Object-Oriented Design & Pattern

Assignment 3 - Report

- Jitesh Gugan Sreeram
  Archith Krishnan.S.R

Table Of Contents

1. Objective

The objective of this Assignment is to provide a skeleton framework for Online Store make a domain model, and classes necessary for MVC design pattern and also implement Java RMI for the project.

2. Primary Goals

2.1 Role of Front Controller Pattern

The front controller is the class which accepts all the requests from the client. Every communication with the server will pass through this controller. It is in this controller too that the lookup is done. As soon as the customer/administrator enters his user credentials, the details are passed on to the front controller for validation by the view. The front controller directs it to the application controller, for user validation. After validation, depending upon the validation, this login handler sends it to the dispatcher from processing the appropriate view.

In our Project, The files which are Front Controllers are FrontController.java and FrontControllerImpl.java and the Dispatcher would be ModelImpl.java

2.2 Role of Command Pattern

In Command Pattern, a command is designed in a way, that it tells the receiver to work on an Action(). And this command on executed using the execute() method, will ask the receiver to perform the action. The command's execute() method is said to invoke this method. The invoker will have a list of commands, and it will start executing them one by one. However, in this project, there is only one command that is the "customerLogin" or "adminLogin" command.

There is an interface Command.java which has an execute() method. The concrete command classes, customerLogin() and adminLogin() implements the Command Interface. In order to execute customerLogin() and adminLogin(), their receivers have to be passed. The receiver will be stored in their instance variables. When the invoker starts executing the execute command of these objects, the execute() has code to make the receiver perform some action. In the project, FrontControllerImpl.java is the invoker.

2.3 Role of Abstract Factory Pattern

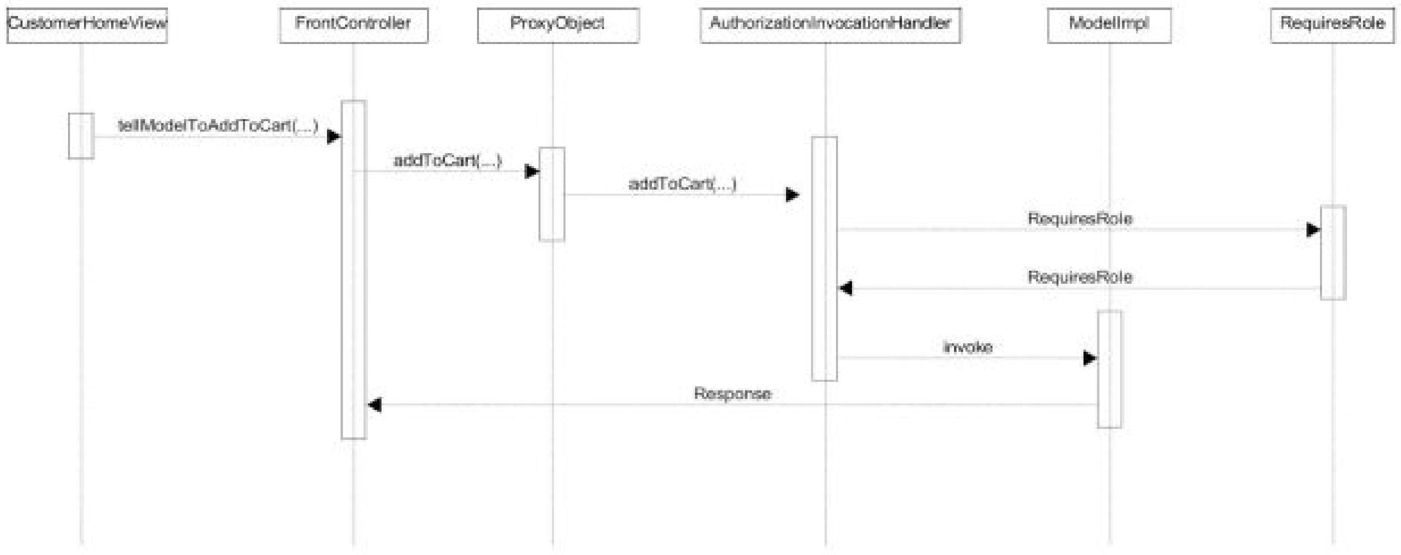
In the Abstract Factory Pattern, we can create a group of factories that can be used to create related objects. In this project, the abstract factory is used to create views. The FactoryCreator.java has the code to create an object of AbstractFactory. The Abstract Factory is an interface which is implemented by concrete classes,View.java and Model, FrontController java files. The View Factory understands the requirement based on the parameter sent “view description” while requesting a view. As the appropriate view is received, the show method is executed.

