

Assignment 6



TABLE OF CONTENTS

ASSIGNMENT OVERVIEWS..... 1

Assignment 1	1
Assignment 2	1
Assignment 3	2
Assignment 4	2
Assignment 5	4

ASSIGNMENT6 DISCUSSION 6

Database Access Layer pattern	6
-------------------------------------	---

FIGURES, SAMPLE RUNS..... 7

UML Diagram	7
Screenshots.....	8

CONCLUSION, REFERENCES..... 16

Conclution	16
References	17

ASSIGNMENTS

OVERVIEW

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.

ASSIGNMENT 1 OVERVIEW

The goal of this assignment was using RMI and design the software architecture using MVC design pattern. During it, we investigate java RMI features and how RMI can fit our desire software requirements.

We found that using MVC, we can reduce coupling while increasing cohesion and it was our goal, however RMI as an infrastructure, add some level of coupling to our software which was unavoidable.

The main challenge of first assignment was designing the controllers and making decision about where they should be with respect to MVC, RMI and Client-Server design.

ASSIGNMENT 2 OVERVIEW

In second assignment we used several pattern such as front controller, command pattern and abstract factory. At the time of assignment, I wasn't so sure about abstract factory and its advantages, however later while using role based access, I found it very useful.

About others, I believe front controller was helpful in terms of centralizing requests but again, about coupling, I'm not still happy with it. However, command pattern helps in terms of extensibility as we can add a new command without changing the existing code.

And about Abstract Factory, I see its pro's very clear since it allows me to hide implementation of an application seam (the core interfaces that make up my application) and also lets me easily test the seam of an application (that is to mock/stub) certain parts of my application so I can build and test the other parts. In addition, this pattern lets me change the design of my application more readily, which is known as loose coupling.

ASSIGNMENT 3 OVERVIEW

The goal of third assignment was restricting access to resources and to do so, we used Role Based Access Control approach which is a tool of regulating access to resources based on the **roles** of individual users within an enterprise. In this context, access is the ability of an individual user to perform a specific task, such as view, create, or modify a file. To implement such a tool, we used java proxy and java annotation. However, still there is a coupling between RMI and this subsystem, we are restricting users through our RMI methods, and also we have to keep in mind that proxy has some overhead. For most use cases the overhead won't be significant, though. The real problem is that the (over)use of dynamic proxies makes an application harder to understand and debug.

ASSIGNMENT 4 OVERVIEW

Forth assignment had focused on concurrency in java RMI and we had to investigate it. In our application, since the system is distributed, logically, blocking in a remote object is simple. Suppose that client A calls a synchronized method of a remote object. To make access to remote objects look always exactly the same as to local objects, it would be necessary to block A in the client-side stub that implements

the object's interface and to which A has direct access. Likewise, another client on a different machine would need to be blocked locally as well before its request can be sent to the server. The consequence is that we need to synchronize different clients at different machines. An alternative approach would be to allow blocking only at the server. In principle, this works fine, but problems arise when a client crashes while its invocation is being handled by the server. We may require relatively sophisticated protocols to handle this situation, and which that may significantly affect the overall performance of remote method invocations.

Therefore, the designers of Java RMI have chosen to restrict blocking on remote objects only to the proxies. This means that threads in the same process will be prevented from concurrently accessing the same remote object, but threads in different processes will not. Obviously, these synchronization semantics are tricky: at the syntactic level (ie, when reading source code) we may see a nice, clean design. Only when the distributed application is actually executed, unanticipated behavior may be observed that should have been dealt with at design time.

If I implement my RMI remote methods as synchronized, RMI does not provide such guarantee that they are mutually exclusive on its own (unlike EJB) and two calls on the same remote object may be executed concurrently unless you implement some synchronization. Also, if I have a method that the server executes periodically. It is used to do cleanups. I have to make sure that this particular method does not execute when there is any RMI method being run/used by remote clients. If the cleanup job is in another class, I will need to define a lock that you will share between the remote object and the cleanup job. In the remote object, define an instance variable that I will use as a lock.

ASSIGNMENT 5 OVERVIEW

During the last assignment, we had to have a distributed system while has all required functionalities as well as being synchronized. In assignment 5, we also had to investigate several pattern such as monitor object, Future, Guarded Suspension, Scoped Locking and Thread-Safe Interface.

To achieve such a system, we first defined the synchronization as having order in execution of clients requests and define our critical blocks. In our 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.

