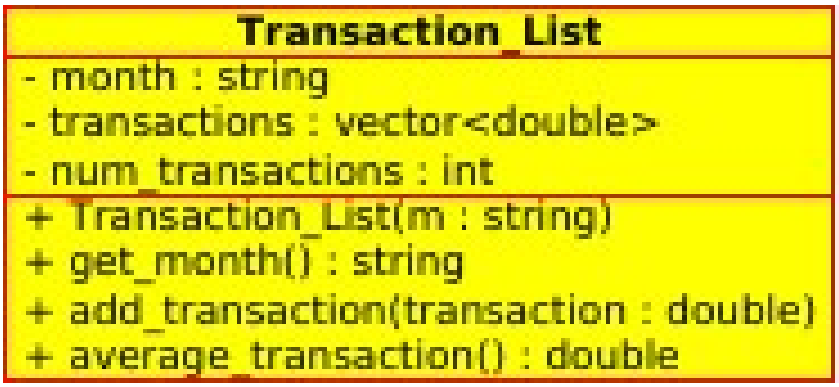
CSE 1325-001 Homework #3 Using Classes

In this homework, you will create a class that stores a list of transactions and computes the average transaction for a given month. The user must input the name of the month and the transactions into the terminal.

The Transaction_List Class

First you will create two C++ files called abc1234_Transaction_List.h and abc1234_Transaction_List.cpp. Below is a UML class diagram that shows the basic design of the Transaction_List class.



This class contains a list of transactions for a certain month. Each transaction is represented a double in the vector transactions. The constructor sets the month equal to the m and the num_transactions to 0. get_name returns the name of the month. add_transaction adds a new transaction to the list of transactions and 1 to num_transactions. average_transaction returns the average transaction for that month. If there are no transactions, then this should return 0.

Testing the Transaction_List

Next you will test your Transaction_List class by creating a new C++ file called abc1234_test.cpp, which consists of just a main method. The main function must be able to do the following three tests.

- Test #1
    - Create a Transaction_List object named “May” and add 4 transactions: 150, 225.3, 78.9, and 523.99
    - Get the Transaction_List’s name. If the month is not “May”, display a useful error message to the terminal (cout).
    - Compute the average transaction for the month. If the average is not 244.5475, display a useful error message to the terminal (cout).
- Test #2
    - Create another Transaction_List called “March” and add 4 transactions: 150, 225.3, 78.9, and 523.99
    - Get the Transaction_List’s name. If the month is not “May”, display a useful error message to the terminal (cout).
    - Compute the average transaction for the month. If the average is not 244.5475, display a useful error message to the terminal (cout).

- Test #3
    - Create a Transaction_List object named “May” and add 4 transactions: 150, 225.8, 78.9, and 523.99
    - Get the Transaction_List’s name. If the month is not “May”, display a useful error message to the terminal (cout).
    - Compute the average transaction for the month. If the average is not 244.5475, display a useful error message to the terminal (cout).

Take a screenshot of your terminal that shows the output from all 3 tests. Name this screenshot abc1234_test.png. If you have multiple screenshots, name them abc1234_test_1.png, abc1234_test_2.png, etc.

The main program

Third, write one more C++ file called abc1234_main.cpp. This main program consists of a main function that should do the following: (you may have to write more loops then what is listed below, or more functions then just main)

- Create a list (vector) of Transaction_Lists
- Prompt the user for the name of a month and take it in as input.
- Create a Transaction_List object with the name you just received as input.
- Create a loop that askes for a transaction and takes it in as input.
    - If the input is -999, exit the loop.
    - If the input is not -999, add it to the list of transactions.
- When the loop is terminated, add this month to the list of Transaction_Lists.
- Ask the user if there is another month he/she want to put in (Y/N)
    - If yes, go back to the second bullet point.
    - If no, print out the name of each Transaction_List (the month name) and the average, each on a different line..

Take a screenshot of your code running in terminal. Name this screenshot abc1234_main.png. If you have multiple screenshots, name them abc1234_main_1.png, abc1234_main_2.png. etc.

When grading this, you do not know how many months the GTA will put in, so make sure your code can handle different number of months.

Makefile

You are REQUIRED to provide a simple makefile that can build and execute your test code given the command “make test”, and build and execute your main given the command “make”. The sample course code from lecture 5 is a good example of a makefile to use for the homework.

You will put all relevant files for the full credit portion of the homework in a folder called full_credit.

Bonus 1 (5 pts)

Copy your abc1234_Transaction_List.h, abc1234_Transaction_List.cpp, and your abc1234_main.cpp into a separate folder called bonus_1. Change your Transaction_List Class so the double num_transactions is not mentioned in your code (delete all references to it). Find some way to still compute the average over the number of transactions submitted.

Once that is complete, make a new makefile for this bonus. Run your main function take screen shots of your code, using the same naming conventions as mention above. Place those screen shots in the bonus_1 folder.

Bonus 2 (15 pts)

For this, you will be modifying the provided UML diagram in Umbrello. The provided UML class diagram is called Transaction_List.xmi. You are to do the following the UML diagram (double clicking the class diagram opens the properties window):

- Include a private vector <string> called person. This list will correspond to the list of transactions and will represent who rang up each transaction.
- Modify the add_transaction() function to take in two arguments, one called high, one called p. The add day function will now add the input to the list of highs and name of the p into the list of person.
- Next, add a method called bonus. This function will return the name of the person who rang up the most transactions in that month. Since that person rang up the most transactions, that person will bet a bonus.

You will save this UML diagram to a new folder called bonus_2.

Copy your abc1234_Transaction_List.h, abc1234_Transaction_List.cpp, and abc1234_main.cpp file from your full credit directory into bonus_2.

Next you will modify your Transaction_List class to reflect any changes made to the UML diagram.

Lastly, modify your abc1234_main.cpp program to allow the user to input both the transactions and the names. If the transaction is still -999, quit the loop. When done taking input, output the name of the month, the average transaction, and who got the bonus, each on a different line. Take screen shots of your code working, and place those screen shots the bonus_2 folder. Follow the same naming conventions as above for the screenshots.

Deliverables

You will submit your code and screenshots via Blackboard. You will upload a zip file, named "abc1234_HW3.zip", contain 1 folder (3 if you did all the bonus)

- full_credit folder
    - abc1234_Transaction_List.h
    - abc1234_Transaction_List.cpp
    - abc1234_test.cpp
    - abc1234_main.cpp
    - abc1234_test.png (or multiple if multiple screenshots were taken)
    - abc1234_main.png (or multiple if multiple screenshots were taken)
    - makefile
    - Instructions for compiling and running (either in comments via blackboard or in a README file)
- bonus_1 folder
    - abc1234_Transaction_List.h
    - abc1234_Transaction_List.cpp
    - abc1234_main.cpp
    - abc1234_main.png (or multiple if multiple screenshots were taken)
    - makefile
    - Instructions for compiling and running (either in comments via blackboard or in a README file)
- bonus_2 folder
    - abc1234_Transaction_List.xmi (the UML file)
    - abc1234_Transaction_List.h
    - abc1234_Transaction_List.cpp
    - abc1234_main.cpp
    - abc1234_main.png (or multiple if multiple screenshots were taken)
    - makefile
    - Instructions for compiling and running (either in comments via blackboard or in a README file)

Full credit files named incorrectly result in a loss of 5 points each.