HW3: Bank Accounts - using Classes with Member Functions and Separate Compilation:

You have been hired as a programmer by a major bank. Your first project is a small banking transaction system. Each account consists of a number and a balance. The user of the program (the teller) can create a new account, as well as perform deposits, withdrawals, balance inquiries, close accounts, etc.

For this assignment, **you must use Classes** and move functionality into the classes. Specifically, you should have at least the following classes:
1. A **Bank** class which consists of an **array of Accounts** and the **number of accounts** currently active in the bank.
2 An **Account** class which consists of a **Depositor, an account number, an account type, and a balance**.
3 A **Depositor** class which has a **Name and a social security number**.
4 A **Name** class which consists of **first and last names**.

You must **add appropriate member functions** (or methods) to each class so as to implement the functionality of each of the functions of the previous assignment (HW2). Each of the functions of HW2 should be re-implemented using a class method. (You will have to decide as to which class each method belongs.) In addition, **each class should minimally have a default constructor and possibly additional parametized constructors**. (Some of the HW2 functions may become constructors.) Also, consider whether any of your classes needs a destructor. Remember, all I/O should be done outside of any class method.

The **data members of each class must be private**. As such, you may need to provide accessor and mutator functions.

You must use **separate compilation** of your program modules: **Bank.h, Bank.cpp, Account.h, Account.cpp, Depositor.h, Depositor.cpp, Name.h, Name.cpp, as well as the module containing the main() function**.

As in previous assignments, initially, the account information of existing customers is to be read into the database. The bank can handle a maximum of **MAX_NUM** accounts. The program keeps tracks of the actual number of currently active accounts. **A table of the initial database of active accounts should be printed.**

As before, the program then allows the user to select from the following menu of transactions:

Select one of the following:
W - Withdrawal
D - Deposit
N - New account
B - Balance
I - Account Info
X - Delete Account
Q - Quit

Once the user has entered a selection, appropriate functions should be called to perform the specific transaction. **At the end, before the user quits, the program prints the contents of the final database**.

As in previous assignments, make sure to use enough test cases so as to completely test program functionality.

EXTRA CREDIT:
1. Use a **Screen Input Form** for entering the new account information.
2. Create and use a **TransactionInputForm class** for use in entering the new account information.
3. Create and use a **TransactionInputForm class** for use in **all** transactions.

HINT: one possible approach
1a. The Bank class should **at least** have a default constructor that would allow statements of the form:
```
Bank bank;
```
1b. The Bank class should have **at least** have the following methods:
```
bool openAccount(...)
int findAccount(...)
int findAccountSSN(...)
Account getAccount(...)
bool deleteAccount(...)
```
2a. The Account class should **at least** have a constructor that would allow statements of the form:
```
account[j] = Account(...);
```
2b. The Account class should have **at least** the following methods:
```
double getBalance(...)
Depositor getDepositor(...)
int getAccountNumber(...)
string getAccountType(...)
void makeDeposit(...)
bool makeWithdrawal(...)
```

3a. The Depositor class should **at least** have a constructor that would allow statements of the form:
    depositor = Depositor(...);
3b. The Depositor class should **at least** the following methods:
    Name getName(...)
    string getSSN(...)
4a. The Name class should **at least** have a constructor that would allow statements of the form:
    name = Name(...);
4b. The Name class should **at least** the following methods:
    string getFirstname(...)
    string getLastname(...)
5. Add additional constructors and methods to each class as necessary.
6. All I/O should be done outside of the class implementations.