ReadMe for Portfolio Valuation

# Prerequisite

The project is developed using IntelliJ IDEA Community Edition v2016.3. In addition, Java 8 / 1.8 is the designated target byte code version as the project relies on some Java 8 features such as Lambda and stream API.

# Project Structure

The project consists of 4 modules

## Module 1 – pv.common.model

* This module contains all the domain data model used by the program

## Module 2 – pv.common.service

* This module contains some common service class such as Pricing Engine and Security Repository

## Module 3 – pv.marketdata

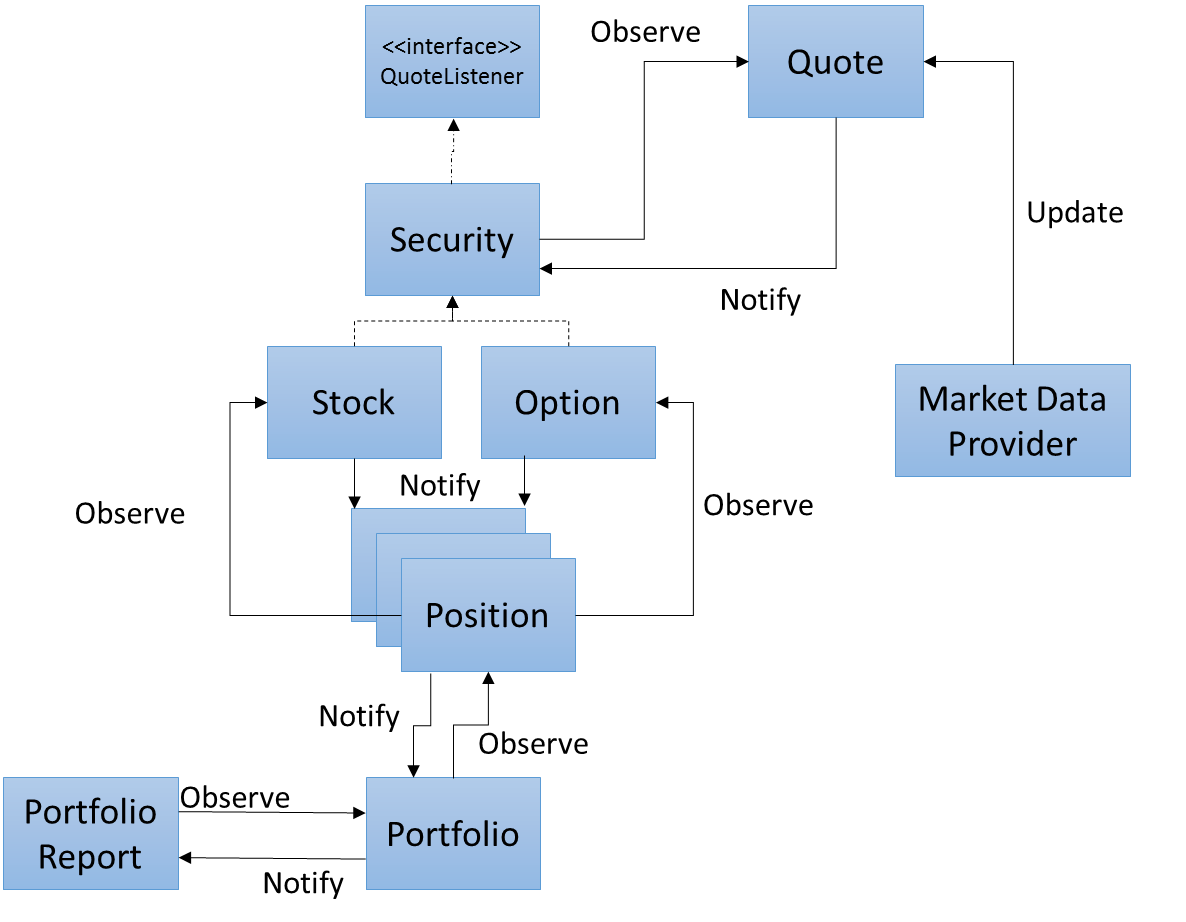
* This module contain the implementation of the Market Data provider

## Module 4 – pv.reporting

* This module depends on above modules. It contains the services that report the portfolio’s value / position values
* The TestExecutor act as the entry point for this application. To launch the application, please use ‘Run TestExecutor’ under the ‘Run’ menu in IntelliJ

# Object Relationship

The project use Observer pattern to receive notification of any price update and publish the changes in the positions and portfolios



# Security Database

The program use a H2 database table to store the security data. This schema of the Security table:

|  |  |  |
| --- | --- | --- |
| Column | Data Type | Remarks |
| Identifier | varchar(25) | Primary key. The unique id of the security.  For stocks, this should be the ticker.  For options, this should be OCC Option symbol which is in the format TTTTTYYMMDDOPPPPPPPP:  TTTTT – ticker of the underlyer, space padding to 6 digit  YYMMDD – expiry date of the option  O – option type: C for call, P for put  PPPPPPPP – Strike price  as the price x 1000, front padded with 0s to 8 digits |
| Name | Varchar(50) | Name of the security |
| SecurityType | Varchar(10) | ‘STOCK’ if it is a stock  ‘OPTION’ if it is an option |
| Volatility | Double | Volatility of the stock |
| MeanReturn | Double | Mean return of the stock |
| LastClose | Double | Last close price of the security. This will be considered as the starting price of the security. |
| Underlyer | Varchar(25) | For Options only, the ticker of the underlying stock. Foreign key constraint referencing the Identifier column |
| ExpiryDate | Date | For Options only, the expiry date of the option |
| OptionType | Varchar(20) | ‘PUT’ for put option, ‘CALL’ for call option |
| Strike | double | Strike price of the option |
|  |  |  |

Every time when the SecurityRepository class reload the securities. It will rebuild the DB using the script file ‘H2TableInit.sql’, which is in the resources folder of the pv.reporting module. Users can feel free to edit the script if you want to add more security for testing purpose.

# Portfolio Data

The portfolio data is read from the PositionFile.csv. The file is in the following format

PortName, Asset, Quantity  
TEST1, APPL, 2000  
TEST1, APPL 171231C00220000, 2000  
TEST1, APPL 171231P00220000, -5000  
TEST2, HSBC, 1000  
TEST2, HSBC 171231C00060000, 2000  
TEST2, HSBC 171231P00044000, -5000

The PortName column is the portfolio name. The system will automatically create a new portfolio for an asset if it does not exists. Users can group the assets into different portfolio by providing different portfolio name.

The Asset column is the security’s identifier. If the Asset does not exist in the Security database, it will be ignored. Quantity column is the quantity of asset hold, +ve Quantity represents long position, -ve represents short

# Output Portfolio Data

In the default implementation, the Market Data provider will run on a separate thread, which the portfolio reporter will run in the main thread. There are two portfolio reporter

1. ConsoleRealTimeReporter will monitor any real time Portfolio NAV change and output the Portfolio’s NAV to the Console
2. TextBaseOnDemandReporter will print a summary positions report together with the portfolio NAV to a text files

By Default, the position report’s location is at C:\Temp\PortfolioReport.txt. If you want to change this, please change the follow line (#27) in the TestExecutor file in the pv.reporting module

String portfileReportFile = **"C:\\Temp\\portfolioReport.txt"**;

Under the current implementation, the program will launch the ConsoleRealTimeReporter first by default. Users can press ‘Enter’ at the console to stop both the ConsoleRealTimeReporter and the MarketDataProvider. After they are stopped, the TextBaseOnDemandReporter will generate the latest positions report and write to the specified file.