Online Market Place Application

Assignment #4: Report

Implementation of Add Items, Browse Items, Purchase Items Functionalities and Usage of Java RMI concurrency simulation for multiple clients

Course: CSCI 50700 - Object-Oriented Design and Programming

By
Mani Manjusha Kottala
IUPUI, Computer Information Science

Table of Contents

Introduction:

In previous assignment 1, an application is developed using (MVC) Model-View-Controller and Java RMI. In assignment 2, implemented the login functionality for user, admin and to use Front Controller pattern, Command pattern and Abstract Factory pattern. In assignment 3, three more patterns Authorization, Proxy and Reflection are used to integrate role-based access to the application. In this assignment, goal is to fully implement the Purchase Item, Add Item and Browse Item functionalities and usage of Java RMI usage for concurrency simulation in multiple clients.

Assignment 3 – Feedback:

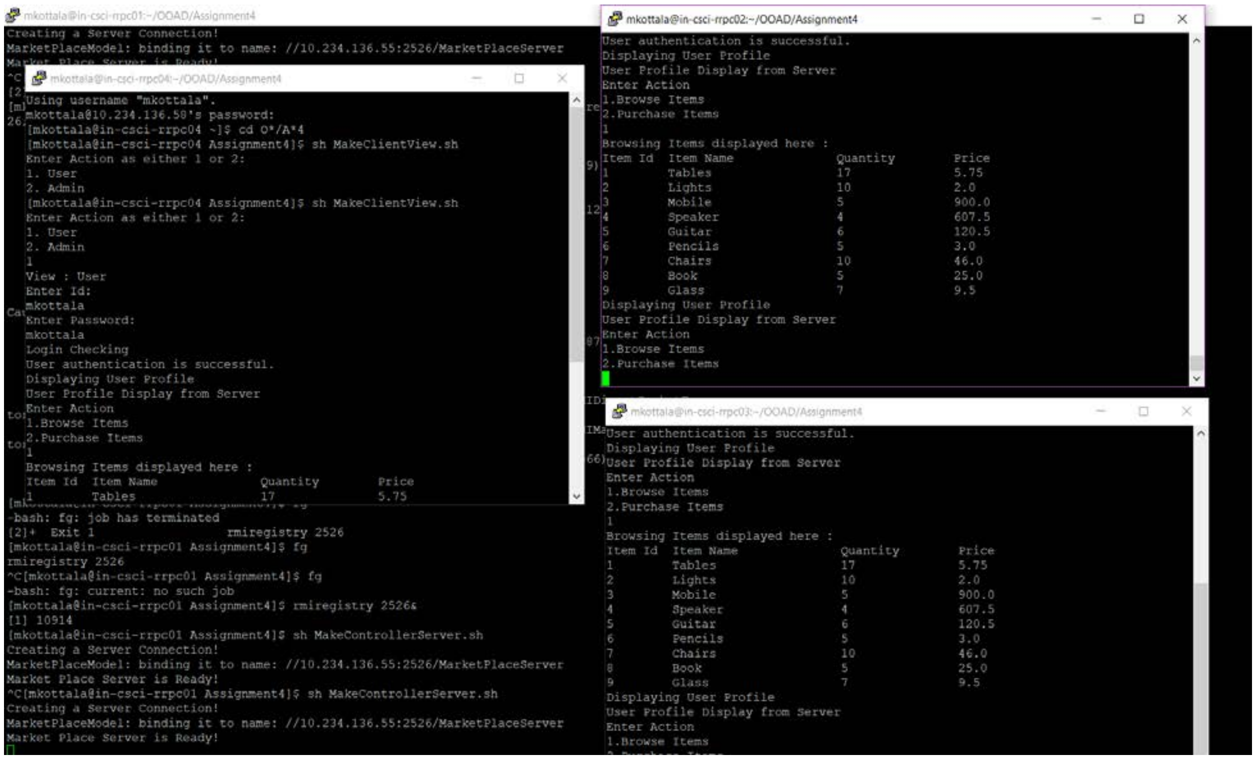
No feedback was mentioned in Assignment 3 comments branch.

Java RMI Concurrency Discussion:

Concurrency can be referred as the ability to perform different tasks or parts of a task in parallel. Multi-threaded programs are typically used for achieving concurrency in Java where different threads can handle different tasks in a program. In the Multi-threaded programming context, the main issue is when different threads try to perform the same action or try to use the same resource. Java provides locking mechanism to handle this situation where a lock is acquired by a thread on the shared resource and when another thread wants to access the resource, it must wait till the lock is released.

Thread usage in Java RMI can be discussed both in client side and server side. In client side, a request is marshalled and send to through remote invocation[1]. The requested thread waits till it gets the response back from the server, So the calling thread is in block state until it receives the response back from the server. So, when multiple clients try to access the remote objects, the client requester threads will be in blocked state until they receive a response back from the server. But in the server side as this statement states – *"A method dispatched by the RMI runtime to a remote object implementation may or may not execute in a separate thread" [3]*, if concurrent calls are made from same client then there is no guarantee that they might execute on different threads [4]. The concurrent calls from different clients run on different threads. But for this concurrent access Java RMI doesn't provide any thread safety, so synchronized keyword can be used to make the Java RMI thread safe. The synchronization should be provided by examining the behavior of the objects to avoid unnecessary bottlenecks in concurrency or deadlocks.

Java RMI in Market Place Application: In the current application, the Java RMI multi-threading is used. The functions for browse items, add items and purchase items can be accessed by multiple clients at the same time, for making the application thread safe these function use synchronizations. The application uses machine **- 10.234.136.55** as the server as it has the database connection. By using the synchronized keywords, the access of the same function is limited to a single thread at any point of time. As Java RMI provides currency using multi-threading the application can run on two clients at the same time. For example, I have clients **10.234.136.56**, **10.234.136.57** and **10.234.136.58** accessing the same browse functionality. These three clients can access the same method concurrently as shown in the screenshots belows:



```
mkottala@in-csci-rrpc01:~/OOAD/Assignment4
Creating a Server Connection!
MarketPlaceModel: binding it to name: //10.234.136.55:2526/MarketPlaceServer
-bash: fg: job has terminated
[2]+  Exit 1                  rmiregistry 2526
[mkottala@in-csci-rrpc01 Assignment4]$ fg
rmiregistry 2526
^C[mkottala@in-csci-rrpc01 Assignment4]$ fg
-bash: fg: current: no such job
[mkottala@in-csci-rrpc01 Assignment4]$ rmiregistry 2526&
[1] 10914
[mkottala@in-csci-rrpc01 Assignment4]$ sh MakeControllerServer.sh
Creating a Server Connection!
MarketPlaceModel: binding it to name: //10.234.136.55:2526/MarketPlaceServer
Market Place Server is Ready!
^C[mkottala@in-csci-rrpc01 Assignment4]$ sh MakeControllerServer.sh
Creating a Server Connection!
MarketPlaceModel: binding it to name: //10.234.136.55:2526/MarketPlaceServer
Market Place Server is Ready!
```

