Week 8 - S8 - Core OOP - Abstract Class And Interface - Practice Problem

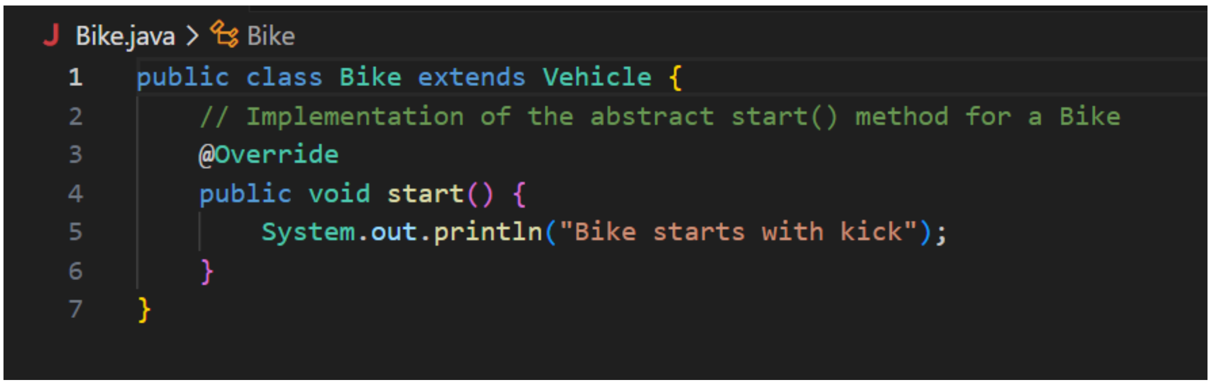
Name: Ramesh Harisabapathi Chettiar

Date of Submission:06/10/2025

QNO1→

Create an abstract class Vehicle with an abstract method start(). Subclasses Car and Bike will extend Vehicle and provide their own implementations for start(). Demonstrate abstraction by using Vehicle references to call the methods.

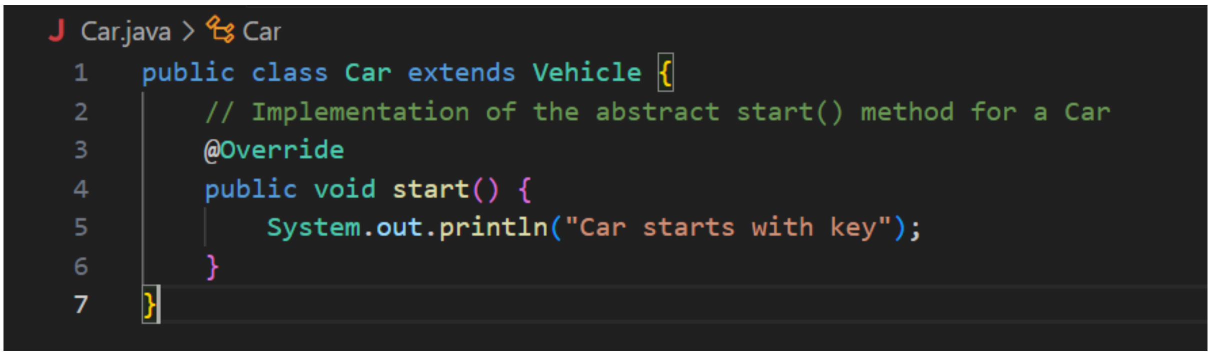
Bike.java



Bike.java > Bike

```java
public class Bike extends Vehicle {
    // Implementation of the abstract start() method for a Bike
    @Override
    public void start() {
        System.out.println("Bike starts with kick");
    }
}
```

