SE1-A8

Javad Kavian, Mehdi Jamalkhah
810100103, 810100111

June 2024

Step 1 & 2:

Table 1: Refactoring Details

SHA Contains the Bad Smell	**Brief Description of Bad Smell**	**Refactored SHA**	**Comments**
bb11620	control classes should be packed in a different package.	80e9ba1	
80e9ba1	Credit of stop limit order is checked in matcher which is not its responsibility.	d7e9965	Move this checking to credit control.
80e9ba1	Whether the stop-limit order is activatable is checked in the matcher, which is not its responsibility.	28e39e4	Create a new control for checking activation.
28e39e4	Credit of buy order is increased after opening market in matcher, which is not its responsibility.	dfe54db	Create a new function, called `marketOpenned`, for `MatchingControl`.
dfe54db	Ugly chain of if statement on same variable.	e19ccd0	Replace it with switch case.
e19ccd0	Enqueue activated orders during auction state in `activateOrders` function, but it's a part of execution. Also activate some orders in `executeActivatedOrders` function which should be done in mentioned function.	a8f9217	Separate this functionalities.

Table 1(cont.): Refactoring Details

SHA Contains the Bad Smell	**Brief Description of Bad Smell**	**Refactored SHA**	**Comments**
e19ccd0	Increase credit of activated order in `activateOrders`, which is not its responsibility.	59c9c58	Move this to credit control.
fb0e162	Passing `enterOrderRq` to `newOrder` function, which make a coupling between these classes.	6efcedb	Create a new order in `OrderHandler` and pass it to `newOrder`
6efcedb	Validation of requests is performed in both the `OrderHandler` and the `Security`.	e84b20b	Move all of them to `OrderHandler`
f5b06d1	All fields of `Broker` class has getter tag.	dfc10c7	Put getter tag for the class.
52869a5	Increase credit then cancel it in an if statement.	52869a5	Reorder checking conditions.
52869a5	Duplicate initializing the order book in security test.	da70428	Extract method.
c36f04e	OrderHandler has nothing to do with the object creation of order.Using **Factory method** design pattern comes in handy here	08dfdf9	Use OrderFactory object to create order from request
865441e	Long parameter list for `validate`	be4ef35	Replace repository parameters with an object
d386552	Handling credit managing and check position in `updateOrder`.	e9b029f	Extract different methods that observe the `updateOrder`.
1a394d3	Calculating `losesPriority` gets all its parameters from the order object, which suggests that it would be better to move this functionality directly into the order class.	1a394d3	Move `losesPriority` to order and override it in `IcebergOrder`, which remove the ugly `instanseof`.
ace2f21	Method `getOpeningPrice` is not readable.	0d27991	Use dictionary for storing values that should be compared.

Table 1(cont.): Refactoring Details

SHA Contains the Bad Smell	**Brief Description of Bad Smell**	**Refactored SHA**	**Comments**
d386552	Duplication in orderbook setup	e83749e	By using multiple setUpOrderBooks, we avoided duplication as much as possible

Step 3:

Validation Interface

The order handler has a lot of different validation logic, resulting in long and complex methods that are hard to understand completely. A better approach is to group related validation checks into separate functions to break down these long methods.

One way to do this is by using the Observer design pattern. This involves an interface that observes the handling of requests, and at the beginning of process will be called. Different validation classes(such as stop price validation, or security validation) can then overload the necessary methods of this interface, each handling a group of related validations.
Final SHA of this refactoring is 7c1e96e.

Publisher Interface

Along with handling requests, there is a lot of event publishing, and they are in different functions, which makes it hard to keep track of them and know which event is publishing in which situation. On the other hand, these publishs are not related to the duties of the functions that publish them; they are just like some logs.

To solve this problem, we can use the observer design pattern and create different classes for each event, in order to publish only that event. All of these classes will implement the desired method of an interface called `Publisher`.
Final SHA of this refactoring is 6699552.