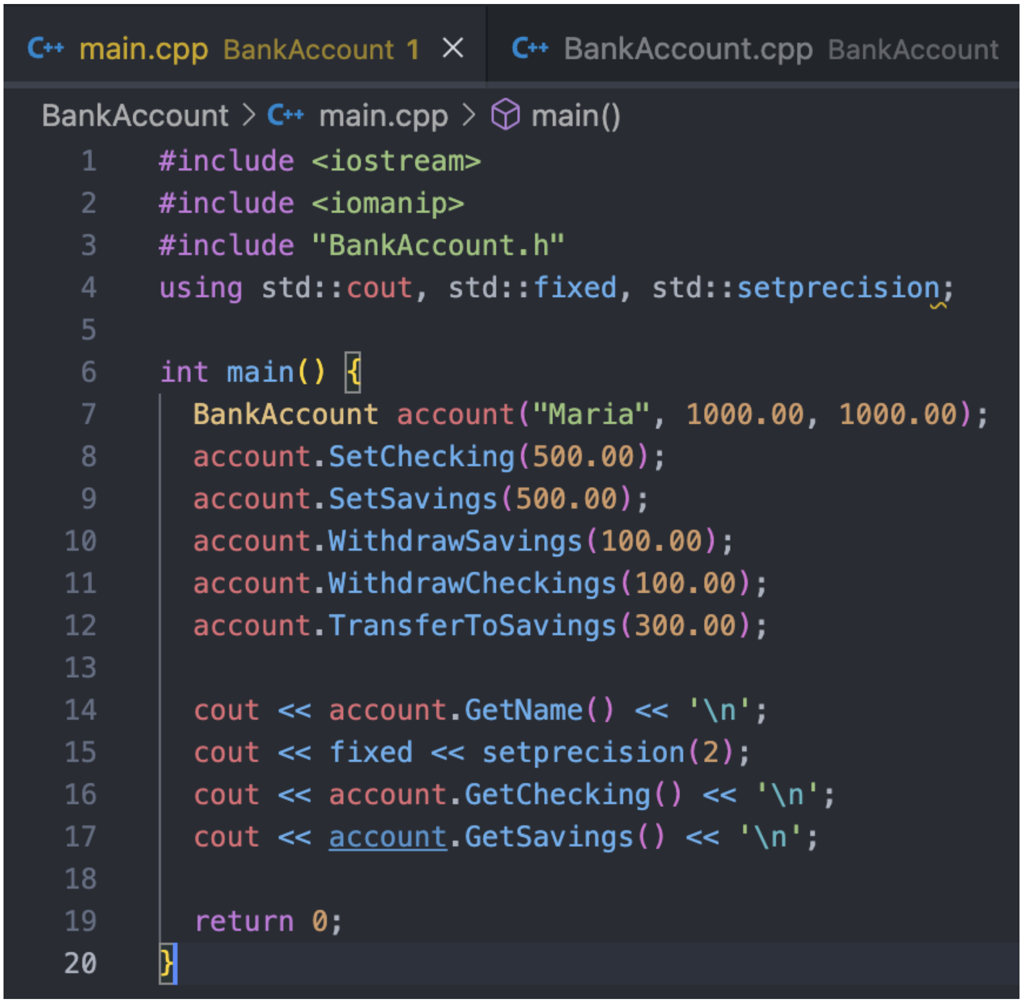
Assignment 1

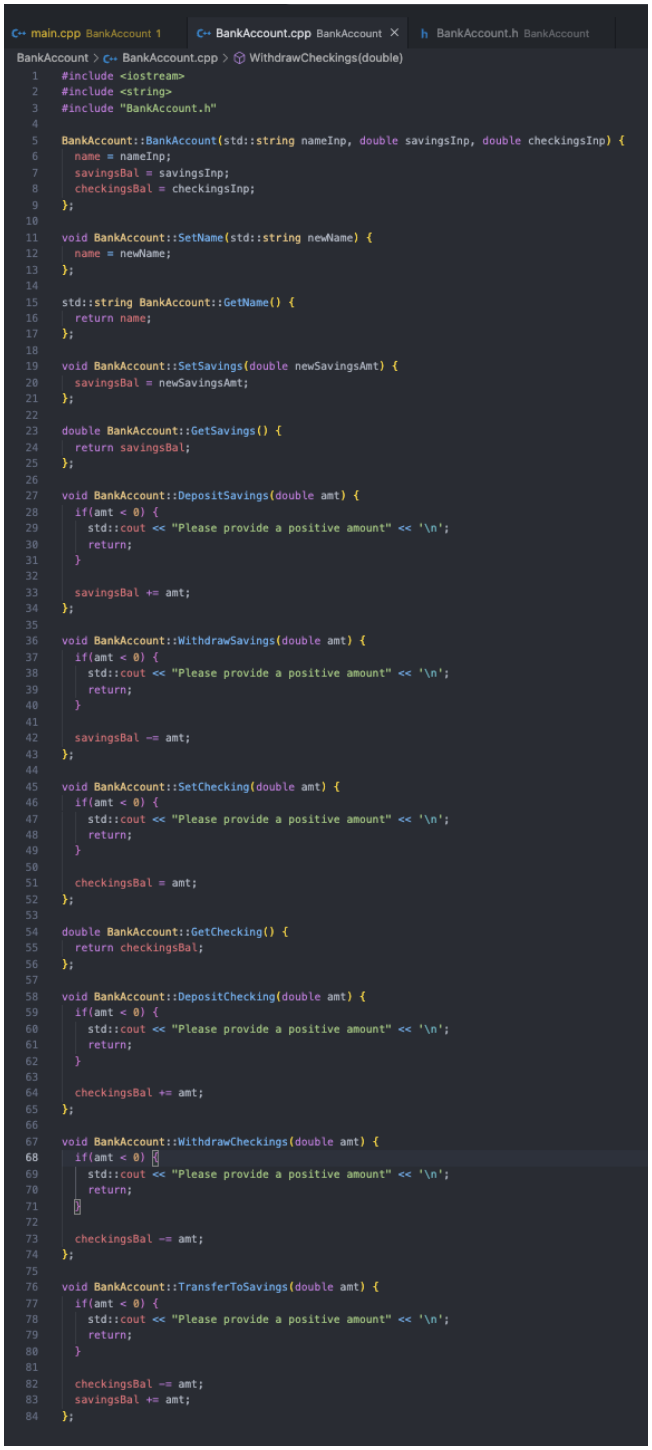
Lab Explanation

The requirements for this assignment were to create a bank account program that uses classes to store and organize data. This class would be included from another external file.

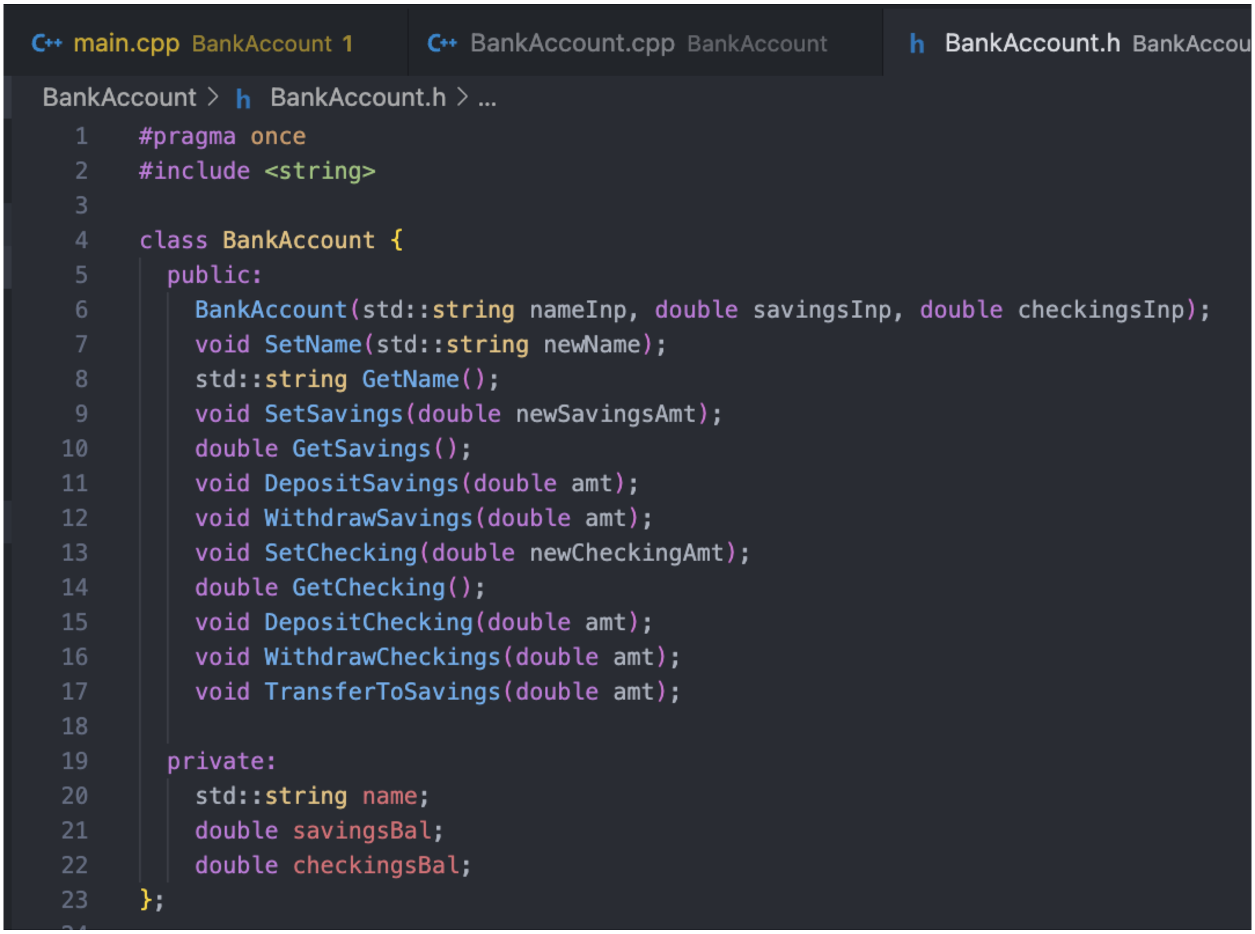
Code



```cpp
#include <iostream>
#include <iomanip>
#include "BankAccount.h"
using std::cout, std::fixed, std::setprecision;

int main() {
  BankAccount account("Maria", 1000.00, 1000.00);
  account.SetChecking(500.00);
  account.SetSavings(500.00);
  account.WithdrawSavings(100.00);
  account.WithdrawCheckings(100.00);
  account.TransferToSavings(300.00);

  cout << account.GetName() << '\n';
  cout << fixed << setprecision(2);
  cout << account.GetChecking() << '\n';
  cout << account.GetSavings() << '\n';

  return 0;
}
```


main.cpp BankAccount 1
BankAccount.cpp BankAccount
BankAccount.h BankAccount
BankAccount > BankAccount.cpp > WithdrawCheckings(double)

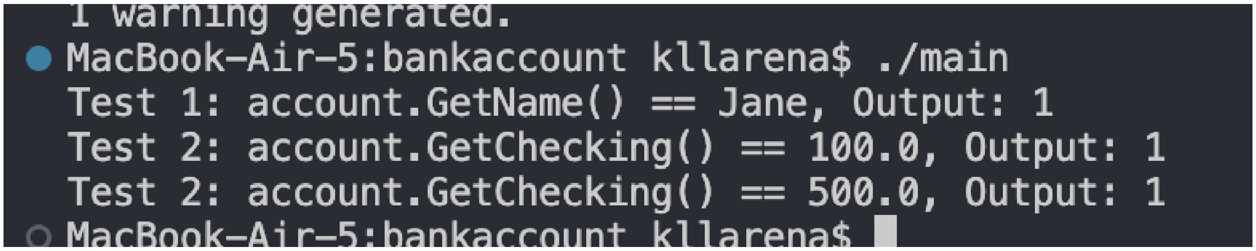
```cpp
#include <iostream>
#include <string>
#include "BankAccount.h"

BankAccount::BankAccount(std::string nameInp, double savingsInp, double checkingsInp) {
  name = nameInp;
  savingsBal = savingsInp;
  checkingsBal = checkingsInp;
};

void BankAccount::SetName(std::string newName) {
  name = newName;
};

std::string BankAccount::GetName() {
  return name;
};

void BankAccount::SetSavings(double newSavingsAmt) {
  savingsBal = newSavingsAmt;
};

double BankAccount::GetSavings() {
  return savingsBal;
};

void BankAccount::DepositSavings(double amt) {
  if(amt < 0) {
    std::cout << "Please provide a positive amount" << '\n';
    return;
  }

  savingsBal += amt;
};

void BankAccount::WithdrawSavings(double amt) {
  if(amt < 0) {
    std::cout << "Please provide a positive amount" << '\n';
    return;
  }

  savingsBal -= amt;
};

void BankAccount::SetChecking(double amt) {
  if(amt < 0) {
    std::cout << "Please provide a positive amount" << '\n';
    return;
  }

  checkingsBal = amt;
};

double BankAccount::GetChecking() {
  return checkingsBal;
};

void BankAccount::DepositChecking(double amt) {
  if(amt < 0) {
    std::cout << "Please provide a positive amount" << '\n';
    return;
  }

  checkingsBal += amt;
};

void BankAccount::WithdrawCheckings(double amt) {
  if(amt < 0) {
    std::cout << "Please provide a positive amount" << '\n';
    return;
  }

  checkingsBal -= amt;
};

void BankAccount::TransferToSavings(double amt) {
  if(amt < 0) {
    std::cout << "Please provide a positive amount" << '\n';
    return;
  }

  checkingsBal -= amt;
  savingsBal += amt;
};
```