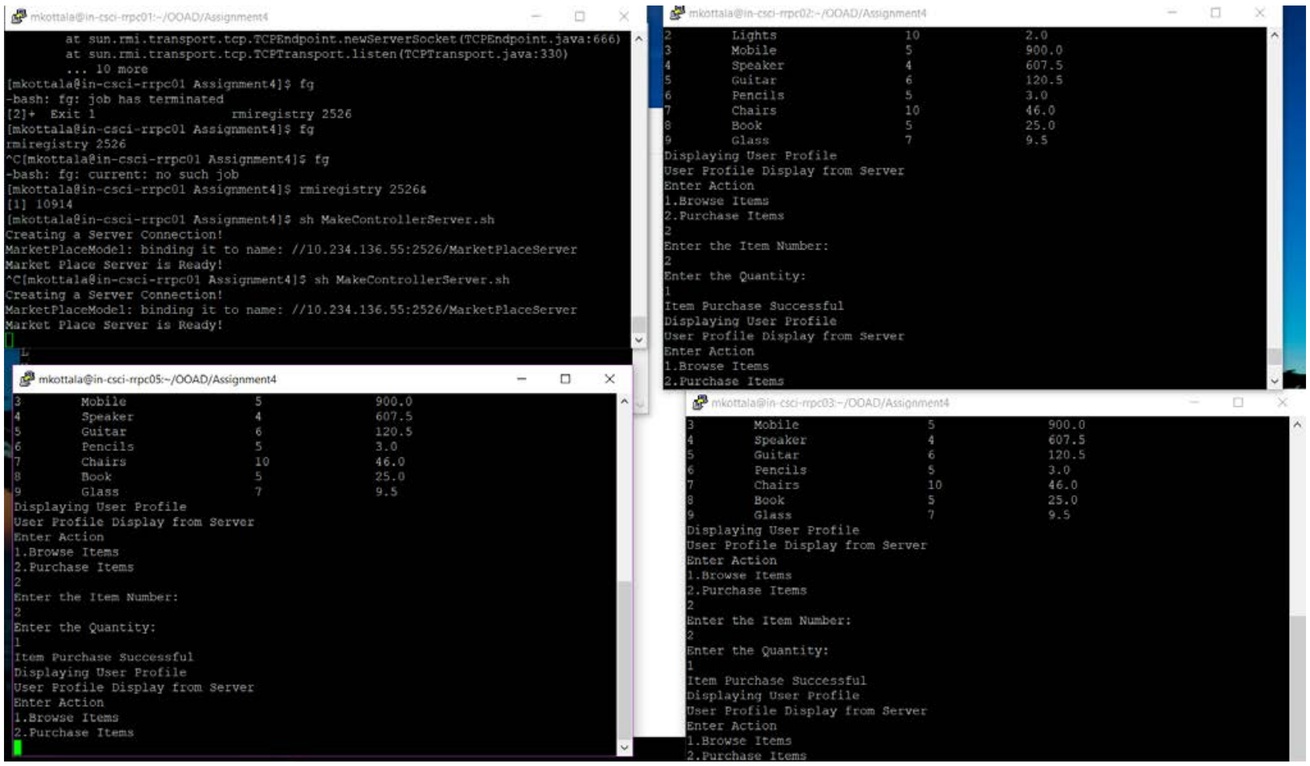
```
mkottala@in-csci-rrpc04:~/OOAD/Assignment4
Using username "mkottala".
mkottala@10.234.136.58's password:
[mkottala@in-csci-rrpc04 ~]$ cd O*/A*4
[mkottala@in-csci-rrpc04 Assignment4]$ sh MakeClientView.sh
Enter Action as either 1 or 2:
1. User
2. Admin
[mkottala@in-csci-rrpc04 Assignment4]$ sh MakeClientView.sh
Enter Action as either 1 or 2:
1. User
2. Admin
1
View : User
Enter Id:
mkottala
Enter Password:
mkottala
Login Checking
User authentication is successful.
Displaying User Profile
User Profile Display from Server
Enter Action
1.Browse Items
2.Purchase Items
1
Browsing Items displayed here :
Item Id  Item Name              Quantity        Price
1        Tables                 17              5.75
```

```
mkottala@in-csci-rrpc02:~/OOAD/Assignment4
User authentication is successful.
Displaying User Profile
User Profile Display from Server
Enter Action
1.Browse Items
2.Purchase Items
1
Browsing Items displayed here :
Item Id  Item Name              Quantity        Price
1        Tables                 17              5.75
2        Lights                 10              2.0
3        Mobile                 5               900.0
4        Speaker                4               607.5
5        Guitar                 6               120.5
6        Pencils                5               3.0
7        Chairs                 10              46.0
8        Book                   5               25.0
9        Glass                  7               9.5
Displaying User Profile
User Profile Display from Server
Enter Action
1.Browse Items
2.Purchase Items
```

```
mkottala@in-csci-rrpc03:~/OOAD/Assignment4
User authentication is successful.
Displaying User Profile
User Profile Display from Server
Enter Action
1.Browse Items
2.Purchase Items
1
Browsing Items displayed here :
Item Id  Item Name              Quantity        Price
1        Tables                 17              5.75
2        Lights                 10              2.0
3        Mobile                 5               900.0
4        Speaker                4               607.5
5        Guitar                 6               120.5
6        Pencils                5               3.0
7        Chairs                 10              46.0
8        Book                   5               25.0
9        Glass                  7               9.5
Displaying User Profile
User Profile Display from Server
Enter Action
1.Browse Items
```

In absence of synchronized keyword, the function accessed can be concurrently but there may be cases where inconsistencies and errors may occur. If we consider the add purchase item and if two clients are trying to purchase the same item at the same time, there can be inconsistencies in update to database as both the clients will try to update the same item's quantity. Here synchronized keyword helps to maintain consistence in the application while concurrent access. I have tried accessing the RMI without the synchronized key word for browse items as shown above and it shows that the Java RMI is multi-thread and allows concurrency.

