Team WINY

Irwin Li (izl2000) Yiming Sun (ys2832)

Wanlin Xie (wx2161) Nancy Xu (nx2131)

Responsibilities:
Irwin and Wanlin: CRC cards
Nancy and Yiming: Class Diagrams

CRC cards

Class Name: User (user story #1: create account)
Responsibilities:

- update portfolio(portfolio id)
- Delete portfolio(portfolio id)

Collaborators: Portfolio

Class Name: Portfolio (user story #3: view portfolio, #4: add to portfolio)
Responsibilities:

- add stocks

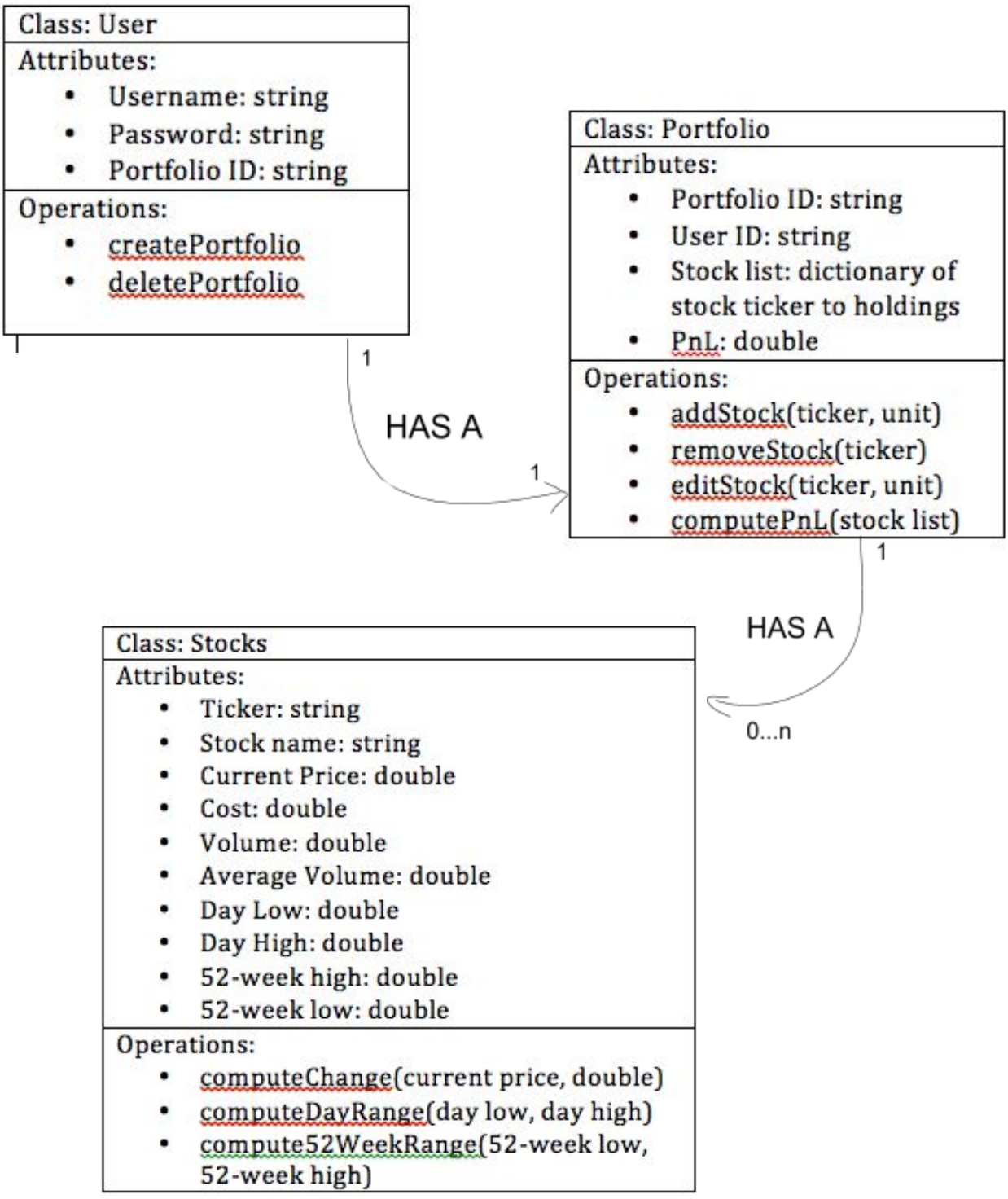
Collaborators: Stocks, User

Class Name: Stocks
Responsibilities:

- Maintain stock information (pull information from web)

Collaborators: Portfolio

Class Diagram



There are 3 classes in our class diagram that work in unison to describe the static structure of our software system. The first of which is the “User” class, which describes a typical user on our application platform. Its attributes include username and password, which are required for the user to login to and access the platform. The username also uniquely identifies the user in the system. Every user has a portfolio assigned to him/her, so the unique portfolio ID is also an attribute of the User class. The User class has the createPortfolio and deletePortfolio functions, which allows the user to modify and empty his/her portfolio respectively.

The second class is the “Portfolio” class, which has the attributes portfolio ID, user ID, stock list, and PnL. The portfolio ID uniquely identifies a portfolio object, whereas the user ID identifies the user who owns the portfolio in question. The stock list attribute is a dictionary that maps the stock ticker (of the stock that the owner owns) to the owner’s holdings in that particular stock. This attribute allows us to find the PnL attribute, which is a computation of the portfolio’s overall gains or losses. Accordingly, the operations in the portfolio class include addStock, removeStock, editStock (which allows the user to modify his/her holdings in a particular stock), as well as computePnL, which updates the portfolio’s PnL at that point in time.

The third class is the Stock class, which contains the attributes ticker, stock name, current price, cost, volume, average volume, day low, day high, as well as 52-week low and high. The ticker uniquely identifies the stock. The class’s computeChange operation computes the difference between the stock’s current price as well as the cost, which is the price that the user bought the stock at. The computeDayRange and compute52WeekRange functions respectively find the difference between the most recent day low and day high price of the stock, as well as the 52-week low and high prices.

Each user has a single portfolio, so the relationship between the two classes is that of association (unidirectional, extending from the user class to the portfolio class). The multiplicity of each class in this relationship is 1.

Each portfolio may contain 0 or more stocks, so the relationship between the two classes is also that of association (unidirectional, extending from the portfolio class to the stock class). The multiplicity of the portfolio class is 1, while that of the stocks class is 0...n.

Additional classes are not needed.