
Online Market Place: Pattern-based Design

CSCI 50700 – Object-Oriented Design and Programming

Assignment-IV

Impact of Concurrency

Summary Report

Under the guidance of

Professor Ryan Rybarczyk

Computer & Information Science, IUPUI

By

Yashwanth Reddy Kuruganti

Computer & Information Science, IUPUI

Table of Contents

1. Introduction.....	3
2. Assignment 3 Feedback	3
3. Functionalities implemented	3
<u> </u> SqlConnection.java	3
<u> </u> Java RMI Concurrency- Set up [1].....	3
<u> </u> Without synchronized keyword	5
4. UML Diagrams.....	5
5. Sample Run.....	7
6. Conclusion	9

1. Introduction

This assignment aims at further integration of concurrent behavior to the previously constructed 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 1st assignment. Along with achieving concurrency, all the other pending functionalities like add Items, browse Items and purchase Items are to be implemented.

2. Assignment 3 Feedback

Fixed all the comments by changing the level of visibility of all the variables in a class.

3. Functionalities implemented

Before exploring more about concurrency and customer/admin commands, let us look at the total flow till assignment 3. Many design patterns have been implemented till the last one through which each pattern adds a specific functionality to the application. First assignment concentrates on building basic skeleton and establish communication between client and server through Java RMI. In the second assignment, importance of front controller pattern which provides an entry point and authentication and abstract factory which can create multiple factories of customer and admin is explored. Third assignment concentrates on providing role-based access using Java annotations and supports distributed exception handling. Current assignment concentrates on integrating multi-threading feature to previous existing model and achieve concurrency. To test this feature, I will make use of browse, add and purchase items methods.

SqlConnection.java

To simulate the concurrency feature, I have made use of given 6 hosts. Out of these one will be a designated server and rest will act as multiple clients. Our application should work fine even when all these clients try to access the application in the same time. To understand how concurrency works, I have implemented a MySQL database which provide persistent storage for my application. in-csci-rrpc01.cs.iupui.edu is used as server which also contains phpMyAdmin support. Jdbc driver requires my sql connector jar file for driver lookup com.mysql.jdbc.Driver. Connection string jdbc:mysql://localhost:3306/yashkuru_db is where our database resides. This class is instantiated to create and establish a new database connection. Items table is created, and all the operations are performed on this table using Java jdbc connection. In the next assignment, I will be using singleton pattern to ensure that only one instance of database is created.

Java RMI Concurrency- Set up [1]

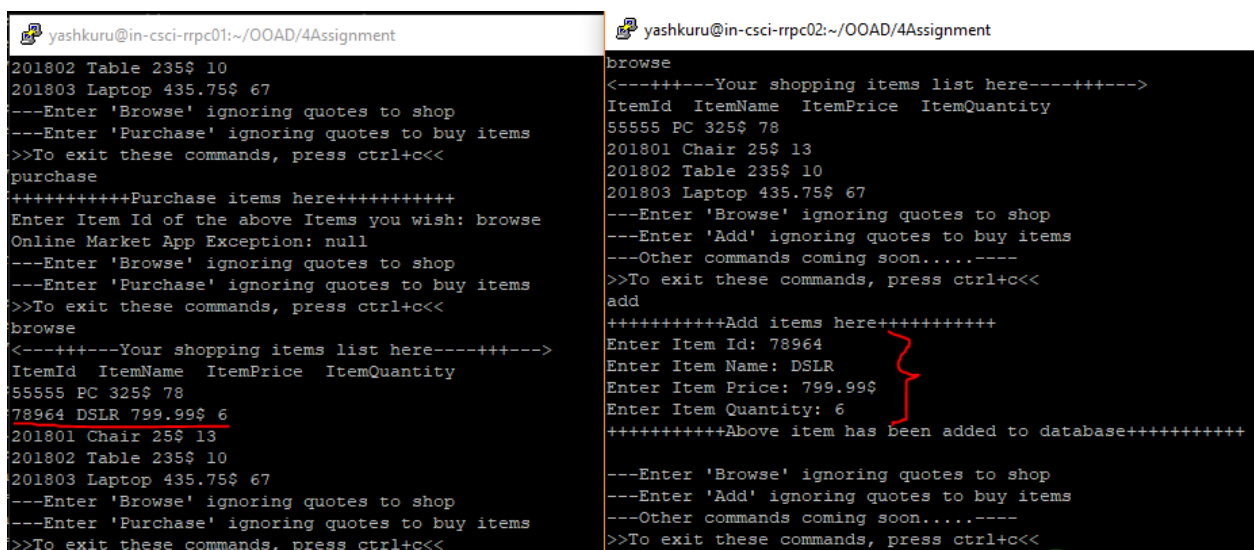
Concurrency is the property in which multiple actions of a code block can be done at the same time. Achieves multi-threading through parallel execution of several operations i.e., several tasks are performed by several java threads. Major problem with concurrency occurs when two operations tries to access the same shared data. This may lead to inconsistent data and user may receive stale data or invalid data or sometimes half processed data [2]. To ensure consistency, Java makes use of locks that prevents other methods to wait till the shared data is available. Keyword synchronized is used for required methods (add, purchase and browse) to achieve such behavior in our application. Only a single thread will execute Using this keyword will ensure that a block

of code will be executed by only a single thread at given time. This also ensures that the current block execution can see all the updates that are done by the same method in previous stage. In the coming sections, I'll jump into setup to observe concurrency mechanism. We'll also see multiple clients try to access same method when synchronized keyword is absent.