As per the requirements the application has to run on five clients, I have tried running the application on all the available six machines **- 10.234.136.55** as server and other five machines as clients. We can see in the below screenshots that all the clients were able to communicate with server simultaneously.

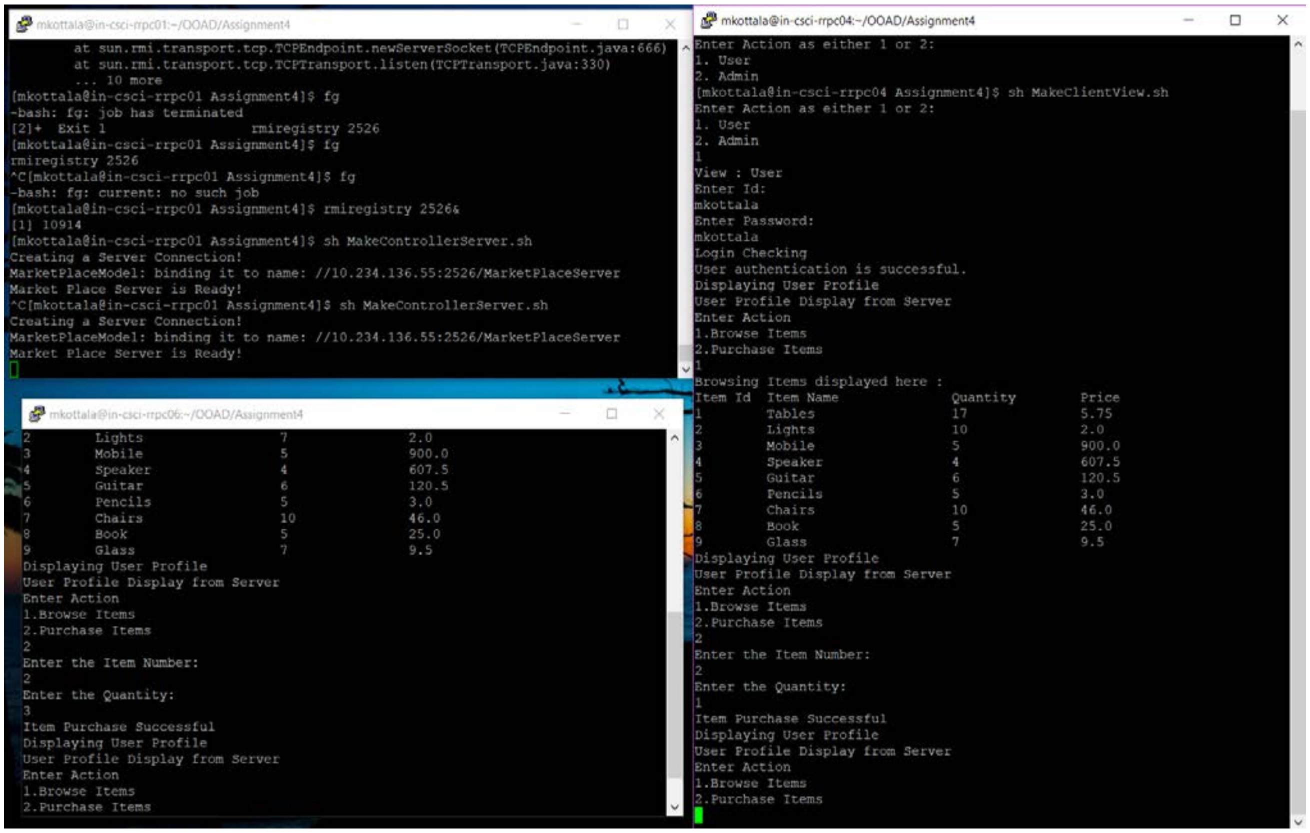

mkottala@in-csci-rrpc01:~/OOAD/Assignment4
mkottala@in-csci-rrpc05:~/OOAD/Assignment4
mkottala@in-csci-rrpc02:~/OOAD/Assignment4
mkottala@in-csci-rrpc03:~/OOAD/Assignment4

```
        at sun.rmi.transport.tcp.TCPEndpoint.newServerSocket(TCPEndpoint.java:666)
        at sun.rmi.transport.tcp.TCPTransport.listen(TCPTransport.java:330)
        ... 10 more
[mkottala@in-csci-rrpc01 Assignment4]$ fg
-bash: fg: job has terminated
[2]+  Exit 1                  rmiregistry 2526
[mkottala@in-csci-rrpc01 Assignment4]$ fg
rmiregistry 2526
^C[mkottala@in-csci-rrpc01 Assignment4]$ fg
-bash: fg: current: no such job
[mkottala@in-csci-rrpc01 Assignment4]$ rmiregistry 2526&
[1] 10914
[mkottala@in-csci-rrpc01 Assignment4]$ sh MakeControllerServer.sh
Creating a Server Connection!
MarketPlaceModel: binding it to name: //10.234.136.55:2526/MarketPlaceServer
Market Place Server is Ready!
^C[mkottala@in-csci-rrpc01 Assignment4]$ sh MakeControllerServer.sh
Creating a Server Connection!
MarketPlaceModel: binding it to name: //10.234.136.55:2526/MarketPlaceServer
Market Place Server is Ready!
```

```
3        Mobile              5                900.0
4        Speaker             4                607.5
5        Guitar              6                120.5
6        Pencils             5                3.0
7        Chairs              10               46.0
8        Book                5                25.0
9        Glass               7                9.5
Displaying User Profile
User Profile Display from Server
Enter Action
1.Browse Items
2.Purchase Items
2
Enter the Item Number:
2
Enter the Quantity:
1
Item Purchase Successful
Displaying User Profile
User Profile Display from Server
Enter Action
1.Browse Items
2.Purchase Items
```

```
2        Lights              10               2.0
3        Mobile              5                900.0
4        Speaker             4                607.5
5        Guitar              6                120.5
6        Pencils             5                3.0
7        Chairs              10               46.0
8        Book                5                25.0
9        Glass               7                9.5
Displaying User Profile
User Profile Display from Server
Enter Action
1.Browse Items
2.Purchase Items
2
Enter the Item Number:
2
Enter the Quantity:
1
Item Purchase Successful
Displaying User Profile
User Profile Display from Server
Enter Action
1.Browse Items
2.Purchase Items
```

```
3        Mobile              5                900.0
4        Speaker             4                607.5
5        Guitar              6                120.5
6        Pencils             5                3.0
7        Chairs              10               46.0
8        Book                5                25.0
9        Glass               7                9.5
Displaying User Profile
User Profile Display from Server
Enter Action
1.Browse Items
2.Purchase Items
2
Enter the Item Number:
2
Enter the Quantity:
1
Item Purchase Successful
Displaying User Profile
User Profile Display from Server
Enter Action
1.Browse Items
2.Purchase Items
```


