Object Oriented Programming

Homework 08 **Marks 10**

Instructions

Work on this homework individually. Absolutely NO collaboration is allowed. Any traces of plagiarism would result in ZERO marks in this homework and possible disciplinary action. Task should be coded in **C++**. You are strictly ***NOT ALLOWED*** to include any additional data-members/functions/constructors in your class. Write the ***main*** function first and keep testing the functionality of each function once created.

Due Date

Upload the solution *(source code **.cpp** file only)* labeled with your complete roll number in capital letters e.g., BITF21M000 till 05:00PM Thursday, May 11, 2023, in course's *Google* classroom.

Bank Account Hierarchy

1. ADT: Account

Design an **Account** class that has the following members:

Protected Member Variable:

- **firstName**, a **string** used to hold the **first name** of the **account holder**.
- **lastName**, a **string** used to hold the **last name** of the **account holder**.
- **currentBalance**, a **double** used to hold the **current balance** of the **account holder**.

Public Member Functions:

- `Account(string firstName, string lastName, double currentBalance);`
- `virtual string getAcctType() const;`
  Returns "*Account*" (and that's all it does).
- `virtual double debitTransaction(double debitamount);`
  Subtract the transaction amount from the account. No condition check is required.
- `virtual double creditTransaction(double creditamount);`
  Add the transaction amount to the account. No condition check is required.
- `void print();`
  Print the three fields on the console.

```
Saud Yaseen          Account        200.00
Hunain Shahid        Account        1000.00
```

2. ADT: CheckingAccount

Minimum balance on the account is **equal to 100**. For every transaction resulting in the amount of the account **being lower than the minimum balance**, a **transaction fee** of **10** is charged. The balance can become **negative**.

Design a **CheckingAccount** class that is derived from the **Account** class. The **CheckingAccount** class should have the following members:

Public Member Functions:

- `CheckingAccount(string firstName, string lastName, double currentBalance);`
  Do not forget to initialize fields other than the three parameters.
- `virtual string getAcctType() const;`
  Returns "*CheckingAccount*" (and that's all it does).
- `virtual double debitTransaction(double debitamount);`
  Subtracts the transaction amount from the account and possibly charges a transaction fee. You must call the base class **debitTransaction()** method inside this method.
- `virtual double creditTransaction(double creditamount);`
  Adds the transaction amount to the account and possibly charges a transaction fee. You must call the base class **creditTransaction()** method inside this method.
- `void chargeFee();`
  This is a private method. It is called internally by the debit and credit transaction methods.

3. ADT: SavingsAccount

A **maximum of 2 transactions** is allowed. After successfully committing any transaction an **interest of 2% is paid** (i.e., added to the current balance). The interest is paid on the **entire amount of the account**.

Design a **SavingsAccount** class that is derived from the **Account** class. The **SavingsAccount** class should have the following members:

Public Member Functions:

- `SavingsAccount(string firstName, string lastName, double currentBalance);`
  Do not forget to initialize fields other than the three parameters.
- `virtual string getAcctType() const;`
  Returns "*SavingsAccount*" (and that's all it does).
- `virtual double debitTransaction(double debitamount);`
  If there are less than two transactions, subtracts the transaction amount from the account and pays interest. You must call the base class **debitTransaction()** method inside this method.
- `virtual double creditTransaction(double creditamount);`
  If there are less than two transactions, adds the transaction amount to the account and pays interest. You must call the base class **creditTransaction()** method inside this method.
- `void payInterest();`
  This is a private method. It is called internally by the debit and credit transaction methods.

4. Main Function

It creates one **Checking account** and one **Savings account**, and does some transactions. After every transaction, the account information is displayed, by calling **print()** in the object.

Requirements:

1. Both objects must be dynamically created and pointed by the base class pointer. For example,

```
Account* ac2 = new SavingsAccount("Saud", "Yaseen", 200.0);
Account* ac1 = new CheckingAccount("Hunain", "Shahid", 1000.0);
```

2. The output should be

```
*** Two accounts ***
Saud Yaseen         CheckingAccount     200.00
Hunain Shahid       SavingsAccount     1000.00

=== Test for CheckingAccount ===

(1) Debitting 150.00 -- The balance is 50.00
(2) Debitting 5.00 -- The balance is 35.00
(3) Crediting 200.00 -- The balance is 235.00
(*) Final account information

Saud Yaseen      CheckingAccount     235.00

=== Test for SavingsAccount ===

(1) Debitting 150.00 -- The balance is 867.00
(2) Crediting 200.00 -- The balance is 1088.34
(3) Debitting 5.00 -- The balance is 1088.34
(*) Final account information

Hunain Shahid    SavingsAccount      1088.34
```

Failure to abide by the submission instructions will cause a penalty of two marks.
No submission will be accepted after the due date and time.

BEST OF LUCK