МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЗАПОРІЗЬКА ПОЛІТЕХНІКА»

кафедра програмних засобів

Звіт

з лабораторної роботи № 1

з дисципліни «Проектний практикум»

на тему:

«Твірні шаблони»

Виконав

Ст. гр. КНТ-137 В.В. Козлов

Прийняв

Доцент, к.т.н. Н.О. Миронова

1. Мета роботи

Ознайомитися з видами шаблонів проектування ПЗ, вивчити твірні шаблони та отримати тримання базових навичок з застосування шаблонів Прототип (Prototype), Одинак (Singleton), Фабричний метод (Factory Method), Абстрактна фабрика (Abstract Factory) та Будівник (Builder).

1. Завдання на лабораторну роботу

Вивчити твірні 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 твірних шаблонів.

1. Результати виконання роботи
   1. Реалізація паттерну Singleton
      1. Лістинг файлу PatternSingleton.cs

namespace PP\_Lab1.Patterns

{

class ProgrammInformation

{

private static ProgrammInformation \_programmInformation;

public string Readme { get; set; }

private ProgrammInformation()

{

Readme = "Author Valeriy Kozlov";

}

public static ProgrammInformation GetProgrammInformation()

{

if(\_programmInformation == null)

\_programmInformation = new ProgrammInformation();

return \_programmInformation;

}

}

}

* 1. Реалізація паттерну Prototype
     1. Лістинг файлу PatternPrototype.cs

namespace PP\_Lab1.Patterns

{

public interface IComputer

{

string ToString();

IComputer clone();

}

public abstract class Computer : IComputer

{

public string Model { get; set; }

public abstract IComputer clone();

public override string ToString()

{

return Model;

}

public Computer(Computer source)

{

this.Model = source.Model;

}

public Computer()

{

;

}

}

class DesktopComputer : Computer

{

public string EthernetAdapter { get; set; }

public DesktopComputer(DesktopComputer source) : base(source)

{

this.EthernetAdapter = source.EthernetAdapter;

}

public DesktopComputer()

{

;

}

public override IComputer clone()

{

return new DesktopComputer(this);

}

}

class Notebook : Computer

{

public string WiFiAdapter { get; set; }

public Notebook(Notebook source) : base(source)

{

this.WiFiAdapter = source.WiFiAdapter;

}

public Notebook()

{

;

}

public override IComputer clone()

{

return new Notebook(this);

}

}

}

* 1. Реалізація паттерну Factory method
     1. Лістинг файлу PatternFactoryMethod.cs

namespace PP\_Lab1.Patterns

{

public interface ITransport

{

public string GetInformation();

}

public class Car : ITransport

{

public string GetInformation()

{

return "Current transport is car";

}

}

public class Ship : ITransport

{

public string GetInformation()

{

return "Current transport is ship";

}

}

public interface ITransportFactory

{

ITransport Create();

}

public class CarFactory : ITransportFactory

{

public ITransport Create()

{

return new Car();

}

}

public class ShipFactory : ITransportFactory

{

public ITransport Create()

{

return new Ship();

}

}

}

* 1. Реалізація паттерну Abstract factory
     1. Лістинг файлу PatternAbstractFactory.cs

namespace PP\_Lab1.Patterns

{

public interface IFlag

{

string ShowFlag();

}

public class UkrainianFlag : IFlag

{

public string ShowFlag()

{

return "Ukrainian";

}

}

public class PortugalFlag : IFlag

{

public string ShowFlag()

{

return "Protugal";

}

}

public interface ICoach

{

string ShowCoach();

}

public class UkrainianCoach : ICoach

{

public string ShowCoach()

{

return "Andrey Shevchenko";

}

}

public class PortugalCoach : ICoach

{

public string ShowCoach()

{

return "Fernando Santos";

}

}

public interface ITeamFactory

{

IFlag CreateFlag();

ICoach CreateCoach();

}

public class UkrainianTeam : ITeamFactory

{

public IFlag CreateFlag()

{

return new UkrainianFlag();

}

public ICoach CreateCoach()

{

return new UkrainianCoach();

}

}

public class PortugalTeam : ITeamFactory

{

public IFlag CreateFlag()

{

return new PortugalFlag();

}

public ICoach CreateCoach()

{

return new PortugalCoach();

