HW #6 (Account Portfolio帳戶組合)

- You will design a set of classes for storing information regarding bank accounts, along with a class that will manage a list of accounts.
- Data must be imported from files for storage in the list, and summary reports with computed balances must be saved to output files.

- You are expect to do the work in C++ (or Java if you prefer).

HW #6 (2)

Classes Details

- Design a set of classes to store bank account information. There should be one base class called **Account** to store common data about bank accounts, and three derived classes that divide the set of accounts into three categories: **Savings**, **Checking**, and **Investment** accounts.
- All data stored in these classes should be **private** or **protected**. Any access to class data from outside (besides accessing base class data from derived classes) should be done through **public** member functions. Make use of **virtual** and **abstract** functions wherever appropriate.

HW #6 (3)

- The base class **Account** should allocate storage for the following data (and only this data): Account holder's **first name** (you may assume it is up to 20 characters long), account holder's **last name** (you may assume it is up to 20 characters long), the **type** of account (**Savings**, **Checking**, or **Investment**).
- Each class (base class and derived classes) should have a function that will compute and return the account's projected balance based on the stored information. All balances are in **dollars**.
- Here is the information that needs to be stored for each account, along with the breakdown for computing each projected balance:

HW #6 (4)

- **Savings Account**
- Stores: -- Current Balance
  -- Interest rate

  ProjectedBalance = CurrentBalance * (1 + Interest rate)

- **Checking Account**
- Stores: -- Current Balance

  ProjectedBalance = CurrentBalance + 0.1

HW #6 (5)

- **Investment Account**
- An investment account is made up of **five** Exchange-Traded Funds (ETFs). The investment class should store the information for the five ETs. Each ETF has the following four pieces of information: 1) amount invested (A), 2) initial value per share (IVS), 3) current value per share (CVS), and 4) interest rate (I).

- The projected balance of the investment account is given by the current value (CV) of the five ETFs + their dividends.

HW #6 (6)

- The current value of an ETF is computed as follows:

$$CV = (A / IVS) * CVS$$

- The dividends yielded by each ETF is computed as follows:

$$DIV = I * A$$

- The projected balance of an investment account is computed as follows: (5 ETFs)

$$ProjectedBalance = \sum_{ETFs} (CV + DIV)$$

HW #6 (7)

- Write a class called **Portfolio**, which will be used to store the list of various accounts, using **a single array of flexible size**.
- Note that this array will act as a **heterogeneous** list, and it will need to be dynamically managed. Each item in the array should be an **Account** pointer (so you will have a dynamically allocated array of **Account** pointers, i.e., **Account** alist**), which should point to the appropriate type of object.
- Your list should only be big enough to store the accounts currently in it.

HW #6 (8)

- Your **Portfolio** class needs to have at least the following public functions available:

- **Portfolio()** -- default constructor. Sets up an empty list of accounts.

- **~Portfolio()** -- destructor. Needs to clean up all dynamically allocated data being managed inside a **Portfolio** object, before it is de-allocated.

HW #6 (9)

- **bool importFile(const char* filename)**
- This function should add all data from the file given as parameter into the internally managed list of accounts. The file format is specified below in this write-up. Note that if there is already data in the list from a previous import, this call should add MORE data to the list. Records should be added at the end of the given list in the same order they appear in the input file.

- If the file does not exist or cannot be opened, return false to indicate failure of the import. Otherwise, after importing the file, return true for success.

HW #6 (10)

- **bool createReportFile(const char* filename)**
- This function should create a banking report and write it to the given output file (filename given in the parameter). The format of the banking report file is described below in this write-up. If the output file cannot be opened, return false for failure. Otherwise, after writing the report file, return true for success.

HW #6 (11)

- **void showAccounts() const**
- This function should print to the screen the current list of accounts, one account per line. The only information needed in this printout is last name, first name, account type, and current balance (in the case of investment accounts, instead of current balance you will print **the total amount invested**, i.e., the sum of the invested amounts (A) in the five ETFs). Format this output so that it lines up in columns.

HW #6 (12)

- Write a main menu program in a separate file called **main.cpp** that creates a single **Portfolio** object and then implements a menu interface to allow interaction with the object. Your main program should implement the following menu loop (any single letter options should work on both **lower**- and **upper**-case inputs):

***** Portfolio Management menu *****

I Import accounts from a file

S Show accounts (brief)

E Export a banking report (to file)

M Show this menu

Q Quit program

HW #6 (13)

- The import and export options should start by asking for user input of a filename (you may assume it will be up to 30 characters long, no spaces), then call the appropriate class function for importing accounts from a file or printing the banking report to a file, respectively.
- The program must print a message indicating the task was unsuccessful if the import/export fails.

