## Exercise 4: Implementing the Adapter Pattern

Scenario:  
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.

## Step-by-step Implementation:

### Step 1: Create a New Java Project

Create a new Java project named AdapterPatternExample.

### Step 2: Define Target Interface

public interface PaymentProcessor {  
 void processPayment(double amount);  
}

### Step 3: Implement Adaptee Classes

public class PayPalGateway {  
 public void makePayment(double amount) {  
 System.out.println("Paid " + amount + " using PayPal.");  
 }  
}  
  
public class StripeGateway {  
 public void doPayment(double amount) {  
 System.out.println("Paid " + amount + " using Stripe.");  
 }  
}

### Step 4: Implement the Adapter Class

public class PayPalAdapter implements PaymentProcessor {  
 private PayPalGateway paypal;  
  
 public PayPalAdapter(PayPalGateway paypal) {  
 this.paypal = paypal;  
 }  
  
 @Override  
 public void processPayment(double amount) {  
 paypal.makePayment(amount);  
 }  
}  
  
public class StripeAdapter implements PaymentProcessor {  
 private StripeGateway stripe;  
  
 public StripeAdapter(StripeGateway stripe) {  
 this.stripe = stripe;  
 }  
  
 @Override  
 public void processPayment(double amount) {  
 stripe.doPayment(amount);  
 }  
}

### Step 5: Test the Adapter Implementation

public class TestAdapterPattern {  
 public static void main(String[] args) {  
 PaymentProcessor paypalProcessor = new PayPalAdapter(new PayPalGateway());  
 paypalProcessor.processPayment(100.00);  
  
 PaymentProcessor stripeProcessor = new StripeAdapter(new StripeGateway());  
 stripeProcessor.processPayment(200.00);  
 }  
}  
OUTPUT:

