Objects and Classes

Assignment 1: BankAccount class

Given main(), define the BankAccount class (in files BankAccount.h and BankAccount.cpp) that manages checking and savings accounts. The class has three private data members:

- customer name (string)
- savings account balance (double)
- checking account balance (double)

Implement the following constructor and public member functions as listed below:

- BankAccount(string newName, double amt1, double amt2) - set the customer name to parameter newName, set the checking account balance to parameter amt1 and set the savings account balance to parameter amt2. (amt stands for amount)
- void SetName(string newName) - set the customer name to parameter newName
- string GetName() - return the customer name
- void SetChecking(double amt) - set the checking account balance to parameter amt
- double GetChecking() - return the checking account balance
- void SetSavings(double amt) - set the savings account balance to parameter amt
- double GetSavings() - return the savings account balance
- void DepositChecking(double amt) - add parameter amt to the checking account balance (only if positive)
- void DepositSavings(double amt) - add parameter amt to the savings account balance (only if positive)
- void WithdrawChecking(double amt) - subtract parameter amt from the checking account balance (only if positive)
- void WithdrawSavings(double amt) - subtract parameter amt from the savings account balance (only if positive)
- void TransferToSavings(double amt) - subtract parameter amt from the checking account balance and add to the savings account balance (only if positive)

File is marked as read only — Current file: main.cpp

```cpp
#include <iostream>
#include <iomanip>
#include "BankAccount.h"
using namespace std;

int main() {
    BankAccount account("Mickey", 500.00, 1000.00);
    account.SetChecking(500);
    account.SetSavings(500);
    account.WithdrawSavings(100);
    account.WithdrawChecking(100);
    account.TransferToSavings(300);

    cout << account.GetName() << endl;
    cout << fixed << setprecision(2);
    cout << account.GetChecking() << endl;
    cout << account.GetSavings() << endl;

    return 0;
}
```

Current file: BankAccount.cpp

```cpp
#include <iostream>
#include "BankAccount.h"
using namespace std;

// TODO: Define public member functions
```

Current file: BankAccount.h

```cpp
#ifndef BANKACCOUNTH
#define BANKACCOUNTH

#include <string>
using namespace std;

class BankAccount {
  public:
    // TODO: Declare public member functions

  private:
    // TODO: Declare private data members
};

#endif
```

Assignment 1 Tests:

Apply the following 5 tests.

1. Unit test (2 points)

Tests that BankAccount("Jane",100.00, 500.00) correctly initializes bank account. Tests bankAccount.GetName().equals("Jane"), bankAccount.GetChecking() == 100.0, and bankAccount.GetSavings() == 500.0.

Show details

2. Unit test (2 points)

Tests that BankAccount("Jane', 500.00, 1000.00) correctly initializes bank account. Tests bankAccount.DepositSavings(123.00), bankAccount.GetSavings() == 1123.00, bankAccount.WithdrawSavings(25.00), and bankAccount.GetSavings() == 1098.00.

Show details

3. Unit test (2 points)

Tests that BankAccount("Jane', 750.00, 450.00) correctly initializes bank account. Tests bankAccount.DepositChecking(75.00), bankAccount.GetChecking() == 825.00, bankAccount.WithdrawChecking(45.00), and bankAccount.GetChecking() == 780.00.

Show details

4. Unit test (2 points)

Tests that BankAccount("Maria", 100.00, 100.00) correctly initializes bank account. Tests bankAccount.DepositChecking(-5.00)/bankAccount.WithdrawChecking(-5.00), bankAccount.GetChecking() == 100.00, bankAccount.DepositSavings(-5.00)/bankAccount.WithdrawSavings(-5.00), and bankAccount.GetSavings() == 100.00

Show details

5. Unit test (2 points)

Tests that BankAccount("James", 1000.00, 1000.00) correctly initializes bank account. Tests bankAccount.TransferToSavings(100.00), and bankAccount.GetSavings() == 1100.00, and bankAccount.GetChecking == 900.00

