stock_broker design

WO1 Clayton E. Williams

October 2023

1 Project Overview

stock_broker is a command-line utility tool that allows a user to manage aspects related to stock portfolio. stock_broker reads a json file of stocks with their respective symbols, initial price, and volatility that shows price changes each time the stock is viewed. The user is able to create customers, and interact with each, create accounts, and for each account, deposit or withdraw funds, as well as buy and sell stocks through the use of interactive menus.
Because this program relies on multiple customers, accounts, holdings, stocks, and menus that share attributes, a class with unique instances of each will be used to hold and display the data. The menus will all be sub-classes of the Interface base class adapted modified from a previous project.

2 Project Structure

Project Structure Breakdown:

```
.
            account.py
            bank.py
            broker.py
            customer.py
            data
|               stock_data.json
            doc
|               broker.pdf
|               testplan.pdf
|               writeup.pdf
            interface.py
            README.md
            stock_broker.py
            test
```

. - Each python module will be contained in the project root directory. stock$_b$*roker.pyisthemainfilefortheprog*
data − *Containsthejsonfileofstocks*

test – Pythontestmodulesfortheprogram

3 Data Structures Needed

There are 7 main classes needed to hold data and functionality, and 3 sub-classes.

```
class Stock:
    '''Provides attributes and methods for interacting with stock objects
    held in a list by bank. Below are the following attributes of a stock
    object.'''
    ticker # stock symbol
    name # stock name
    start_price # initial stock price
    volitality # Amount stock price can change over time

class Bank
    '''There is only one instance of the Bank object. Its purpose is to
    manage the transactions for customers and accounts.'''
    '''Unique IDs maintained by the Bank to ensure each new account and
    customer has a unique ID.'''
    acct_unique_id
    customer_unique_id
    _stock_list # dictionary of stock objects, ticker is key
    _customer_list # dictionary of customer objects, id is key

class Customer
    '''Provides attributes and methods for interacting with customer
    objects. Below are the following attributes of a customer object.'''
    name # Customer name
    id # Unique id assigned to a customer by bank
    zipcode # Customer zipcode
    date_enrolled # YMD customer was created
    accts # Dictionary of accounts owned by customer, key is account id

class Account
    '''Provides attributes and methods for interacting with account objects.
    Below are the following attributes of an account object.'''
    owner_name # Customer name that owns account
    owner_id # Unique id of the account owner
    acct_type # Str of the account type
    balance # current account balance
    acct_number # Unique id of the account
    holdings # dictionary of current stock holdings. Key is stock ticker
    transactions # list of transactions made on account object
```

```
class Holding
    '''Provides attributes and methods for interacting with holding objects.
    Below are the following attributes of a holding object.'''
    stock # A stock object
    shares # Number of shares of specified stock
    purchase_price # Purchase price, stock price * shares

class Transaction
    '''Provides attributes and methods for interacting with transaction
    objects. Below are the following attributes of a transaction object.'''
    timestamp # Y-M-D H:M:S when transaction initiated
    tran_type # Purchase or Sale
    price # Cost(purchase) or revenue(sale) from the transaction

class Interface
    '''Provides a base class for the interactive menus. It is inherited by
    the three necessary menus for the project to display options and call
    the respective functionality.'''
    class
    class MainMenu(Interface)
    class CustomerMenu(Interface)
    class AcctMenu(Interface)
```

4 Project Flow

- MainMenu run
  Call run on the MainMenu object. This is the entry point the project. This, and all Interface subclass calls are wrapped in while loop to continue to display untill the name of the function returned is 'back', indicating the user selected the respective menu option to go back. MainMenu.run itself is wrapped in a try/except to handle KeyboardInterrupt.

- Get input
  Interface handles getting menu option input from user, checks that it is a valid option (numeric, and within range of menu options), and retrieves the corresponding function from list of menu options.

- Function Call
  The returned function is called, progressing the user further into the program. The customer list is stored in Bank, so that there is only one instance of it, and can be passed to each function to ensure actions taken on a customer, or its accounts, holdings, and transactions are persistent throughout. Each function is responsible for its own validation of input, checking int vs float vs string, withdrawls don't exceed balance, etc.

- Return to menu
  Because the menus are wrapped in the while loop, when a function completes, it falls off the stack and the user is sent back to the menu. This helps keep the call stack short, with only one instance of a menu ever being on the call stack.