**Exercise 4: Implementing the Adapter Pattern**

**Scenario:**

You are developing a payment processing system that needs to integrate with multiple third-party payment gateways with different interfaces. Use the Adapter Pattern to achieve this.

**Steps:**

**1. Create a New Java Project**:

* Create a new Java project named **AdapterPatternExample**.

**2. Define Target Interface**:

* Create an interface PaymentProcessor with methods like processPayment().

**3. Implement Adaptee Classes**:

* Create classes for different payment gateways with their own methods.

**4. Implement the Adapter Class**:

* Create an adapter class for each payment gateway that implements PaymentProcessor and translates the calls to the gateway-specific methods.

**5. Test the Adapter Implementation**:

* Create a test class to demonstrate the use of different payment gateways through the adapter.

**Code**:

package AdapterPatternExample;

// Step 2: Define Target Interface

interface PaymentProcessor {

void processPayment(double amount);

}

// Step 3: Implement Adapter Classes

// Paypal.java

class Paypal {

public void sendPayment(double amount) {

System.out.println("Paypal: Sending payment of Rs." + amount);

}

}

// Stripe.java

class Stripe {

public void makePayment(double amount) {

System.out.println("Stripe: Making payment of Rs." + amount);

}

}

// Square.java

class Square {

public void executePayment(double amount) {

System.out.println("Square: Making payment of Rs." + amount);

}

}

// Step 4: Implement the Adapter Classes

// PaypalAdapter.java

class PaypalAdapter implements PaymentProcessor {

private Paypal paypal;

public PaypalAdapter(Paypal paypal) {

this.paypal = paypal;

}

@Override

public void processPayment(double amount) {

paypal.sendPayment(amount);

}

}

// StripeAdapter.java

class StripeAdapter implements PaymentProcessor {

private Stripe stripe;

public StripeAdapter(Stripe stripe) {

this.stripe = stripe;

}

@Override

public void processPayment(double amount) {

stripe.makePayment(amount);

}

}

// SquareAdapter.java

class SquareAdapter implements PaymentProcessor {

private Square square;

public SquareAdapter(Square square) {

this.square = square;

}

@Override

public void processPayment(double amount) {

square.executePayment(amount);

}

}

// Step 5: Create AdapterPatternExample Class

// AdapterPatternExample.java

public class AdapterPatternExample {

public static void main(String[] args) {

AdapterPatternExample example = new AdapterPatternExample();

example.run();

}

public void run() {

Paypal paypal = new Paypal();

PaymentProcessor paypalProcessor = new PaypalAdapter(paypal);

processPayment(paypalProcessor, 150.00);

Stripe stripe = new Stripe();

PaymentProcessor stripeProcessor = new StripeAdapter(stripe);

processPayment(stripeProcessor, 250.00);

Square square = new Square();

PaymentProcessor squareProcessor = new SquareAdapter(square);

processPayment(squareProcessor, 350.00);

}

private void processPayment(PaymentProcessor processor, double amount) {

processor.processPayment(amount);

}

}