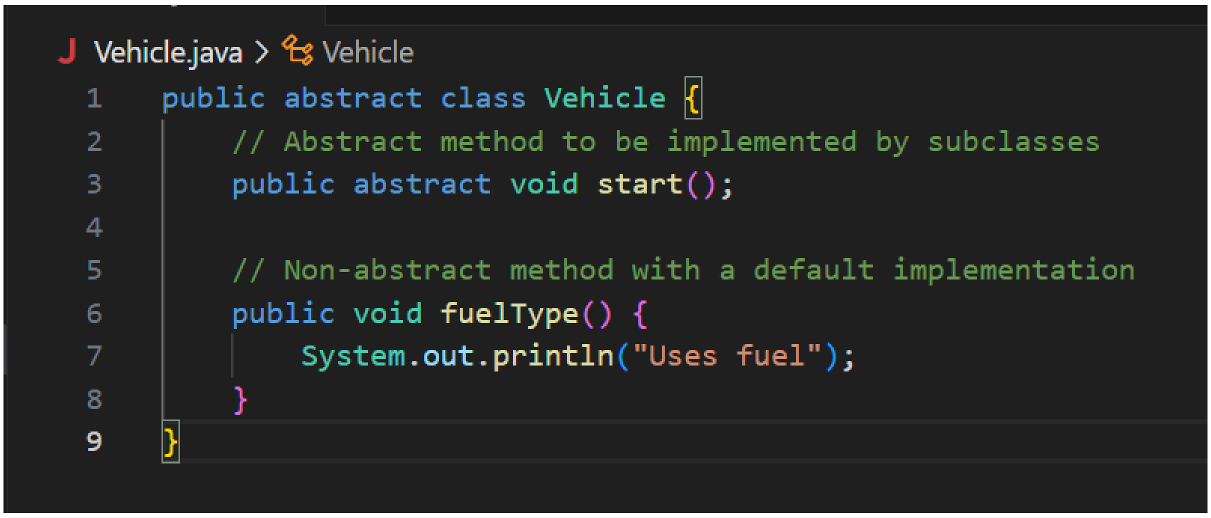
Car.java



Car.java > Car

```java
public class Car extends Vehicle {
    // Implementation of the abstract start() method for a Car
    @Override
    public void start() {
        System.out.println("Car starts with key");
    }
}
```

Vehicle.java

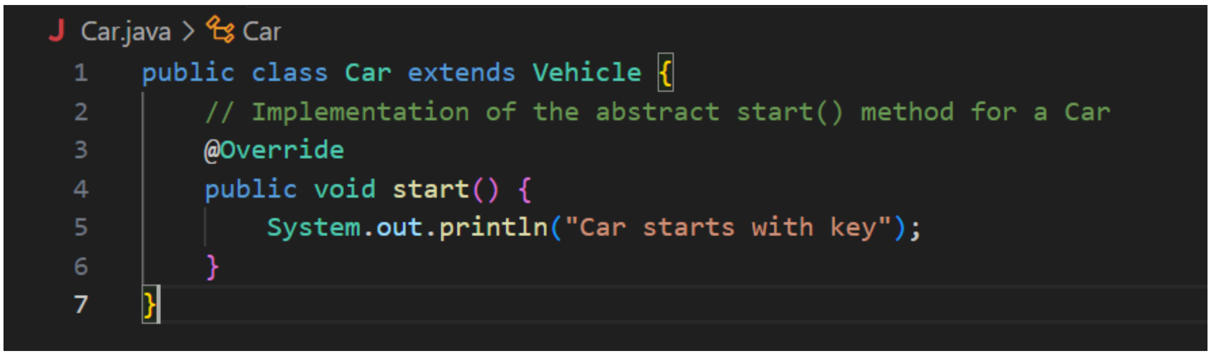


Vehicle.java > Vehicle

```java
public abstract class Vehicle {
    // Abstract method to be implemented by subclasses
    public abstract void start();

    // Non-abstract method with a default implementation
    public void fuelType() {
        System.out.println("Uses fuel");
    }
}
```

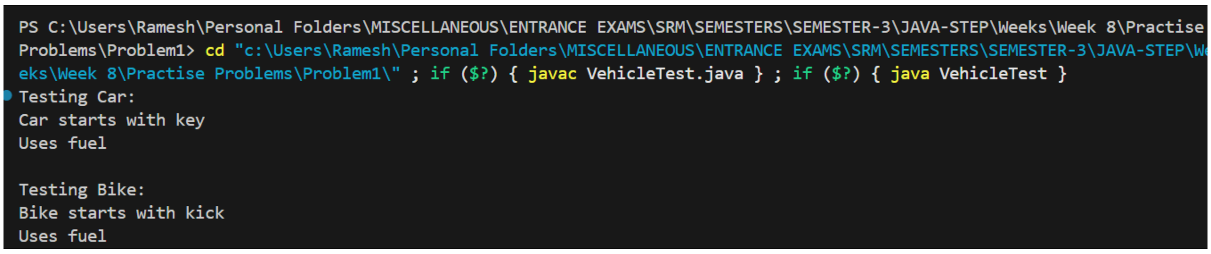
VehicleTest.java



Car.java > Car

```java
public class Car extends Vehicle {
    // Implementation of the abstract start() method for a Car
    @Override
    public void start() {
        System.out.println("Car starts with key");
    }
}
```