mkottala@in-csci-rrpc01:~/OOAD/Assignment4
mkottala@in-csci-rrpc06:~/OOAD/Assignment4
mkottala@in-csci-rrpc04:~/OOAD/Assignment4

```
        at sun.rmi.transport.tcp.TCPEndpoint.newServerSocket(TCPEndpoint.java:666)
        at sun.rmi.transport.tcp.TCPTransport.listen(TCPTransport.java:330)
        ... 10 more
[mkottala@in-csci-rrpc01 Assignment4]$ fg
-bash: fg: job has terminated
[2]+  Exit 1                  rmiregistry 2526
[mkottala@in-csci-rrpc01 Assignment4]$ fg
rmiregistry 2526
^C[mkottala@in-csci-rrpc01 Assignment4]$ fg
-bash: fg: current: no such job
[mkottala@in-csci-rrpc01 Assignment4]$ rmiregistry 2526&
[1] 10914
[mkottala@in-csci-rrpc01 Assignment4]$ sh MakeControllerServer.sh
Creating a Server Connection!
MarketPlaceModel: binding it to name: //10.234.136.55:2526/MarketPlaceServer
Market Place Server is Ready!
^C[mkottala@in-csci-rrpc01 Assignment4]$ sh MakeControllerServer.sh
Creating a Server Connection!
MarketPlaceModel: binding it to name: //10.234.136.55:2526/MarketPlaceServer
Market Place Server is Ready!
```

```
2        Lights              7                2.0
3        Mobile              5                900.0
4        Speaker             4                607.5
5        Guitar              6                120.5
6        Pencils             5                3.0
7        Chairs              10               46.0
8        Book                5                25.0
9        Glass               7                9.5
Displaying User Profile
User Profile Display from Server
Enter Action
1.Browse Items
2.Purchase Items
2
Enter the Item Number:
2
Enter the Quantity:
3
Item Purchase Successful
Displaying User Profile
User Profile Display from Server
Enter Action
1.Browse Items
2.Purchase Items
```

```
Enter Action as either 1 or 2:
1. User
2. Admin
[mkottala@in-csci-rrpc04 Assignment4]$ sh MakeClientView.sh
Enter Action as either 1 or 2:
1. User
2. Admin
1
View : User
Enter Id:
mkottala
Enter Password:
mkottala
Login Checking
User authentication is successful.
Displaying User Profile
User Profile Display from Server
Enter Action
1.Browse Items
2.Purchase Items
1
Browsing Items displayed here :
Item Id  Item Name           Quantity         Price
1        Tables              17               5.75
2        Lights              10               2.0
3        Mobile              5                900.0
4        Speaker             4                607.5
5        Guitar              6                120.5
6        Pencils             5                3.0
7        Chairs              10               46.0
8        Book                5                25.0
9        Glass               7                9.5
Displaying User Profile
User Profile Display from Server
Enter Action
1.Browse Items
2.Purchase Items
2
Enter the Item Number:
2
Enter the Quantity:
1
Item Purchase Successful
Displaying User Profile
User Profile Display from Server
Enter Action
1.Browse Items
2.Purchase Items
```

To know the ideal behavior of the synchronized keyword, the functions should be accessed exactely at the same time from two different machines. So, the Java RMI provides multi-threading with concurrency and the application can be made thread safe using the synchronized keyword. But the usage of synchronized should be determined on the behavior of the functions on which it is applied to avoid any unnecessary deadlocks.

Database Connectivity:

For this assignment, I have used MYSQL database as the persistent storage for the Items information. The steps required for establishing the connection and accessing database are [6]:

- Importing the JDBC package using *'import java.sql.*;'* in DBConnection class for accessing JDBC APIs in the class.
- A connect() function is implemented in the DBConnection class. In this method, Class.forName() is used to register and load the JDBC driver. JDBC driver is *'com.mysql.jdbc.Driver'*.
- In the connect() method, getConnection() method with three parameters i.e., url, username and password is used for connecting to the database and getting the Connection object. url is the database url to the mkottala_db database – *'jdbc:mysql://localhost:3306/mkottala_db'* .
- Once the connection is established, the Queries can be executed using the Statement Object. A statement is created using connection.createStatement() and this statement can be used to execute the queries using these methods – executeQuery() which returns a ResultSet object or executeUpdate() used for updation of database.
- ResultSet contains the result from the Query. They can be accessed using the functions next(), getInt(), getString() etc.
- Close() function is used to close the Statement, ResultSet and Connection objects.

Functionalities Implementation:

One of the requirements in this assignment is complete implementation of the following functions of Admin and User:

- **Browse Items**: Both Admin and User should have the browse functionality to view the items in the database. This function is used for the display of items and other details by getting the data from Items table in the database. In admin and user, browseItems() method is called remotely which is implemented in the MarketPlaceModel class. This method uses the database connection and executes the query to get the result from the database. Statement.executeQuery() method is used for executing the select query and results are returned as ReseultSet object. The resultant rows in the ResultSet object are converted to String and added to a ArrayList<String>. This ArrayList is returned to the User and Admin views. Proper formatting is used for displaying the result to users and Admins.
- **Add Items**: This method can only be accessed by Admin which is achieved by role-based access. Admin can use this to add new items to the database. A remote method addItems() implemented in MarketPlaceModel class is called by the Admin view. This method takes a String array as parameter for the function and this String array elements are used for creating the insert query. The insert query is executed using the statement.executeUpdate() method to insert the data into database. If the insertion is successful, then a success message is returned or else an error is returned as a String to the Admin View.
- **Purchase Items**: This method can only be accessed by User using role-based access. For purchasing an item, the user should enter item id and quantity. The purchase() function of

MarketPlaceModel class is remotely called with the item id and quantity as parameters, this function returns a string “Item Purchase Successful” on success or “Out of Stock or enter an existing item Id" if item is out of stock or item doesn’t exist. A select query is used to get the available quantity of the item from the database. If the available quantity is greater than or equal to purchase quantity then the item can be purchased, else the item is out of stock. An update query is executed using statement.executeUpdate() for updating the quantity of the item.

