Stock Cart Case Explanation

The app basically lets you make CRUD operations for Stock Carts. It uses layered architecture to make the code modular. That way if you want to change the database source for the project you won't need to change the software totally. You will only need to make changes on the Data layer. The code basically has 4 layers which are Entity, Data, Business & UI.

Entity Layer

This layer is where all entities are stored. The only entity created for this case is StockCart. StockCart consists of stockCode/stockName/unit/barcode/taxType/description/creationDate. There are annotations for these variables(fields) to determine the restrictions in the database. For example, stockCode is a column with stock_code name. Its length is 50 and it cannot be null. This entity will let us use the data we get from database as objects in code, which is one of the reasons Hibernate is used as an ORM (Object Relational Mapping) tool.

Data Layer

Data layer basically lets us connect and do the CRUD operations for out project. In the project the data layer is created with the intention of connecting mySQL. This layer has two sections which are Abstract and Concrete.

The Abstract section has interfaces that has basic methods. There is a IBaseRepositor, which has the intention to be the base interface for all others and has base operation methods in it. The other interface is IStockRepository, and that inherits from IBaseRepository. Abstract section lets us make more flexible options for using different concrete classes, which will be more clear in Command section.

The Concrete section on the other side has the concrete methods that are filled with basic operations. They all use Sessions. They are opened on each operation and closed when the operation is done. One of the main reasons for it is, that continuous connection with database can cause database security problems.

Each Session used in the Concrete section needs SessionFactory to work. So, we create an instance for it. But it uses lots of memory when it’s created, because it uses a file to read the commands and execute them each time. As a result, it’s more sensible to use Singleton Pattern to prevent it from causing ant memory issues. The SessionFactory we initiated adds annotated class which is the Entity class (only StockCart class in our case) as well.

Business Layer

Business Layer is a bridge between Data and UI Layers. Its main purpose is to prevent the UI Layer directly using database related structures. This will make the Data Layer changes more possible. Users won’t be able to manipulate the data from database directly as well (there will be one more layer to prevent harmful intentions of some users).

UI01 Layer

This layer consists of Commands, Controller and View sections. It’s the layer where user directly interacts with. View section has visual parts of the project. This part is where the user sees the results of their interactions. It doesn’t handle any operations.

Controller part is the area that get what users want and operates according to it. For this project it has two classes. StockCartFrameController is the main controller for the UI. It sets every event and

sets the UI visible. On the other hand, GeneralStockCartOperations class has all methods that needs to be used in the events that are set. These methods can be called form the class to manipulate the data and they used in commands for that reason.

Commands are the last section that needs to be mentioned. Commands are the methods that are objectified. Which means we can keep track of the used methods. Furthermore, they're undoable if you used/created unexecute method for each command (not in this project.). Command Pattern basically needs abstract commands, concrete commands, invoker, and receiver. For the project there is only one abstract command, which lets us store in one place. The project has to invokers which are GeneralAction and GeneralSelection. Reciever for the commands on the other hand is StockCartFrameController. It receives from GeneralAction and GeneralSelection then uses related Command.

Lastly Launcher class is the that starts the project. Its in the outside of the all Layers.