main.cpp BankAccount 1
BankAccount.cpp BankAccount
BankAccount.h BankAccou
BankAccount > BankAccount.h > ...

```cpp
#pragma once
#include <string>

class BankAccount {
  public:
    BankAccount(std::string nameInp, double savingsInp, double checkingsInp);
    void SetName(std::string newName);
    std::string GetName();
    void SetSavings(double newSavingsAmt);
    double GetSavings();
    void DepositSavings(double amt);
    void WithdrawSavings(double amt);
    void SetChecking(double newCheckingAmt);
    double GetChecking();
    void DepositChecking(double amt);
    void WithdrawCheckings(double amt);
    void TransferToSavings(double amt);

  private:
    std::string name;
    double savingsBal;
    double checkingsBal;
};
```

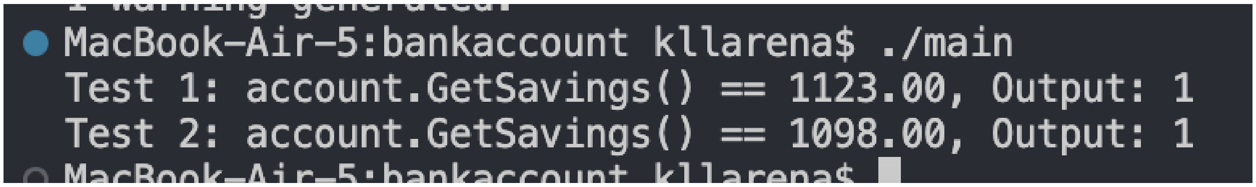
These are pictures of my code from each of the files I created. The 1st picture is my main.cpp, the 2nd one is my BankAccount.cpp file, and my 3rd one is my BankAccount.h file. I decided to not include the entire std library in my header and cpp files to prevent potential naming conflicts and improve code clarity and maintainability by explicitly referencing elements from the "std" namespace. Additionally, I used #pragma once instead of #ifndef and #endif since it was cleaner to me.

Test 1



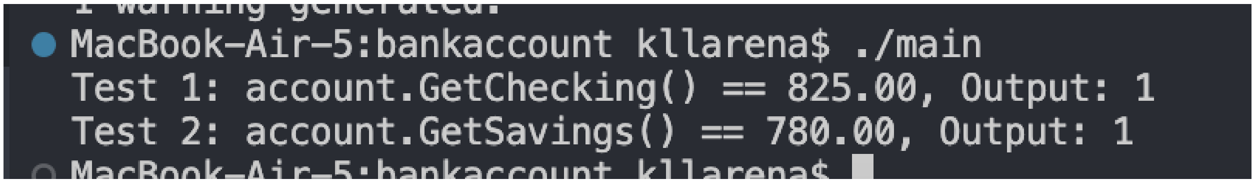
```
  1 warning generated.
● MacBook-Air-5:bankaccount kllarena$ ./main
  Test 1: account.GetName() == Jane, Output: 1
  Test 2: account.GetChecking() == 100.0, Output: 1
  Test 2: account.GetChecking() == 500.0, Output: 1
○ MacBook-Air-5:bankaccount kllarena$ 
```

Test 2



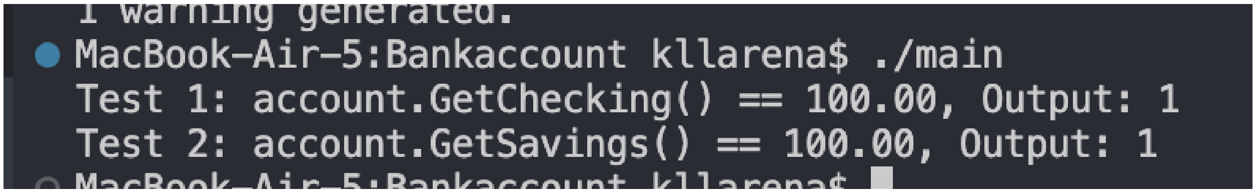
```
  1 warning generated.
● MacBook-Air-5:bankaccount kllarena$ ./main
  Test 1: account.GetSavings() == 1123.00, Output: 1
  Test 2: account.GetSavings() == 1098.00, Output: 1
○ MacBook-Air-5:bankaccount kllarena$ 
```