OUTPUT→



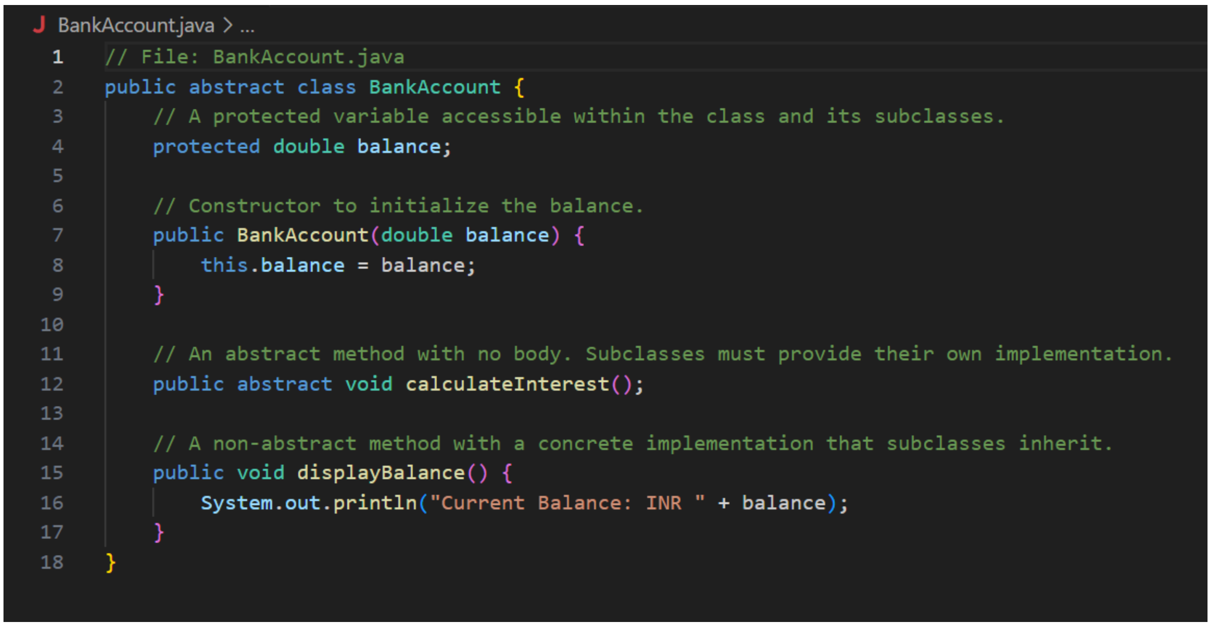
```
PS C:\Users\Ramesh\Personal Folders\MISCELLANEOUS\ENTRANCE EXAMS\SRM\SEMESTERS\SEMESTER-3\JAVA-STEP\Weeks\Week 8\Practise
Problems\Problem1> cd "c:\Users\Ramesh\Personal Folders\MISCELLANEOUS\ENTRANCE EXAMS\SRM\SEMESTERS\SEMESTER-3\JAVA-STEP\We
eks\Week 8\Practise Problems\Problem1\" ; if ($?) { javac VehicleTest.java } ; if ($?) { java VehicleTest }
Testing Car:
Car starts with key
Uses fuel

Testing Bike:
Bike starts with kick
Uses fuel
```

QNO2→

Design an abstract class BankAccount with abstract method calculateInterest(). Subclasses SavingsAccount and CurrentAccount should implement it differently. Demonstrate abstraction by handling different account types.

BankAccount.java



BankAccount.java > ...

