

---

# Online Market Place: Pattern-based Design

---

CSCI 50700 – Object-Oriented Design and Programming

Assignment-V

Implementation of Synchronization-Understanding design patterns

Summary Report

Under the guidance of

Professor Ryan Rybarczyk

Computer & Information Science, IUPUI

By

Yashwanth Reddy Kuruganti

Computer & Information Science, IUPUI

## Table of Contents

<b>1. Introduction.....</b>	<b>3</b>
<b>2. Assignment 4 Feedback .....</b>	<b>3</b>
<b>3. Functionalities implemented .....</b>	<b>3</b>
<b>Classes Added.....</b>	<b>3</b>
<b>Synchronization.....</b>	<b>3</b>
<b>Monitor Object.....</b>	<b>4</b>
<b>Future pattern .....</b>	<b>4</b>
<b>Guarded Suspension .....</b>	<b>4</b>
<b>Scoped Locking .....</b>	<b>4</b>
<b>Thread-safe Interface .....</b>	<b>5</b>
<b>Scenario Discussion.....</b>	<b>5</b>
<b>4. UML Diagrams.....</b>	<b>7</b>
<b>5. Sample Run.....</b>	<b>9</b>
<b>6. Conclusion .....</b>	<b>12</b>

## **1. Introduction**

This assignment aims at further integration of synchronization behavior to the previously constructed concurrent market place application using authorization, role-based access control, front controller, abstract factory and command pattern using a skeleton framework for Online Market place application, which is built in 1<sup>st</sup> assignment. Along with achieving synchronization using monitor object, future, guarded suspension, scoped locking and thread-safe interface, all the other pending functionalities like customer/admin registration, add items, browse items and purchase items are to be implemented.

## **2. Assignment 4 Feedback**

Fixed all the comments by importing only the required packages that are required in the java source.

## **3. Functionalities implemented**

Till assignment 4, several design patterns have been implemented to improve the functionality overall and enhance the performance of the application. Previous assignment ensures the application to exhibit concurrent behavior. In this assignment, synchronization and its related design patterns are to be implemented. Along with ensuring synchronization, all the other remaining functionalities like customer registration, check out items, add customers, remove items, update items etc., should be taken care of.

### **Classes Added**

AddUsers – allows admin to add customer and admin, ViewCart – allows customer to view cart items , AddItemsToCart – allows customer add browsed items to the cart for checkout, PurchaseItem – allows a customer to checkout from cart, DeleteCustomer – allows admin to remove customers, RemoveItems - allows admin to remove items, UpdateMarketItems – allows admin to update market items, ViewCustomers – allows admin to view customers and delete them if necessary.

### **Synchronization**

Previous assignment ensures that market application exhibits concurrent behavior with the help of Java RMI. Even though the application is concurrent, in a multi- threaded application several threads try to access a specific resource or resources at the same time. Access to specific resource at the same time by two or more threads may result in unexpected results for other threads. This is where concept of synchronization comes into play. Java provides synchronization by making use of synchronized keyword. This synchronized keyword is related to few design patterns which helps the application to behave both concurrently and synchronously. Synchronization can be provided by making use of either synchronized blocks or synchronized methods. In my market place application to provide this feature, I have made use of both the blocks and methods depending on the critical usage of that method. So, if there are multiple blocks are trying to access a specific object, synchronized keyword restricts to run only one thread making use of that object. All the other blocks remain in wait state till the other process releases the lock on that object. Few

scenarios which I have seen in my application will be explained after discussing the design patterns[1] which provide synchronization.

