# Exercise 4: Implementing the Adapter Pattern

PaymentProcessor.java:

package com.example;

public interface PaymentProcessor {

void processPayment(double amount);

boolean verifyPayment(String transactionId);

void refundPayment(String transactionId, double amount);

}

PayPalGateway.java:

package com.example;

public class PayPalGateway {

public void sendPayment(double amount) {

System.***out***.println("Processing PayPal payment of Rs" + amount);

}

public boolean checkTransaction(String paypalTransactionId) {

System.***out***.println("Verifying PayPal transaction: " + paypalTransactionId);

return true;

}

public void issueRefund(String paypalTransactionId, double amount) {

System.***out***.println("Issuing PayPal refund of Rs" + amount + " for transaction: " + paypalTransactionId);

}

}

StripeGateway.java:

package com.example;

public class StripeGateway {

public void chargeCard(double amountInCents) {

System.***out***.println("Processing Stripe payment of Rs" + (amountInCents/100));

}

public boolean verifyCharge(String stripeChargeId) {

System.***out***.println("Verifying Stripe charge: " + stripeChargeId);

return true;

}

public void createRefund(String stripeChargeId, int amountInCents) {

System.***out***.println("Issuing Stripe refund of Rs" + (amountInCents/100) + " for charge: " + stripeChargeId);

}

}

PaypalAdapter.java

package com.example;

public class PayPalAdapter implements PaymentProcessor {

private final PayPalGateway payPalGateway;

public PayPalAdapter(PayPalGateway payPalGateway) {

this.payPalGateway = payPalGateway;

}

*@Override*

public void processPayment(double amount) {

payPalGateway.sendPayment(amount);

}

*@Override*

public boolean verifyPayment(String transactionId) {

return payPalGateway.checkTransaction(transactionId);

}

*@Override*

public void refundPayment(String transactionId, double amount) {

payPalGateway.issueRefund(transactionId, amount);

}

}

StripeAdapter.java:

package com.example;

public class StripeAdapter implements PaymentProcessor {

private final StripeGateway stripeGateway;

public StripeAdapter(StripeGateway stripeGateway) {

this.stripeGateway = stripeGateway;

}

*@Override*

public void processPayment(double amount) {

stripeGateway.chargeCard(amount \* 100);

}

*@Override*

public boolean verifyPayment(String transactionId) {

return stripeGateway.verifyCharge(transactionId);

}

*@Override*

public void refundPayment(String transactionId, double amount) {

stripeGateway.createRefund(transactionId, (int)(amount \* 100));

}

}

PaymentSystemTest.java:

package com.example;

public class PaymentSystemTest {

public static void main(String[] args) {

PayPalGateway payPalGateway = new PayPalGateway();

StripeGateway stripeGateway = new StripeGateway();

// Create adapters for each gateway

PaymentProcessor payPalProcessor = new PayPalAdapter(payPalGateway);

PaymentProcessor stripeProcessor = new StripeAdapter(stripeGateway);

// Test PayPal through adapter

System.***out***.println("Processing PayPal payment:");

payPalProcessor.processPayment(100.50);

payPalProcessor.verifyPayment("PAYPAL123");

payPalProcessor.refundPayment("PAYPAL123", 50.25);

System.***out***.println("\nProcessing Stripe payment:");

// Test Stripe through adapter

stripeProcessor.processPayment(75.99);

stripeProcessor.verifyPayment("STRIPE456");

stripeProcessor.refundPayment("STRIPE456", 75.99);

}

}

Output:

