Market Place

Assignment Report 4

Virinchi Sainath Nalluri

vnalluri@iupui.edu

3rd April, 2018

Table of Contents

Objective

The objective of this assignment is mainly to understand concurrency in Java RMI, consequences of concurrency and the “guarantee” that the Java RMI provides or does not provide. Also to implement additional functionalities to the framework such as Purchase Item, Add Item and Browse Item.

Assignment Discussion

Concurrency in Java RMI

Concurrency is running two or more jobs by the same machine in the same time periods by alternating between these jobs. It is different from parallelism in which both jobs are run simultaneous. Concurrency is achieved using threads.

How Remote Method Invocation in Java RMI works?

The server class which has methods which are to be remotely accessed is first bound to a string and added to the registry. The client then looks up for this string from registry and get this remote object. The client then accesses this remote object as if it is any other object. However, (internally which even the client even doesn’t have to know) when the object’s method is called a stub in invoked which wires this method to the method in the server. The stub sends the arguments and gets back the return object (must be serializable to be sent over to the client side). This is how remote method invocation works in Java RMI.

What is the need of Concurrency in Java RMI?

However, this object in the server whose remote object is accessed by more than one clients might send remote method invocations to the server at the same time. In order to provide stable consistent service to all the clients, Java RMI internally uses threads to execute these jobs.

“3.2 Thread Usage in Remote Method Invocations

A method dispatched by the RMI runtime to a remote object implementation may or may not execute in a separate thread. The RMI runtime makes no guarantees with respect to mapping remote object invocations to threads. Since remote method invocation on the same remote object may execute concurrently, a remote object implementation needs to make sure its implementation is thread-safe."

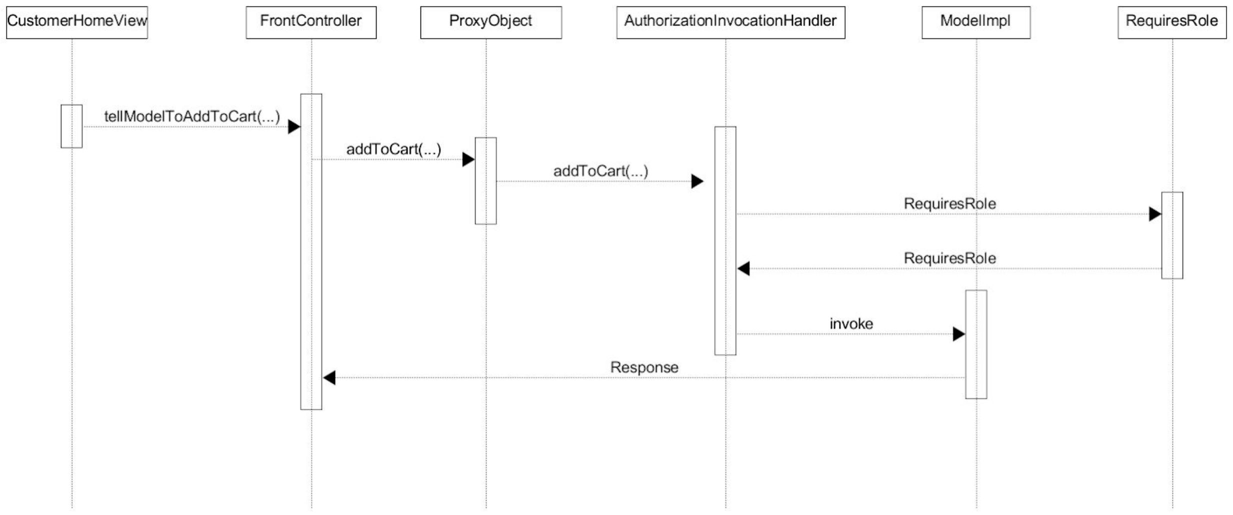
Elaborating the above text

There is a need or no need for Java RMI to create a new thread for every customers' every method invocation. It may or may not use the already existing threads to do these method invocations. The Java RMI doesn't "guarantee" that a new thread is created or how the mapping of thread to method invocation is done. However, it does say that if the remote method's implementation is made thread safe, then every client can execute his remote method invocation.

Consequences of Concurrency in Java RMI

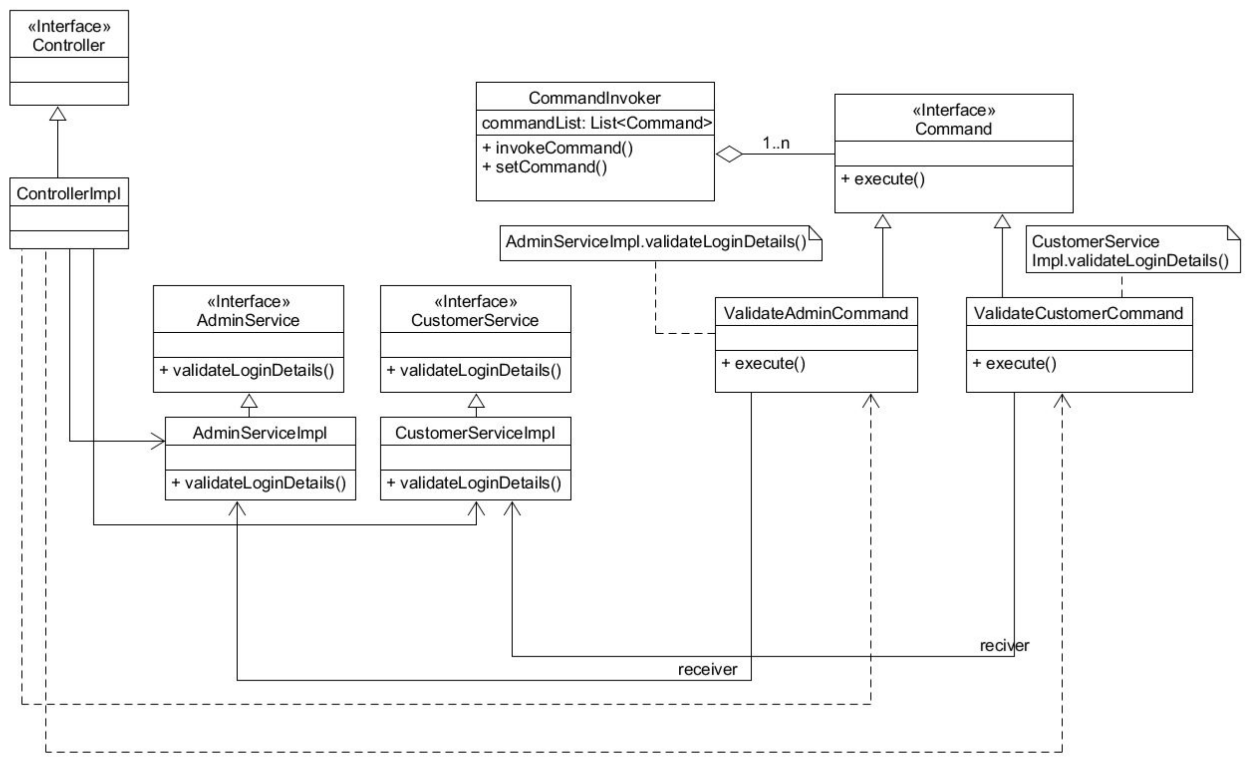
Though at first glance, it might look like a nice feature of Java RMI to give performance, it doesn't give control over how these threads are created. Hence, you can't modify it according to your use case to get high performance for your needs.

