TheKiranAcademy

Problem Statement: Implementing Abstraction in Java

Create a Java program that demonstrates the concept of abstraction by modeling a simple banking system. The system should have a base abstract class `BankAccount` with two subclasses `SavingsAccount` and `CheckingAccount`. The abstract class should contain essential attributes and methods, while the subclasses should provide concrete implementations.

Requirements:

1. Create an abstract class `BankAccount` with the following attributes and methods:
   - Attributes:
     - `accountNumber` (String): To store the account number.
     - `accountHolderName` (String): To store the account holder's name.
     - `balance` (double): To store the account balance.

   - Methods:
     - `getAccountNumber()`: Abstract method to get the account number.
     - `getAccountHolderName()`: Abstract method to get the account holder's name.
     - `getBalance()`: Abstract method to get the account balance.
     - `deposit(double amount)`: Abstract method to deposit the given amount.
     - `withdraw(double amount)`: Abstract method to withdraw the given amount.

2. Create a concrete subclass `SavingsAccount` that extends `BankAccount` with the following additional features:
   - Implement the abstract methods from the base class.
   - Include an attribute `interestRate` (double) to store the interest rate.
   - Implement a constructor to initialize attributes.
   - Override the `withdraw` method to check for a minimum balance.

3. Create another concrete subclass `CheckingAccount` that extends `BankAccount` with the following additional features:

- Implement the abstract methods from the base class.
- Include an attribute `overdraftLimit` (double) to store the overdraft limit.
- Implement a constructor to initialize attributes.
- Override the `withdraw` method to consider the overdraft limit.

4. In the main program:

- Create instances of `SavingsAccount` and `CheckingAccount`.
- Demonstrate deposit and withdrawal operations on both account types.
- Display the account information including balance after each transaction.

Note:

- Use appropriate access modifiers and encapsulation principles.
- Ensure that you showcase abstraction by defining abstract methods in the base class and providing concrete implementations in the subclasses.
- Properly handle scenarios like insufficient balance during withdrawals.

This problem statement focuses on implementing abstraction using Java's abstract classes and methods. It aims to showcase how abstract classes can define a blueprint while leaving certain implementation details to concrete subclasses.