CEGEP VANIER COLLEGE
Department of Computer Science

Assignment-02: Object Oriented Programming - Inheritance

Assignment-02 weight is 5% of final grade
Max Marks (55)

Course Title : Application Development-1 (Sec-01)		**Teacher** : Syed Naseem Afzal	
Course Number : 420-331-VA		**Due Date** : February 12, 2026	

What to hand in?

After successful demonstration of your completed work, create a compressed file [**Filename**: LastNameFirstNameAssignment02.ZIP], with the following:

a) MS-Word document [**Filename**: LastNameFirstNameAssignment02.docx] containing screen shots of your examples' output window with input values and produced output/result.
b) Your working application project folder(s).

Upload your **LastNameFirstNameAssignment02.ZIP** file in LEA's assignment section.

Note: Your uploaded (ZIP) file should contain all files of the working examples' project folders. Incomplete and / or non-working app's files will not be graded and have zero value.

Problem: Account Inheritance Hierarchy — Max Marks (55)

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 base class **Account** and derived classes **SavingsAccount** and **CheckingAccount** that inherit from class **Account**.

1) Base class **Account** should include: **Max Marks (15)**

   i) one **private `string`** instance variable to represent the account number (e.g **accNumber**).
   ii) one **private `decimal`** instance variable to represent the account balance (e.g **accBalance**).
   iii) one **private `string`** variable to represent the person's last name (e.g. **lastName**)
   iv) one **private `string`** variable to represent the person's first name (e.g. **firstName**)

2) The **Account** class should provide a constructor with four parameters:

   i) first parameter to receive account number and uses it to initialize the private instance variable using **public property AccNumber**.
   ii) second parameter that receives an initial balance and uses it to initialize the private instance variable using a **public property Balance**. The **property Balance** should validate the initial balance to ensure that it is greater than or equal to 0.0; if not, ignore the received initial balance and assign a zero to the private instance variable & display the message "**Account initial balance *amount should be a positive value***."

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

iii) third parameter to receive person's last name and uses it to initialize the private instance variable using **public property LastName**.

iv) fourth parameter to receive person's first name and uses it to initialize the private instance variable using **public property FirstName**.

3) The class **Account** should provide two public methods.

   i) Method **Credit**, to add (credit) an non-negative (positive) amount to the current balance.

   ii) Method **Debit**, to withdraw (subtract) 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 "***Debit amount exceeded account balance***."
   [**Note:** Define **Account**'s **Debit** method of **bool** type, so that the method should returns a **bool** value indicating whether money was withdrawn successfully or not.]

4) The class **Account** should have a public method **DisplayAccount**, that display values of four instance variables in tabular form as shown below, using C# "**String Interpolation**" to insert values in string literals to create formatted strings, and format specifier to format numeric values and alignment.

ACCOUNT	
Account Number	7431592
Balance Amount	$284.97
Last Name	Dumitrescu Tivig
First Name	Craig Justin Vinoya

5) Derived class **SavingsAccount** (from the **Account** class) **Max Marks (15)**
SavingsAccount class should inherit the functionality of class **Account**, but also include a **private decimal** instance variable indicating the **interest rate** (percentage) assigned to the Account.

6) **SavingsAccount**'s constructor should receive five parameter values, account number, initial balance, last name, first name, and a value for the **interest rate**.

   i) First four parameters' values will be passed to constructor of base class (**Account**) to initialize the inherited four instance variables from the base class (**Account**), by the derived class **SavingsAccount** constructor.

   ii) Fifth parameter value (interest rate) will be used to initialize the private instance variable using a **public property InterestRate**. The **InterestRate** property should validate the interest rate value received via constructor's parameter, to ensure that it is greater than zero (non-negative); if not, ignore the received interest rate value, and assign 0.05 (5%) as interest rate to the instance variable (for **interest rate**) and display the message "**Interest rate should be a positive value**."

