

Market Place

Assignment Report 6

Virinchi Sainath Nalluri

vnalluri@iupui.edu

27th April, 2018



Table of Contents

1. Assignment Overviews
 - 1.1. Assignment #1 Overview
 - 1.2. Assignment #2 Overview
 - 1.3. Assignment #3 Overview
 - 1.4. Assignment #4 Overview
 - 1.5. Assignment #5 Overview
2. Assignment 6 Discussion
 - 2.1. Database Access Layer Pattern
3. Final Documentation
 - 3.1. UML Diagrams
 - 3.2. Sample Runs
4. Final Conclusions
5. References



Assignment Overviews

Assignment #1 Overview

In Assignment #1 we started of the application by first creating the Domain Model. In the model, representation of all the objects we will be using in the future is made. Also as a part of this assignment, the basis for the application that is using Java RMI using the machines available to us is coded. For the Java RMI application, a server side and client side are integrated. Also as a part of this assignment, we used MVC pattern to clearly separate the server side and client side and setting a communication between these two components through a controller.

Assignment #2 Overview

In Assignment #2, over the MVC pattern we implemented the Front Controller, Command Pattern and Abstract Factory. Front Controller acts at the single point of contact for all the communications between model and view. In this pattern, we also implemented functionalities such as login for custom and administrator using hardcoded user credentials. For implementing this functionality we used command pattern which takes commands such as validateCustomerLogin etc,. The Abstract factory is used to generate view objects according to the requirement.

Assignment #3 Overview

In Assignment #3, we implemented **Authorization pattern** and **Reflection Pattern**. Using Authorization pattern, we made sure the only the right actors are accessing their functions i.e., only an admin should be able to add a product to the inventory and only a customer has to be able to add an item to cart. This is achieved using the annotation RequiresRole over method signatures. The reflection pattern, is providing a mirror image of the internal layers. The pattern dynamically decides what logic to implement. In this case, it decides depending upon the call, where to allow it or to block it with an invalid authorization exception.

Assignment #4 Overview

In Assignment #4, we understood the internals of Java RMI, more specifically its concurrency and also the *guarantee* that the Java RMI gives in concurrency. In this assignment we also implemented the Purchase Item, Add Item and Browse Item functionalities.

Assignment #5 Overview

In Assignment #5, we implemented synchronization patterns such as **Monitor Object**, **Future Object**, **Guarded Suspension**, **Scoped locking** and **thread safe interface**. Also along the process we understood synchronization in java. Also in this assignment, every functionality for both roles have been implemented meeting all project requirements and guidelines. From this assignment, the project is fully functional.

Assignment #6 — Discussion

Database Access Layer Pattern

According to the database access layer pattern, the project should have a business layer, a database and a layer which does the database access for the business layer. This layer is called the database access layer. The business layer has the logic of the project. In the project, Service package has all the business layers. The Service package has classes such as CustomerService, ItemService, AdminService. The service classes has business logic for their respective entities. The Dao layer in the projects is daoservice package. It has classes BaseDaoService, CustomerDaoService, ItemDaoService and AdminDaoService. The dao layer uses the JDBC connection libraries to connect to database and do CRUD operations on the database. The BaseDaoService declares all the required variables for prepared statement and database connector object and gets all the object required by every dao service class. All the other dao service classes extends the BaseDaoService, there by gaining access to the database. Every database operation has to pass through and will be done by this dao layer.

