# Group 1

# Kuen Wai Chan, Ben Hubbard, Fraser Steel, Elena Xiao

# Initial GUI and Back-end API Design

In this program users will be able to track the value of stock portfolios using a graphical interface. The users can see the current stock price of the stocks they current own and the overall value of all their holdings.

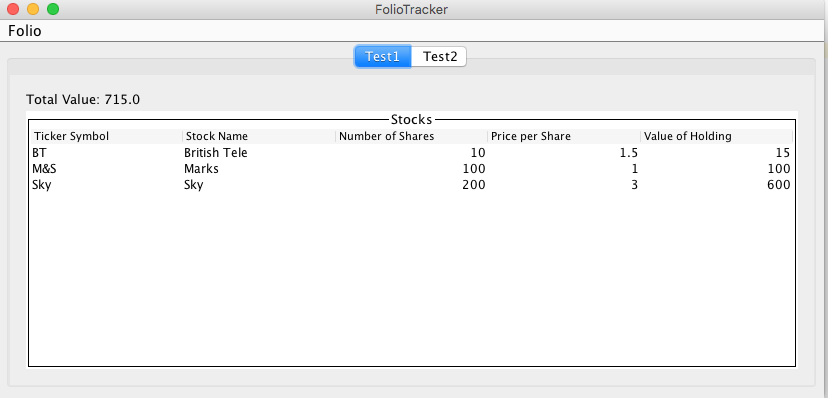


Figure 1 - Overall Graphical User Interface

Here it can be seen how a user may open another stock. After this pop up the user will be asked to enter how many of the stock they would like to purchase.

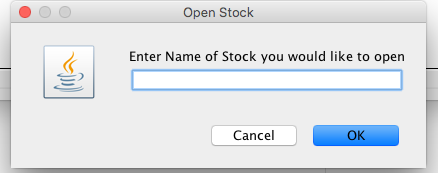


Figure 2 - Pop up for opening a new stock

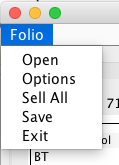


Figure 3 - Drop down options menu

# UML Diagram

# Specification of API

Role of Interfaces

IPortfolio : Outlines the functionality of portfolio. Methods may include changing portfolio name, opening new stocks, buying more of one type of stocks and selling stocks.

IPortfolioTracker: Outlines the functionality of portfolioTracker. Methods may include retrieving folios by name, saving portfolios and loading from file and also removing portfolios.

IStock: Outlines the functionality of stock. This methods this may include could be buying, removing and returning value of a stock.

Role of Classes

Stock: A concrete implementation of the IStock interface. Holds most of the information about a specific stock, such as name, number of shares, price per share and value of holding.

Portfolio: A concrete implementation of the IPortfolio interface. Is used to hold stocks. Stocks are held within Map with their ticket name being used to map to a Stock object.

PortfolioTracker: A concrete implementation of the IPortfolioTracker interface. This class is used to save and load portfolio’s. An arraylist of type portfolio’s is to store each one.

Prices: This class Is used to store prices of stocks. It stores history of prices in Hashmap and stores names that need to be known in an Arraylist.

StockListener : Used to communicate with the GUI and model. This is the controller part of MVC. The GUI tells the listener what it would like it to do and the listener performs the desired method. Methods may include things like buy, sell, rename.

PortfolioListener : Used to communicate with the GUI and the model. Methods may include Save portfolios, sell all, and open new stock.

Relationship between interfaces and classes

Method descriptions

Portfolio

* buyStock(String, int) – Buys a new stock with a given Ticket symbol and number of shares required
* createStock(String, int) – creates another share with the given ticket symbol and number of desired shares.
* getPortfolioName() – returns the name of the portfolio.

- getStocks() – returns the list of shares in the stockMap

- getTotalValue() – returns the total value of the portfolio

- removeStock(String) - removes a specific stock from the portfolio with a given ticket symbol.

- setPortfolioName(String) – Changes the name of the portfolio with the given name.

PortfolioTracker

* deletePortfolio(Portfolio) – removes given portfolio.
* getPortfolioList() – returns list of portfolios
* getPortfolioByName(String) – returns a portfolio with a given name
* getPortfolios() –
* getStockList() – returns list of stocks within a portfolio
* loadPortfolioFromFile() – loads portfolios from a file.
* savePortfolios() – saves the data to a file.

Prices

* getPriceOfTicker(String) – returns the most recent prices of a given stock.
* refresh() – refreshes prices of stocks.

Stock

* deleteStock(Stock) – deletes the stock passed into the method.
* getNumShares() – returns the number of shares for a given stock.
* getPricePerShare() – returns price per share of a stock.
* getStockName() – returns name of the stock.
* getTicketSymbol() – returns ticket symbol of the stock.
* getValueOfHolding() – returns price per share multiplied by number of shares.
* setNumShares(int) – modifies the number of shares of the stock.
* setPricePerShare(double) – changes the price per share of a stock.
* setStockName(String) – changes the name of the stock
* setValueOfHolding(double) – sets the value of the holding.

StockListener

* actionPerformed(ActionEvent) – Performs the action that the GUI asks for the program to perform. Case statement is used to determine which task the GUI is hoping to take place.

PortfolioListener

* actionPerformed(ActionEvent) - Performs the action that the GUI asks for the program to perform. Case statement is used to determine which task the GUI is hoping to take place.