**Design Pattern And Principles**

**EXERCISE 8: Implementing the Strategy Pattern**

**Source Code**

// Main.java

// Step 2: Strategy Interface

interface PaymentStrategy {

void pay(double amount);

}

// Step 3: Concrete Strategies

class CreditCardPayment implements PaymentStrategy {

private String cardNumber;

public CreditCardPayment(String cardNumber) {

this.cardNumber = cardNumber;

}

public void pay(double amount) {

System.out.println("Paid ₹" + amount + " using Credit Card (\*\*\*\*" + cardNumber.substring(cardNumber.length() - 4) + ")");

}

}

class PayPalPayment implements PaymentStrategy {

private String email;

public PayPalPayment(String email) {

this.email = email;

}

public void pay(double amount) {

System.out.println("Paid ₹" + amount + " using PayPal (Account: " + email + ")");

}

}

// Step 4: Context Class

class PaymentContext {

private PaymentStrategy strategy;

public void setPaymentStrategy(PaymentStrategy strategy) {

this.strategy = strategy;

}

public void processPayment(double amount) {

if (strategy == null) {

System.out.println("No payment method selected.");

} else {

strategy.pay(amount);

}

}

}

// Step 5: Test Class

public class Main {

public static void main(String[] args) {

PaymentContext context = new PaymentContext();

// Pay using Credit Card

context.setPaymentStrategy(new CreditCardPayment("1234567890128756"));

context.processPayment(5980.00);

// Pay using PayPal

context.setPaymentStrategy(new PayPalPayment("user@example.com"));

context.processPayment(2090.75);

}

}

**OUTPUT**

