CSC4350/6350

Spring 2017

A stock trading software. The price is real-time and the exchange is virtual.

Virtual Stock

BATMAN

2/28/2017

Mengyuan Zhu, Sungjae Kim, Sharon Kim, Jakub Pietrasik, Hyeun Kang

Virtual Stock

Contents

[Software Architecture 2](#_Toc476087161)

[Requirements Traceability Matrix 2](#_Toc476087162)

[Design of each object in the application 3](#_Toc476087163)

[Work Structure Document 5](#_Toc476087164)

[Dictionary 6](#_Toc476087165)

[Rational 6](#_Toc476087166)

[Category Interaction Diagram 8](#_Toc476087167)

[Gantt Chart 8](#_Toc476087168)

# Software Architecture

Virtual Stock software is in the Model/View/Controller (MVC) architecture style.

Models: Stock information and user information

Views: SignIn, SignUp, MainApp, History, Account, Portfolio, Banking and Settings

Controllers: controller is used to start the views and process information of stocks and users.

# Requirements Traceability Matrix

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entry#** | **Para#** | **Requirements Traceability Matrix** | **Type** | **Use Case Name** |
| 1 | 1 | Batman shall develop a stock trading software with a user-friendly GUI. | SW, HW | Use Case 3 Show\_Main\_Window |
| 2 | 2 | The system shall provide a window for user to register an account in Batman. | SW | Use Case 2 User\_Sign\_Up |
| 3 | 2.1 | The system shall let user know and ask the user to refill the form for registration. | SW | n/a |
| 4 | 3 | The system shall have a window for user to log in. | SW | Use Case 1 User\_Sign\_In |
| 5 | 3.1 | The window shall provide a user name, password input, forgot password button, and sign in and sign up buttons. | SW | n/a |
| 6 | 3.2 | The system shall let user into the interface if the user type the right user information | SW | n/a |
| 7 | 3.3 | If the user failed to log in to the system, the system shall ask the user to refill the password. | SW | n/a |
| 8 | 3.4 | The system shall provide a pin number for quick log in | NTH | n/a |
| 9 | 4 | The system shall have a portfolio interface. | SW | Use Case 4Show\_Portfolio |
| 10 | 4.1 | The system shall show a chart of user’s balance chart of today, total balance value, breaking news of today, stocks that a user keeps and his/her shares, and a watch list. | NTH | n/a |
| 11 | 4.2 | The search button shall also be shown in the corner for user to search a specific stock. | NTH | n/a |
| 12 | 4.3 | The color of the GUI shall be green if the user’s balance goes up and be red if the user’s balance goes down. | NTH | n/a |
| 13 | 5 | The system shall provide a window to show user’s account. | SW | Use Case 5 Show\_Account |
| 14 | 5.1 | The account shall show the total balance, stocks balance, and cash balance. | NTH | n/a |
| 15 | 6 | The system shall give a function for user to transfer money to a bank or to the Batman app. | SW | Use Case 6 Show\_Banking |
| 6 | 6.1 | The linked accounts shall be shown in the bottom. | NTH | n/a |
| 17 | 6.2 | The system shall provide a bank account for user to deposit money. | SW | n/a |
| 18 | 7 | The system shall provide a list of history of user’s trading log. | SW | Use Case 7 Show\_History |
| 19 | 7.1 | The system shall also show the date. | NTH | n/a |
| 20 | 7.2 | The system shall allow the user to see the history from/to a specific date. | NTH | n/a |
| 21 | 8 | The system shall have a setting interface for user to reset pin number and update user information including name, password, email, phone number and address. | SW | Use Case 8 Show\_Settings |
| 22 | 8.1 | The system shall also have a log out button for the user to log out. | SW | n/a |

