PROGRAMMING FINANCE - 16:332:503

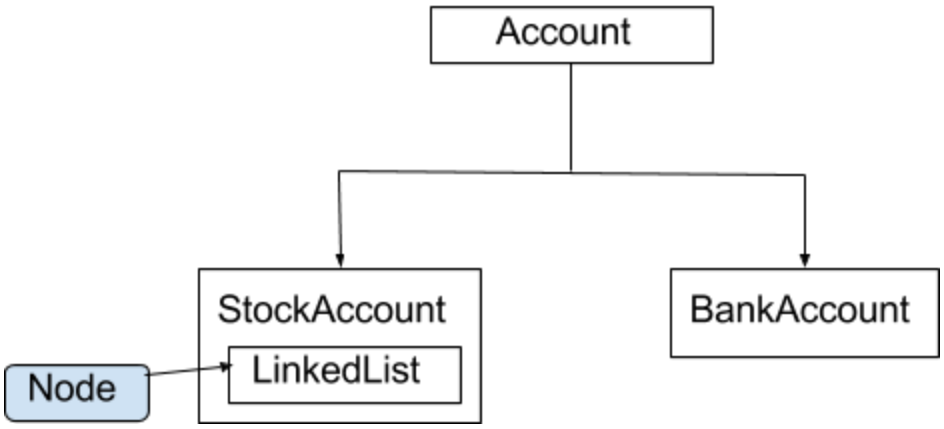
ACCOUNT MANAGEMENT SYSTEM

Juhi Tripathi
(Net ID - jt866)

Brief Description -

The project is a Account Management system, that enables the user to work on the bank account or the stock account. While banking account would deal with all the banking related operations; view balance, withdrawal, deposits and transaction history, the stock account deals with the portfolio information; buying/selling stocks, check stock value, transaction history, display current portfolio and its graph.

The project structure could be depicted as -



Account is the abstract base class, that has derived classes - StockAccount and BankAccount. Stock account class uses an object of LinkedList class, as its member variable. LinkedList class is a friend of Node, and the nodes of linked list are made using the Node class. Thus, LinkedList makes object of Node, and forms the list. The project code uses behavioral design pattern for the virtual functions. Transaction_history is defined in both the derived classes.

Text Files used/created-

Text Files used : In the stock portfolio account, stock information is retrieved randomly from the following 2 files, that have same stock symbols with different prices (mimics the changes in stock prices) for read and sorting operations -

- Results_1

- Results_2

Text files created : The text files created are mentioned below, along with their purpose-

- Stock_transaction_history : stores all the stock buy and sell transactions
- Storelinkedlist : stores the linkedlist data, created for stock account, to be accessed if the user works on stock account, goes to bank account, but then wants to go back to stock account and resume the previous work
- Bank_transaction_history : stores all the bank account deposit and withdrawal transactions
- CashBalance : stores the updated cash balance

Classes and functions implemented -

ACCOUNT CLASS :

- Public functions -
    - getCashBalance() - returns the current value of cashbalance
    - setCashBalance() - sets the cashbalance to a certain value, that is passed
    - transaction_history() - pure virtual function defined in both the derived classes
- Private member -
    - cashbalance - private data member that is accessed through the get and set functions

STOCKACCOUNT CLASS :

- Constructor :
    - Opens the CashBalance file and updates the cash balance value.
    - Initializes the header of empty stock_transaction_history.txt.
- Public functions -
    - Transaction_history : Retrieves the transaction_history file and outputs the data line by line.
    - UpdateCashBalance : Writes current cash balance to file
    - getstock : Gets the price of stock requested by user from one of the two results file, chosen randomly
    - buyshare : function is for letting the user buy a certain share. User enters the stock to be bought, number of shares, and the amount he is willing to pay for it. In order to let the user buy shares, function checks if the amount entered by user is greater than or equal to the value (of stocks to be bought), and accordingly -
        - updates cashbalance after deducting the total stock price