}

}

public class FootballTeam

{

public IFlag Flag { get; set; }

public ICoach Coach { get; set; }

public FootballTeam(ITeamFactory TeamFactory)

{

Flag = TeamFactory.CreateFlag();

Coach = TeamFactory.CreateCoach();

}

public string GetInformation()

{

return $"Flag: {Flag.ShowFlag()}, Coach: {Coach.ShowCoach()}";

}

}

}

* 1. Реалізація паттерну Builder
     1. Лістинг файлу PatternBuilder.cs

using System.Collections.Generic;

namespace PP\_Lab1.Patterns

{

public class Pizza

{

public string Name { get; set; }

public List<string> Ingredients { get; set; }

public Pizza()

{

Ingredients = new List<string>();

}

public override string ToString()

{

string result = Name + ": ";

foreach (string ingredient in Ingredients)

result += (ingredient + ", ");

return result.TrimEnd(new char[] { ',', ' '});

}

}

public class Recipe

{

public string Name { get; set; }

public List<string> Ingredients { get; set; }

public Recipe()

{

Ingredients = new List<string>();

}

public override string ToString()

{

string result = Name + ": ";

foreach (string ingredient in Ingredients)

result += (ingredient + ", ");

return result.TrimEnd(new char[] { ',', ' ' });

}

}

public interface IPizzaMaker

{

void reset();

void SetName(string name);

void SetDough(string dough);

void SetCheese(string cheese);

void SetSausage(string sausage);

}

public class RecipePizzaMaker : IPizzaMaker

{

public Recipe recipe { get; set; }

public void reset()

{

recipe = new Recipe();

}

public void SetName(string name)

{

recipe.Name = name;

}

public void SetDough(string dough)

{

recipe.Ingredients.Add(dough);

}

public void SetCheese(string cheese)

{

recipe.Ingredients.Add(cheese);

}

public void SetSausage(string sausage)

{

recipe.Ingredients.Add(sausage);

}

public Recipe getRecipe()

{

return recipe;

}

}

public class PizzaMaker : IPizzaMaker

{

public Pizza pizza { get; set;}

public void reset()

{

pizza = new Pizza();

}

public void SetName(string name)

{

pizza.Name = name;

}

public void SetDough(string dough)

{

pizza.Ingredients.Add(dough);

}

public void SetCheese(string cheese)

{

pizza.Ingredients.Add(cheese);

}

public void SetSausage(string sausage)

{

pizza.Ingredients.Add(sausage);

}

public Pizza getPizza()

{

return pizza;

}

}

public class Baker

{

public void MakePepperoni(IPizzaMaker pizzaMaker)

{

pizzaMaker.reset();

pizzaMaker.SetName("Pepperoni");

pizzaMaker.SetDough("Wheat bread");

pizzaMaker.SetCheese("Mozzarella");

pizzaMaker.SetSausage("Smoked sausage");

}

public void MakeMargarita(IPizzaMaker pizzaMaker)

{

pizzaMaker.reset();

pizzaMaker.SetName("Margarita");

pizzaMaker.SetDough("Wheat bread");

pizzaMaker.SetCheese("Parmezan");

}

}

}

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

using System;

using System.Collections.Generic;

using System.Linq;

using PP\_Lab1.Patterns;

namespace PP\_Lab1

{

class Program

{

static void Main(string[] args)

{

ShowSingleton();

ShowPrototype();

ShowFactoryMethod();

ShowAbstractFactory();

ShowBuilder();

}

static void ShowSingleton()

{

Console.WriteLine("Singletone:");

ProgrammInformation info = ProgrammInformation.GetProgrammInformation();

Console.WriteLine("Default information: " + info.Readme);

info.Readme = "Special for Natalya Oleksiyivna";

Console.WriteLine("Modified information: " + ProgrammInformation.GetProgrammInformation().Readme);

Console.WriteLine();

}

static void ShowPrototype()

{

Console.WriteLine("Prototype:");

Notebook myNotebook = new Notebook();

myNotebook.Model = "Lenovo Ideapad 320 IKB";

myNotebook.WiFiAdapter = "Realtek";

DesktopComputer myComputer = new DesktopComputer();

myComputer.Model = "Asus M5QL PRO";

myComputer.EthernetAdapter = "Realtek";

List<IComputer> Computers = new List<IComputer>();

