CSCI 50700 - Object-Oriented Design and Programming

Assignment 6 Report

Jingyi Hui

hui@iu.edu

Department of Computer and Information Science

IUPUI

Table of Contents

CSCI 50700 - Object-Oriented Design and Programming

Assignment 6 Report

1. Assignment Overviews

1) Assignment #1 Overviews

To build the Marketplace application, we are in the situation that we need a user interface (front-end, View) and a persistent storage (back-end, Model). It's desirable for us to separate the interface and system functionality since the UI changes more frequently than system functionality. As an architectural pattern, MVC pattern provides independency between UI and system functionality, so that there is no direct connection between UI and system data model, all the views are associated with one or more controllers that manipulate the model's state. With this kind of separation, if we need to make changes to the UI, the evolution can be achieved without impacting other views or the model itself.

To create a distributed application, Java RMI is introduced to allow communication between remote JVMs. It allows an object running in one machine to invoke methods on an object running in machine. With this infrastructure, our server can provide service to our remote customers.

2) Assignment #2 Overviews

2 different views based on different roles are implemented by Front Controller pattern together with the Abstract Factory pattern and Command Pattern. Front Controller provides centralized control for our application, it handles all the requests from users.
When user sees the LoginView, he/she will be required to input username and password. Based on the user's information, user will be dispatched to either customer's menu or admin's menu created by the Abstract Factory. This is kind of a way to provide security to our application. Since different users will see different menus, which can prevent them have access to unauthorized operations.

A Request class is created to wrapped a user request under an object as command and passed to invoker (Action) object. Invoker object looks for the appropriate object which can handle this command and passes the command to the corresponding object which executes the command.

3) Assignment #3 Overviews

We implemented an another way to provide access control – RABC, which is achieved by making use of Java Annotation. Instead of hiding invalid commands and showing different views based on roles, in this assignment, both of these 2 roles will see the same menu after login, and different roles have different authorized access to some commands.

In assignment 2, my Customer/Admin view didn't work, since I didn't retrieve user's role after I validate their information. To fix the problem, a Session class is added for passing parameters (username, password, and role) for future use.

4) Assignment #4 Overviews

In Assignment 4, 5 functions are implemented, and we are using 6 different machines to simulate a real distributed system. But I didn't realize that there are some bugs (I store my inventory as an array list, and set product ID as a property of the product, but I muddled it with the array index.) in my implementation until I got my feedback.

We also explore the concurrency provided by Java RMI. I did some test runs to verify that RMI provides us concurrency. From some research I did, I got to know more about threading in RMI.

5) Assignment #5 Overviews

Application functions are fully implemented in this assignment. Several bugs from Assignment 4 are fixed in this one.

MySQL database is used as the persistent storage in this assignment. We are no longer using *.txt* or *.property* files to store inventory or user information, which makes this project more like a real distributed application. *MySQL* class is added to establish the connection between my application and the database server, implement actions with MySQL queries, and works as a layer to shield my business logic from queries.

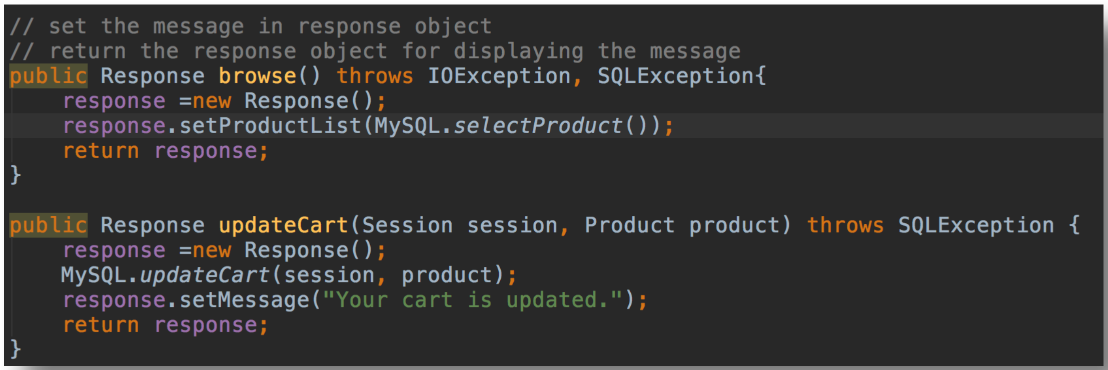
To achieve synchronization in our application, key word: synchronized is used for several methods like: Update Inventory, Purchase, Remove Inventory…to achieve thread safety. When several users are using our application, the system will apply some lock mechanism to ensure at a given time, only one thread can have the right to access that resource.

