# Command Pattern

## Homework number 6 – Command Pattern

### Authors: Bilger Yahov – 2442450

### Hua Tao – 2346230

Description of the assignment:

The assignment is implemented in a very simple way. The main idea is clearly shown in the implementation.

First we created a command interface. It is called “Order”. There is one method inside called “Execute”.

The second step is to create a so-called request class. The request class in our case is called “Stock”. Inside there are two methods “Buy” and “Sell”.

After that we create two concrete classes implementing the “Order” interface. They are called “BuyStock” and “SellStock”. Respectively in their “Execute” methods” buying or selling the stock is performed.

Then we create a class command invoker. In our case it is called “Broker”. It has a list with orders. Two methods - “TakeOrder” and “PlaceOrders”. With the first one we add a new order, with the second one we execute all the orders in the list.

The last step is to test the pattern. In the “Main” method of the application we create a new stock, then we create two orders which are instances of “BuyStock” and “SellStock”. Then we create one object of class “Broker” and take these two before created orders. Then we execute “PlaceOrders” method of the broker and the results are clearly shown.

Some explanations about the pattern:

Four terms always associated with the command pattern are command, receiver, invoker and client. A command object knows about receiver and invokes a method of the receiver. Values for parameters of the receiver method are stored in the command. The receiver then does the work. An invoker object knows how to execute a command, and optionally does bookkeeping about the command execution. The invoker does not know anything about a concrete command, it knows only about command interface. Both an invoker object and several command objects are held by a client object. The client decides which commands to execute at which points. To execute a command, it passes the command object to the invoker object.

Attached there is also a picture of the class diagram. Everything is clearly specified on it.