UML Sequence Diagram

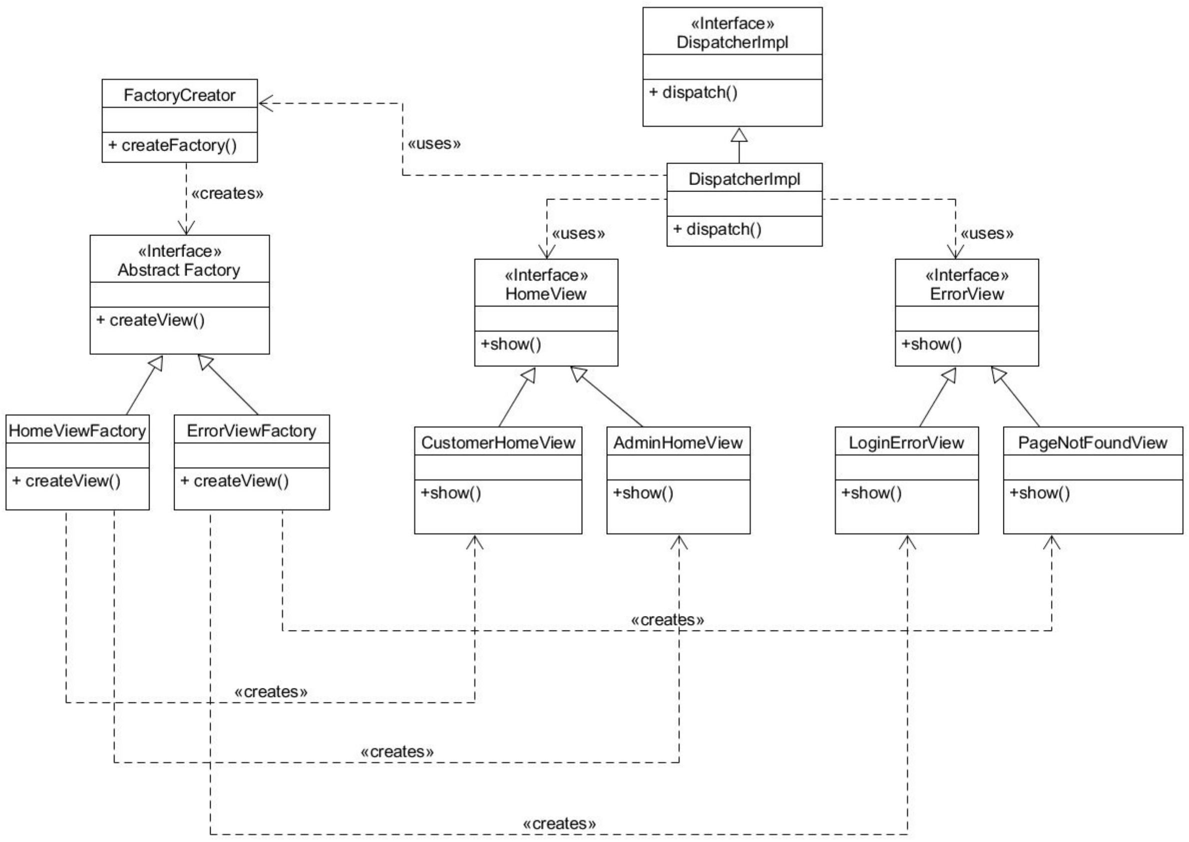


Other diagrams such as class diagram and domain model diagram are still valid.

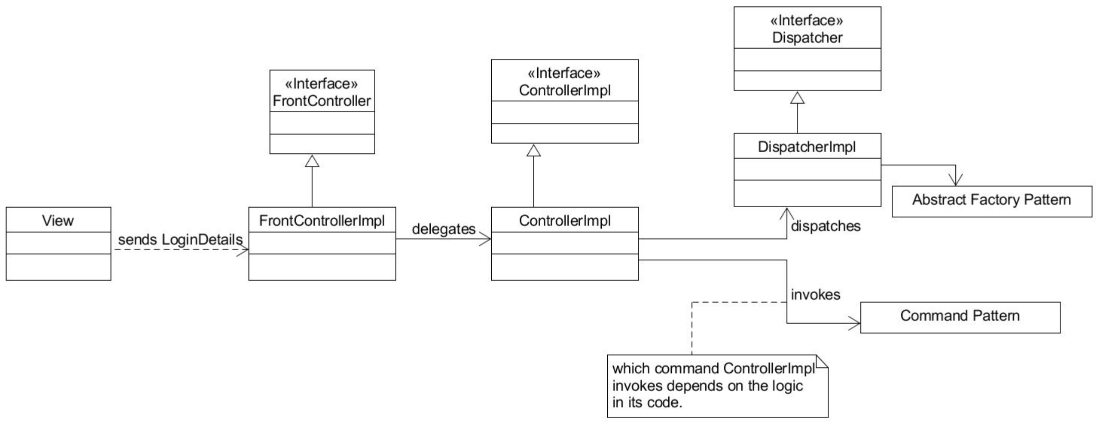
UML Class Diagram for Command Pattern



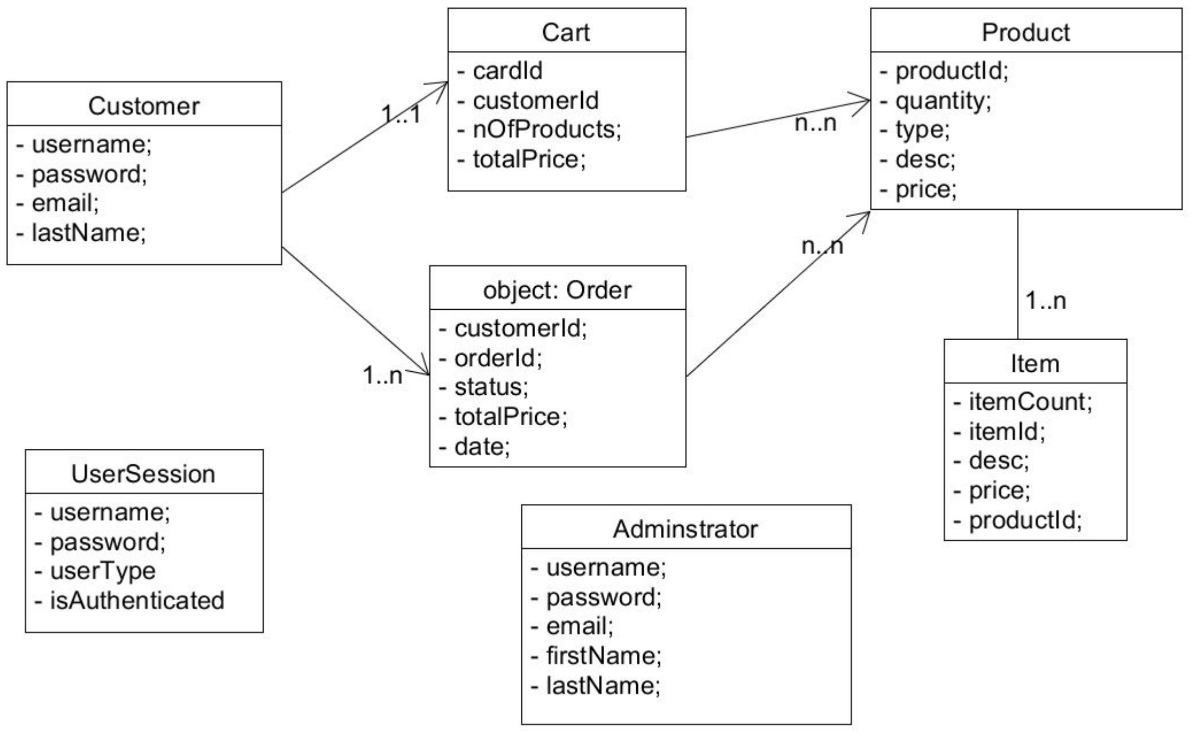
UML Diagram for Abstract Factory



UML Class Diagram for Front Controller



UML Domain Model (Updated)



Transition From Assignment 3 to Assignment 4

New Classes Created

ProductService class: The purpose of this class to have all the hardcoded products and other methods to carry out methods for the Products and Items. Later this layer, instead of being hardcoded will be used to connect to the persistence layer.

Item class: For example., Dell Latitude is a laptop which is product. And browsed the user seee this products but when he/she tires to add it to his cart, he is actually adding the item. The number of items decide the quantity of the product. Updated the Domain Model.

CartView class: This class will be called when view cart is requested by the user.

NewProductView class: This view is responsible for getting details of the new product the admin is adding to the Inventory.