In our application, Java RMI provides us the feature of multi-threading. To this I will make use of synchronized keyword and observe the concurrent behavior. Server is 10.234.136.55 and all the other machines will act as clients. Along with this my application should ensure concurrent execution of threads or all the operations performed should be thread safe. Java RMI stub request will always be synchronized internally. This stub will help in preventing the same method being invoked by several threads at the same time.

```
^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<<
```

Two 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

Without synchronized keyword

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

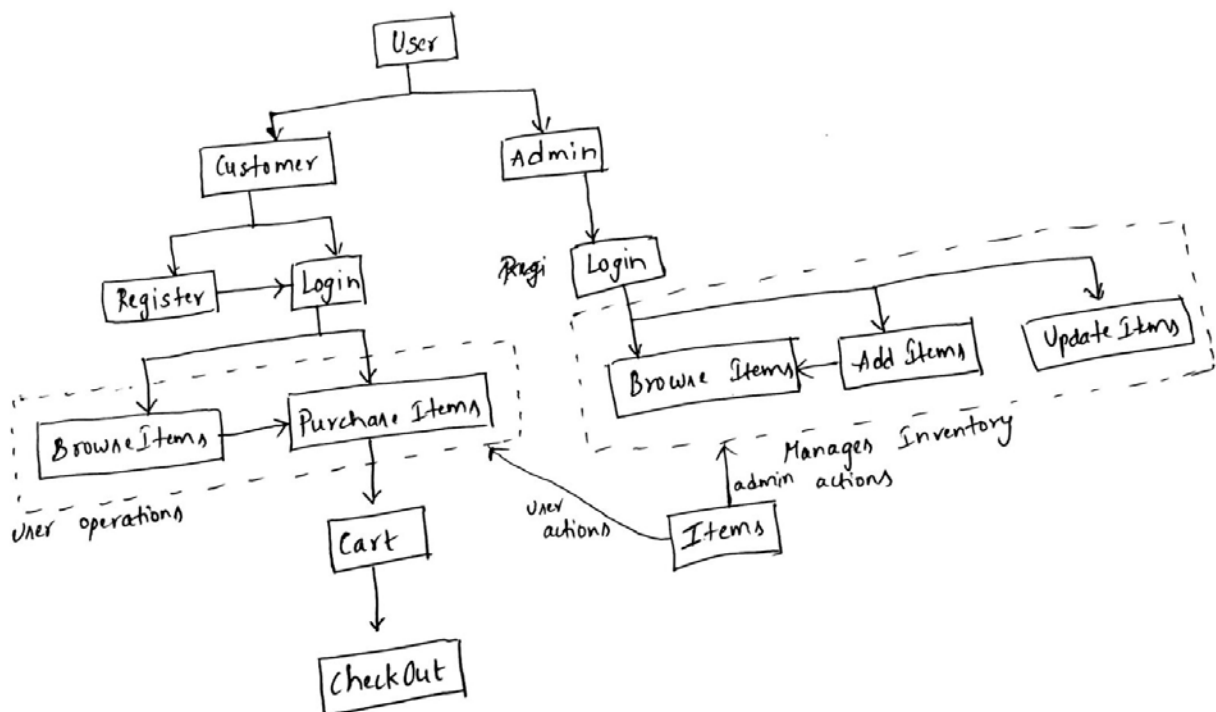


Fig 1. Domain Model updated

5. Sample Run

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

- Add and browse items for Admin

```
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+++++++

---Enter 'Browse' ignoring quotes to shop
---Enter 'Add' ignoring quotes to buy items
---Other commands coming soon.....----
>>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
90898 Microwave 234$ 123
201801 Chair 25$ 13
201802 Table 235$ 10
201803 Laptop 435.75$ 67
```

- Purchase and browse items for customer on a different client

yashkuru@in-csci-rrpc05:~/OOAD/4Assignment

```
[yashkuru@in-csci-rrpc05 4Assignment]$ sh makefileC.sh
Registration ID: 2
Registration Status: Registered
~~~~~
Enter 'Admin' for Administrator login without quotes
Enter 'Customer' for Customer login without quotes
-----Enter one from above-----
customer
Enter Login ID:
customer
Enter Password:
test
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 '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
90898 Microwave 234$ 123
201801 Chair 25$ 13
201802 Table 235$ 10
201803 Laptop 435.75$ 67
234567 USB Cable 89
---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: 78964
Enter Item Quantity to be purchased: 2
Your item DSLR has been purchased successfully
---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----+++--->
```



```
<---+++---Your shopping items list here---+++-->
ItemId  ItemName  ItemPrice  ItemQuantity
555555  PC 325$ 78
78964   DSLR 799.99$ 4
90898   Microwave 234$ 123
201801  Chair 25$ 13
201802  Table 235$ 10
201803  Laptop 435.75$ 67
234567  USB Cable 89
---Enter 'Browse' ignoring quotes to shop
---Enter 'Purchase' ignoring quotes to buy items
>>To exit these commands, press ctrl+c<<
```

6. Conclusion

From this assignment, I have learnt and understood how concurrency can be achieved in a distributed online application. Also explored how Java RMI tries to simulate the concurrent executions of various methods in a single application, hence supporting multi-threading. Adding to this, I have observed how multiple clients communicate with single designated server and proceed with their respective operations.

7. References

[1] In-Class slides

[2] <http://www.vogella.com/tutorials/JavaConcurrency/article.html>