Java 2 Final Project

Banking Application: 20%

You are to create a simple banking application with the basic functionalities of checking, deposit, withdrawal, transfer, and credit.

Properties of a Bank:

- A Bank Account must have a unique identifier(account number)
- A Bank Account must have a name(this is the name of the beholder)
- A Bank Account must have an initial balance of 0.0 GMD
- A Bank Account must have an opening date
- A Bank Account must have a type(saving, current, and fixed deposit) account.

Class Properties(instances variables):

Create a `BankAccount` class with all the above properties. Choose the appropriate data type for the properties, and be cautious when creating this instance variable because properties like `accountNumber`, `accountName`, `openingDate`, and `accountType` will not be modified once created. These instance variables should be constant; find an appropriate modifier to restrict their modifications.

Constructor:

The `BankAccount` class should also process a constructor to initialize the instance variables. At every object creation time, this constructor should be invoked and passed in all the properties of a bank account.

Behaviors(function) of a Bank Account:

Create functions that would be called outside of this class to either access or modify the properties of your `BankAccount`. Remember not all properties are allowed to be modified, so therefore, choose wisely. As for the properties whose data wouldn't be modifiers do not necessarily need to have a setter.

Banking Class

A Bank Account needs some operational, like checking account info, depositing money, withdrawal money, transfers funds from one bank account to another or even taking credit from the bank. So, therefore, this class will perform all the basic operations of a bank account.

Checking: Create a function that will print information about a Bank Account, this function will take a bank account and print the information. This info will include the following; account number, name, current balance, opening date, and type.

Deposit: This function will enable an account's current balance to be modified with monetary value. It is ideal to check the amount been deposited before updating an account. For standards, no account should be credited with an amount less than GMD200.00. Therefore, check the input amount and throw an exception or print an error message and terminate the program otherwise use the input `bankAccount` call `setCurrentBalance` function from `bankAccount` and pass it the input amount.

Withdrawal: This function will take a `bankAccount` and an amount to be withdrawn, this function will first check if the inputted amount is less than GMD200.00 when it does throw an error and terminate function execution with an appropriate message. Otherwise, proceed with the withdrawal process. After deducting the inputted amount from the `currentBalance` check to see whether the leftover is less than GMD200.00 if it does terminate the program by printing an appropriate message moreover update the `currentBalance` with the leftover and acknowledge the user.

Transfer: Make it possible for one account(sender) to make a withdrawal, get that value, and deposit it to another account(receiver). In totality, this function would take three arguments; sender's account, receiver's account, and amount. Please, remember the **DRY** principle.

Credit: Banks can give credit to their customer(account holders), but there might be some criteria for an account to be eligible for a credit or loan. For the simplistic approach before granting a load to any bank account check to see whether the bank account has money over GMD5,000.00 otherwise do not grant a loan and terminate the program with an appropriate message.

Account Manager

This class will serve as an entry point for your application and an interface for interacting with the different components of your banking application. Within this class, you will utilize the collections to store bank account objects and perform all the functionalities of a bank.

Testing You App:

1. Create a Banking object that will be used to carry out all the functionality of the bank.
2. Create a `BankAccount` object to make sure the balance for some accounts is less than 200.00
3. Use a HashMap data structure to store your bank account objects
4. Using the banking object to test deposit functionality with a bank account object less than 200.00.
5. Continue testing your application with the rest of the functions from the banking class.

Your project will be marked based on the following criteria:

- Banak account class with the above-mentioned properties and functions
- Banking class with all functionalities (checking, deposit, withdrawal, transfer, and credit).
- The functionalities of the Banking should work seamlessly without any error.
- Error and exceptional handling
- AccountManager class
- Create an object of the banking class
- Create objects of bank account class
- Format your monetary value to standard money value for example `1,500.00` and append GMD before it; `GMD1,500.00`
- Provide a clear and cohesive documentation for your functions(Java Standard docs)
- Have a thorough understanding of your project and can be able to defend it well.

Grading for your project:

CRITERIA	WEIGHT (%)	DESCRIPTION
Presentation	40%	Clarity, delivery, and explanation of the project.
Documentation	15%	Completeness, clarity, and quality of the project's report or documentation.
Code Structure	20%	Organization, readability, and adherence to coding standards.
Final Output	25%	Functionality, correctness, and efficiency of the final program or product.

Happy coding!