Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій

Кафедра програмних систем і технологій

ЗВІТ

з лабораторної роботи № 3

Тема: “Патерни”

Дисципліна «Об’єктно-орієнтоване конструювання програм»

Підготував:

студент гр. ІПЗ-23(1)

Гоша Давід Олександрович

Дата : 10.04.22

Перевірила:

доц. Зубик Л.В.

# Київ – 2022

**Тема: Патерни**

**ЗАВДАННЯ**

1. Продумати власний проект (3-й семестр або з ЛР 1-2) так, щоб в коді можливо було застосувати один із розглянутих вище патернів (Стратегія, Спостерігач або Декоратор). Реалізуйте його.

2. Окремо розробити проект і застосувати в коді один із розглянутих вище патернів (Стратегія, Спостерігач або Декоратор) для наведених нижче алгоритмів.

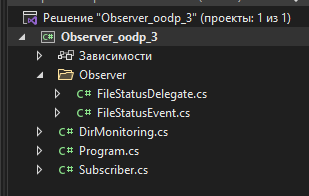
**Варіант 3**

**(**Обчислення периметра правильних багатокутників.**)**

**Завдання 1**

Паттерн – Спостерігач. Програма, що стежить за видаленям файлів у каталозі, та повідомляє підписників у разі видалення.

**Код**



Структура проекту 1

Зміст

[Код структура підписника 2](#_Toc100482899)

[Класс моніторінгу директорії. 3](#_Toc100482900)

[Метод реалізації за допомогою делегатів. 3](#_Toc100482901)

[Метод реалізації за допомогою подій 4](#_Toc100482902)

[Програм файл , який викликає методи і друкує результати. 5](#_Toc100482903)

# Код структура підписника

using System;

namespace Observer\_oodp\_3

{

public class Subscriber

{

private readonly string \_name;

public Subscriber(string name)

{

\_name = name;

}

public void ItIsSubscriber(string fileName)

{

Console.WriteLine($"{\_name} {fileName} was deleted!");

}

public void ItIsSubscriber(object sender, string fileName)

{

Console.WriteLine($"{\_name} {fileName} was deleted!");

}

public void ItIsSecondSubscriber(object sender, string fileName)

{

Console.WriteLine("---");

Console.WriteLine($"{\_name} {fileName} was deleted!");

Console.WriteLine("---");

}

}

}

# Класс моніторінгу директорії.

using System.Collections.Generic;

using System.IO;

namespace Observer\_oodp\_3

{

public class DirMonitoring

{

private List<string> \_files;

private readonly string \_directory;

public DirMonitoring(string directory)

{

\_directory = directory;

}

public bool StartMonitor()

{

if (!Directory.Exists(\_directory))

{

return false;

}

\_files = new List<string>();

var directoryInfo = new DirectoryInfo(\_directory);

foreach (var fileInfo in directoryInfo.GetFiles())

{

\_files.Add(fileInfo.FullName);

}

return true;

}

public List<string> DeletedFiles()

{

var result = new List<string>();

foreach (var file in \_files.ToArray())

{

if (!File.Exists(file))

{

\_files.Remove(file);

result.Add(file);

}

}

return result;

}

}

}

# Метод реалізації за допомогою делегатів.

using System;

using System.Timers;

namespace Observer\_oodp\_3.Observer

{

//Observer pattern - using delegate

//The name of the class is given to understand the type of implementation

public class FileStatusDelegate : IDisposable

{

private readonly Action<string> \_subscriber;

private readonly Timer \_timer;

private readonly DirMonitoring \_dirMonitor;

public FileStatusDelegate(string directory, Action<string> subscriber)

{

\_subscriber = subscriber;

\_dirMonitor = new DirMonitoring(directory);

if (\_dirMonitor.StartMonitor())

{

\_timer = new Timer(1000);

\_timer.Elapsed += CheckRemoval;

\_timer.Start();

}

else

{

Console.WriteLine("Specified directory does not exist");

Dispose();

}

}

private void CheckRemoval(Object source, ElapsedEventArgs e)

{

foreach (var fileName in \_dirMonitor.DeletedFiles())

{

\_subscriber($"{fileName}");

}

}

public void Dispose()

{

\_timer.Dispose();

