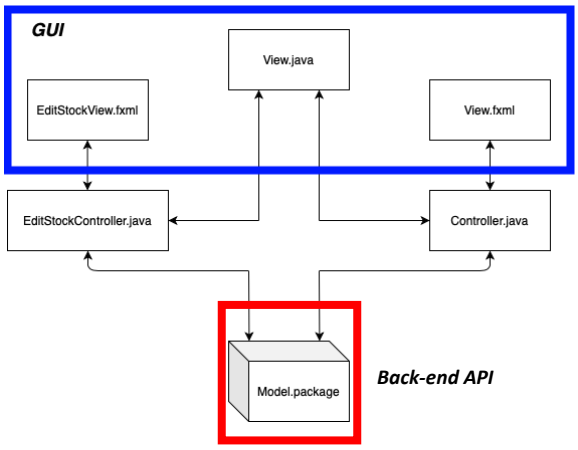
CS308: Specification of API

Group 27: Ross Lyons (201724258), Syeda Ayela Gilani (201810821), Karl Rivett (201727785), Ismael Ahmed (201707978), Koni Watson (201710203)

API Overview-

This API was designed with a GUI in mind from the beginning.
The API allows for a user to obtain real stock information through the StrathQuoteServer. From the server the stock information is formatted through the methods within the Stock class. The Stock class being at the bottom of the API structure. At the top of the structure is the FolioHolder class, which from this access to all the other classes can be obtained. These other classes being Folio, Holding and Stock. Each having their own respective interface in which they obey by – FolioHolderInterface, FolioInterface, HoldingInterface and StockInteface.
To initialise the use of the API correctly an instance of a FolioHolder object is needed. This therefore allowing access to the connected classes. Upon the creation/Loading of a folio on the GUI, a FolioInterface is added to the ArrayList of FolioInterface's in FolioHolder. This therefore allowing access to the Folio class methods. One of these methods being to add a holding. Once called - with the correct parameters - a HoldingInterface is added to the ArrayList of HoldingInterface's within Folio. The system then has full access to the Holding class and the methods within. This therefore meaning the details of Stock. When saving a Folio, the Stocks and therefore, the Holdings that are held within a Folio are saved and the FolioInterface added to the ArrayList with FolioHolder is updated. This making FolioHolder the base of the entire back end system.

The connection between the GUI system and the API is represented in the figure below:



From the figure above, it can be seen that the GUI doesn't have any direct connections to the API (Model.package). The way the GUI gets information like Stocks is via the Controller.java and EditStockController.java classes which contain instances of the API structure.

API Methods-

Java Class	Method Name	Description
Stock	**getPrice()** *Returns* → double price	Returns the price of a stock that is retrieved from the server.
	setPrice(double p) *Parameters* → double p	Adds the price of a stock at the current time to the list of prices.
	getTickerID() *Returns* → String tickerStumbol	Returns the Ticker ID/Symbol of a Stock.
	getName() *Returns* → String name	Gets the name of the Stock.
	getPriceLocally() *Returns* → double priceLocally	Retrieves the last fetched price from the server that has been saved locally.
	timeOfLastPrice() *Returns* → LocalDateTime timeOfLastPrice	Used to find the time of the last Stock update.
	getPriceHistory() *Returns* → ArrayList<Double> priceHistory	This method gets the prices of all the last updated Stocks.

	getLow() *Returns* → double low	Obtains the lowest price for Stock that has been saved.
	getHigh() *Returns* → double high	Returns the highest price for Stock that has been saved.
	currentProfitOnSingleUnit() *Returns* → double profitOnSingleUnit	Returns a double that will inform the user how much they would fair if they were to sell their Stock.
	updateInLastDay() *Returns* → Boolean hasBeenUpdated	Confirms if the price has been updated in the last 24hrs
	getPriceOneDayAgo() *Returns* → Price comparator	Finds the oldest possible price within the last 24hrs and returns it.
	comparePrice(double old, double new) *Parameters* → double old, new *Returns* → double d	Compares the value of two prices and returns the difference.
	isPriceUpOrDown() *Returns* → double comparedPrice	Checks if the price has gone up or down within the last 24hrs
	priceBoughtAt() *Returns* → double boughtPrice	Returns the price the Stock was initially bought at.
Holding (For GUI TableView use)	**changeShares(int change)** *Parameters* → int change	This changes the shares of a holding.
	getNoOfShares() *Returns* → int noOfShares	Returns the number of shares.
	setHoldingValue()	Sets the total value of a holding
	getValue() *Returns* → double value	Returns the total value of the holding.
	getTickerSymbol() *Returns* → String price	Obtains the tickerSymbol of a stock
	getName() *Returns* → String name	Returns the name of the Stock
	getPrice() *Returns* → double price	Retrieves the price of a single share of Stock
	getProfit() *Returns* → double profit	Gets the value of the holding i.e. the value for the number of shares owned
	getChangeOfPrice() *Returns* → double changeOfPrice	Gets the double value of the price if it goes up or down
	refreshPrices()	Refreshes the prices of the current Stock
	getPriceHistory() *Returns* → ArrayList<Double> priceHidtory	Gets the price history of all Stocks
	getLocalDateTime() *Returns* → LocalDateTime timeOfLastPrice	Receives the LocalDateTime of the time of last price of the Stock
	setPrice(double price) *Parameters* → double price *Returns* → double price	Sets the price of a specific Stock
	getHighest() *Returns* → double highest	Retrieves the highest Stock price
	getLowest() *Returns* → double lowest	Retrieves the Lowest Stock price
	getUpOrDown() *Returns* → String isUpotDown	Refreshes prices and returns a String indicating if the price is same, went up or down
	refreshUpOrDown()	Gets change in prices
Folio	**refreshPrices()**	Refreshes all prices held within a Folio
	getTotalHolding() *Returns* → double totalHolding	Returns the total value of the Folio
	addHolding(String name, String tickerSymbol, int shares) *Parameters* → String name, tickerSymbol int shares *Returns* → boolean hasBeenAdded	Adds a stock with a number of shares set in the parameter (int shares) to the folio and returns a boolean value if that stock exists.
	deleteStock(String tickerSymbol) *Parameters* → String tickerSymbol	Deletes the stock selected by its ticker symbol.
	getStock(String tickerSymbol)	Returns an instance of a stock by its ticker symbol.

	Parameters → String tickerSymbol *Returns* → HoldingInterface stock	
	getFolioName() *Returns* → String folioName	Returns name of the current folio
	setFolioID() *Returns* → String ID	Sets an ID for the folio based on four alphanumeric characters.
	getFolioID() *Returns* → String folioID	Returns the ID of the current folio.
	checkID(String id) *Parameters* → String id *Returns* → boolean alreadyUsed	Checks if the ID from saved file is unique, is not found elsewhere
	getStocks() *Returns* → ArrayList<HoldingInterface> stockList	Returns all the stocks in a portfolio
FolioHolder	**getFolio(String name)** *Parameters* → String name *Returns* → FolioInterface folio	Returns an instance of a folio with the given name
	getFolioByID(String id) *Parameters* → String id *Returns* → FolioInterface folio	Returns an instance of a folio with the given id
	createfolio(String name) *Parameters* → String none	Adds a new folio with a new name to the holder class
	deleteFolio(String id) *Parameters* → String id *Returns* → boolean deleted	Deletes a folio with a given id
	openFoliosWithSameName(String name) *Parameters* → String name *Returns* → int numOfSameFolio	Returns an index indicating the number of times a folio with the same name is opened to allow differentiation between different folios
	loadFolio(String folioID) *Parameters* → String folioID *Returns* → boolean loaded	Request a folio from a file by its ID
	saveFolio(FolioInterface folio) *Parameters* → FolioInterface folio	Saves folio in the file
	getExistingFolios() *Returns* → ArrayList<FolioInterface> folios	Returns all folios