This question is asking you to design and implement a Java program using inheritance to represent a bank's account system with two specific types of accounts: Savings Account and Current Account. Here's a step-by-step explanation of the requirements:

---

1. Base Class Account

The Account class is the base (or parent) class that contains common properties and methods for all accounts:

Properties (Fields):

customerName: The name of the account holder.

accountNumber: The unique number for the account.

accountType: The type of account (Savings or Current).

balance: The current balance in the account.

Methods:

Deposit Money: Accept an amount from the customer, add it to the balance, and update it.

Display Balance: Show the current balance of the account.

Other methods like initialization can also be added here.

---

2. Derived Classes

Two classes will be derived from the Account class:

a. SavAcct (Savings Account Class):

Additional Features:

Calculate and deposit compound interest into the account.

Allow withdrawals but no cheque book facility.

Necessary Methods:

1. Compute and Deposit Interest: Calculate compound interest using a formula and add it to the balance.

2. Permit Withdrawal: Allow customers to withdraw money if sufficient balance is available and update the balance.

b. CurrAcct (Current Account Class):

Additional Features:

No interest is earned on the balance.

Allows cheque book facilities (not implemented here, but implied as a capability).

Requires maintaining a minimum balance; if the balance falls below this, impose a penalty (service charge).

Necessary Methods:

1. Check for Minimum Balance: Verify if the balance is below the specified minimum.

2. Impose Penalty: Deduct a service charge from the balance if the minimum balance is not maintained.

---

3. Key Points

1. Inheritance: Use inheritance to share common properties and methods from the Account class to the SavAcct and CurrAcct classes.

2. No Constructors: Initialize fields (class members) using regular methods instead of constructors.

3. Methods: Implement specific methods to perform the required tasks like deposit, withdrawal, balance display, etc.

---

4. Outline of the Code

Here is a high-level structure of the program:

*// Base class*

*class Account {*

*protected String customerName;*

*protected String accountNumber;*

*protected String accountType;*

*protected double balance;*

*// Method to initialize account details*

*public void initialize(String name, String accNum, String type, double initialBalance) {*

*customerName = name;*

*accountNumber = accNum;*

*accountType = type;*

*balance = initialBalance;*

*}*

*// Method to deposit money*

*public void deposit(double amount) {*

*balance += amount;*

*}*

*// Method to display balance*

*public void displayBalance() {*

*System.out.println("Account Balance: " + balance);*

*}*

*}*

*// Savings Account class*

*class SavAcct extends Account {*

*private double interestRate;*

*// Method to set interest rate*

*public void setInterestRate(double rate) {*

*interestRate = rate;*

*}*

*// Method to compute and deposit interest*

*public void computeAndDepositInterest(int years) {*

*double interest = balance \* Math.pow((1 + interestRate / 100), years) - balance;*

*balance += interest;*

*System.out.println("Interest added: " + interest);*

*}*

*// Method to withdraw money*

*public void withdraw(double amount) {*

*if (balance >= amount) {*

*balance -= amount;*

*System.out.println("Withdrawal successful. New balance: " + balance);*

*} else {*

*System.out.println("Insufficient balance.");*

*}*

*}*

*}*

*// Current Account class*

*class CurrAcct extends Account {*

*private double minimumBalance;*

*private double penalty;*

*// Method to set minimum balance and penalty*

*public void setMinimumBalance(double minBalance, double serviceCharge) {*

*minimumBalance = minBalance;*

*penalty = serviceCharge;*

*}*

*// Method to check for minimum balance and impose penalty*

*public void checkMinimumBalance() {*

*if (balance < minimumBalance) {*

*balance -= penalty;*

*System.out.println("Penalty imposed. New balance: " + balance);*

*}*

*}*

*// Method to withdraw money*

*public void withdraw(double amount) {*

*if (balance >= amount) {*

*balance -= amount;*

*System.out.println("Withdrawal successful. New balance: " + balance);*

*} else {*

*System.out.println("Insufficient balance.");*

*}*

*}*

*}*

*// Main class to test the program*

*public class BankAccount {*

*public static void main(String[] args) {*

*// Example of Savings Account*

*SavAcct savings = new SavAcct();*

*savings.initialize("John Doe", "SA123", "Savings", 1000.0);*

*savings.setInterestRate(5.0);*

*savings.deposit(500.0);*

*savings.computeAndDepositInterest(2);*

*savings.withdraw(300.0);*

*savings.displayBalance();*

*// Example of Current Account*

*CurrAcct current = new CurrAcct();*

*current.initialize("Jane Smith", "CA456", "Current", 2000.0);*

*current.setMinimumBalance(1500.0, 50.0);*

*current.withdraw(600.0);*

*current.checkMinimumBalance();*

*current.displayBalance();*

*}*

*}*

---

5. Explanation of Tasks

Accept Deposit: Implemented in the deposit method of the Account class.

Display Balance: Implemented in the displayBalance method of the Account class.

Compute Interest (Savings): computeAndDepositInterest in SavAcct calculates compound interest.

Permit Withdrawal: Both SavAcct and CurrAcct have withdraw methods.

Check Minimum Balance (Current): checkMinimumBalance in CurrAcct checks and imposes penalties if needed.

This design follows the principles of object-oriented programming (OOP) with a focus on inheritance and method reuse.