Design Patterns with Java (5041.24)

Assignment 01 – Strategy & Observer

Deadline 17. Jan. 2025 23:59:59

01: Refactoring Deck Storage to Strategy

Description

In this assignment you will be refactor an existing implementation to the Strategy pattern.

Download the Starter Code from the Assignment01.zip the starter code is **StoringCards.**

The present version of **Deck** has been extended with methods for loading and saving the state of a Deck object by means of either binary or JSON files in the methods

```
class Deck implements Iterable<Card> {
    private List<Card> cards;

    public void load(StorageFormat format) throws IOException {
        ...
    }

    public void save(StorageFormat format) throws IOException {
        ...
    }
}
```

The **Deck** class now contains a multitude of bad aspects, e.g.

- The class has a more than one responsibility.
- The **load()** and **save()** methods both contain (de)serialization details which has no justification to be located within Deck itself.
- The **Deck** class now needs to change when a new storage format is introduced.
- There is a very high degree of coupling between unrelated parts.
- Also, it is possible to load a **Deck** object from a binary file and saving it as JSON (and vice versa).

Task

Your task is to alleviate the above deficiencies by refactoring the existing code for storage mechanisms.

1. Refactor the existing code to employ the Strategy pattern
    a. Fix the storage strategy at construction time to prevent mixing storage strategies.
    b. There is **no need** to spend time on additional error handling.
2. Test your refactoring by modifying the setup code in Main.java.
3. Using DrawIO create a class diagram that shows the classes in the project and how they are associated. Try and show where the strategy pattern is applied. Use the Cheate Sheet in moodel for guideance.

02: My awsome stockmarket

Description

In this assignment you will be implementing a simulation of a stock market.

Download the Starter Code from the Assignment01.zip the starter code is **StockMarket.**

Task

1. Your task is to implement the **StockMarket** and the **StockObserver** using **The Observer Pattern**. You can **NOT** change the main method.
   The expected output when running the main method should be:

```java
public static void main(String[] args) {
    // Create the stock market
    StockMarket stockMarket = new StockMarket();

    // Create StockObservers that will sell when
values is above threshold
    StockObserver stockObserver1 = new
StockObserver(stockMarket, "MSFT", 87.5);
    StockObserver stockObserver2 = new
StockObserver(stockMarket, "FB", 177);

    // Register Observers
    stockMarket.registerObserver(stockObserver1);
    stockMarket.registerObserver(stockObserver2);

    // Simulate some trades
    stockMarket.tradeStock("MSFT", 87.1);
    stockMarket.tradeStock("FB", 176.1);
    stockMarket.tradeStock("GOOG", 1234.57);
    stockMarket.tradeStock("MSFT", 87.4);
    stockMarket.tradeStock("FB", 177.0);
    stockMarket.tradeStock("MSFT", 87.6);
    stockMarket.tradeStock("FB", 176.9);
    stockMarket.tradeStock("MSFT", 87.4);
    stockMarket.tradeStock("FB", 177.2);
    stockMarket.tradeStock("MSFT", 87.7);
    stockMarket.tradeStock("FB", 176.2);
    stockMarket.tradeStock("FB", 177.9);
    stockMarket.tradeStock("GOOG", 1234.58);
}
```

```
StockMarket - MSFT traded at USD 87.1
StockMarket - FB traded at USD 176.1
StockMarket - GOOG traded at USD 1234.57
StockMarket - MSFT traded at USD 87.4
StockMarket - FB traded at USD 177.0
StockMarket - MSFT traded at USD 87.6
StockObserver - Selling MSFT position
StockMarket - FB traded at USD 176.9
StockMarket - MSFT traded at USD 87.4
StockMarket - FB traded at USD 177.2
StockObserver - Selling FB position
StockMarket - MSFT traded at USD 87.7
StockMarket - FB traded at USD 176.2
StockMarket - FB traded at USD 177.9
StockMarket - GOOG traded at USD 1234.58
```

2. Using DrawIO create a class diagram that shows the classes in the project and how they are associated. Try and show where the observer pattern is applied. Use the Cheate Sheet in moodel for guideance.

Submission

The assignment solution must be submitted (pushed) in your assigned student repo in the folder Assignments/Assignment**{Number}**, which you will find in the course's Bitbucket Team.

The program code must contain a functional program with no compilation errors. The codes should be well structured, understandable and easy to read which include:

- Follow naming conventions. (e.g., descriptive names for class, method, variable and others).
- Use Clear and Consistent Formatting (e.g., correct indentations, spacing and bracket placement)
- Be as self documenting as possible, so only add comments if needed. No need to add java doc for every class and method.

Perform a formal submission in Moodle, via the intended submission box, by submitting a text file with your repo url in it!

NB! Changes made in the repo after the deadline will not be taken into consideration. Please contact the teach if submission deadline has elapsed.