МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЗАПОРІЗЬКА ПОЛІТЕХНІКА»

кафедра програмних засобів

Звіт

з лабораторної роботи № 2

з дисципліни «Проектний практикум»

на тему:

«Структурні шаблони проектування»

Виконав

Ст. гр. КНТ-137 В.В. Козлов

Прийняв

Доцент, к.т.н. Н.О. Миронова

1. Мета роботи

Вивчити структурні шаблони, отримати базові навички із застосування шаблонів Адаптер (Adapter), Декоратор (Decorator), Замісник (Proxy), Компонувальник (Composite), Міст (Bridge), Легковаговик (Flyweight), Фасад (Facade).

1. Завдання на лабораторну роботу

Вивчити структурні шаблони проектування, 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 структурних шаблонів.

1. Результати виконання роботи
   1. Реалізація паттерну Adapter
      1. Лістинг файлу PatternAdapter.cs

namespace PP\_Lab2.Patterns

{

class AmericanMetric

{

public double FeetHeight { get; set; }

public double PoundWeight { get; set; }

public AmericanMetric(double feetHeight, double poundWeight)

{

FeetHeight = feetHeight;

PoundWeight = poundWeight;

}

}

interface IMetric

{

public double getHeight();

public double getWeight();

}

class AmericanToUkrainianMetricAdapter : IMetric

{

private AmericanMetric \_AmericanMetric { get; set; }

public AmericanToUkrainianMetricAdapter(AmericanMetric americanMetric)

{

\_AmericanMetric = americanMetric;

}

public double getHeight()

{

return \_AmericanMetric.FeetHeight / 3.2808;

}

public double getWeight()

{

return \_AmericanMetric.PoundWeight / 2.2046;

}

}

}

* 1. Реалізація паттерну Bridge
     1. Лістинг файлу PatternBridge.cs

using System;

namespace PP\_Lab2.Patterns

{

public interface ICar

{

public bool isEngineStarted();

public void startEngine();

public void stopEngine();

public void blinkHeadlights();

}

public class Ford : ICar

{

private bool isStarted { get; set; } = false;

public Ford () { ;}

public bool isEngineStarted()

{

return isStarted;

}

public void startEngine()

{

if (isEngineStarted())

Console.WriteLine("Ford engine is already started");

else

{

Console.WriteLine("Ford start engine");

isStarted = true;

}

}

public void stopEngine()

{

if (!isEngineStarted())

Console.WriteLine("Ford engine is already stopped");

else

{

Console.WriteLine("Ford stop engine");

isStarted = false;

}

}

public void blinkHeadlights()

{

Console.WriteLine("Ford blinked headlights");

}

}

public class CarKey

{

protected ICar \_Car { get; set; }

public CarKey(ICar car)

{

\_Car = car;

}

public void startEngine()

{

Console.Write("Try to start engine: ");

\_Car.startEngine();

}

public void stopEngine()

{

Console.Write("Try to stop engine: ");

\_Car.stopEngine();

}

}

public class AdvancedCarKey : CarKey

{

public AdvancedCarKey(ICar car) : base(car) { ;}

public void blinkHeadlights()

{

Console.Write("Try to blink headlights: ");

\_Car.blinkHeadlights();

}

}

}

* 1. Реалізація паттерну Composite
     1. Лістинг файлу PatternComposite.cs

using System;

using System.Collections.Generic;

namespace PP\_Lab2.Patterns

{

public interface IPackage

{

public void showInfo();

}

public class Package: IPackage

{

public string Name { get; set; }

public Package(String name)

{

Name = name;

}

public void showInfo()

{

Console.WriteLine(Name);

}

}

public class PackageBox : IPackage

{

string Name { get; set; }

public List<IPackage> Packages { get; set; }

public PackageBox(string name)

{

Packages = new List<IPackage>();

Name = name;

}

public void showInfo()

{

Console.WriteLine(Name + '\*');

foreach (IPackage package in Packages)

{

package.showInfo();

}

}

public void add(IPackage package)

{

Packages.Add(package);

}

public void remove(IPackage package)

{

Packages.Remove(package);

}

}

}

* 1. Реалізація паттерну Decorator
     1. Лістинг файлу PatternDecorator.cs

namespace PP\_Lab2.Patterns

{

public interface IHuman

{

public string showName();

public string showСlothes();

}

public class Human : IHuman

{

string Name { get; set; }

public Human(string name)

{

Name = name;

}

public string showName()

{

return Name;

}

public string showСlothes()

{

return Name + " wearing:";

}

}

public class HumanDecorator : IHuman

{

private IHuman \_Wrapee {get; set;}

public HumanDecorator(IHuman wrapee)

{

\_Wrapee = wrapee;

}

public string showName()

{

return \_Wrapee.showName();

}

public string showСlothes()

{

return \_Wrapee.showСlothes();

}

}

public class FootDecorator : HumanDecorator, IHuman

{

public string FootClothes { get; set; }

public FootDecorator(IHuman wrapee, string footClothes) : base(wrapee)

{

FootClothes = footClothes;

}

new public string showСlothes()

{

string res = base.showСlothes() + "\nOn foot " + FootClothes;

return res;

}

}

public class BodyDecorator : HumanDecorator, IHuman

{

public BodyDecorator(IHuman wrapee) : base(wrapee) { ;}

new public string showСlothes()

{

string res = base.showСlothes() + "\nOn body Jacket";

return res;

}

}

}

* 1. Реалізація паттерну Facade
     1. Лістинг файлу PatternFacade.cs

using System;

namespace PP\_Lab2.Patterns

{

class Mechanic

{

public Mechanic() { ;}

public void RepairCar()

{

Console.WriteLine("Mechanic fixed a malfunction in a car");

}

}

class Painter

{

public Painter() {; }

public void PaintCar()

{

Console.WriteLine("Painter primed and painted the car");

}

}

public class ServiceStation

{

private Mechanic \_Mechanic { get; set; }

private Painter \_Painter { get; set; }

public ServiceStation()

{

\_Mechanic = new Mechanic();

\_Painter = new Painter();

}

public void ServiceCar()

{

Console.WriteLine("Service work:");

\_Mechanic.RepairCar();

\_Painter.PaintCar();

}

}

}

* 1. Реалізація паттерну Flyweight
     1. Лістинг файлу PatternFlyweight.cs

using System;

using System.Collections.Generic;

namespace PP\_Lab2.Patterns

{

public class CarType

{

public string Type { get; }

public CarType(string type)

{

Type = type;

}

public void ShowCarType()

{

Console.Write(Type + " ");

}

}

public class Car

{

public string Number { get; set; }

public CarType CarType { get; set; }

public Car(string number, CarType carType)

{

Number = number;

CarType = carType;

}

public void ShowCar()

{

CarType.ShowCarType();

Console.WriteLine(Number);

}

}

public class CarFactory

{

public List<CarType> \_CarTypes { get; set; }

public CarFactory()

{

\_CarTypes = new List<CarType>();

}

public CarType getCarType(string type)

{

foreach(CarType carType in \_CarTypes)

if (carType.Type == type) return carType;

CarType newCarType = new CarType(type);

\_CarTypes.Add(newCarType);

return newCarType;

}

}

public class Parking

{

public CarFactory CarFactory { get; set; }

public List<Car> Cars { get; set; }

public Parking()

{

CarFactory = new CarFactory();

Cars = new List<Car>();

}

public void addCar(string type, string number)

{

Cars.Add(new Car(number, CarFactory.getCarType(type)));

}

public void showCars()

{

foreach (Car car in Cars)

car.ShowCar();

}

}

}

* 1. Реалізація паттерну Proxy
     1. Лістинг файлу PatternProxy.cs

using System;

namespace PP\_Lab2.Patterns

{

public interface IDataBase

{

public void set(string data);

public string get();

}

public class DataBase : IDataBase

{

private string \_Data {get; set;}

public DataBase()

{

\_Data = "Security data";

}

public void set(string data)

{

\_Data = data;

}

public string get()

{

return \_Data;

}

}

public class DataBaseProxy : IDataBase

{

private DataBase \_DataBase;

string lastData = "";

public DataBaseProxy() { ;}

public DataBaseProxy(DataBase service)

{

\_DataBase = service;

}

private void Init()

{

if (\_DataBase == null)

{

\_DataBase = new DataBase();

Console.WriteLine("Init DataBase");

}

}

public void set(string data)

{

Init();

if (lastData == data)

Console.WriteLine("Attempt to set the same data");

else

{

\_DataBase.set(data);

lastData = data;

Console.WriteLine("Update data in Database to {0}", lastData);

}

}

public string get()

{

Init();

lastData = \_DataBase.get();

return lastData;

}

}

}

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

using System;

using PP\_Lab2.Patterns;

namespace PP\_Lab2

{

class Program

{

static void Main(string[] args)

{

ShowAdapter();

ShowDecorator();

ShowProxy();

ShowComposite();

ShowBridge();

ShowFlyweight();

ShowFacade();

}

static void ShowAdapter()

{

Console.WriteLine("Adapter:");

AmericanMetric PersonInfo = new AmericanMetric(6, 183);

Console.WriteLine("Person information in American metric:");

Console.WriteLine(String.Format("Height: {0:0.##} ft.", PersonInfo.FeetHeight));

Console.WriteLine(String.Format("Weight: {0:0.##} lb.", PersonInfo.PoundWeight));

IMetric UkrainianMetric = new AmericanToUkrainianMetricAdapter(PersonInfo);

Console.WriteLine("Person information in Ukrainian metric:");

Console.WriteLine(String.Format("Height: {0:0.##} m.", UkrainianMetric.getHeight()));

