ONLINE MARKETPLACE

Assignment 5

Samira Khorshidi

TABLE OF CONTENTS

ASSIGNMENT DISCUSSION

The purpose of this practice is designing Domain Model of online Marketplace using Java RMI and MVC design pattern and by leveraging useful and proper design patterns.

OVERVIEW, IMPLEMENTATION

The goal of this assignment is having a synchronized distributed system using available tools and patterns. To achieve such a system, we first defined the synchronization as having order in execution of clients requests and define our critical blocks. In this implementation, the methods are well defined tasks with specific rule within entire system, so we decide to have each of them as a block that needs to be synchronized.

In this implementation we found **synchronized** keyword sufficient in order to achieve a desire system. As a test of synchronization, we put a function call in line 33 of FrontController which call server.concurrencyTestSync(); in a synchronized fashion.

To use database in our system we defined a specific class named "databaseManager" and the only classes that are using an object of this class are models, UserModel and ProductModel. So we separate our database layer from other parts of systems.

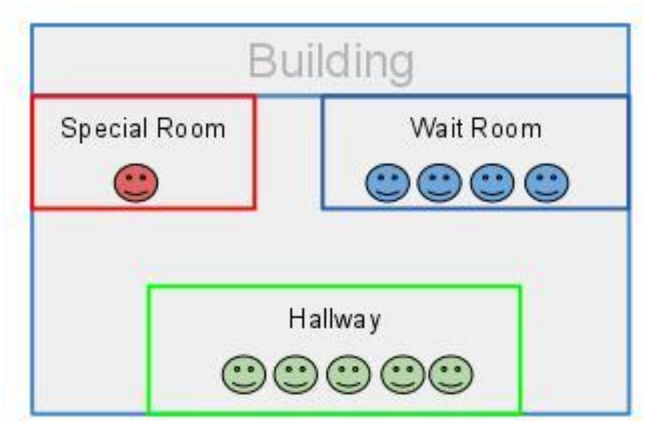
In following we will explain synchronization briefly:

SYNCHRONIZATION AND MONITOR CONCEPT

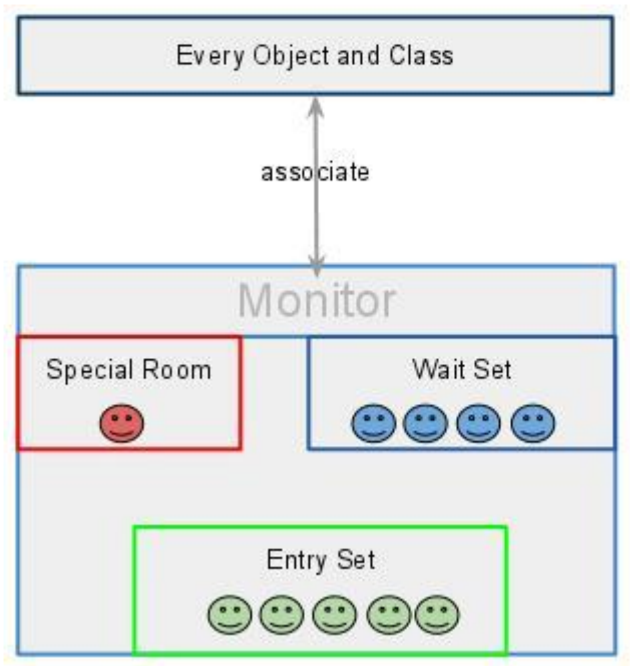
Monitor is an important concept of synchronization in operating systems. It is also used in Java synchronization, in following we will go through monitor concept and synchronization.

1. What is a Monitor?

A monitor can be considered as a building which contains a special room. The special room can be occupied by only one customer(thread) at a time. The room usually contains some data and code.



If a customer wants to occupy the special room, he has to enter the Hallway(Entry Set) to wait first. Scheduler will pick one based on some criteria(e.g. FIFO). If he is suspended for some reason, he will be sent to the wait room, and be scheduled to reenter the special room later. As it is shown in the diagram above, there are 3 rooms in this building.



In brief, a monitor is a facility which monitors the threads' access to the special room. It ensures that only one thread can access the protected data or code.

2. How is it implemented in Java?

In the Java virtual machine, every object and class is logically associated with a monitor. To implement the mutual exclusion capability of monitors, a lock (sometimes called a mutex) is associated with each object and class. This is called a semaphore in operating systems books, mutex is a binary semaphore.

If one thread owns a lock on some data, then no others can obtain that lock until the thread that owns the lock releases it. It would be not convenient if we need to write a semaphore all the time when we do multi-threading programming. Luckily, we don't need to since JVM does that for us automatically.

To claim a monitor region which means data not accessible by more than one thread, Java provide synchronized statements and synchronized methods. Once the code is embedded with synchronized keyword, it is a monitor region. The locks are implemented in the background automatically by JVM.

3. In Java synchronization code, which part is monitor?

We know that each object/class is associated with a Monitor. I think it is good to say that each object has a monitor, since each object could have its own critical section, and capable of monitoring the thread sequence.

To enable collaboration of different threads, Java provide wait() and notify() to suspend a thread and to wake up another thread that are waiting on the object respectively. In addition, there are 3 other versions:

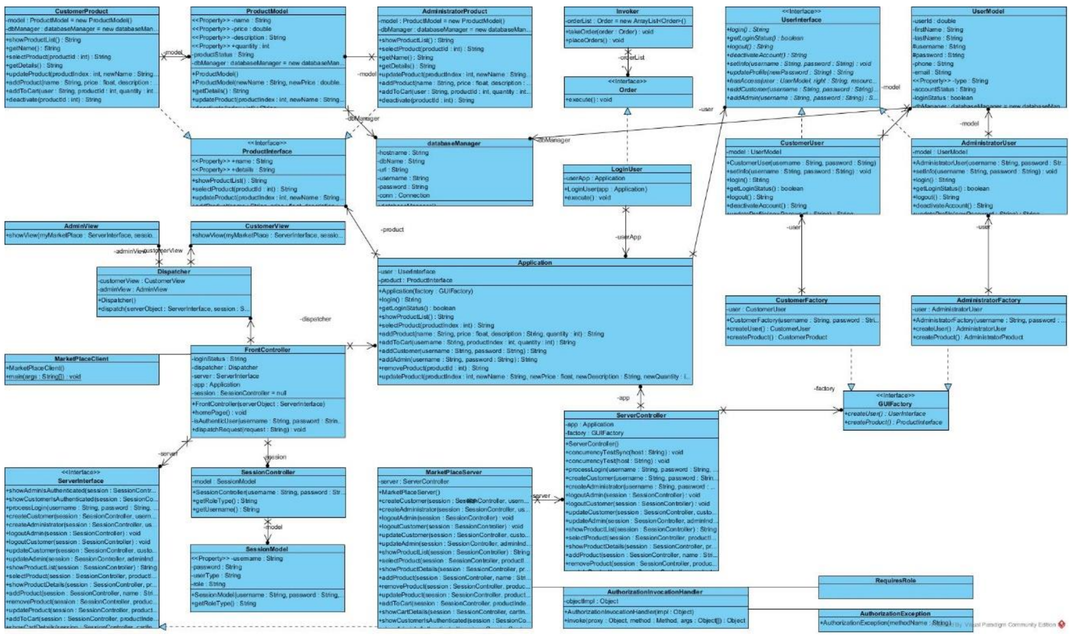
- wait(**long** timeout, **int** nanos)
- wait(**long** timeout) notified by other threads or notified by timeout.
- notify(all)

