Assignment # 5 DUE: Specified on Canvas

**Assignment Purpose**

This lab will give you practice regular expressions while using classes, polymorphism, iterators, and inheritance.

You can work in your groups (2 members). If your partner is not present in class, each will need to complete the assignment on your own by the due date/time.

**Mandatory Instructions**

In this assignment, you will develop a WinForms application that will exhibit polymorphic behavior.

First, ~~create an inheritance hierarchy that a bank might use to represent customers’ bank accounts~~.

~~All customers at this bank can deposit (i.e., credit) money into their accounts and withdraw (i.e., debit) money from their accounts. More specific types of accounts also exist.~~

~~Savings accounts, for instance, earn interest on the money they hold. Checking accounts, on the other hand, charge a fee per transaction.~~

~~Create a public interface~~ *~~I~~****~~Account~~*** ~~and derived classes~~ **~~SavingsAccount~~** ~~and~~ **~~CheckingAccount~~** ~~that realize the~~ *~~I~~****~~Account~~*** ~~interface~~.

~~The~~ *~~I~~****~~Account~~*** ~~interface should include three method prototypes:~~ ***~~Debit~~***~~,~~ ***~~Credit~~***~~, and~~ ***~~Balance~~*** ~~read-only property.~~

~~Create an Account class that will be a parent to~~ **~~SavingsAccount~~** ~~and~~ **~~CheckingAccount~~** ~~that will contain some additional data members~~ that all accounts have in common, i.e.,

owner, ID, balance

Remember to use ~~protected access~~ specifier on those data members.

~~Each of the bank classes should provide implementations for the interface methods/properties~~.

~~Method~~ **~~Credit~~** ~~should add an amount to the current balance.~~

~~Method~~ **~~Debit~~** ~~should withdraw money from the~~ **~~Account~~** ~~and ensure that the debit amount does not exceed the~~ **~~Account’s~~** ~~balance. If it does, the balance~~~~should be left unchanged, and the method should display the message “Insufficient funds” in the message textbox on the form.~~

~~The classes should also provide a get accessor in property~~ **~~Balance~~** ~~that returns the current balance~~.

~~Override~~ **~~ToString~~** ~~returning account name/type and balance to the caller. Use string interpolation and format decimal values to two digits after the decimal point.~~

~~Derived class~~ **~~SavingsAccount~~** ~~should include a decimal instance variable indicating the annual interest rate (percentage) assigned to the Account.~~

**~~SavingsAccount’s~~** ~~constructor should receive the initial balance, as well as an initial value for the interest rate.~~

**~~SavingsAccount~~** ~~also provides a public method,~~ **~~CalculateInterest,~~** ~~that returns a decimal indicating the amount of interest earned by an account.~~

~~Method~~ **~~CalculateInterest~~** ~~should determine this amount by multiplying the interest rate by the account balance. The~~~~interest rate provided to the constructor is the annual interest rate, but the~~ **~~CalculateInterest~~** ~~method should calculate daily interest (i.e., divide the annual rate by 360).~~

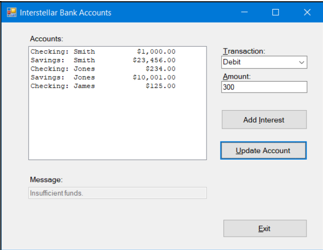
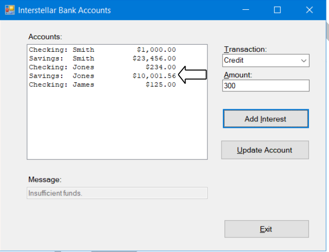
Assignment # 5 DUE: Specified on Canvas

~~Derived class~~ **~~CheckingAccount~~** ~~should include a decimal instance variable that represents the fee charged per transaction.~~

**~~CheckingAccount’s~~** ~~constructor should receive the initial balance, as well as a parameter indicating a fee amount.~~

~~Class~~ **~~CheckingAccount~~** ~~should implement methods~~ **~~Credit~~** ~~and~~ **~~Debit~~** ~~to subtract the fee from the account balance whenever either transaction is performed successfully.~~

**~~CheckingAccount’s~~****~~Debit~~** ~~method should charge a fee only if money is withdrawn (i.e., the debit amount does not exceed the~~~~account balance). A similar message should be displayed if the debit amount exceeds the funds available.~~

**~~AddInterest~~** ~~button to add interest to the selected~~ **~~SavingsAccount~~** ~~object by first invoking its~~ **~~CalculateInterest~~** ~~method, then passing the returned interest~~~~amount to the object’s~~ **~~Credit~~** ~~method~~.

~~Update the Listbox to show added interest. What should happen if the selected Account is not a savings account?~~

~~After adding interest to Jones’ savings account.~~

~~Create a class~~ **~~InterstellarBank~~** ~~that will provide both~~ **~~SavingsAccounts~~** ~~and~~ **~~ChcckingAccounts~~** ~~to its customers. This class should contain a private data member to store all the accounts.~~

~ ~~let’s use an ArrayList that will hold objects of~~ **~~SavingsAccount~~** ~~and~~ **~~CheckingAccount~~**. ~~Define an integer iterator for the~~ **~~InterstellarBank~~** ~~class to access the internal ArrayList with [ ] returning account object.~~

~~Create a form that will display account objects in a list box showing account type, owner’s name, and current balance~~. ~~Create a bank object in the form’s constructor. Load the following accounts into the form:~~

~~m\_bank = new InterstellarBank();~~

~~CS3160, Instructor: Carlson~~

Assignment # 5 DUE: Specified on Canvas

~~m\_bank.AddChecking("Smith", 1000, 1.0M);~~

~~m\_bank.AddSavings("Smith", 23456, 0.03M);~~

~~m\_bank.AddChecking("Jones", 234, 1.0M);~~

~~m\_bank.AddSavings("Jones", 10001, 0.02M);~~

~~m\_bank.AddChecking("James", 125, 1.0M);~~

~~// add accounts for two people, each with checking and savings, i.e., 4 more in all~~

~~For each account object in the array, allow the user to specify an amount of money to withdraw from the account using method~~ **~~Debit~~** ~~and deposit an amount into the account using method~~ **~~Credit~~**~~.~~

~~After processing, display the updated account balance obtained by using property~~ **~~Balance~~** ~~on each object.~~

**Code Documentation**

Each source file should contain a documentation header with a description, author name, and class information (CS3160, Spring/Fall 20xx). Each function should have a documentation header with a minimally function name, description, parameters, return value. Use proper program style at all times, which means indentation, alignment, and whitespace. Utilize self-documenting code which means use mnemonic, i.e., meaningful variable/function/namespace/class names, etc.

**What to turn in?**

Submit a compressed (.zip) solution folder via Canvas.

Make sure that all necessary files to compile/link/execute your projects are provided in your solution folder.

Here are some files that may be required:

1. All required C# source files (.cs)
2. makefile (if the solution requires it)
3. The entire Visual Studio 2019 solution folder (if the solution requires it)

CS3160, Instructor: Carlson