Ryan Jung

Final Project Write-Up

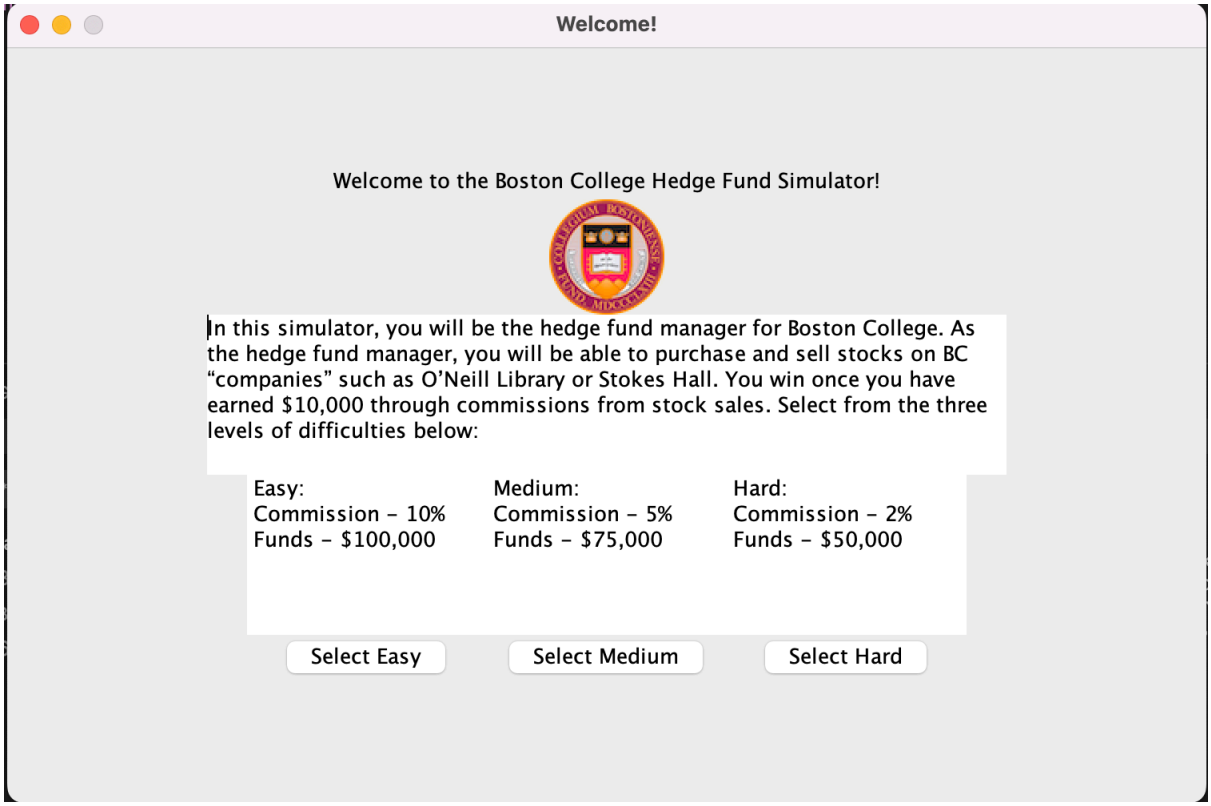
Motivation:

As someone who trades stocks, I wanted to create a project in Java to simulate the stock market, where the user would be able to trade stocks just like in real life.

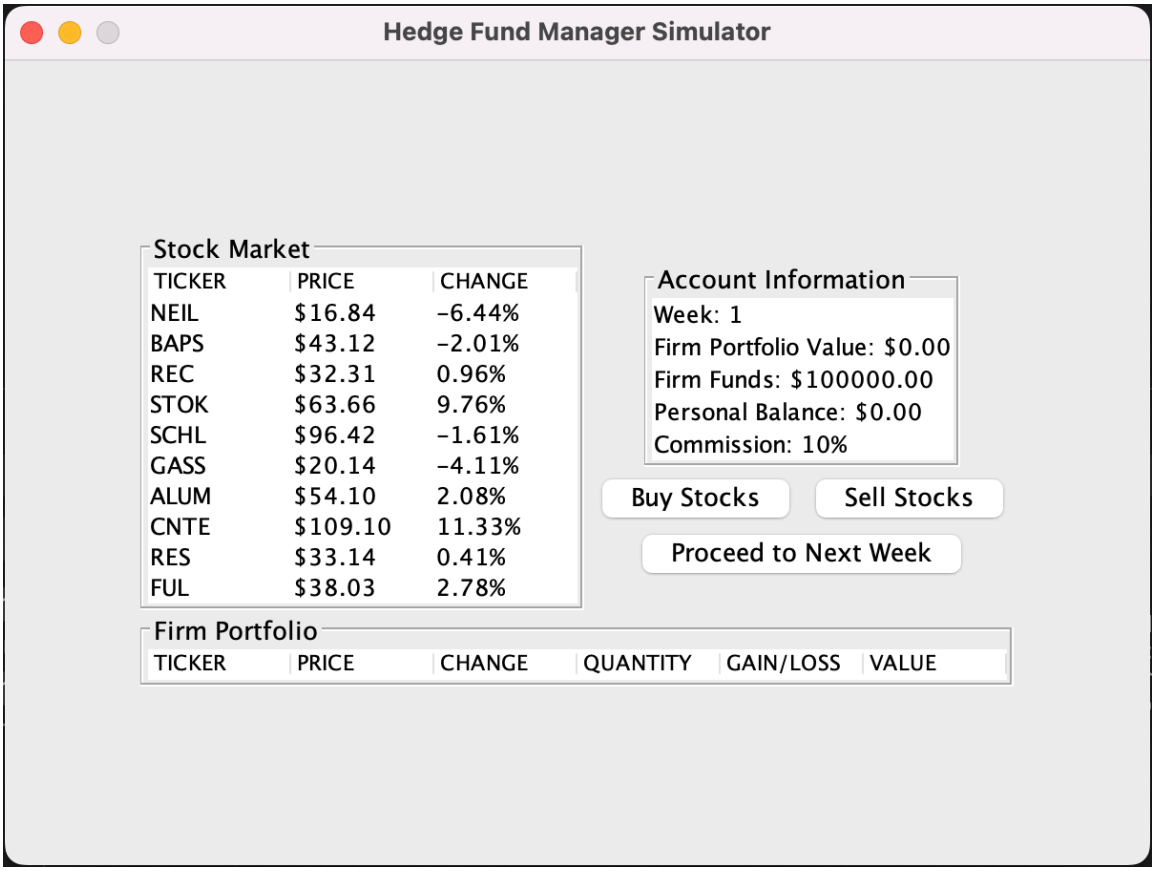
Description:

