Exercise 8: Implementing the Strategy Pattern

Objective

To build a payment system that allows selecting different payment methods (Credit Card, PayPal, etc.) at runtime using the Strategy Pattern.

Java Code – StrategyPatternExample.java

// Step 1: Strategy Interface interface PaymentStrategy {

void pay(double amount);

}

// Step 2a: Concrete Strategy - Credit Card

class CreditCardPayment implements PaymentStrategy { private String cardNumber;

public CreditCardPayment(String cardNumber) { this.cardNumber = cardNumber;

}

@Override

public void pay(double amount) {

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

}

}

// Step 2b: Concrete Strategy - PayPal

class PayPalPayment implements PaymentStrategy { private String email;

public PayPalPayment(String email) { this.email = email;

}

@Override

public void pay(double amount) {

System.out.println("Paid $" + amount + " using PayPal account: "

+ email);

}

}

// Step 3: Context Class class PaymentContext {

private PaymentStrategy strategy;

// Set payment strategy dynamically

public void setPaymentStrategy(PaymentStrategy strategy) { this.strategy = strategy;

}

// Execute payment strategy public void pay(double amount) {

if (strategy == null) {

System.out.println("No payment strategy selected!");

} else {

strategy.pay(amount);

}

}

}

// Step 4: Test Class

public class StrategyPatternExample {

public static void main(String[] args) { PaymentContext context = new PaymentContext();

5432"));

// Pay using Credit Card

context.setPaymentStrategy(new CreditCardPayment("1234-5678-9876- context.pay(250.00);

// Pay using PayPal context.setPaymentStrategy(new

PayPalPayment("[user@example.com"](mailto:user@example.com))); context.pay(500.00);

}

}

Sample Output

Paid $250.0 using Credit Card: 1234-5678-9876-5432 Paid $500.0 using PayPal account: [user@example.com](mailto:user@example.com)

Key Benefits of Strategy Pattern

Feature Description

Encapsulation of Algorithms Each payment method is isolated in its own class

Runtime Flexibility Strategy can be changed dynamically during execution

Code Reusability & Maintainability Easy to extend with new payment types