2. Assignment #6 Discussion

Database Access Layer Pattern

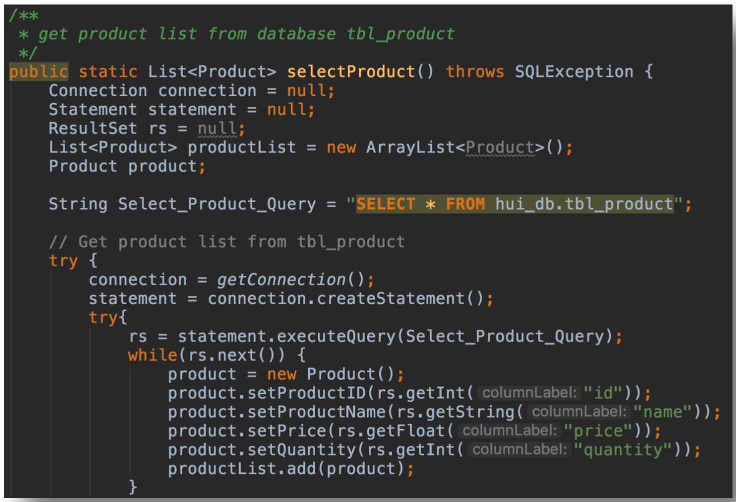
I implemented the database access layer by creating a class (*MySQL.java*) of data access methods that directly reference a corresponding set of database stored procedures. This class works as an interface between my application logic and the database (hui_db).

My application can store and retrieve data by calling an appropriate method (Figure 1) in my access layer, which decouples the core business logic and the database queries (Figure 2). If there is any change in database schema, I only need to modify this access layer.



```java
// set the message in response object
// return the response object for displaying the message
public Response browse() throws IOException, SQLException{
    response =new Response();
    response.setProductList(MySQL.selectProduct());
    return response;
}

public Response updateCart(Session session, Product product) throws SQLException {
    response =new Response();
    MySQL.updateCart(session, product);
    response.setMessage("Your cart is updated.");
    return response;
}
```

Figure 1 How the method is used in ActionServerModel class



```java
/**
 * get product list from database tbl_product
 */
public static List<Product> selectProduct() throws SQLException {
    Connection connection = null;
    Statement statement = null;
    ResultSet rs = null;
    List<Product> productList = new ArrayList<Product>();
    Product product;

    String Select_Product_Query = "SELECT * FROM hui_db.tbl_product";

    // Get product list from tbl_product
    try {
        connection = getConnection();
        statement = connection.createStatement();
        try{
            rs = statement.executeQuery(Select_Product_Query);
            while(rs.next()) {
                product = new Product();
                product.setProductID(rs.getInt( columnLabel: "id"));
                product.setProductName(rs.getString( columnLabel: "name"));
                product.setPrice(rs.getFloat( columnLabel: "price"));
                product.setQuantity(rs.getInt( columnLabel: "quantity"));
                productList.add(product);
            }
```