Show details

Assignment 2: Calculator Class

Given main(), complete the Calculator class (in files Calculator.h and Calculator.cpp) that emulates basic functions of a calculator: add, subtract, multiple, divide, and clear. The class has one private data member called value for the calculator's current value. Implement the following constructor and public member functions as listed below:

- Calculator() - default constructor to set the data member to 0.0
- void Add(double val) - add the parameter to the data member
- void Subtract(double val) - subtract the parameter from the data member
- void Multiply(double val) - multiply the data member by the parameter
- void Divide(double val) - divide the data member by the parameter
- void Clear( ) - set the data member to 0.0
- double GetValue( ) - return the data member

Given two double input values num1 and num2, the program outputs the following values:

1. The initial value of the data member, value
2. The value after adding num1
3. The value after multiplying by 3
4. The value after subtracting num2
5. The value after dividing by 2
6. The value after calling the clear() method

Example: If the input is: 10.0 5.0 => the output is: 0.0

10.0

30.0

25.0

12.5

0.0

File is marked as read only — Current file: main.cpp

```cpp
#include <iostream>
#include <iomanip>
#include "Calculator.h"
using namespace std;

int main() {
  Calculator calc;
  double num1;
  double num2;

  cin >> num1;
  cin >> num2;

  cout << fixed << setprecision(1);
  // 1. The initial value
  cout << calc.GetValue() << endl;

  // 2. The value after adding num1
  calc.Add(num1);
  cout << calc.GetValue() << endl;

  // 3. The value after multiplying by 3
  calc.Multiply(3);
  cout << calc.GetValue() << endl;

  // 4. The value after subtracting num2
  calc.Subtract(num2);
  cout << calc.GetValue() << endl;

  // 5. The value after dividing by 2
  calc.Divide(2);
  cout << calc.GetValue() << endl;

  // 6. The value after calling the clear() method
  calc.Clear();
  cout << calc.GetValue() << endl;

  return 0;
}
```

Current file: Calculator.h

```cpp
#ifndef CALCULATORH
#define CALCULATORH

class Calculator {
  public:
    // TODO: Declare default constructor

    // TODO: Declare member functions -
    //       Add(), Subtract(), Multiply(), Divide(), Clear(), GetValue()

  private:
    // TODO: Declare private data member - value
};

#endif
```

Current file: Calculator.cpp

```cpp
#include <iostream>
#include "Calculator.h"
using namespace std;

// TODO: Define default constructor

// TODO: Define member functions -
//       Add(), Subtract(), Multiply(), Divide(), Clear(), GetValue()
```

Assignment 2 Tests

Apply the following 4 tests.

1. Compare output (2 points)

When input is

```
10.0 5.0
```

Standard output exactly matches

```
0.0
10.0
30.0
25.0
12.5
0.0
```

3. Compare output (3 points)

When input is

```
0.0 0.0
```

Standard output exactly matches

```
0.0
0.0
0.0
0.0
0.0
0.0
```

2. Compare output (2 points)

When input is

```
13.5 12.7
```

Standard output exactly matches

```
0.0
13.5
40.5
27.8
13.9
0.0
```

4. Compare output (3 points)

When input is

```
-53.5 -100.3
```

Standard output exactly matches

```
0.0
-53.5
-160.5
-60.2
-30.1
0.0
```

Submissions

Note: Do not forget to submit all two assignments and corresponding test outputs to receive full credit.

1 - Name your C++ files FirstName_Lastname_Calculate.cpp, FirstName_Lastname_BankAccount.cpp.

2 - Prepare your report in docx or pdf format and name it Firstname_Lastname.docx or Firstname_Lastname.pdf. Put both your assignments and corresponding tests in ONE report file.

3 - Add the screenshot of your code to the report. All tests should be performed and the result screenshot be included in the report.

Note: Make sure to have your report containing both explanatnations and screenshots.