## **Monitor Object**

For a concurrent application to behave in a thread safe manner, there should be some efficient mechanism to access resources. Also, other than shared resources, there will always be few objects that shouldn't be shared. Access to such resources can be done using a queue from which multiple threads try to access or insert the data. So, a monitor facilitates the thread to have mutual exclusion as well as make it wait till a condition is valid or true [2]. When a condition is met, other threads receive a signal saying that condition is met. Making use of this design pattern will help each thread to run a shared object allowing interleaved and self-co ordination execution. Then this object access is restricted only to synchronized methods, thus only one method at a time can access. Separation of concerns by providing how the data should be accessed. Making use of this pattern in market place application with the help of synchronized keyword provides wait, notify and notifyall methods and always ensures the only one method at a time. Examples in market place include purchaseItems(), addItem() etc.,

## **Future pattern**

When a block tries to access an object, which is already in execution must wait till the object is available. So, when a client tries to access the service or shared resource, then a virtual data object is returned to the client promising that it will notify once the result is available. Future helps to maintain the status of concurrent execution of the requested service and reports back to waiting block that execution is completed, hence working as a virtual proxy. To make this happen in my market place application, synchronized keyword is used in OnlineMarketController and OnlineMarketModel for various methods.

## **Guarded Suspension**

Suppose in my marketplace application, I should provide access to a critical section only when a specific condition is met, this can be done by making use of guarded suspension. In this pattern, along with acquiring the lock, a pre-condition should be satisfied for the execution of the method call. If the pre-condition is not met then the client method call is sent to wait state. This method is in wait state for a point of time and will resume execution only when the pre-condition is satisfied. This pattern should only be used if we know that the method call will be suspended for a reasonable time period.

## **Scoped Locking**

The motivation behind Scoped locking is that locks should be acquired and released automatically for avoiding the deadlock situations and problems for other thread to acquired locks. This pattern states to define a scope for the critical section and acquire a lock automatically when the control enters the scope, locks are released automatically when the control leaves the scope. This helps in reducing the chance of error in program execution. This pattern is used in the Monitor object pattern for synchronizing a block. If not used appropriately a self-deadlock situation may

arise. This pattern is internally implemented when we used monitor object pattern in my market place application.

### Thread-safe Interface

An application may contain multiple functionalities and several users try accessing the same functions at the same time. All the operations performed should be thread safe and avoid self-deadlock which occurs due to intra-component calls. To overcome such problems, I have made use of thread-safe pattern that ensures the separation of public interface methods and their respective private implementations. This way only the public methods can call their definitions and acquire a lock. OnlineMarket has all the interface methods which are implemented in the OnlineMarketModel. So if addUsers method is called by multiple users, the interface method request a lock and releases it when the actual addUsers() implementation is processed. This way synchronization check is done at the top layer instead of digging into the application. All the other private or protected methods will only function when the interface calls them, thus improving locking mechanism and synchronization.

### Scenario Discussion

```
^C[yashkuru@in-csci-rrpc01 4Assignment]$ sh makefileS.sh
You are now entering Online Market Place
Reaching server://10.234.136.55:5432/OnlineMarketServer
Interface is Ready!You can register, login and shop
Registration page. Register here
Success
Connecting to Market App Database.....
=====Your can Browse Market App to shop=====
Registration page. Register here
Success
Connecting to Market App Database.....
=====Your can Browse Market App to shop=====
Connecting to Market App Database.....
=====Accessed Admin add method=====
Connecting to Market App Database.....
=====Your can Browse Market App to shop=====
^C[yashkuru@in-csci-rrpc01 4Assignment]$ fg
rmiregistry 5432
```