We found that "monitor" is a mechanism that ensures that only one thread can be executing a given section (or sections) of code at any given time. This can be implemented using a lock (and "condition variables" that allow threads to wait for or send notifications to other threads that the condition is fulfilled), but it is more than just a lock.

In context of asynchronization, Future represents the result of an asynchronous computation. Methods are provided to check if the computation is complete, to wait for its completion, and to retrieve the result of the computation. The result can only be retrieved using method `get` when the computation has completed, blocking if necessary until it is ready. Cancellation is performed by the `cancel` method. Additional methods are provided to determine if the task completed normally or was cancelled. Once a computation has completed, the computation cannot be cancelled. If you would like to use a Future for the sake of cancellability but not provide a usable result, you can declare types of the form `Future<?>` and return `null` as a result of the underlying task.

And about Guarded Suspension, I found it useful to handle a situation when you want to execute a method on object which is not in a proper state, so the developer uses it when knows that the method execution will be blocked for a finite period of time.

The Thread-Safe Interface design pattern minimizes locking overhead and ensures that intra-component method calls do not incur 'self-deadlock' by trying to reacquire a lock that is held by the component already.

And finally, about scoped lock and its implementation, we simply can say this is a way that ensure a lock is acquired when control enters a scope and release it automatically when leave the scope.

During last assignment, we also had to use database and to do so, 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.

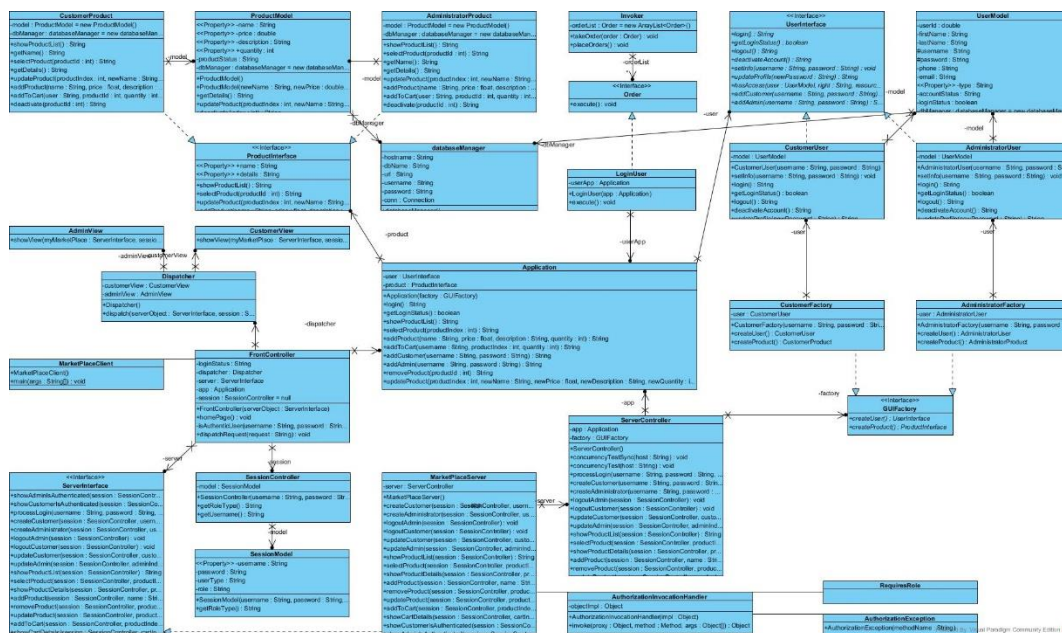
ASSIGNMENT DISCUSSION

In this assignment, we are investigating the usage of database access layer pattern in order to decoupling the application logic from the database functionality. In different resources, this layer has been implemented in several ways. In my design, I have two models(userModel and productModel) and one query manager. At first glance, I decided to add two new classes to separate data layer from logic of my models but I found there is no logic in my models and based on what I found on DAL and DAO, it seems I can name a composition of my models and my database manager as database access layer and I already achieved decoupling and separation. The most benefit from DAL is that it simplifies database access operations through some method rather than making connection and executing some queries.