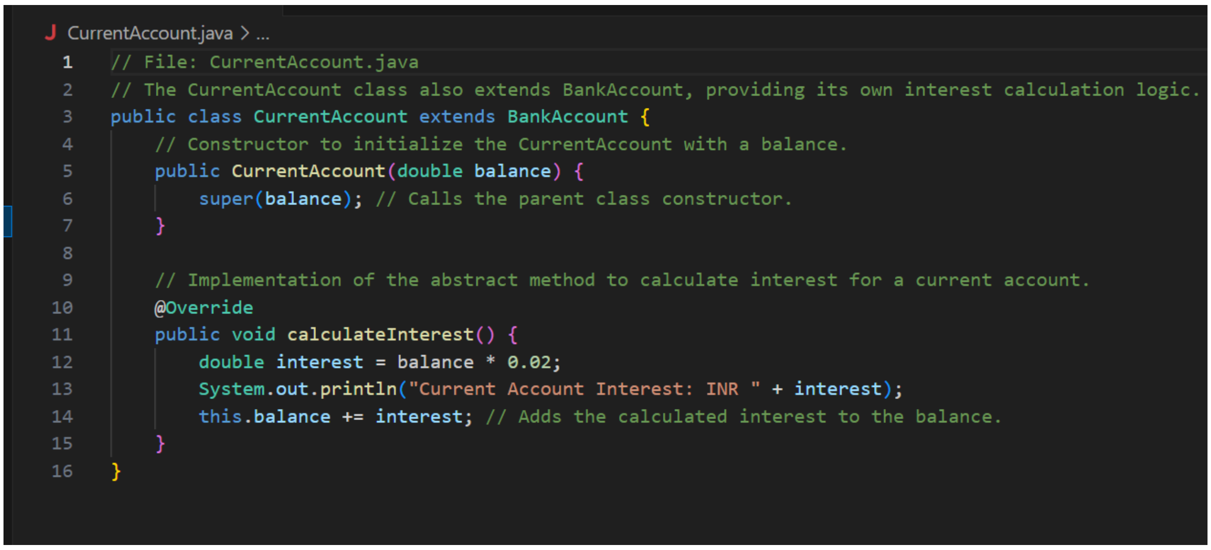
```java
// File: BankAccount.java
public abstract class BankAccount {
    // A protected variable accessible within the class and its subclasses.
    protected double balance;

    // Constructor to initialize the balance.
    public BankAccount(double balance) {
        this.balance = balance;
    }

    // An abstract method with no body. Subclasses must provide their own implementation.
    public abstract void calculateInterest();

    // A non-abstract method with a concrete implementation that subclasses inherit.
    public void displayBalance() {
        System.out.println("Current Balance: INR " + balance);
    }
}
```

CurrentAccount.java

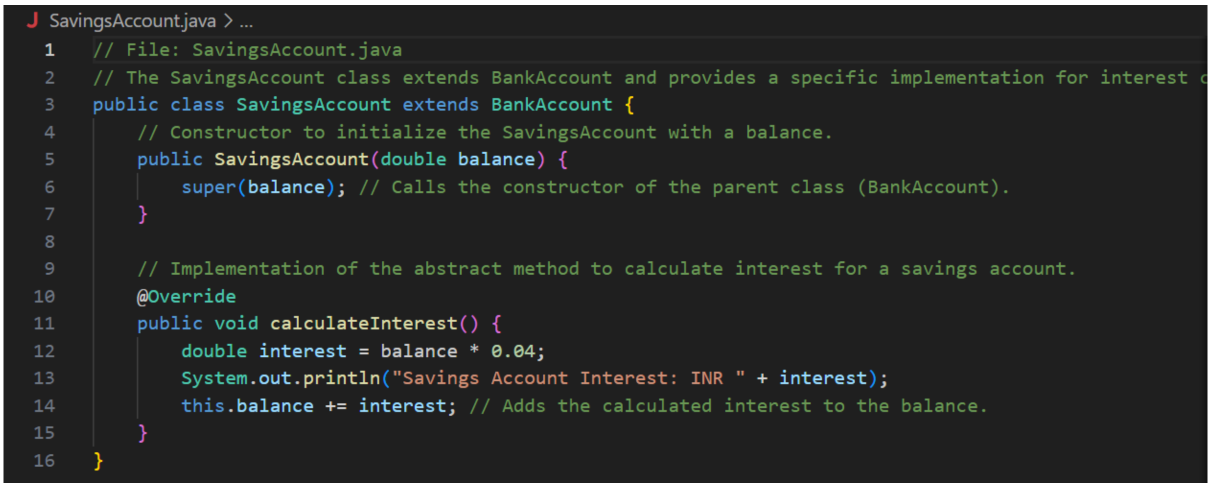


CurrentAccount.java > ...

```java
// File: CurrentAccount.java
// The CurrentAccount class also extends BankAccount, providing its own interest calculation logic.
public class CurrentAccount extends BankAccount {
    // Constructor to initialize the CurrentAccount with a balance.
    public CurrentAccount(double balance) {
        super(balance); // Calls the parent class constructor.
    }

    // Implementation of the abstract method to calculate interest for a current account.
    @Override
    public void calculateInterest() {
        double interest = balance * 0.02;
        System.out.println("Current Account Interest: INR " + interest);
        this.balance += interest; // Adds the calculated interest to the balance.
    }
}
```