Figure 2 SQL used to retrieve product inventory in MySQL class

3. UML Diagrams

1) Domain Model

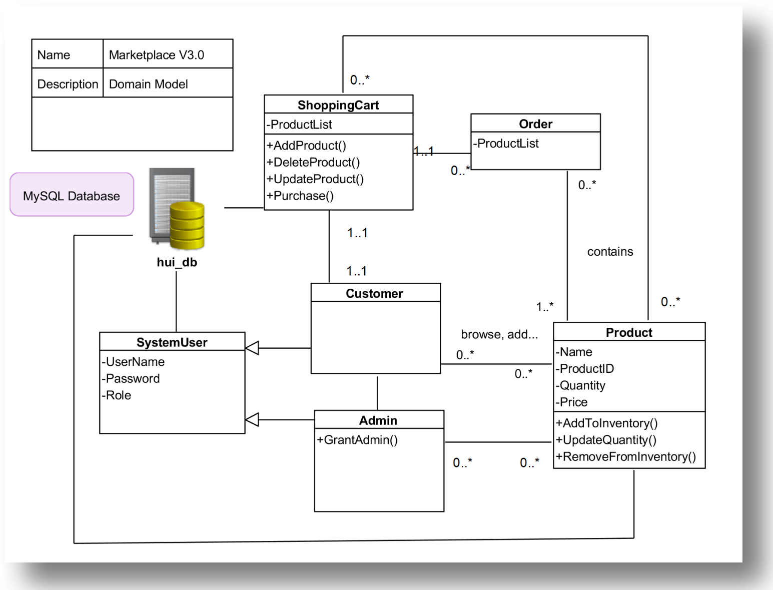


Figure 3 Marketplace Domain Model

2) Software design diagram

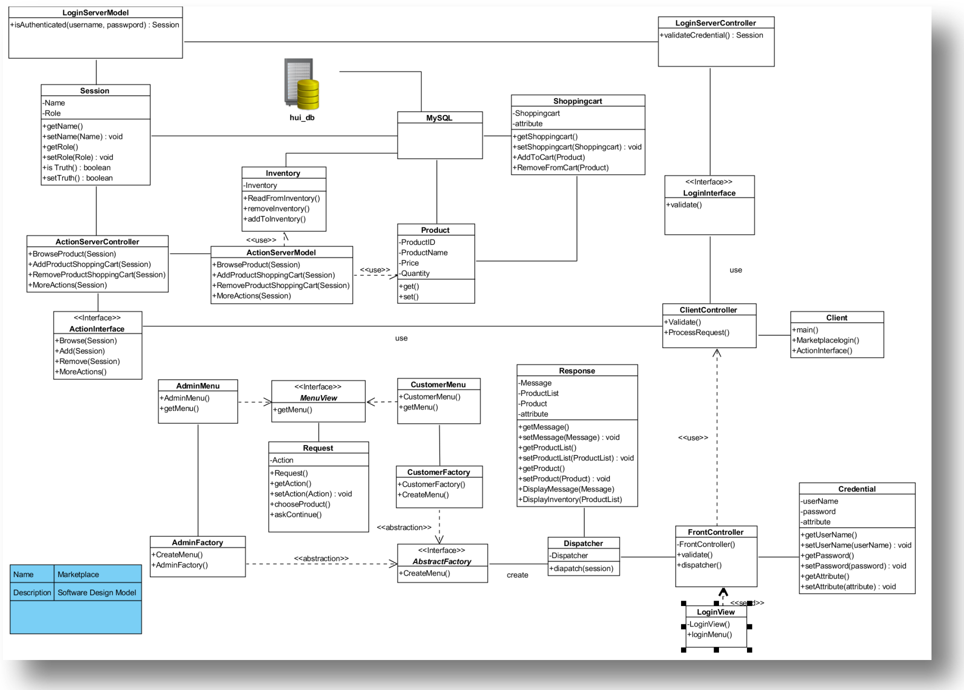
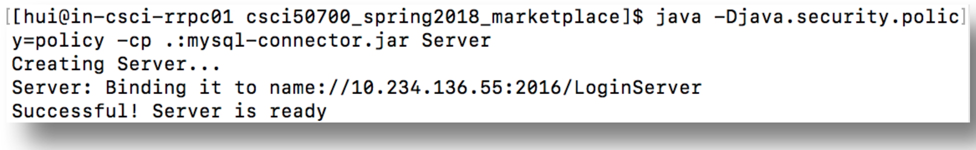


Figure 4 Marketplace Software Design Diagram

4. Sample Runs

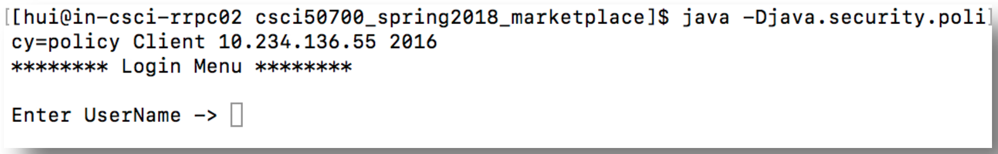
1) Setup server on rrpc01



```
[hui@in-csci-rrpc01 csci50700_spring2018_marketplace]$ java -Djava.security.polic
y=policy -cp .:mysql-connector.jar Server
Creating Server...
Server: Binding it to name://10.234.136.55:2016/LoginServer
Successful! Server is ready
```

Figure 5 Setup Server

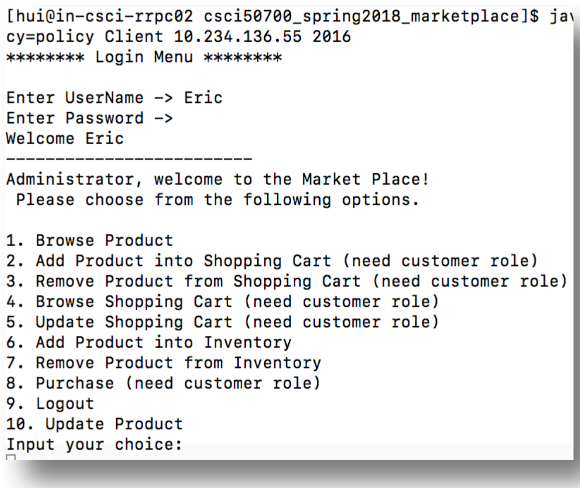
2) Setup client on rrpc02



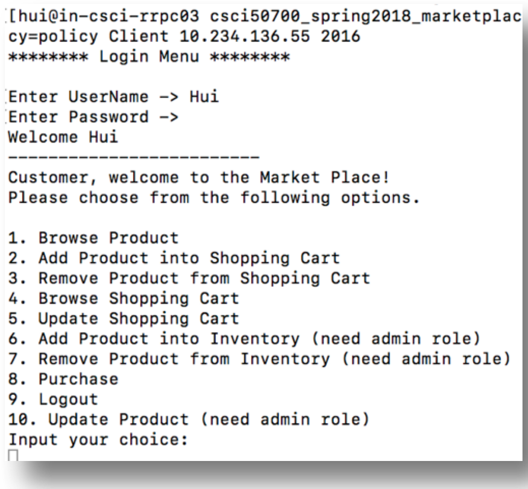
```
[hui@in-csci-rrpc02 csci50700_spring2018_marketplace]$ java -Djava.security.poli
cy=policy Client 10.234.136.55 2016
******** Login Menu ********

Enter UserName -> 
```