Console.WriteLine(String.Format("Weight: {0:0.##} kg.", UkrainianMetric.getWeight()));

Console.WriteLine();

}

static void ShowDecorator()

{

Console.WriteLine("Decorator:");

IHuman Valeriy = new Human("Valeriy");

Valeriy = new FootDecorator(Valeriy, "Socks");

Valeriy = new FootDecorator(Valeriy, "Pants");

Valeriy = new BodyDecorator(Valeriy);

Console.WriteLine(Valeriy.showСlothes());

Console.WriteLine();

}

static void ShowProxy()

{

Console.WriteLine("Proxy:");

IDataBase MyDataBase = new DataBaseProxy();

Console.WriteLine("Proxy DataBase created, but not initialized");

Console.WriteLine("Data in DataBase: {0}", MyDataBase.get());

MyDataBase.set("Valeriy Kozlov");

MyDataBase.set("Valeriy Kozlov");

Console.WriteLine();

}

static void ShowComposite()

{

Console.WriteLine("Composite:");

PackageBox Smartphone = new PackageBox("Smartphone");

Smartphone.add(new Package("Meizu M6s"));

Smartphone.add(new Package("Power supply"));

Smartphone.add(new Package("Warranty"));

PackageBox Gift = new PackageBox("Gift box");

Gift.add(new Package("Headphones"));

Gift.add(Smartphone);

Gift.add(new Package("Candy"));

Console.WriteLine("Gift consists of: ");

Gift.showInfo();

Console.WriteLine();

}

static void ShowBridge()

{

Console.WriteLine("Bridge:");

ICar FordMustang = new Ford();

CarKey SimpleKey = new CarKey(FordMustang);

AdvancedCarKey AdvancedKey = new AdvancedCarKey(FordMustang);

SimpleKey.startEngine();

AdvancedKey.startEngine();

AdvancedKey.blinkHeadlights();

AdvancedKey.stopEngine();

Console.WriteLine();

}

static void ShowFlyweight()

{

Console.WriteLine("Flyweight:");

Parking MyParking = new Parking();

MyParking.addCar("Ford Mustang GT", "AA0001AA");

MyParking.addCar("Ford Mustang GT", "AA0002AA");

MyParking.addCar("Ford Fusion", "AA0003AA");

MyParking.addCar("Ford Fusion", "AA0004AA");

MyParking.addCar("Ford Fusion", "AA0005AA");

Console.WriteLine("Cars in parking:");

MyParking.showCars();

Console.WriteLine("Cars count: {0}", MyParking.Cars.Count);

Console.WriteLine("Car types count: {0}", MyParking.CarFactory.\_CarTypes.Count);

Console.WriteLine();

}

static void ShowFacade()

{

Console.WriteLine("Facade:");

ServiceStation SimpleSTO = new ServiceStation();

SimpleSTO.ServiceCar();

Console.WriteLine();

}

}

}

* + 1. Скріншоти роботи програми

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

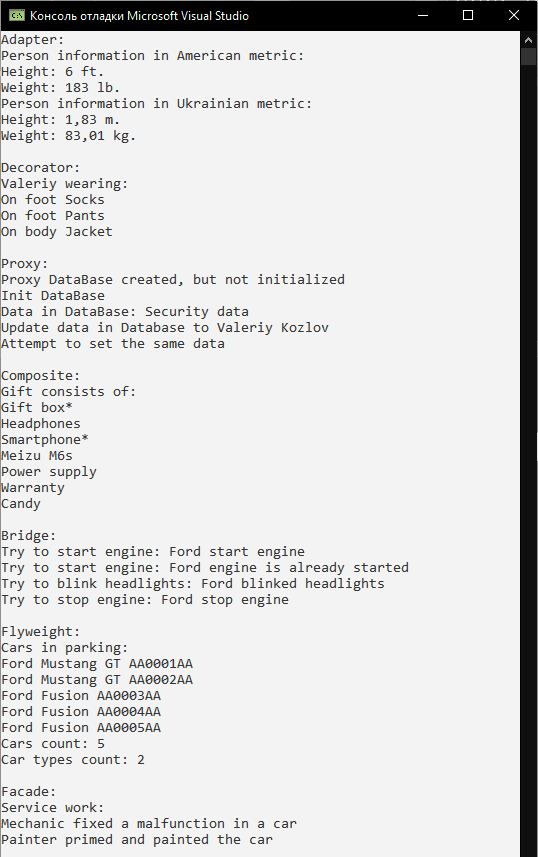


Рисунок 3.1 – Резульати виконання програми

Висновки

Протягом виконання цієї лабораторної роботи ознайомився з видами шаблонів проектування ПЗ, вивчив структурні шаблони та отримав базові навички із застосування шаблонів Адаптер (Adapter), Декоратор (Decorator), Замісник (Proxy), Компонувальник (Composite), Міст (Bridge), Легковаговик (Flyweight) та Фасад (Facade).

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