CSCI 1730 Assignment 02

First Classes

Document Last Updated: September 14, 2020
Due: 9/20/2020 11:55 pm

This assignment is to be completed individually.

Introduction

This assignment will have you apply your knowledge of classes and objects in C++ to implement some basic functionality of an Account class. This account class will maintain both a balance(double) and an account name(std::string).

You have been provided with skeleton code for you to complete as well as a makefile which will be used to compile your code.

It is very important that you do not change the names of any file, function, or class. It is also very important that you do not change the signatures of any function.

Class invariant(s)

- The account should never be allowed to have a less-than-zero balance.

  $0 <= balance$

- The name of the account should always be at least 5 characters and not more than 20 characters.

  $5 <= |accountName| <= 20$

  - If an attempt is made to use an account name shorter than 5, the name should be set to "Dawgs" (case-sensitive).
  - If an attempt is made to use an account name longer than 20, the name should be set to the first 20 characters of the argument.

General notes

You are not permitted to use pointers on this assignment.

Individual implementation

You are expected to complete this exercise individually.

1 Problem / Exercise

Account Class (Account.cpp)

The account class will represent a simple bank account. You will implement the basic functionality, the internal representation, and a few member functions.
The basic skeleton for your program is outlined below:

```
#ifndef ACCOUNT_H
#define ACCOUNT_H

```

```
#include <string>

class Account {
public:
  // add constructors here -- see description in assignment

  double getBalance() {
    // add code here -- a basic getter, return the balance of the account

    return 0;                   /* replace this */
  }

  void setBalance(double balance) {
    // add code here -- a basic setter, return nothing.
  }

  double deposit(double depositAmount) {
    // your code should check to ensure that depositAmount is positive (e.g.
    // deposit 100 dollars). It should only update the account balance if it is
    // positive. Returns balance after any changes

    // add code here

    return 0;                   /* replace this */
  }

  double withdraw(double withdrawalAmount) {
    // This method should ensure that the withdrawalAmount is positive (e.g.
    // withdraw 100 dollars). It should only update the account balance if the
    // account can cover the withdrawal. Returns balance after any changes

    // add code here

    return 0;                   /* replace this */
  }

  double interest(double percent) {
    // This method should accept percent (positive or negative). (e.g. to grow
    // the account by 10% you would call interest(0.1). To decay the account by
    // 20% you would call interest(-0.2)) Returns balance after any changes

    // add code here

    return 0;                   /* replace this */
  }

  std::string getName() const {
    // add code here

    return "";                  /* replace this */
  }

  void setName(std::string newName) {
    // Your code should ensure that the name of the account has a max of 20
    // characters. If the newName passed to this function is longer than 20, you
    // should set the account name to be the first 20 characters of the
    // argument.
    // add code here
  }

  // write the transferTo method here -- see assignment
  bool transferTo(double amount, Account& otherAccount) {
    // add code here

    return false;               /* replace this */
  }

private:
  // add more code here -- see description in assignment
};

#endif
```

constructor note: You should implement one constructor–a single argument constructor which takes the account name as a parameter. This constructor will set the data member appropriately and the initial balance of the account to zero.

private notes: You should add whatever data members you need to implement this class in the **private** section of the above code.

transferTo method. You should write a member function called `transferTo`. Its first parameter should be an double ('amount')–the amount to transfer. The second parameter should be (a reference to) the 'otherAccount'–the account to transfer the money to. **note:** You should check that the account this method is called on has enough funds to cover the transfer. If it does not, the `transferTo` method should return false. If the account does have adequate funds, the money should be transferred to the 'otherAccount' and the `transferTo` function should return true.

other notes: You may want to read up on std::string and bool values in c++ again.

2 Requirements

If your program does not compile on odin, then you'll receive a grade of 0 on this assignment. Otherwise, your program will will be graded using the criteria below.

Your program will be compiled with the `-std=c++11 -Wall and -pedantic-errors` options enabled. You should ensure that your program compiles without warnings and without errors with these options.

compilation without warnings/errors	20 points
correct program behavior on `odin`	80 points
any renamed file, function or modifed signature	-100 points
compilation error(s)	-100 points

You must test, test, and re-test your code to make sure it compiles and runs correctly on `odin` with any given set of valid inputs.

3 Submission

Before the deadline, you need to submit your assignment. You will still be submitting your project via Odin. Make sure your work is on `odin.cs.uga.edu` in a directory called `LastName-FirstName-assn02`. From within the parent directory, execute the following command:

```
$ submit LastName-FirstName-assn02 csci-1730
```