New Classes and Interfaces created:

- **DBConnection**: DBConnection class has the functions for managing the connection to the database. This class has implementation for methods connect() and disconnect() for managing the database connection.
- **UserActions**: UserActions is a command interface for the User Actions browse and purchase.This is used by the Invoker class for creating a list of command and invoke them based on the type of object.
- **BrowseUserItems**: This is a concrete class for browse items function which implements the UserActions interface. It implements the execute() method which calls the user browse function.
- **Purchase**: This is a concrete class for purchase items function which implements the UserActions interface. It implements the execute() method which calls the user purchase function.

Class Diagram [6]:

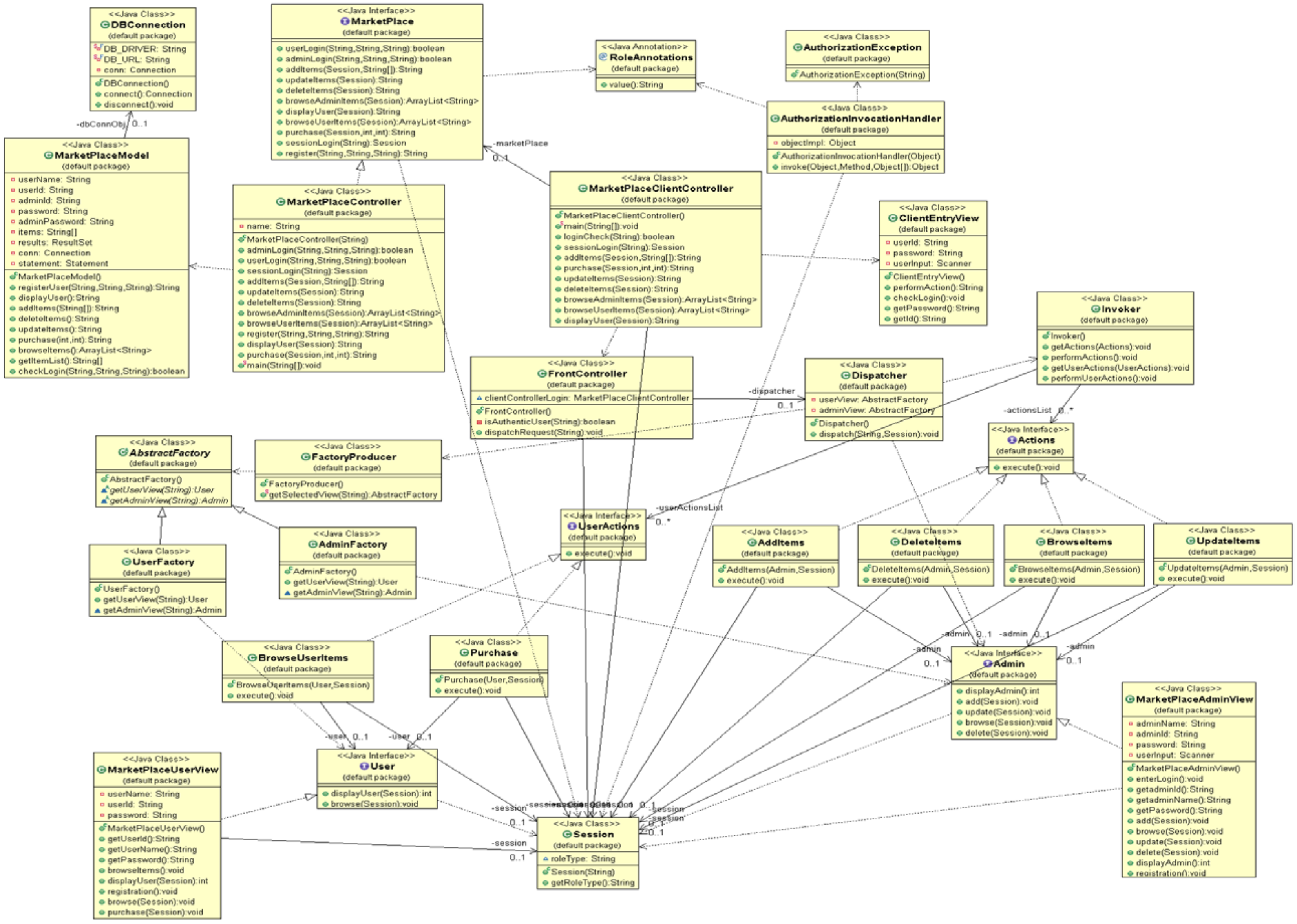
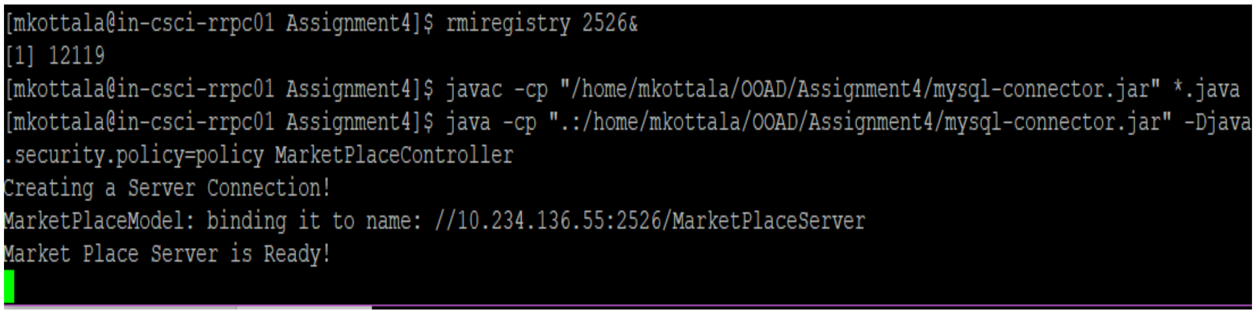


Fig 2: Class Diagram for MarketPlace Application

Sample runs:

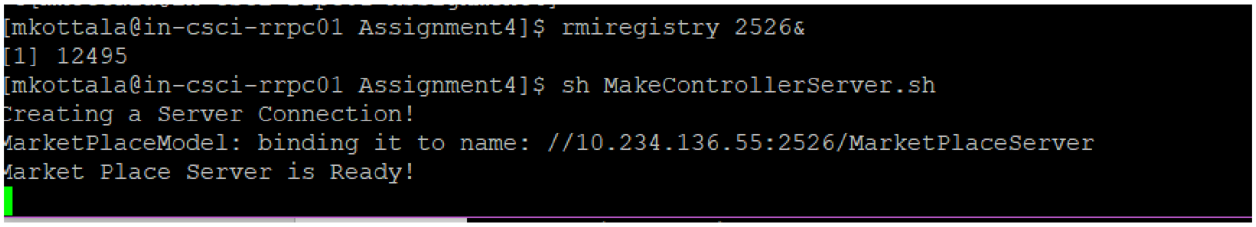
1. Starting the RMI registry and Running the server application on Machine **- 10.234.136.55**



```
[mkottala@in-csci-rrpc01 Assignment4]$ rmiregistry 2526&
[1] 12119
[mkottala@in-csci-rrpc01 Assignment4]$ javac -cp "/home/mkottala/OOAD/Assignment4/mysql-connector.jar" *.java
[mkottala@in-csci-rrpc01 Assignment4]$ java -cp ".:/home/mkottala/OOAD/Assignment4/mysql-connector.jar" -Djava
.security.policy=policy MarketPlaceController
Creating a Server Connection!
MarketPlaceModel: binding it to name: //10.234.136.55:2526/MarketPlaceServer
Market Place Server is Ready!
```

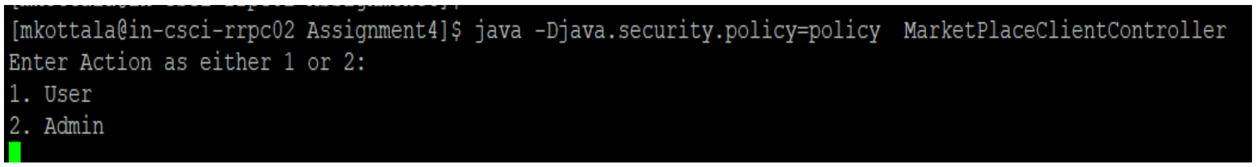
Or

Using makefile: MakeControllerServer.sh



```
[mkottala@in-csci-rrpc01 Assignment4]$ rmiregistry 2526&
[1] 12495
[mkottala@in-csci-rrpc01 Assignment4]$ sh MakeControllerServer.sh
Creating a Server Connection!
MarketPlaceModel: binding it to name: //10.234.136.55:2526/MarketPlaceServer
Market Place Server is Ready!
```

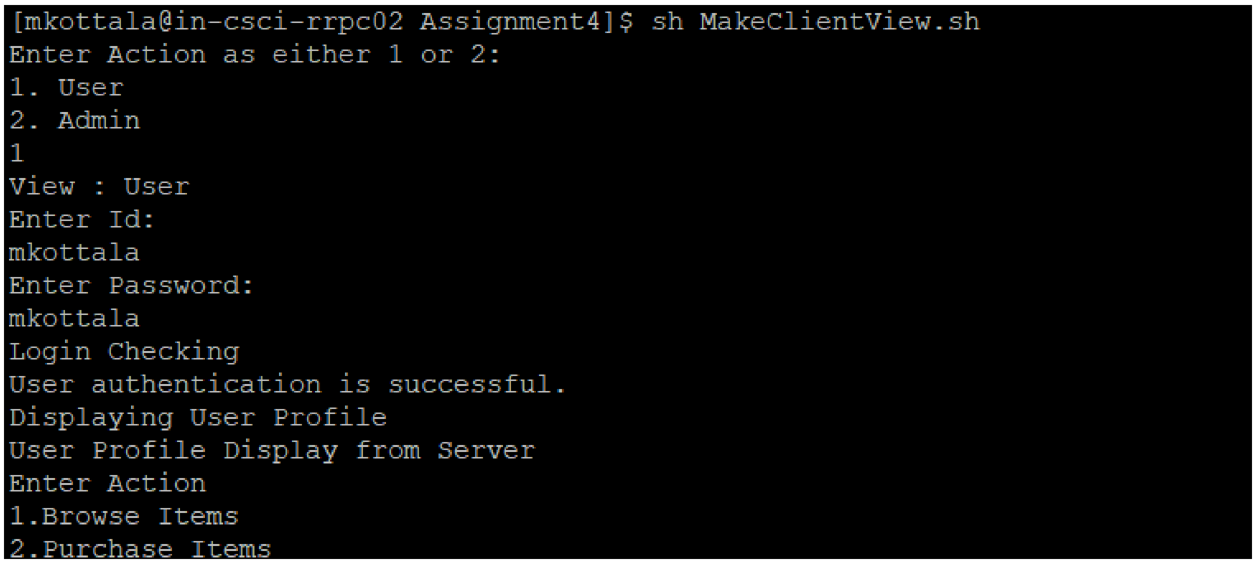
2. Running the Client application on any of the machines - 10.234.136.56, 10.234.136.57, 10.234.136.58, 10.234.136.59 and 10.234.136.60



```
[mkottala@in-csci-rrpc02 Assignment4]$ java -Djava.security.policy=policy  MarketPlaceClientController
Enter Action as either 1 or 2:
1. User
2. Admin
```