# Design of each object in the application

* **MainApp.java**
  + **Start() –** initializes the layout and enables the view window. Sets stage title and icons.
  + **MainApp() –** constructor.
  + **showSignIn() –** sets file location, loads title, stage, and scene data for the Sign In page.
  + **showAccount() -** sets file location, loads title, stage, and scene data for the Account page.
  + **showBanking() -** sets file location, loads title, stage, and scene data for the Banking page.
  + **showSetting() -** sets file location, loads title, stage, and scene data for the Setting page.
  + **showHistory() -** sets file location, loads title, stage, and scene data for the History page.
  + **showStockOverview() -** sets file location, loads title, stage, and scene data for the Stock Overview page.
  + **showStockPurchaseDialog() -** sets file location, loads title, stage, and scene data for the Stock Purchase Dialog page.
  + **getPrimaryStage() –** returns primary stage.
  + **main() –** launches program.
  + **getPersonData() –** returns personData object.
  + **getPersonFilePath() -** returns file path for user’s personal preferences.
  + **setPersonFilePath() -** sets file path for user’s personal preferences.
  + **loadPersonDataFromFile() –** loads user data using file path.
  + **savePersonDataToFile() -** saves user data using file path to a file location.
  + **initRootLayout() –** initializes scene layout.
  + **showPortfolio() -** sets file location, loads title, stage, and scene data for the Portfolio page.
  + **showSignUp() -** sets file location, loads title, stage, and scene data for the Sign Up page.
  + **showDayDataStatistics() -** sets file location, loads title, stage, and scene data for the Show Day Data Statistics page.
* **Stock.java**
  + **Stock() –** constructor.
  + **getStockName() –** gets StockName object’s value.
  + **setStockName() –** sets StockName object’s value.
  + **stockNameProperty() -** gets StockName object.
  + **getPrice() –** gets Price object’s value.
  + **setPrice() –** sets Price object’s value.
  + **priceProperty() –** gets Price object.
  + **getShares() –** gets Shares object’s value.
  + **setShares() –** sets Shares object’s value.
  + **sharesProperty() –** gets Shares object.
  + **getOpen() –** gets Open object’s value.
  + **setOpen() –** sets Open object’s value.
  + **openPorperty() –** gets Open object.
  + **getTodayHigh() –** gets TodayHigh object’s value.
  + **setTodayHigh() –** sets TodayHigh object’s value.
  + **todayHighProperty() –** gets TodayHigh object.
  + **getTodayLow() –** gets TodayLow object’s value.
  + **setTodayLow() –** sets TodayLow object’s value.
  + **todayLowPorperty() –** gets TodayLow object.
  + **getYearHigh() –** gets YearHigh object’s value.
  + **setYearHigh() –** sets YearHigh object’s value.
  + **yearHighProperty() –** gets YearHigh object.
  + **getYearLow() –** gets YearLow object’s value.
  + **setYearLow() –** sets YearLow object’s value.
  + **yearLowProperty() –** gets YearLow object.
  + **getVolume() –** gets Volume object’s value.
  + **setVolume() –** sets Volume object’s value.
  + **volumeProperty() –** gets Volume object.
  + **getAverageVolume()** – returns value within averageVolume object.
  + **setAverageVolume()** - sets averageVolume object.
  + **averageVolumeProperty()** – returns averageVolume object.
  + **getMarketCap() –** gets MarketCap object’s value.
  + **setMarketCap()–** sets MarketCap object’s value.
  + **marketCapProperty() –** gets MarketCap object.
  + **getPERatio() –** gets PERatio object’s value.
  + **setPERatio() –** sets PERatio object’s value.
  + **PEratioProperty() –** gets PERatio object.
  + **getDivYield() –** gets DivYield object’s value.
  + **setDivYield() –** sets DivYield object’s value.
  + **divYieldProperty() –** gets DivYield object.
* **StockDayData.java**
  + **main() -** main method for yahoo data retrieval class.
  + **getURLSource() –** yahoo data parsing method.
* **StockListWrapper.java**
  + **getPersons()** – returns the stocks value.
  + **setPersons() –** sets the stocks value to that of input.
* **User.java**
  + **User()** – constructor.
  + **getFirstName() –** gets FirstName object’s value.
  + **setFirstName() –** sets FirstName object’s value.
  + **firstNameProperty() –** gets FirstName object.
  + **getLastName() –** gets LastName object’s value.
  + **setLastName() –** sets LastName object’s value.
  + **firstLastProperty() –** gets LastName object.
  + **getStreet() –** gets Street object’s value.
  + **setStreet() –** sets Street object’s value.
  + **streetProperty() –** gets Street object.
  + **getSSN() –** gets SSN object’s value.
  + **setSSN() –** sets SSN object’s value
  + **SSNProperty() –** gets SSN object.
  + **getBirthday() –** gets Birthday object’s value.
  + **setBirthday() –** sets Birthday object’s value.
  + **birthdayProperty() –** gets Birthday object.
* **DateUtil.java**
  + **format() –** receives the date values and formats them together.
  + **parse() –** parses through dateString for date values.
  + **validDate() –** returns true if dateString isn’t null.

# Work Structure Document

Group name: **Batman**

|  |  |
| --- | --- |
| **Name** | **Tasks** |
| Mengyuan Zhu | Team Coordinator  Documents handler  Java coder  Problem Statement  Requirements Traceability Matrix  Dictionary  A horizontal prototype of the software to be developed  Use cases and Interaction Diagrams - example as per given in class  Database to be used  Rational  Architecture |
| Sungjae Kim | Finalize code documentation  Java coder  Requirements Traceability Matrix  Gannt Chart  Category Interaction Diagram |
| Sharon Kim | User Guide  Function Point Cost Analysis.  Program tester  Rational |
| Jakub Pietrasik | Rational |
| Hyeun Kang | Object Design |

# Dictionary

**Portfolio:** In finance, a portfolio is a collection of investments held by an investment company, hedge fund, financial institution or individual.