Figure 6 Setup Client

3) **Login function** for *admin* and *customer* roles



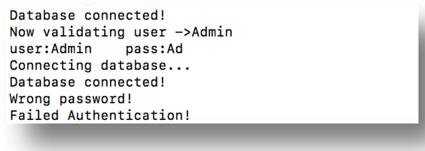
```
[hui@in-csci-rrpc02 csci50700_spring2018_marketplace]$ jav
cy=policy Client 10.234.136.55 2016
******** Login Menu ********

Enter UserName -> Eric
Enter Password ->
Welcome Eric
-------------------------
Administrator, welcome to the Market Place!
 Please choose from the following options.

1. Browse Product
2. Add Product into Shopping Cart (need customer role)
3. Remove Product from Shopping Cart (need customer role)
4. Browse Shopping Cart (need customer role)
5. Update Shopping Cart (need customer role)
6. Add Product into Inventory
7. Remove Product from Inventory
8. Purchase (need customer role)
9. Logout
10. Update Product
Input your choice:
```



```
[hui@in-csci-rrpc03 csci50700_spring2018_marketplac
cy=policy Client 10.234.136.55 2016
******** Login Menu ********

Enter UserName -> Hui
Enter Password ->
Welcome Hui
-------------------------
Customer, welcome to the Market Place!
Please choose from the following options.

1. Browse Product
2. Add Product into Shopping Cart
3. Remove Product from Shopping Cart
4. Browse Shopping Cart
5. Update Shopping Cart
6. Add Product into Inventory (need admin role)
7. Remove Product from Inventory (need admin role)
8. Purchase
9. Logout
10. Update Product (need admin role)
Input your choice:
```

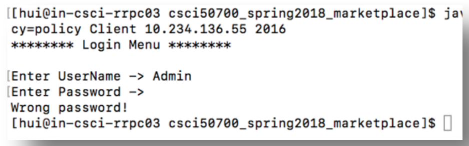
Figure 7 Login-admin view

Figure 8 Login-customer view



```
Database connected!
Now validating user ->Admin
user:Admin    pass:Ad
Connecting database...
Database connected!
Wrong password!
Failed Authentication!
```

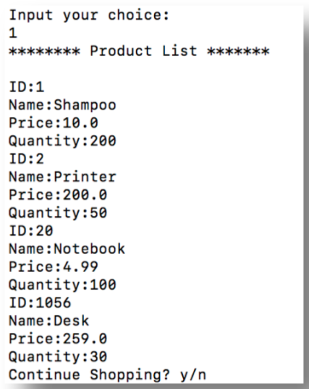
Figure 9 Login with Incorrect Password (Server Side)



```
[[hui@in-csci-rrpc03 csci50700_spring2018_marketplace]$ jav
cy=policy Client 10.234.136.55 2016
******** Login Menu ********

[Enter UserName -> Admin
[Enter Password ->
Wrong password!
[hui@in-csci-rrpc03 csci50700_spring2018_marketplace]$ 
```

Figure 10 Login with Incorrect Password (Client Side)

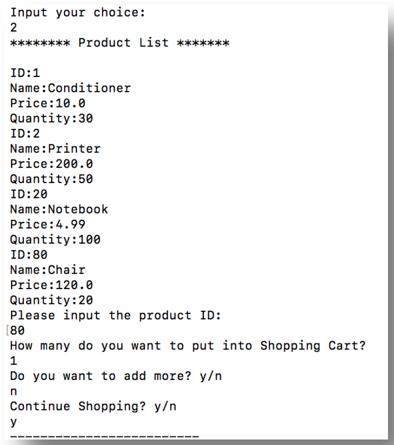
4) Choice 1: Browse Product Inventory



```
Input your choice:
1
******** Product List *******

ID:1
Name:Shampoo
Price:10.0
Quantity:200
ID:2
Name:Printer
Price:200.0
Quantity:50
ID:20
Name:Notebook
Price:4.99
Quantity:100
ID:1056
Name:Desk
Price:259.0
Quantity:30
Continue Shopping? y/n
```

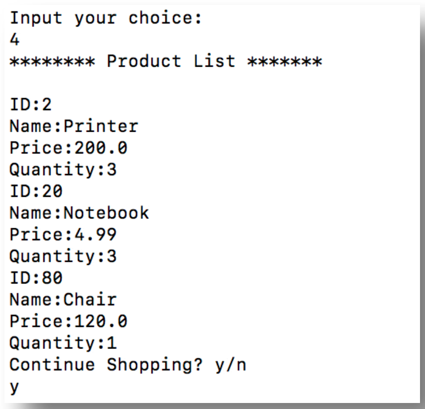
Figure 11 Browse Product Inventory

5) Choice 2: Add Product into Shopping Cart



```
Input your choice:
2
******** Product List *******

ID:1
Name:Conditioner
Price:10.0
Quantity:30
ID:2
Name:Printer
Price:200.0
Quantity:50
ID:20
Name:Notebook
Price:4.99
Quantity:100
ID:80
Name:Chair
Price:120.0
Quantity:20
Please input the product ID:
[80
How many do you want to put into Shopping Cart?
1
Do you want to add more? y/n
n
Continue Shopping? y/n
y
-------------------------
```

