**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.

**Steps:**

1. **Create a New Java Project:**
   * Create a new Java project named **StrategyPatternExample**.
2. **Define Strategy Interface:**
   * Create an interface PaymentStrategy with a method **pay()**.
3. **Implement Concrete Strategies:**
   * Create classes **CreditCardPayment**, **PayPalPayment** that implement **PaymentStrategy**.
4. **Implement Context Class:**
   * Create a class **PaymentContext** that holds a reference to **PaymentStrategy** and a method to execute the strategy.
5. **Test the Strategy Implementation:**
   * Create a test class to demonstrate selecting and using different payment strategies.

**CODE:**

using System;

interface IPaymentStrategy

{

void Pay(decimal amount);

}

class CreditCardPayment : IPaymentStrategy

{

public void Pay(decimal amount)

{

Console.WriteLine($"Paid {amount} using Credit Card.");

}

}

class PayPalPayment : IPaymentStrategy

{

public void Pay(decimal amount)

{

Console.WriteLine($"Paid {amount} using PayPal.");

}

}

class PaymentContext

{

private IPaymentStrategy strategy;

public void SetPaymentStrategy(IPaymentStrategy strategy)

{

this.strategy = strategy;

}

public void ExecutePayment(decimal amount)

{

if (strategy == null)

{

Console.WriteLine("No payment strategy set.");

}

else

{

strategy.Pay(amount);

}

}

}

class Program

{

static void Main()

{

PaymentContext context = new PaymentContext();

// Use CreditCardPayment strategy

context.SetPaymentStrategy(new CreditCardPayment());

context.ExecutePayment(250.00m);

// Switch to PayPalPayment strategy

context.SetPaymentStrategy(new PayPalPayment());

context.ExecutePayment(150.75m);

}

}

**OUTPUT:**

