# Online Market Place: Pattern-based Design

CSCI 50700 – Object-Oriented Design and Programming

Assignment-VI

Implementation of Database Access Layer &

Design Patterns Overview

Summary Report

Under the guidance of

Professor Ryan Rybarczyk

Computer & Information Science, IUPUI

By

Yashwanth Reddy Kuruganti

Computer & Information Science, IUPUI

# Table of Contents

## Introduction

The six assignments aim at building a reliable and easily manageable distributed online market place application by making use of object-oriented design patterns and frameworks. This way of building an application helps in improving software quality and reusability. This sixth assignment implements database access layer pattern which helps to separate the application logic from database(MySQL) access and queries. Along with this, all the rest functionalities should be accomplished to make the online market application fully functional. Following section gives an overview of all the assignments and various patterns[1] involved in each of them.

## Overview

### Assignment #1 Overview - MVC and Java RMI

Each software application will have both a front-end which act as a user interface and a back-end which acts as server for business logic and data storage. Instead of combining all these modules, maintaining them independently helps in lowering the coupling among components. As a result, even though the user interface changes as per client requirements, logic remains same in the background. This nature can be implemented in any software application involving user interface using Model-View-Controller software design pattern. In this application, user interface i.e., View provides interaction for admin to manage customers, other admins and items and customer can register, login, view products, purchase items and manage his/her profile. But all these actions or events when performed on the interface, this triggers the respective functions in Model and processes the data. Before this reaches the model, controller acts as intermediate to propagate two-way messages.

OnlineMarketView component provides methods that run when a customer or admin interacts with the application. Interface provides a means of presenting the model's data. Each time a user interacts with the application, view sends a request to controller, which then forwards this to the model. All the manipulations occur in model and changes as based on type of request send by view. OnlineMarketController acts as glue between market's View and Model. Though these look dependent on one another, all the three components can function independently. To further separate the concerns another layer called MarketViewController is added in between View and server side controller.

From this assignment, I have learnt and understood how Java RMI works and the way its implemented to provide reliable communication between a client and server. Also, MVC helped me to separate layers of code and user interface from server implementation. Integrating RMI along with MVC is a bit challenge, but this skeleton structure seems to help in modularizing tasks in future assignments to implement all the business logic and user interface.

### Assignment #2 Overview - Front Controller, Abstract Factory Design and Command Pattern

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.,

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.

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. 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.

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 an invoker object which plays an important role in looking up for appropriate object that can execute the requested command. CustomerCmdInterface and AdminCmdInterface acts as a command and OnlineMarketCustomer and OnlineMarketAdmin view acts a request. Implementing the command interface, there are several concrete classes like BrowseMarketItems, PurchaseMarketItems etc., 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.

From this assignment, I have learnt and understood how various design patterns like 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.

### Assignment #3 Overview - Authorization Pattern, Reflection and Proxy Pattern

This assignment mainly focuses on creating a role-based access control to the application's functional modules instead of the generic login page built in previous assignment. Through this a user with either of the roles, admin or a customer can login and access their specific views. Also, customer and admin can perform various commands like browse, update and delete etc., I have implemented Authorization pattern using Java annotations along with two more patterns Reflection and Proxy using Java reflections. These patterns are integrated with Command pattern and Front controller pattern to implement the annotations interface and extend the functionalities such that customer

can't access admin functionalities and admin may or may not access customer functionalities. So, role is given importance in this build instead of an individual user.

First step I have done is to implement a Session class and checked if it can be transferred over the application from the entry point. After successful creation, I have tried passing the session object using serialization. Next step is to provide user login through which a customer or admin can be granted a session after successful authentication. FrontController provides the feature of user authentication. Logic for this user authentication is written in OnlineMarketModel. When a user enters his/her details front controller takes care of this and then send a request to client-side controller(MarketViewController) and then to server-side controller(OnlineMarketController). Model contains the logic for user verification. This checks for authenticity and then send back response to client-side front controller through RMI.