}

}

}

# Метод реалізації за допомогою подій

using System;

using System.Timers;

namespace Observer\_oodp\_3.Observer

{

public class FileStatusEvent : IDisposable

{

//Observer pattern - using event

//The name of the class is given to understand the type of implementation

public event EventHandler<string> RemoveFiles;

private readonly Timer \_timer;

private readonly DirMonitoring \_dirMonitor;

public FileStatusEvent(string directory)

{

\_dirMonitor = new DirMonitoring(directory);

if (\_dirMonitor.StartMonitor())

{

\_timer = new Timer(1000);

\_timer.Elapsed += CheckRemoval;

\_timer.Start();

}

else

{

Console.WriteLine("Specified directory does not exist");

Dispose();

}

}

private void CheckRemoval(Object source, ElapsedEventArgs e)

{

foreach (var fileName in \_dirMonitor.DeletedFiles())

{

var handler = RemoveFiles;

handler?.Invoke(this, fileName);

}

}

public void Dispose()

{

\_timer.Dispose();

}

}

}

# Програм файл , який викликає методи і друкує результати.

using System;

using Observer\_oodp\_3.Observer;

namespace Observer\_oodp\_3

{

internal class Program

{

static void Main(string[] args)

{

var directory = @"C:\Users\admin\Desktop\Новая папка (2)";

#region FileStatusDelegate

var fileStatusDelegate = new FileStatusDelegate(directory, new Subscriber("").ItIsSubscriber);

Console.ReadKey();

#endregion

#region FileStatusEvent

var eventObserver = new FileStatusEvent(directory);

var subscriber1 = new Subscriber("");

var subscriber2 = new Subscriber("Second");

eventObserver.RemoveFiles += subscriber1.ItIsSubscriber;

eventObserver.RemoveFiles += subscriber2.ItIsSecondSubscriber;

Console.ReadKey();

Console.WriteLine("--- Remove second subscriber ---");

eventObserver.RemoveFiles -= subscriber2.ItIsSecondSubscriber;

Console.ReadKey();

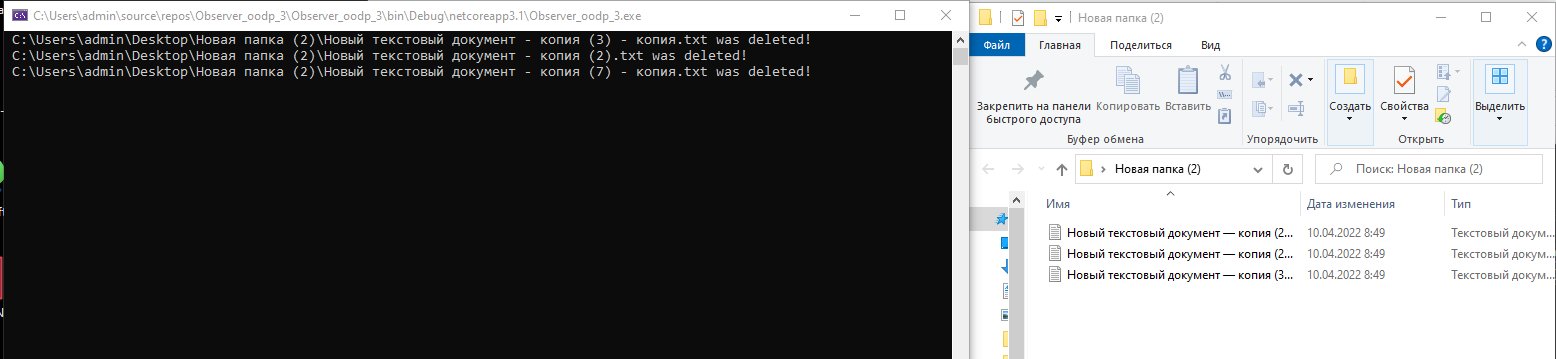
#endregion

}

}

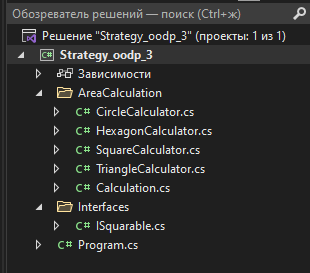
}

## Скриншоти Виконання



**Завдання 2**

Паттерн – Стратегія. Програма, що обчислює периметр правильних багатокутників.