SavingsAccount.java

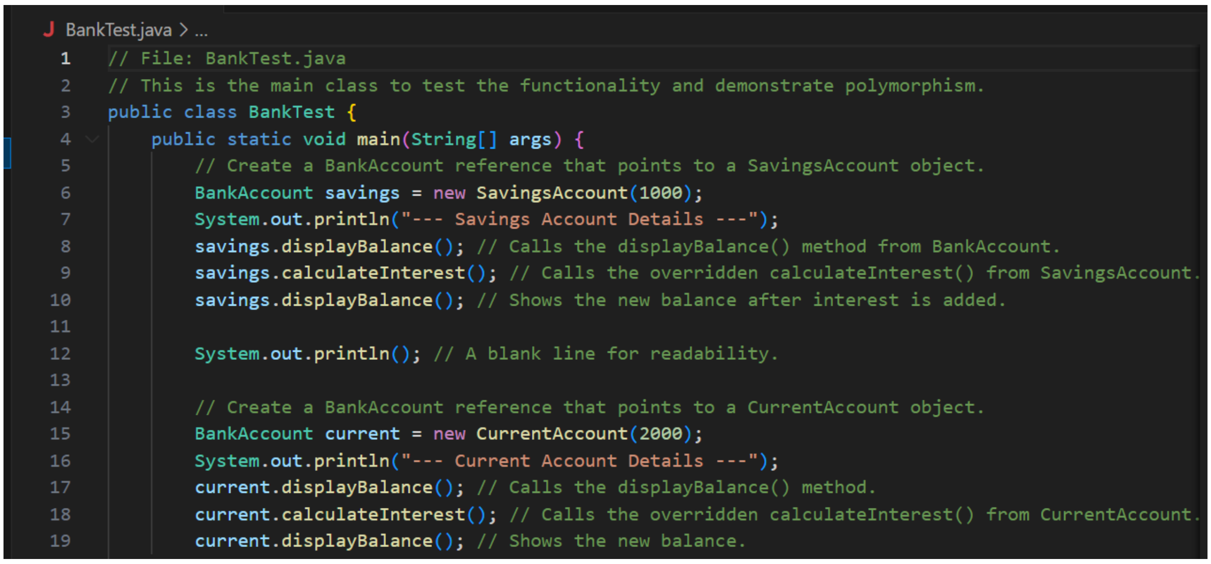


SavingsAccount.java > ...

```java
// File: SavingsAccount.java
// The SavingsAccount class extends BankAccount and provides a specific implementation for interest
public class SavingsAccount extends BankAccount {
    // Constructor to initialize the SavingsAccount with a balance.
    public SavingsAccount(double balance) {
        super(balance); // Calls the constructor of the parent class (BankAccount).
    }

    // Implementation of the abstract method to calculate interest for a savings account.
    @Override
    public void calculateInterest() {
        double interest = balance * 0.04;
        System.out.println("Savings Account Interest: INR " + interest);
        this.balance += interest; // Adds the calculated interest to the balance.
    }
}
```

BankTest.java



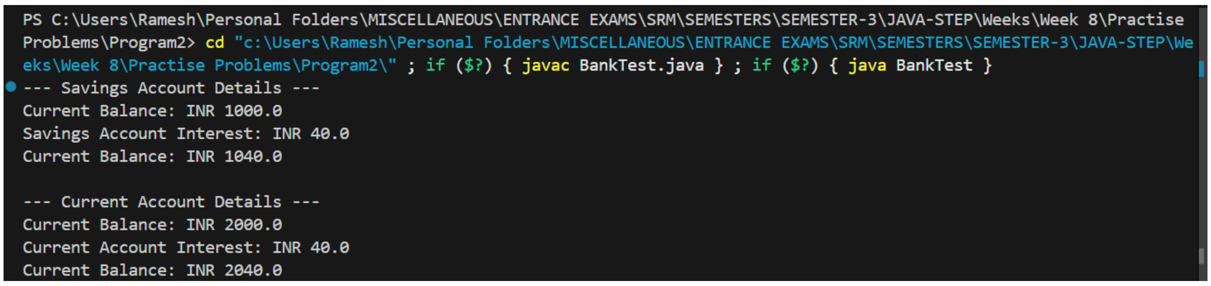
BankTest.java > ...

```java
// File: BankTest.java
// This is the main class to test the functionality and demonstrate polymorphism.
public class BankTest {
    public static void main(String[] args) {
        // Create a BankAccount reference that points to a SavingsAccount object.
        BankAccount savings = new SavingsAccount(1000);
        System.out.println("--- Savings Account Details ---");
        savings.displayBalance(); // Calls the displayBalance() method from BankAccount.
        savings.calculateInterest(); // Calls the overridden calculateInterest() from SavingsAccount.
        savings.displayBalance(); // Shows the new balance after interest is added.

        System.out.println(); // A blank line for readability.

        // Create a BankAccount reference that points to a CurrentAccount object.
        BankAccount current = new CurrentAccount(2000);
        System.out.println("--- Current Account Details ---");
        current.displayBalance(); // Calls the displayBalance() method.
        current.calculateInterest(); // Calls the overridden calculateInterest() from CurrentAccount.
        current.displayBalance(); // Shows the new balance.
```