Database Schema

Customer Table : customer_id, firstname, lastname, username, password

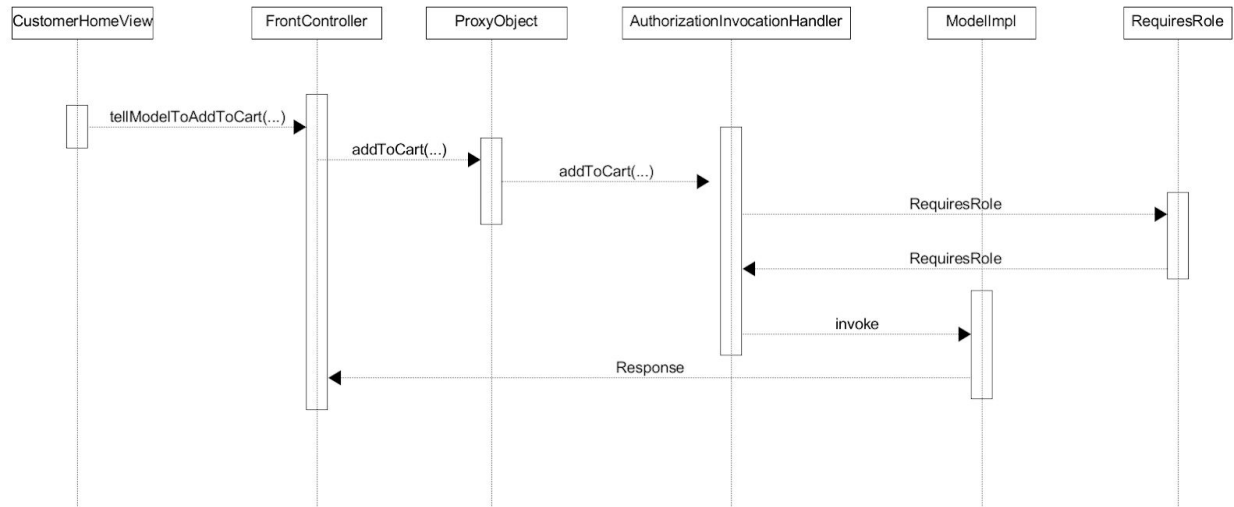
Admin Table: admin_id, firstname, lastname, username, password

Item Table: item_id, type, description, price, quantity

Cart Table: id, customer_id, item_id

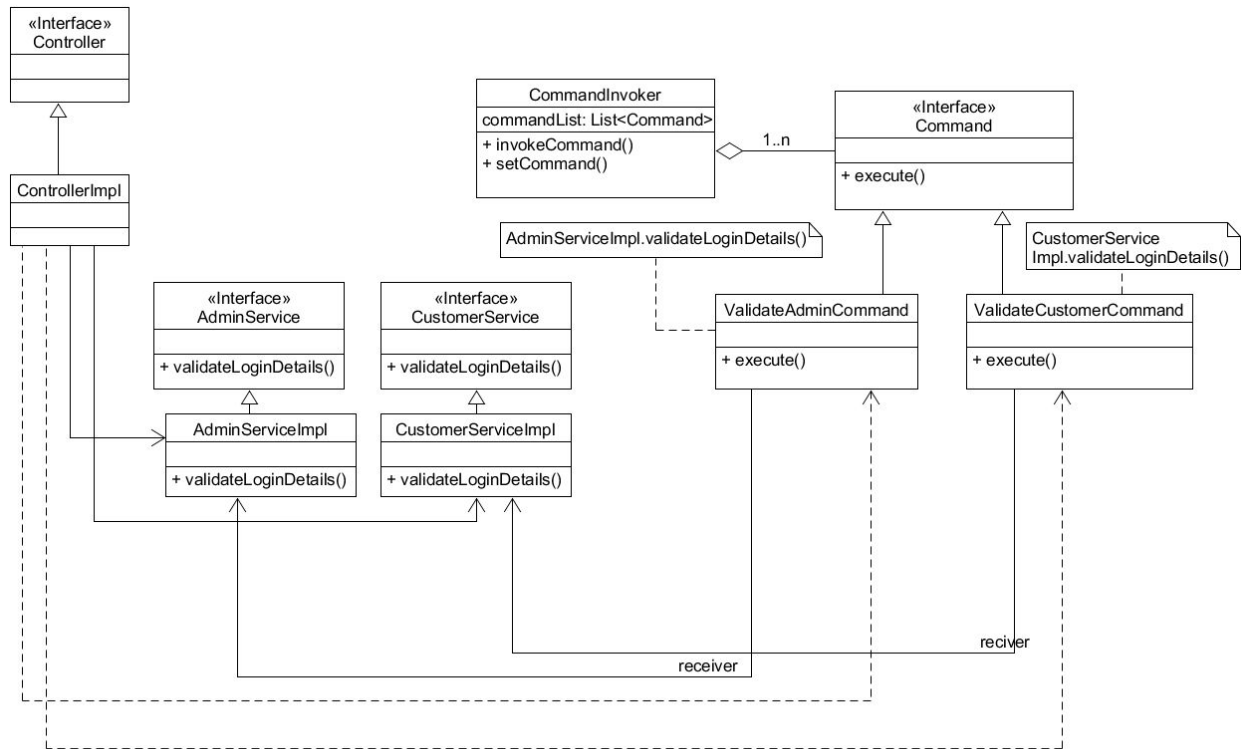
Here, the cart table has only customer_id and item_id, as everytime, the customer logs in, the cart of the customer is computed from this table. For computing the cart, find the number of customer_id, item_id pairs and add the item_id pairs to the cart. Everytime there is a change in the cart, the changes are persisted in this table.

