

Market Place

Assignment Report 3

Virinchi Sainath Nalluri

vnalluri@iupui.edu

13th February, 2018



Table of Contents

1. Objective
2. Assignment Discussion
 - 2.1. Role of Authorization Pattern
 - 2.2. Role of Reflection Pattern
3. UML Sequence Diagrams
4. Transition from Assignment 1
5. Addressing Comments from Assignment 1
6. Sample Runs
7. Conclusion and Analysis
8. References

Objective

The objective of this assignment is to implement role-based access control (RBAC) using authorization pattern using Java Annotations. Also to implement proxy pattern and reflection pattern in implemented as part of this assignment.

Assignment Discussion

Role of **Authorization Pattern**

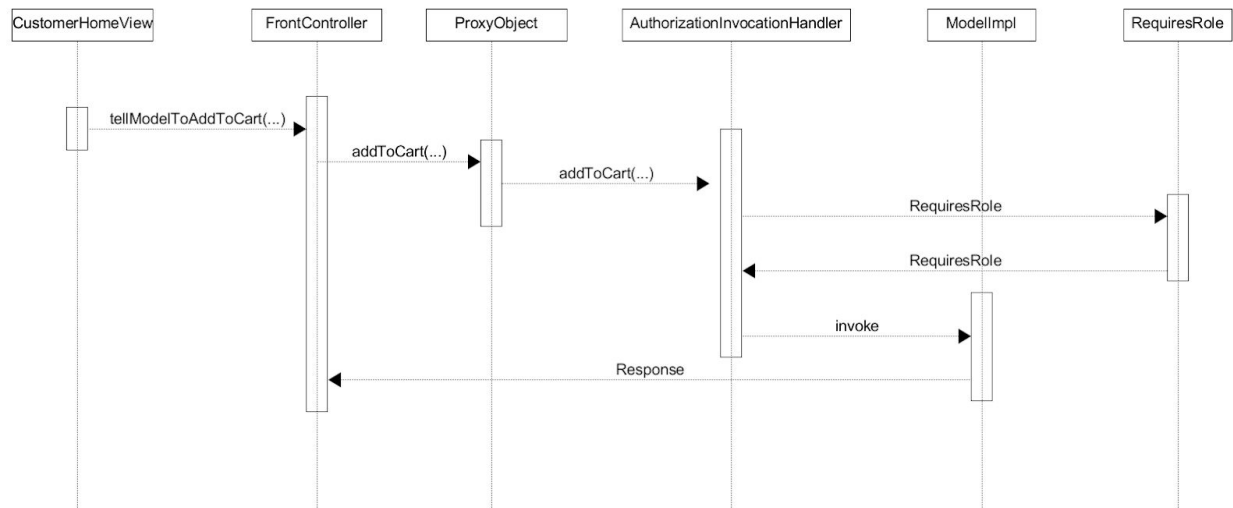
The authorization pattern is used to provide role-based access control. That is few methods are custom to particular roles for example., an admin can't access can't add to cart and purchase. Only a customer can add to a cart and purchase. Therefore, when he is prompted with not authorized message when he tries to add a product to cart.

We can accomplish this using annotations. The methods custom to customer are give an annotation `@RequiredRole("customer")`. There the class `AuthorizationInvokationHandler` class only allows the method if the caller has a user type customer. It checks whether the caller and the required role is same or not by using Java Refecltion API (`getAnnotation`). If they don't match then authorization error is raised.