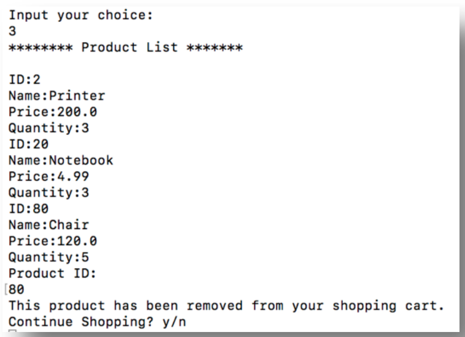
Figure 12 Add Product (ID: 80, quantity: 1) into Cart



```
Input your choice:
4
******** Product List *******

ID:2
Name:Printer
Price:200.0
Quantity:3
ID:20
Name:Notebook
Price:4.99
Quantity:3
ID:80
Name:Chair
Price:120.0
Quantity:1
Continue Shopping? y/n
y
```

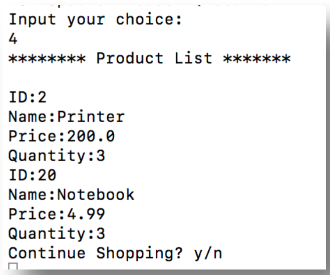
Figure 13 Updated Cart

6) Choice 3: Remove Product from Shopping Cart



```
Input your choice:
3
******** Product List *******

ID:2
Name:Printer
Price:200.0
Quantity:3
ID:20
Name:Notebook
Price:4.99
Quantity:3
ID:80
Name:Chair
Price:120.0
Quantity:5
Product ID:
80
This product has been removed from your shopping cart.
Continue Shopping? y/n
```

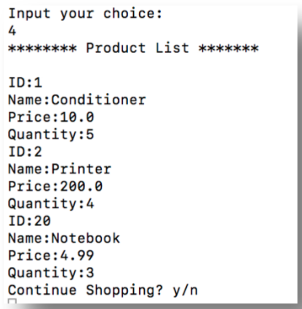
Figure 14 Before Remove Product (ID: 80)



```
Input your choice:
4
******** Product List *******

ID:2
Name:Printer
Price:200.0
Quantity:3
ID:20
Name:Notebook
Price:4.99
Quantity:3
Continue Shopping? y/n
```

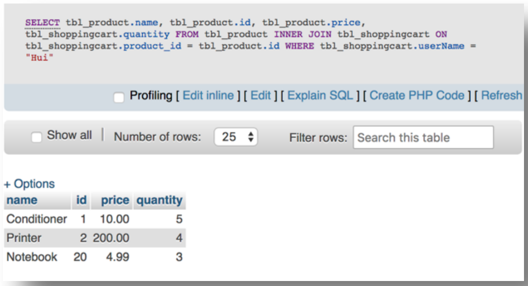
Figure 15 After Removal

7) Choice 4: Browse Shopping Cart under Username "Hui"



```
Input your choice:
4
******** Product List *******

ID:1
Name:Conditioner
Price:10.0
Quantity:5
ID:2
Name:Printer
Price:200.0
Quantity:4
ID:20
Name:Notebook
Price:4.99
Quantity:3
Continue Shopping? y/n
```

Figure 16 Browse Shopping Cart under Username "Hui"

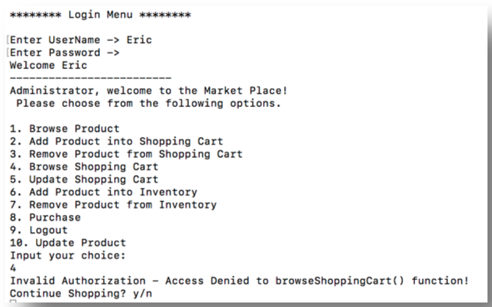


```
SELECT tbl_product.name, tbl_product.id, tbl_product.price,
tbl_shoppingcart.quantity FROM tbl_product INNER JOIN tbl_shoppingcart ON
tbl_shoppingcart.product_id = tbl_product.id WHERE tbl_shoppingcart.userName =
"Hui"
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP Code] [Refresh
Show all
Number of rows: 25
Filter rows: Search this table
+ Options

name	id	price	quantity
Conditioner	1	10.00	5
Printer	2	200.00	4
Notebook	20	4.99	3

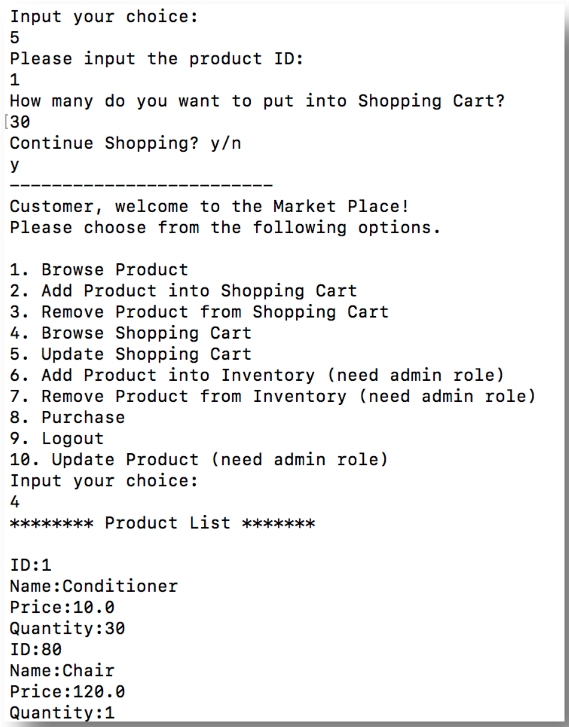
Figure 17 Browse Shopping Cart in Database under Username "Hui"



```
******** Login Menu ********

Enter UserName -> Eric
Enter Password ->
Welcome Eric
------------------------
Administrator, welcome to the Market Place!
 Please choose from the following options.