### Single Server

```
yashkuru@in-csci-rrpc01:~/OOAD/4Assignment
201802 Table 235$ 10
201803 Laptop 435.75$ 67
---Enter 'Browse' ignoring quotes to shop
---Enter 'Purchase' ignoring quotes to buy items
>>To exit these commands, press ctrl+c<<
purchase
+++++Purchase items here+++++
Enter Item Id of the above Items you wish: browse
Online Market App Exception: null
---Enter 'Browse' ignoring quotes to shop
---Enter 'Purchase' ignoring quotes to buy items
>>To exit these commands, press ctrl+c<<
browse
<-----Your shopping items list here----->
ItemId ItemName ItemPrice ItemQuantity
55555 PC 325$ 78
78964 DSLR 799.99$ 6
201801 Chair 25$ 13
201802 Table 235$ 10
201803 Laptop 435.75$ 67
---Enter 'Browse' ignoring quotes to shop
---Enter 'Purchase' ignoring quotes to buy items
>>To exit these commands, press ctrl+c<<

yashkuru@in-csci-rrpc02:~/OOAD/4Assignment
browse
<-----Your shopping items list here----->
ItemId ItemName ItemPrice ItemQuantity
55555 PC 325$ 78
201801 Chair 25$ 13
201802 Table 235$ 10
201803 Laptop 435.75$ 67
---Enter 'Browse' ignoring quotes to shop
---Enter 'Add' ignoring quotes to buy items
---Other commands coming soon.....<
>>To exit these commands, press ctrl+c<<
add
+++++Add items here+++++
Enter Item Id: 78964
Enter Item Name: DSLR
Enter Item Price: 799.99$
Enter Item Quantity: 6
+++++Above item has been added to database+++++
---Enter 'Browse' ignoring quotes to shop
---Enter 'Add' ignoring quotes to buy items
---Other commands coming soon.....<
>>To exit these commands, press ctrl+c<<
```

```
yashkuru@in-csci-rrpc04:~/OOAD/4Assignment
^C[yashkuru@in-csci-rrpc04 4Assignment]$ sh makefileC.sh
Registration ID: 1
Registration Status: Registered
-----
Enter 'Admin' for Administrator login without quotes
Enter 'Customer' for Customer login without quotes
-----Enter one from above-----
admin
Enter Login ID:
admin
Enter Password:
test
Login Status:true
You are now accessing market application as: admin
-----
---Welcome to the Admin Home Page---
-----
You can now Browse, Add, Delete, Update
Hi Admin! You have the following commands to perform
---Enter 'Browse' ignoring quotes to shop
---Enter 'Add' ignoring quotes to buy items
---Other commands coming soon.....<
>>To exit these commands, press ctrl+c<<
add
+++++Add items here+++++
Enter Item Id: 90898
Enter Item Name: Microwave
Enter Item Price: 234$
Enter Item Quantity: 123
+++++Above item has been added to database+++++
```

Fig 1. Three Clients showing concurrent execution

Using sync keyword for all the methods in a distributed application is not an efficient way to do things. Doing this may lead to concurrency blockage or sometimes deadlocks. To demonstrate Java RMI concurrency, consider the above two clients which tries to interact with the server. In the first step, rmi registry will register its services with a port number 5432 and makes its services available. Now all the java source files will be compiled and then the server-side controller. Now the server-side stub publishes its services on the registered port. Client-side controller can make requests to the server. Getting into details, on the client side as soon as the request is sent by the client 1, it gets marshalled and passed through the network. This thread which sends the first request tries to receive a response from the server side -meaning, current thread moves to block state till it receives a response. On the other side i.e., server side another thread takes care of this request and processes it to send a response. Two clients in the above example tried to add items during which another client tries to retrieve the items from database. So, the first insert request is taken by a thread from Java RMI and this gets a lock on the database till its writing is finished. Now the second client waits till the first one finishes as it must do the same writing. Meanwhile another thread send a request to select all the items from the database and gets all the recent entries that are input to the database.

One should be able to make concurrent requests to the server regarding same operation at the same time. This would happen in the real time with high probability. But in my application as I was unable to do concurrent operations at same time, due to Java RMI application still provides effective communication in performing requested operations. All this is possible due to remote method invocation and internal thread behavior of Java RMI. Sometimes this may lead to problems like: Admin can browse and add items to the inventory whereas customer can only browse and purchase those items. Customer should always see an updated list and quantity of items. As soon as customer purchases an item its quantity should be updated so the customer with faster access gets the item and the other one loses the chance. If there is no concurrency both the users may purchase the same item even though available stock is 1 unit.