Test 3



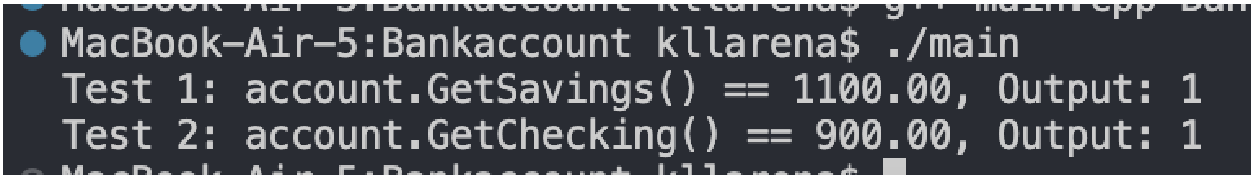
```
  1 warning generated.
● MacBook-Air-5:bankaccount kllarena$ ./main
  Test 1: account.GetChecking() == 825.00, Output: 1
  Test 2: account.GetSavings() == 780.00, Output: 1
○ MacBook-Air-5:bankaccount kllarena$ 
```

Test 4



```
  1 warning generated.
● MacBook-Air-5:Bankaccount kllarena$ ./main
  Test 1: account.GetChecking() == 100.00, Output: 1
  Test 2: account.GetSavings() == 100.00, Output: 1
○ MacBook-Air-5:Bankaccount kllarena$ 
```

Test 5



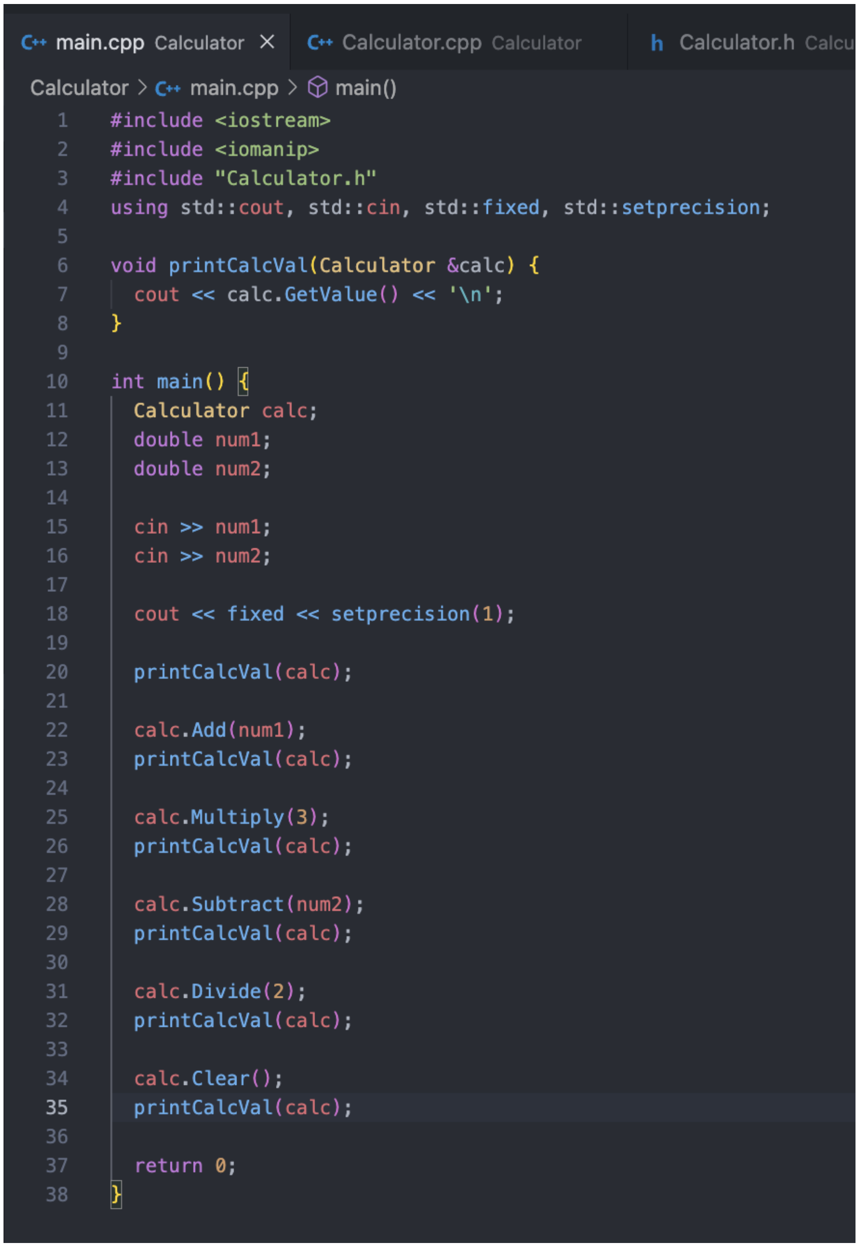
```
● MacBook-Air-5:Bankaccount kllarena$ g++ main.cpp Ban
● MacBook-Air-5:Bankaccount kllarena$ ./main
  Test 1: account.GetSavings() == 1100.00, Output: 1
  Test 2: account.GetChecking() == 900.00, Output: 1
○ MacBook-Air-5:Bankaccount kllarena$ 
```