Структура проекту

Зміст

[Інтерфейс роботи методів обрахунку периметру 4](#_Toc100483923)

[Метод обрахунку периметру кола 4](#_Toc100483924)

[Метод обрахунку периметру шестикутника 4](#_Toc100483925)

[Метод обрахунку периметру квадрату 4](#_Toc100483926)

[Метод обрахунку периметру трикутника 5](#_Toc100483927)

[Інтерфейсний метод, що приймає спосіб обрахунку 5](#_Toc100483928)

[Програм файл , що викликає метод для обрахунку периметру 5](#_Toc100483929)

# Інтерфейс роботи методів обрахунку периметру

using System;

using System.Collections.Generic;

using System.Text;

namespace Strategy\_oodp\_3.Interfaces

{

internal interface ISquarable

{

int GetValue(int Square);

}

}

# Метод обрахунку периметру кола

using System;

using Strategy\_oodp\_3.Interfaces;

namespace Strategy\_oodp\_3.AreaСalculation

{

public class CircleCalculator :ISquarable

{

const double pi = 3.14;

public int GetValue(int size)

{

if (size == 0)

return 0;

return Convert.ToInt16(size \* pi);

}

}

}

# Метод обрахунку периметру шестикутника

using Strategy\_oodp\_3.Interfaces;

namespace Strategy\_oodp\_3.AreaСalculation

{

public class HexagonCalculator : ISquarable

{

public int GetValue(int size)

{

if (size == 0)

return 0;

return size \* 6;

}

}

}

# Метод обрахунку периметру квадрату

using Strategy\_oodp\_3.Interfaces;

namespace Strategy\_oodp\_3.AreaСalculation

{

public class SquareCalculator : ISquarable

{

public int GetValue(int size)

{

if (size == 0)

return 0;

return size \* 4;

}

}

}

# Метод обрахунку периметру трикутника

using Strategy\_oodp\_3.Interfaces;

namespace Strategy\_oodp\_3.AreaСalculation

{

class TriangleCalculator : ISquarable

{

public int GetValue(int size)

{

if (size == 0)

return 0;

return size \* 3;

}

}

}

# Інтерфейсний метод, що приймає спосіб обрахунку (тип фігури) та довжину сторони або діаметр.

using Strategy\_oodp\_3.Interfaces;

namespace Strategy\_oodp\_3.AreaСalculation

{

class Сalculation

{

public int GetValue(ISquarable calculator, int size)

{

return calculator.GetValue(size);

}

}

}

# Програм файл , що викликає метод для обрахунку периметру

using System;

using Strategy\_oodp\_3.AreaСalculation;

namespace Strategy\_oodp\_3

{

internal class Program

{

static void Main(string[] args)

{

var squarecalculator = new Сalculation();

Console.WriteLine("Площа правильного трикутника зi стороню 5 = " + squarecalculator.GetValue(new TriangleCalculator(), 5));

Console.WriteLine("Площа правильного квадрата зi стороню 5 = " + squarecalculator.GetValue(new SquareCalculator(), 5));

Console.WriteLine("Площа правильного трикутника зi стороню 5 = " + squarecalculator.GetValue(new TriangleCalculator(), 5));

Console.WriteLine("Площа правильного шестикутника зi стороню 5 = " + squarecalculator.GetValue(new HexagonCalculator(), 5));

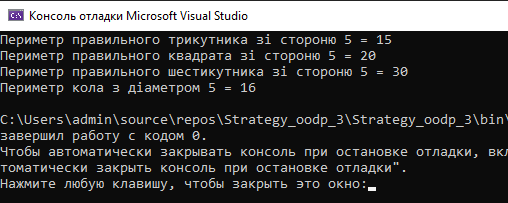
Console.WriteLine("Площа кола з дiаметром 5 = " + squarecalculator.GetValue(new CircleCalculator(), 5));

}

}

}

## Скриншоти Виконання



Площа кола була округлена до значення типу int

**Висновки:**

У лабораторній роботі номер 3 було розроблено власний проект так, щоб в коді можливо було застосувати один із розглянутих вище патернів (Стратегія, Спостерігач або Декоратор).

Та Окремо розроблено проект і застосувати в коді один із розглянутих вище патернів (Стратегія, Спостерігач або Декоратор) для обчислення периметра правильних багатокутників.