Documentation of Design Patterns

SDJI2 Assignment

For the purpose of this assignment, we hardcoded the linked list to hold the collection of items as one of the delimitations was making a database. However, two of our four design patterns rely on a database connection, that being the Adapter and the Singleton pattern. Here, we will explain how the system would be implemented if the system was made wholly, with a connection to a database, and how would those two design patterns fit there. The explanation is supported by an UML diagram for an easier understanding.

Our library system is divided into model, view and controller packages, making the Model View Controller design pattern. The purpose of the Model View Controller design pattern is to distribute the classes which handle what the user sees, that is the View, from the actual handling of the user’s actions. The user chooses an option from the view, to lend out a book, to return a book, to search for an item and so on, and the action is passed to the controller, which then contacts the collection and returns to the view what was asked.

The Model View Manager may be more difficult to implement than simply binding a couple of classes. However, it is more suitable if one wants to make a system which would be, in the future, easier to maintain and alter. For instance, if we wanted to change how the user sees the system, we would only change the View class, as the View does not know anything about how the system handles the actions. Similarly, if we wanted to alter the action handling, we would only change the manager classes without making any changes to the rest of the system implementation.

The Observer pattern provides

The Singleton pattern ensures there is only one connection to a database at a LibraryDatabase class. When the server model manager would ask the database for some information, the LibraryDatabase would first check whether a connection has already been established, and if not, establish one. This pattern is very simple to implement but very handy. Having more connections to a database would mean using more resources than necessary.

The Adapter design pattern consist of an interface which the user has an instance of, and is implemented by an adapter which has an instance of the class adaptee. In our system, the LibraryDatabase class acts like an adapter, MyDatabase as an adaptee and the LibPersistence interface is the interface the ServerModelManager has an instance of.

We have implemented an adapter between the database and the ServerModelManager to ensure the ServerModelManager gets the information from the database in an expected format. When the ServerModelManager requests information from the database, the LibraryDatabase class, that is the adapter, takes the request and converts it into a command the database understands. The command is then passed to the MyDatabase class, which executes the command and passes the given information to the LibraryDatabase, which converts the information into readable form for the ServerModelManager.

