Bank Account Management System – Report

Student Name: Trần Gia Kiệt - 24110103

Course: Object-Oriented Programming

Assignment: Bank Account Management System

1. Object-Oriented Analysis (OOA)

Objects (Nouns):
- Account
- SavingsAccount
- Customer
- Transaction

Attributes:
- Account: accountNumber, balance, ownerName, history
- SavingsAccount: interestRate, withdrawalsThisPeriod (inherited: all from Account)
- Customer: name, customerID, accounts
- Transaction: amount, type, date, note

Methods (Verbs):
- Account: deposit(), withdraw(), transfer(), displayInfo(), operator+=, operator==
- SavingsAccount: withdraw() (override), applyInterest()
- Customer: openAccount(), displayInfo()
- Transaction: displayInfo()

Inheritance:
- SavingsAccount : public Account

2. Class Design and Implementation

Account is the base class for all account types.
SavingsAccount inherits from Account and overrides withdraw() with withdrawal limits and interest features.
Customer owns multiple accounts.
Transaction models money movements.
Operator overloading:
- operator+= → add transaction to account and update balance.
- operator== → compare two accounts (e.g., by balance).

3. Key Code Walkthrough

- Operator +=: updates account balance and history safely. Prevents overdrafts.
- Operator ==: compares account balances to check equality.
- Virtual Functions: displayInfo() in Account is overridden in SavingsAccount.
- Polymorphism: Account* acc = new SavingsAccount(...); acc->displayInfo(); calls the derived method.

4. Testing

Test cases included:
- Creating customers and accounts.
- Depositing and withdrawing.
- Transfer between accounts.
- Applying interest to SavingsAccount.
- Using operator overloading (+=, ==).

Sample Output:
Customer: John Doe (ID: C001)
Account ACC1001 owned by John Doe | Balance: 500
[Deposit] 100 added.
[Savings Account] Interest rate: 2.5% applied.

5. UML Diagrams

Class Diagram (simplified):
Account
+ accountNumber : string
+ balance : double
+ ownerName : string
+ history : vector<Transaction>
+ deposit(), withdraw(), displayInfo(), operator+=, operator==

SavingsAccount
+ interestRate : double
+ withdrawalsThisPeriod : int
+ withdraw() override, applyInterest()

Customer
+ name : string
+ customerID : string
+ accounts : vector<Account*>
+ openAccount(), displayInfo()

Transaction
+ amount : double, type : enum, date : string, note : string

Sequence Diagram (Deposit Example):
Customer -> Account : deposit(amount)
Account -> Transaction : create new Transaction
Account -> Account : update balance
Account -> history : add transaction

6. Use of LLM (ChatGPT)

I used ChatGPT to:
- Brainstorm operator overloading ideas (+=, ==).
- Clarify differences between virtual and override.
The code and design remain my own; ChatGPT assisted in brainstorming and explanation.

7. Conclusion

The Bank Account Management System demonstrates OOP principles: encapsulation, inheritance, polymorphism, and operator overloading. It models real-world bank operations while keeping the design extensible for future features (e.g., fees, checking accounts).