16:332:503 Project: Account Management System

Please read the project description **very carefully**:

For this project, you will be writing an account management system which will manage a stock portfolio account and a bank account.

Please use an inheritance structure for the classes used in the program. Create an abstract base class 'Account' that has two derived classes 'StockAccount' and 'BankAccount'.

The 'StockAccount' and 'BankAccount' information should be connected to each other through CASH BALANCE.

Stock portfolio account and Bank account will share a common initial balance of 10000$ available in your bank account. This balance keeps changing as the transactions goes on.

Stock Portfolio Account

For the stock portfolio account portion, you will need to use stock information stored in Results.txt files to create a portfolio managing system. (There are two files of result.txt attached with the project description. This is to mimic the changes in stock prices. The two files have same stock symbols with different prices. Hence your program should randomly choose one stock value from one of the two given files for all the transactions. That is, you need to choose a random value from one of the two files whenever(!) you need to read a stock value, including when you do sorting operation. Feel free to generate as many of your own txt files or you may pull real time data from any financial websites.)

This portfolio system needs to store the account information in text files so that when the program is closed the account information does not get lost. Each time the program is run, it needs to read the current account information from the text files. Whenever a transaction is made, please store the information in a text file (See below for more details). Whenever the program exits, store the total portfolio value and current cash balance along with the current date and time in a separate text file to keep a record of the history of the portfolio values (to be used when graphing the history of portfolio values). This file will be accessed by both 'BankAccount' and 'StockAccount' class. The common purpose of both the accounts is to access the current cash balance information in this file. Whenever the program exits and starts again, it shall use this updated balance and not the initial balance of $10000.

You must use a **doubly linked list** in the 'StockAccount' class to store the portfolio stock information. Each node should contain the stock symbol and the number of shares of that stock in the portfolio. Please implement the doubly linked list yourself (**don't use STL**). You can use your implementation from the homework assignment. You can use whatever data structures (including STL

structures) you want for the other data used in the program, but please use doubly linked lists for the portfolio stock information.

You must keep your stock portfolio in the doubly linked list **sorted at all times** in the **decreasing order** of the total value of any particular stock (i.e. number of shares * price per share). That is, after any update on your stock portfolio, you need to check and maintain the doubly linked list so as to keep it sorted.

To implement the sorting, **do not** copy the values into a different data structure and then sort it. The doubly linked list **must be** sorted in place, that is, by changing the links of nodes in the list.

You are required to use **at least two Design Patterns** learned in class in your program. You need to write clear comments in your program to highlight the use of the design patterns, and explain them in your written project description. You should **be creative** in using design patterns for your project – for example, you can allow the user to choose/change to different sorting methods to keep your linked list in order, and use a design pattern to implement this.

Your program should be able to perform the following:

1. **Display the price of a stock** – Display the price for a stock symbol based on the information obtained **randomly** from one of the two Results.txt files. The user would enter a stock symbol (that is contained in the Results.txt) and the program will return a price per share. If the symbol is not found, return that the symbol cannot be found.

Example:

Company-Symbol	Price per share
GOOG	$577.49

2. **Display the current portfolio** – The cash balance and information about the stocks in the portfolio should be displayed **in the order of** the sorted list. For each stock please display the symbol, the number of shares owned, the price per share (based on the value from Results.txt), and the total value of that stock. Please also display the total value of the portfolio.

Example:

Cash balance = $5000

CompanySymbol	Number	PricePerShare	TotalValue
GOOG	10	$577.49	$5774.90
MSFT	100	$30.00	$3000.00

Total portfolio value: $13774.90

3. **Buy shares** – The user will send a request to buy shares of a stock. The user should enter the ticker symbol of the stock he/she wants to buy, enter the amount of shares he/she wants to buy and the maximum amount he/she is willing to pay for each share of the stock (the limit). If the user has entered a stock purchase

amount that is more than his or her current cash balance in bank account, the transaction should fail and the program should print out the reason.

Each time the user requests to buy shares, the stock pricing information from Results.txt should be consulted (or you can just store the information into a data structure upon the start of the program and use that data structure whenever you need the information). If the stock ticker is not found in the text files, the program should print that the stock is not available. If the cost per stock is higher than the amount the user is willing to pay, the transaction should fail and the program should print out the reason the transaction failed.

If the transaction goes through, the cost of the transaction should be deducted from the bank cash balance and the stock that has been purchased should be added to the portfolio. Make sure to use the price of the stock that is in the Result.txt file when updating the portfolio. If the user already has that stock in his or her portfolio, add the number of shares purchased to that entry instead of creating a new entry (remember, you want to use a linked list to store the information).

The program should display on the screen the information from the transaction that has taken place. Please add this transaction to the stock_transaction_history.txt file. You also need to add a withdrawal transaction to your bank account transaction history for this buying operation.

4. **Sell shares** – The user will send a request to sell shares of a stock. The user should enter the ticker symbol of the stock he/she wants to sell, the amount of shares he/she wants to sell and the minimum amount he/she wants to sell each share of the stock for. If the user has entered a stock that is not in the portfolio, or if there are insufficient shares in the portfolio, the program should display this and the transaction should fail.

Each time the user requests to sell shares (and the shares are available), the stock pricing information from Results.txt should be used. If the price per stock is lower than the amount the user is willing to sell for, the transaction should fail and the program should print out the reason the transaction failed.

If the transaction goes through, the amount obtained from the "sell transaction " should be added to the bank cash balance and the shares that have been sold should be subtracted from the portfolio. If the number of shares of the stock is 0, the stock should be removed from the portfolio. Make sure to use the price of the stock that is in Results.txt when updating the portfolio. The program should display the information from the transaction that has taken place. Please add this transaction to the stock_transaction_history.txt file. You also need to add a deposit transaction to your bank account transaction history for this selling operation.

5. **View a graph for the portfolio value** – As mentioned earlier, when the program exits, the total portfolio value should be stored in a text file along with the date and time. The total portfolio value is the sum of your current cash

balance and money worth of the stocks you own. Make sure that you use both the "Results.txt" files for buy/sell transactions, so that the total portfolio value keeps changing and not always constant. Plot the variation in the value of the portfolio over a period of time. Use MATLAB to plot the graph.

You can use any period of time and the only information you need to display is the change in total value of the portfolio. The following code below shows two ways to print out the time:

```
#include <time.h>
time_t seconds;
seconds = time(NULL);
cout<<"The number of seconds since January 1, 1970 is:" <<seconds<<"\n";
tm * timeinfo;
timeinfo = localtime(&seconds);
cout<<"The current local time and date is:"<<asctime(timeinfo);
```

Please look at this code to see how to obtain time information.

6. **View transaction history** – The buy and sell transaction should be saved in stock_transaction_history.txt. If the user chooses this option, the transaction history should be printed in order of transaction time.

Example:

Event	CompSymbol	Number	PricePerShare	TotalValue	Time
Buy	GOOG	10	$577.49	$5774.90	09:40:07
Buy	MSFT	100	$30.00	$3000.00	13:37:00
Sell	MSFT	50	$20.00	$1000.00	14:39:21

Bank Account

The bank account portion is simpler than the stock portfolio part.

The bank account should initially have a balance of $10,000. The amount in the bank account should be stored in a text file upon exit so that it can be retrieved the next time the program starts. Save all transaction history in bank_transaction_history.txt.

As indicated before, when you buy or sell stock shares, money will come out of or go into your bank account. These activities should all be recorded in your bank account transaction history (as withdrawal or deposit, respectively).

Please have the following options:
1. **View account balance** – Prints out the account balance.
2. **Deposit Money** – The user selects the amount of money to deposit. This amount should be added to the balance.
3. **Withdraw Money** – The user selects the amount of money to withdraw. If the balance of the account is not sufficient to withdraw the amount, please print out

an error.
4. **Print out history** - The program should print out the history of transactions on the account in order of transaction time.

