Design Document: Stock Market System

Introduction

The Stock Market System is a java swing application that allows customers to buy and sell stocks, while also providing managers with tools to manage customer portfolios and monitor and manage stock prices. The system will be built using several design patterns, including Factory, MVC, Composite, Proxy, and Observer patterns, to ensure scalability, maintainability, and security.

Factory Design Pattern

The Factory design pattern will be used to create instances of four main classes: Request, Stock, Customer, and Account. (See in Model folder: RequestFactory.java, StockFactory.java as well as CustomerLogin.java) Each of these classes will create new instances of the class mentioned before. This will make it easier to manage and organize the creation of these objects.

MVC Design Pattern

The MVC design pattern will be used to separate the concerns of the application into three distinct layers: Model, View, and Controller.

a) Model: The Model layer will contain the backend model logic that will interact with the database. This layer will be responsible for retrieving and updating data from the database.

b) Controller: The Controller layer will connect the backend model logic and send data to the frontend Java Swing objects. This layer will handle user input and manage the flow of data between the Model and View layers.

c) View: The View layer will use the API provided by the Controller to retrieve data and display it on the Java Swing window. This layer will handle the graphical user interface and user interaction.

Composite Design Pattern

The Composite design pattern will be used to manage multiple java objects in the Controller layer. The ManagerPortfolioSystem.java, CustomerTradingAccountSystem.java, CustomerPersonalAccountSystem.java, and CustomerLogin.java files will take multiple Java objects from the Model folder. This will make it easier to manage the different objects in the system and keep the code organized.

Observer Design Pattern

The Observer design pattern will be used to manage changes to the Transaction, Customer, Request, and other objects in the backend. The backend will hold observers for these objects, which will be notified when changes occur. The frontend will be event-driven, with Java Swing objects responding to these notifications and updating the display accordingly.

Proxy Model

The Proxy design pattern will be used to provide an additional layer of security to the Stock Market System. (See: SMProxy.java)

a) Customer/Manager Proxy: Customers and managers will need to access a proxy to perform operations on the stock market. This proxy will authenticate the user and verify that they have the necessary permissions to access the requested resources.

b) Manager Authentication: Managers will need an extra key to validate their access and manage the stock market. This key will be required to access sensitive areas of the application, such as managing customer portfolios and monitoring stock prices.

Conclusion

The Stock Market System will be designed using several design patterns, including Factory, MVC, Composite, Observer, and Proxy patterns, to ensure a scalable, maintainable, and secure system. The application will provide customers with a user-friendly interface to buy and sell stocks, while also providing managers with the tools to manage customer portfolios and monitor stock prices.