Comments from Assignment 3

1. Visibility of Controller: added public access modifier to all methods

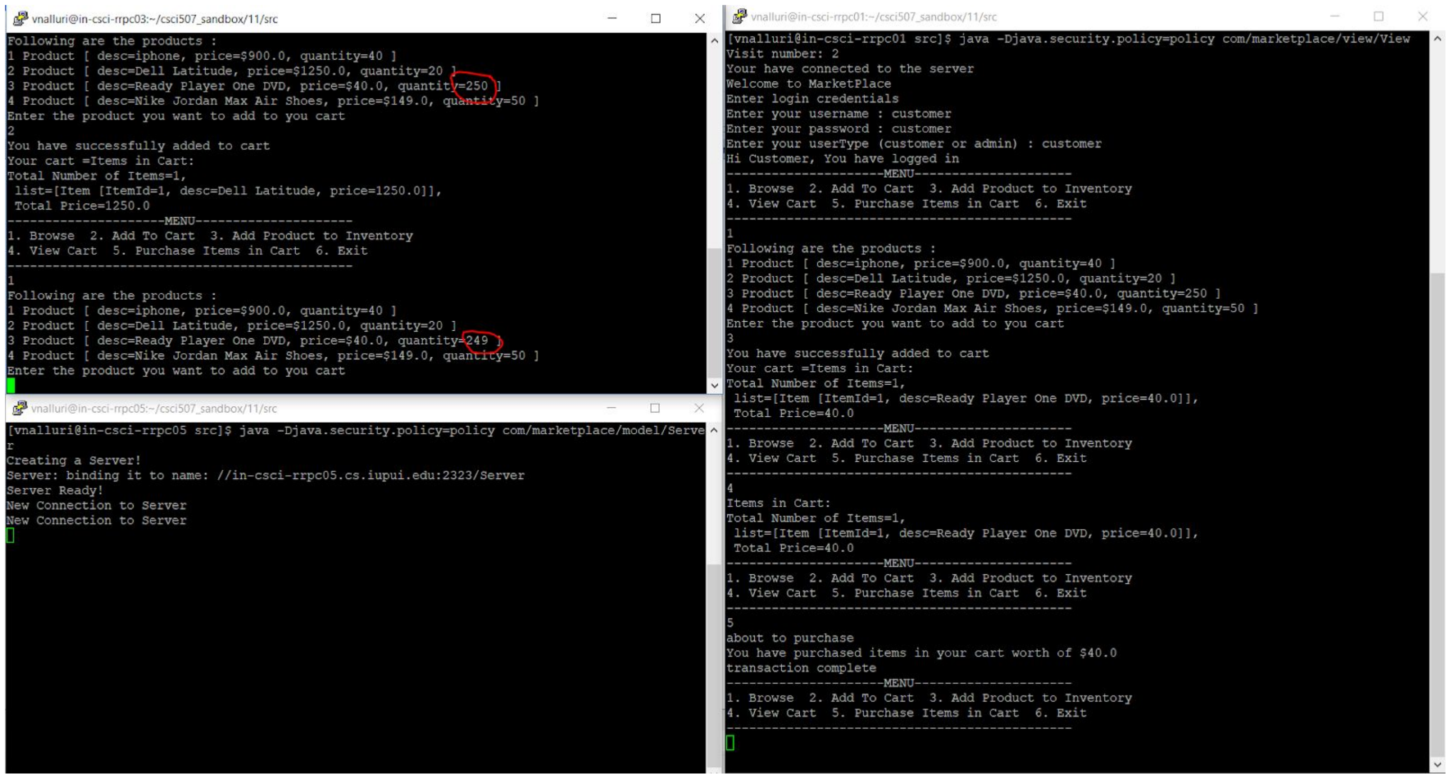
2. Visibility of UserSession variables: Encapsulated.

3. Do you really need everything in this package?

Addressed in the code.

Sample Runs

<u>Add Item To Cart, Purchase Items from Cart and Browse Items Implementation</u>



```
Following are the products :
1 Product [ desc=iphone, price=$900.0, quantity=40 ]
2 Product [ desc=Dell Latitude, price=$1250.0, quantity=20 ]
3 Product [ desc=Ready Player One DVD, price=$40.0, quantity=250 ]
4 Product [ desc=Nike Jordan Max Air Shoes, price=$149.0, quantity=50 ]
Enter the product you want to add to you cart
2
You have successfully added to cart
Your cart =Items in Cart:
Total Number of Items=1,
 list=[Item [ItemId=1, desc=Dell Latitude, price=1250.0]],
 Total Price=1250.0
--------------------MENU---------------------
1. Browse  2. Add To Cart  3. Add Product to Inventory
4. View Cart  5. Purchase Items in Cart  6. Exit
---------------------------------------------
1
Following are the products :
1 Product [ desc=iphone, price=$900.0, quantity=40 ]
2 Product [ desc=Dell Latitude, price=$1250.0, quantity=20 ]
3 Product [ desc=Ready Player One DVD, price=$40.0, quantity=249 ]
4 Product [ desc=Nike Jordan Max Air Shoes, price=$149.0, quantity=50 ]
Enter the product you want to add to you cart
```

```
[vnalluri@in-csci-rrpc05 src]$ java -Djava.security.policy=policy com/marketplace/model/Serve
r
Creating a Server!
Server: binding it to name: //in-csci-rrpc05.cs.iupui.edu:2323/Server
Server Ready!
New Connection to Server
New Connection to Server
```

```
[vnalluri@in-csci-rrpc01 src]$ java -Djava.security.policy=policy com/marketplace/view/View
Visit number: 2
Your have connected to the server
Welcome to MarketPlace
Enter login credentials
Enter your username : customer
Enter your password : customer
Enter your userType (customer or admin) : customer
Hi Customer, You have logged in
--------------------MENU---------------------
1. Browse  2. Add To Cart  3. Add Product to Inventory
4. View Cart  5. Purchase Items in Cart  6. Exit
---------------------------------------------
1
Following are the products :
1 Product [ desc=iphone, price=$900.0, quantity=40 ]
2 Product [ desc=Dell Latitude, price=$1250.0, quantity=20 ]
3 Product [ desc=Ready Player One DVD, price=$40.0, quantity=250 ]
4 Product [ desc=Nike Jordan Max Air Shoes, price=$149.0, quantity=50 ]
Enter the product you want to add to you cart
3
You have successfully added to cart
Your cart =Items in Cart:
Total Number of Items=1,
 list=[Item [ItemId=1, desc=Ready Player One DVD, price=40.0]],
 Total Price=40.0
--------------------MENU---------------------
1. Browse  2. Add To Cart  3. Add Product to Inventory
4. View Cart  5. Purchase Items in Cart  6. Exit
---------------------------------------------
4
Items in Cart:
Total Number of Items=1,
 list=[Item [ItemId=1, desc=Ready Player One DVD, price=40.0]],
 Total Price=40.0
--------------------MENU---------------------
1. Browse  2. Add To Cart  3. Add Product to Inventory
4. View Cart  5. Purchase Items in Cart  6. Exit
---------------------------------------------
5
about to purchase
You have purchased items in your cart worth of $40.0
transaction complete
--------------------MENU---------------------
1. Browse  2. Add To Cart  3. Add Product to Inventory
4. View Cart  5. Purchase Items in Cart  6. Exit
---------------------------------------------
```

