Xinyu Lyu, RUID: 182007396

**A brief description of the Account Management System Project**

My project is about the account management system. Create an abstract class Account. Two derived class stockaccount and bankaccount with Cash\_Balance to connect them with each other. These two accounts are with all the functions fulfilling all the request from the project. The friend class realtime price class is used to provide the random price to two accounts. I use doublelinkedlist to implement the storage of portfolio along with others information such as the event, price, stock symbol, and so on. I also use different txt to store different records. When restart the project, it can keep all the data before close last time. I use three design patterns Bridge, adapter and Singleton. Use vectors to load the data in the result txt. In the main function I load all the data in the txt files for the recovery of the necessary data.

The base and abstract **class Account.**

1. It is inherited from the class global.

It is a design pattern called: Singleton. It can be a substitute for a global variable, ensuring a class has only one object instance, and provide an access method.

2. Account(); Construction functions.

virtual ~Account();Destruction functions.

3. double getCash\_Balance(); Get cash balance.

void setCash\_Balance(double); Set cash balance.

4. void View\_Portfolio\_Graph(); Interface to the MATLAB, read the graph data to MATLAB from Matlab\_Graph.txt and output the graph.

5 void Write\_Graph\_Data(DoubleLinkList & list); Write the sum of stock portfolio and cash balance to Matlab\_Graph.txt using the double linked list structure.

6. void virtual bridge() {}; The bridge design pattern, the implement part.

**Class Bank account** inherited from the base and abstract class Account

1. BankAccount(); Construction.
2. ~BankAccount(); Desdurction.
3. void View\_Account\_Balance(); Get cash balance from Account
4. void Withdraw(DoubleLinkList &); Withdraw money using get and set cash balance using the double linked list structure.
5. void Deposit(DoubleLinkList &);
6. void Print\_Bank\_History(); Print bank transcation history read the data from Bank\_Transaction\_History\_Xinyu\_Lyu.txt.
7. void Read\_CashBalance(); Read the event number date cash balance information from Bank\_Transaction\_History\_Xinyu\_Lyu.txt to the bank account.
8. void Write\_Bank\_History(string,double,double); Write bank transcation history to the Bank\_Transaction\_History\_Xinyu\_Lyu.txt.
9. virtual void print\_Title(); This is an adapter design pattern for different clients they have different requests for to print the history, this function is used to adapt the different print requests. Very simple but necessary.

**Class Stock Account** inherited from the base and abstract class Account. Has the friend class RealTime Price which gets realtime price.

1 .StockAccount();

2.~StockAccount();

3. void Display\_Stock\_Price(RealTime\_Price&); Display the price for a specific stock if found getting the realtime price from the class RealTime Price.

4. void Display\_currentPortfolio(DoubleLinkList &); Display the portfolio with the information of stock name, number, price per share and total value.

5. void Buy(DoubleLinkList&,RealTime\_Price& ); Buy shares for the specific stock and give the failed reason, write graph data and transaction history using the Double Linked List structure with the real time price from the friend class RealTime Price.

6. void Sell(DoubleLinkList&,RealTime\_Price&); The same with buy.

7. void Print\_Stock\_History(); Print the stock transaction history, opening Stock\_Transaction\_History\_Xinyu\_Lyu.txt and read.

8. void Write\_History(string,string,int,double,double); Write the stock transaction history to Stock\_Transaction\_History\_Xinyu\_Lyu.txt.

**Class RealTime\_Price**

1. RealTime\_Price();

2. ~RealTime\_Price();

3. vector<string> Stock\_Company; Used to set the vectors of stock company from result txt

4. vector<double> Result\_One\_Price,Result\_Two\_Price; The vectors for the price in the result txt

5. Double getStock\_Price(string company); Get random price with the help of the function of getRealTime\_Price

6. double getRealTime\_Price(int index);

7. void set\_Result\_Vectors(vector<string> company, vector<double> Price1, vector<double> Price2); Set the vector with the information of company, and price form the result one and the result two.

**Class DoubleLinkList**

inherited from realtime price class, getting random price. Friend class to Node class.

1. DoubleLinkList(RealTime\_Price & price);

2. ~DoubleLinkList();

3. bool isEmpty() const; Tell if the double linked list is empty .

4. void insertNode(Node\*); Insert node to the double linked list

5. void deleteNode(string); Delete node from double linked list

6. void sortList(); Bubble sort the list.

7. bool isExist(string); Tell if a company is exist in the list

8. void changePrice(); Randomly choose the price from the result one or result two

9. int changeNumber(string,int&); Change the number for a node

10. int getNumber(string); Get the number information of the node

11. double getTolPortfolio(); Get the total portfolio

12. void print();

13. void Write\_currentPortfolio(); Write the portfolio information to the Portfolio\_Xinyu\_Lyu

**Class Node:**

The basic get set functions of number, company name, price.

**Class print:**

The adapter design pattern, implements in the class bank account to fulfill different requests for print from different client.

**Bank\_Transaction\_History\_Xinyu\_Lyu.txt** is used to record the bank history with event, total amount, date, cash balance.

**Matlab\_Graph.txt** is used to store the total portfolio data for graph.

**Portfolio\_Xinyu\_Lyu.txt** is used to record the Portfolio with the stock in your stock account with company symbol, numbers of shares and price and total value.

**Stock\_Transaction\_History\_Xinyu\_Lyu.txt** is just like the Bank\_Transaction\_History\_Xinyu\_Lyu.txt but for the stock account.