FIGURE, SAMPLE RUNS

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

UML DIAGRAM



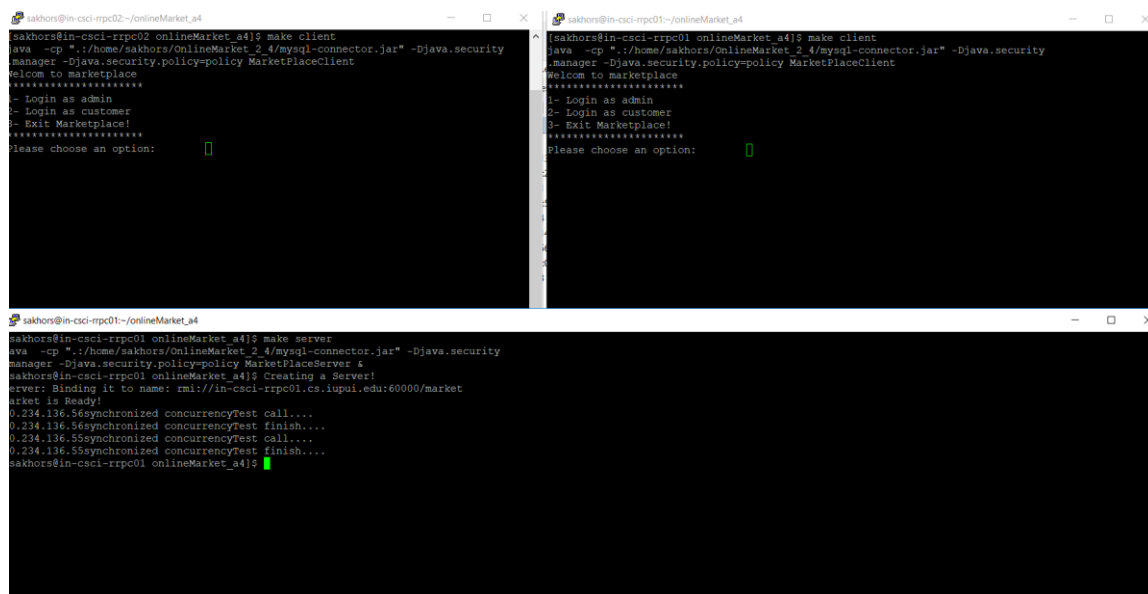
The original UML diagram has been provided in Documentation directory. As you see, in our proposed architecture, we can name the

set of two models and the database manager class as Database Access layer.

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.

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-rpc02:~/onlineMarket_a4$ make client
java -cp ".*:/home/sakhors/OnlineMarket_2_4/mysql-connector.jar" -Djava.security.manager -Djava.security.policy=policy MarketPlaceClient
Welcome to marketplace
*****
1- Login as admin
2- Login as customer
3- Exit Marketplace!
*****
Please choose an option: 0

sakhors@in-csci-rpc01:~/onlineMarket_a4$ make client
java -cp ".*:/home/sakhors/OnlineMarket_2_4/mysql-connector.jar" -Djava.security.manager -Djava.security.policy=policy MarketPlaceClient
Welcome to marketplace
*****
1- Login as admin
2- Login as customer
3- Exit Marketplace!
*****
Please choose an option: 0

sakhors@in-csci-rpc01:~/onlineMarket_a4$ make server
java -cp ".*:/home/sakhors/OnlineMarket_2_4/mysql-connector.jar" -Djava.security.manager -Djava.security.policy=policy MarketPlaceServer &
sakhors@in-csci-rpc01:~/onlineMarket_a4$ creating a Server!
server: Binding it to name: rmi:///in-csci-rpc01.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-rpc01:~/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:      █
```

CONCLUTION

REFERENCES

CONCLUTION

During this semester, several design patterns have been introduced to us and I found most of them useful with respect to our goals.

If I want to be specific in order to answer questions, what I liked about this assignment was overview of previous assignments which help me to see the whole project and different decisions that we made in a broader picture and relate all of them together and I have nothing to dislike about this assignment!

If I had time and can go back, probably I change my design with respect to get more control over handling synchronization which currently I'm using synchronized keyword.

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/>
- <http://www.cs.wustl.edu/~schmidt/PDF/ScopedLocking.pdf>
- <https://wiki.hsr.ch/PnProg/files/ThreadSafeInterface.pdf>
- <http://www.cs.wustl.edu/~schmidt/PDF/locking-patterns.pdf>
- <https://docs.oracle.com/javase/8/docs/api/index.html?java/util/concurrent/Future.html>
- *Java and developer's forums*
- *Lecture slides*