UML Sequence Diagram

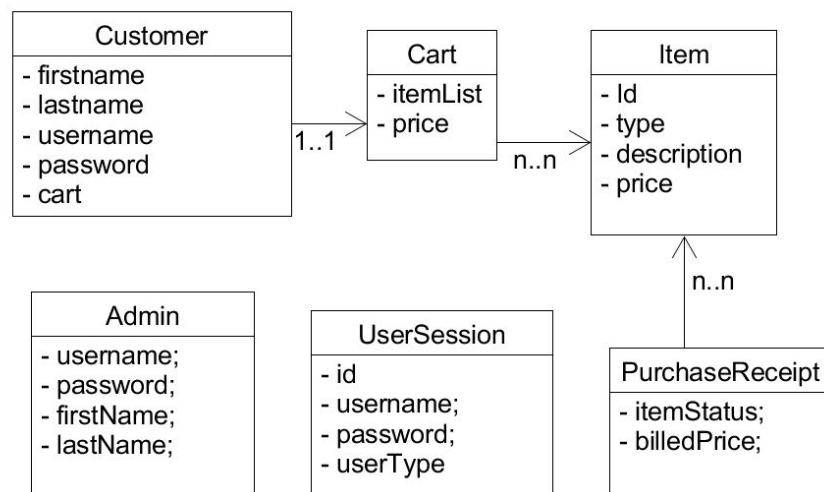


Other diagrams such as class diagram and domain model diagram are still valid.