Now, to provide role-based access control, AnnotateInterface class is created with user defined annotations which contains a method for storing annotated value over a session. Next required step is to convert the current RMI proxy to more dynamic proxy which can handle incoming requests in annotations environment. AuthenticateUserInvocationHandler contains code for session value and annotation value check and call respective methods based on the annotation value. If the values don't match then these errors should be caught, which is done by using a user defined exception class AuthenticateUserException. OnlineMarketController is changed in such a way that it can handle the requests using a Java dynamic proxy. Other classes like MarketDispatcher, MarketFrontController and MarketViewController are modified to support session transfer and perform user actions in that given session.

From this assignment, I have learnt and understood how various design patterns like Authorization, Reflection and Proxy pattern can be used to provide role based accessing to a distributed application instead of assigning rights to each user. Also gained knowledge on creating and transferring a session through the network and maintaining its life when a user is still acting on the application. Java dynamic proxy instead of RMI made the communication more robust.

### Assignment #4 Overview - Concurrency and Add, Browse and Purchase items

Till assignment 3, various design patterns have been implemented that improves the functionality and modularity of the market place application. In this one, admin related tasks like add items, browse items and customer related tasks like browse and purchase items have been implemented with the help of MySQL database. Along with this, I have observed how Java RMI tries to implement concurrent behavior through multi-threading in my application with the help of five clients and a single server hosted on in-csci-rrpc01.cs.iupui.edu.

### Assignment #5 Overview - Synchronization and its patterns

Previous assignment ensures that market application exhibits concurrent behavior with the help of Java RMI. Even though the application is concurrent, in a multi- threaded application several threads try to access a specific resource or resources at the same time. Access to specific resource at the same time by two or more threads may result in unexpected results for other threads. This is where concept of synchronization comes into play. Java provides synchronization by making use of synchronized keyword. This synchronized keyword is related to few design patterns which helps the

application to behave both concurrently and synchronously. Synchronization can be provided by making use of either synchronized blocks or synchronized methods. In my market place application to provide this feature, I have made use of both the blocks and methods depending on the critical usage of that method. So, if there are multiple blocks are trying to access a specific object, synchronized keyword restricts to run only one thread making use of that object. All the other blocks remain in wait state till the other process releases the lock on that object. Patterns implemented are Monitor object, Future, Guarded Suspension, Thread-safe interface and Scoped locking.

## Database Access Layer Pattern

For a huge distributed application to function well and act as per user actions, there should be a persistent storage to keep track of user details, item details etc., In this assignment MySQL database which is on in-csci-rrpc01.cs.iupui.edu server is used as storage. Writing all the application related logic and database queries logic in the model makes the application avoid separation of concerns and reduces code reusability. So, to avoid this, I have made use of layers pattern to implement database access layer between the application layer and database layer. This way of implementing separate layer ensures that model logic avoids any database related functions, thus reducing coupling and improving cohesion[1].

SqlConnection: This class contains all the information regarding database like connection string, database driver, database user id and password. Creating an instance of this class can help an application to connect to the required database.

DbAccess: This class contains all the querying logic such as retrieve item Id, retrieve customer details, insert a record to items/customers/admin table, update items, delete items/customers etc., Obtains connection details from SqlConnection class.

OnlineMarketModel: This class retrieves the query result from the DbAccess class and further processes it along with application logic.