Or

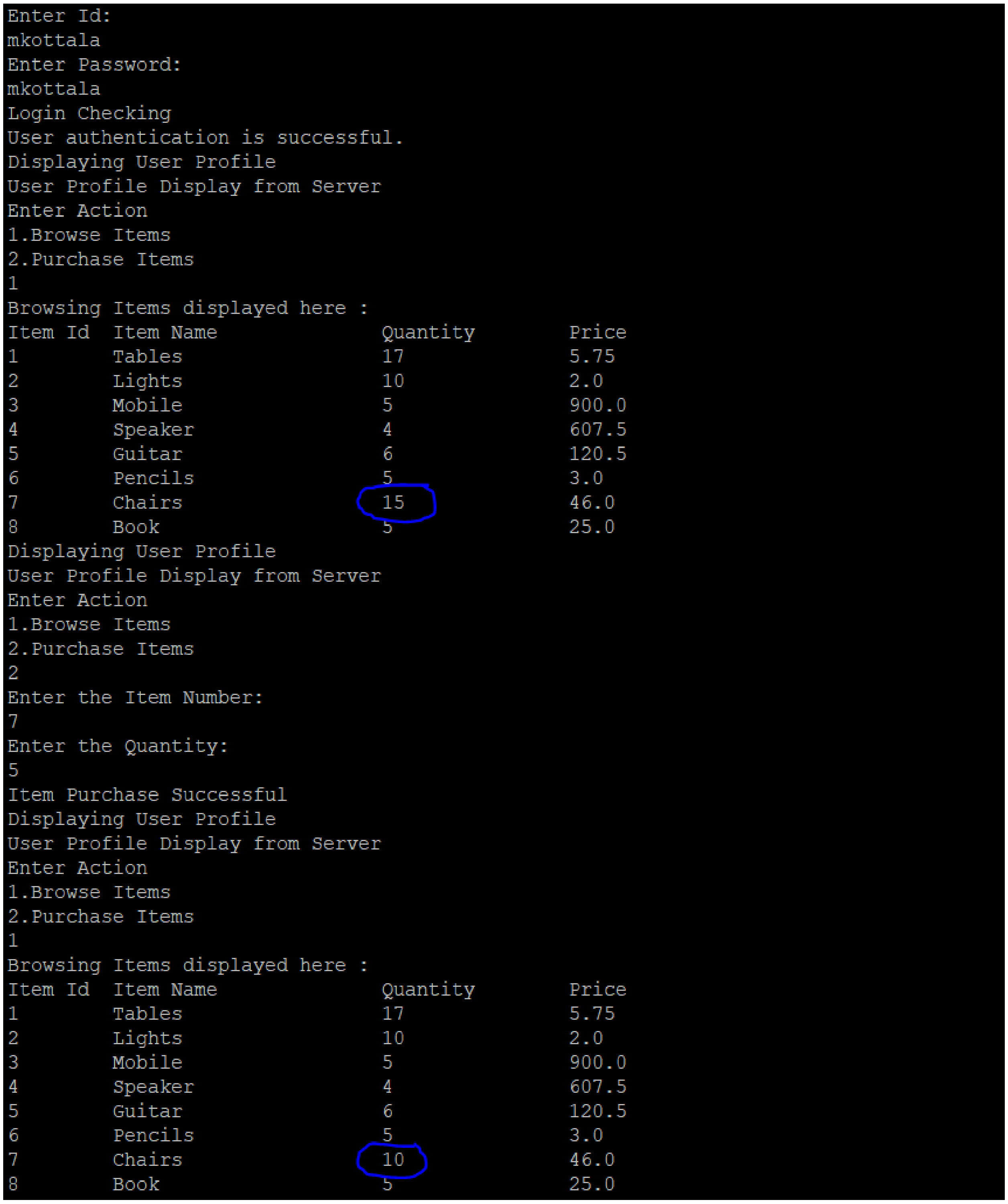
Using MakeFile: MakeClientView.sh



```
[mkottala@in-csci-rrpc02 Assignment4]$ sh MakeClientView.sh
Enter Action as either 1 or 2:
1. User
2. Admin
1
View : User
Enter Id:
mkottala
Enter Password:
mkottala
Login Checking
User authentication is successful.
Displaying User Profile
User Profile Display from Server
Enter Action
1.Browse Items
2.Purchase Items
```

3. User Functions:

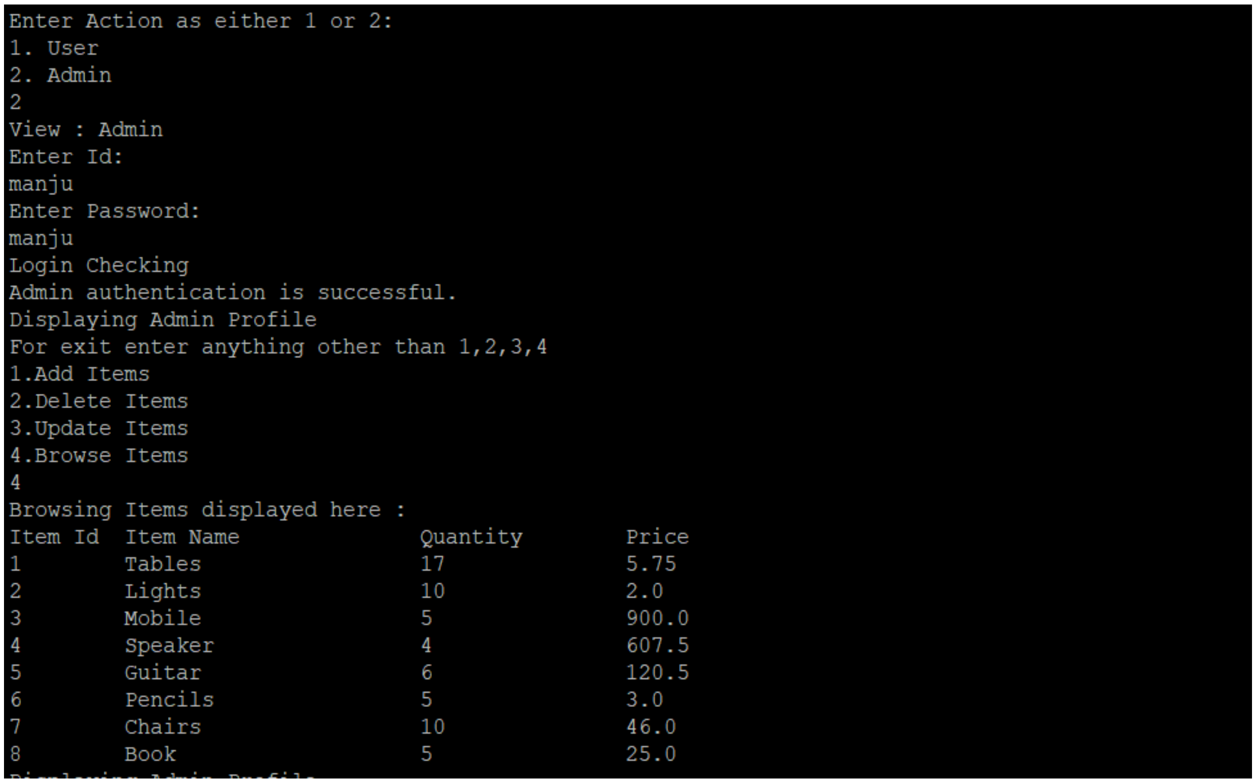
User can Browse the items and purchase the items. Below are the screenshots for user functionalities:



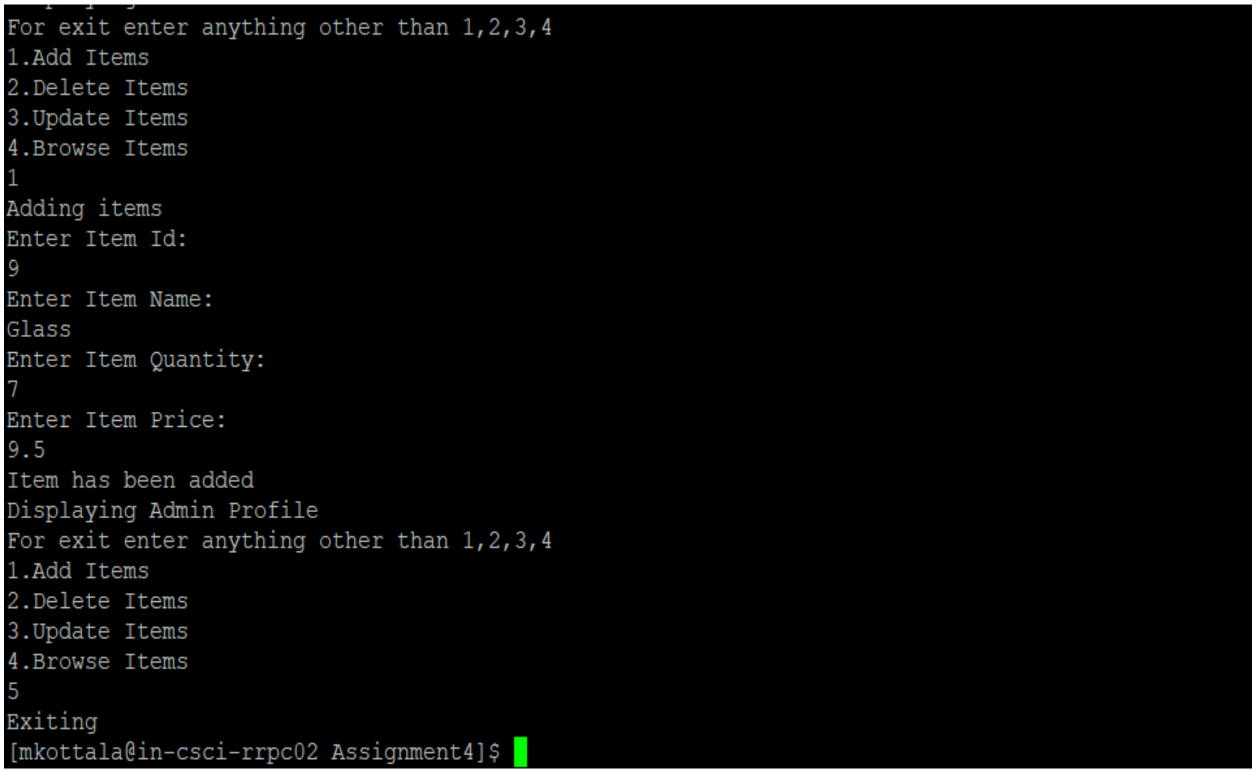
```
Enter Id:
mkottala
Enter Password:
mkottala
Login Checking
User authentication is successful.
Displaying User Profile
User Profile Display from Server
Enter Action
1.Browse Items
2.Purchase Items
1
Browsing Items displayed here :
Item Id  Item Name              Quantity        Price
1        Tables                 17              5.75
2        Lights                 10              2.0
3        Mobile                 5               900.0
4        Speaker                4               607.5
5        Guitar                 6               120.5
6        Pencils                5               3.0
7        Chairs                 15              46.0
8        Book                   5               25.0
Displaying User Profile
User Profile Display from Server
Enter Action
1.Browse Items
2.Purchase Items
2
Enter the Item Number:
7
Enter the Quantity:
5
Item Purchase Successful
Displaying User Profile
User Profile Display from Server
Enter Action
1.Browse Items
2.Purchase Items
1
Browsing Items displayed here :
Item Id  Item Name              Quantity        Price
1        Tables                 17              5.75
2        Lights                 10              2.0
3        Mobile                 5               900.0
4        Speaker                4               607.5
5        Guitar                 6               120.5
6        Pencils                5               3.0
7        Chairs                 10              46.0
8        Book                   5               25.0
```

4. Admin Functionalities:

In this assignment, Add Items and browse items functionalities are implemented for the admin. Screenshot for them are below:

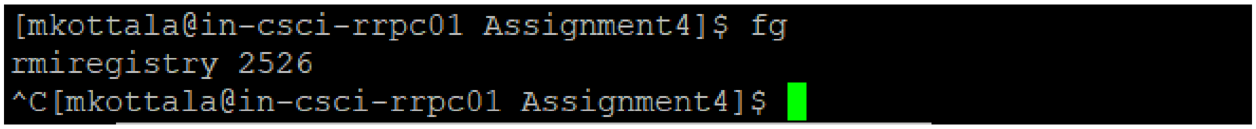


```
Enter Action as either 1 or 2:
1. User
2. Admin
2
View : Admin
Enter Id:
manju
Enter Password:
manju
Login Checking
Admin authentication is successful.
Displaying Admin Profile
For exit enter anything other than 1,2,3,4
1.Add Items
2.Delete Items
3.Update Items
4.Browse Items
4
Browsing Items displayed here :
Item Id  Item Name              Quantity        Price
1        Tables                 17              5.75
2        Lights                 10              2.0
3        Mobile                 5               900.0
4        Speaker                4               607.5
5        Guitar                 6               120.5
6        Pencils                5               3.0
7        Chairs                 10              46.0
8        Book                   5               25.0
Displaying Admin Profile
```



```
For exit enter anything other than 1,2,3,4
1.Add Items
2.Delete Items
3.Update Items
4.Browse Items
1
Adding items
Enter Item Id:
9
Enter Item Name:
Glass
Enter Item Quantity:
7
Enter Item Price:
9.5
Item has been added
Displaying Admin Profile
For exit enter anything other than 1,2,3,4
1.Add Items
2.Delete Items
3.Update Items
4.Browse Items
5
Exiting
[mkottala@in-csci-rrpc02 Assignment4]$
```

5. Terminating RMI registry:



```
[mkottala@in-csci-rrpc01 Assignment4]$ fg
rmiregistry 2526
^C[mkottala@in-csci-rrpc01 Assignment4]$ 
```

Conclusion:

In this assignment, I have understood the use of Java RMI and thread synchronization. I have understood how the RMI multi-threading works over multiple clients and how Java RMI concurrency works.

References:

[1] https://www.safaribooksonline.com/library/view/java-rmi/1565924525/ch11s06.html

[2] https://stackoverflow.com/questions/3507253/java-rmi-and-synchronized-methods

[3] https://docs.oracle.com/javase/6/docs/platform/rmi/spec/rmi-arch3.html

[4] https://stackoverflow.com/questions/1300145/race-condition-in-rmi-java-2-client-1-server

[5] http://www.informit.com/articles/article.aspx?p=26251&seqNum=3

[6] https://dzone.com/articles/uml2-class-diagram-java

[7] class slides