Assignment 2

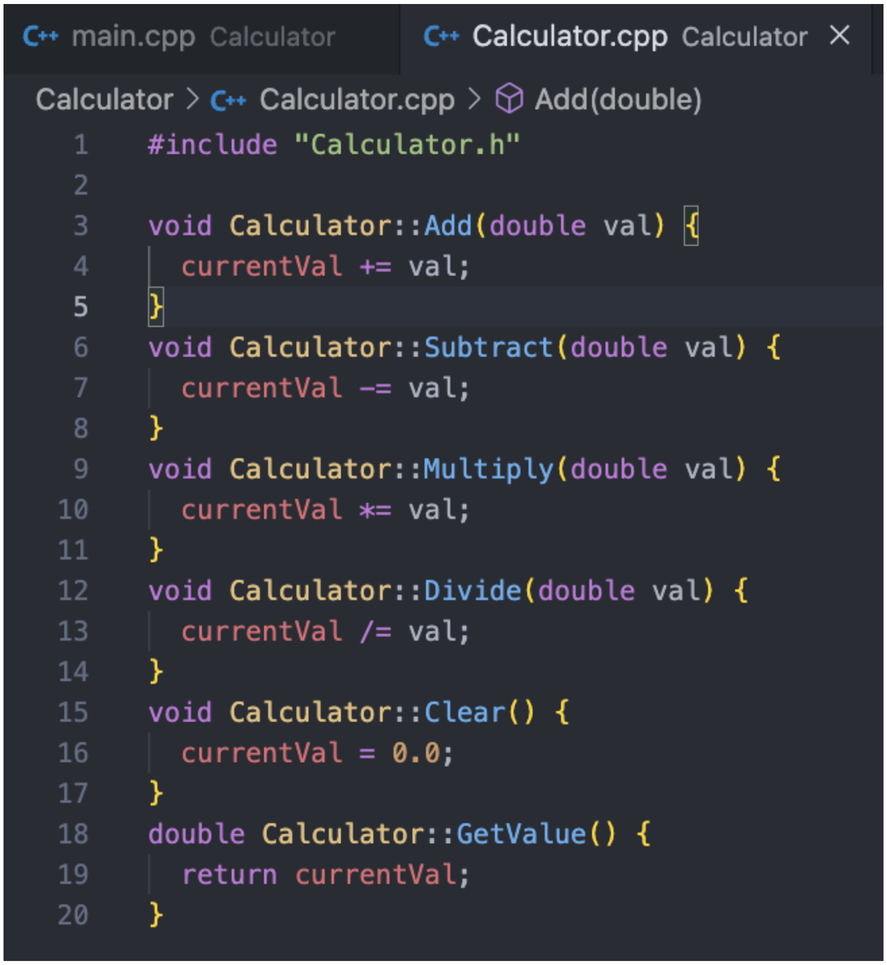
Lab Explanation

The requirements for this assignment were to create a calculator app that would use a calculator class to handle the calculations. This calculator class would be included from another external file.

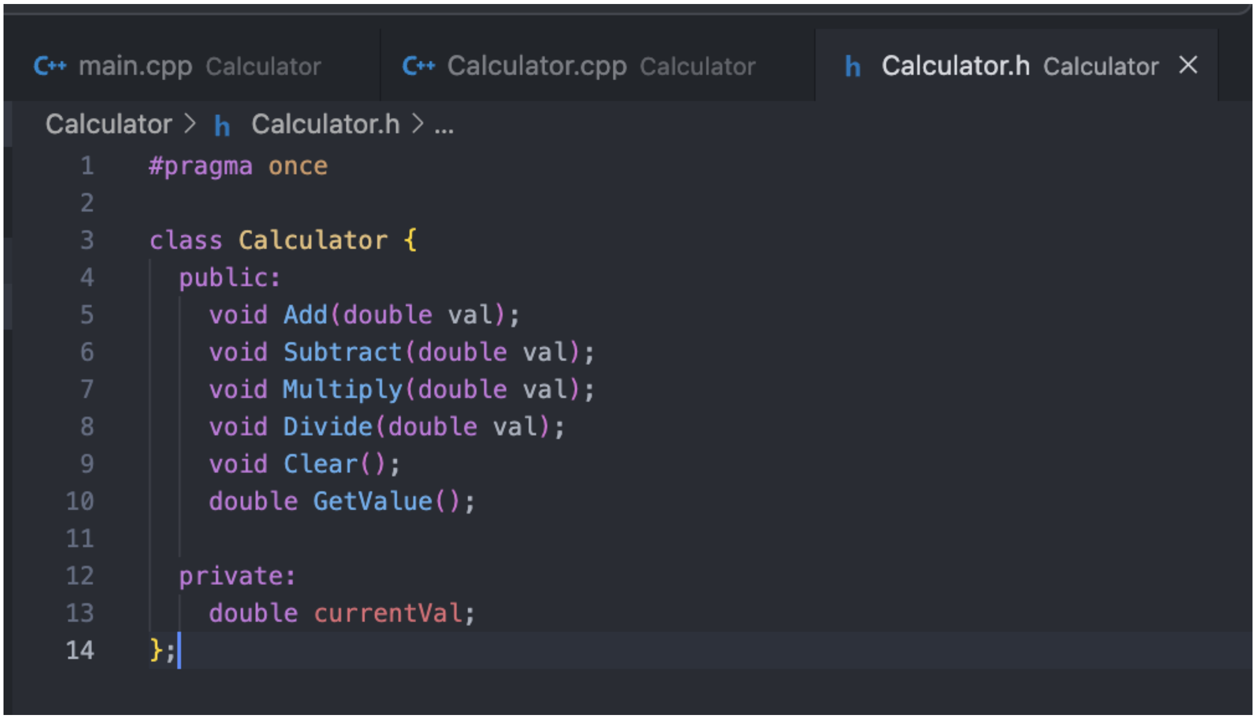
Code



```cpp
#include <iostream>
#include <iomanip>
#include "Calculator.h"
using std::cout, std::cin, std::fixed, std::setprecision;

void printCalcVal(Calculator &calc) {
  cout << calc.GetValue() << '\n';
}

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

  cin >> num1;
  cin >> num2;

  cout << fixed << setprecision(1);

  printCalcVal(calc);

  calc.Add(num1);
  printCalcVal(calc);

  calc.Multiply(3);
  printCalcVal(calc);

  calc.Subtract(num2);
  printCalcVal(calc);

  calc.Divide(2);
  printCalcVal(calc);

  calc.Clear();
  printCalcVal(calc);

  return 0;
}
```


main.cpp Calculator
Calculator.cpp Calculator
Calculator > Calculator.cpp > Add(double)

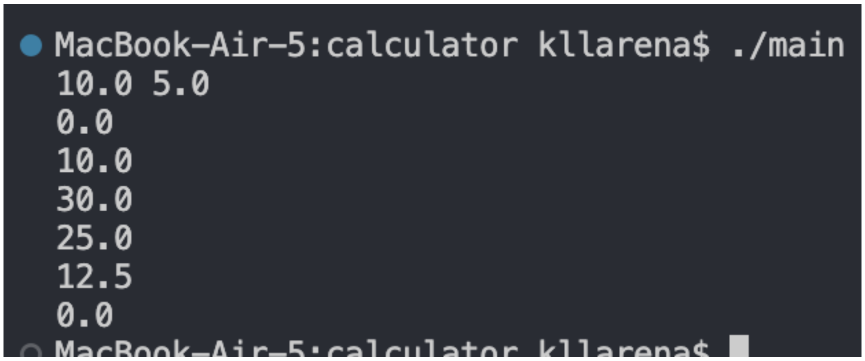
```cpp
#include "Calculator.h"

void Calculator::Add(double val) {
  currentVal += val;
}
void Calculator::Subtract(double val) {
  currentVal -= val;
}
void Calculator::Multiply(double val) {
  currentVal *= val;
}
void Calculator::Divide(double val) {
  currentVal /= val;
}
void Calculator::Clear() {
  currentVal = 0.0;
}
double Calculator::GetValue() {
  return currentVal;
}
```



```cpp
#pragma once

class Calculator {
  public:
    void Add(double val);
    void Subtract(double val);
    void Multiply(double val);
    void Divide(double val);
    void Clear();
    double GetValue();

  private:
    double currentVal;
};
```

