Online Market Place: Pattern-based Design

CSCI 50700 – Object-Oriented Design and Programming

Assignment-II

Front Controller, Abstract Factory Design and Command Pattern

Summary Report

Under the guidance of

Professor Ryan Rybarczyk

Computer & Information Science, IUPUI

By

Yashwanth Reddy Kuruganti

Computer & Information Science, IUPUI

Table of Contents

1. **Introduction** 3
2. **Assignment 1 Feedback** 3
3. **Front Controller, Abstract Factory and Command[2]** 3
   a. **Front Controller pattern** 3
   b. **Abstract Factory Pattern** 3
   c. **Command Pattern** 4
4. **UML Diagrams** 4
5. **Sample Run** 5
6. **Conclusion** 7

1. Introduction

This assignment aims at further constructing market place application using front controller, abstract factory and command pattern using a skeleton framework for Online Market place application, which is built in 1st assignment.

2. Assignment 1 Feedback

For the first assignment, I was asked to separate the RMI framework and MVC pattern logic. To fix this, I have now created two controllers MarketViewController and OnlineMarketController. View Controller acts as an interface between Client-side view and Server-side controller. MarketController acts as interface between client-side controller and server-side business logic or model. Both these controllers now provide RMI communication for the whole market place application. Doing this way separates the framework, presentation and business logic.

3. Front Controller, Abstract Factory and Command[2]

This assignment mainly focuses on creating a generic login page through which both admin and a customer can login and access their specific views. Also, customer and admin can perform various commands like browse, update and delete etc., First design pattern implemented is front controller pattern. I have implemented these by understanding and referring InClass examples. [1]

a. Front Controller pattern

For a distributed online application, unauthorized access should be denied. Also, there could be many types of users for whom they should only see the things that they have access to. To achieve this functionality, front controller pattern would be more suitable. In our application, customer and admin should be provided with separate views as they have different options to operate. MarketFrontController is responsible for generating views for a user and provides user authentication. Initially MarketCommonView provides a common interface for a user (admin or customer) in which he enters login information. This is captured by the MarketViewController which calls FrontController to generate view respective to login type. This client controller transmits this login data to Model via Server Controller using RMI for validation. Now FrontController, based on the response from server it authenticates and sends a request to MarketDispatcher which generates a view OnlineMarketAdmin or OnlineMarketCustomer.

b. Abstract Factory Pattern

This pattern is a form of Creational design pattern which helps in creating a factory that generates objects that may further create concrete items. In simple terms, this pattern creates a parent factory which can generate several sub-factories. This pattern could be useful to invoke a specific view, if there are many types of users or admin i.e., user may be of type prime (like amazon) or non-prime or admin may be a root admin or less privileged admin. OnlineMarketCustomer and OnlineMarketAdmin are two concrete views which are implemented using MarketCustomerInterface and MarketAdminInterface. Two factories MarketAdminFactory

and MarketCustomerFactory are created which inherits from a common MarketAbstractFactory. Each of these factories creates a Customer and Admin factory respectively. MarketDispatcher is now responsible for creating factories based on the information passed to MarketFactoryCreator.

c. Command Pattern

Command pattern is a type of Behavioral pattern which helps to create and execute various commands or actions several times. In this pattern, there would be a invoker object which plays an important role in looking up for appropriate object that can execute the requested command. For my application, I have scope of creating command pattern to implement customer as well as admin options. But as of now, I have only implemented this for customer side. If this implementation seems feasible, I would follow the same to implement admin functionalities. CustomerCmdInterface acts as a command and OnlineMarketCustomer view acts a request. Implementing the command interface, there are two concrete classes BrowseMarketItems which allows a customer to browse the items database and PurchaseMarketItems which helps the customer to purchase interested items. These classes would perform the commands as they are received. But before these can process the commands there should be an intermediate class CustomerInvoker, which acts as invoker. Based on type of command, invoker object determines which object can perform a given command.

4. UML Diagrams

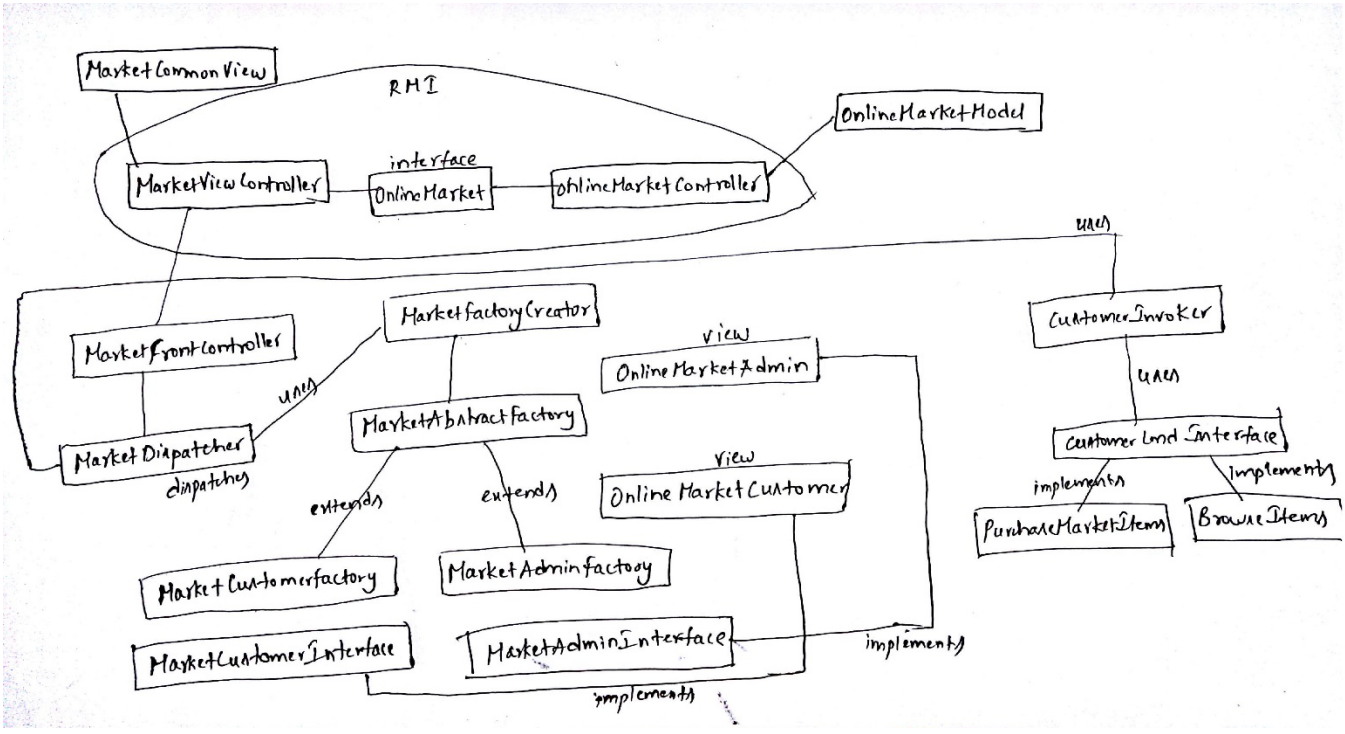

Fig 1. Domain Model

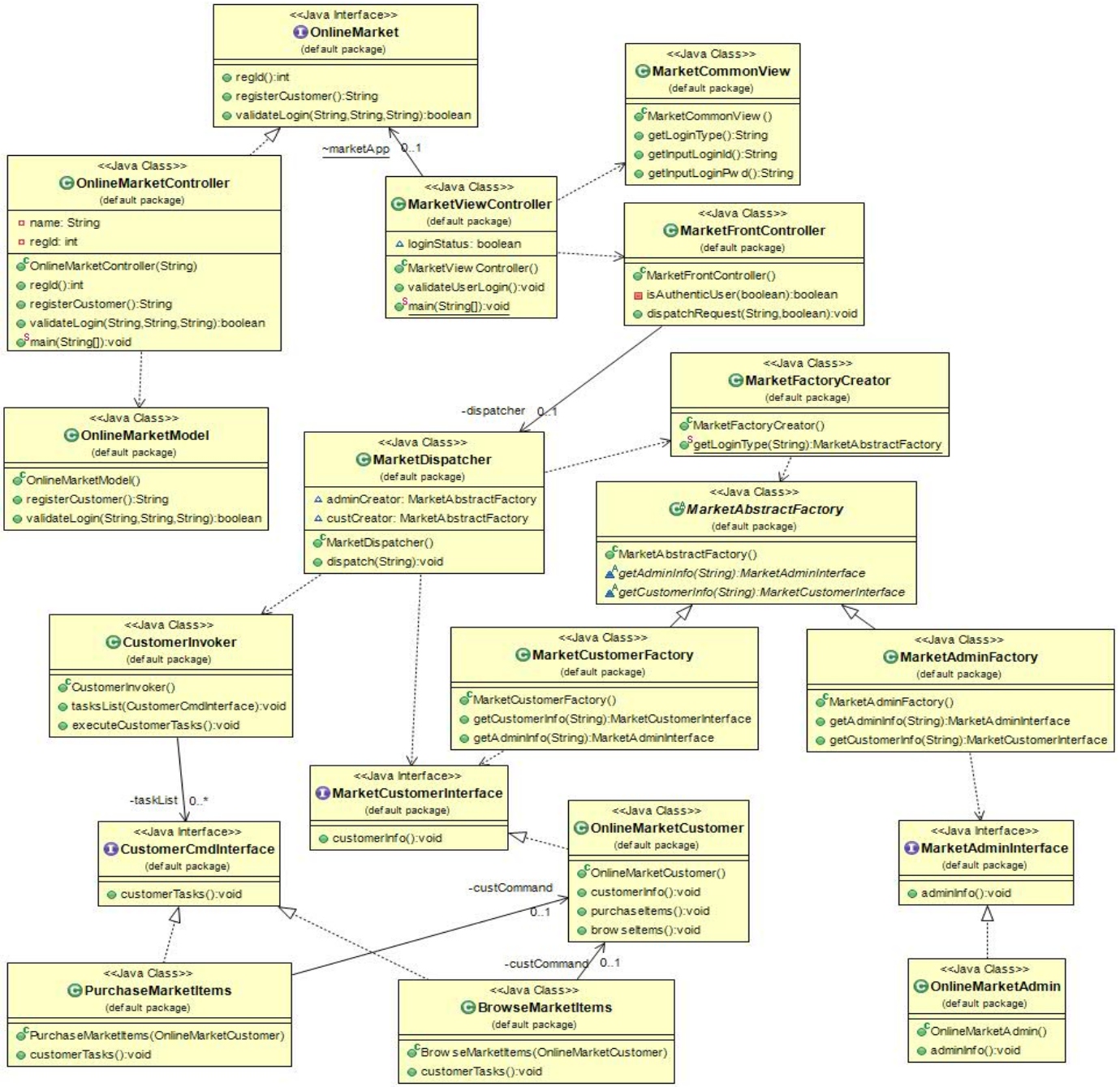

Fig 2. Class Diagram

5. Sample Run

Steps on running this software is given in Readme file. Simple way of executing is to run makefiles.

- Running RMI registry and server makefile

```
[yashkuru@tesla csci50700_spring2018_marketplace-master]$ rmiregistry 5432&
[1] 161794
[yashkuru@tesla csci50700_spring2018_marketplace-master]$ sh makefileS.sh
You are now entering Online Market Place
Reaching server://tesla.cs.iupui.edu:5432/onlineMarketServer
Interface is Ready!You can register, login and shop
Registration page. Register here
Success
Registration page. Register here
Success
Registration page. Register here
Denied
```

- Running client side make file

```
[yashkuru@tesla csci50700_spring2018_marketplace-master]$ sh makefileC.sh
Registration ID: 2
Registration Status: Registered
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Enter 'Admin' for Administrator login without quotes
Enter 'Customer' for Customer login without quotes
------------Enter one from above------------------
customer
Enter Login ID:
customer
Enter Password:
test
You are now accessing market application as: customer
-----------------------------------
-Welcome to the Customer Home Page-
-----------------------------------
Hi! You have the following commands to perform
---Enter 'Browse' ignoring quotes to shop
---Enter 'Purchase' ignoring quotes to buy items
browse
======Browse Market App to shop======
<-Your shopping items list here->
[yashkuru@tesla csci50700_spring2018_marketplace-master]$ sh makefileC.sh
Registration ID: 3
Registration Status: Registered
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Enter 'Admin' for Administrator login without quotes
Enter 'Customer' for Customer login without quotes
------------Enter one from above------------------
customer
Enter Login ID:
cust
Enter Password:
test
Authorization denied for user type: customer
[yashkuru@tesla csci50700_spring2018_marketplace-master]$
```

6. Conclusion

From this assignment, I have learnt and understood how various design patterns like MVC, Front Controller, Abstract factory and Command pattern can be used to build an online application and how these can be integrated with each other for reliable running of the application. Also got acquainted with dispatching to multiple views based on the input request, various commands that a customer or admin can perform and generate various factories for an admin or customer.

7. References

[1] In-Class examples

[2] https://www.tutorialspoint.com/design_pattern