ICS4U Assignment: Database

In this assignment, you will design and implement a database system for VP Bank. You must include multiple classes and interactions among different classes to demonstrate OO concepts and design.

VP Bank has three main products, as follows. A customer can have a maximum of three of these following products each, but chequing accounts and credit cards are available only to customers over 18.

- Chequing Account
- Savings Account
- Credit Card

The main menu of the system should allow the user to select the following:

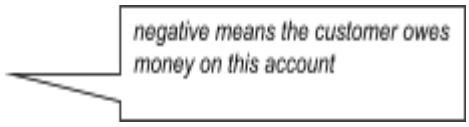
- Add a customer
    - User is required to enter customer information: name, SIN, birth date
    - User is required to select an account, with opening balance, to be added to the customer's profile
- Delete a customer
- Sort the database by customer name or SIN
- Display in sorted order the names and SINs of all customers
- Search for the profile of a customer by name or SIN. Once a profile is retrieved, its display should include the balances of all accounts. A submenu should allow the user to:

    - View account activity: for each account selected by the user, the system will display the last five transactions including their description, amount, and resulting balance.
    - Accept a deposit to a chequing or savings account.
    - Withdraw funds from a chequing or savings account, or a credit card.
    - Process a cheque drawn on a chequing account. If the balance is less than $1000 before a cheque is processed, $0.15 is charged for the cheque.
    - Process a purchase by credit card.
    - Process a payment from chequing or saving account to credit card.
    - Transfer funds between chequing and savings accounts.
    - Open an account or issue a credit card.
    - Cancel an account or credit card. The credit card cannot be canceled if there is still outstanding balance. If there are no accounts or credit cards left in a customer's profile after the cancellation, the profile is deleted from the system.

Assume that VP Bank will never have more than 50 customers at any one time. When the system starts, it should ask the user for the name of the data file which contains customers' information and their account balances. The data file (.txt) will have the following format:

< Last Name>	Jones
< First Name>	Ann
< SIN>	500700234
<Birth Year>	1973
<Birth Month>	3
<Birth Day>	30
<Saving Account Balance>	1250.34
<Chequing Account Balance>	none
<Credit Card Balance>	-569.34
…….	

none means this customer does not have a chequing account



Before execution terminates, the system should write all information, with the prescribed format, to the data file.

Your OO design will incorporate, **at a minimum,** the following classes: Customer, ChequingAccount, SavingAccount, CreditCard, and Bank (which contains the main program), and a text file (.txt) for customer information. Besides these classes, you might want to consider other classes as well.

The main menu should look like this:

```
Welcome to the VP bank.
-----------------------
Please choose an action from the following:
      1: Add a customer
      2: Delete a customer
      3: Sort customers by last name, first name
      4: Sort customers by SIN
      5: Display customer summary (name, SIN)
      6: Find profile by last name, first name
      7: Find profile by SIN
      8: Quit
```

After finding a profile, the system should offer the following submenu:

```
PROFILE MENU
-----------------------
      1: View account activity
      2: Deposit
      3: Withdraw
      4: Process cheque
      5: Process purchase
      6: Process payment for credit card
      7: Transfer funds
      8: Open account or issue card
      9: Cancel account or card
     10: Return to main menu
```

Assessment Rubrics

Thinking / Inquiry

UML Diagram /8

0	2	4	6	8
No UML diagram standard is followed	Few UML diagram standards are followed	Some UML diagram standards are followed	Most UML diagram standards are followed	All UML diagram standards are followed

OO Design & Classes Properly Constructed /10

0	4	6	8	10
The solution has a procedural design	The solution consists of classes, but they are not representation of logical objects	The solution has an object-oriented design, but some functions are defined in the wrong class	The solution has an object-oriented design, but a few functions are defined in inappropriate classes	The solution has a good object-oriented design: each class has a well-defined purpose

Knowledge / Understanding

Correctness, Functions & Efficiency of Program /10

0	2	4	6	8	10
Program is not implemented or does not compile	Program compiles but does not perform any required functions	Program performs a few required functions with limited success and efficiency	Program performs some required functions with some success and efficiency	Program performs most required functions with considerable success and efficiency	Program performs all required functions with complete success and efficiency

File Input / Output Implemented & Exceptions handled Properly /10

0	2	4	6	8	10
Program does not implement file input and output; no exception can be handled	Program implements file input, output, and exception handling, but these do not work as required	Program implements file input, output, and exception handling with limited success	Program implements file input, output, and exception handling with some success	Program implements file input, output, and exception handling with considerable success	Program implements file input, output, and exception handling with complete success

Customer is added & deleted properly /10

0	2	4	6	8	10
Program does not add or delete customers	Customer addition and deletion do not work as required	Program adds and deletes customers with limited success	Program adds and deletes customers with some success	Program adds and deletes customers with considerable success	Program adds and deletes customers with complete success

Communication

Programming Style /12

(Check the Style Guide at http://www.touque.ca/EC/programming/Java/resources/Style_Guide.php)

0	4	8	12
No good programming style is evident	Few requirements of good programming style have been met	Most requirements of good programming style have been met	All requirements of good programming style have been met

Application

Sorting Implemented Properly /10

0	2	4	6	8	10
Program does not implement sorting	Program implements sorting, but it does not work as required	Program sorts by SIN or name with limited success	Program sorts by SIN and name with some success	Program sorts by SIN and name with considerable success	Program sorts by SIN and name with complete success

Searching Implemented Properly /10

0	2	4	6	8	10
Program does not implement searching	Searches do not work as required	Program searches by SIN or name with limited success	Program searches by SIN and name with some success	Program searches by SIN and name with considerable	Program searches by SIN and name with complete success

Services Implemented Properly /20

0	2	4	6	8	10
No services implemented	Services do not work as required	A limited number of services implemented with limited success	Some services implemented with some success	Most services implemented with considerable success	All services implemented with complete success

Total: /100

Side notes for the teacher

The four pillars for OOP are Abstraction, Encapsulation, Inheritance, Polymorphism. This is a good assignment for OOP (Objected-Oriented programming) as it requires students to demonstrate their understanding of OOP of how to use:

1. Abstraction: the process of showing only essential/necessary features of an entity/object to the outside world and hide the other irrelevant information.

   For example, in this assignment, students will create a Customer class to store all the information of a customer.

2. Encapsulation: packing data and member function (Method) together into a single unit like class. It automatically achieves the concept of data hiding providing security to data.

   For example, student will make variables as *private* to hide the information and create some accessor and mutator method to access the private data if needed

3. Inheritance: creating a new class from an existing class, allowing a class (subclass) to acquire the properties and behavior of another class (super-class). It helps to reuse, customize and enhance the existing code.

   In this assignment, students can create ChequingAccount, SavingAccount and CreditCard classes from Account class.

4. Polymorphism: the ability of subclass being defined its own unique behavior while sharing the same functionalities or behavior of its parent class. Overloading is an example of compiler time polymorphism and it has the same method with different signatures. Overriding is an example of run time polymorphism and it has the same method, same signature but different classes connected through inheritance.

   In this assignment, students will have some opportunities to use the concept of polymorphism and implement method overloading and overriding.

In addition, this assignment will require students to:

- use input/output file to read and write information
- utilize Exception catching to make the program robust and prevent it from crashing
- implement different sorting and searching algorithms
- apply good programming style and great programming documentation

Future Extension/Improvement

students can create user-friendly GUI (Graphical User Interface) program to fully implement functionalities if students are confident and comfortable with GUI