**Banking System with OOP:**

You have been hired as a **Software Engineer** to design a **Banking System** using **Object-Oriented Programming (OOP)** in Java. The bank provides two types of accounts:

1. **Savings Account** - Earns interest but has a withdrawal limit.
2. **Current Account** - Allows unlimited transactions but has a minimum balance requirement.

Your task is to **design a class hierarchy** using **Encapsulation and Inheritance** to model these accounts. Follow the requirements below:

**Core Requirements:**

* Implement an **abstract class** BankAccount with:
  + A **private balance** (should not be directly accessible outside the class).
  + Methods: deposit(double amount), withdraw(double amount), and checkBalance().
  + An **abstract method** calculateInterest() (only for SavingsAccount).
* Implement two subclasses:
  + **SavingsAccount**
    - Has an **interest rate (e.g., 4%)**.
    - Allows only **3 withdrawals per month** (if exceeded, display a warning).
    - Interest is added to the balance every month.
  + **CurrentAccount**
    - Requires a **minimum balance of $500**.
    - If balance falls below **$500**, impose a **penalty of $50**.
    - No interest is earned.
* Create a **Bank class** to simulate:
  + Account creation (createAccount()).
  + User transactions (deposit(), withdraw(), checkBalance()).
  + A method to simulate a **monthly cycle** where interest is applied and penalties are charged.

**Conditions:**

* **Prevent direct balance access** (force balance modification only through deposit/withdraw).
* If a withdrawal in **SavingsAccount** exceeds the limit, display **"Withdrawal limit exceeded!"**.
* If CurrentAccount balance falls below $500, charge $50 and notify the user.
* Ensure **zero-argument withdrawal is not allowed** (invalid transactions should be handled).
* Simulate an **N-month period** (where N is user-defined) and check if accounts behave correctly over time.

**Example Scenarios:**

1. **John opens a SavingsAccount** with $2000, withdraws **4 times**, and gets a warning.
2. **Lisa has a CurrentAccount** and her balance drops to $400 → She is penalized.
3. **A bank cycle runs for 6 months**, adding interest to SavingsAccount and charging penalties to CurrentAccount where applicable.

**Example 1: Opening a Savings Account and Depositing**

Welcome to the Banking System!

Select Account Type:

1. Savings Account

2. Current Account

>> 1

Enter initial deposit (minimum $1000):

>> 2000

**MENU:**

1. Deposit

2. Withdraw

3. Check Balance

4. Apply Interest

5. Exit

Enter your choice:

>> 1

Enter deposit amount:

>> 500

Deposit successful. New Balance: $2500

**Example 2: Withdrawing and Checking Balance in Current Account**

Welcome to the Banking System!

Select Account Type:

1. Savings Account

2. Current Account

>> 2

Enter initial deposit (minimum $500):

>> 600

**MENU:**

1. Deposit

2. Withdraw

3. Check Balance

5. Exit

Enter your choice:

>> 2

Enter withdrawal amount:

>> 200

Withdrawal successful. New Balance: $400

Minimum balance required. Penalty of $50 applied.

Final Balance: $350

Note :

**Encapsulation:** Balance is **private** and modified only via methods.  
**Inheritance:** SavingsAccount and CurrentAccount **extend** BankAccount.  
**Dynamic Method Calls:** Uses **abstract methods** for different withdrawal rules.  
**Validation:** Prevents **negative deposits, overdrafts, and invalid selections**.  
**User-Friendly:** Uses **loops & switch-case** for a **smooth experience**.