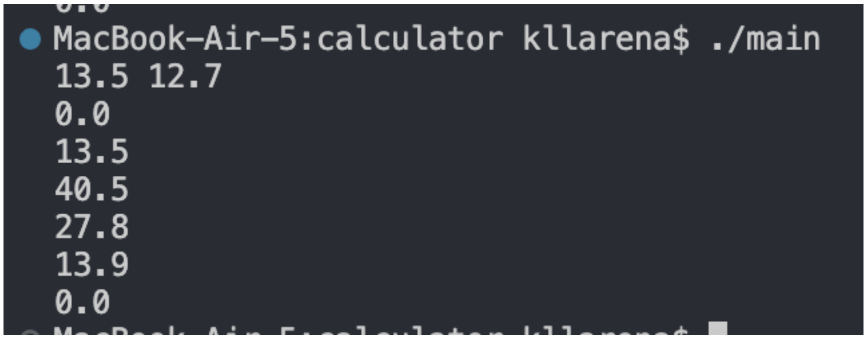
These are pictures of my code from each of the files I created. The 1st picture is my main.cpp, the 2nd one is my BankAccount.cpp file, and my 3rd one is my BankAccount.h file. I decided to not include the entire std library of my files to prevent potential naming conflicts and improve code clarity and maintainability by explicitly referencing elements from the "std" namespace. Additionally, I used #pragma once instead of #ifndef and #endif since it was cleaner to me.

Test 1



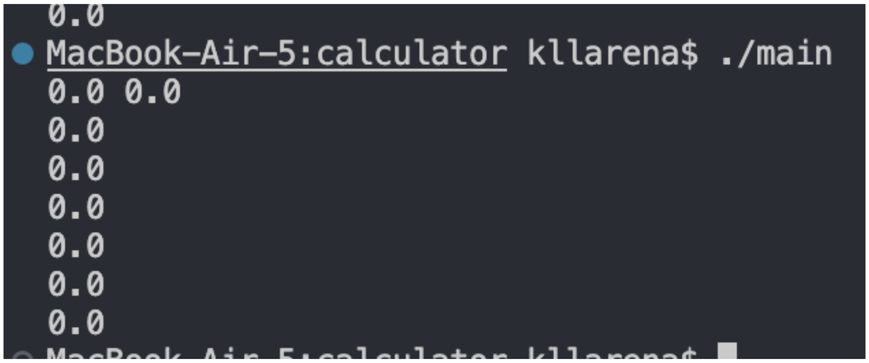
```
● MacBook-Air-5:calculator kllarena$ ./main
  10.0 5.0
  0.0
  10.0
  30.0
  25.0
  12.5
  0.0
○ MacBook-Air-5:calculator kllarena$ █
```

Test 2



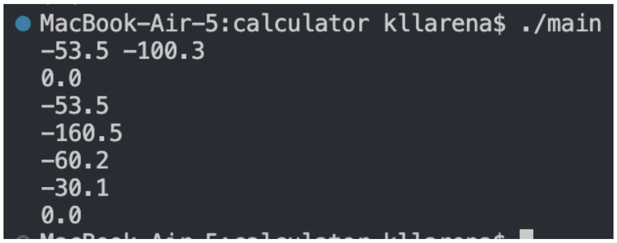
```
  0.0
● MacBook-Air-5:calculator kllarena$ ./main
  13.5 12.7
  0.0
  13.5
  40.5
  27.8
  13.9
  0.0
○ MacBook-Air-5:calculator kllarena$ █
```

Test 3



```
  0.0
● MacBook-Air-5:calculator kllarena$ ./main
  0.0 0.0
  0.0
  0.0
  0.0
  0.0
  0.0
  0.0
○ MacBook-Air-5:calculator kllarena$ █
```

Test 4



```
● MacBook-Air-5:calculator kllarena$ ./main
  -53.5 -100.3
  0.0
  -53.5
  -160.5
  -60.2
  -30.1
  0.0
○ MacBook-Air-5:calculator kllarena$ █
```