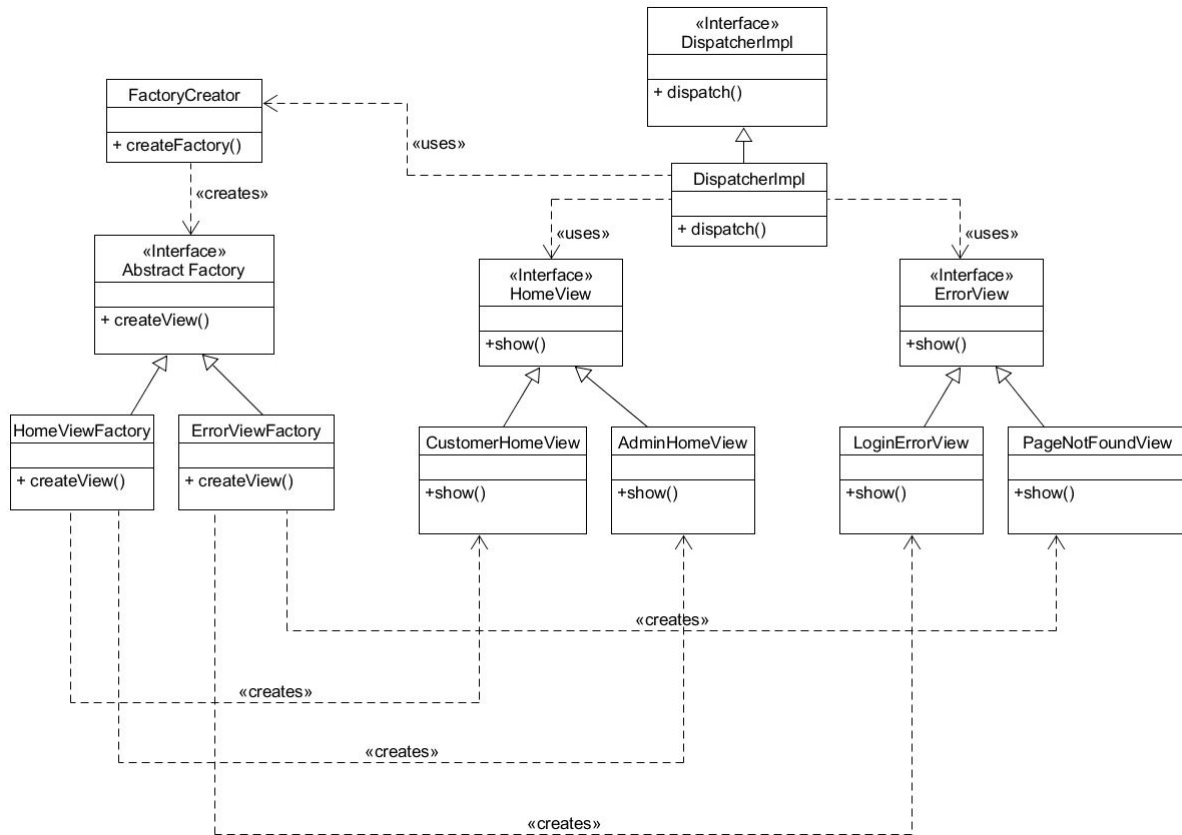
UML Class Diagram for Command Pattern



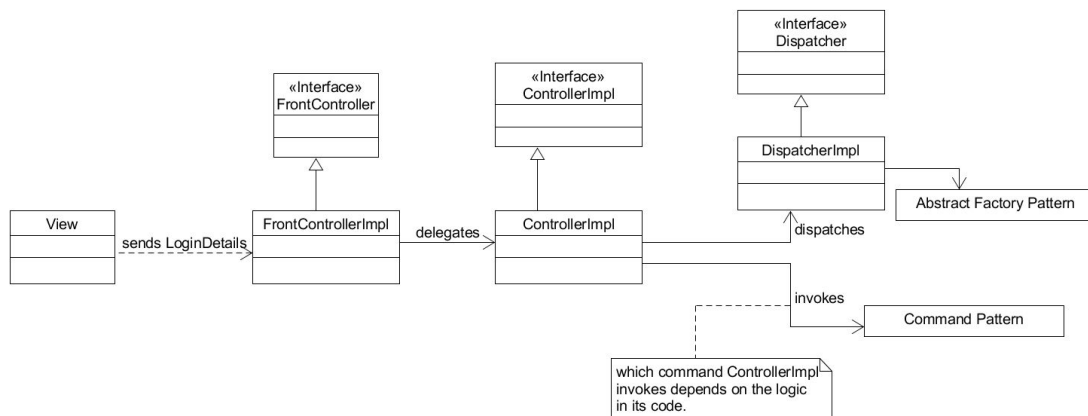
UML Domain Model



UML Diagram for Abstract Factory

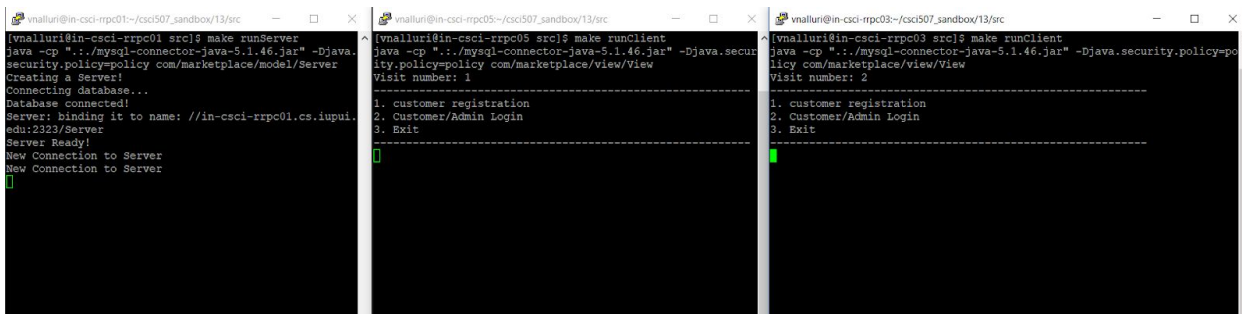


UML Class Diagram for Front Controller



Sample Runs

Initially, running server and two clients



The image shows three terminal windows side-by-side, each running a different Java application. The first window on the left is titled 'vnalluri@in-csci-rpc01:~/csci507_sandbox/13/src' and shows the execution of 'make runServer'. It outputs 'Creating a Server!', 'Connecting database...', 'Database connected!', 'Server: binding it to name: //in-csci-rpc01.cs.iupui.edu:2323/Server', 'Server Ready!', 'New Connection to Server', and 'New Connection to Server'. The middle window is titled 'vnalluri@in-csci-rpc05:~/csci507_sandbox/13/src' and shows the execution of 'make runClient'. It outputs 'Visit number: 1' followed by a menu: '1. customer registration', '2. Customer/Admin Login', and '3. Exit'. The third window on the right is titled 'vnalluri@in-csci-rpc03:~/csci507_sandbox/13/src' and shows the execution of 'make runClient'. It outputs 'Visit number: 2' followed by the same menu as the middle window.

```
[vnalluri@in-csci-rpc01 src]$ make runServer
java -cp ".:/mysql-connector-java-5.1.46.jar" -Djava.security.policy=policy.com/marketplace/model/Server
Creating a Server!
Connecting database...
Database connected!
Server: binding it to name: //in-csci-rpc01.cs.iupui.edu:2323/Server
Server Ready!
New Connection to Server
New Connection to Server

[vnalluri@in-csci-rpc05 src]$ make runClient
java -cp ".:/mysql-connector-java-5.1.46.jar" -Djava.security.policy=policy.com/marketplace/view/View
Visit number: 1
-----
1. customer registration
2. Customer/Admin Login
3. Exit
-----

[vnalluri@in-csci-rpc03 src]$ make runClient
java -cp ".:/mysql-connector-java-5.1.46.jar" -Djava.security.policy=policy.com/marketplace/view/View
Visit number: 2
-----
1. customer registration
2. Customer/Admin Login
3. Exit
-----
```