Role of **Reflection Pattern**

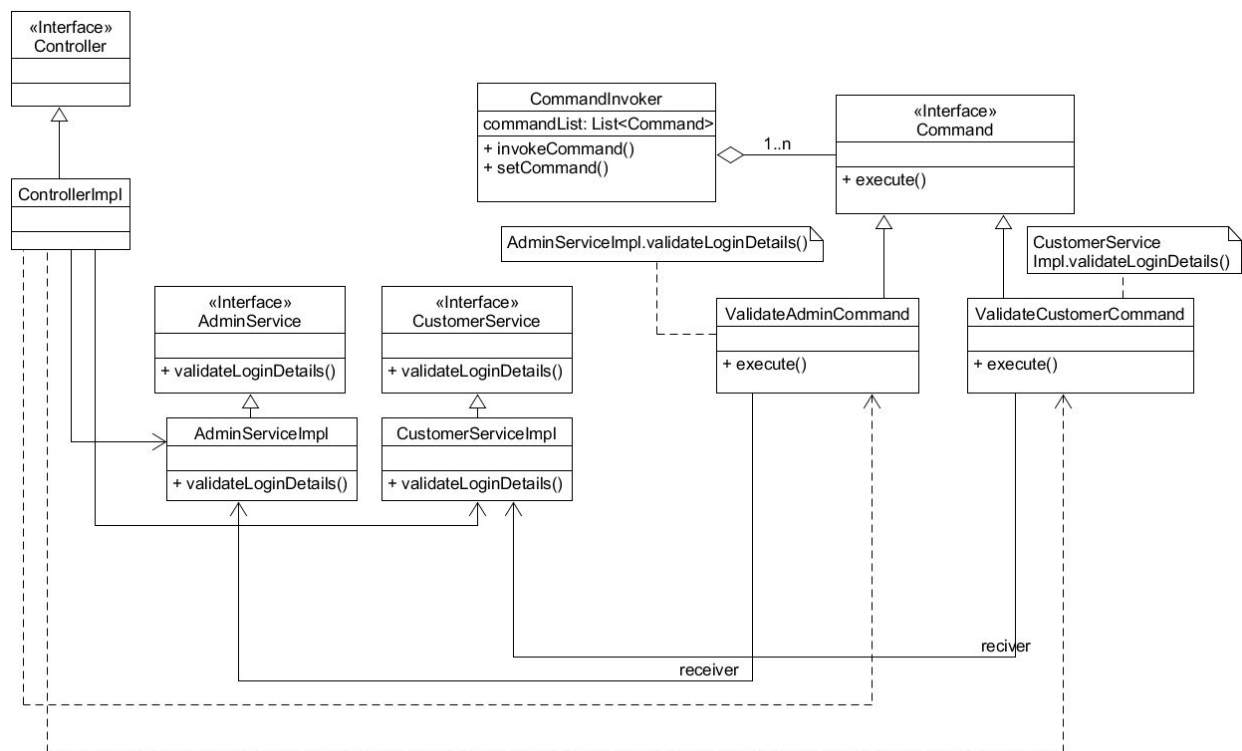
The reflection pattern, is providing a mirror image of the internal layers. The pattern dynamically decides what logic to implement. In this case, it decides depending upon the call, where to allow it or to block it with an invalid authorization exception.

