**Exercise 8: Implementing the Strategy Pattern**

**Scenario:**

You are developing a payment system where different payment methods (e.g., Credit Card, PayPal) can be selected at runtime. Use the Strategy Pattern to achieve this.

1. Create a new Java Project:

* Create a new Java Project named StrategyPatternExample.
* A Java project named StrategyPatternExample is created in eclipse IDE.

2. Define Strategy Interface:

* Create an interface PaymentStrategy with a method **pay()**.

public interface PaymentStrategy {

void pay(double amount);

}

* The PaymentStrategy interface defines a single method pay() to be implemented by all payment strategies

3. Implement Concrete Strategies:

* Create classes **CreditCardPayment**, **PayPalPayment** that implement **PaymentStrategy.**

public class CreditCardPayment implements PaymentStrategy {

private String cardNumber;

private String cardHolderName;

private String cvv;

private String expirationDate;

public CreditCardPayment(String cardNumber, String cardHolderName, String cvv, String expirationDate) {

this.cardNumber = cardNumber;

this.cardHolderName = cardHolderName;

this.cvv = cvv;

this.expirationDate = expirationDate;

}

@Override

public void pay(double amount) {

System.out.println("Paid $" + amount + " using Credit Card.");

}

}

public class PayPalPayment implements PaymentStrategy {

private String email;

private String password;

public PayPalPayment(String email, String password) {

this.email = email;

this.password = password;

}

@Override

public void pay(double amount) {

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

}

}

* The CreditCardPayment and PayPalPayment classes implement the PaymentStrategy interface and provide specific implementations of the pay() method.

4. Implement Context Class:

* Create a class **PaymentContext** that holds a reference to **PaymentStrategy** and a method to execute the strategy.

public class PaymentContext {

private PaymentStrategy paymentStrategy;

public void setPaymentStrategy(PaymentStrategy paymentStrategy) {

this.paymentStrategy = paymentStrategy;

}

public void executePayment(double amount) {

if (paymentStrategy != null) {

paymentStrategy.pay(amount);

} else {

System.out.println("Payment strategy not set.");

}

}

}

* The PaymentContext class holds a reference to a PaymentStrategy and has a method executePayment() to execute the payment strategy.

5. Test the Strategy Implementation:

* Create a test class to demonstrate selecting and using different payment strategies.

import java.util.Scanner;

public class Test {

public static void main(String[] args) {

PaymentContext context = new PaymentContext();

Scanner scanner = new Scanner(System.in);

while (true) {

System.out.println("Select payment method: 1. Credit Card 2. PayPal 3. Exit");

int choice = scanner.nextInt();

scanner.nextLine();

if (choice == 3) {

break;

}

System.out.print("Enter amount to pay: ");

double amount = scanner.nextDouble();

scanner.nextLine();

switch (choice) {

case 1:

System.out.print("Enter Credit Card Number: ");

String cardNumber = scanner.nextLine();

System.out.print("Enter Card Holder Name: ");

String cardHolderName = scanner.nextLine();

System.out.print("Enter CVV: ");

String cvv = scanner.nextLine();

System.out.print("Enter Expiration Date: ");

String expirationDate = scanner.nextLine();

context.setPaymentStrategy(new CreditCardPayment(cardNumber, cardHolderName, cvv, expirationDate));

break;

case 2:

System.out.print("Enter PayPal Email: ");

String email = scanner.nextLine();

System.out.print("Enter PayPal Password: ");

String password = scanner.nextLine();

context.setPaymentStrategy(new PayPalPayment(email, password));

break;

default:

System.out.println("Invalid choice.");

continue;

}

context.executePayment(amount);

}

scanner.close();

}

}

* Select different payment methods and enter the required information.
* Observe that the appropriate payment strategy is executed.

Output:

