Lab 2: Pair Programming

Complete this exercise using one of the pair programming methods we discussed. You only need to come up with one solution between you - but it must be a joint effort.

Think about how you are going to share code as you work on it together, especially when you are remote from each other. Can you use **git** as the mechanism to share a common code base?

There is a C# and Java version of the starter project - which are basically empty (you will need to add the necessary classes to the project). If you are using Python or JavaScript, you will need to start fresh.

Task 1

Write a class to represent **BankAccount** objects. The class should include a suitable constructor, and should include fields/getters/setters/properties for:

1. Customer name (string).
2. Account number (integer).
3. Balance (number with two decimal points).

The account number should be auto-generated if it is not supplied to the constructor, and should start at 100000 then increment by +1 for every new account created.

The **BankAccount** class should also include methods for:

4. Deposit (amount) – changes the balance appropriately.
5. Withdraw (amount) – changes the balance appropriately.

Your class should compile. There is no need to write any client code.

Task 2

Write two more child classes to represent specialised versions of the **BankAccount** class.

1. **CurrentAccount** is a kind of **BankAccount**
    - But with an overdraft limit.
    - When you withdraw money you are not permitted to go beyond your overdraft limit (which might be zero).
2. **SavingsAccount** is a kind of **BankAccount**
    - But with an interest rate.
    - It should have an additional method, **AddInterest ()**, which calculates and adds interest to the account, based on the current value of the account's balance of the current value of the account's interest rate (NB: this isn't how real bank accounts work!)

Your class should compile. There is no need to write any client code.