Java Overriding

Topics : JAVA
Written on April 10, 2023

Java method overriding is a feature that allows a subclass to provide its own implementation of a method that is already defined in its superclass. When a method is overridden, the subclass implementation of the method is called instead of the superclass implementation.

To override a method in Java, the subclass method must have the same method signature (name, parameters, and return type) as the superclass method.

Here's an example of method overriding in Java:

```
public class BankAccount {
    private String accountNumber;
    private double balance;

    public BankAccount(String accountNumber, double balance) {
        this.accountNumber = accountNumber;
        this.balance = balance;
    }

    public String getAccountNumber() {
        return accountNumber;
    }

    public double getBalance() {
        return balance;
    }

    public void deposit(double amount) {
        balance += amount;
        System.out.println("Deposit of " + amount + " successful.");
    }

    public void withdraw(double amount) {
        if (balance >= amount) {
            balance -= amount;
            System.out.println("Withdrawal of " + amount + " successful.");
        } else {
            System.out.println("Insufficient balance.");
        }
    }

    public void displayBalance() {
        System.out.println("Account balance: " + balance);
    }
}

public class SavingsAccount extends BankAccount {
    private double interestRate;

    public SavingsAccount(String accountNumber, double balance, double
interestRate) {
        super(accountNumber, balance);
        this.interestRate = interestRate;
    }

    @Override
    public void displayBalance() {
        double balanceWithInterest = getBalance() + (getBalance() *
interestRate / 100);
        System.out.println("Savings account balance: " +
balanceWithInterest);
    }
}

public class CheckingAccount extends BankAccount {
    private double overdraftLimit;

    public CheckingAccount(String accountNumber, double balance, double
overdraftLimit) {
        super(accountNumber, balance);
        this.overdraftLimit = overdraftLimit;
    }

    @Override
    public void withdraw(double amount) {
        if (balance + overdraftLimit >= amount) {
            balance -= amount;
            System.out.println("Withdrawal of " + amount + " successful.");
        } else {
            System.out.println("Insufficient balance and overdraft limit.");
        }
    }
}

public class Main {
    public static void main(String[] args) {
        BankAccount account1 = new BankAccount("001", 1000);
        account1.displayBalance(); // Output: "Account balance: 1000.0"
        SavingsAccount account2 = new SavingsAccount("002", 2000, 2.5);
        account2.displayBalance(); // Output: "Savings account balance: 
2050.0"

        CheckingAccount account3 = new CheckingAccount("003", 3000, 500);
        account3.withdraw(3500); // Output: "Withdrawal of 3500.0 
successful."
        account3.displayBalance(); // Output: "Account balance: -500.0"
    }
}
```

In this example, we have a `BankAccount` class that represents a simple bank account with an account number and balance. It has methods for depositing, withdrawing, and displaying the balance.

The `SavingsAccount` class extends the `BankAccount` class and overrides the `displayBalance()` method to display the account balance with the added interest rate.

The `CheckingAccount` class also extends the `BankAccount` class and overrides the `withdraw()` method to allow overdrafts up to the specified limit.

In the `main()` method, we create three different types of accounts and demonstrate their unique features through method overriding. We create a `BankAccount`, a `SavingsAccount`, and a `CheckingAccount`, and call their methods accordingly. The output shows how the methods are overridden and how the subclasses provide their own implementation of the methods.