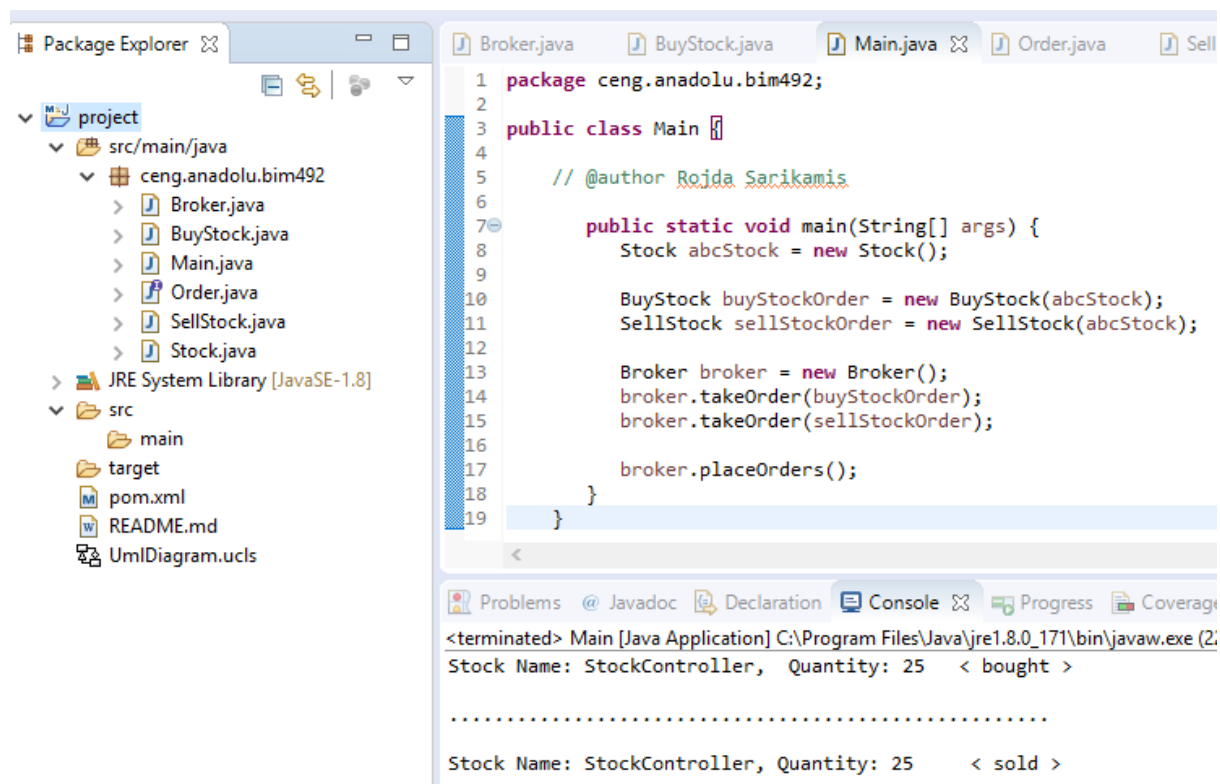


Statement of Work:

In the command design pattern, the process and the triggering of the process are separated. So I chose this design pattern. In the event that there are more than one objects to be stocked, we will be able to execute them in order. And I can use the same stock transaction objects in more than one place in the application. I have created an interface Order which is acting as a command. I have created a Stock class which acts as a request. I have concrete command classes BuyStock and SellStock implementing Order interface which will do actual command processing. A class Broker is created which acts as an invoker object. It can take and place orders.

Broker object uses command pattern to identify which object will execute which command based on the type of command. Main.java our main class, will use Broker class to demonstrate command pattern.