OUTPUT→



```
PS C:\Users\Ramesh\Personal Folders\MISCELLANEOUS\ENTRANCE EXAMS\SRM\SEMESTERS\SEMESTER-3\JAVA-STEP\Weeks\Week 8\Practise
Problems\Program2> cd "c:\Users\Ramesh\Personal Folders\MISCELLANEOUS\ENTRANCE EXAMS\SRM\SEMESTERS\SEMESTER-3\JAVA-STEP\We
eks\Week 8\Practise Problems\Program2\" ; if ($?) { javac BankTest.java } ; if ($?) { java BankTest }
--- Savings Account Details ---
Current Balance: INR 1000.0
Savings Account Interest: INR 40.0
Current Balance: INR 1040.0

--- Current Account Details ---
Current Balance: INR 2000.0
Current Account Interest: INR 40.0
Current Balance: INR 2040.0
```

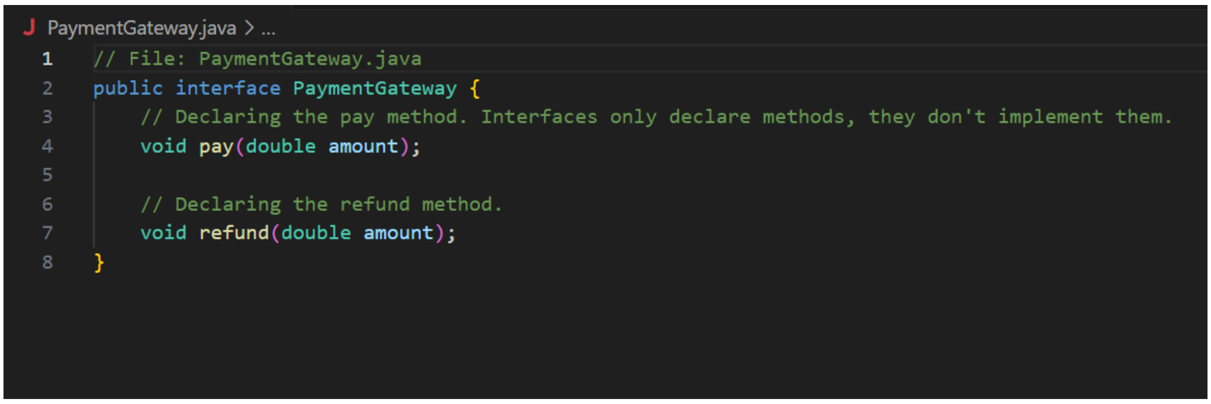
QNO3→

Create an interface PaymentGateway with methods pay() and refund().

Implement this interface in CreditCardPayment and UPIPayment. Demonstrate

multiple payment methods using interfaces.