1. Browse Product
2. Add Product into Shopping Cart
3. Remove Product from Shopping Cart
4. Browse Shopping Cart
5. Update Shopping Cart
6. Add Product into Inventory
7. Remove Product from Inventory
8. Purchase
9. Logout
10. Update Product
Input your choice:
4
Invalid Authorization - Access Denied to browseShoppingCart() function!
Continue Shopping? y/n
```

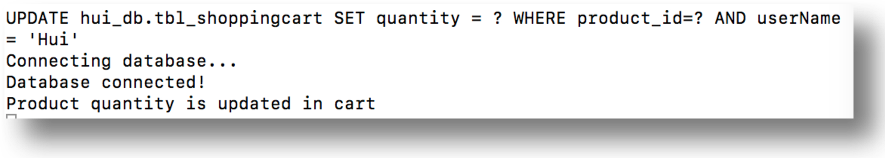
Figure 18 Browse Shopping Cart Denied under Admin Account

8) Choice 5: Update Shopping Cart



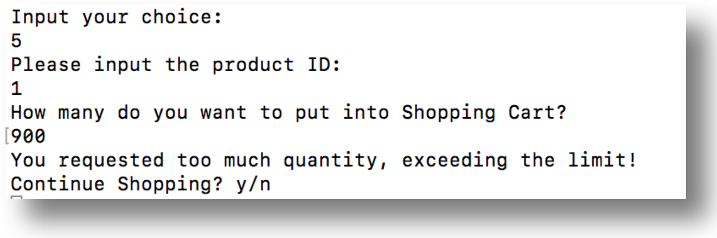
```
Input your choice:
5
Please input the product ID:
1
How many do you want to put into Shopping Cart?
30
Continue Shopping? y/n
y
-------------------------
Customer, welcome to the Market Place!
Please choose from the following options.

1. Browse Product
2. Add Product into Shopping Cart
3. Remove Product from Shopping Cart
4. Browse Shopping Cart
5. Update Shopping Cart
6. Add Product into Inventory (need admin role)
7. Remove Product from Inventory (need admin role)
8. Purchase
9. Logout
10. Update Product (need admin role)
Input your choice:
4
******** Product List *******

ID:1
Name:Conditioner
Price:10.0
Quantity:30
ID:80
Name:Chair
Price:120.0
Quantity:1
```

Figure 19 Update Product Quantity in Shopping Cart (ID:1, Quantity:30)



```
UPDATE hui_db.tbl_shoppingcart SET quantity = ? WHERE product_id=? AND userName
= 'Hui'
Connecting database...
Database connected!
Product quantity is updated in cart
```

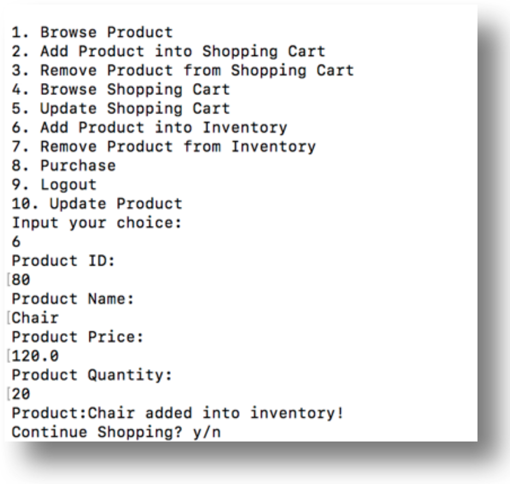
Figure 20 Update Product Quantity in Shopping Cart (Server Side Message)



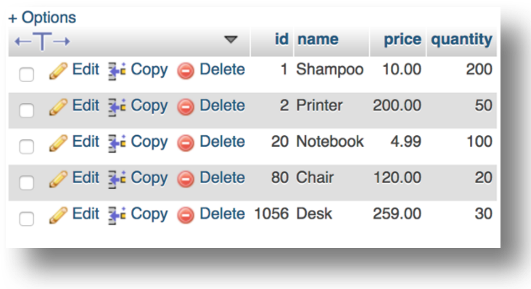
```
Input your choice:
5
Please input the product ID:
1
How many do you want to put into Shopping Cart?
900
You requested too much quantity, exceeding the limit!
Continue Shopping? y/n
```

Figure 21 Request Quantity Exceed Inventory

9) Choice 6: Add Product into Inventory

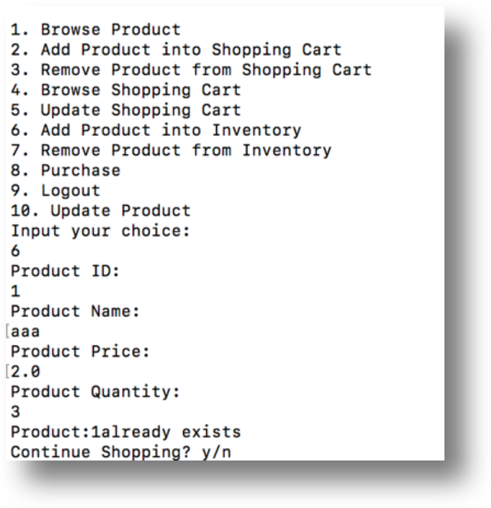


