Assignment: WeBankApp - Banking Management System (With Inheritance)

Objective:

Develop a simple banking management system for WeBankApp using C# concepts like Aggregation, Association, and Inheritance.

Solution Structure:

- **Solution Name:** WeBankApp
- **Class Library:** WeBankBusinessLayer
- **Console Application:** WeBankConsoleApp

Class Diagram Enhancements:

Assignment Steps:

Step 1: Create the Visual Studio Solution

1. Open Visual Studio and create a new solution named **WeBankApp**.
2. Add a **Class Library** project named **WeBankBusinessLayer**.
3. Add a **Console Application** project named **WeBankConsoleApp**.
4. Add a reference to **WeBankBusinessLayer** in **WeBankConsoleApp**.

Step 2: Create Classes in WeBankBusinessLayer

1. Create Customer

- **Properties:**
  - CustomerId (int)
  - Name (string)
  - Age (int)
  - Gender (string)
  - PhoneNumber (string)
- **Constructor:** Initialize CustomerId, Name, Age, Gender, PhoneNumber.
- **Method:** string DisplayCustomerInfo() → Virtual method which returns customer details as a concatenated string (Refer output screenshot).

2. Create SavingsAccountHolder

- **Inherits the Customer class**
- **Additional Properties:**
  - InterestRate (double)
  - Balance (double)
- **Constructor:** Initialize InterestRate and Balance along with base properties.
- **Method:** string DisplayCustomerInfo()
  - Overridden method which returns the customer account details.
  - Invoke the base class method DisplayCustomerInfo().
  - Concatenate the result obtained with the InterestRate and Balance values and return the details (Refer output screenshot).

3. Create CurrentAccountHolder

- **Inherits the Customer class**
- **Additional Properties:**
  - OverdraftLimit (double)

- **Constructor:** Initialize OverdraftLimit along with base properties.
- **Method:** string GetAccountDetails()
    - Overridden method which returns the customer account details.
    - Invoke the base class method DisplayCustomerInfo().
    - Concatenate the result obtained with the OverdraftLimit value and return the details (Refer output screenshot).

4. Create Banker Class

- **Properties:**
    - BankerId (int)
    - Name (string)
    - Branch (string)
    - PhoneNumber (string)
- **Constructor:** Initialize BankerId, Name, Branch, PhoneNumber.
- **Method:**
    - bool UpdateBankerDetails(string newBranch)
        - Check if the newBranch is not the same as Branch.
            - If yes, update the Branch and return true.
        - Else, return false
    - bool UpdateBankerDetails(string newBranch, string newPhoneNumber)
        - Declare a bool variable status.
        - Check if the newBranch is not the same as Branch.
            - If yes, update the Branch and set status as true.
        - Check if the newPhoneNumber is not the same as PhoneNumber.
            - If yes, update the PhoneNumber and set status as true.
        - If any of the fields is not updated, set status as false.
        - Return the status.
    - string DisplayBankerInfo() → Returns banker details as a concatenated string (Refer output screenshot).

5. Create Bank Class

- **Properties:**
    - BankId (string) - auto-generated
    - Name (string)
    - Location (string)
    - Banker (Banker)
- **Field:** counter - private static variable to auto-generate BankId like B1001, B1002, B1003.
- **Static Constructor:** Initialize counter variable appropriately.
- **Constructor:** Initialize Name, Location, and Banker. Initialize BankId using the counter.
- **Method:** string DisplayBankInfo() → Returns bank details as a concatenated string (Refer output screenshot).

6. Create Transaction Class

- **Properties:**
    - TransactionId (string)
    - TransactionDate (DateTime)
    - Amount (double)
    - Type (string)
    - Status (string)
- **Field:** counter - private static variable to auto-generate TransactionID like C501, D502, C503… etc.
- **Static Constructor:** Initialize counter variable appropriately.
- **Parameterless constructor** - No logic needed
- **Methods:**
    - string DisplayTransactionDetails() → Returns transaction details as a concatenated string (Refer output screenshot).
    - bool ProcessTransaction(SavingsAccountHolder customer, Banker banker, double amount, string type)
        - If customer and banker are valid:
            - Check the transaction type.
            - If it is "Debit"

            - Check if the customer’s Balance is greater than or equal to the amount.
            - If yes,
                - Auto-generate TransactionID with the prefix "D".
                - Assign amount and type to Amount and Type respectively.
                - Set TransactionDate to the current **date and time**.
                - Set Status = "Completed".
        - If it is “Credit”
            - Auto-generate TransactionID with the prefix "C".
            - Assign amount and type to Amount and Type respectively.
            - Set TransactionDate to the current **date and time**.
            - Set Status = "Completed".
        - Return “Transaction completed for customer <customer name> with banker <banker name> with Transaction ID : <TransactionId>”.
    - Else, return “Transaction could not be completed”.
    - Implement exception handling in this method and handle any exception that may occur. In case of an exception, return “Some error occurred, transaction failed!”.

Step 3: Implement Business Logic in WeBankConsoleApp

1. **Instantiate Objects:**
    - Create a **SavingsAccountHolder**.
    - Create a **CurrentAccountHolder**.
    - Create a **Banker**.