- outputs the result for user to know that the stocks are bought
- updates the stock_transaction_history file
- makes necessary changes to the corresponding node of linkedlist (increase the number of stocks to existing or add a node if stock doesnt already exist in portfolio)

    - sellshare : User enters the stock to be sold, number of shares, and the minimum amount he/she wants to sell each share for. In order to let the user sell shares, function checks if the amount entered by user is less than or equal to the value (of each stock to be bought), and accordingly -
        - updates cashbalance after adding total price, stocks are sold for
        - outputs the result for user to know that the stocks are sold
        - updates the stock_transaction_history file
        - makes necessary changes to the corresponding node of linkedlist (reduce the number of stocks remaining or delete the node if none remains)
    - displayportfolio : displays the portfolio (creates iterator to traverse the linkedlist created) . Function also displays the cash balance and the total portfolio value
    - getBalanceFile : function to get the cash balance value from file and set to the cash balance variable of program. Used when an existing cash balance is to be retrieved from the file.
    - savePortfolioValue : saves the portfolio value passed to function, in portfoliovalue.txt, along with the time
    - getTotalPortfolioValue : function defined to retrieve the total portfolio value
    - updateTransactionFile : function to store portfolio transactions, along with their time in transaction file. Function also stores the stock, number of stocks, their value and the total value of stocks owned
    - plotgraph : function to plot graph that displays the variation in value of portfolio over time
- Private members -
    - File: fstream file to access (read/write) the files created/used in the functions
- Destructor : Updates the cash balance

BANKACCOUNT CLASS :

- Constructor :
    - Initializes the cash balance
    - If bank_transaction_history file is empty, then creates the header

- Public functions -
    - updatecashbalance : stores the updated cash balance to the cash balance file
    - deposit : to deposit the user entered amount to the account; adds this amount to the cash balance, updates the cash balance file and stores the deposit event in transaction file. Also makes sure that user doesnt try to deposit $0
    - Withdraw: to withdraw the user entered amount from account; reduces this amount from account, updates the cash balance file and stores the withdrawal even in transaction file. Also checks for invalid withdrawals when either the amount in less than 0 or user doesnt have that much amount in account
    - getBalanceFile : function to retrieve the cashbalance value from the cashbalance.txt file
    - Transaction_history : function to print the transaction history, from the transaction file, line by line
    - Viewbalance : function to retrieve the account balance
    - updateTransactionFile : function to store events deposit/withdrawal events, along with their time in transaction file. Function also stores the initial value of amount, amount deposited/withdrawn and final value of amount in the given format
- Private members -
    - File: fstream file to access (read/write) the files created/used in the functions
- Destructor : updates the cash balance

LINKEDLIST CLASS :

- Public functions -
    - addNode : function defined to add a node in the linkedlist
    - delNode : function defined to delete a node in the linkedlist
    - get_value : function to get the value of desired stock from one of the files (chosen randomly)
    - createIterator : function creates iterator, that would traverse the linkedlist
    - next : function to traverse the iterator through the linkedlist from start to end
    - getNumber : function to retrieve the number of stocks in a node, as the iterator traverses through the nodes of linkedlist
    - setIteratorZero : function to set the iterator to 0

    - getStock : function to retrieve the stock stored in a node, as the iterator traverses through the nodes of linkedlist
    - sortList : function to implement bubble sorting to sort the linkedlist sorted in decreasing order of total value of stock in portfolio
- Private members -
    - Node *iter : private member variable that stores the address of the node the iterator is at.
    - Node *start : variable to store the address of the start node of the linkedlist
    - Node *end : variable to store the address of the end node of the linkedlist
    - size : variable to store the size of the linkedlist

Summary -

The project implements account management system, that lets user access and operate on the bank account and stock account. Along with the basic functionalities of checking stock prices, buying/selling shares and depositing/withdrawing amount, user can also view cash balance, transaction history details and the graphical representation of varying stock values, and access the other functionalities implemented in the program.