```
1. Browse Product
2. Add Product into Shopping Cart
3. Remove Product from Shopping Cart
4. Browse Shopping Cart
5. Update Shopping Cart
6. Add Product into Inventory
7. Remove Product from Inventory
8. Purchase
9. Logout
10. Update Product
Input your choice:
6
Product ID:
80
Product Name:
Chair
Product Price:
120.0
Product Quantity:
20
Product:Chair added into inventory!
Continue Shopping? y/n
```



	id	name	price	quantity
Edit Copy Delete	1	Shampoo	10.00	200
Edit Copy Delete	2	Printer	200.00	50
Edit Copy Delete	20	Notebook	4.99	100
Edit Copy Delete	80	Chair	120.00	20
Edit Copy Delete	1056	Desk	259.00	30

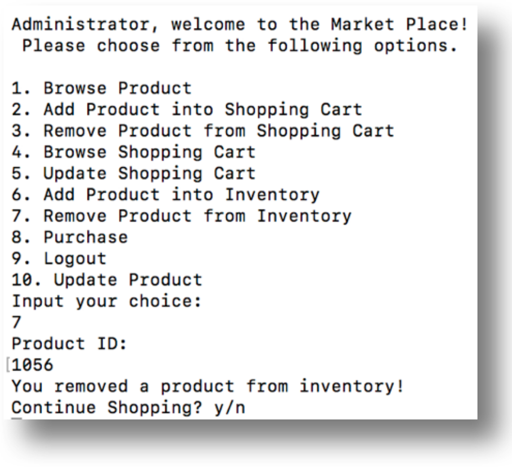
Figure 22 Chair Added into Database *Figure 23 Add Chair (ID:80) into Inventory*



```
1. Browse Product
2. Add Product into Shopping Cart
3. Remove Product from Shopping Cart
4. Browse Shopping Cart
5. Update Shopping Cart
6. Add Product into Inventory
7. Remove Product from Inventory
8. Purchase
9. Logout
10. Update Product
Input your choice:
6
Product ID:
1
Product Name:
aaa
Product Price:
2.0
Product Quantity:
3
Product:1already exists
Continue Shopping? y/n
```

Figure 24 Add Product Failed (ID: 1 Already Exist)

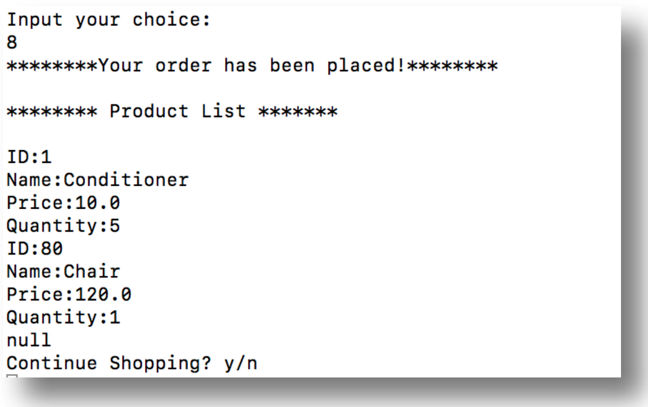
10) Choice 7: Remove Product from Inventory



```
Administrator, welcome to the Market Place!
 Please choose from the following options.

1. Browse Product
2. Add Product into Shopping Cart
3. Remove Product from Shopping Cart
4. Browse Shopping Cart
5. Update Shopping Cart
6. Add Product into Inventory
7. Remove Product from Inventory
8. Purchase
9. Logout
10. Update Product
Input your choice:
7
Product ID:
1056
You removed a product from inventory!
Continue Shopping? y/n
```

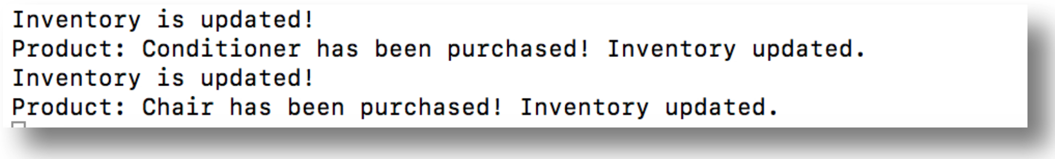
Figure 25 Remove Product (ID: 1056) from Inventory

11) Choice 8: Purchase



