МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЗАПОРІЗЬКА ПОЛІТЕХНІКА»

кафедра програмних засобів

Звіт

з лабораторної роботи № 4

з дисципліни «Проектний практикум»

на тему:

«Шаблони поведінки. Частина 2»

Виконав

Ст. гр. КНТ-137 В.В. Козлов

Прийняв

Доцент, к.т.н. Н.О. Миронова

1. Мета роботи

Вивчити шаблони поведінки та отримати базові навички із застосування шаблонів Знімок (Memento), Стан (State), Команда (Command), Інтерпретатор (Interpreter), Шаблонний метод (Template method).

1. Завдання на лабораторну роботу

Вивчити шаблони поведінки, їхню загальну характеристику та призначення. Для кожного з цих шаблонів:

* вивчити призначення шаблону, альтернативні назви, призначення, випадки коли його застосування є доцільним та результати такого застосування;
* знати особливості реалізації шаблону, споріднені шаблони, відомі випадки його застосування в програмних додатках;
* вільно володіти структурою шаблону, призначенням його класів та відносинами між ними.

За допомогою джерел зі списку рекомендованих для виконання лабораторної роботи, виконати програмну реалізацію шаблонів поведінки.

1. Результати виконання роботи
   1. Реалізація паттерну Command
      1. Лістинг файлу PatternCommand.cs

using System;

namespace PP\_Lab4.Patterns

{

public interface ICommand

{

public void MakeOrder();

}

public class ChefCook

{

public string Name { get; set; }

public ChefCook(string name)

{

Name = name;

}

public void MakePizza()

{

Console.WriteLine($"Chef cook {Name} get pizza order");

}

public void MakeSushi()

{

Console.WriteLine($"Chef cook {Name} get sushi order");

}

}

public class PizzaCommand : ICommand

{

private ChefCook ChefCook { get; set; }

public PizzaCommand(ChefCook chefCook)

{

ChefCook = chefCook;

}

public void MakeOrder()

{

ChefCook.MakePizza();

}

}

public class SushiCommand : ICommand

{

private ChefCook ChefCook { get; set; }

public SushiCommand(ChefCook chefCook)

{

ChefCook = chefCook;

}

public void MakeOrder()

{

ChefCook.MakeSushi();

}

}

public class Client

{

public ICommand Command { get; set; }

public Client() { ;}

public Client(ICommand command)

{

SetCommand(command);

}

public void SetCommand(ICommand command)

{

Command = command;

}

public void MakeOrder()

{

Command.MakeOrder();

}

}

}

* 1. Реалізація паттерну Interpreter
     1. Лістинг файлу PatternInterpreter.cs

using System.Collections.Generic;

namespace PP\_Lab4.Patterns

{

public class Context

{

public Dictionary<char, int> Dictionary;

public Context(Dictionary<char, int> dictionary)

{

Dictionary = dictionary;

}

}

public interface IExpression

{

public int Interpret(Context context);

}

public class Number : IExpression

{

public char Char { get; set; }

public Number(char number)

{

Char = number;

}

public int Interpret(Context context)

{

return context.Dictionary[Char];

}

}

public class Inversion : IExpression

{

public IExpression Expression { get; set; }

public Inversion(IExpression expression)

{

Expression = expression;

}

public int Interpret(Context context)

{

return -Expression.Interpret(context);

}

}

public class Add : IExpression

{

private IExpression \_Left { get; set; }

private IExpression \_Right { get; set; }

public Add(IExpression left, IExpression right)

{

\_Left = left;

\_Right = right;

}

public int Interpret(Context context)

{

return \_Left.Interpret(context) + \_Right.Interpret(context);

}

}

}

* 1. Реалізація паттерну Memento
     1. Лістинг файлу PatternMemento.cs

using System;

using System.Collections.Generic;

namespace PP\_Lab4.Patterns

{

public class Memento

{

private List<string> \_Shapes;

public Memento(List<string> shapes)

{

\_Shapes = new List<string>(shapes);

}

public List<string> getShapes()

{

return \_Shapes;

}

}

public class Picture

{

private List<string> \_Shapes;

public Picture()

{

\_Shapes = new List<string>();

}

public Memento Save()

{

return new Memento(\_Shapes);

}

public void ReDraw(Memento memento)

{

\_Shapes = memento.getShapes();

}

public void ShowPicture()

{

Console.Write("On this picture: ");

foreach (string shape in \_Shapes)

Console.Write(shape + " ");

Console.WriteLine();

}

public void AddShape(string shape)

{

\_Shapes.Add(shape);

}

}

public class GraphicEditor

{

private Picture \_Picture { get; set; }

private Stack<Memento> \_History { get; set; }

public GraphicEditor()

{

\_Picture = new Picture();

\_History = new Stack<Memento>();

}

public void ShowPicture()

{

\_Picture.ShowPicture();

}

public void AddShape(string shape)

{

\_Picture.AddShape(shape);

}

public void Save()

{

\_History.Push(\_Picture.Save());

}

public void Undo()

{

if (\_History.Count > 0)

\_Picture.ReDraw(\_History.Pop());

}

}

}

* 1. Реалізація паттерну State
     1. Лістинг файлу PatternState.cs

using System;

namespace PP\_Lab4.Patterns

{

public interface IState

{

public void Tap();

public void Block();

public void SetSmartphone(Smartphone smartphone);

}

public class Smartphone

{

private IState State { get; set; }

public Smartphone()

{

ChangeState(new BlockedState());

}

public void ChangeState(IState state)

{

State = state;

State.SetSmartphone(this);

}

public void BlockButton()

{

State.Block();

}

public void Tap()

{

State.Tap();

}

}

public class BlockedState : IState

{

public Smartphone Smartphone { get; set; }

public BlockedState() { ;}

public void SetSmartphone(Smartphone smartphone)

{

Smartphone = smartphone;

}

public void Tap()

{

Console.WriteLine("On tap : No reaction");

}

public void Block()

{

Console.WriteLine("On block button : Smartphone unblocked");

Smartphone.ChangeState(new UnblockedState());

}

}

public class UnblockedState : IState

{

public Smartphone Smartphone { get; set; }

public UnblockedState() {; }

public void SetSmartphone(Smartphone smartphone)

{

Smartphone = smartphone;

}

public void Tap()

{

Console.WriteLine("On tap : tap commited");

}

public void Block()

{

Console.WriteLine("On block button : Smartphone blocked");

Smartphone.ChangeState(new BlockedState());

}

}

}

* 1. Реалізація паттерну Template method
     1. Лістинг файлу PatternTemplateMethod.cs

using System;

namespace PP\_Lab4.Patterns

{

public abstract class CarMaker

{

public CarMaker() { ;}

public void MakeBody()

{

Console.WriteLine("Car body is made");

}

public void Paint()

{

Console.WriteLine("Car is painted");

}

public abstract void StickLogo();

}

public class FordCar : CarMaker

{

public FordCar() { ;}

public override void StickLogo()

{

Console.WriteLine("Ford logo is stiked");

}

}

public class ZAZCar : CarMaker

{

public ZAZCar() {; }

public override void StickLogo()

{

Console.WriteLine("ZAZ logo is stiked");

}

}

}

* 1. Використання паттернів
     1. Лістинг файлу Program.cs

using System;

using System.Collections.Generic;

using PP\_Lab4.Patterns;

namespace PP\_Lab4

{

class Program

{

static void Main(string[] args)

{

showMemento();

showState();

showCommand();

showInterpreter();

showTemplateMethod();

}

static void showMemento()

{

Console.WriteLine("Memento:");

GraphicEditor Paint = new GraphicEditor();

Paint.AddShape("Dot");

Paint.AddShape("Circle");

Paint.Save();

Paint.AddShape("Square");

Console.WriteLine("Before undo: ");