#### **4. UML Diagrams**

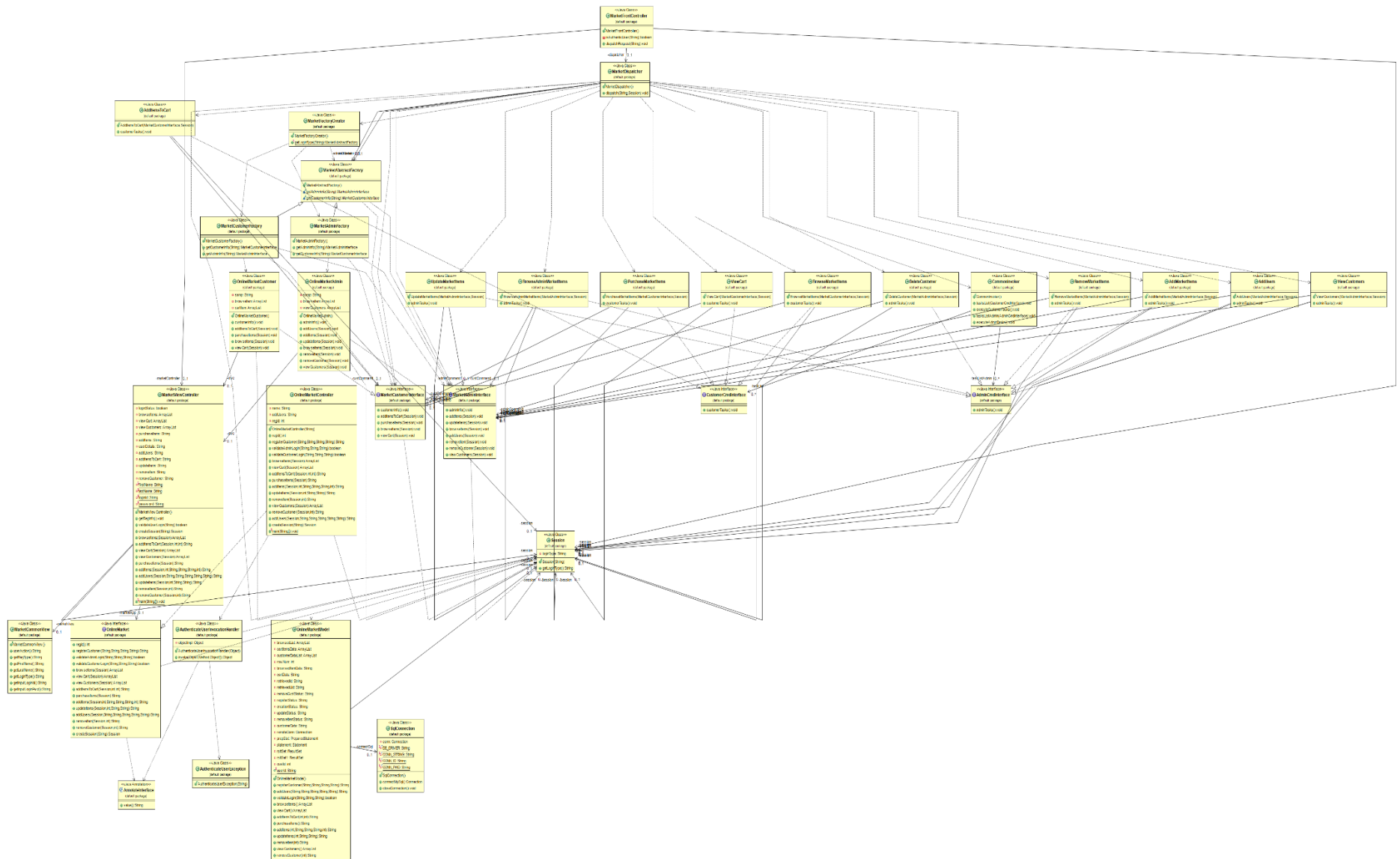


Fig 2. Class Diagram [please refer git for a better view]



## 5. Sample Run

Steps on running this software is given in Readme file. Simple way of executing is to run makefiles.

