**Exercise 4: Implementing the Adapter Pattern**

**Code:**

**PaymentProcessor Interface: -**

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

**Pay Class: -**

public class Pay {  
 private PaymentProcessor paymentProcessor;  
 public PaymentProcessor getPaymentProcessor(){  
 return paymentProcessor;  
 }  
 public void set(PaymentProcessor paymentProcessor){  
 this.paymentProcessor=paymentProcessor;  
 }  
 public void payMoney(int amount){  
 paymentProcessor.processPayment(amount);  
 }  
  
}

**Paytm Class: -**

public class Paytm {  
 public void payThroughPaytm(int amount){  
 System.*out*.println(amount+" rupees payed through Paytm");  
 }  
}

**Razorpay Class: -**

public class Razorpay {  
 public void payThroughRazorpay(int amount) {  
 System.*out*.println(amount+" rupees payed through Razorpay");  
 }  
}

**PaytmAdapter Class: -**

public class PaytmAdapter implements PaymentProcessor {  
 Paytm paytm=new Paytm();  
 @Override  
 public void processPayment(int amount) {  
 paytm.payThroughPaytm(amount);  
 }  
}

**RazorpayAdapter Class: -**

public class RazorpayAdapter implements PaymentProcessor{  
 Razorpay razorpay=new Razorpay();  
 @Override  
 public void processPayment(int amount) {  
 razorpay.payThroughRazorpay(amount);  
 }  
}

**TestClass (Main Class): -**

public class TestClass {  
 public static void main(String[] args){  
 RazorpayAdapter razorpayAdapter=new RazorpayAdapter();  
 Pay pay=new Pay();  
 pay.set(razorpayAdapter);  
 pay.payMoney(20000);  
  
 PaytmAdapter paytmAdapter=new PaytmAdapter();  
 pay.set(paytmAdapter);  
 pay.payMoney(10000);  
 }  
}

**Output:**

**A screenshot of a computer program

AI-generated content may be incorrect.**