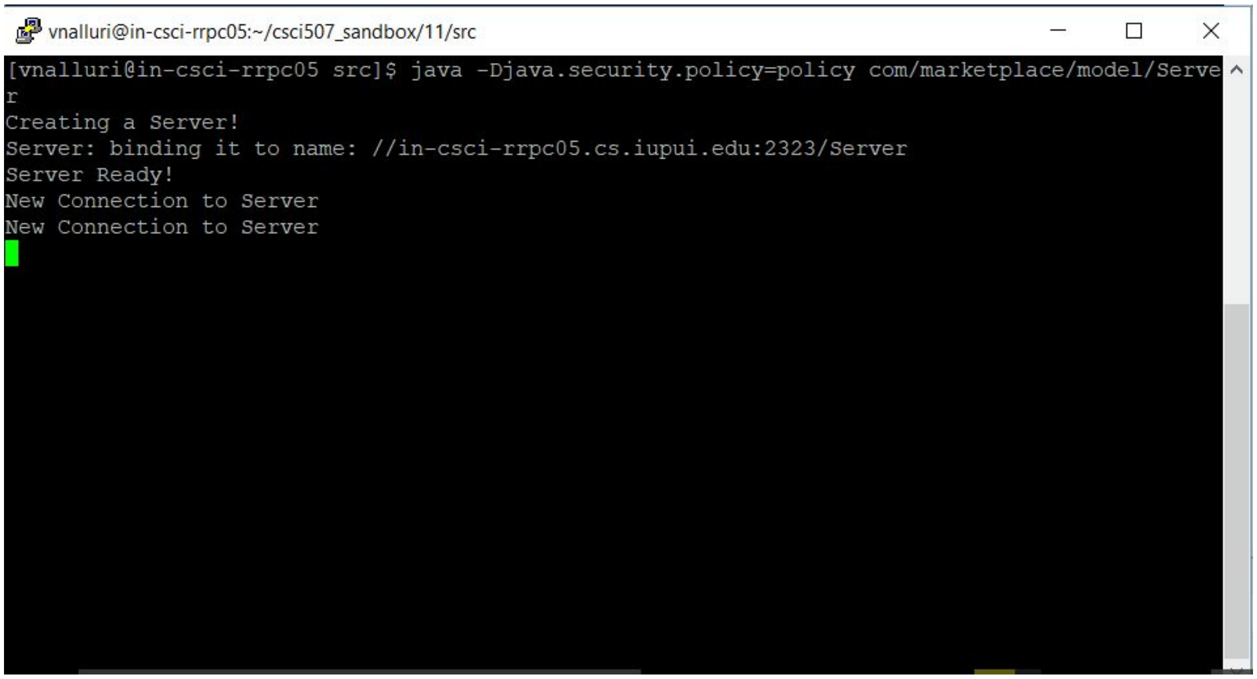
Refer to ScreenShot#1 in the ScreenShots Folder.

<u>How I achieved the above sample run</u>

1. Started the Server in one window (left-bottom window)
2. Opened another two windows and started two clients.

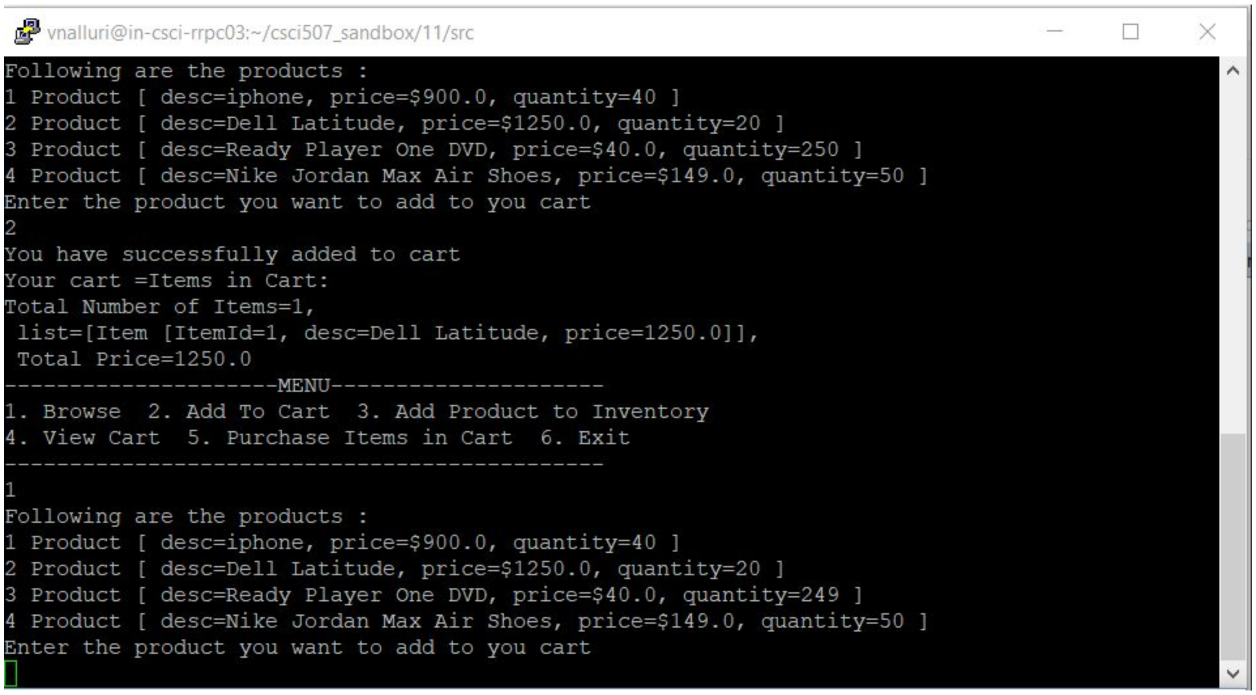
3. In one client window (top-left), logged in and selected “1”, Browse”. The red marking shows initial quantity of product 3. Then select any product to add to cart (1,2,3,4).
4. In another client window (right), logged in and “1” Browse, selected “3” Product — Ready Player One DVD” to add to cart. Then “4” Purchase Items in Cart”. Now the items are purchased. The quantity has been decreased.
5. To see the change in quantity, in the top-left window, select “1” Browse. The quantity of the product bought in other window (right) is reflected by the decrease in quantity.

