**Name : Vikram Sahai Saxena**

**Net ID: vs799**

**16:332:503 Project: Account Management System**

**Project Description and Overview**

1. **Setup**
2. IDE : Visual Studio Code
3. Compiler: g++
4. C++ version: C++11
5. Install R
6. **Project source files**
7. Account\_Vikram\_Sahai\_Saxena.h and Account\_Vikram\_Sahai\_Saxena.cpp

* Purpose: Base Abstract class for accounts.
* Functions: Account management (constructor, virtual destructor, virtual functions, function to get the balance, credit and debit functions to modify the account balance value).
* Data Structures: Class with static cash balance member.

1. BankAccount\_Vikram\_Sahai\_Saxena.h and BankAccount\_Vikram\_Sahai\_Saxena.cpp

* Purpose: Implements a bank account, inheriting from Account.
* Functions: Bank account options display, balance display, deposits, withdrawals, write/display bank transaction history.
* Data Structures: Inherits from Account.

1. StockAccount\_Vikram\_Sahai\_Saxena.h and StockAccount\_Vikram\_Sahai\_Saxena.cpp

* Purpose: Manages stock account transactions.
* Functions: Stock Linked List management, buy/sell shares, display stock portfolio, write/display stock transaction history, plot R graph for change in portfolio value over time.
* Data Structures: Doubly linked list, Node class, Sort Implementation class, Bubble Sort class, Selection Sort class, Sort Context class, Map of stock symbols to their price per share.

1. main\_Vikram\_Sahai\_Saxena.cpp

* Purpose: Main program to test the application.
* Functions: Initializes different account types, user interface for bank operations.
* Data Structures: Vector for account pointers.

1. CurrentTime\_Vikram\_Sahai\_Saxena.h and CurrentTime\_Vikram\_Sahai\_Saxena.cpp

* Purpose: Manages current time using singleton pattern.
* Functions: Getting current time in specific format.
* Data Structures: Singleton class.

1. Result1.txt : Set 1 for Stock symbols and their price per share.
2. Result2.txt : Set 2 for Stock symbols and their price per share.
3. **Project output files**
4. bank\_transaction\_history.txt : stores the bank transaction details (Event, Amount, Date, Balance)
5. stock\_transaction\_history.txt : stores the stock transaction details (Event, Symbol, Number of Shares, Price per share, Total Value, Time)
6. portfolio\_history.txt : contains the portfolio information on program exit (Total portfolio value, Balance, Date/Time).
7. Rplots.pdf: contains plots for difference in portfolio value over time and actual portfolio value over time.
8. **Project source files working**
9. Account\_Vikram\_Sahai\_Saxena.h and Account\_Vikram\_Sahai\_Saxena.cpp

* Contains the static cash balance used by StockAccount and BankAccount classes. Contains a getter method for the balance.
* The balance is initially set to $10000. The constructor reads the latest balance present in portfolio\_history.txt and updates the account balance value with the read data.
* Credit function adds an amount to the balance.
* Debit function subtracts an amount from the balance. It fails if Debit amount exceeds the account balance.
* Contains virtual function to display account options. These are also implemented in the derived classes StockAccount and BankAccount.
* Contains pure virtual functions to write/display transaction history. These are implemented in the derived classes StockAccount and BankAccount.