UML Sequence Diagram

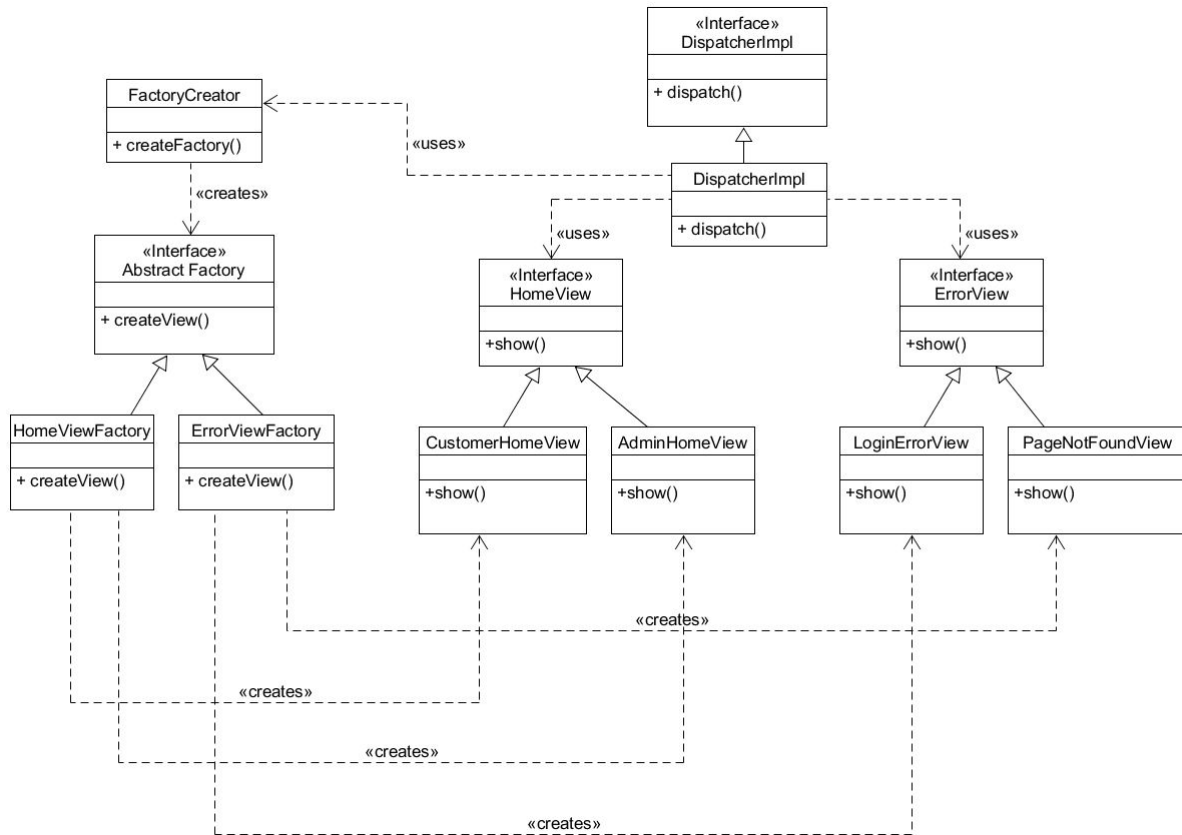


Other diagrams such as class diagram and domain model diagram are still valid.

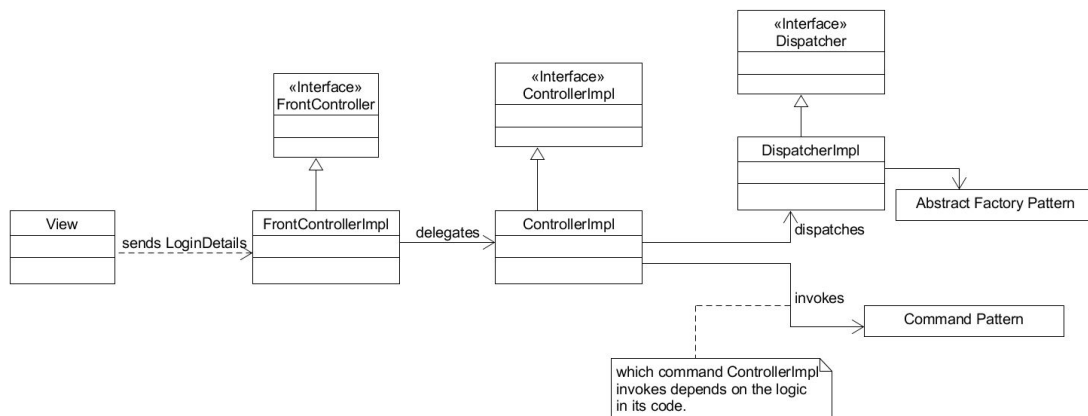
UML Class Diagram for Command Pattern



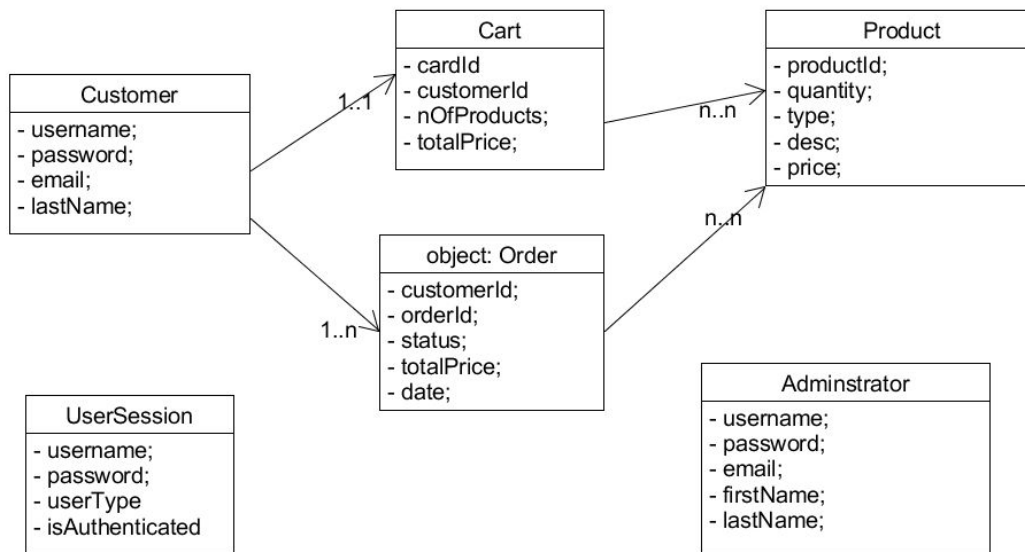
UML Diagram for Abstract Factory



UML Class Diagram for Front Controller



UML Domain Model (Updated)



Transition From Assignment 2 to Assignment 3

New Classes Created

Changed the name of LoginDetails class to UserSession (to better communicate its purpose).

Created an exception class AuthorizationException, which is executed when a user comes out of his role.

RequiresRole.java for creating the annotation which enforces authorization for using those methods.

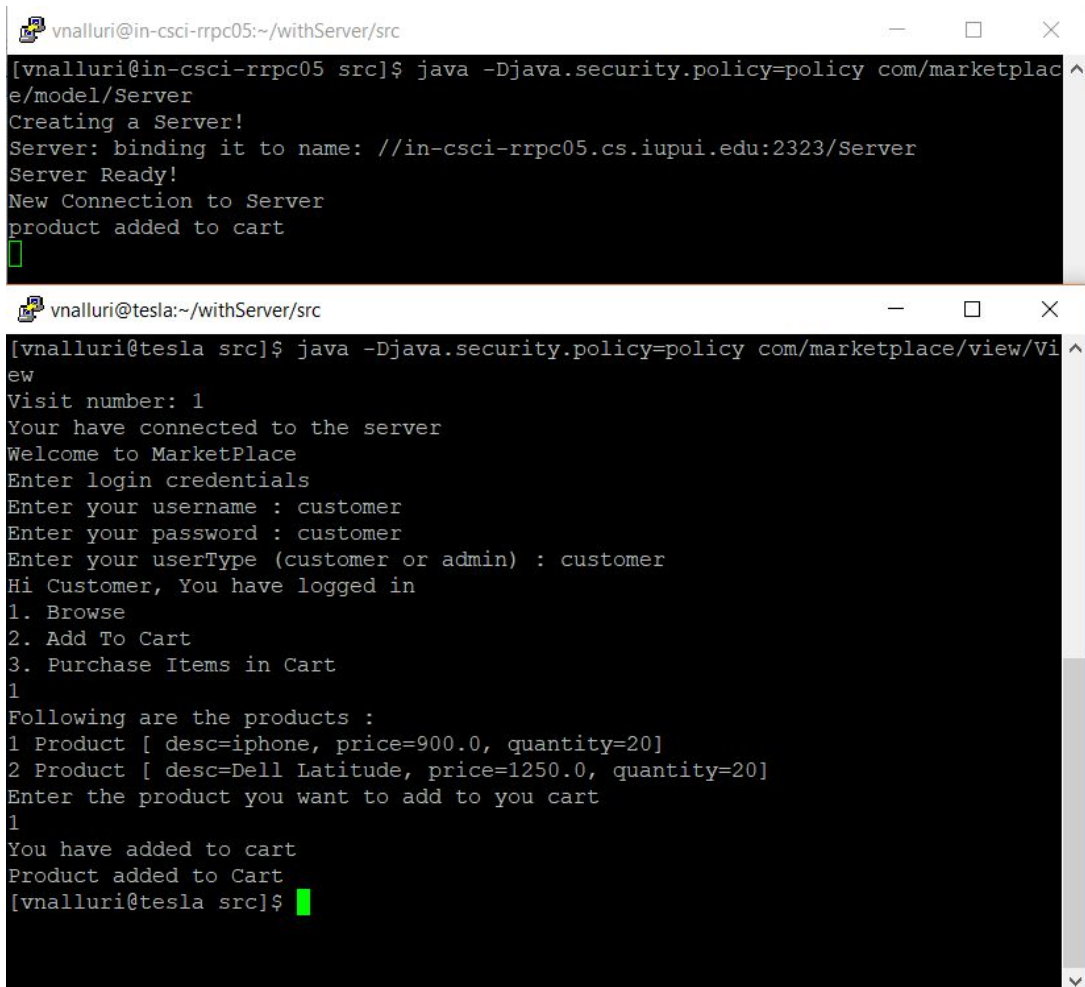
Server.java renamed the file ModelImpl to better communicate its purpose.

New class ModelImpl which has all the methods/funtionalities provided by the project to its users.