2.4 Role of Authorization Pattern

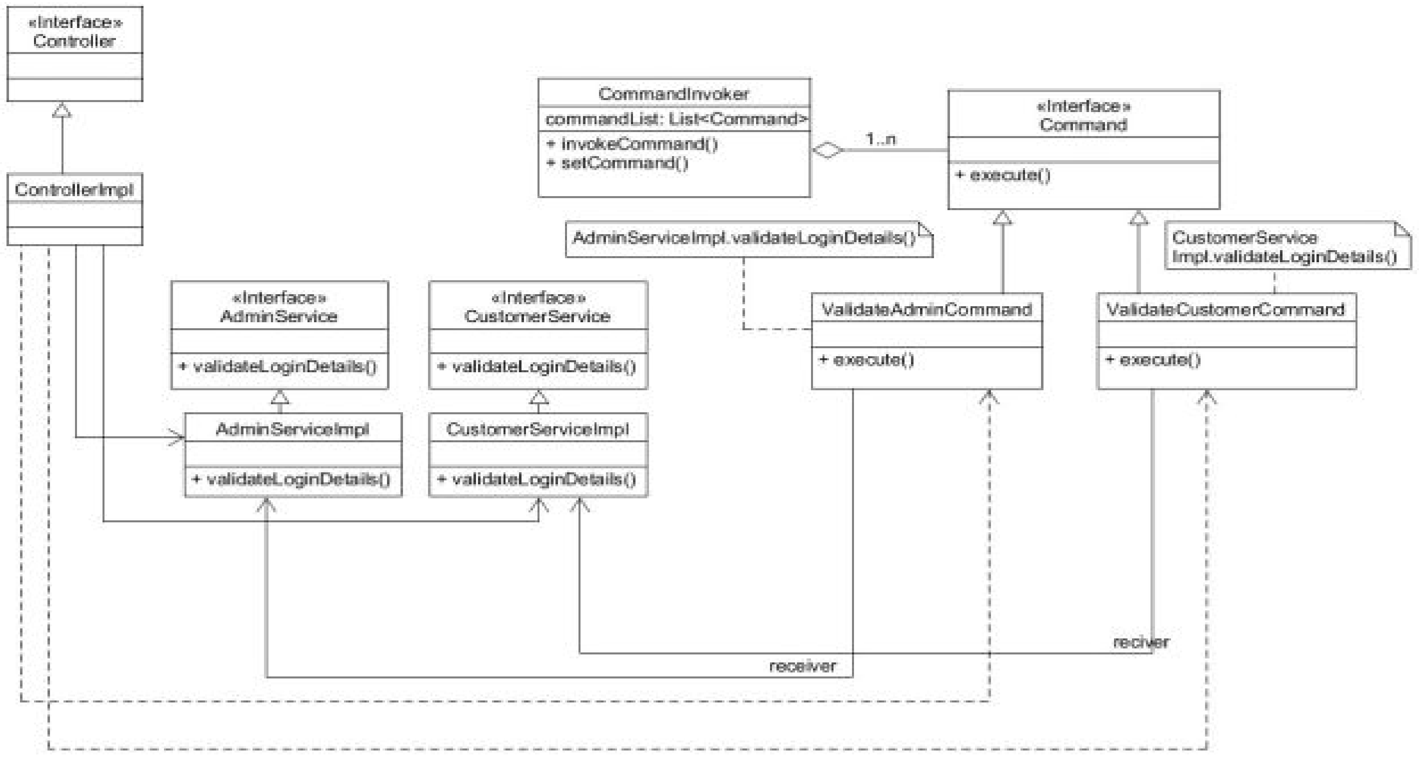
The authorization pattern is used to provide role-based access control. That is few methods are custom to particular roles for example., an admin can’t access can’t add to cart and purchase. Only a customer can add to a cart and purchase. Therefore, when he is prompted with a “not authorized” message when he tries to add a product to cart.

3. UML Diagrams

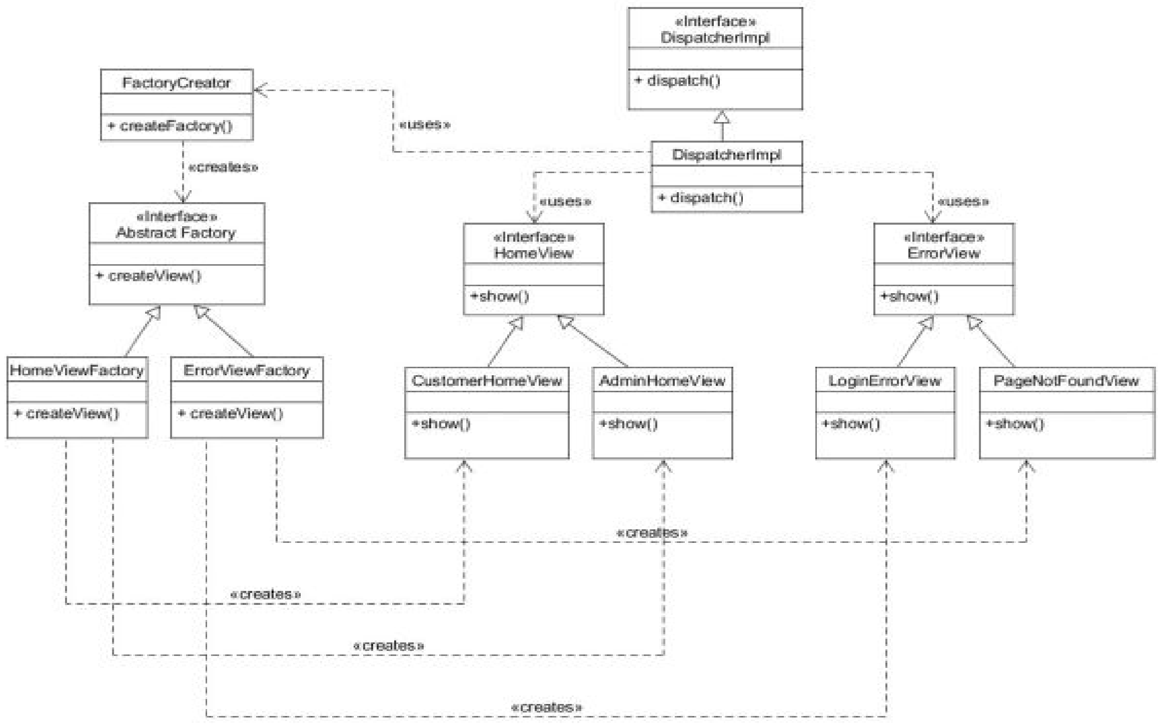
UML Sequence Diagram



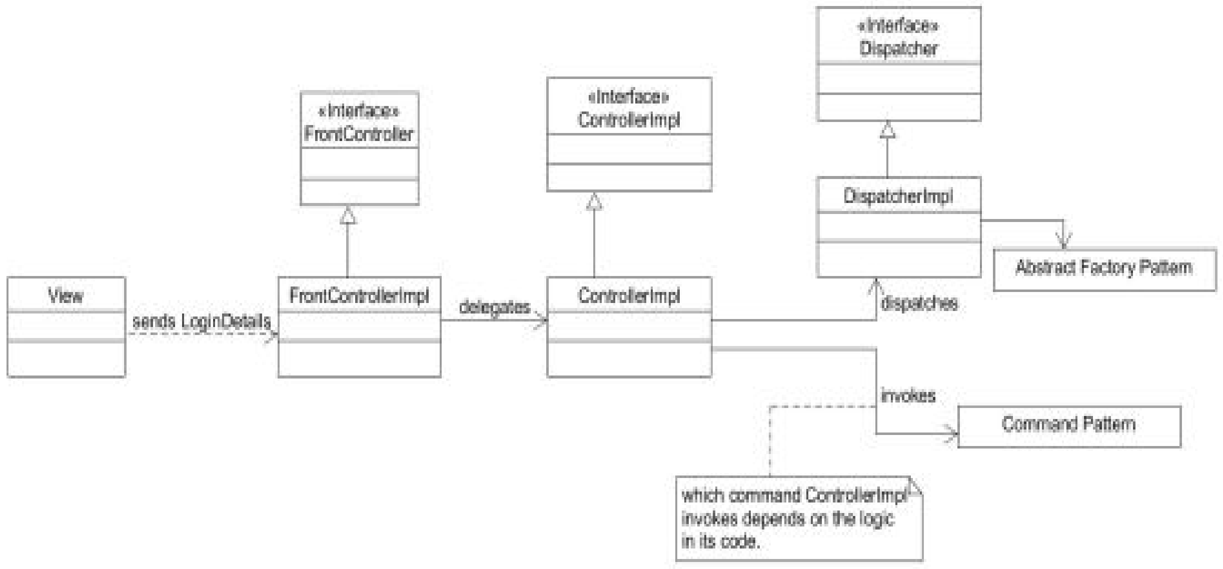
UML Class Diagram for Command Pattern



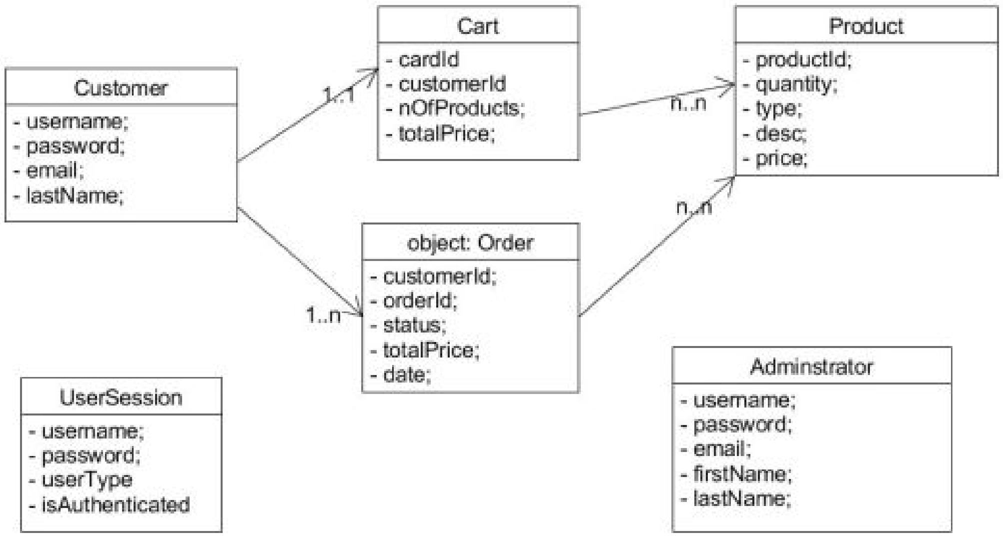