Customer Registration and Login

First I registered as vnalluri and logged in as vnalluri and got success message.


```
vnulluri@in-csci-rrpc05:~/csci507_sandbox/13/src
[vnulluri@in-csci-rrpc05 src]$ make runClient
java -cp "../mysql-connector-java-5.1.46.jar" -Djava.security.policy=policy com/marketplace/view/View
Visit number: 1
-----
1. customer registration
2. Customer/Admin Login
3. Exit
-----
1
Enter customer firstname :
vnulluri
Enter customer lastname :
vnulluri
Enter customer username :
vnulluri
Enter customer password :
vnulluri
-----
1. customer registration
2. Customer/Admin Login
3. Exit
-----
2
Your have connected to the server
Welcome to MarketPlace
Enter login credentials
Enter your username : vnulluri
Enter your password : vnulluri
Enter your userType (customer or admin) : customer
Hi Customer, You have logged in
-----MENU-----
1. Browse 2. Add Item To Cart 3. View Cart 4. Clear Items in Cart
5. Purchase Items in Cart 6. Exit
-----
█
```

Customer Browse

```

-----MENU-----
1. Browse  2. Add Item To Cart  3. View Cart  4. Clear Item
s in Cart
5. Purchase Items in Cart  6. Exit
-----
1
Following are the products :
Item [id=1, type=Movies, description=Ready Player One DvD, price=35, quantity=43]
Item [id=2, type=Games, description=UNO Cards, price=10, quantity=114]
Item [id=3, type=Electronics, description=XBox, price=450, quantity=89]
Item [id=4, type=Electronics, description=iphone, price=950, quantity=59]
Item [id=5, type=Shoes, description=Nike Jordan, price=300, quantity=0]
Item [id=6, type=Clothes, description=Levi's Denim Jacket, price=35, quantity=60]
-----MENU-----
1. Browse  2. Add Item To Cart  3. View Cart  4. Clear Items in Cart
5. Purchase Items in Cart  6. Exit
-----

```

Customer Add Item To Cart

On selecting add item to cart (2) , a list is printed to pick id. On picking 6, it is added to cart.

```

-----MENU-----
1. Browse  2. Add Item To Cart  3. View Cart  4. Clear Items in Cart
5. Purchase Items in Cart  6. Exit
-----
2
Following are the products :
Item [id=1, type=Movies, description=Ready Player One DvD, price=35, quantity=43]
Item [id=2, type=Games, description=UNO Cards, price=10, quantity=114]
Item [id=3, type=Electronics, description=XBox, price=450, quantity=89]
Item [id=4, type=Electronics, description=iphone, price=950, quantity=59]
Item [id=5, type=Shoes, description=Nike Jordan, price=300, quantity=0]
Item [id=6, type=Clothes, description=Levi's Denim Jacket, price=35, quantity=60]
Enter the item id you want to add to cart.
6
adding Levi's Denim Jacket to cart
Successfully added to Cart
-----MENU-----
1. Browse  2. Add Item To Cart  3. View Cart  4. Clear Items in Cart
5. Purchase Items in Cart  6. Exit
-----

```

Customer View Cart (after adding the above item - 6 to cart)

```

-----MENU-----
1. Browse  2. Add Item To Cart  3. View Cart  4. Clear Items in Cart
5. Purchase Items in Cart  6. Exit
-----
3
Item [id=6, type=Clothes, description=Levi's Denim Jacket, price=35, quantity=60]
Total Price of Cart = 35
-----MENU-----
1. Browse  2. Add Item To Cart  3. View Cart  4. Clear Items in Cart
5. Purchase Items in Cart  6. Exit
-----
3
Item [id=6, type=Clothes, description=Levi's Denim Jacket, price=35, quantity=60]
Total Price of Cart = 35
-----MENU-----
1. Browse  2. Add Item To Cart  3. View Cart  4. Clear Items in Cart
5. Purchase Items in Cart  6. Exit
-----