Those methods can only be invoked within a synchronized statement or synchronized method. The reason is that if a method does not require mutual exclusion, there is no need to monitor or collaborate between threads, every thread can access that method freely.

FIGURE,
SAMPLE CODES

In following we provide you a Class Diagram of our system along with some sample codes.

UML DIAGRAM



The original UML diagram has been provided in Documentation directory. As you see, in our proposed architecture, we introduced a new class named SessionModel and also a temporary databaseManager class.

SAMPLE CODES

In following we provide a sample code of the different new functionalities of the system:

Add a Administrator/Customer user:

```
public String addUser(String newUsername, String newPassword, String
newType){

        String commandStatus = null;

        try{

            String stm = String.format("INSERT INTO user(username,
password, type, accountStatus) VALUES
(' %s','%s','%s','active');",newUsername,newPassword, newType);

            System.out.println(stm);

            int result = dbManager.updateMyRecord(stm);

            if(result>0)

                commandStatus = "User has been added
successfully!";

            else

                commandStatus = "Please try again later!";

        }catch(Exception e){

            commandStatus = "Please try again later!";
```

- *Remove product:*

```
	public String deactivate(int index) {
		String commandStatus = null;
		try{
			String product_stm = String.format("UPDATE product SET 
status = 'deactive' WHERE productId = %s",index);
			int result = dbManager.updateMyRecord(product_stm);
			System.out.println(result);
			if(result>0){
				commandStatus = "Product has been removed 
successfully!";

			} else{
				commandStatus = "Please try again later!";
			}
			return commandStatus;
		}catch(Exception e){
			commandStatus = "Please try again later!";
			System.out.println("Database Exception" + e.getMessage());
		}
```

- *Update Product:*

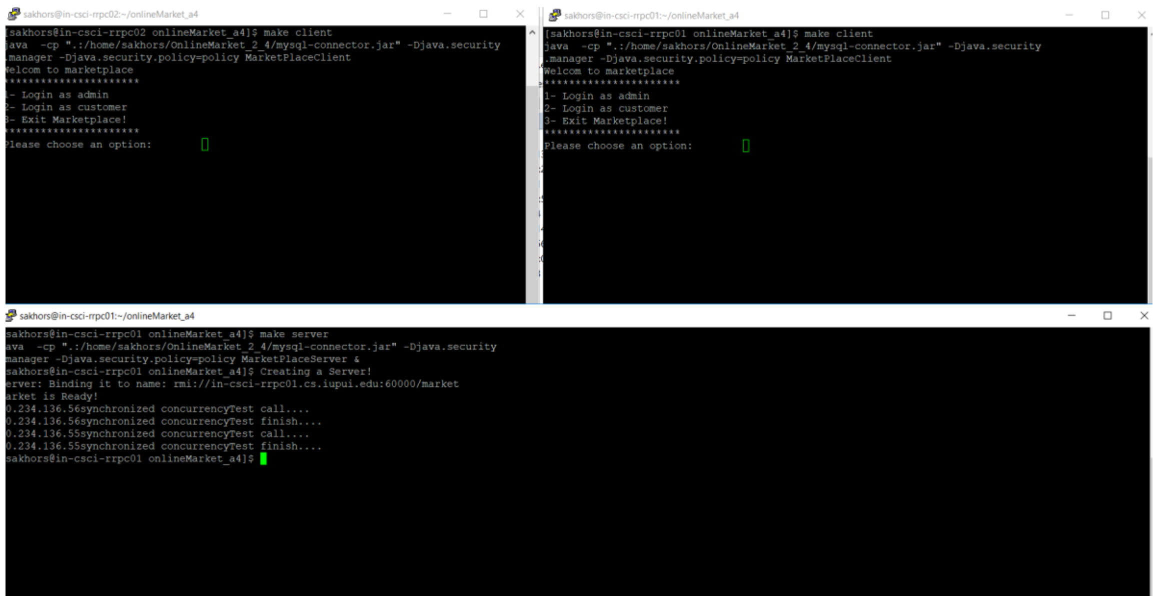
```
public String updateProduct(int productIndex, String newName, float 
newPrice, String newDescription, int newQuantity){

		String commandStatus = null;

		try{

			String product_stm = String.format("UPDATE product 
SET name = '%s', price=%s, description='%s', quantity = %s  WHERE 
productId = %s",newName, newPrice, newDescription, newQuantity, 
productIndex);

			int result = 
dbManager.updateMyRecord(product_stm);

			System.out.println(result);
```

SAMPLE RUNS

In this section you'll see screenshots of sample runs. The system is minimal, so there is no GUI, instead every interaction happen using command prompt.

SCREENSHOTS

All required functionalities has been implemented and here is a sample run of an action from two different machine in synchronized way:

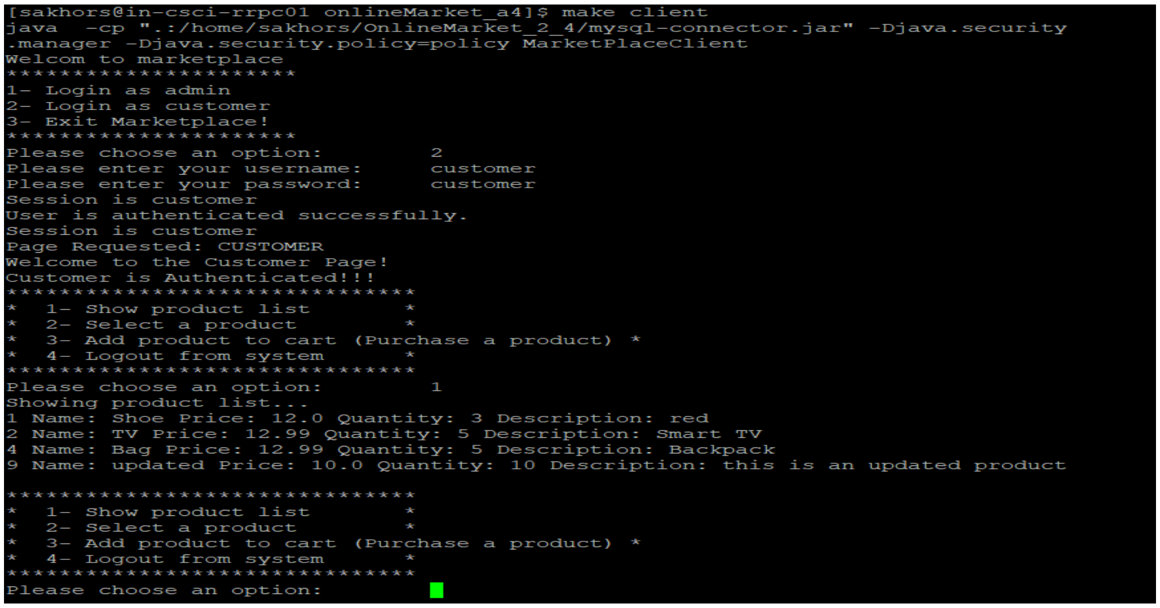

sakhors@in-csci-rrpc02:~/onlineMarket_a4
sakhors@in-csci-rrpc01:~/onlineMarket_a4
sakhors@in-csci-rrpc01:~/onlineMarket_a4

```
[sakhors@in-csci-rrpc02 onlineMarket_a4]$ make client
java  -cp ".:/home/sakhors/OnlineMarket_2_4/mysql-connector.jar" -Djava.security
.manager -Djava.security.policy=policy MarketPlaceClient
Welcom to marketplace
**********************
1- Login as admin
2- Login as customer
3- Exit Marketplace!
**********************
Please choose an option:
```

```
[sakhors@in-csci-rrpc01 onlineMarket_a4]$ make client
java  -cp ".:/home/sakhors/OnlineMarket_2_4/mysql-connector.jar" -Djava.security
.manager -Djava.security.policy=policy MarketPlaceClient
Welcom to marketplace
**********************
1- Login as admin
2- Login as customer
3- Exit Marketplace!
**********************
Please choose an option:
```

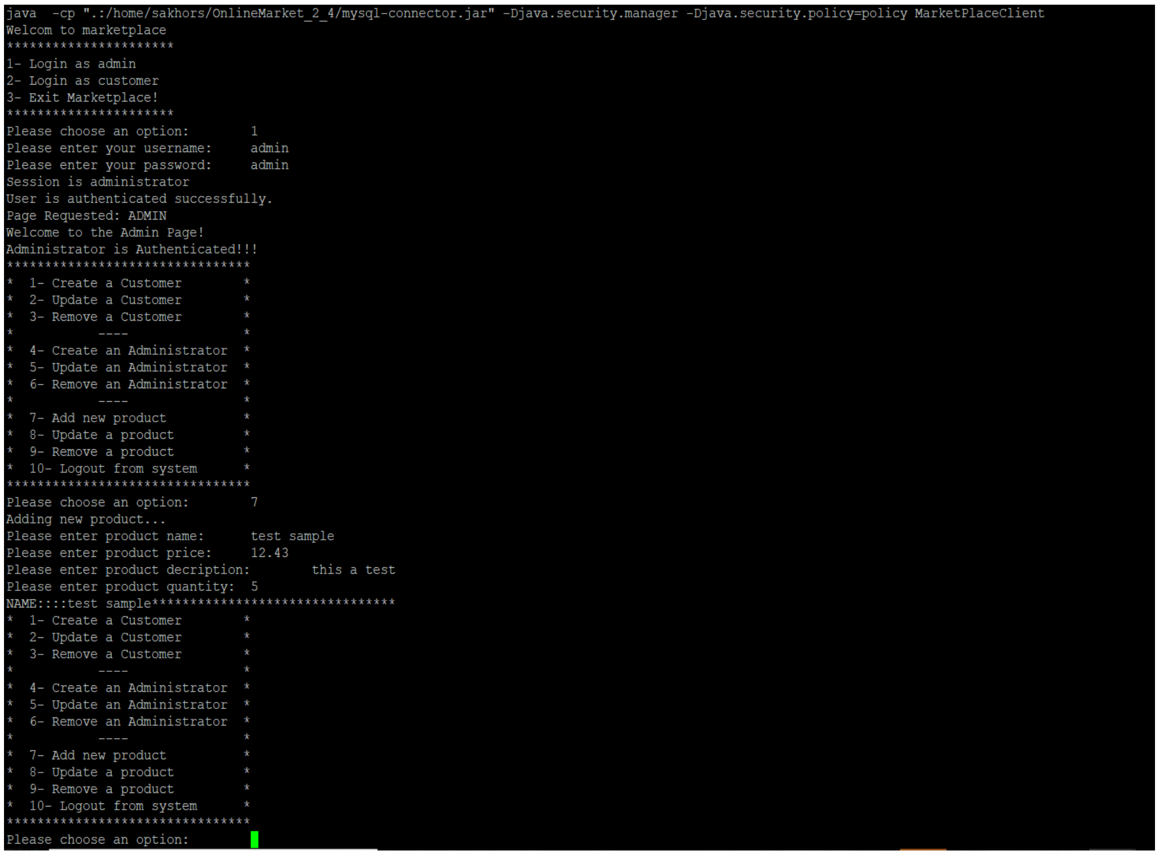
```
sakhors@in-csci-rrpc01 onlineMarket_a4]$ make server
ava  -cp ".:/home/sakhors/OnlineMarket_2_4/mysql-connector.jar" -Djava.security
manager -Djava.security.policy=policy MarketPlaceServer &
sakhors@in-csci-rrpc01 onlineMarket_a4]$ Creating a Server!
erver: Binding it to name: rmi://in-csci-rrpc01.cs.iupui.edu:60000/market
arket is Ready!
0.234.136.56synchronized concurrencyTest call....
0.234.136.56synchronized concurrencyTest finish....
0.234.136.55synchronized concurrencyTest call....
0.234.136.55synchronized concurrencyTest finish....
sakhors@in-csci-rrpc01 onlineMarket_a4]$
```

This is a sample run of browsing system products from database:



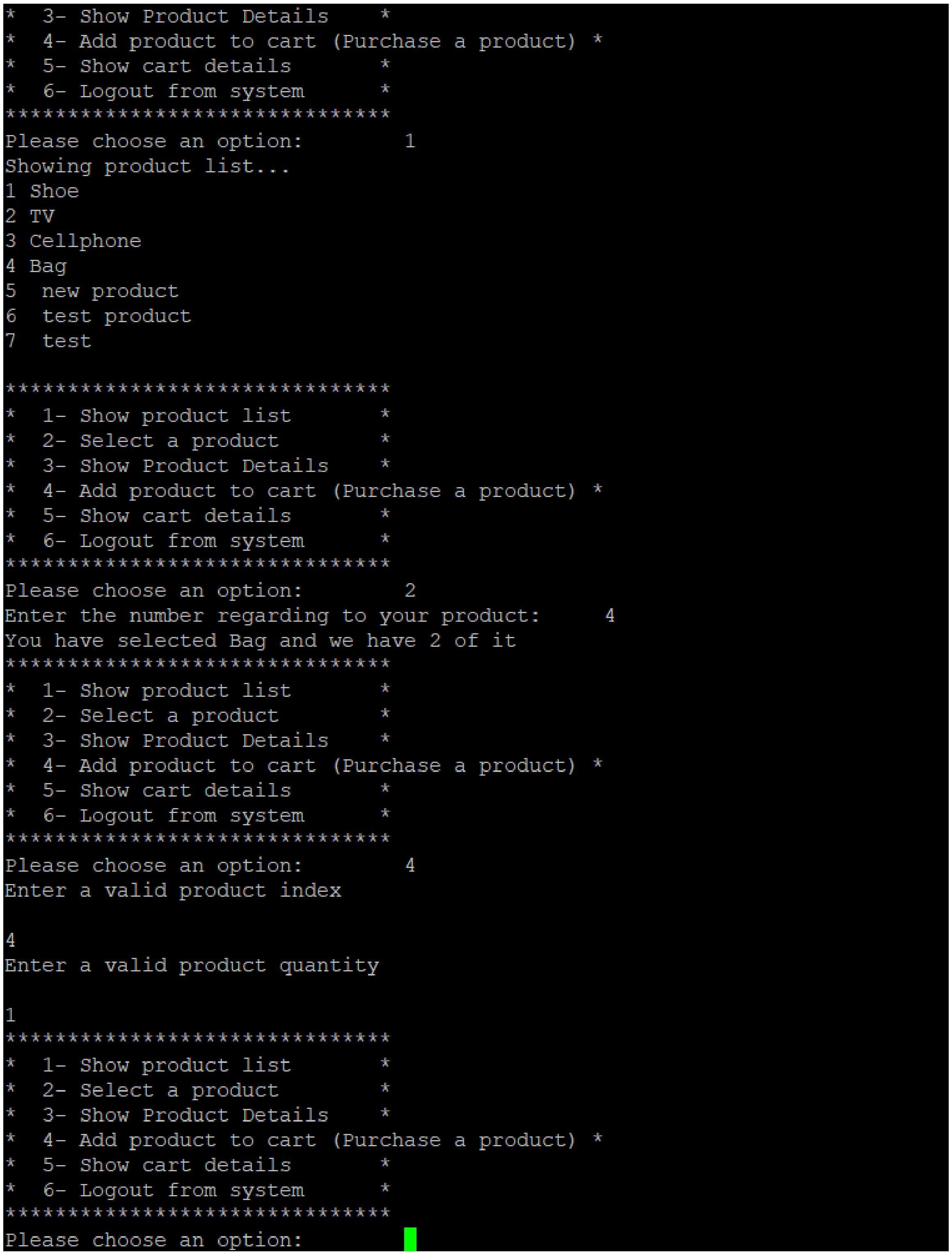
```
[sakhors@in-csci-rrpc01 onlineMarket_a4]$ make client
java  -cp ".:/home/sakhors/OnlineMarket_2_4/mysql-connector.jar" -Djava.security
.manager -Djava.security.policy=policy MarketPlaceClient
Welcom to marketplace
**********************
1- Login as admin
2- Login as customer
3- Exit Marketplace!
**********************
Please choose an option:        2
Please enter your username:     customer
Please enter your password:     customer
Session is customer
User is authenticated successfully.
Session is customer
Page Requested: CUSTOMER
Welcome to the Customer Page!
Customer is Authenticated!!!
*******************************
*   1- Show product list        *
*   2- Select a product         *
*   3- Add product to cart (Purchase a product) *
*   4- Logout from system       *
*******************************
Please choose an option:        1
Showing product list...
1 Name: Shoe Price: 12.0 Quantity: 3 Description: red
2 Name: TV Price: 12.99 Quantity: 5 Description: Smart TV
4 Name: Bag Price: 12.99 Quantity: 5 Description: Backpack
9 Name: updated Price: 10.0 Quantity: 10 Description: this is an updated product

*******************************
*   1- Show product list        *
*   2- Select a product         *
*   3- Add product to cart (Purchase a product) *
*   4- Logout from system       *
*******************************
Please choose an option:
```

This a sample run of adding a new product by admin user:



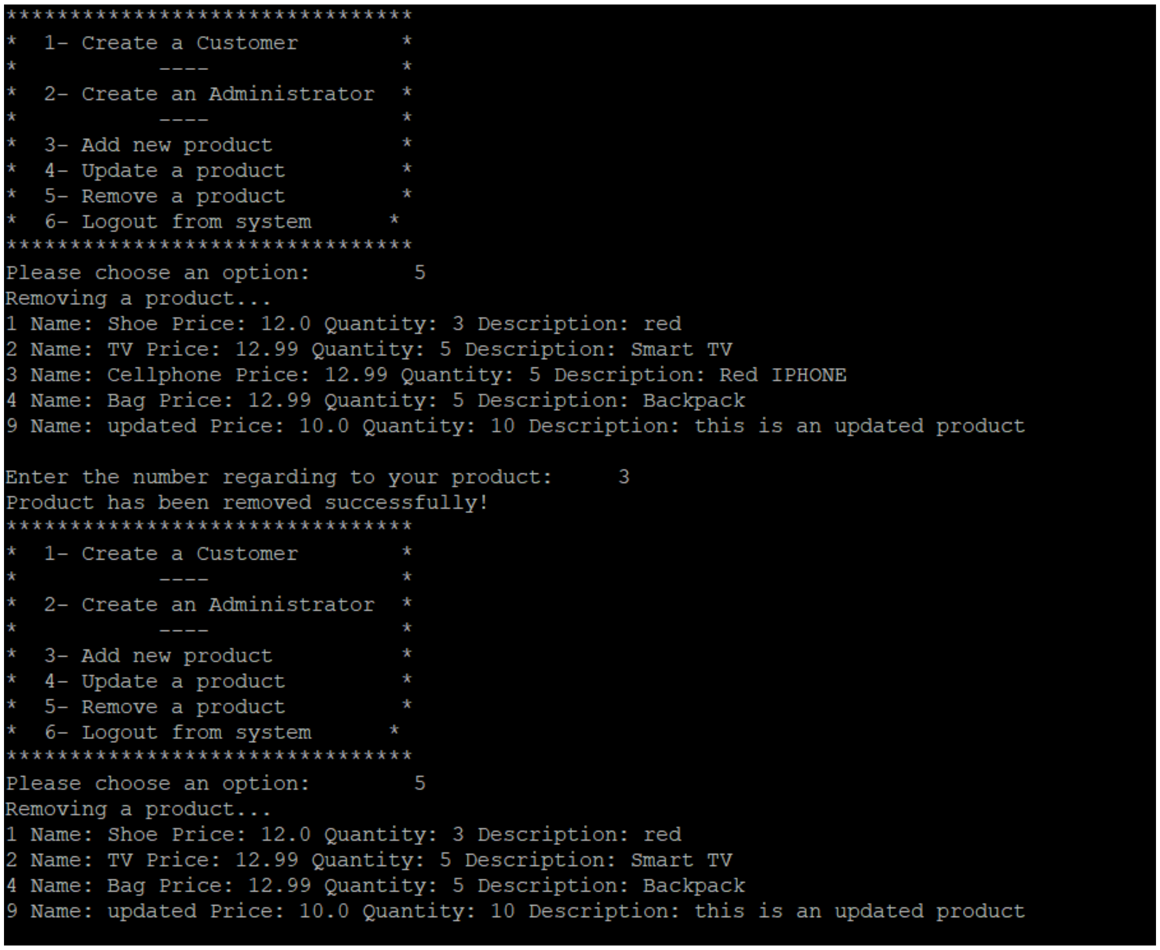
```
java  -cp ".:/home/sakhors/OnlineMarket_2_4/mysql-connector.jar" -Djava.security.manager -Djava.security.policy=policy MarketPlaceClient
Welcom to marketplace
**********************
1- Login as admin
2- Login as customer
3- Exit Marketplace!
**********************
Please choose an option:        1
Please enter your username:     admin
Please enter your password:     admin
Session is administrator
User is authenticated successfully.
Page Requested: ADMIN
Welcome to the Admin Page!
Administrator is Authenticated!!!
********************************
*  1- Create a Customer        *
*  2- Update a Customer        *
*  3- Remove a Customer        *
*        ----                  *
*  4- Create an Administrator  *
*  5- Update an Administrator  *
*  6- Remove an Administrator  *
*        ----                  *
*  7- Add new product          *
*  8- Update a product         *
*  9- Remove a product         *
*  10- Logout from system      *
********************************
Please choose an option:        7
Adding new product...
Please enter product name:      test sample
Please enter product price:     12.43
Please enter product decription:        this a test
Please enter product quantity:  5
NAME::::test sample********************************
*  1- Create a Customer        *
*  2- Update a Customer        *
*  3- Remove a Customer        *
*        ----                  *
*  4- Create an Administrator  *
*  5- Update an Administrator  *
*  6- Remove an Administrator  *
*        ----                  *
*  7- Add new product          *
*  8- Update a product         *
*  9- Remove a product         *
*  10- Logout from system      *
********************************
Please choose an option:
```

And this is a sample run of adding a product to the user shopping cart (Purchase):



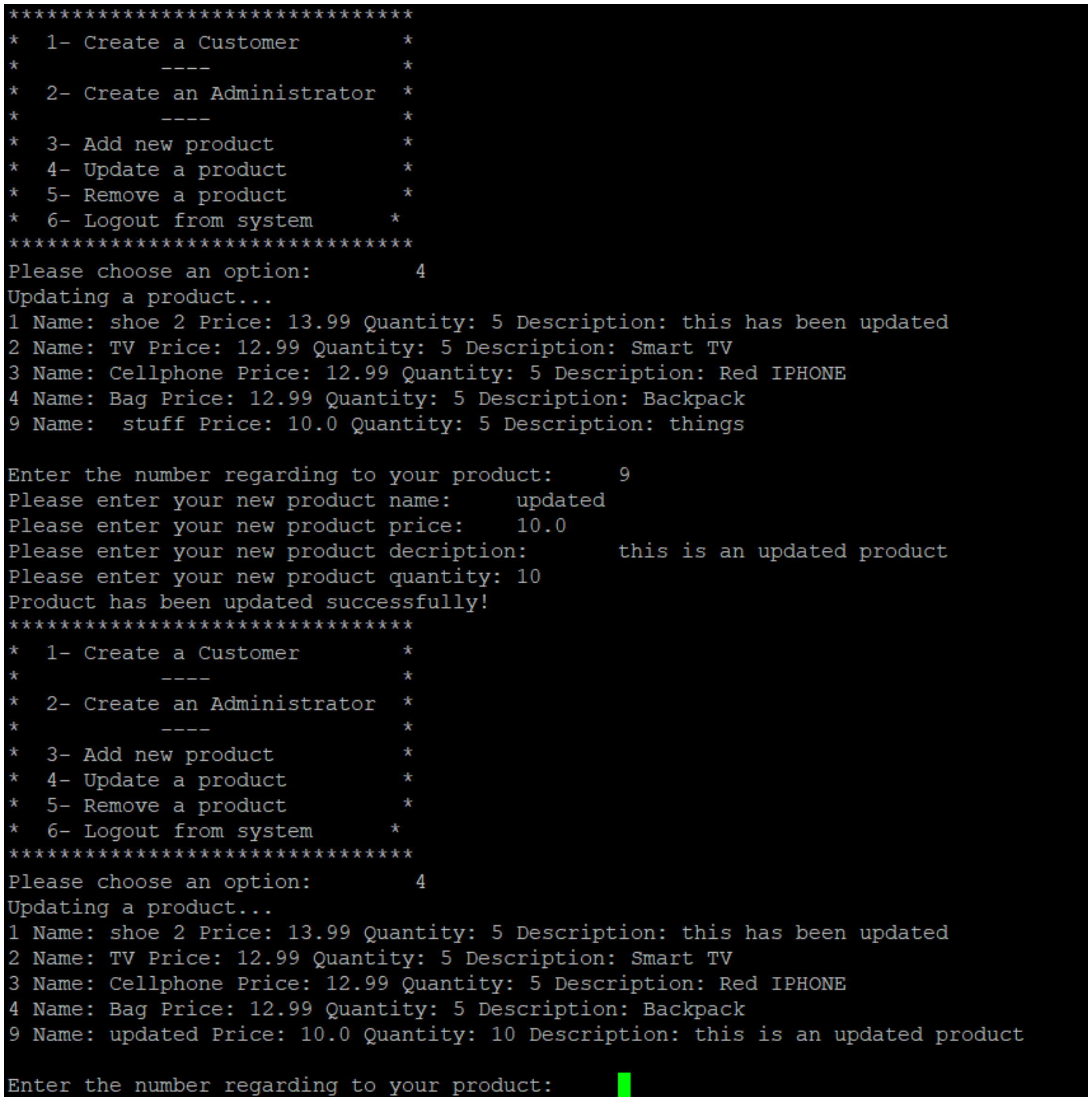
```
*   3- Show Product Details     *
*   4- Add product to cart (Purchase a product) *
*   5- Show cart details        *
*   6- Logout from system       *
*******************************
Please choose an option:        1
Showing product list...
1 Shoe
2 TV
3 Cellphone
4 Bag
5  new product
6  test product
7  test

*******************************
*   1- Show product list        *
*   2- Select a product         *
*   3- Show Product Details     *
*   4- Add product to cart (Purchase a product) *
*   5- Show cart details        *
*   6- Logout from system       *
*******************************
Please choose an option:        2
Enter the number regarding to your product:     4
You have selected Bag and we have 2 of it
*******************************
*   1- Show product list        *
*   2- Select a product         *
*   3- Show Product Details     *
*   4- Add product to cart (Purchase a product) *
*   5- Show cart details        *
*   6- Logout from system       *
*******************************
Please choose an option:        4
Enter a valid product index

4
Enter a valid product quantity

1
*******************************
*   1- Show product list        *
*   2- Select a product         *
*   3- Show Product Details     *
*   4- Add product to cart (Purchase a product) *
*   5- Show cart details        *
*   6- Logout from system       *
*******************************
Please choose an option:
```