Individual Pictures for higher resolution.



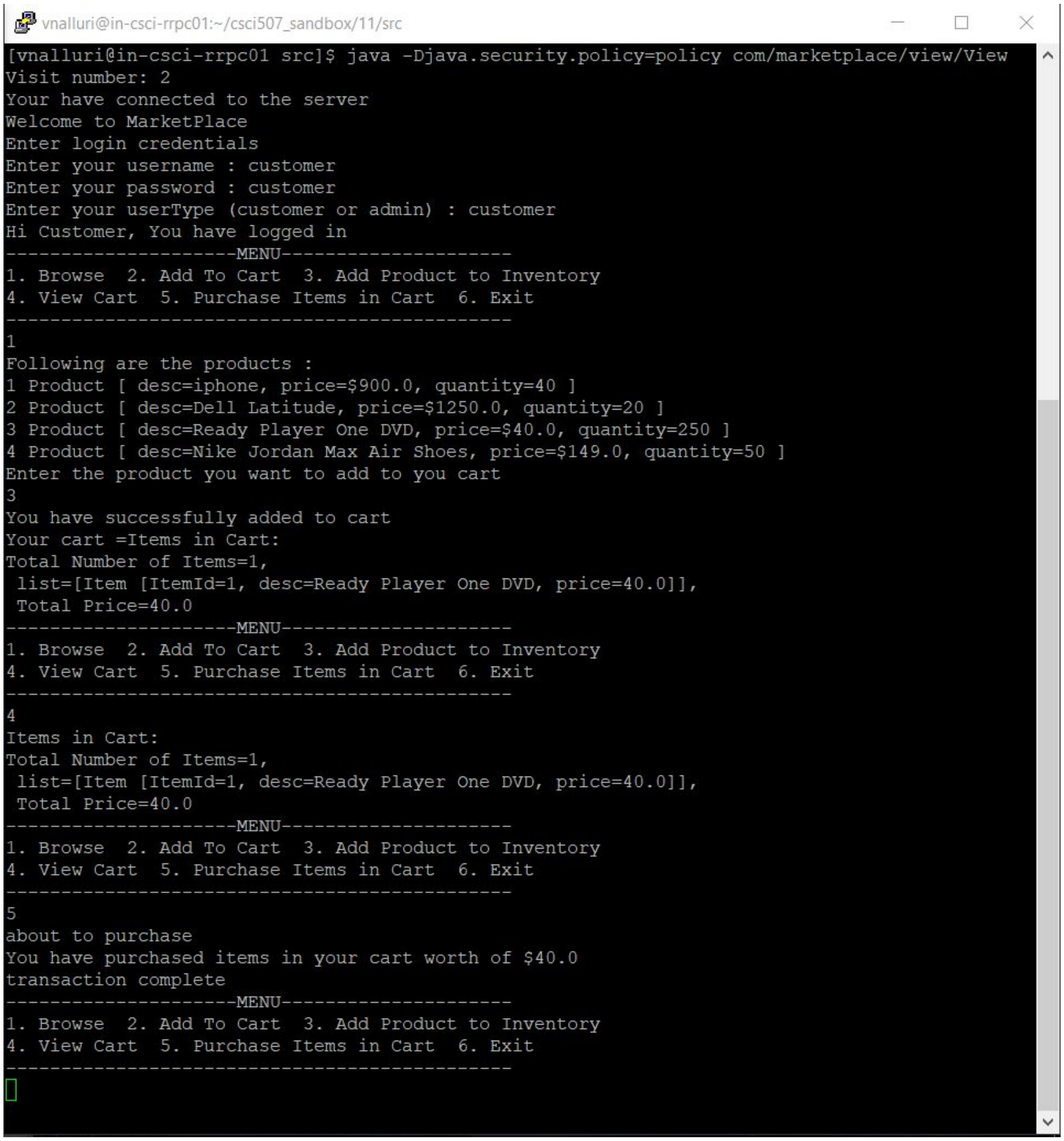
```
[vnalluri@in-csci-rrpc05 src]$ java -Djava.security.policy=policy com/marketplace/model/Serve
r
Creating a Server!
Server: binding it to name: //in-csci-rrpc05.cs.iupui.edu:2323/Server
Server Ready!
New Connection to Server
New Connection to Server
```

Server Window



```
Following are the products :
1 Product [ desc=iphone, price=$900.0, quantity=40 ]
2 Product [ desc=Dell Latitude, price=$1250.0, quantity=20 ]
3 Product [ desc=Ready Player One DVD, price=$40.0, quantity=250 ]
4 Product [ desc=Nike Jordan Max Air Shoes, price=$149.0, quantity=50 ]
Enter the product you want to add to you cart
2
You have successfully added to cart
Your cart =Items in Cart:
Total Number of Items=1,
 list=[Item [ItemId=1, desc=Dell Latitude, price=1250.0]],
 Total Price=1250.0
---------------------MENU--------------------
1. Browse  2. Add To Cart  3. Add Product to Inventory
4. View Cart  5. Purchase Items in Cart  6. Exit
----------------------------------------------
1
Following are the products :
1 Product [ desc=iphone, price=$900.0, quantity=40 ]
2 Product [ desc=Dell Latitude, price=$1250.0, quantity=20 ]
3 Product [ desc=Ready Player One DVD, price=$40.0, quantity=249 ]
4 Product [ desc=Nike Jordan Max Air Shoes, price=$149.0, quantity=50 ]
Enter the product you want to add to you cart
```