The project is a hedge fund manager simulator, where the user is the manager of Boston College's imaginary hedge fund. As the manager, the user is able to buy and sell stocks of BC's "businesses", such as Stokes Hall or O'Neill Library.

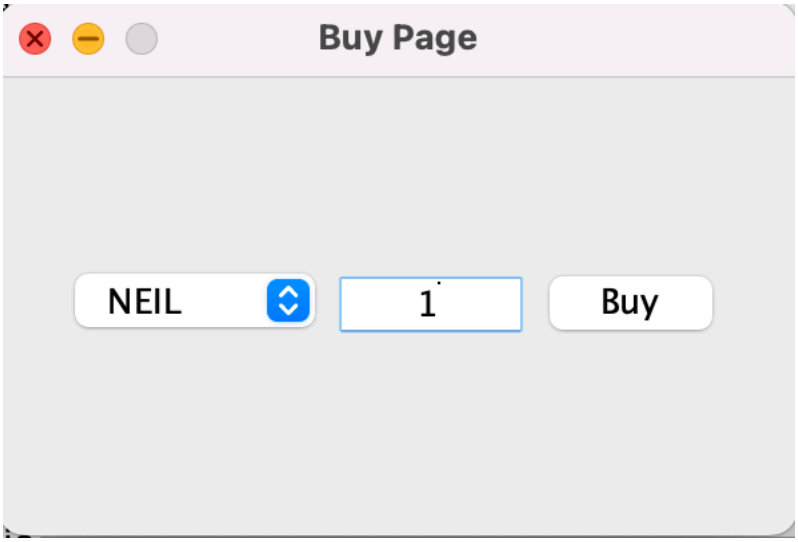
At the beginning of the program, the user is greeted by the start page with a description of the simulator along with the objective which is to make at least $10,000 through commissions from selling the stocks in the firm's portfolio. Below the description are three options for the three levels of difficulty for this simulator. Each level has differing levels of commission and starting funds which can drastically impact how quickly the user will be able to make $10,000 through commissions.



After selecting a difficulty, the user is greeted by the main interface for managing and trading stocks. All the stocks in the market are displayed in the table on the left, and all of the account information is displayed on the right. If the user wants to buy or sell a stock, they simply click the appropriate button to enter the transaction page.



Here, the user can specify which stock and how much they want to buy/sell. Trying to buy/sell an erroneous amount will result in error warnings.



Once the stocks have been bought/sold, the main interface will update after pressing “Proceed to Next Week”, which will prompt the stock prices to change as well. Additionally, the stocks in the portfolio will update at the bottom of the interface.

Lastly, once the user has accumulated at least $10,000 in commissions, they will have won the game!

To run the program:

1. `cd RyanJungFinal`
2. `javac Backend/*.java`
3. `javac Frontend/*.java`
4. `java Frontend/main`

Since I separated the code into two packages (Frontend and Backend), you will need to run the main.java file in the Frontend package from the outside RyanJungFinal package.

Upon successful completion, the start page should pop up.

Design Patterns:

I used the following design patterns for this project:

Singleton

Since I refreshed the stock market each week after the user pressed “Proceed to Next Week”, there could have been an issue that the stock market re-initialized each week with different prices, different changes, etc. In order to prevent this, I used the singleton creational pattern in order to avoid re-initializing the stock market if it already existed. I also used this same design with the account as well. This was to avoid re-initializing the account, which would have effectively wiped all the data on the account and started from the beginning.

Façade

In the beginning of the program, the user is able to select between three different difficulties. Rather than creating a separate class for each difficulty, I utilized the façade structural pattern to delegate each difficulty to its own method call in a DifficultyFacade class. Depending on what level of difficulty the user had selected, the façade would then initialize the starting funds along with the commission for the account.

Iterator

Throughout this program, there were multiple instances where I needed to iterate through stocks in order to update/display information. For example, I needed to iterate through the user's portfolio in order to display the stocks at the bottom of the main interface. To make this easier, I utilized the iterator behavioral pattern in order to streamline iteration. I used this also for displaying all the stocks in the stock market on the left side of the interface as well.

UML Diagram For Facade (Only instance of inheritance):