I am using the MVC design pattern as the core of the application. I also use the Observer pattern within the tracker which notifies the GUI of changes happening in the Model. In the GUI the user can add, rename, open, close, save and delete a 'Folio'. The user can also search and refresh and switch between their folios via a drop down menu or the opened tabs.

An example of the use of the Observer patterns is the "Last refreshed:" message at the bottom of the GUI, indicating to the user when the page was last refreshed.

Once a folio has been created, you can add stock and shares using the 'Add Stock' button. This then adds it to the folio table and calculates the price per share and value of the holding. It is also possible to sell stocks, in which case the number of shares will decrease by however many the user chooses to sell. This in turn will change the price per share and value of the holding accordingly. The GUI accesses the backend through an API which uses 3 interfaces - **IFolioTracker**, **IStock** and **IFolio**.

**IFolioTracker.java**

public String getFolioTrackerName();

effects: return the name of the FolioTracker

public void setFolioTrackerName(String folioTrackerName);

effects: changes name of FolioTracker

modifies: this

public boolean addFolio(IFolio portfolio);

requires: portfolio != null

effects: if IFolio successfully added return true else return false

modifies: this

public boolean renameFolio(String oldName, String newName);

effects: if folio exists with oldName, change name and return true, else return false.

modifies: this

public boolean deleteFolio(String folioName);

effects: if folio with folioName exists, delete it and return true, else return false.

modifies: this

public IFolio getFolio(String portfolioName);

effects: if folio with name portfolioName exists return the folio, else return null.

public List<IFolio> getFolios();

effects: returns a list of all folios

IStock.java

public String getTicker();

effects: returns the ticker

public String getStockName();

effects: returns the stock name

public int getNumberOfShares();

effect: returns the number of shares

public double getPricePerShare();

effect: returns the price per share

public double getValueOfHolding();

effect: returns the total value of the holding

public void setTicker(String ticker);

effects: sets ticker to ticker

modifies: this

public void setStockName(String name);

effects: sets stock name to name

modifies: this

public void setNumberOfShares(int value);

effects: sets number of shares to value

modifies: this

public void setPricePerShare(double value);

effects: sets price per share to value

modifies: this

public Change getPriceChange();

effects: gets price change (type enum Change)

public void setPriceChange(Change priceChange);

effects: sets the price change to priceChange

modifies: this

**IFolio.java**

public String getFolioName();

effects: returns the name of the folio

public List<IStock> getFolioStocks();

effects: returns a list that contains all the stocks stored in the folio

public double getFolioTotalValue();

effects: returns the total value of the folio

public void setFolioName(String name);

modifies: this

public boolean addStock(IStock stock);

effects: if stock exists in the folio, the quantity is increased and true is returned. If stock is not in the folio and it is added successfully, true is returned. Otherwise return false.

modifies: this, Stock

public boolean deleteStock(String stockTicker);

effects: if stock exists with ticket stockTicker, delete it and return true, else return false

modifies: this

public IStock getStock(String stockTicker);

effects: if stock with ticker stockTicker exists return the Stock else return null

**Extra Features:**

* File saving and loading using file explorer.
* Automatic refresh option.
* Sorting table by any of the columns.
* Showing if share price increased/decreased/remained same on refresh.
* Last Refreshed label to show when the stock prices were last checked.