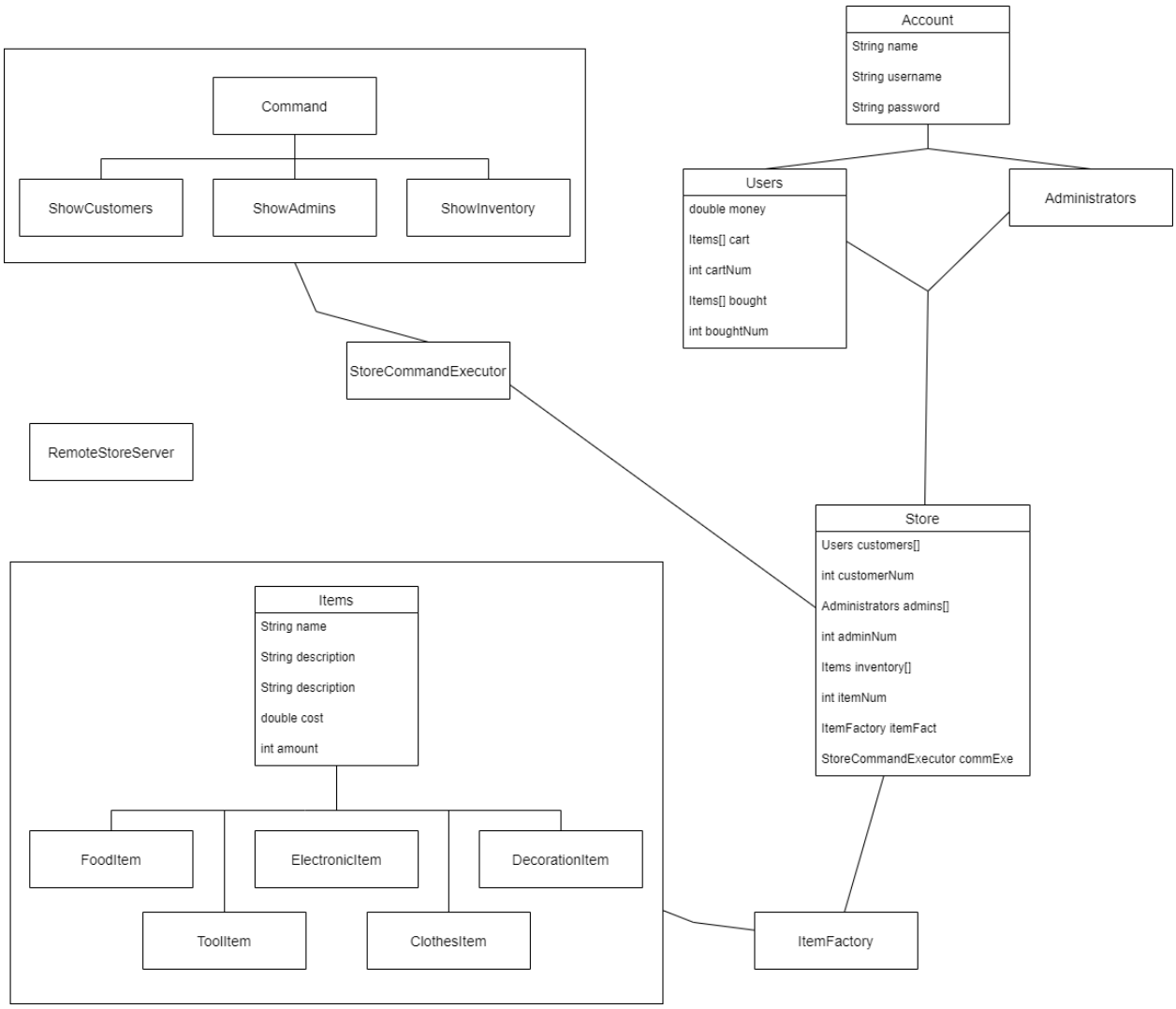
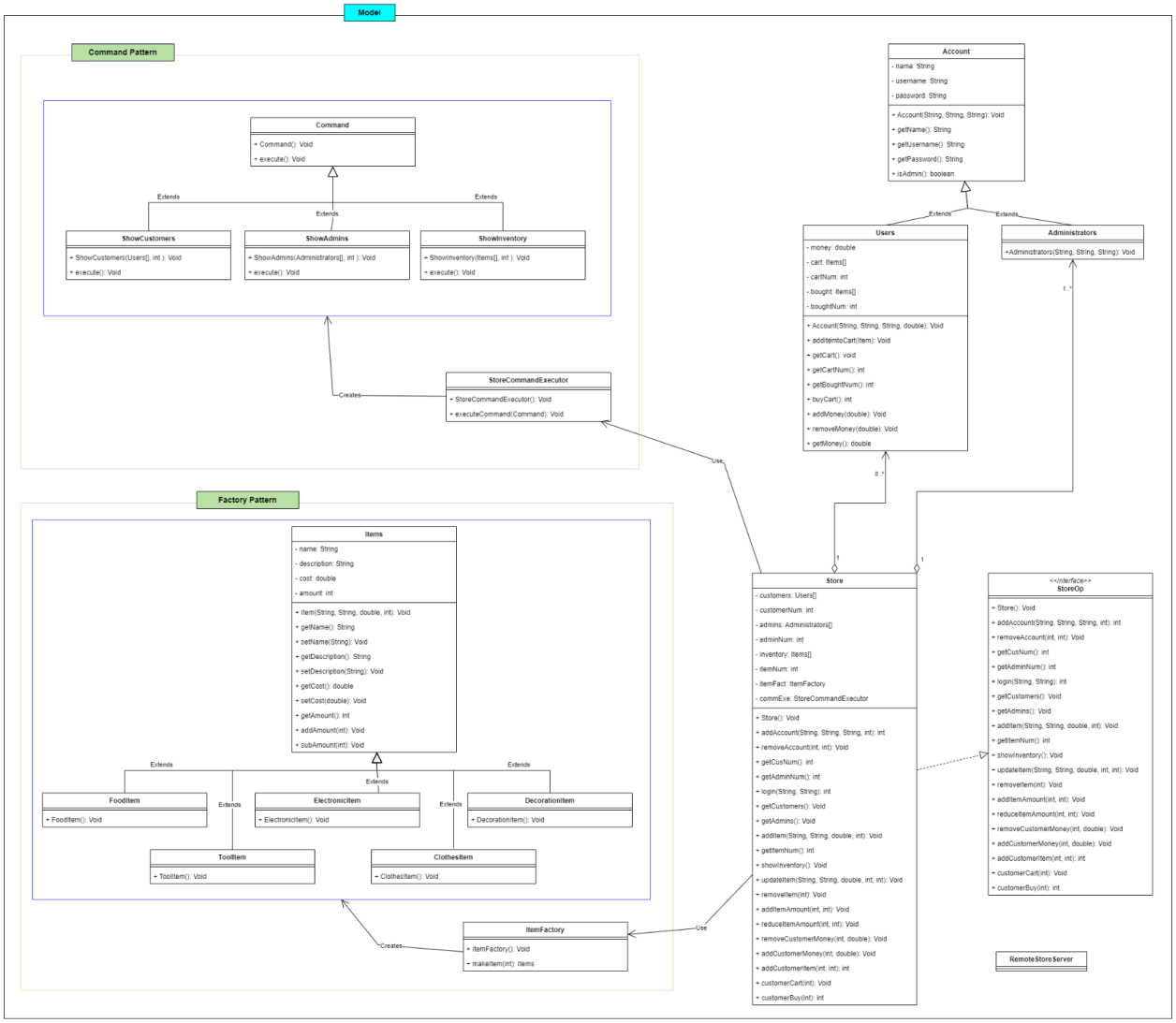
Introduction:

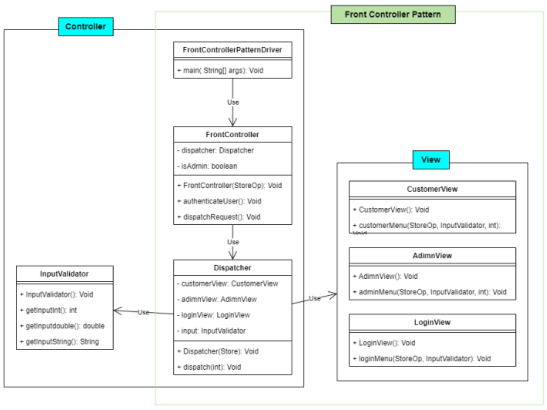
For this assignment we had to add on to our online store from assignment two. This required us to implement three different patterns which were the front controller pattern, factory pattern, and the command pattern. In order to implement these three patterns into my existing code I had to add and modify many classes to the project. In this report I will show the domain model, class diagram, and detail any changes from assignments 2 to 3.

Domain Model:



Class Diagram:





(Light blue labels = MVC sections | Green labels = classes used in specified patterns)

Changes:

As can be seen in the images above many additions were made to the online stores code between the two assignments. The first modification that I will mention is to the view section of the MVC. I decided to split the StoreView class which held all of the store's UI into three classes which are named after the view they provide. Also since the StoreView class was the class that the clients ran to access the remote store object from the server that code was moved to another class in the controller section which I will talk about later on. The reason to split StoreView into three view classes was to better accommodate the front controller pattern which I will now talk about. To implement the front controller pattern I needed to add three classes to the controller section of the MVC. These three classes are named Dispatcher, FrontController, and FrontControllerPatternDriver. The client code to access the remote store object which was moved from StoreView was put into the FrontControllerPatternDriver class. The FrontController class is used to tell the Dispatcher to authenticate the user and get the required views. The Dispatcher class takes the dispatch request from the FrontController class and displays the required views. The last new change in the controller section is that I renamed StoreController to InputValidator.

The model section of the program saw the biggest change because this is where the command and factory patterns were implemented. I am first going to detail the changes made by the command pattern to the program. To implement this pattern five classes were added: Command, ShowCustomers, ShowAdmin, ShowInventory, and StoreCommandExecutor. The Command class is the parent class to the three Show__ classes and it has the execute method which will be overridden by the child classes. The three Show__ classes override the parent execute method to print their respective area of the online store. The StoreCommandExecutor class takes the commands that the online store wants done and executes them. For this pattern to work the store needed to add a StoreCommandExecutor object to execute the Command objects the store needs executed.

The factory pattern was the second large change to the model section and it added six classes to the program. These classes are the five child classes of Items and the ItemFactory class. In this assignment the Items class is the parent class used in the factory pattern and its five child classes make generic products based off of their class name. The ItemFactory class takes the store’s request to add a new product, makes the desired item, and then returns the item to the store. The store class needs an ItemFactory object in order to make the factory pattern work.

Additional:

I was unfortunately unable to get the JAR file to work correctly after multiple attempts to do so. Not having the time to find a fix for this I zipped all the files up in a standard zip file. Sorry for the inconvenience