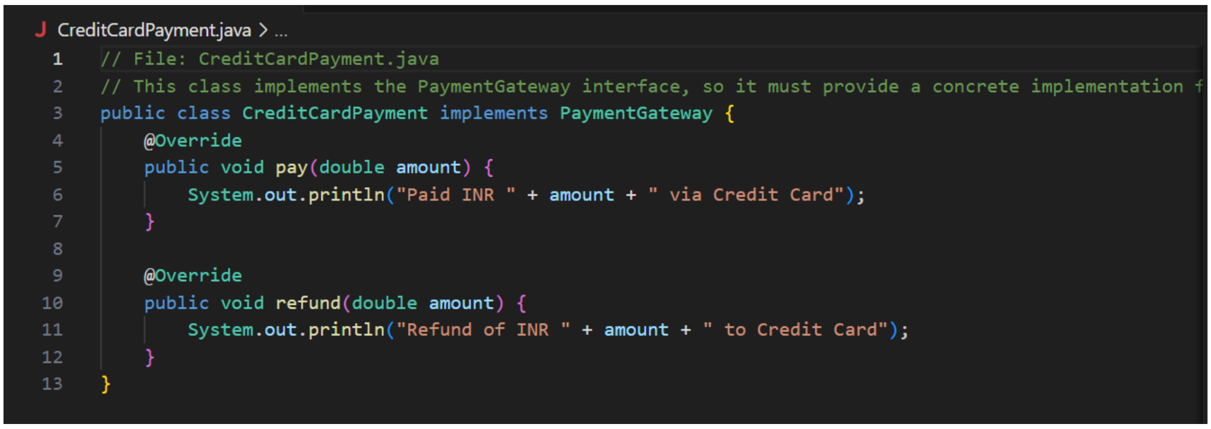
PaymentGateway.java



```java
// File: PaymentGateway.java
public interface PaymentGateway {
    // Declaring the pay method. Interfaces only declare methods, they don't implement them.
    void pay(double amount);

    // Declaring the refund method.
    void refund(double amount);
}
```

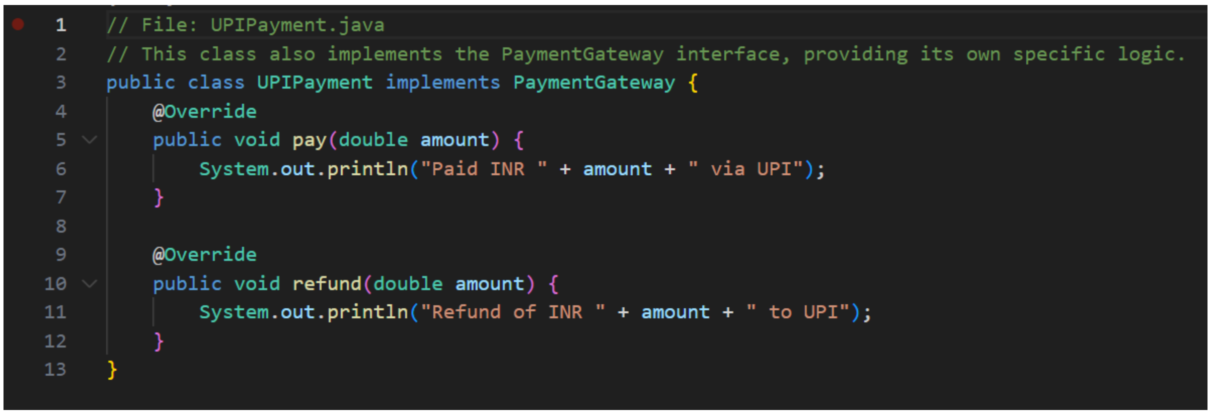
CreditCardPayment.java



```java
// File: CreditCardPayment.java
// This class implements the PaymentGateway interface, so it must provide a concrete implementation f
public class CreditCardPayment implements PaymentGateway {
    @Override
    public void pay(double amount) {
        System.out.println("Paid INR " + amount + " via Credit Card");
    }

    @Override
    public void refund(double amount) {
        System.out.println("Refund of INR " + amount + " to Credit Card");
    }
}
```

