to interface with these two classes. This problem was solved by the AFP because by encapsulating similar factory classes, it also acts as an interface between them and the user.

Plugin Pattern:

Also known as the extensibility pattern, the plugin pattern is meant to make software more able to support added functionality at a later date. In the case of our phase 4 implementation, our server program accepts .jar files (plugins) which contain the needed DAO factories. Given this scheme, should one persistence mechanism prove flawed or a better one present itself, all we would need to do is create a jar containing a factory for that DAO and "plug" it into our server. Thus adding functionality becomes trivial, whereas without this pattern classes would need to be changed, method calls altered etc.