Computers.Add(myNotebook);

Computers.Add(myNotebook.clone());

Computers.Add(myComputer);

Computers.Add(myComputer.clone());

Console.WriteLine("Computers and copies: ");

foreach (IComputer computer in Computers)

Console.WriteLine($"Model: {computer.ToString()}, Class: {computer.GetType().ToString().Split(".").Last()}");

Console.WriteLine();

}

static void ShowFactoryMethod()

{

Console.WriteLine("Factory method:");

List<ITransport> Transport = new List<ITransport>();

CarFactory carFactory = new CarFactory();

ShipFactory shipFactory = new ShipFactory();

Transport.Add(carFactory.Create());

Transport.Add(shipFactory.Create());

Console.WriteLine("Transport: ");

foreach (ITransport transport in Transport)

Console.WriteLine($"Information: {transport.GetInformation()}, Class: {transport.GetType().ToString().Split(".").Last()}");

Console.WriteLine();

}

static void ShowAbstractFactory()

{

Console.WriteLine("Abstract factory: ");

List<FootballTeam> FootballTeams = new List<FootballTeam>();

FootballTeams.Add(new FootballTeam(new UkrainianTeam()));

FootballTeams.Add(new FootballTeam(new PortugalTeam()));

Console.WriteLine("Football teams: ");

foreach (FootballTeam footballTeam in FootballTeams)

Console.WriteLine(footballTeam.GetInformation());

Console.WriteLine();

}

static void ShowBuilder()

{

Console.WriteLine("Builder: ");

Baker ValeriyKozlov = new Baker();

List<Pizza> myPizzas = new List<Pizza>();

List<Recipe> myRecipes = new List<Recipe>();

PizzaMaker myBake = new PizzaMaker();

RecipePizzaMaker myJournal = new RecipePizzaMaker();

ValeriyKozlov.MakePepperoni(myBake);

myPizzas.Add(myBake.getPizza());

ValeriyKozlov.MakeMargarita(myBake);

myPizzas.Add(myBake.getPizza());

ValeriyKozlov.MakePepperoni(myJournal);

myRecipes.Add(myJournal.getRecipe());

ValeriyKozlov.MakeMargarita(myJournal);

myRecipes.Add(myJournal.getRecipe());

Console.WriteLine("My pizzas: ");

foreach (Pizza pizza in myPizzas)

Console.WriteLine(pizza.Name);

Console.WriteLine("My recipes: ");

foreach (Recipe recipe in myRecipes)

Console.WriteLine(recipe.ToString());

Console.WriteLine();

}

}

}

* + 1. Скріншоти роботи програми

На рисунку 3.1 наведено результати виконання програми, яка демонструє використання усіх представлених у завданні твірних шаблонів.

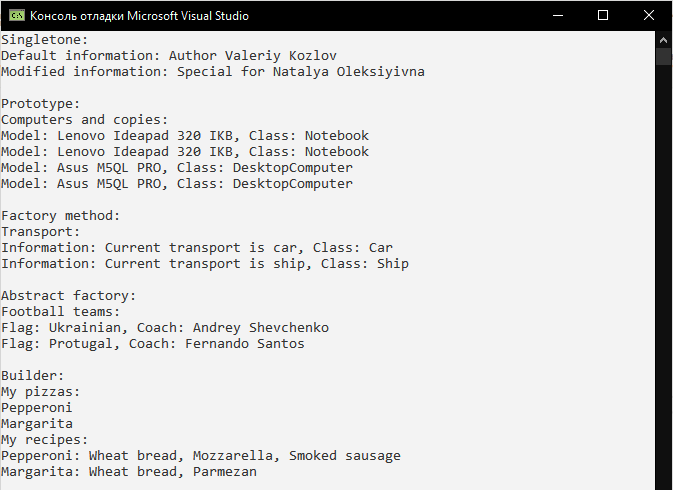


Рисунок 3.1 – Резульати виконання програми

Висновки

Протягом виконання цієї лабораторної роботи ознайомився з видами шаблонів проектування ПЗ, вивчив твірні шаблони та отримав базові навички з застосування шаблонів Прототип (Prototype), Одинак (Singleton), Фабричний метод (Factory Method), Абстрактна фабрика (Abstract Factory) та Будівник (Builder).

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