Comments from Assignment 2

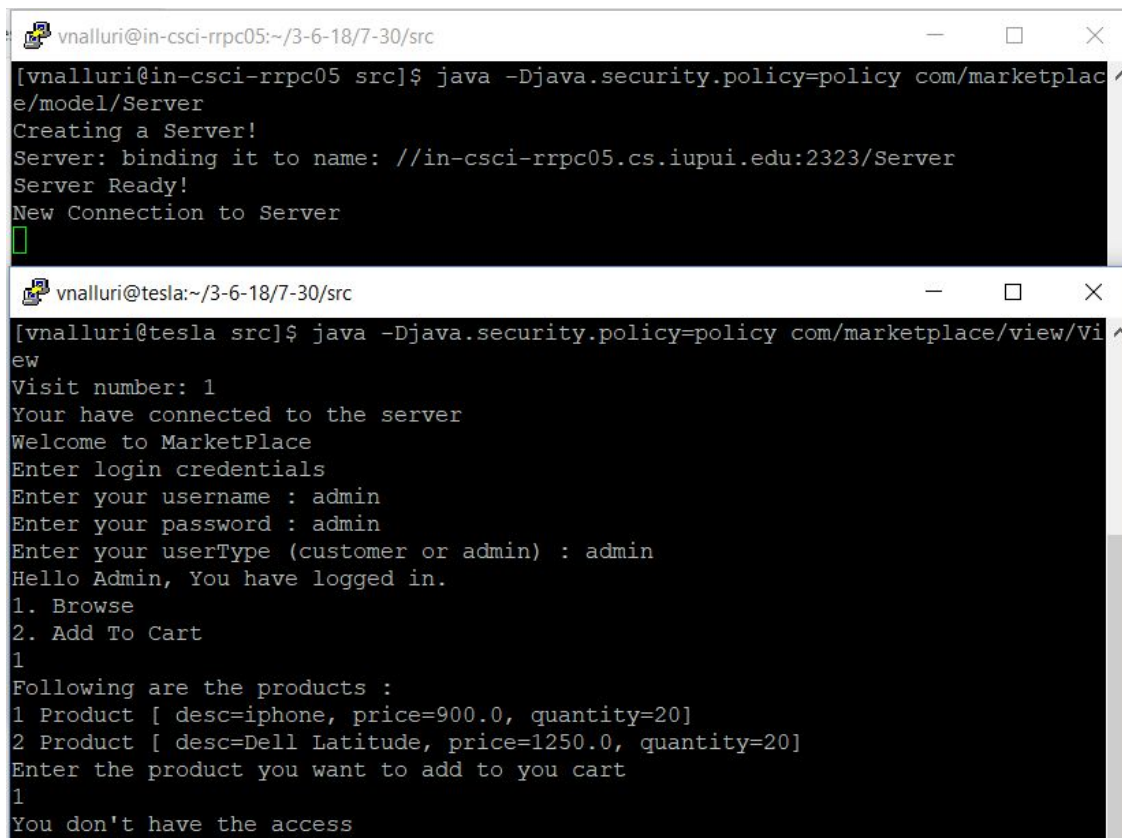
Few classes were missing honor pledge and necessary comments. They have been addressed this time.

Sample Runs



```
vnulluri@in-csci-rrpc05:~/withServer/src
[vnulluri@in-csci-rrpc05 src]$ java -Djava.security.policy=policy com/marketplace/Server
Creating a Server!
Server: binding it to name: //in-csci-rrpc05.cs.iupui.edu:2323/Server
Server Ready!
New Connection to Server
product added to cart
█

vnulluri@tesla:~/withServer/src
[vnulluri@tesla src]$ java -Djava.security.policy=policy com/marketplace/view/View
Visit number: 1
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
1. Browse
2. Add To Cart
3. Purchase Items in Cart
1
Following are the products :
1 Product [ desc=iphone, price=900.0, quantity=20]
2 Product [ desc=Dell Latitude, price=1250.0, quantity=20]
Enter the product you want to add to you cart
1
You have added to cart
Product added to Cart
[vnulluri@tesla src]$ █
```



The image shows two terminal windows. The top window is titled 'vnalluri@in-csci-rrpc05:~/3-6-18/7-30/src' and shows the execution of a Java command to start an RMI server. The output indicates the server is created and ready, and a new connection is established. The bottom window is titled 'vnalluri@tesla:~/3-6-18/7-30/src' and shows the execution of a Java command to start an RMI client. The output shows the client connecting to the server, logging in as 'admin', and viewing a list of products. The client then attempts to add a product to the cart but receives an 'Access Denied' message.

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


[vnalluri@tesla src]$ java -Djava.security.policy=policy com/marketplace/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.
1. Browse
2. Add To Cart
1
Following are the products :
1 Product [ desc=iphone, price=900.0, quantity=20]
2 Product [ desc=Dell Latitude, price=1250.0, quantity=20]
Enter the product you want to add to you cart
1
You don't have the access
```

Conclusion And Analysis

With this assignment, using design patterns such as reflection pattern and proxy pattern a reflection on the ModelImpl is provided for the front controller to interact. Also only a user role with appropriate role can communicate with the ModelImpl method. This is implemented using Authorization pattern using Java Annotations to provide role based access control.

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, WinSep, PUTTY, Tesla, Java

