Kolbe Williams-Wimmer

C++ Midterm 2 Extra Credit:

Write a program that shows the usage of the friend keyword:

The friend keyword can be used to allow a function or class that is not a member of a class to access private members of the class that it is not a part of.

This program contains two classes, BankAccount and BankManager. The BankAccount class holds information about a user's bank account (their balance and account number). This information is sensitive, so it is part of the class's private members. However, The BankManager class is designed to create manager objects that can modify the user's account. The only way that the BankManager class can access and modify the private members of the BankAccount class is if it is a friend of that class. Then, in the main method, an object of the BankManager class is created that can modify the user's account, which is represented by an object of the BankAccount class. The user can then use the BankManager object to make deposits and withdraw from their account while keeping their data private.

Source Code:

```
#include <iostream>
#include <string>
using namespace std;

//BankAccount class hold bank account information, such as balance and account
number
class BankAccount
{
private:
	double balance;
	int accountNumber;
public:
	//The BankManager class is friends with this class, so it can access its
private members
	friend class BankManager;
	BankAccount(double bal, int num)
	{
		balance = bal;
		accountNumber = num;
	}

	int getAccountNumber()
	{ return accountNumber; }

	double getBalance()
	{ return balance; }
};

//BankManager class manages the accounts created in the BankAccount class, so it has
to be able to
//access the private members of the BankAccount class so that it can update the
balance.
```

```
//This means that is has to be friends with the BankAccount class, wich is defined
in the public section
//of the BankAccount class
class BankManager
{
private:
	string name;
public:
	BankManager(string managerName)
	{
		name = managerName;
	}

	double increase(BankAccount &obj, double amount)
	{ return obj.balance += amount; }

	double decrease(BankAccount &obj, double amount)
	{ return obj.balance -= amount; }
};

int main()
{
	//A bank account is created with user given information
	double accountBal;
	int accountNum;
	string managerName;
	cout << "Enter the amount you want to start your account with: ";
	cin >> accountBal;
	cout << "Enter your new account's number: ";
	cin >> accountNum;
	BankAccount b1(accountBal, accountNum);
	cout << "You have started an account with a balance of $" << b1.getBalance()
<<
		" and an account number of " << b1.getAccountNumber() << endl;

	//User gives the name of the account manager, and the created BankManager
object can modify
	//the user's account since the BankManager class is friends with the
BankAccount class
	cout << "Enter the name of your account manager: ";
	cin.ignore();
	getline(cin, managerName);
	BankManager m1(managerName);

	//The user can chose to deposite or withdraw monay from their account to
update the balance
	//through the object of the BankManager class
	char decision, decision2;
	double amount;
	cout << "Would you like to deposite to your account?  (y/n) ";
	cin >> decision;
	if (decision == 'y')
	{
		cout << "How much would you like to deposite? ";
		cin >> amount;
		m1.increase(b1, amount);
	}
	else if (decision == 'n')
```

```
	{
		cout << "Would you like to withdraw money from your account? (y/n) ";
		cin >> decision2;
		if (decision2 == 'y')
		{
			cout << "How much would you like to withdraw? ";
			cin >> amount;
			m1.decrease(b1, amount);
		}
		else if (decision2 != 'y' && decision2 != 'n')
		{
			cout << "Invalid Input...\n";
		}
	}
	else
	{
		cout << "Invalid Input...\n";
	}
	cout << "Thank you, your account balance is: " << b1.getBalance() << endl;
	return 0;
}
```

Output: