OCA Task – Banking System

Project Overview

You will design and implement a **console-based Smart Banking System** that simulates basic banking operations for customers.

- The application must run **continuously in a loop** until the user chooses to exit.
- Upload the full project to GitHub and submit the repository link
- Do **NOT** use any AI tools

System Requirements

Main Menu

Class: `BankApplication`

```
Welcome to Smart Banking System
===============================
(1) Create New Customer
(2) Open Bank Account
(3) Deposit Money
(4) Withdraw Money
(5) Display Account Details
(6) Apply Monthly Processing
(7) Display All Customers
(8) Exit
Choose an option:
```

The menu must repeat until **option 8** is selected.

👤Customer Management

Class: `Customer`

Attributes (private)

- `customerId : int`
- `name : String`

- `accounts : List<BankAccount>`

Behaviors

- Add new account
- Retrieve account by account number
- Display customer details

Rules

- Customer name must **not be empty**
- Each customer can own **multiple accounts**

Account Hierarchy

Abstract Class: `BankAccount`

Attributes

- `protected int accountNumber`
- `protected double balance`

Abstract Method

```
public abstract void applyMonthlyUpdate();
```

Concrete Methods

- `deposit(double amount)`
- `withdraw(double amount)`
- `displayAccountInfo()`

SavingsAccount

Extends: `BankAccount`

Additional Attribute

- `interestRate : double`

Rules

- Interest is added during **monthly processing**
- No overdraft allowed

Overrides

- `applyMonthlyUpdate()`

CheckingAccount

Extends: `BankAccount`

Additional Attribute

- `monthlyFee : double`

Rules

- Monthly fee deducted during processing
- Overdraft allowed up to **-500**

Overrides

- `applyMonthlyUpdate()`
- `withdraw(double amount)`

When the user selects **Apply Monthly Processing**:

- Loop through **all customers**
- Loop through **all accounts**
- Call `applyMonthlyUpdate()` **polymorphically**

`instanceof` is **NOT allowed**

Business Rules

Deposit

- Amount must be **greater than 0**

Withdraw

- Amount must be **greater than 0**
- Savings account → balance cannot go below **0**
- Checking account → balance cannot go below **-500**

Input Validation

- Invalid numeric input must be rejected
- System must **not crash**

Display Requirements

Account Display Example

```
Account Number: 1001
Account Type  : Savings
Balance       : $1250.50
Interest Rate : 3.5%
```

Required Classes Summary

Class Name	Purpose
BankApplication	Main menu & system control
Customer	Customer data & accounts
BankAccount	Abstract base class
SavingsAccount	Interest-based account
CheckingAccount	Fee-based account
AccountNumberGenerator	Generate unique IDs