## UML Diagrams

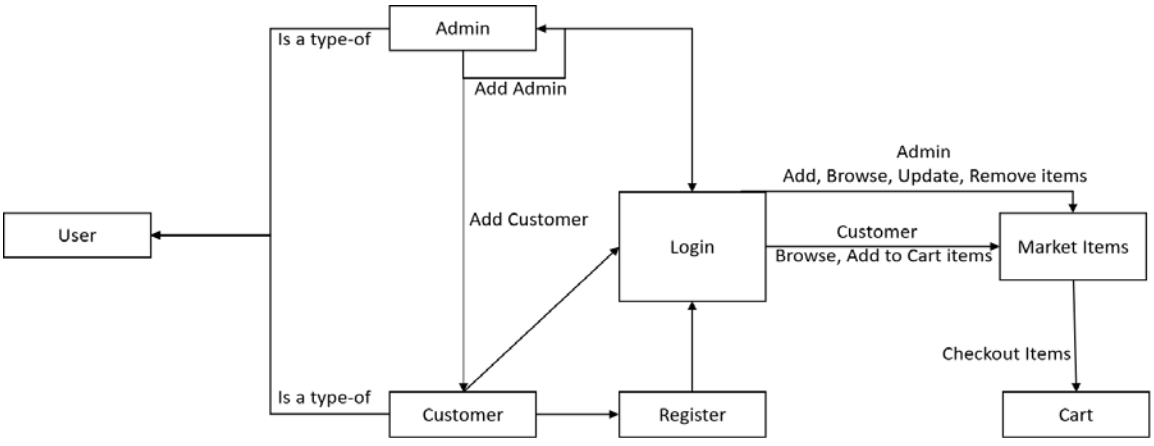


**Fig 1. Domain Model**

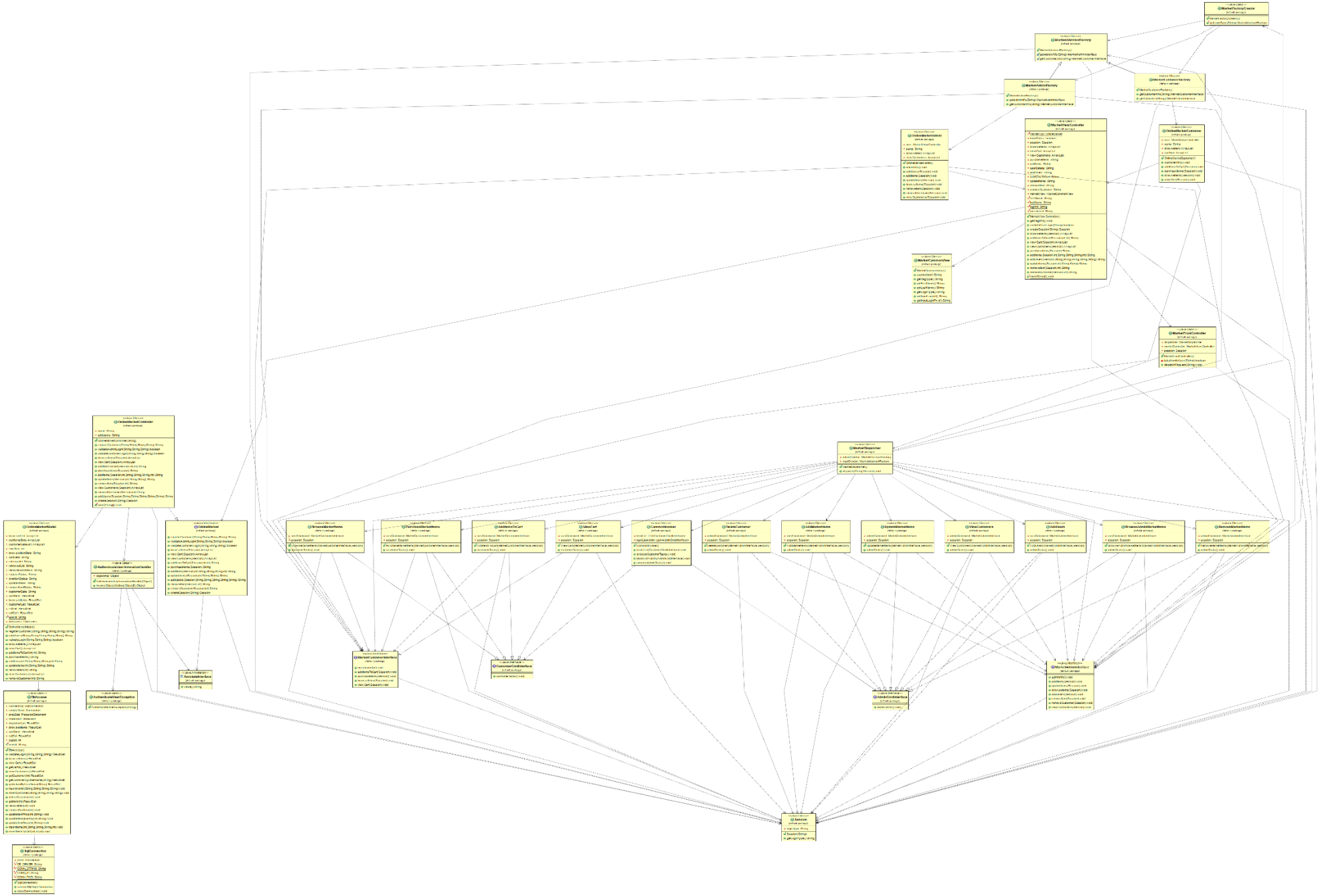

**Fig 2. Class Diagram** [please refer git for a better view]

## Sample Run

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

- Running server and registry

```
[yashkuru@in-csci-rrpc01 ~]$ cd OOAD/6Assignment/
[yashkuru@in-csci-rrpc01 6Assignment]$ rmiregistry 5432&
[1] 17102
[yashkuru@in-csci-rrpc01 6Assignment]$ sh makefileS.sh
You are now entering Online Market Place
Reaching server://10.234.136.55:5432/OnlineMarketServer
Interface is Ready!You can register, login and shop
```

- New Customer Registration and Login

yashkuru@in-csci-rrpc05:~/OOAD/6Assignment

```
Last login: Tue Apr  3 23:12:13 2018 from 140-182-64-30.ssl-vpn.iupui.edu
[yashkuru@in-csci-rrpc05 ~]$ cd OOAD/6Assignment/
[yashkuru@in-csci-rrpc05 6Assignment]$ sh makefileC.sh
~~~~~~~~~~~~~~~Login/Registration~~~~~~~~~~~~~~~
Enter 'Login' without quotes for Customer/Admin Login
Enter 'Register' without quotes for registration a new Customer account
--------------Enter one from above-------------------
register
------>> New Customer Registration <<----
Enter First Name:
yashwanth
Enter Last Name:
kuruganti
Enter UserId:
yashwanth
Enter Password:
yashwanth
----- You are now successfully Registered ----
~~~~~~~~~~~~~~~Login/Registration~~~~~~~~~~~~~~~
Enter 'Login' without quotes for Customer/Admin Login
Enter 'Register' without quotes for registration a new Customer account
--------------Enter one from above-------------------
login
~~~~~~~~~~~~~~~~~~~~~Login~~~~~~~~~~~~~~~~~~~~~~~
Enter 'Admin' for Administrator login without quotes
Enter 'Customer' for Customer login without quotes
------------Enter one from above-------------------
customer
Enter UserId:
yashwanth
Enter Password:
yashwanth
Login Status:true
You are now accessing market application as: customer
-------------------------------------
-Welcome to the Customer Home Page-
-------------------------------------
Hi Customer! You have the following commands to perform
---Enter 'Browse' ignoring quotes to shop
---Enter 'Add' ignoring quotes to add items to the cart
---Enter 'View' ignoring quotes to view items in cart
---Enter 'Purchase' ignoring quotes to checkout items from the cart
>>To exit these commands, press ctrl+c<<
```

- Admin Login and his/her functionalities

yashkuru@tesla:~/OOAD/6Assignment

```
Using username "yashkuru".
yashkuru@tesla.cs.iupui.edu's password:
Last login: Fri Apr 20 23:54:18 2018 from 140-182-64-199.ssl-vpn.iupui.edu
[yashkuru@tesla ~]$ cd OOAD/6Assignment/
[yashkuru@tesla 6Assignment]$ sh makefileC.sh
~~~~~~~~~~~~~~Login/Registration~~~~~~~~~~~~~~~~
Enter 'Login' without quotes for Customer/Admin Login
Enter 'Register' without quotes for registration a new Customer account
--------------Enter one from above------------------
login
~~~~~~~~~~~~~~~~~~~~Login~~~~~~~~~~~~~~~~~~~~~~~~
Enter 'Admin' for Administrator login without quotes
Enter 'Customer' for Customer login without quotes
------------Enter one from above------------------
admin
Enter UserId:
admin
Enter Password:
admin
Login Status:true
You are now accessing market application as: admin
-----------------------------------
---Welcome to the Admin Home Page---
-----------------------------------
You can now Browse, Add, Delete, Update
Hi Admin! You have the following commands to perform
---Enter 'browse' ignoring quotes to browse items
---Enter 'addI' ignoring quotes to add items
---Enter 'addU' ignoring quotes to add Customer/Admin
---Enter 'update' ignoring quotes to update items
---Enter 'removeC' ignoring quotes to delete Customer
---Enter 'removeI' ignoring quotes to delete Item
---Enter 'view' ignoring quotes to view all Customers
>>>...To exit these commands, press ctrl+c...<<<
```

- Admin Browse Items

```
browse
<---++---Your shopping items list here----+++--->
ItemId  ItemName  ItemType  ItemPrice  ItemQuantity
45696-----TypeMachine-----Electronics-----74.99$-----70
55555-----Computer-----Tech-----599.99$-----55
78784-----Cooker----------59.99$-----20
123456-----Papers-----Office-----6.99$-----25
252525-----Tshirt-----Clothing-----9.99$-----60
<------------------------------------------------>
---Enter 'browse' ignoring quotes to browse items
---Enter 'addI' ignoring quotes to add items
---Enter 'addU' ignoring quotes to add Customer/Admin
---Enter 'update' ignoring quotes to update items
---Enter 'removeC' ignoring quotes to delete Customer
---Enter 'removeI' ignoring quotes to delete Item
---Enter 'view' ignoring quotes to view all Customers
>>>...To exit these commands, press ctrl+c...<<<
```

- Admin - View Customer list

```
---Enter 'browse' ignoring quotes to browse items
---Enter 'addI' ignoring quotes to add items
---Enter 'addU' ignoring quotes to add Customer/Admin
---Enter 'update' ignoring quotes to update items
---Enter 'removeC' ignoring quotes to delete Customer
---Enter 'removeI' ignoring quotes to delete Item
---Enter 'view' ignoring quotes to view all Customers
>>>...To exit these commands, press ctrl+c...<<<
view
*----*****-------List of Customers-------*****----*
CustomerId  UserName
65---------maya
66---------paul
68---------ramu
67---------sidhu
61---------tank
55---------yash
63---------yash1
69---------yashwanth
<------------------------------------------------>
```

- Admin Add Customers/Admin

```
<------------------------------------------------>
---Enter 'browse' ignoring quotes to browse items
---Enter 'addI' ignoring quotes to add items
---Enter 'addU' ignoring quotes to add Customer/Admin
---Enter 'update' ignoring quotes to update items
---Enter 'removeC' ignoring quotes to delete Customer
---Enter 'removeI' ignoring quotes to delete Item
---Enter 'view' ignoring quotes to view all Customers
>>>...To exit these commands, press ctrl+c...<<<
addu
++++++++++Add Customer/Admin here++++++++++
Enter 'Admin' to add a new admin to database
Enter 'Customer' to add a new admin to database
-------------Enter one from above-----------------
admin
Enter First Name:
admin5
Enter Last Name:
admin5
Enter UserId:
admin5
Enter Password:
admin5
You have now successfully created a Admin account
```

- Admin- Update Items

```
update
++++------++++Update items here++++------++++
Enter Item Id to update: 55555
----Enter Item attribute to be updated----
Any of the attributes either: price or quantity or desc::
price
Enter Item attribute Value: 755.49$
++++++++++Above item has been updated to database++++++++++