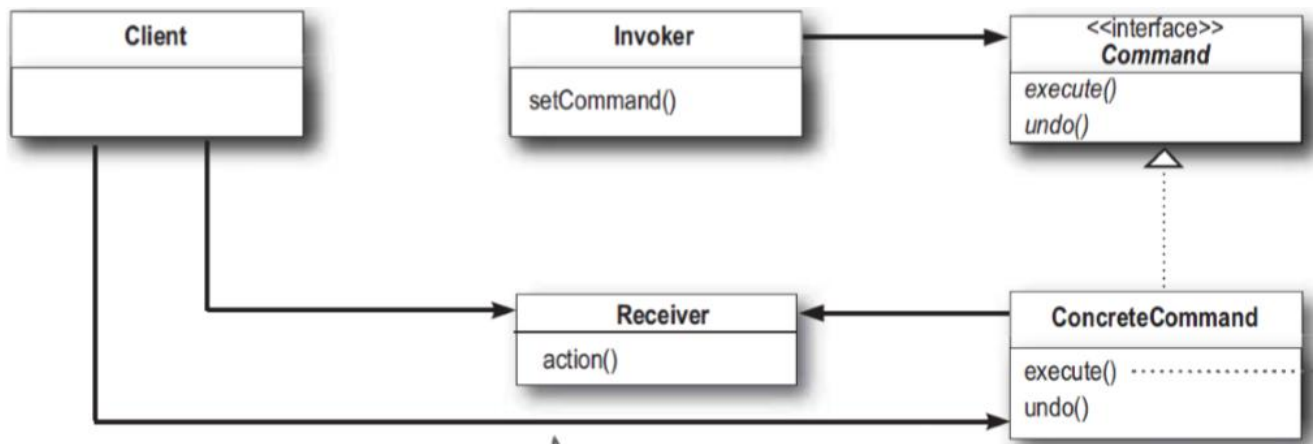
```
1 package ceng.anadolu.bim492;
2
3 public class Main {
4
5     // @author Rojda Sarikamis
6
7     public static void main(String[] args) {
8         Stock abcStock = new Stock();
9
10        BuyStock buyStockOrder = new BuyStock(abcStock);
11        SellStock sellStockOrder = new SellStock(abcStock);
12
13        Broker broker = new Broker();
14        broker.takeOrder(buyStockOrder);
15        broker.takeOrder(sellStockOrder);
16
17        broker.placeOrders();
18    }
19 }
```

<terminated> Main [Java Application] C:\Program Files\Java\jre1.8.0_171\bin\javaw.exe (2
Stock Name: StockController, Quantity: 25 < bought >
.....
Stock Name: StockController, Quantity: 25 < sold >

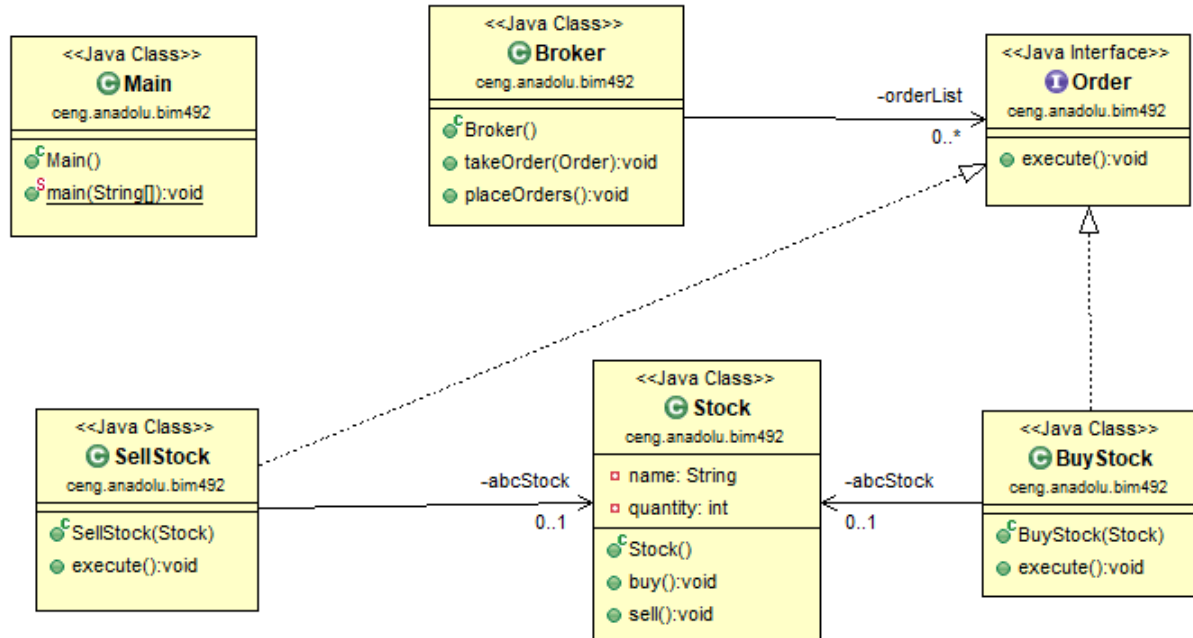
Design Pattern:

I choose The Command Pattern. Command pattern is a data driven design pattern and falls under behavioral pattern category. A request is wrapped under an object as command and passed to invoker object. Invoker object looks for the appropriate object which can handle this command and passes the command to the corresponding object which executes the command. In object-oriented programming, the command pattern is a behavioral design pattern in which an object is used to encapsulate all information needed to perform an action or trigger an event at a later time. This information includes the method name, the object that owns the method and values for the method parameters.

A command object encapsulates a request by binding together a set of actions on a specific receiver. The Command Pattern decouples an object, making a request from the one that knows how to perform it.



UML:



Also I have the file "UmlDiagram.ucls" I Draw a detailed class diagram using UML Diagram Drawing Tool. You can look at uml in the program.

ROJDA SARİKAMIŞ

14888364024