- Running server and registry

```
^C[yashkuru@in-csci-rrpc01 5Assignment]$ rmiregistry 5432&
[1] 15858
[yashkuru@in-csci-rrpc01 5Assignment]$ sh makefileS.sh
You are now entering Online Market Place
Reaching server://10.234.136.55:5432/OnlineMarketServer
Interface is Ready!You can register, login and shop
```

- Admin\_functionalities


```
yashkuru@tesla:~/OOAD/5Assignment
[yashkuru@tesla 5Assignment]$ sh makefileC.sh
~~~~~Login/Registration~~~~~
Enter 'Login' without quotes for Customer/Admin Login
Enter 'Register' without quotes for registration a new Customer account
-----Enter one from above-----
login
~~~~~Login~~~~~
Enter 'Admin' for Administrator login without quotes
Enter 'Customer' for Customer login without quotes
-----Enter one from above-----
admin
Enter UserId:
admin
Enter Password:
admin
Login Status:true
You are now accessing market application as: admin
-----
---Welcome to the Admin Home Page---
-----
You can now Browse, Add, Delete, Update
Hi Admin! You have the following commands to perform
---Enter 'browse' ignoring quotes to browse items
---Enter 'addI' ignoring quotes to add items
---Enter 'addU' ignoring quotes to add Customer/Admin
---Enter 'update' ignoring quotes to update items
---Enter 'removeC' ignoring quotes to delete Customer
---Enter 'removeI' ignoring quotes to delete Item
---Enter 'view' ignoring quotes to view all Customers
>>>...To exit these commands, press ctrl+c...<<<
view
*-----List of Customers-----*
CustomerId  UserName
59-----john
60-----pete
57-----ramu
61-----tank
58-----tom
55-----yash
63-----yash1
64-----yash2
56-----yashkuru
<----->
---Enter 'browse' ignoring quotes to browse items
```

```

<----->
---Enter 'browse' ignoring quotes to browse items
---Enter 'addI' ignoring quotes to add items
---Enter 'addU' ignoring quotes to add Customer/Admin
---Enter 'update' ignoring quotes to update items
---Enter 'removeC' ignoring quotes to delete Customer
---Enter 'removeI' ignoring quotes to delete Item
---Enter 'view' ignoring quotes to view all Customers
>>>...To exit these commands, press ctrl+c...<<<
addU
+++++++Add Customer/Admin here+++++++
Enter 'Admin' to add a new admin to database
Enter 'Customer' to add a new admin to database
-----Enter one from above-----
customer
Enter First Name:
paul
Enter Last Name:
pete
Enter UserId:
paul
Enter Password:
paul
You have now successfully created a Customer account
-----Enter one from above-----
addU
+++++++Add Customer/Admin here+++++++
Enter 'Admin' to add a new admin to database
Enter 'Customer' to add a new admin to database
-----Enter one from above-----
customer
Enter First Name:
paul
Enter Last Name:
pete
Enter UserId:
paul
Enter Password:
paul
You have now successfully created a Customer account
---Enter 'browse' ignoring quotes to browse items
---Enter 'addI' ignoring quotes to add items
---Enter 'addU' ignoring quotes to add Customer/Admin
---Enter 'update' ignoring quotes to update items
---Enter 'removeC' ignoring quotes to delete Customer
---Enter 'removeI' ignoring quotes to delete Item
---Enter 'view' ignoring quotes to view all Customers
>>>...To exit these commands, press ctrl+c...<<<
removedC
*-----Remove Customers here-----*
Enter Customer Id to delete the customer: 59
Delete failed. Invalid Customer Id
---Enter 'browse' ignoring quotes to browse items
---Enter 'addI' ignoring quotes to add items
---Enter 'addU' ignoring quotes to add Customer/Admin
---Enter 'update' ignoring quotes to update items
---Enter 'removeC' ignoring quotes to delete Customer
---Enter 'removeI' ignoring quotes to delete Item
---Enter 'view' ignoring quotes to view all Customers
>>>...To exit these commands, press ctrl+c...<<<
update
++++-----++Update items here++++-----++
Enter Item Id to update: 55555
----Enter Item attribute to be updated----
Any of the attributes either: price or quantity or desc::
price
Enter Item attribute Value: 599.99$
+++++++Above item has been updated to database+++++++

