**Design Pattern And Principles**

**EXERCISE 4: Implementing the Adapter Pattern**

**Source Code**

// Main.java

// Step 2: Target Interface

interface PaymentProcessor {

void processPayment(double amount);

}

// Step 3: Adaptee Classes (Third-party gateways)

class PayPalGateway {

public void sendPayment(double amount) {

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

}

}

class StripeGateway {

public void makePayment(double amount) {

System.out.println("Paid ₹" + amount + " using Stripe.");

}

}

class RazorpayGateway {

public void executeTransaction(double amount) {

System.out.println("Paid ₹" + amount + " using Razorpay.");

}

}

// Step 4: Adapter Classes

class PayPalAdapter implements PaymentProcessor {

private PayPalGateway payPal;

public PayPalAdapter(PayPalGateway payPal) {

this.payPal = payPal;

}

public void processPayment(double amount) {

payPal.sendPayment(amount);

}

}

class StripeAdapter implements PaymentProcessor {

private StripeGateway stripe;

public StripeAdapter(StripeGateway stripe) {

this.stripe = stripe;

}

public void processPayment(double amount) {

stripe.makePayment(amount);

}

}

class RazorpayAdapter implements PaymentProcessor {

private RazorpayGateway razorpay;

public RazorpayAdapter(RazorpayGateway razorpay) {

this.razorpay = razorpay;

}

public void processPayment(double amount) {

razorpay.executeTransaction(amount);

}

}

// Step 5: Test Class

public class Main {

public static void main(String[] args) {

// Using PayPal

PaymentProcessor paypalProcessor = new PayPalAdapter(new PayPalGateway());

paypalProcessor.processPayment(3932.00);

// Using Stripe

PaymentProcessor stripeProcessor = new StripeAdapter(new StripeGateway());

stripeProcessor.processPayment(5000.50);

// Using Razorpay

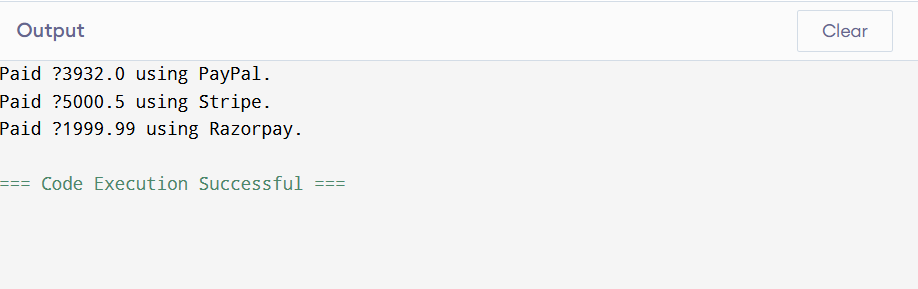
PaymentProcessor razorpayProcessor = new RazorpayAdapter(new RazorpayGateway());

razorpayProcessor.processPayment(1999.99);

}

}

**OUTPUT**

****