Programming Finance

Yilin Yang (yy450)

Final Project Report

- Account Management System developed
- Two accounts, Stock Account and Bank Account, are implemented that manage funds and trading through a common cash balance, a file named Balance.txt
- Balance is initially $10000 but will be modified and remembered even between different program executions – the code can easily be changed to always initialize to $10000 if desired
- Stock Account
    - Two text files, Result1.txt and Result2.txt, store stock symbols and prices different from each other
    - Program will randomly select one file to read stock prices to simulate price fluctuation
    - Transactions will be recorded in Stock history through the stock_transaction_history.txt file
    - Portfolio worth will be updated based on stock trading through the two files, portfolio.txt and pvalue.txt
- Bank Account
    - Responsible for depositing or withdrawing money
    - Transactions will be recorded in Bank history through the bank_transaction_history.txt file
- Design Patterns
    - Polymorphism is implemented to streamline function design by making history actions, such as recording or printing, virtual functions – since they are all structurally similar but have slightly different constraints depending on the account being used, the base class Account has a generalized function that is adapted by the derived StockAccount and BankAccount classes
    - Hashing is implemented when reading stock data from the dataset provided, the constructor creates a hash table and stores prices into it with key being the stock symbol

Classes

- Account
    - Void refresh_cin() – wipes the standard input stream clean for user input
    - Double get_balance() – returns current cash balance
    - Void set_balance(double) – overwrites cash balance with new value
    - Virtual void write_history(string, double) – appends transaction history with new transaction, string is the type of transaction made, double is the cash amount involved

    - Virtual void print_history() – prints the transaction history
- Stock_Account
    - Sym get_File(sym, sym) – chooses one the two databases to pull stock data from
    - Void display_stock() – prints the stock data for a desired company
    - Void buy() – purchases a stock amount
    - Void buy_update(string, int) – update portfolio after buying
    - Void sell() – sells a stock amount
    - Void sell_update(Node*, int) – update portfolio after selling
    - Node *search(string) – locate stock symbol in linked list
    - Void isEmpty() - check if linked list is empty
    - Void sort() – sorts linked list using bubble sort
    - Virtual void write_history(string, double) – appends transaction history with new transaction, string is the type of transaction made, double is the cash amount involved
    - Virtual void print_history() – prints the transaction history
    - Void read_portfolio() – reads portfolio transactions
    - Void read_value() – reads portfolio worth
    - Void write_portfolio() – record portfolio transaction
    - Void write_value() – record portfolio worth
    - Void display_portfolio() – prints current portfolio state
    - Void plot() – plots portfolio values using matlab
- Bank_Account
    - Void view_balance() – view current cash balance
    - Void deposit() – deposit money into balance
    - Void withdraw() – withdraw money from balance