```

Clear Cart

First on view cart (3), shows two items, after clearing cart (4), no items in cart.

```

-----MENU-----
1. Browse  2. Add Item To Cart  3. View Cart  4. Clear Items in Cart
5. Purchase Items in Cart  6. Exit
-----
3
Item [id=6, type=Clothes, description=Levi's Denim Jacket, price=35, quantity=60]
Item [id=3, type=Electronics, description=XBox, price=450, quantity=89]
Total Price of Cart = 485
-----MENU-----
1. Browse  2. Add Item To Cart  3. View Cart  4. Clear Items in Cart
5. Purchase Items in Cart  6. Exit
-----
4
Cleared Cart
-----MENU-----
1. Browse  2. Add Item To Cart  3. View Cart  4. Clear Items in Cart
5. Purchase Items in Cart  6. Exit
-----
3
No items in cart
Total Price of Cart = 0
-----MENU-----
1. Browse  2. Add Item To Cart  3. View Cart  4. Clear Items in Cart
5. Purchase Items in Cart  6. Exit
-----

```

Purchase Items

Showing failed and successfully purchased items.

```
-----MENU-----
1. Browse  2. Add Item To Cart  3. View Cart  4. Clear Items in Cart
5. Purchase Items in Cart  6. Exit
-----
3
Item [id=3, type=Electronics, description=XBox, price=450, quantity=89]
Item [id=5, type=Shoes, description=Nike Jordan, price=300, quantity=1]
Item [id=5, type=Shoes, description=Nike Jordan, price=300, quantity=1]
Item [id=2, type=Games, description=UNO Cards, price=10, quantity=114]
Total Price of Cart = 1060
-----MENU-----
1. Browse  2. Add Item To Cart  3. View Cart  4. Clear Items in Cart
5. Purchase Items in Cart  6. Exit
-----
5
Your receipt:
Item: UNO Cards - Successful
Item: Nike Jordan - Failed
Item: Nike Jordan - Successful
Item: XBox - Successful
Price Billed = 760
-----MENU-----
1. Browse  2. Add Item To Cart  3. View Cart  4. Clear Items in Cart
5. Purchase Items in Cart  6. Exit
-----
```

Trying to add item whose quantity is 0

```
-----MENU-----
1. Browse  2. Add Item To Cart  3. View Cart  4. Clear Items in Cart
5. Purchase Items in Cart  6. Exit
-----
2
Following are the products :
Item [id=1, type=Movies, description=Ready Player One DvD, price=35, quantity=43]
Item [id=2, type=Games, description=UNO Cards, price=10, quantity=114]
Item [id=3, type=Electronics, description=XBox, price=450, quantity=89]
Item [id=4, type=Electronics, description=iphone, price=950, quantity=59]
Item [id=5, type=Shoes, description=Nike Jordan, price=300, quantity=0]
Item [id=6, type=Clothes, description=Levi's Denim Jacket, price=35, quantity=60]
Enter the item id you want to add to cart.
5
Add to Cart - Not Successful
-----
```


Admin Browse

```
-----MENU-----
1. Browse  2. Add another Admin  3. Add Item to Inventory
4. Update Item in Inventory  5. Remove Item in Inventory
6. Remove a customer account  7. Add a customer account
8. Exit
-----
1
Following are the products :
Item [id=1, type=Movies, description=Ready Player One DvD, price=35, quantity=43]
Item [id=2, type=Games, description=UNO Cards, price=10, quantity=114]
Item [id=3, type=Electronics, description=XBox, price=450, quantity=89]
Item [id=4, type=Electronics, description=iphone, price=950, quantity=59]
]
Item [id=5, type=Shoes, description=Nike Jordan, price=300, quantity=0]
Item [id=6, type=Clothes, description=Levi's Denim Jacket, price=35, quantity=60]
-----MENU-----
1. Browse  2. Add another Admin  3. Add Item to Inventory
4. Update Item in Inventory  5. Remove Item in Inventory
6. Remove a customer account  7. Add a customer account
8. Exit
-----
█
```

