Tennessee Tech University
College of Engineering: Computer Science

CSC1310- Data Structures and Algorithms

Exercise 1: Understanding Encapsulation

Objective:

The goal of this Exercise is to understand the concept of encapsulation in object-oriented programming (OOP) using C++. You will create a simple class that demonstrates how to encapsulate data members and provide controlled access through member functions.

Exercise Description:
Create a C++ program that models a simple banking system. The system should include a class `BankAccount` that encapsulates the account holder's name, account number, and balance. The class should allow users to deposit money, withdraw money, and check the account balance.

Requirements:

1. **Class Definition:**
   - Create a class named `BankAccount`. The class should have the following private data members:
     1. `string accountHolderName;` // Name of the account holder
     2. `int accountNumber;` // Unique account number
     3. `double balance;` // Account balance

2. **Member Functions:**
   - Constructor: Create a constructor that initializes the account holder's name, account number, and initial balance.
   - Deposit Function: Create a public member function `void deposit(double amount);` that adds the specified amount to the account balance.
   - Withdraw Function: Create a public member function `void withdraw(double amount);` that subtracts the specified amount from the account balance, ensuring that the balance does not become negative.
   - Get Balance Function: Create a public member function `double getBalance() const;` that returns the current balance.
   - Get Account Info Function: Create a public member function `void displayAccountInfo() const;` that displays the account holder's name, account number, and current balance.

3. **Main Function:**
   - In the `main()` function, create an object of `BankAccount` and demonstrate the use of the above functions by:
   - Depositing money into the account.

- Withdrawing money from the account.
- Displaying the account information after each operation.

You may start with the following code:

```
int main() {
    // Creating a bank account
    BankAccount myAccount("John Doe", 123456, 500.00);

    // Display initial account information
    myAccount.displayAccountInfo();

    // Deposit money
    myAccount.deposit(150.00);
    cout << "After depositing $150:" << endl;
    myAccount.displayAccountInfo();

    // Withdraw money
    myAccount.withdraw(100.00);
    cout << "After withdrawing $100:" << endl;
    myAccount.displayAccountInfo();

    return 0;
}
```

4. Submission Instructions:

- Submit your C++ source code file (`.cpp`) along with a screenshot of the program output.
- Ensure your code is well-documented with comments explaining each part of the code.

This Exercise will help you practice the fundamental concept of encapsulation in C++, a key principle in object-oriented programming that contributes to modular, maintainable, and secure code.

Output Sample

```
Account Holder: John Doe
Account Number: 123456
Current Balance: $500
--------------------------------
Deposited: $150
After depositing $150:
Account Holder: John Doe
```

CSC1310- Data Structures and Algorithms

```
Account Number: 123456
Current Balance: $650
--------------------------------
Withdrawn: $100
After withdrawing $100:
Account Holder: John Doe
Account Number: 123456
Current Balance: $550
--------------------------------
Insufficient balance!
After attempting to withdraw $600:
Account Holder: John Doe
Account Number: 123456
Current Balance: $550
--------------------------------
```