Client Window 1 (or top-left window in larger picture)

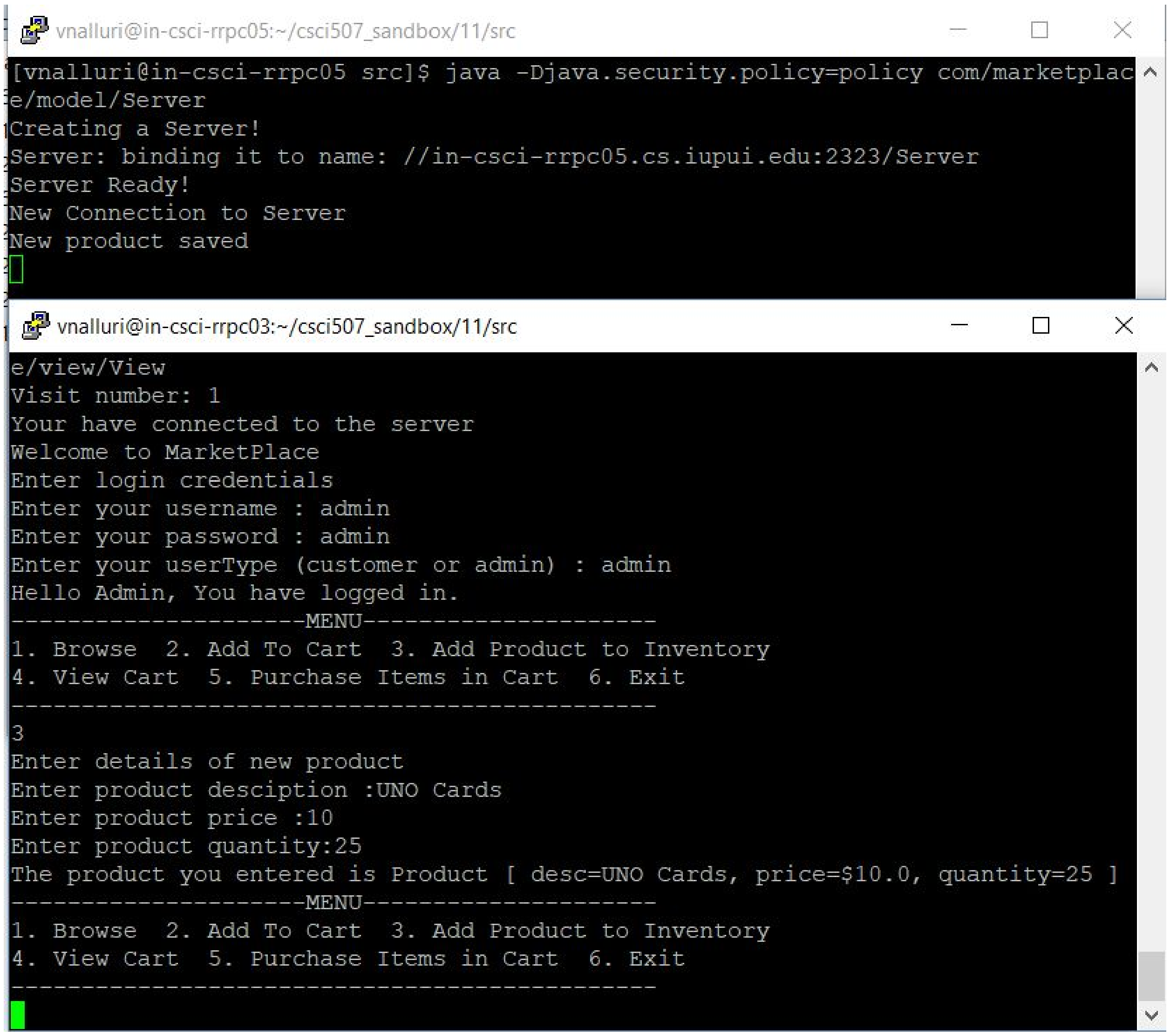


vnalluri@in-csci-rrpc01:~/csci507_sandbox/11/src

```
[vnalluri@in-csci-rrpc01 src]$ java -Djava.security.policy=policy com/marketplace/view/View
Visit number: 2
Your have connected to the server
Welcome to MarketPlace
Enter login credentials
Enter your username : customer
Enter your password : customer
Enter your userType (customer or admin) : customer
Hi Customer, You have logged in
---------------------MENU---------------------
1. Browse  2. Add To Cart  3. Add Product to Inventory
4. View Cart  5. Purchase Items in Cart  6. Exit
---------------------------------------------
1
Following are the products :
1 Product [ desc=iphone, price=$900.0, quantity=40 ]
2 Product [ desc=Dell Latitude, price=$1250.0, quantity=20 ]
3 Product [ desc=Ready Player One DVD, price=$40.0, quantity=250 ]
4 Product [ desc=Nike Jordan Max Air Shoes, price=$149.0, quantity=50 ]
Enter the product you want to add to you cart
3
You have successfully added to cart
Your cart =Items in Cart:
Total Number of Items=1,
 list=[Item [ItemId=1, desc=Ready Player One DVD, price=40.0]],
 Total Price=40.0
---------------------MENU---------------------
1. Browse  2. Add To Cart  3. Add Product to Inventory
4. View Cart  5. Purchase Items in Cart  6. Exit
---------------------------------------------
4
Items in Cart:
Total Number of Items=1,
 list=[Item [ItemId=1, desc=Ready Player One DVD, price=40.0]],
 Total Price=40.0
---------------------MENU---------------------
1. Browse  2. Add To Cart  3. Add Product to Inventory
4. View Cart  5. Purchase Items in Cart  6. Exit
---------------------------------------------
5
about to purchase
You have purchased items in your cart worth of $40.0
transaction complete
---------------------MENU---------------------
1. Browse  2. Add To Cart  3. Add Product to Inventory
4. View Cart  5. Purchase Items in Cart  6. Exit
---------------------------------------------
```