```
Input your choice:
8
********Your order has been placed!********

******** Product List *******

ID:1
Name:Conditioner
Price:10.0
Quantity:5
ID:80
Name:Chair
Price:120.0
Quantity:1
null
Continue Shopping? y/n
```

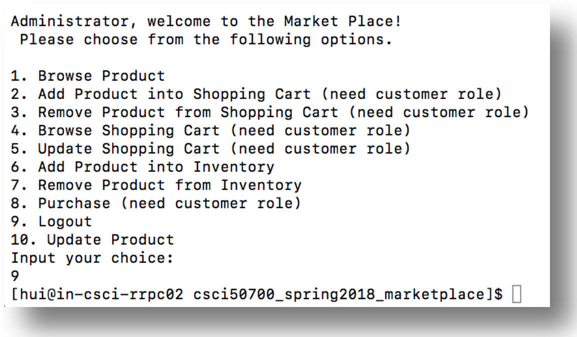
Figure 26 Purchase Shopping Cart



```
Inventory is updated!
Product: Conditioner has been purchased! Inventory updated.
Inventory is updated!
Product: Chair has been purchased! Inventory updated.
```

Figure 27 Purchase Shopping Cart (Server Side Message)

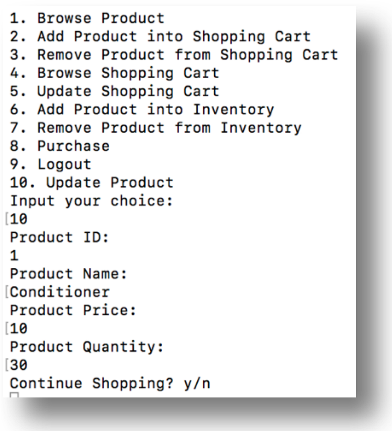
12) Choice 9: Logout



```
Administrator, welcome to the Market Place!
 Please choose from the following options.

1. Browse Product
2. Add Product into Shopping Cart (need customer role)
3. Remove Product from Shopping Cart (need customer role)
4. Browse Shopping Cart (need customer role)
5. Update Shopping Cart (need customer role)
6. Add Product into Inventory
7. Remove Product from Inventory
8. Purchase (need customer role)
9. Logout
10. Update Product
Input your choice:
9
[hui@in-csci-rrpc02 csci50700_spring2018_marketplace]$
```

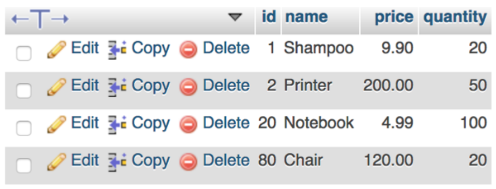
Figure 28 Exit Application

13) Choice 10: Update Product



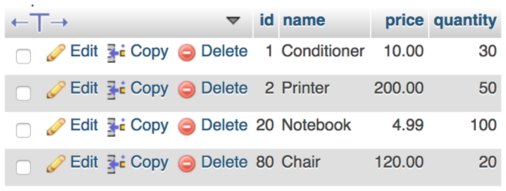
```
1. Browse Product
2. Add Product into Shopping Cart
3. Remove Product from Shopping Cart
4. Browse Shopping Cart
5. Update Shopping Cart
6. Add Product into Inventory
7. Remove Product from Inventory
8. Purchase
9. Logout
10. Update Product
Input your choice:
10
Product ID:
1
Product Name:
Conditioner
Product Price:
10
Product Quantity:
30
Continue Shopping? y/n
```

Figure 29 Update Product (ID:1)



	id	name	price	quantity
Edit Copy Delete	1	Shampoo	9.90	20
Edit Copy Delete	2	Printer	200.00	50
Edit Copy Delete	20	Notebook	4.99	100
Edit Copy Delete	80	Chair	120.00	20

Figure 24 Before Update



	id	name	price	quantity
Edit Copy Delete	1	Conditioner	10.00	30
Edit Copy Delete	2	Printer	200.00	50
Edit Copy Delete	20	Notebook	4.99	100
Edit Copy Delete	80	Chair	120.00	20

Figure 30 Updated (name, price, quantity)

5. Final Conclusions

1) What do I like about the assignment?
   My favorite thing about this assignment is that I got a chance to apply several design patterns in one product. This helps me get a better understanding of how these patterns cooperate to achieve some goal.
   I used to know little about operation system, For the second half of the project, together with the lecture, I learnt quite a lot about process, thread, concurrency and synchronization. I believe this can be a good start before I take an OS course.

2) What do I dislike about the assignment?
   The project is built upon all the previous work. This is a good thing to enhance the knowledge we learnt, but is also a pain to modify our design again and again. In the beginning, I have little experience in design patterns, and it takes so many nights to learn a new pattern and implement it in our application. Once I got feedback that I need to deal with problems in my design, I would be extra stressed. : P

3) What would you change about your design if I could go back? Why?
   I will probably change my design of RMI interface. I think my design used RMI in a wrong way especially after listening to others' presentations. Currently, I have the RMI interface specially for login, and all the other user actions are registered in another interface. There must be a better way to reorganize the structure, maybe I should declare all the methods (user actions) under the remote interface.