Example:

Event	Amount	Date	Balance
Deposit	$500.00	11/05/2009	$10,500
Withdrawal	$1000.00	11/06/2009	$9,500
Deposit	$1500.00	11/07/2009	$11,000

(Here we assume there is no stock transaction in this example; otherwise, stock transactions should also be printed for the bank account.)

Extra credit
Create a graphical user interface (GUI) for the program. You can use whatever C++ libraries (such as MFC, QT or fltk) you choose to do this. If you decide to do this please create it in a separate project (in FinalProject_gui_yourname.zip) and also submit the non-graphical version of the program. (**up to 20 extra points**)

Submission Guidelines

1. Give a brief description of your project, the functions you have implemented, data structures used, text files used or created and for what purpose and any other descriptions of your project. If you made any extra features for the program please clearly state it. Include the description in a Word file.

2. For the submission, please submit the entire project folder in a zip/rar file, FinalProject_yourname.zip. Make sure you comment your code. Your visual studio project should be named FinalProject_yourname. Please make sure all the cpp filenames have your name in it.

3. **Do not copy code from ANY other people. The project should be your own work. Otherwise, you will FAIL the course.**

Sample Program Output

Welcome to the Account Management System.
Please select an account to access:
1. Stock Portfolio Account
2. Bank Account
3. Exit

Option: 1
Stock Portfolio Account

Please select an option:
1. Display the price for a stock symbol
2. Display the current portfolio
3. Buy shares
4. Sell shares
5. View a graph for the portfolio value
6. View transaction history
7. Return to previous menu

Option: 1
Please enter the stock symbol: GOOG Company Symbol Price per share
GOOG $577.49

Please select an option:
1. Display the price for a stock symbol
2. Display the current portfolio
3. Buy shares
4. Sell shares
5. View a graph for the portfolio value
6. View transaction history
7. Return to previous menu

Option: 3
Please enter the stock symbol you wish to purchase: GOOG Please enter the number of shares: 10
Please enter the maximum amount you are willing to pay per share: 580
You have purchased 10 shares of GOOG at $577.49 each for a total of $5774.90.

Please select an option:
1. Display the price for a stock symbol
2. Display the current portfolio
3. Buy shares
4. Sell shares
5. View a graph for the portfolio value
6. View transaction history
7. Return to previous menu

Option: 3
Please enter the stock symbol you wish to purchase: MSFT Please enter the number of shares: 100
Please enter the maximum amount you are willing to pay per share: 30
You have purchased 100 shares of MSFT at $30 each for a total of $3000.00

Please select an option:
1. Display the price for a stock symbol
2. Display the current portfolio

3. Buy shares
4. Sell shares
5. View a graph for the portfolio value
6. View transaction history
7. Return to previous menu

Option: 2
Cash balance = $1225.10

Company-Symbol	Number	Price-per-share	Total value
GOOG	10	$577.49	$5774.90
MSFT	100	$30.00	$3000.00

Total portfolio value: $10000.00

Please select an option:
1. Display the price for a stock symbol
2. Display the current portfolio
3. Buy shares
4. Sell shares
5. View a graph for the portfolio value
6. View transaction history
7. Return to previous menu

Option: 6

Event	Company Symbol	Number	Price per share	Total value	Time
Buy	GOOG	10	$577.49	$5774.90	13:14:15
Buy	MSFT	100	$30.00	$3000.00	13:37:42

Please select an option:
1. Display the price for a stock symbol
2. Display the current portfolio
3. Buy shares
4. Sell shares
5. View a graph for the portfolio value
6. View transaction history
7. Return to previous menu

Option: 7

Please select an account to access:
1. Stock Portfolio Account
2. Bank Account
3. Exit

Option: 2
Bank Account

Please select an option:
1. View account balance
2. Deposit money

3. Withdraw money
4. Print out history
5. Return to previous menu

Option: 2
Please select the amount you wish to deposit: $1000

Please select an option:
1. View account balance
2. Deposit money
3. Withdraw money
4. Print out history
5. Return to previous menu

Option: 1
1. You have $2225.10 in your bank account.