    - Create a **Bank** and associate it with the **Banker**.
2. **Call Methods:**
    - Display **SavingsAccountHolder** and **CurrentAccountHolder** details.
    - Display **Banker** details.
    - Update banker’s Branch.
    - Display **Banker’s** updated details.
    - Display **Bank** details.
    - Process a Debit and a Credit transaction for the SavingsAccountHolder.
    - Display the status message as received from the transaction process.

Step 4: Sample Program.cs Code

```
static void Main(string[] args)

{

    SavingsAccountHolder savingsAccountHolder = new
SavingsAccountHolder(2.5,

        50000, 101, "Anna Miller", 34, "Female", "9999999999");



    CurrentAccountHolder currentAccountHolder = new
CurrentAccountHolder(20000,

        102, "Frank Lawson", 29, "Male", "8888888888");



    Banker banker = new Banker(1, "Katie Otto", "Church Street",
"7777777777");



    Bank bank = new Bank("We Trust Bank", "New York", banker);
    Console.WriteLine("Savings Account Holder details : " +
savingsAccountHolder.DisplayCustomerInfo());



Console.WriteLine("--------------------------------------------------------"
);


    Console.WriteLine("Current Account Holder details : " +
currentAccountHolder.DisplayCustomerInfo());



Console.WriteLine("--------------------------------------------------------"
);


    Console.WriteLine("Banker Details : " +
banker.DisplayBankerInfo());


    Console.WriteLine("Updating Banker Contact Info");


    banker.UpdateBankerDetails("Mall Road");


    Console.WriteLine("Banker Details : " +
banker.DisplayBankerInfo());
```

```
Console.WriteLine("----------------------------------------------------------
");

    Console.WriteLine("Bank Details: " + bank.DisplayBankInfo());

Console.WriteLine("----------------------------------------------------------
");

    Console.WriteLine("----------------Processing
Transactions------------------");

    Transaction transactionOne = new Transaction();
    Transaction transactionTwo = new Transaction();

    string messageOne =
transactionOne.ProcessTransaction(savingsAccountHolder, banker,
1000, "Debit");
    Console.WriteLine("Transaction One");
    Console.WriteLine(messageOne);

Console.WriteLine("----------------------------------------------------------
");
```

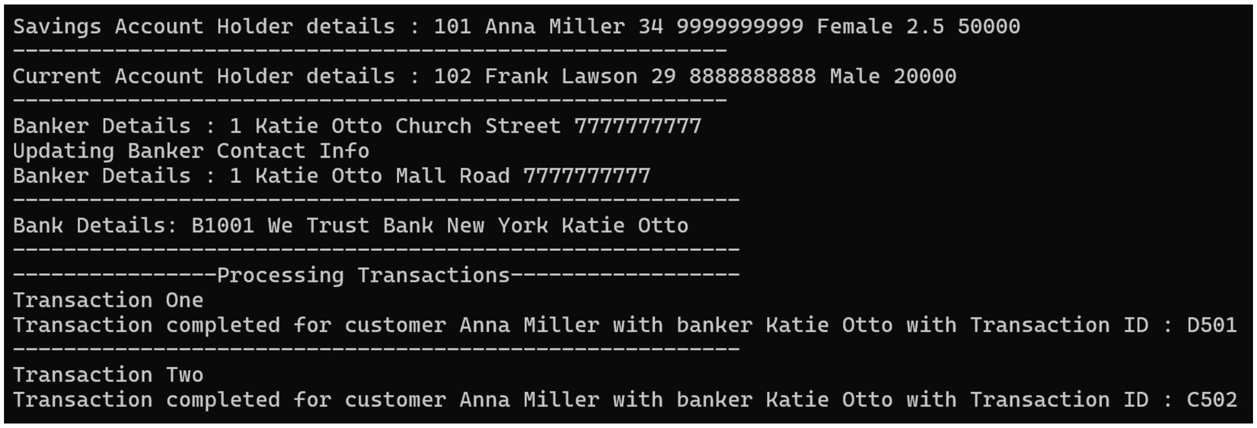
```
    Console.WriteLine("Transaction Two");

    string messageTwo =
transactionTwo.ProcessTransaction(savingsAccountHolder, banker,
3000, "Credit");

    Console.WriteLine(messageTwo);

}
```

Step 5: Sample Console Output (Expected Behavior)



```
Savings Account Holder details : 101 Anna Miller 34 9999999999 Female 2.5 50000
---------------------------------------------------------
Current Account Holder details : 102 Frank Lawson 29 8888888888 Male 20000
---------------------------------------------------------
Banker Details : 1 Katie Otto Church Street 7777777777
Updating Banker Contact Info
Banker Details : 1 Katie Otto Mall Road 7777777777
----------------------------------------------------------
Bank Details: B1001 We Trust Bank New York Katie Otto
----------------------------------------------------------
----------------Processing Transactions------------------
Transaction One
Transaction completed for customer Anna Miller with banker Katie Otto with Transaction ID : D501
----------------------------------------------------------
Transaction Two
Transaction completed for customer Anna Miller with banker Katie Otto with Transaction ID : C502
```