Client Window 2 (or right window in larger picture)

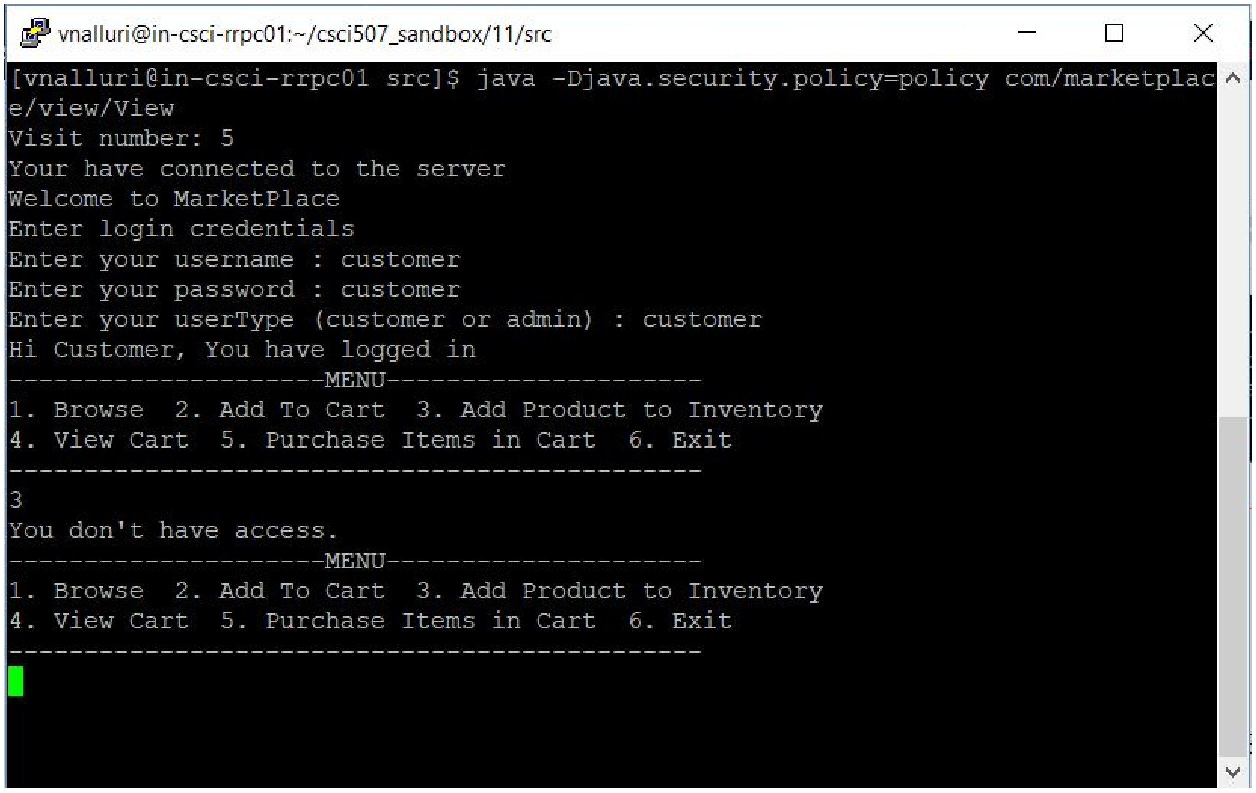
Implementation Of Add Product To Inventory



```
vnalluri@in-csci-rrpc05:~/csci507_sandbox/11/src
[vnalluri@in-csci-rrpc05 src]$ java -Djava.security.policy=policy com/marketplac
e/model/Server
Creating a Server!
Server: binding it to name: //in-csci-rrpc05.cs.iupui.edu:2323/Server
Server Ready!
New Connection to Server
New product saved
```

```
vnalluri@in-csci-rrpc03:~/csci507_sandbox/11/src
e/view/View
Visit number: 1
Your have connected to the server
Welcome to MarketPlace
Enter login credentials
Enter your username : admin
Enter your password : admin
Enter your userType (customer or admin) : admin
Hello Admin, You have logged in.
---------------------MENU---------------------
1. Browse  2. Add To Cart  3. Add Product to Inventory
4. View Cart  5. Purchase Items in Cart  6. Exit
---------------------------------------------
3
Enter details of new product
Enter product desciption :UNO Cards
Enter product price :10
Enter product quantity:25
The product you entered is Product [ desc=UNO Cards, price=$10.0, quantity=25 ]
---------------------MENU---------------------
1. Browse  2. Add To Cart  3. Add Product to Inventory
4. View Cart  5. Purchase Items in Cart  6. Exit
---------------------------------------------
```

Add Product to Inventory can only be done by Admin. If a customer uses this functionality, then,


vnalluri@in-csci-rrpc01:~/csci507_sandbox/11/src

```
[vnalluri@in-csci-rrpc01 src]$ java -Djava.security.policy=policy com/marketplac
e/view/View
Visit number: 5
Your have connected to the server
Welcome to MarketPlace
Enter login credentials
Enter your username : customer
Enter your password : customer
Enter your userType (customer or admin) : customer
Hi Customer, You have logged in
---------------------MENU---------------------
1. Browse  2. Add To Cart  3. Add Product to Inventory
4. View Cart  5. Purchase Items in Cart  6. Exit
----------------------------------------------
3
You don't have access.
---------------------MENU---------------------
1. Browse  2. Add To Cart  3. Add Product to Inventory
4. View Cart  5. Purchase Items in Cart  6. Exit
----------------------------------------------
```

Conclusion And Analysis

Discussion is done on concurrency in Java RMI, how it works, it's "guarantee", and consequences of concurrency in Java RMI. Also the sample run, of how the concurrency is maintained between two different clients is documented.

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