**Broker:** A person who buys or sells an investment for you in exchange for a fee (a commission). Here is Tim’s favorite broker.

**Dividend:** this is a portion of a company’s earnings that is paid to shareholders, or people that own hat company’s stock, on a quarterly or annual basis. Not all company’s do this.

**Exchange:** An exchange is a place in which different investments are traded. The most well-known in the United States are the New York Stock Exchange and the Nasdaq.

**Quote:** Information on a stock’s latest trading price. This is sometimes delayed by 20 minutes unless you are using an actual broker trading platform.

**Volume:** The number of shares of stock traded during a particular time period, normally measured in average daily trading volume.

**Yield:** This usually refers to the measure of the return on an investment that is received from the payment of a dividend. This is determined by dividing the annual dividend amount by the price paid for the stock. If you bought stock XYZ for $40-a-share and it pays a $1.00-per-year dividend, you have a “yield” of 2.5%.

**JDK:** The Java Development Kit (JDK) is an implementation of either one of the Java Platform, Standard Edition; Java Platform, Enterprise Edition or Java Platform, Micro Edition platforms released by Oracle Corporation in the form of a binary product aimed at Java developers on Solaris, Linux, Mac OS X or Windows.

**GUI:** The graphical user interface is a type of user interface that allows users to interact with electronic devices through graphical icons and visual indicators such as secondary notation, instead of text-based user interfaces, typed command labels or text navigation.

**HTTP:** The Hypertext Transfer Protocol (HTTP) is an application protocol for distributed, collaborative, and hypermedia information systems. HTTP is the foundation of data communication for the World Wide Web.

**XML:** In computing, Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. The W3C's XML 1.0 Specification and several other related specifications-all of them free open standards—define XML.

# Rational

**Object Rational**

**Stock.java**

Stores information pertaining to stock attributes. Includes name, price, shares, open, TodayHigh, TodayLow, YearHigh, YearLow, Volume, Marketcap, PARatio, and DivyYield values.

**StockDayData**

Manages retrieval of the stock data from Yahoo and contains method for getting the URL source.

**StockListWrapper**

Wrapper for Stock List that has a getter and a setter for the person object.

**User**

Stores information about the user including attributes such as name, address, social security number, and date of birth.

**DateUtil**

Finds and formats the date from dateString.

**AccountController**

Provides the elements of the visual interface for the Account screen. This includes labels for total, stock and cash, as well as an OK and a withdraw button.

**BankingController**

Provides the elements of the visual interface for the Banking screen. This includes buttons for transferring between the bank and VS, automatic depositing, and linking accounts.

**HistoryController**

Provides the elements of the visual interface for the History screen. This includes a combo box for the date range and a table of the history.

**PortfolioController**

Provides the elements of the visual interface for the Portfolio screen. This includes elements such as a bar chart, buttons for managing time intervals for day/week/month, and allows for the setting of person data.

**RootLayoutController**

Provides the elements of the visual interface for the Root Layout screen. Handles the Sign In, Sign Up and Showing of portfolio, banking, account and history screens.

**SettingController**

Provides the elements of the visual interface for the Setting screen. This includes text fields for account name and password, email, phone, address as well as the buttons for updating the content with that entered as well as one for logging out.

**SignInController**

Provides the elements of the visual interface for the Sign In screen. Includes buttons for signing in, registering and text fields to allow for the entering of a username and a password.

**SignUpController**

Provides the elements of the visual interface for the Sign Up screen. Includes labels for the name, password, account name, address, routing number, account number, initial deposit, phone and email contact information. Also includes a button to confirm and one to cancel.

**StatisticsController**

Provides the elements of the visual interface for the Statistics screen. Includes a bar chart and category axis of stock information. Allows for the viewing of and setting of Person Data.

**StockDayController**

Provides the elements of the visual interface for the Stock Day screen. Allows for the setting of stock quote data as well as the controller for Stock Day.

**StockOverviewController**

Provides the elements of the visual interface for the Stock Overview screen. Manages overview elements including stock name, stock price, shares, open, today high, today low, year high, year low, volume, average volume, market cap, PERatio, and div field. Shows person details.

**StockPurchaseController**

Provides the elements of the visual interface for the Stock Purchase screen. Displays the elements of stock name, shares, and the dialog box that appear when making a stock purchase.

**Architecture Rational**

Virtual Stock software is in the Model/View/Controller (MVC) architecture style. Subsystems are classified into three different type. It has these reasons: Simultaneous Development - Multiple developers can work simultaneously on the model, controller and views. High Cohesion - MVC enables logical grouping of related actions on a controller together. The views for a specific model are also grouped together. Low Coupling - The very nature of the MVC framework is such that there is low coupling among models, views or controllers. Ease of modification - Because of the separation of responsibilities, future development or modification is easier. Multiple views for a model - Models can have multiple views.

# Category Interaction Diagram

Next page

# Gantt Chart

Next page