**Programming Assignment 3**

**Purpose**: Demonstrate the use of class inheritance and operator overloading

Using Programming Assignment 2 class **bankAccount** as a basis, do the following:

Derive the class **checkingAccount** from the class bankAccount. This class inherits all members of the class **bankAccount**. In addition, add data members to contain:

**Allowed minimum balance – have the allowed minimum balance default to $5.**

**Monthly Service Charge – have the monthly service charge default to $1**

**All data members are to be private.**

**All base class data members are to remain private.**

For the **checkingAccount** class, write the following methods:

Set and get methods for allowed minimum balance

Set and get methods for monthly service charge

Proper constructor(s) as necessary

**In checkingAccount, Override the** bankAccount **withdraw** method to check for minimum balance. If the withdrawal causes the balance to fall below the allowed minimum balance, apply the monthly service charge. If the withdrawal and service charge take the balance below zero, deny the withdrawal.

**In checkingAccount, Override the << (put to) operator** to print the information in the account. For example, if you create a **checkingAccount** object named account1, in the client program (test program), you would be able to code: *cout << account1[i]*. **The print method will no longer be called “directly”, by the test program, to print the account information**. The << operator will be used. Include in the output the new fields allowed minimum balance and monthly service charge.

**In bankAccount, Override the + (plus) operator** to add the amount to the balance of the checking account. For example, if you create a checkingAccount object named account1, in the client program (test program) you would be able to write *account1[i] + 100.0*. **THIS WILL REPLACE the deposit method, i.e., the deposit method will no longer be used to deposit into the checkingAccount**. You may leave it in the code; however, you must no longer call the method.

You may need to add functions to the base class to allow the derived class access to private data.

Modify the test program to **ask** for, retrieve, input and set the new values for **minimum allowed balance** and **monthly service charge** for the checking account. If zeros are entered, use the defaults specified above. Create **checkingAccount** objects instead of **bankAccount objects.**

**Remember: Replace** the call to the **print** method with the **<<** operator you’ve overridden. **Replace** the call to the **deposit** method with the  **+** operator you’ve overridden.

Except where indicated in this Assignment, all other requirements of Assignment 2 still apply.

EXTRA CREDIT:

(1) Override the + operator in the **derived** class (2 points).

(2) In the overridden << operator code, print the base class data without calling a base class print method (2 points)

**All classes** must have a class definition header file (\*.h) and a class implementation file (\*.cpp) for each class. You must have a main program that uses these classes (\*.cpp). You must create a project in Dev C++, Codeblocks or other IDE. You must submit all header files, source files (\*.cpp) and the project file (\*.dev/\*.cbp). Submit a README file that contains: The names of all your files including all the files in the project; how to compile; and how to run.

**All assignments are to be submitted to the appropriate Assignment posting in Canvas.**

**Each solution is to be uniquely your own; student collaboration not allowed.**

**Submit your code files (\*.cpp, \*.h, \*.dev/\*.cbp), and readme files**. Program **must** compile and all files must be submitted to receive credit. DO NOT submit the executable (\*.exe).

**If you had points deducted from Programming Assignment 2 and correct the issues in Programming Assignment 3, I will return 100% of points lost to Programming Assignment 2 grade not attributed to late penalties and missing readme files.**

**A program that does not compile will receive a zero (0).**

You may zip all your files and submit the zip file. A zipped submission will receive 1 point extra credit.

The code file **must** contain the following as documentation. If the following is not included, **10% will be deducted**:

The name of your C++ file

Your name

Some kind of date, either the due date or the date you finished

The type of input

The type of output

A brief description of the algorithm or purpose of the program

For example:

/ \* Program name: assignment1.cpp

Author: Pam Smith

Date: 8/25/21

Input: requests an input and output file name from the user (inputfile.txt provided).

Output is name of your choice

Output: displays output on the console and writes output to file name provided

Description: This program translates a word or phrase using the ROT13 cipher

\*/

**All** procedures and functions should be documented with the following information:

A brief description of the purpose of the function or procedure

Signature : return type, name, parameters (including type)

A list of the parameters and what each represents

A description of the function return type as applicable

Precondition(s) as applicable

Postcondition(s) as applicable

For example:

/\*

myfunction outputs the values contained in the input parameters.

myfunction returns no value; it takes an integer parameter as input and a character parameter as input

The integer parameter contains the ordinal number representing a letter in the alphabet

The character parameter contains a letter of the alphabet

No value is returned

Precondition: the integer value must contain a valid number representing the ordinal number of the letter contained in the character parameter; the character parameter must contain a value representing a letter of the alphabet

Postcondition: there are no post conditions

\*/

void myfunction (int a, char b) {}