HW7: Bank Accounts: Using Inheritance, Polymorphism, AbstractClasses and Methods

Modify HW6 by adding several **subclasses** to the Account Class.

The **Account** class consists of a **Depositor, an account number, an account type, account status (open or closed), account balance, and an ArrayList of TransactionReceipts.**

The **SavingsAccount** class is a subclass of the Account class.
For Saving Accounts, deposits and withdrawals are allowed at any time.

The **CheckingAccount** class is a subclass of the Account class.
For Checking Accounts, deposits, withdrawals, and check clearing are allowed at any time. Remember, you may only clear a check if the date on the check is no more than six months ago. No post-dated checks (checks with a future date) may be cleared. Use the **Calendar class** to implement this. In addition, a check will clear only if there is sufficient funds in the account. If the account lacks sufficient funds, the check will not clear and the account will be charged a $2.50 Service Fee for "bouncing" a check. In addition to the previous rules, if the current balance of the account is below $2500, each withdrawal or cleared check is charged a fee of $1.50.

The **CDAccount** class is a subclass of the Account class.
The class has a data member: **a maturityDate** which is a **Calendar** class object.
As before, deposits and withdrawals will be allowed only on or after the maturity date. When a deposit or withdrawal is made, have the user select a new maturity date for the CD. The choices are either 6, 12, 18, or 24 months from the date of the deposit or withdrawal. Again, use the **Calendar** class to implement this.

An Account object should access subclass methods using **polymorphism**.

Rewrite classes and methods as appropriate to implement the above subclasses.:

Extra Credit 1:
Implement the CDAccount Class as a subclass of the SavingsAccount Class.

Extra Credit 2:
Create **abstract classes** called **genName, genDepositor, genAccount, genTransactionTicket, genTransactionReceipt.** These classes should respectively have the same data members as the Name, Depositor, Account, TransactionTicket and TransactionReceipt classes. You should include **abstract getter and setter methods** in each abstract class. You should then implement the Name, Depositor, Account, TransactionTicket, and TransactionReceipt classes as subclasses of the respective, generic, abstract ones.

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

Submission Requirements:
Create a folder on Google Drive that will contain the following:
1. The source files (i.e., *.java files) for each of the implemented Classes:
pgmHW7.java
Bank.java; Account.java; Depositor.java, Name.java
Check.java; TransactionTicket.java; TransactionReceipt.java;
SavingsAccount.java; CheckingAccount.java; CDAccount.java;
etc.
2. The text file containing the initial database of accounts (e.g., initAccounts.txt)
3. The test cases text file (e.g., myTestCases.txt)
4. The output text file which contains all of the required program output (e.g., pgmOutput.txt)
Then, make the folder shareable and send me a link to the folder.