BANK ACCOUNT MANAGEMENT SYSTEM

Object-Oriented Analysis (OOA) Model:

In the bank account management system, the main objects are identified as Transaction, Account, SavingsAccount, and Customer:

- Transaction represents each deposit, withdrawal, or transfer, serving as a record of balance changes in an account.
- Account (regular account) manages the essential information of an account, such as the account number, current balance, account holder’s name, and transaction history.
- SavingsAccount is a specialized variant of Account, adding an interest rate and a minimum balance requirement.
- Customer represents the account owner, managing the list of regular accounts and savings accounts they own.

Regarding attributes, each object has its own characteristics:

- Transaction includes the amount, transaction type, and date.
- Account stores the account number, balance, owner’s name, and a list of transactions (transactionHistory).
- SavingsAccount inherits all attributes of Account while extending them with an interestRate and minBalance (minimum balance).
- Customer stores the customer’s name, identification number (ID), a list of regular accounts (ownedAccounts), and savings accounts (ownedSavingsAccounts).

For methods, the classes provide specific behaviors:

- Transaction mainly describes data and therefore does not include complex methods.
- Account offers operations such as deposit() for deposits, withdraw() for withdrawals, balanceInquiry() to check the balance, the += operator to add

transactions to the history, and compareAccount() to compare balances between two accounts.

- SavingsAccount inherits the methods of Account while overriding the withdraw() method to calculate interest before withdrawal and to verify the minimum balance requirement.
- Customer provides account management operations such as displayInfo() to show customer details, openNewAccount() to create a new account, calculateTotalBalance() to calculate the total balance, and transfer() to move funds between accounts.

With regard to inheritance relationships, the system has a clear hierarchy: SavingsAccount inherits from Account. This allows the savings account to reuse all data and behaviors of a regular account, while adding its own attributes and constraints such as the interest rate and minimum balance, as well as adjusting the withdraw() method to comply with the specific rules of a savings account.

Explanation About Class Design

The class design of the bank account management system is built to clearly separate responsibilities and promote code reuse. The Transaction class acts as a simple data container, recording information about each deposit, withdrawal, or transfer through its attributes: amount, transaction type, and execution date. The Account class manages the core behaviors and data of a bank account, including storing the account number, balance, owner’s name, and transaction history. It also defines key operations such as depositing, withdrawing, and checking the balance, forming the foundation for managing financial activities. SavingsAccount is a specialized variant of Account that leverages inheritance to reuse all attributes and methods from the base class, while extending them with additional properties such as an interest rate and a minimum balance requirement. By overriding the withdraw method, SavingsAccount adapts the inherited behavior to include interest calculation and balance validation according to its own constraints.

Operator overloading is applied to improve the usability of the classes. The += operator is overloaded in the Account class to allow transactions to be added to the account history in a concise and intuitive way, making the management and updating of the transaction list easier. In addition, a comparison function is implemented through compareAccount, enabling two accounts to be compared based on their balances. This design choice simplifies client code and makes interaction with account objects more natural, reflecting how such comparisons are usually made in real-world scenarios. The combination of inheritance and operator overloading helps the system achieve flexibility, maintainability, and readability, while ensuring that the code structure remains clear when new account types or methods are introduced in the future.

Code Walkthrough

The class design of the bank account management system focuses on separating responsibilities and reusing code. Transaction stores basic data about each deposit, withdrawal, or transfer. Account handles essential account information, including the account number, balance, owner’s name, and transaction history, and provides core operations like deposit, withdraw, and balance inquiry. SavingsAccount extends Account through inheritance, adding an interest rate and a minimum balance while overriding the withdraw method to include its own rules. Operator overloading is used in Account to simplify interaction: the += operator lets transactions be added to the history easily, and a comparison function allows balances between accounts to be evaluated. Together, inheritance and operator overloading make the system clear, flexible, and easy to maintain as new features are introduced.

Test Result:

- Test case 1:

+ Input:

```
1
Regular
ACC003
```

+ Output:

```
Name: Nguyen Khanh Hung
ID: C001

Choose function:
1. Open new account
2. Deposit
3. Withdraw
4. Transfer
5. Show total balances
6. Compare 2 accounts
Choose: 1
=====================
Enter account type (Regular / Savings): Regular
Enter account number: ACC003
Enter owner name: Nguyen Khanh Hung
Open new account sucessful!

Account number: ACC003
Current balance: 0 VND
```

+ Explanation about the test case:

In this test case, the user uses the open new account feature in the banking management program. After displaying the customer's information, the system presents a menu of functions, and the user selects option 1 to open an account. When the user be asked for the type of account to create, the user enters "Regular." Next, they provide the new account number "ACC003" and specify the account holder's name as "Nguyen Khanh Hung." The program receives this information, creates a new regular account object with an initial balance of 0, and adds it to the list of accounts owned by the customer. Finally, the system confirms that the account has been successfully opened and displays its details, including the account number and the current balance of 0 VND.

- Test case 2:

+ Input:

```
2
1
1
36000
```

+ Output:

```
Name: Nguyen Khanh Hung
ID: C001

Choose function:
1. Open new account
2. Deposit
3. Withdraw
4. Transfer
5. Show total balances
6. Compare 2 accounts
```

```
Choose: 2
======================
Choose type of account:
1. Regular
2. Savings
Choose: 1

Choose one of account numbers below:
1. ACC001
2. ACC002
Choose: 1

Enter the amount you want to deposit: 36000
Deposit sucessful!
Account number: ACC001
Current balance: 186000 VND
```

+ Explanation about the test case:

In this test case, the user uses the deposit function in the bank account management program. After the system displays the customer information “Nguyen Khanh Hung – C001” and the menu of functions, the user selects option 2 to perform a deposit. Next, the program asks for the type of account, and the user enters 1, which means selecting a regular account. The list of regular account numbers owned by the customer is displayed, including ACC001 and ACC002; the user chooses number 1, corresponding to account ACC001. The system then asks the user for the amount to deposit, and the user enters 36000 VND. The program adds this amount to the current balance of ACC001, then confirms that the deposit was successful and displays the account details, including the account number ACC001 and the new balance of 186000 VND.

About LLM Usage

During the coding process, I used ChatGPT to learn about savings accounts. I prompted: "What is the difference between regular accounts and savings accounts?" and "How to calculate interest?". Then, I selected the ideas suggested by ChatGPT to create my own class SavingsAccount.