UML Diagram for Abstract Factory Pattern



UML Diagram for Front Controller



UML Domain Model



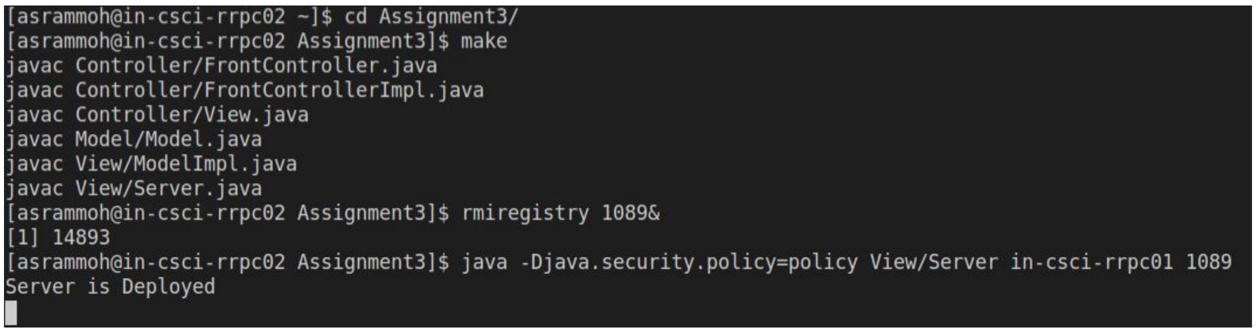
4.Transition From Assignment 2 to Assignment 3

New classes created

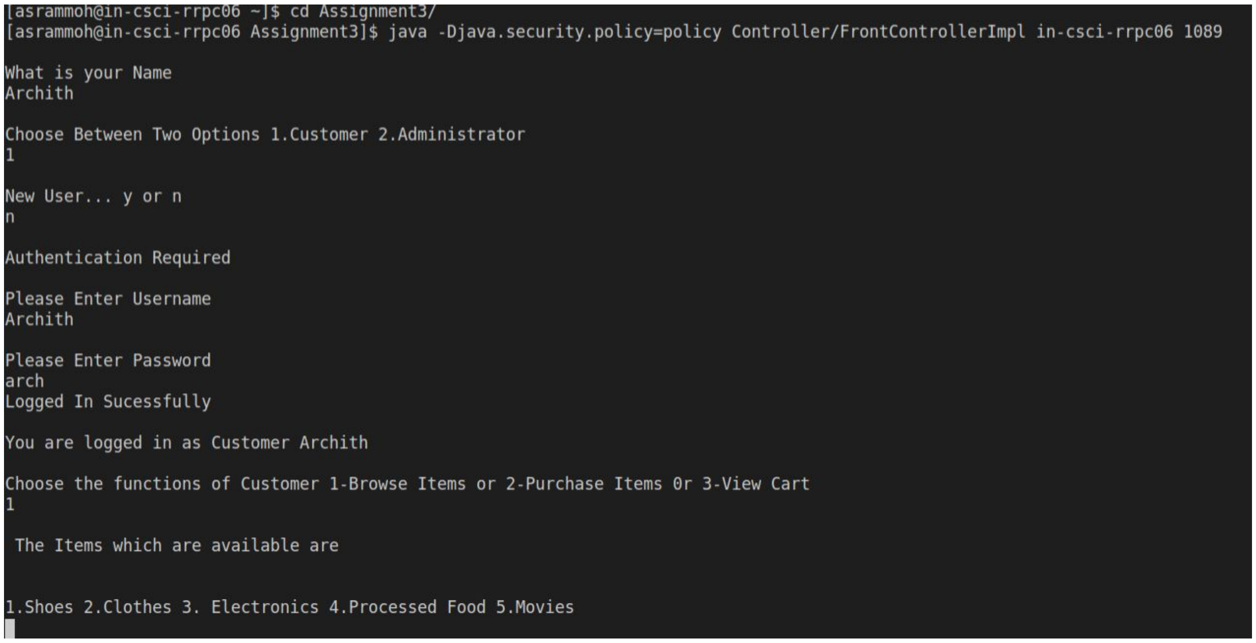
Classes which are newly added are FrontController.java and FrontControllerImpl.java which implement the Front controller pattern functions. Added Login Authentications functions which follows the Authorization pattern. The pattern is implemented in the dispatcher class itself and is not implemented as a separate class. Exception Handling is very much improved in this Assignment. Added a policy file that grants permission to run and execute all the functions.