7) **SavingsAccount** should provide public method **CalculateInterest** that returns a **decimal** indicating the amount of interest earned by an account. Method **CalculateInterest** should calcualte this amount by multiplying the **interest rate** with the **account balance**.
[**Note: SavingsAccount** should inherit methods **Credit** and **Debit** without redefining them.]

8) The class **SavingsAccount** should override the inherited public method **DisplayAccount**, so that it display values of five instance variables in tabular form as shown below, using C# "**String Interpolation**"

to insert values in string literals to create formatted strings, and format specifier to format numeric values and alignment.

SAVING ACCOUNT	
Account Number	8753124
Balance Amount	$537.68
Last Name	Dumitrescu Tivig
First Name	Craig Justin Vinoya
Interest Rate	$4.13
Interest Amount	$22.21

9) Derived class **CheckingAccount** (from the **Account** class) **Max Marks (15)**
Derived class **CheckingAccount** should inherit functionality of class **Account** and include a **private `decimal`** instance variable that represents the transaction fee charged per transaction.

10) **CheckingAccount**'s constructor should receive five parameter values, account number, initial balance, last name, first name, and transaction fee.
   i) First four parameters' values will be passed to constructor of base class (**Account**) to initialize the inherited four instance variables from the base class (**Account**), by the derived class **CheckingAccount** constructor.
   ii) Fifth parameter value (transaction fee) will be used to initialize the private instance variable using a **public property `TransactionFee`**. The **`TransactionFee`** property should validate the transaction fee value received via constructor's parameter, to ensure that it is greater than zero (non-negative); if not, ignore the received interest rate value, and assign 1.5 ($1.50) as transaction fee to the instance variable (for transaction fee) and display the message "**Transaction fee should be a positive value**."

11) Class **CheckingAccount** should override methods **Credit** and **Debit** so that the **CheckingAccount**'s versions of these methods should subtract the transaction fee from the account balance whenever either transaction is performed successfully. **CheckingAccount**'s versions of these methods should invoke the base-class **Account** version to perform the updates to an account balance.

12) **CheckingAccount**'s **Debit** method should charge a transaction fee only if money is actually withdrawn (i.e., the debit amount does not exceed the account balance). In **CheckingAccount**'s **Debit** method use the **`bool`** value return from **Account**'s **Debit** method to determine whether a transaction fee should be charged. [**Note:** Define **Account**'s **Debit** method of **`bool`** type, so that the method should returns a **`bool`** value indicating whether money was withdrawn successfully or not.]

13) The class **CheckingAccount** should override the inherited public method **`DisplayAccount`**, so that it display values of five instance variables in tabular form as shown below, using C# "**String Interpolation**" to insert values in string literals to create formatted strings, and format specifier to format numeric values and alignment.

CHECKING ACCOUNT	
Account Number	9125467
Balance Amount	$823.87

Last Name	Dumitrescu Tivig
First Name	Craig Justin Vinoya
Transaction Fee	$1.50

14) After defining the above mentioned three classes in the specified hierarchy and saving them in three separate files, write a driver application class that creates objects of each class and tests their methods. Save the driver application class in separate fourth file. **Max Marks (10)**

15) The driver application class should accept following six input values via keyboard from the application user:
    i) Account Number
    ii) Account Balance
    iii) Last Name
    iv) First Name
    v) Interest Rate
    vi) Transaction Fee

16) Driver application should use the **SavingsAccount** class **CalculateInterest** method to compute the amount of interest for the current account balance of the **SavingsAccount** object, and then call the **Credit** method with computed interest amount to add the computed interest amount to the current balance amount of the **SavingsAccount** object.
    Driver application should display the account current balance before and after:
    i) addition of interest amount to the **SavingsAccount** object balance.
    ii) performing credit and / or debit transaction to the **SavingsAccount** object.

17) Driver application should perform credit and debit transactions to **CheckingAccount** object by calling **Credit** and **Debit** methods of the **CheckingAccount** class. Driver application should display the **CheckingAccount** object's current balance before and after each credit and / or debit transaction.

[**Note**: Please use C# "**String Interpolation**" to insert values in string literals to create formatted strings, and format specifier to format numeric values and alignment of your app output].