```

- Customer functionalities

 yashkuru@in-csci-rrpc02:~/OOAD/5Assignment

```
Last login: Tue Apr  3 23:59:13 2018 from 140-182-64-30.ssl-vpn.iupui.edu
[yashkuru@in-csci-rrpc02 ~]$ cd OOAD/5Assignment/
[yashkuru@in-csci-rrpc02 5Assignment]$ sh makefileC.sh
~~~~~Login/Registration~~~~~
Enter 'Login' without quotes for Customer/Admin Login
Enter 'Register' without quotes for registration a new Customer account
-----Enter one from above-----
register
----->> New Customer Registration <<-----
Enter First Name:
maya
Enter Last Name:
yama
Enter UserId:
maya
Enter Password:
maya
----- You are now successfully Registered -----
~~~~~Login/Registration~~~~~
Enter 'Login' without quotes for Customer/Admin Login
Enter 'Register' without quotes for registration a new Customer account
-----Enter one from above-----
login
~~~~~Login~~~~~
Enter 'Admin' for Administrator login without quotes
Enter 'Customer' for Customer login without quotes
-----Enter one from above-----
customer
Enter UserId:
maya
Enter Password:
maya
Login Status:true
You are now accessing market application as: customer
-----
-Welcome to the Customer Home Page-
-----
Hi Customer! You have the following commands to perform
---Enter 'Browse' ignoring quotes to shop
---Enter 'Add' ignoring quotes to add items to the cart
---Enter 'View' ignoring quotes to view items in cart
---Enter 'Purchase' ignoring quotes to checkout items from the cart
>>To exit these commands, press ctrl+c<<
```

yashkuru@in-csci-rrpc04:~/OOAD/5Assignment

```
---Enter 'Purchase' ignoring quotes to checkout items from the cart
>>To exit these commands, press ctrl+c<<
browse
<---+++---Your shopping items list here---+++--->
ItemId ItemName ItemType ItemPrice ItemQuantity
55555----Computer-----Tech-----499.99$-----65
78784----Cooke-----59.99$-----22
78964----Dslr-----Tech-----799.99$-----4
123456----Papers-----Stationery-----4.99-----6
<----->
---Enter 'Browse' ignoring quotes to shop
---Enter 'Add' ignoring quotes to add items to the cart
---Enter 'View' ignoring quotes to view items in cart
---Enter 'Purchase' ignoring quotes to checkout items from the cart
>>To exit these commands, press ctrl+c<<
add
+++++++Add items to cart here+++++++
Enter Item Id of the above Items you wish: 55555
Enter Item Quantity to be purchased: 5
Your item Computer has been added to cart successfully
---Enter 'Browse' ignoring quotes to shop
---Enter 'Add' ignoring quotes to add items to the cart
---Enter 'View' ignoring quotes to view items in cart
---Enter 'Purchase' ignoring quotes to checkout items from the cart
>>To exit these commands, press ctrl+c<<
view
<---+++---Your Cart items list here---+++--->
CartId ItemId ItemQuantity
100-----55555-----5
---Enter 'Browse' ignoring quotes to shop
---Enter 'Add' ignoring quotes to add items to the cart
---Enter 'View' ignoring quotes to view items in cart
---Enter 'Purchase' ignoring quotes to checkout items from the cart
>>To exit these commands, press ctrl+c<<
purchase
+++++++Check out items here+++++++
<<<<< 55555 Purchase Successful >>>>

---Enter 'Browse' ignoring quotes to shop
---Enter 'Add' ignoring quotes to add items to the cart
---Enter 'View' ignoring quotes to view items in cart
---Enter 'Purchase' ignoring quotes to checkout items from the cart
>>To exit these commands, press ctrl+c<<
```

## 6. Conclusion

From this assignment, I have learnt and understood how synchronization and concurrency can be achieved in a distributed online application. Also explored various synchronization design patterns that involve thread-safe behavior of the online app. Implemented full functionalities related to admin and customer along with mysql database usage.

## References

[1] In-Class slides

[2] [https://en.wikipedia.org/wiki/Monitor\\_\(synchronization\)](https://en.wikipedia.org/wiki/Monitor_(synchronization))