5. Sample Runs

Server

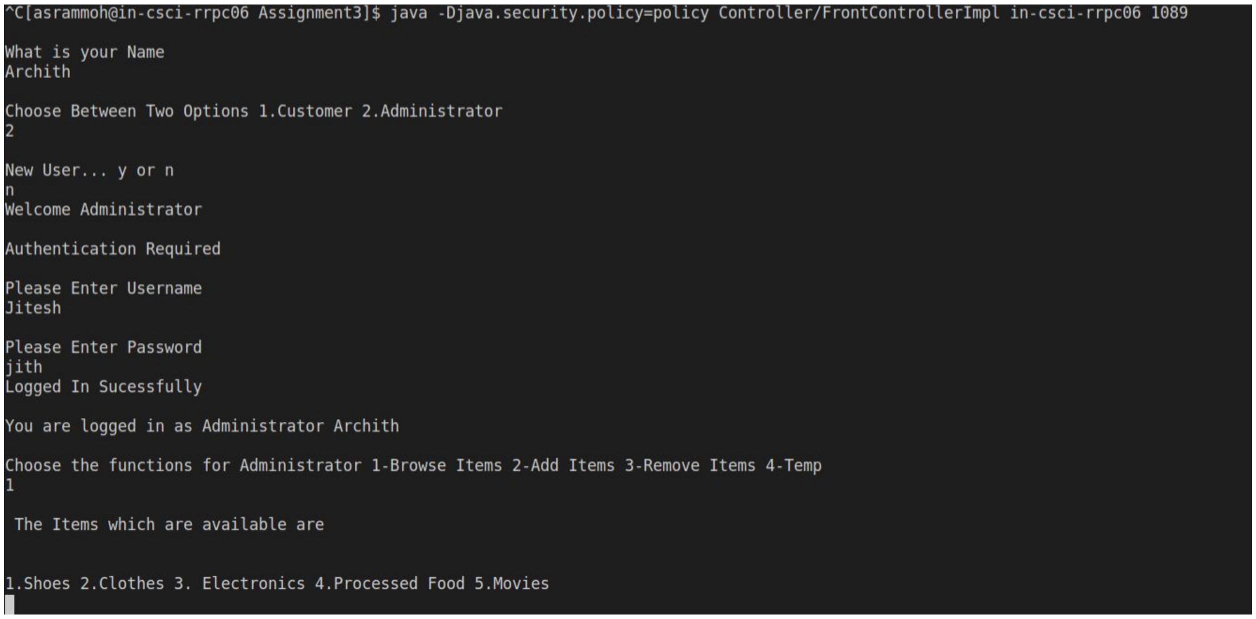

```
[asrammoh@in-csci-rrpc02 ~]$ cd Assignment3/
[asrammoh@in-csci-rrpc02 Assignment3]$ make
javac Controller/FrontController.java
javac Controller/FrontControllerImpl.java
javac Controller/View.java
javac Model/Model.java
javac View/ModelImpl.java
javac View/Server.java
[asrammoh@in-csci-rrpc02 Assignment3]$ rmiregistry 1089&
[1] 14893
[asrammoh@in-csci-rrpc02 Assignment3]$ java -Djava.security.policy=policy View/Server in-csci-rrpc01 1089
Server is Deployed
```

Client - Customer



```
[asrammoh@in-csci-rrpc06 ~]$ cd Assignment3/
[asrammoh@in-csci-rrpc06 Assignment3]$ java -Djava.security.policy=policy Controller/FrontControllerImpl in-csci-rrpc06 1089

What is your Name
Archith

Choose Between Two Options 1.Customer 2.Administrator
1

New User... y or n
n

Authentication Required

Please Enter Username
Archith

Please Enter Password
arch
Logged In Sucessfully

You are logged in as Customer Archith

Choose the functions of Customer 1-Browse Items or 2-Purchase Items 0r 3-View Cart
1

 The Items which are available are


1.Shoes 2.Clothes 3. Electronics 4.Processed Food 5.Movies
```

Client - Administrator



```
^C[asrammoh@in-csci-rrpc06 Assignment3]$ java -Djava.security.policy=policy Controller/FrontControllerImpl in-csci-rrpc06 1089

What is your Name
Archith

Choose Between Two Options 1.Customer 2.Administrator
2

New User... y or n
n
Welcome Administrator

Authentication Required

Please Enter Username
Jitesh

Please Enter Password
jith
Logged In Sucessfully

You are logged in as Administrator Archith

Choose the functions for Administrator 1-Browse Items 2-Add Items 3-Remove Items 4-Temp
1

 The Items which are available are


1.Shoes 2.Clothes 3. Electronics 4.Processed Food 5.Movies
```

6. Conclusion And Analysis

With this assignment, using design patterns such as Abstract factory pattern & Authorization pattern on the ModelImpl is provided for the front controller to interact. Also, only a user role with appropriate role can communicate with the ModelImpl method. This is implemented using the Authorization pattern using Java Annotations to provide role-based access control.

7. References

1. Front Controller Pattern: Front Pattern Example in Canvas
2. Abstract Factory Pattern: Abstract Pattern Bank Example in slides.
3. Authorization Pattern, RMI annotation, policy from example in Canvas.
4. Technologies used: Tesla, Java, Linux terminal commands with Scp and Ssh.