**503 final project (Account Management System) description**

Bowei Kang

***I. Header file and Source code files (8)***

1. Account\_BoweiKang.h and Account\_BoweiKang.cpp: abstract base class

2. Node\_BoweiKang.h: node used in linked list of StockAccount

3. StockAccount\_BoweiKang.h and StockAccount\_BoweiKang.cpp: derived class 1

4. BankAccount\_BoweiKang.h and BankAccount\_BoweiKang.cpp: derived class 2

5. main\_BoweiKang.cpp: implementation

***II. text files (5)***

1. balance.txt: store cash balance

one record, one dimension: balance(double)

establish when instantiate any object

update when balance changes (buy, sell, deposit, withdraw)

relating functions: readBalance(), updateBalance()

2. portfolio.txt: record stock portfolio you hold

two dimensions: stockname(string), shares(int)

establish when instantiate StockAccount object

update after successfully stocks buying and selling   
 relating functions: readPortfolio(), updatePortfolio(), printPortfolio()

3. stock\_transaction\_history.txt: record buying and selling stock history

5 dimensions: event(bool), name(string), price(double), value(double), time(string)

establish after first transaction happens or first print history operation

update after successfully stocks buying and selling

relating functions: appTransStock(), printTransHistory()

4. bank\_transaction\_history.txt: record history of transaction that changes balance

4 dimensions: event(int), amount(double), value(double), time(string)

establish after first transaction happens

update after successfully stocks buying and selling, cash deposit and withdraw

relating functions: appTransCash(), printTransHistory()

5. total\_value.txt: record total value (cash + stock portfolio)

prepared for value variation plot

2 dimensions: value(double), time(string)

establish when first print or update total value

update after successfully stocks buying and selling

relating functions: updateTotalValue(), printTotalValue()

***III. class Account***

*A. data member (1)*

1. double balance: cash balance

*B. function member (7)*

1. constructor: Read latest balance.

2. readBalance: newly build if there is no balance.txt, set balance to initial value

and write it to file; else read current cash balance from file

3. getBalance: get balance

4. setBalance: set balance

5. updateBalance: update balance.txt after transaction which can change balance

6. printTransHistory: print transaction history (pure virtual)

7. printPortfolio: print current portfolio and balance (pure virtual)

***IV. class StockAccount***

*A. data member (5)*

1. Node \*myHead: first pointer of linked list

2. Node \*myTail: last pointer of linked list

3. map <string, double> currentPrice: stock price after randomly selecting

4. map <string, double> price1/ price2: stock price from results1/2.txt

*B. function member (21)*

1. StockAccount(): constructor. Set myHead(0), myTail(0); read latest balance (if not   
 exist then build it); set currentPrice; read current portfolio from file

2. ~StockAccount(): destructor. Delete nodes in linked list

3. map <string, double> readStockPrice(string): read stock price into map from file

4. void setStockPrice (): set price1 and price2 using readStockPrice

5. map <string, double> randomSelectPrice(): randomly select stock price, return a map

6. void readPortfolio(): read portfolio information from file

7. void addToEnd(Node \*): insert node at end of linked list

8. double calculateTotalValue (): calculate total account value (stock + cash), used in   
 function updateTotalValue()

9. void printPrice(string &): print current price of specific stock user defined

10. virtual void printPortfolio(): print current portfolio. (1) Get current price. (2) Sorting.   
 (3) Cout current balance, portfolio (name + shares + price + value), total account   
 value

11. virtual void printTransHistory(): print transaction history from txt file. If file does not   
 exsit, then build it and cout relating information to user

12. void printTotalValue(): print total value and calling Matlab to plot latest 100 values.   
 When first establish txt file, the initial value equals to the initial cash balance, i.e.,   
 10K.

13. void buyStock(Node \*, double): one purchasing stock transaction. (1) Update   
 balance and stock price. (2) Guarantee the stock user claims in the price list. (3)   
 Guarantee the current price is not higher than user’s tolerance. (4) Guarantee the   
 current balance is enough to buy. (5) If the stock is already in current portfolio, then   
 increase its shares; add to end of linked list, otherwise. (6) update balance for data   
 member and txt file; update 2 transaction history txt file, portfolio txt file, totalvalue   
 txt file. (7) Cout transaction summary information.

14. void sellStock(Node \*, double) ): one selling stock transaction. (1) Update   
 balance and stock price. (2) Make sure the current portfolio (linked list) is not empty.   
 (3) Guarantee the stock user claims in the price list. (4) Guarantee the current price is   
 not lower than user’s tolerance. (5) Guarantee the current shares are enough to sell.   
 (6) If sell out one stock, then delete corresponding node in linked list; reduce shares   
 otherwise. (7) update balance for data member and txt file; update 2 transaction   
 history txt file, portfolio txt file, totalvalue txt file. (7) Cout transaction summary   
 information.

15. void appTransStock(bool, Node \*): append transaction history to file (0-buy, 1-sell)

16. void appTransCash(int, double): append transaction history to file (0-buy, 1-sell)

17. void updatePortfolio(): update portfolio file after transaction

18. void updateTotalValue(): update total portfolio value after stock transaction, use   
 function calculateTotalValue()

19. void swap(Node \*, Node \*): utility function for sorting

20. Node \* partition(Node \*, Node \*): partition for quick sorting

21. void quickSort(Node \*, Node \*): quick sort, used in

***V. class BankAccount***

*A. function member(7)*

1. BankAccount(): constructor. read latest balance (if not exist then build it)

2. virtual void printPortfolio(): print current cash balance

3. void deposit(int): one deposit to bank account. (1) Read balance. (2) Increase balance.   
 (3) Update balance, transaction history and totalvalue in file.

4. void withdraw(int): one withdraw from bank account. (1) Read balance. (2) Make sure   
 there is enough money in current balance to withdraw. (3) Reduce balance. (4)   
 Update balance, transaction history and totalvalue in file.

5. virtual void printTransHistory(): print transaction history

6. void appTransCash(int, double): append transaction history to file (2-deposit, 3-  
 withdraw)

7. void updateTotalValue(bool, double): update total portfolio value to file

***VI. Main function***

*A. instruction menu (3)*

1. void menuMain()

2. void menuStock()

3. void menuBank()

*B. operation (3)*

1. void operationMain(StockAccount &, BankAccount &);

2. void operationStock(StockAccount &, BankAccount &);

3. void operationBank(StockAccount &, BankAccount &);