- The “Show accounts (brief)” option should call the class function that prints the brief account info to screen (names, account type, and current balance/invested amount only).

HW #6 (14)

- The “Show this menu” option should re-display the menu.

- “Quit” should print “Goodbye” and exit program. (Until this option is selected, keep looping back for menu selections.)

HW #6 (15)

- **Input File** -- The first line of the input file to import accounts from will contain the number of accounts listed in the file. This will tell you how many accounts are being imported from this file.

- After the first line, every set of two lines constitutes a single account entry. The first line of an entry is the account holder's full name, in the format lastName, firstName. Note that a name could include spaces -- the comma will delimit last name from first name.

HW #6 (16)

- The second line will contain the type of account ("Savings", "Checking", or "Investment"), followed by a list of account information, all separated by spaces. There will be no extra spaces at the end of the line in the file. The order of the account information for each type is as follows:

- for Savings accounts: current Balance, then Interest rate (the values on a line will be separated by space).

- for Checking accounts: current Balance.

HW #6 (17)

- for Investment accounts: 1) amount invested for ETF1, 2) initial value for ETF1, 3) current value for ETF1, 4) interest rate for ETF1;

  1) amount invested for ETF2, 2) initial value for ETF2, 3) current value for ETF2, 4) interest rate for ETF2;

  1) amount invested for ETF3, 2) initial value for ETF3, 3) current value for ETF3, 4) interest rate for ETF3;

  1) amount invested for ETF4, 2) initial value for ETF4, 3) current value for ETF4, 4) interest rate for ETF4;

  1) amount invested for ETF5, 2) initial value for ETF5, 3) current value for ETF5, 4) interest rate for ETF5. (The values on a line will be separated by space.)

Example 1 of input file (test1.txt)

```
4
Alfaro, Emily-Grace
Savings 900.89 0.001
Morgan, Arthur
Checking 89
Dipwart, Marvin
Investment 40 80 80.84 0.02 45 20 20 0.2 150 76.3 76 0.05
    45 80 76 0.07 408 5 100 0.9
van Houten, Milhouse
Savings 45.80 0.79
```

Example 2 of input file (test2.txt)

```
2
Polar Bear, Maya
Checking 40.90
Polar Bear, Maya
Savings 60.72 7.8
```

HW #6 (18)

- **Output File** -- The output file that you print should list each account holder's name (firstName lastName - no extra punctuation between), Initial Balance, and projected balance. All the balances should be printed to **two** decimal places.

- The Output should be separated by subject, with an appropriate heading for each section, and each account's information listed on a separate line, in an organized fashion.

HW #6 (19)

- The order of the accounts within any given category should be the same as they appear in the portfolio.
- Data must line up appropriately in columns when multiple lines are printed in the file.
- At the bottom of the output file, print the account distribution of ALL accounts in the portfolio (i.e., how many Saving accounts, Checking accounts, and Investment accounts there are) and the average projected value per type of account.

Sample run of Portfolio management program.

It includes screen input and output, where the keyboard input start with ">" here to differentiate output from input.

```
*** Portfolio Management menu ***
I       Import accounts from a file
S       Show accounts (brief)
E       Export a banking report (to file)
M       Show this menu
Q       Quit Program

> i
Enter filename: test1.txt
```

```
> s
Holder                          Type          Balance

Alfaro Emily Grace              Savings       900.89
Morgan Arthur                   Checking      89.00
Dipwart Marvin                  Investment    688.00
van Houten Milhouse             Savings       45.80

> l
Enter filename: banking.txt
Invalid file. No data imported

> l
Enter filename: test2.txt
```

```
>S
Holder                          Type          Balance

Alfaro Emily Grace              Savings       900.89
Morgan Arthur                   Checking      89.00
Dipwart Marvin                  Investment    688.00
van Houten Milhouse             Savings       45.80
Polar Bear Maya                 Checking      40.90
Polar Bear Maya                 Savings       60.72

> E
Enter filename: summary.txt

> q
Goodbye!
```

Sample output (summary.txt)

Banking Summary

Saving Accounts

Holder's Name	Initial Balance	Projected Balance
Emily-Grace Alfaro	900.89	901.79
Milhouse van Houten	45.80	81.98
Maya Polar Bear	60.72	534.34

Checking Accounts

Holder's Name	Initial Balance	Projected Balance
Arthur Morgan	89.00	89.10
Maya Polar Bear	40.90	41.00

Investment Accounts

Holder's Name	Initial Balance	Projected Balance
Marvin Dipwart	688.00	8825.23

Overall Account distribution

Savings:	3	-	506.04
Checking:	2	-	65.05
Investment:	1	-	8825.23