Mini Project - Build a Simple Banking System

Description:

This task involves creating a basic Banking System using C++. The system will allow users to create accounts, make deposits and withdrawals, and view account details. Through this project, students will practice C++ fundamentals, including classes, inheritance, pointers, and polymorphism.

Steps:

1. Create an **Account** Class (Base Class)
    a. Define a simple **Account** class with:
        i. **accountHolderName**: Name of the account holder.
        ii. **accountNumber**: Unique account number.
        iii. **balance**: Account balance.
    b. Add methods to:
        i. **Deposit money** deposit().
        ii. **Withdraw money** withdraw().
        iii. **Display account info** displayAccountInfo().
            1. Goal: Learn to set up a basic class with attributes and functions.
2. Create Different Account Types with Inheritance
    a. Create two classes, **SavingsAccount** and **CheckingAccount**, that inherit from Account.
        i. SavingsAccount:
            1. It has an additional attribute: **interestRate**.
            2. Adds a function **applyInterest()** to increase the balance based on the interest rate.
        ii. **Checking Account**:
            1. It has an attribute **overdraftLimit** to allow limited overdrafts.
            2. Overrides **withdraw()** to allow overdrafts within the limit.
                a. Goal: Practice using inheritance to add specialized features to each account type.
3. Use Polymorphism to Manage Accounts
    a. Use pointers to the Account class to manage both SavingsAccount and CheckingAccount objects in one place.
        i. Use a **displayAccountInfo()** function in each derived class so each type of account can display its own details.

1. Goal: Understand polymorphism by treating different accounts in a unified way.

4. Create a **Bank** Class to Handle Accounts
    a. This class should:
        i. Add new accounts.
        ii. Display all account details.
        iii. Manage a list of **Account** pointers.
            1. Goal: Practice organizing and managing multiple objects using a container.
5. Build a **Simple Menu** for User Interaction
    a. Add options for users to:
        i. Create a **SavingsAccount** or **CheckingAccount**.
        ii. Deposit or withdraw money.
        iii. Apply interest to savings accounts.
        iv. View all account details.
            1. Goal: Implement a user-friendly console menu.

Important points before you start:

1. **Ensure Proper Memory Management**
   Use pointers for accounts and ensure that memory is properly freed when done.
2. **Create Class Diagram**
3. **Use .cpp files and header files for more clarity**
4. **Advance Task**
    a. Create Test Cases to support your implementation
        i. Boost Library
        ii. Catch2

Expected Output:

The user should be able to perform actions like:

Using the Bank Class

1. Creating an Account

   Enter account type (1 for Savings, 2 for Checking): 1
   Enter name: Alice
   Enter initial balance: 500
   Enter interest rate: 0.03

Savings account created!

2. Viewing Account Details

Savings Account - Name: Alice, Balance: $500, Interest Rate: 3%