Paint.ShowPicture();

Paint.Undo();

Console.WriteLine("After undo: ");

Paint.ShowPicture();

Console.WriteLine();

}

static void showState()

{

Console.WriteLine("State:");

Smartphone MeizuM6s = new Smartphone();

MeizuM6s.Tap();

MeizuM6s.BlockButton();

MeizuM6s.Tap();

MeizuM6s.Tap();

MeizuM6s.BlockButton();

Console.WriteLine();

}

static void showCommand()

{

Console.WriteLine("Command:");

ChefCook Valeriy = new ChefCook("Valeriy");

Client IlonMask = new Client(new PizzaCommand(Valeriy));

Console.WriteLine("Client make order: ");

IlonMask.MakeOrder();

IlonMask.SetCommand(new SushiCommand(Valeriy));

Console.WriteLine("Client make order: ");

IlonMask.MakeOrder();

Console.WriteLine();

}

static void showInterpreter()

{

Console.WriteLine("Interpreter:");

Context Parametrs = new Context(new Dictionary<char, int>() { { 'x', 3 }, { 'y', 2 }, { 'z', 1 } });

Console.WriteLine("x = 3, y = 2, z = 1 ");

IExpression MathExpression = new Add(new Add(new Number('x'), new Number('y')), new Inversion(new Number('z')));

Console.Write("(x + y) + -z = ");

Console.WriteLine(MathExpression.Interpret(Parametrs));

MathExpression = new Add(new Inversion(new Add(new Number('x'), new Number('y'))), new Inversion(new Number('z')));

Console.Write("-(x + y) + -z = ");

Console.WriteLine(MathExpression.Interpret(Parametrs));

Console.WriteLine();

}

static void showTemplateMethod()

{

Console.WriteLine("Template method:");

CarMaker Manufacturer = new FordCar();

Console.WriteLine("Manufacturer make car: ");

Manufacturer.MakeBody();

Manufacturer.Paint();

Manufacturer.StickLogo();

Manufacturer = new ZAZCar();

Console.WriteLine("Manufacturer make car: ");

Manufacturer.MakeBody();

Manufacturer.Paint();

Manufacturer.StickLogo();

Console.WriteLine();

}

}

}

* + 1. Скріншоти роботи програми

На рисунку 3.1 наведено результати виконання програми, яка демонструє використання усіх шаблонів поведінки.

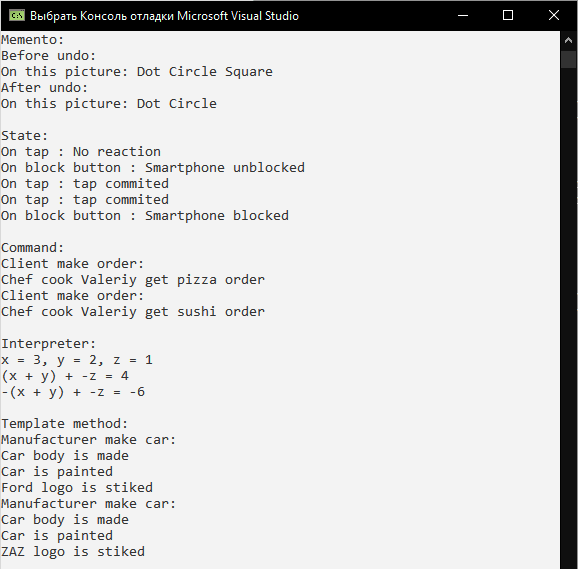


Рисунок 3.1 – Резульати виконання програми

Висновки

Протягом виконання цієї лабораторної роботи ознайомився з видами шаблонів проектування ПЗ, вивчив шаблони поведінки та отримав базові навички із застосування шаблонів Знімок (Memento), Стан (State), Команда (Command), Інтерпретатор (Interpreter), Шаблонний метод (Template method).

Створив програму, з використанням різних шаблонів поведінки мовою програмування C#.