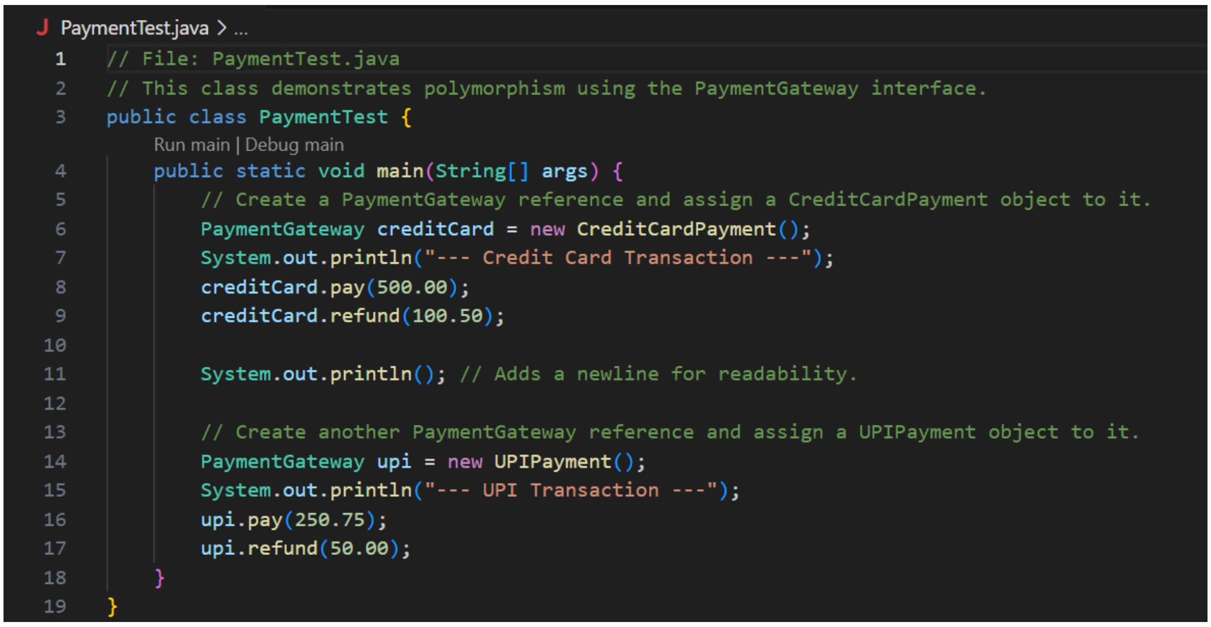
UPIPayment.java



```java
// File: UPIPayment.java
// This class also implements the PaymentGateway interface, providing its own specific logic.
public class UPIPayment implements PaymentGateway {
    @Override
    public void pay(double amount) {
        System.out.println("Paid INR " + amount + " via UPI");
    }

    @Override
    public void refund(double amount) {
        System.out.println("Refund of INR " + amount + " to UPI");
    }
}
```

PaymentTest.java



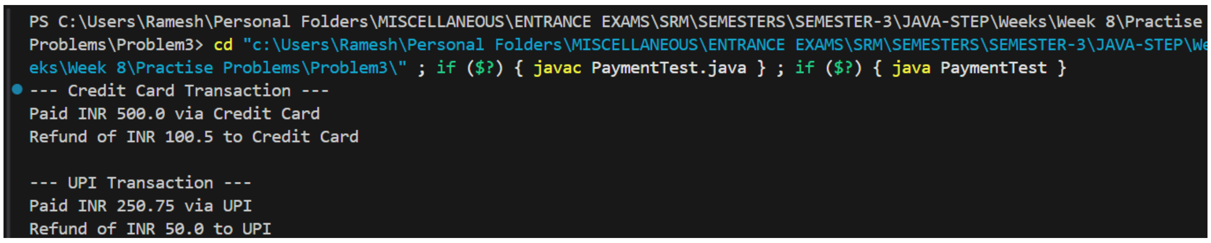
PaymentTest.java > ...

```java
// File: PaymentTest.java
// This class demonstrates polymorphism using the PaymentGateway interface.
public class PaymentTest {
    Run main | Debug main
    public static void main(String[] args) {
        // Create a PaymentGateway reference and assign a CreditCardPayment object to it.
        PaymentGateway creditCard = new CreditCardPayment();
        System.out.println("--- Credit Card Transaction ---");
        creditCard.pay(500.00);
        creditCard.refund(100.50);

        System.out.println(); // Adds a newline for readability.

        // Create another PaymentGateway reference and assign a UPIPayment object to it.
        PaymentGateway upi = new UPIPayment();
        System.out.println("--- UPI Transaction ---");
        upi.pay(250.75);
        upi.refund(50.00);
    }
}
```

OUTPUT→



```
PS C:\Users\Ramesh\Personal Folders\MISCELLANEOUS\ENTRANCE EXAMS\SRM\SEMESTERS\SEMESTER-3\JAVA-STEP\Weeks\Week 8\Practise
Problems\Problem3> cd "c:\Users\Ramesh\Personal Folders\MISCELLANEOUS\ENTRANCE EXAMS\SRM\SEMESTERS\SEMESTER-3\JAVA-STEP\We
eks\Week 8\Practise Problems\Problem3\" ; if ($?) { javac PaymentTest.java } ; if ($?) { java PaymentTest }
--- Credit Card Transaction ---
Paid INR 500.0 via Credit Card
Refund of INR 100.5 to Credit Card

--- UPI Transaction ---
Paid INR 250.75 via UPI
Refund of INR 50.0 to UPI
```