Admin Add Another Admin

```
-----MENU-----
1. Browse  2. Add another Admin  3. Add Item to Inventory
4. Update Item in Inventory  5. Remove Item in Inventory
6. Remove a customer account  7. Add a customer account
8. Exit
-----
2
Enter admin firstname :
Sam
Enter admin lastname :
Morgan
Enter admin username :
sam
Enter admin password :
sam
-----MENU-----
1. Browse  2. Add another Admin  3. Add Item to Inventory
4. Update Item in Inventory  5. Remove Item in Inventory
6. Remove a customer account  7. Add a customer account
8. Exit
-----
█
```

Add Item To Inventory

```
-----MENU-----
1. Browse  2. Add another Admin  3. Add Item to Inventory
4. Update Item in Inventory  5. Remove Item in Inventory
6. Remove a customer account  7. Add a customer account
8. Exit
-----
3
Enter product type :
Sports
Enter product description :
Basketball
Enter product price :
19
Enter product quantity :
50
-----MENU-----
1. Browse  2. Add another Admin  3. Add Item to Inventory
4. Update Item in Inventory  5. Remove Item in Inventory
6. Remove a customer account  7. Add a customer account
8. Exit
-----
1
Following are the products :
Item [id=1, type=Movies, description=Ready Player One DVD, price=35, quantity=43]
Item [id=2, type=Games, description=UNO Cards, price=10, quantity=113]
Item [id=3, type=Electronics, description=XBox, price=450, quantity=88]
Item [id=4, type=Electronics, description=iphone, price=950, quantity=59]
Item [id=5, type=Shoes, description=Nike Jordan, price=300, quantity=0]
Item [id=6, type=Clothes, description=Levi's Denim Jacket, price=35, quantity=60]
Item [id=7, type=Sports, description=Basketball, price=19, quantity=50]
-----MENU-----
1. Browse  2. Add another Admin  3. Add Item to Inventory
4. Update Item in Inventory  5. Remove Item in Inventory
6. Remove a customer account  7. Add a customer account
8. Exit
-----
```

Update item in inventory (updating the just added item)

```
-----MENU-----
1. Browse  2. Add another Admin  3. Add Item to Inventory
4. Update Item in Inventory  5. Remove Item in Inventory
6. Remove a customer account  7. Add a customer account
8. Exit
-----
4
Following are the products in the inventory:
Item [id=1, type=Movies, description=Ready Player One DvD, price=35, quantity=43]
Item [id=2, type=Games, description=UNO Cards, price=10, quantity=113]
Item [id=3, type=Electronics, description=XBox, price=450, quantity=88]
Item [id=4, type=Electronics, description=iphone, price=950, quantity=59]
Item [id=5, type=Shoes, description=Nike Jordan, price=300, quantity=0]
Item [id=6, type=Clothes, description=Levi's Denim Jacket, price=35, quantity=60]
Item [id=7, type=Sports, description=Basketball, price=19, quantity=50]
Enter the id of the item you want to update
7
Enter the updated item's type
Outdoor Sports
Enter the updated item's description
Basketball and Ring
Enter the updated item's price
40
Enter the updated item's quantity
35
Product successfully added to Inventory
-----MENU-----
1. Browse  2. Add another Admin  3. Add Item to Inventory
4. Update Item in Inventory  5. Remove Item in Inventory
6. Remove a customer account  7. Add a customer account
8. Exit
-----
1
Following are the products :
Item [id=1, type=Movies, description=Ready Player One DvD, price=35, quantity=43]
Item [id=2, type=Games, description=UNO Cards, price=10, quantity=113]
Item [id=3, type=Electronics, description=XBox, price=450, quantity=88]
Item [id=4, type=Electronics, description=iphone, price=950, quantity=59]
Item [id=5, type=Shoes, description=Nike Jordan, price=300, quantity=0]
Item [id=6, type=Clothes, description=Levi's Denim Jacket, price=35, quantity=60]
Item [id=7, type=Outdoor Sports, description=Basketball and Ring, price=40, quantity=35]
-----MENU-----
1. Browse  2. Add another Admin  3. Add Item to Inventory
4. Update Item in Inventory  5. Remove Item in Inventory
6. Remove a customer account  7. Add a customer account
8. Exit
-----
```


Remove Item in Inventory (removing the just updated item)

```
-----MENU-----
1. Browse  2. Add another Admin  3. Add Item to Inventory
4. Update Item in Inventory  5. Remove Item in Inventory
6. Remove a customer account  7. Add a customer account
8. Exit
-----
5
Following are the products in the inventory:
Item [id=1, type=Movies, description=Ready Player One DvD, price=35, quantity=43]
Item [id=2, type=Games, description=UNO Cards, price=10, quantity=113]
Item [id=3, type=Electronics, description=XBox, price=450, quantity=88]
Item [id=4, type=Electronics, description=iphone, price=950, quantity=59]
Item [id=5, type=Shoes, description=Nike Jordan, price=300, quantity=0]
Item [id=6, type=Clothes, description=Levi's Denim Jacket, price=35, quantity=60]
Item [id=7, type=Outdoor Sports, description=Basketball and Ring, price=40, quantity=35]
Enter the id of the item you want to remove
7
Successfully removed item
-----MENU-----
1. Browse  2. Add another Admin  3. Add Item to Inventory
4. Update Item in Inventory  5. Remove Item in Inventory
6. Remove a customer account  7. Add a customer account
8. Exit
-----
1
Following are the products :
Item [id=1, type=Movies, description=Ready Player One DvD, price=35, quantity=43]
Item [id=2, type=Games, description=UNO Cards, price=10, quantity=113]
Item [id=3, type=Electronics, description=XBox, price=450, quantity=88]
Item [id=4, type=Electronics, description=iphone, price=950, quantity=59]
Item [id=5, type=Shoes, description=Nike Jordan, price=300, quantity=0]
Item [id=6, type=Clothes, description=Levi's Denim Jacket, price=35, quantity=60]
-----MENU-----
1. Browse  2. Add another Admin  3. Add Item to Inventory
4. Update Item in Inventory  5. Remove Item in Inventory
6. Remove a customer account  7. Add a customer account
8. Exit
-----
```


Remove customer account

```
-----MENU-----
1. Browse  2. Add another Admin  3. Add Item to Inventory
4. Update Item in Inventory  5. Remove Item in Inventory
6. Remove a customer account  7. Add a customer account
8. Exit
-----
6
Enter the username of the customer to delete
bill
successfully removed customer
-----MENU-----
1. Browse  2. Add another Admin  3. Add Item to Inventory
4. Update Item in Inventory  5. Remove Item in Inventory
6. Remove a customer account  7. Add a customer account
8. Exit
-----
```

Add a customer account

```
-----MENU-----
1. Browse  2. Add another Admin  3. Add Item to Inventory
4. Update Item in Inventory  5. Remove Item in Inventory
6. Remove a customer account  7. Add a customer account
8. Exit
-----
7
Enter customer firstname :
Bill
Enter customer lastname :
Freeman
Enter customer username :
bill
Enter customer password :
bill
customer account created
-----MENU-----
1. Browse  2. Add another Admin  3. Add Item to Inventory
4. Update Item in Inventory  5. Remove Item in Inventory
6. Remove a customer account  7. Add a customer account
8. Exit
-----
```

Final Conclusions

What I like about the project?

In such a simple project, I didn't know those many design patterns can be implemented. Implementing design patterns in every step, greatly increase the scalability and workability of the code. Also as a part of the project, one gets to discover the multithreading concepts of Java. Not only that we understand how Java RMI works.

What did you dislike about the project?

The only thing one can dislike about the project is that if one wants to learn more about Java RMI, there is not a lot many online resources on this topic. However, the concepts taught in class reduces that affect of having less online resources.

What would you change about your design if you could go back? Why?

The only thing I would change is that in earlier Assignments we were using hard coded values to do implement few functionalities. Later in Assignment 5, we will have to use databases. I think if we used databases from the earlier assignments itself, a lot of refactoring of code for Assignment 5 would have been decreased. Nevertheless, the structure and order of assignments are well set to distribute work evenly among assignments.

References

1. For writing make file :
https://www.cs.swarthmore.edu/~newhall/unixhelp/howto_makefiles.html
2. Java RMI : Bank example in Canvas
3. MVC Pattern : Calculator Example in Canvas
4. Front Controller Pattern : Front Pattern Example in Canvas
5. Abstract Factory Pattern : Abstract Pattern Bank Example in slides.
6. Command Pattern: Command Pattern example in Canvas.
7. Reflection Pattern, Proxy Pattern : example from Canvas.
8. Authorization Pattern, RequiresRole annotation, AuthorizationInvocationHandler : from example in Canvas.
9. Thread-Safe, Scoped Locking —
<http://www.cs.wustl.edu/~schmidt/PDF/locking-patterns.pdf>

10. Future —

<https://docs.oracle.com/javase/8/docs/api/index.html?java/util/concurrent/Future.html>

11. Guarded Suspension— https://en.wikipedia.org/wiki/Guarded_suspension

12. Technologies Used : Eclipse IDE, WinScp, PUTTY, Tesla, Java 8

■ ■ ■