---Enter 'browse' ignoring quotes to browse items
---Enter 'addI' ignoring quotes to add items
---Enter 'addU' ignoring quotes to add Customer/Admin
---Enter 'update' ignoring quotes to update items
---Enter 'removeC' ignoring quotes to delete Customer
---Enter 'removeI' ignoring quotes to delete Item
---Enter 'view' ignoring quotes to view all Customers
>>>...To exit these commands, press ctrl+c...<<<
browse
<---+++---Your shopping items list here----+++--->
ItemId  ItemName  ItemType  ItemPrice  ItemQuantity
45696-----TypeMachine-----Electronics-----74.99$-----65
55555-----Computer-----Tech-----755.49$-----50
78784-----Cooker----------59.99$-----20
123456-----Papers-----Office-----6.99$-----25
252525-----Tshirt-----Clothing-----9.99$-----60
<------------------------------------------------>
```

- Admin – Delete items

```
removei
*--------------Remove Items here--------------*
Enter Item Id to delete the item: 78784
Success-->Deleted requested Item.
---Enter 'browse' ignoring quotes to browse items
---Enter 'addI' ignoring quotes to add items
---Enter 'addU' ignoring quotes to add Customer/Admin
---Enter 'update' ignoring quotes to update items
---Enter 'removeC' ignoring quotes to delete Customer
---Enter 'removeI' ignoring quotes to delete Item
---Enter 'view' ignoring quotes to view all Customers
>>>...To exit these commands, press ctrl+c...<<<
browse
<---+++---Your shopping items list here----+++--->
ItemId  ItemName  ItemType  ItemPrice  ItemQuantity
45696-----TypeMachine-----Electronics-----74.99$-----65
55555-----Computer-----Tech-----755.49$-----50
123456-----Papers-----Office-----6.99$-----25
252525-----Tshirt-----Clothing-----9.99$-----60
<------------------------------------------------>
```

- Admin – Add items

```
addi
++++++++++Add items here++++++++++
Enter Item Id: 741258
Enter Item Name without space: Perfume
Enter Item Type without space: Accessories
Enter Item Price: 19.99$
Enter Item Quantity: 45
++++++++++Above item has been added to database++++++++++

---Enter 'browse' ignoring quotes to browse items
---Enter 'addI' ignoring quotes to add items
---Enter 'addU' ignoring quotes to add Customer/Admin
---Enter 'update' ignoring quotes to update items
---Enter 'removeC' ignoring quotes to delete Customer
---Enter 'removeI' ignoring quotes to delete Item
---Enter 'view' ignoring quotes to view all Customers
>>>...To exit these commands, press ctrl+c...<<<
browse
<---+++---Your shopping items list here----+++--->
ItemId  ItemName  ItemType  ItemPrice  ItemQuantity
45696-----TypeMachine-----Electronics-----74.99$-----65
55555-----Computer-----Tech-----755.49$-----50
123456-----Papers-----Office-----6.99$-----25
252525-----Tshirt-----Clothing-----9.99$-----60
741258-----Perfume-----Accessories-----19.99$-----45
<------------------------------------------------>
```

- Customer View Cart

```
add
++++++++++Add items to cart here++++++++++
Enter Item Id of the above Items you wish: 45696
Enter Item Quantity to be purchased: 5
Your item TypeMachine has been added to cart successfully
---Enter 'Browse' ignoring quotes to shop
---Enter 'Add' ignoring quotes to add items to the cart
---Enter 'View' ignoring quotes to view items in cart
---Enter 'Purchase' ignoring quotes to checkout items from the cart
>>To exit these commands, press ctrl+c<<
view
<---+++---Your Cart items list here----+++--->
CartId  ItemId  ItemQuantity
100-----45696-----5
100-----55555-----5
```

- Customer Browse Items and Add to Cart

