W03 PortfolioTracker Design Rationale

for PortfolioTracker

Portfolio Manager is an application that allows you to keep track of Stock prices using portfolios. Price are fetched in Stocks can easily be bought an easy-to-use interface. You can save and load Portfolios from your disk.

Overall Rationale

To decouple the components portfolio tracker has been split into 2 parts. Backend and Frontend. FrontEnd is broken down more into 3 components. Portfolio Manager, Portfolio and Stock.
We have followed the MVC and Observer design patterns closely. Each model is an Observable and Views are Observers. When the model is changed the view will be notified and updated.

To maintain decoupling across the application, Action Listeners were implemented as separate classes rather than anonymous inner classes or lambda functions.

Packages were used to separate different functionality blocks of the application. Action listeners for each functional module were grouped together under ‘controller packages’. Controllers processes the events from views to models, and such functionality as input validation reside there. It should be noted that controller use interface reference types exclusively for design decoupling.

Features

- Manage multiple portfolios
- Create and delete portfolios
- See the positions (stock name, number of shares held, price per share, and value of the holding) in a portfolio
- Add and remove stocks to a portfolio
- Increase and decrease the number of shares of a stock
- Refresh stock prices

Bonus

- Save and load portfolios from disk
- Rename stocks
- Input validation with meaningful error messages
- Confirmation on critical actions (buy/sell stock and delete)
- Auto refresh prices in stock edit/buy/sell view
- Sorting positions by columns (value, ticker symbol, etc).

Changes

It was noted during review that we did not include any interfaces. We decoupled the class further by adding interfaces. Also we replaced all references with the interface type to further abstract the modules. It is also a design principle in SOLID known as Interface Segregation Principle.

It was noted that the save, close and save item menu should not be enabled when no portfolios are open. We resolved this by disabling those components in such case.

We removed unused methods from PortfolioManager extended implemented IPortfolio which extended Collections. It contained lots of unused and methods that were not necessary for the functionality.

Backend API

Interface IQuoteServer

Method Detail

- **getStockPrice**
  - double getStockPrice(java.lang.String tickerSymbol)
    throws StrathQuoteServer.NoSuchTickerException,
    StrathQuoteServer.WebsiteDataException
  - retrieve the latest market value of a stock
  - requires: tickerSymbol != null
  - effects: returns a current value for tickerSymbol as a dollar amount, with a period separating dollars and cents (eg, "120.50" for one hundred and twenty dollars and fifty cents)
  - unless tickerSymbol is not a valid NYSE or NASDAQ symbol, when throws NoSuchTickerException
  - or unless an error connecting to the website or some other error occurs, when throws WebsiteDataException
  - The amount returned may contain commas, for example, "2,243.87"
  - **Throws:**
  - StrathQuoteServer.NoSuchTickerException
  - StrathQuoteServer.WebsiteDataException

- **isValidStockTicker**
  - boolean isValidStockTicker(java.lang.String tickerSymbol)
  - Checks if a ticker symber is valid
  - requires: tickerSymbol != null
  - effects: Returns true if the string passed in is a valid ticker symbol