**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:**

* **Project Name**: AdapterPatternExample
* Use your preferred IDE to create a new Java project.

**2. Define Target Interface:**

**PaymentProcessor Interface**:

* Create an interface PaymentProcessor with a method processPayment().

public interface PaymentProcessor {

void processPayment(double amount);

}

**3. Implement Adaptee Classes:**

**Adaptee Classes**:

* Create classes for different payment gateways, each with its own method(s) to process payments.

// Adaptee class for PayPal

public class PayPalGateway {

public void makePayment(double amount) {

System.out.println("Processing payment of $" + amount + " through PayPal.");

}

}

// Adaptee class for Stripe

public class StripeGateway {

public void charge(double amount) {

System.out.println("Charging $" + amount + " through Stripe.");

}

}

// Adaptee class for Square

public class SquareGateway {

public void pay(double amount) {

System.out.println("Paying $" + amount + " through Square.");

}

}

**4. Implement the Adapter Class:**

**Adapter Classes**:

* Create adapter classes for each payment gateway that implement PaymentProcessor and translate the calls to the gateway-specific methods.

// Adapter class for PayPal

public class PayPalAdapter implements PaymentProcessor {

private PayPalGateway payPalGateway;

public PayPalAdapter(PayPalGateway payPalGateway) {

this.payPalGateway = payPalGateway;

}

@Override

public void processPayment(double amount) {

payPalGateway.makePayment(amount);

}

}

// Adapter class for Stripe

public class StripeAdapter implements PaymentProcessor {

private StripeGateway stripeGateway;

public StripeAdapter(StripeGateway stripeGateway) {

this.stripeGateway = stripeGateway;

}

@Override

public void processPayment(double amount) {

stripeGateway.charge(amount);

}

}

// Adapter class for Square

public class SquareAdapter implements PaymentProcessor {

private SquareGateway squareGateway;

public SquareAdapter(SquareGateway squareGateway) {

this.squareGateway = squareGateway;

}

@Override

public void processPayment(double amount) {

squareGateway.pay(amount);

}

**5. Test the Adapter Implementation:**

**TestAdapterPattern Class**:

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

public class TestAdapterPattern {

public static void main(String[] args) {

// Using PayPal through the adapter

PayPalGateway payPal = new PayPalGateway();

PaymentProcessor payPalProcessor = new PayPalAdapter(payPal);

payPalProcessor.processPayment(150.00);

// Using Stripe through the adapter

StripeGateway stripe = new StripeGateway();

PaymentProcessor stripeProcessor = new StripeAdapter(stripe);

stripeProcessor.processPayment(200.00);

// Using Square through the adapter

SquareGateway square = new SquareGateway();

PaymentProcessor squareProcessor = new SquareAdapter(square);

squareProcessor.processPayment(250.00);

}

}