```
>>To exit these commands, press ctrl+c<<
browse
<---+++---Your shopping items list here----+++--->
ItemId  ItemName  ItemType  ItemPrice  ItemQuantity
45696-----TypeMachine-----Electronics-----74.99$-----70
55555-----Computer-----Tech-----599.99$-----55
78784-----Cooker----------59.99$-----20
123456-----Papers-----Office-----6.99$-----25
252525-----Tshirt-----Clothing-----9.99$-----60
<------------------------------------------------>
---Enter 'Browse' ignoring quotes to shop
---Enter 'Add' ignoring quotes to add items to the cart
---Enter 'View' ignoring quotes to view items in cart
---Enter 'Purchase' ignoring quotes to checkout items from the cart
>>To exit these commands, press ctrl+c<<
add
++++++++++Add items to cart here++++++++++
Enter Item Id of the above Items you wish: 55555
Enter Item Quantity to be purchased: 5
Your item Computer has been added to cart successfully
---Enter 'Browse' ignoring quotes to shop
---Enter 'Add' ignoring quotes to add items to the cart
---Enter 'View' ignoring quotes to view items in cart
---Enter 'Purchase' ignoring quotes to checkout items from the cart
>>To exit these commands, press ctrl+c<<
```

- Customer Purchase/Checkout items

```
purchase
++++++++++Check out items here++++++++++
 <<<<< 45696 Purchase Successful >>>>
 <<<<< 55555 Purchase Successful >>>>

---Enter 'Browse' ignoring quotes to shop
---Enter 'Add' ignoring quotes to add items to the cart
---Enter 'View' ignoring quotes to view items in cart
---Enter 'Purchase' ignoring quotes to checkout items from the cart
>>To exit these commands, press ctrl+c<<
browse
<---+++---Your shopping items list here----+++--->
ItemId  ItemName  ItemType  ItemPrice  ItemQuantity
45696-----TypeMachine-----Electronics-----74.99$-----65
55555-----Computer-----Tech-----599.99$-----50
78784-----Cooker----------59.99$-----20
123456-----Papers-----Office-----6.99$-----25
252525-----Tshirt-----Clothing-----9.99$-----60
<------------------------------------------------>
---Enter 'Browse' ignoring quotes to shop
---Enter 'Add' ignoring quotes to add items to the cart
---Enter 'View' ignoring quotes to view items in cart
---Enter 'Purchase' ignoring quotes to checkout items from the cart
>>To exit these commands, press ctrl+c<<
```

## Conclusion

Before registering for this course, I am little aware of what Object-oriented design is, but unaware of how these patterns really help an industry to build a distributed software with improved quality. At the end of this course, I really learnt various design patterns, their implementations and their consequences effect the online market application. Also tried to avoid anti-design patterns which may negatively affect the purpose to be achieved. Along with these design patterns, this is my first Java-based network application. The way one single huge market application project divided into six assignments helped me to learn Java in parallel to its implementation of design patterns. Assignment feedback provided by professor supplemented my thinking of implementing a correct design pattern ensuring low coupling and high cohesion among various components of the application. Learnt the importance of coding standards as well as software documentation. For few design patterns, it took me time to understand their implementations and linkage to other design patterns. In the start, I didn't like one thing about the course i.e., Java for implementing the design patterns because I am new to Java and like to code in Python. This prevented me from building a GUI to make the application more interactive and fully concentrated on understanding patterns and its implementation in Java. If I would have a chance to go back and change something, that would be making use of graphical interface instead of command line. Doing this would have helped me build a both fully functional and fully interactive, also user friendly.

One thing I have learnt from the past week presentations is that, the first built skeleton and design decisions made will affect the overall application either in a good or not so good way. Java RMI helps for providing communication among the client and server, but I would love to change this framework and make use of other messaging services and observe how those patterns would help the application to overcome the disadvantages of RMI. In this last assignment, I have made the market application fully functional and ensured to maintain low coupling and high cohesion along with the implementation of database access layer pattern.

## References

[1] In-Class slides and examples