1. StockAccount\_Vikram\_Sahai\_Saxena.h and StockAccount\_Vikram\_Sahai\_Saxena.cpp

* Contains a Node class that stores the share symbol, number of shares, pointer to previous node and pointer to the next node. It has a getter method to return the number of shares. Also, it has a static method to swap two nodes in-place by changing the previous and next links.
* The Doubly Linked List and Sorting classes are friend classes to Node to access the Node links and data.
* The Sorting classes implement a Strategy Design Pattern.
* The Doubly Linked List maintains the head, size of the list and the sorting context.
* The Doubly Linked list always remains sorted in decreasing order of number of stocks \* price per share.
* The Doubly Linked List updates its Nodes on adding/selling stock. If all shares of a stock are sold, the share is removed from the list.
* The destructor of Doubly Linked List class frees up the memory allocated to all the nodes and also deletes the sort context.
* The StockAccount class is a friend classes to Doubly Linked List so that the linked list sorting context set by the user can be used to call the appropriate sorting method.
* On program start, the StockAccount initializes the Doubly Linked List with data from stock\_transaction\_history.txt if present. Each line of stock\_transaction\_history.txt is read and if the line has a Buy Event, the corresponding symbol and number of shares are added to the Linked List. If the line has a Sell Event, the corresponding Stock node shares are reduced (Node is removed if the number of shares reached zero).
* StockAccount contains a function to get the map of stock symbols and their price per share by randomly selecting the price per share value from Result\_1.txt and Result\_2.txt.
* StockAccount implements the Stock Account display options. StockAccount displays the current stock price by using the map of stock symbols. It uses setw() function from <iomanip.h> to display the result with fixed width.
* StockAccount contains a function to display the Stock Account portfolio using the sorted doubly linked list. It uses setw() and setprecision() functions from <iomanip.h> to display the result with fixed width and precision.
* StockAccount buys shares from user-inputted amount by calling the add shares function from Doubly Linked List class. It then debits the amount from the bank account balance and writes the withdrawal transaction details into ank\_transaction\_history.txt. It writes the stock transaction details into stock\_transaction\_history.txt. It uses setw() and setprecision() functions from <iomanip.h> to write the result with fixed width and precision.
* StockAccount sells shares from user-inputted amount by calling the sell shares function from Doubly Linked List class. It then credits the amount to the bank account balance and writes the deposit transaction details into ank\_transaction\_history.txt. It writes the stock transaction details into stock\_transaction\_history.txt. It uses setw() and setprecision() functions from <iomanip.h> to write the result with fixed width and precision.
* StockAccount contains a function to plot an R graph for plotting the change in portfolio value over time and the actual portfolio value over time. It uses a system call to plotGraph.R to plot the graph in a pdf.
* StockAccount contains a function to allow user to select the sorting method for the Doubly Linked List of Stock nodes. It sets the sorting context between Bubble and Selection Sort. If the user does not select a sorting option, the default sorting context is set for Bubble Sort.
* The destructor of StockAccount class writes the total portfolio value, current balance, and the current date/time to portfolio\_history.txt. It uses setw() function from <iomanip.h> to write the result with fixed width.

1. BankAccount\_Vikram\_Sahai\_Saxena.h and BankAccount\_ Vikram\_Sahai\_Saxena.cpp

* Implements Bank Account display options.
* Displays current account balance by using the getter method from Account class. It uses setprecision() functions from <iomanip.h> to display the result with fixed precision.
* Deposits user-inputted amount to the balance by calling the Credit function from Account class. It then writes the deposit transaction details into ank\_transaction\_history.txt.
* Withdraws user-inputted amount to the balance by calling the Debit function from Account class. If the balance of the account is not sufficient to withdraw the amount, an error is displayed. On successful withdrawal, it writes the withdrawal transaction details into ank\_transaction\_history.txt. It uses setw() and setprecision() functions from <iomanip.h> to write the result with fixed width and precision.
* Displays bank transaction history by reading data from bank\_transaction\_history.txt.

1. main\_Vikram\_Sahai\_Saxena.cpp

* Contains the test program for the account management system.
* Uses a vector of Account pointers that are dynamically cast to StockAccount\* and BankAccount\* for the executing the banking operations.
* The pointers are freed on program exit.

1. CurrentTime\_ Vikram\_Sahai\_Saxena.h and CurrentTime\_ Vikram\_Sahai\_Saxena.cpp

* Uses a Singleton Design pattern to get the current date/time for use in Stock Account and Bank Account transactions. Also gives the current date/time in HH:MM:SS and MM/DD/YYYY formats.

1. **Design Patterns used**
2. Singleton: CurrentTime uses Singleton Design pattern by ensuring only one instance of it is present for time management.
3. Strategy: Sorting of the Doubly Linked List is implemented using Strategy Design Pattern in SortAccount.cpp. The Sort Implementation class is an Abstract class, that contains a pure virtual sort function. The sort function is implemented by derived Bubble Sort and Selection Sort classes, that sort the nodes of the list in-place by changing the node links. The user-inputted sort selection option is set as the sorting context for implementing the sort. The Context class executes the set context to call the appropriate sort.