Here is remove product sample run:



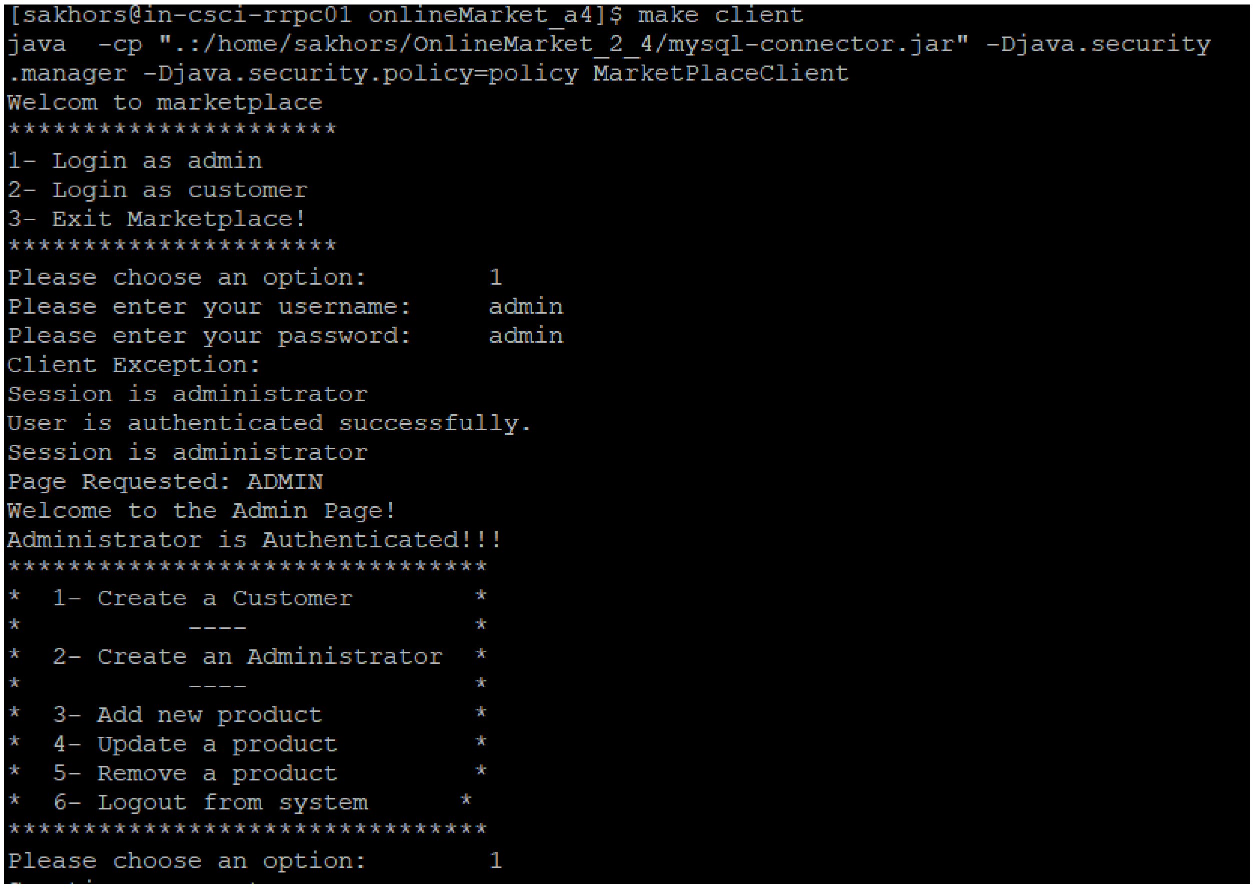
```
********************************
*  1- Create a Customer        *
*           ----               *
*  2- Create an Administrator  *
*           ----               *
*  3- Add new product          *
*  4- Update a product         *
*  5- Remove a product         *
*  6- Logout from system      *
********************************
Please choose an option:        5
Removing a product...
1 Name: Shoe Price: 12.0 Quantity: 3 Description: red
2 Name: TV Price: 12.99 Quantity: 5 Description: Smart TV
3 Name: Cellphone Price: 12.99 Quantity: 5 Description: Red IPHONE
4 Name: Bag Price: 12.99 Quantity: 5 Description: Backpack
9 Name: updated Price: 10.0 Quantity: 10 Description: this is an updated product

Enter the number regarding to your product:     3
Product has been removed successfully!
********************************
*  1- Create a Customer        *
*           ----               *
*  2- Create an Administrator  *
*           ----               *
*  3- Add new product          *
*  4- Update a product         *
*  5- Remove a product         *
*  6- Logout from system      *
********************************
Please choose an option:        5
Removing a product...
1 Name: Shoe Price: 12.0 Quantity: 3 Description: red
2 Name: TV Price: 12.99 Quantity: 5 Description: Smart TV
4 Name: Bag Price: 12.99 Quantity: 5 Description: Backpack
9 Name: updated Price: 10.0 Quantity: 10 Description: this is an updated product
```

Update product:



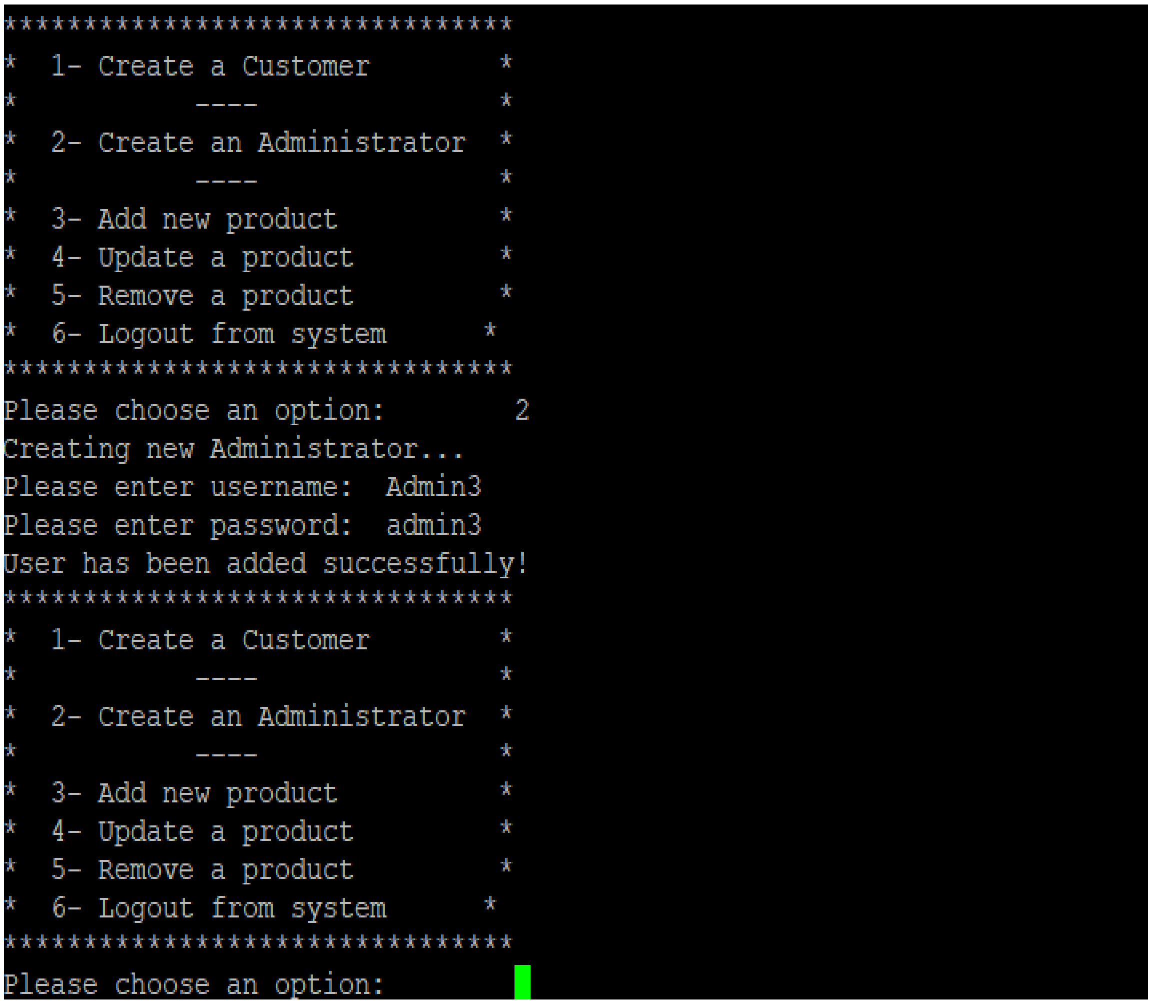
```
********************************
*  1- Create a Customer        *
*          ----                *
*  2- Create an Administrator  *
*          ----                *
*  3- Add new product          *
*  4- Update a product         *
*  5- Remove a product         *
*  6- Logout from system      *
********************************
Please choose an option:        4
Updating a product...
1 Name: shoe 2 Price: 13.99 Quantity: 5 Description: this has been updated
2 Name: TV Price: 12.99 Quantity: 5 Description: Smart TV
3 Name: Cellphone Price: 12.99 Quantity: 5 Description: Red IPHONE
4 Name: Bag Price: 12.99 Quantity: 5 Description: Backpack
9 Name:  stuff Price: 10.0 Quantity: 5 Description: things

Enter the number regarding to your product:     9
Please enter your new product name:     updated
Please enter your new product price:    10.0
Please enter your new product decription:       this is an updated product
Please enter your new product quantity: 10
Product has been updated successfully!
********************************
*  1- Create a Customer        *
*          ----                *
*  2- Create an Administrator  *
*          ----                *
*  3- Add new product          *
*  4- Update a product         *
*  5- Remove a product         *
*  6- Logout from system      *
********************************
Please choose an option:        4
Updating a product...
1 Name: shoe 2 Price: 13.99 Quantity: 5 Description: this has been updated
2 Name: TV Price: 12.99 Quantity: 5 Description: Smart TV
3 Name: Cellphone Price: 12.99 Quantity: 5 Description: Red IPHONE
4 Name: Bag Price: 12.99 Quantity: 5 Description: Backpack
9 Name: updated Price: 10.0 Quantity: 10 Description: this is an updated product

Enter the number regarding to your product:
```

In term of user functionalities, you can find login in following:



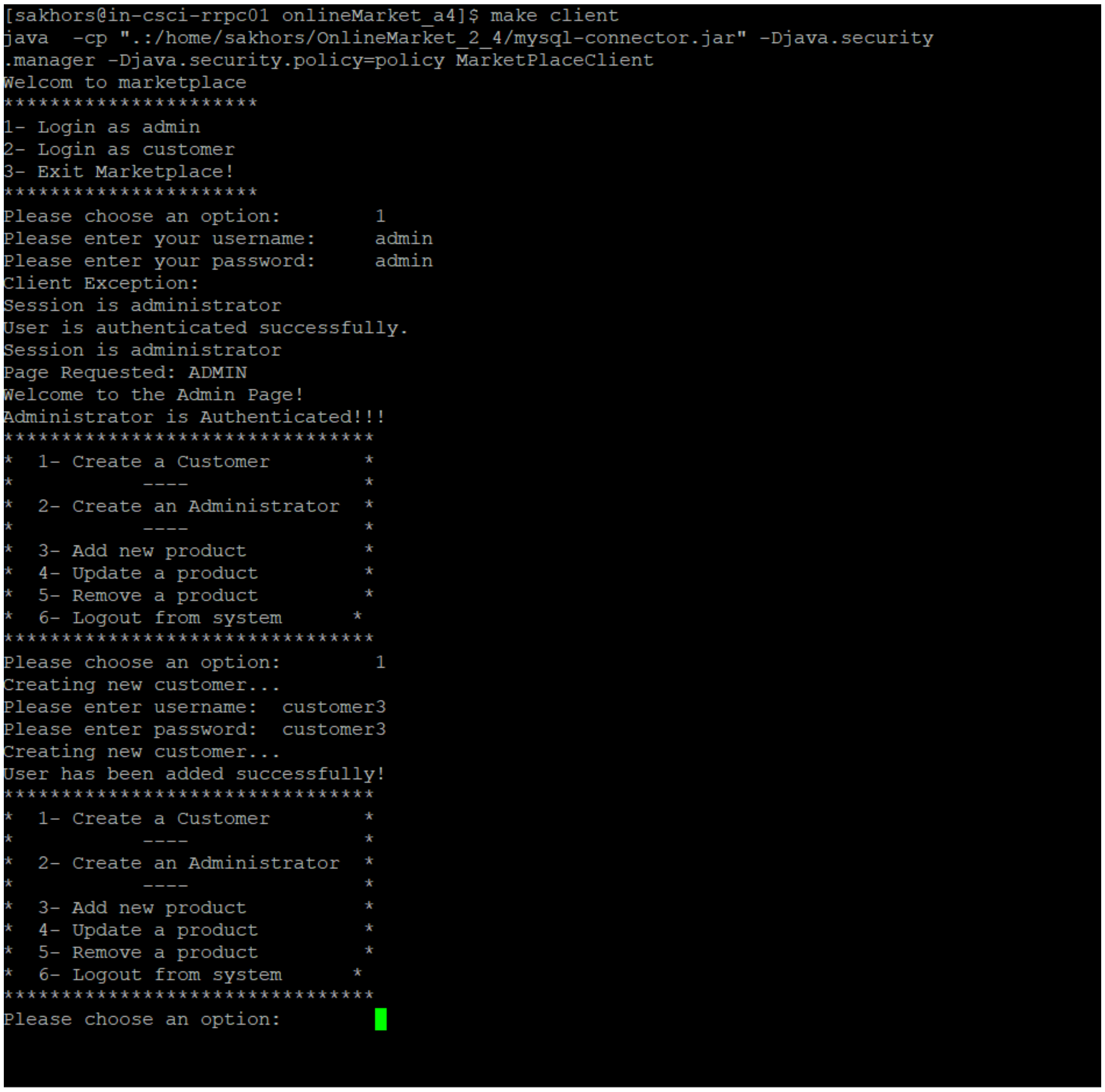
```
[sakhors@in-csci-rrpc01 onlineMarket_a4]$ make client
java  -cp ".:/home/sakhors/OnlineMarket_2_4/mysql-connector.jar" -Djava.security
.manager -Djava.security.policy=policy MarketPlaceClient
Welcom to marketplace
**********************
1- Login as admin
2- Login as customer
3- Exit Marketplace!
**********************
Please choose an option:        1
Please enter your username:     admin
Please enter your password:     admin
Client Exception:
Session is administrator
User is authenticated successfully.
Session is administrator
Page Requested: ADMIN
Welcome to the Admin Page!
Administrator is Authenticated!!!
********************************
*  1- Create a Customer        *
*           ----               *
*  2- Create an Administrator  *
*           ----               *
*  3- Add new product          *
*  4- Update a product         *
*  5- Remove a product         *
*  6- Logout from system      *
********************************
Please choose an option:        1
```

Add an Admin user:



```
********************************
*   1- Create a Customer        *
*           ----                *
*   2- Create an Administrator  *
*           ----                *
*   3- Add new product          *
*   4- Update a product         *
*   5- Remove a product         *
*   6- Logout from system      *
********************************
Please choose an option:        2
Creating new Administrator...
Please enter username:  Admin3
Please enter password:  admin3
User has been added successfully!
********************************
*   1- Create a Customer        *
*           ----                *
*   2- Create an Administrator  *
*           ----                *
*   3- Add new product          *
*   4- Update a product         *
*   5- Remove a product         *
*   6- Logout from system      *
********************************
Please choose an option:
```

Add a customer user:



```
[sakhors@in-csci-rrpc01 onlineMarket_a4]$ make client
java  -cp ".:/home/sakhors/OnlineMarket_2_4/mysql-connector.jar" -Djava.security
.manager -Djava.security.policy=policy MarketPlaceClient
Welcom to marketplace
**********************
1- Login as admin
2- Login as customer
3- Exit Marketplace!
**********************
Please choose an option:        1
Please enter your username:     admin
Please enter your password:     admin
Client Exception:
Session is administrator
User is authenticated successfully.
Session is administrator
Page Requested: ADMIN
Welcome to the Admin Page!
Administrator is Authenticated!!!
********************************
*  1- Create a Customer        *
*           ----               *
*  2- Create an Administrator  *
*           ----               *
*  3- Add new product          *
*  4- Update a product         *
*  5- Remove a product         *
*  6- Logout from system      *
********************************
Please choose an option:        1
Creating new customer...
Please enter username:  customer3
Please enter password:  customer3
Creating new customer...
User has been added successfully!
********************************
*  1- Create a Customer        *
*           ----               *
*  2- Create an Administrator  *
*           ----               *
*  3- Add new product          *
*  4- Update a product         *
*  5- Remove a product         *
*  6- Logout from system      *
********************************
Please choose an option:
```

DISCUSSION

During this assignment, we focused on applying the information we have learned regarding the use of synchronization in the Java to ensure that access to our shared resources are indeed thread-safe and also we complete our system functionalities by implementing different functionalities including login, update Product, add users (customer and administrator) and remove a product. Based on the concept behind purchase, browse and add products, and having multiple clients, we would like to have them in a synchronized way rather than concurrent way.

CONCLUTION
REFERENCES

CONCLUTION

In conclusions, we found that in a way that we used java JVM, RMI and synchronized keyword, even if they give us no guarantee in term of keeping our system thread safe, but we found that each client has its own thread pool and we achieved synchronized trade-safe system.

REFERENCES

- *https://dzone.com/articles/java-callable-future-understanding*
- *https://link.springer.com/chapter/10.1007/978-3-540-45209-6_63*
- *http://winterbe.com/posts/2015/04/30/java8-concurrency-tutorial-synchronized-locks-examples/*
- *https://docs.oracle.com/javase/tutorial/essential/concurrency/syncrgb.html*
- *https://docs.oracle.com/javase/tutorial/essential/concurrency/guardmeth.html*
- *https://docs.oracle.com/javase/tutorial/essential/concurrency/syncmeth.html*

- *http://winterbe.com/posts/2015/04/07/java8-concurrency-tutorial-thread-executor-examples/*
- *https://www.programcreek.com/2011/12/monitors-java-synchronization-mechanism/*
- *http://blog.e-zest.com/java-monitor-pattern/*
- *Java and developer's forums*
- *Lecture slides*