Національний технічний університет України «КПІ ім. І. Сікорського»

Факультет інформатики і обчислювальної техніки

Кафедра автоматизованих систем обробки інформації та управління

**Лабораторна робота №4**

**з дисципліни**

**«ОРПЗ .NET»**

на тему

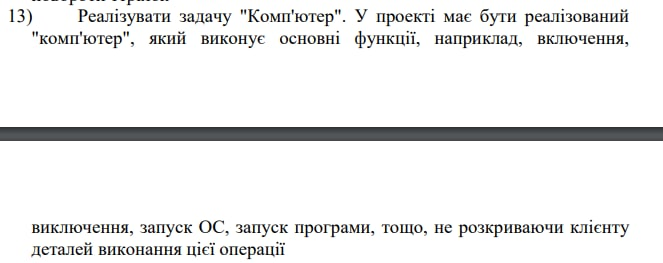
# «Шаблони проектування. Структурні шаблони»

Виконав

студент групи ІС-03

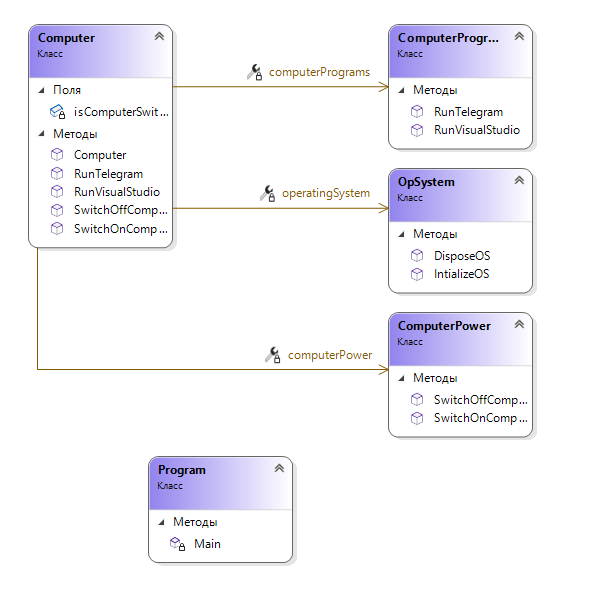
Овсепян Роман

Київ – 2022



Я обрав паттерн Фасад, оскільки він дозволяє зробити одну точку взаємодії клієнта та системою(в даному випадку, комп’ютер), Комп’ютер має багато складних операцій, про реалізацію яких клієнт не повинен знати. Фасад дозволяє розділити бізнес-логіку з реалізацією системних задач

Діаграма класів:



Код програми:

Facade:

using Lab\_4.Subsystems;

using System;

namespace Lab\_4.Facade

{

//FACADE

public class Computer

{

ComputerPower computerPower { get; set; }

OpSystem operatingSystem { get; set; }

ComputerPrograms computerPrograms { get; set; }

private bool isComputerSwithcedOn;

public Computer(ComputerPower computerPower, OpSystem operatingSystem, ComputerPrograms computerPrograms)

{

this.computerPower = computerPower;

this.operatingSystem = operatingSystem;

this.computerPrograms = computerPrograms;

isComputerSwithcedOn = false;

}

public void SwitchOnComputer()

{

if (isComputerSwithcedOn)

{

Console.WriteLine("Computer is already switched on");

return;

}

computerPower.SwitchOnComputer();

isComputerSwithcedOn = true;

operatingSystem.IntializeOS();

}

public void SwitchOffComputer()

{

if (!isComputerSwithcedOn)

{

Console.WriteLine("Computer is already switched off");

return;

}

operatingSystem.DisposeOS();

computerPower.SwitchOffComputer();

isComputerSwithcedOn = false;

}

public void RunVisualStudio()

{

computerPrograms.RunVisualStudio();

}

public void RunTelegram()

{

computerPrograms.RunTelegram();

}

}

}

Subsystems

using System;

using System.Threading;

namespace Lab\_4.Subsystems

{

public class ComputerPower

{

public void SwitchOnComputer()

{

Console.WriteLine("Loading...");

//... very complicated process for switching on the computer

Thread.Sleep(1000);

Console.WriteLine("Computer is switched on");

}

public void SwitchOffComputer()

{

Console.WriteLine("Waiting...");

//... very complicated process for switching off the computer

Thread.Sleep(1000);

Console.WriteLine("Computer is switched off");

}

}

public class OpSystem

{

public void IntializeOS()

{

Console.WriteLine("Waiting...");

//... very complicated process for initialazing OS

Thread.Sleep(1000);

Console.WriteLine("OS System is initialized");

}

public void DisposeOS()

{

Console.WriteLine("Waiting...");

//... very complicated process for disposing OS

Thread.Sleep(1000);

Console.WriteLine("OS System is disposed");

}

}

public class ComputerPrograms

{

public void RunVisualStudio()

{

Console.WriteLine("Visual studio is running");

}

public void RunTelegram()

{

Console.WriteLine("Telegram is running");

}

}

}

Client

using Lab\_4.Facade;

using Lab\_4.Subsystems;

using System;

namespace Lab\_4

{

public class Program

{

static void Main(string[] args)

{

Computer computer = new Computer(new ComputerPower(), new OpSystem(), new ComputerPrograms());

Console.WriteLine("Trying to switch on Computer");

computer.SwitchOnComputer();

Console.WriteLine();

Console.WriteLine("Trying to switch on Computer again");

computer.SwitchOnComputer();

Console.WriteLine();

Console.WriteLine("Trying to run Visual Studio");

computer.RunVisualStudio();

Console.WriteLine();

Console.WriteLine("Trying to run Telegram");

computer.RunTelegram();

Console.WriteLine();

Console.WriteLine("Trying to switch of Computer");

computer.SwitchOffComputer();

Console.WriteLine();

Console.WriteLine("Trying to switch of Computer again");

computer.SwitchOffComputer();

Console.WriteLine();

}

}

}

using Google.Cloud.Translation.V2;

using System.Text;

namespace lab\_4

{

public class Normalizer

{

public string FirstCapitalOtherNot(string text)

{

if (text == null || text.Length == 0)

throw new ArgumentNullException("Text is empty");

var specialSymbols = ".!?";

var textList = text.ToList();

for (int i = 0; i < textList.Count; i++)

{

if (i == 0 || (i >= 2 && specialSymbols.Contains(textList[i - 2])))

textList[i] = char.ToUpper(textList[i]);

else

{

textList[i] = char.ToLower(textList[i]);

}

}

string res = "";

textList.ForEach(x => res += x);

return res;

}

}

public class Translator

{

private TranslationClient \_client;

public Translator()

{

Setup();

\_client = TranslationClient.Create();

}

private static void Setup()

{

Console.OutputEncoding = Encoding.UTF8;

var path = "..\\..\\..\\credentials.json";

Environment.SetEnvironmentVariable("GOOGLE\_APPLICATION\_CREDENTIALS", path);

}

public string Translate(string translateToCode, string textTotrnaslate)

{

string result = "";

var detection = \_client.DetectLanguage(textTotrnaslate);

try

{

result = \_client.TranslateText(textTotrnaslate,

translateToCode,

detection.Language)

.TranslatedText;

}

catch (Exception ex)

{

Console.WriteLine("Failed to translate: ", ex.Message);

}

return result;

}

}

public class FileOperator

{

public string ReadFromFile(string path)

{

return File.ReadAllText(path);

}

public void WriteToFile(string path, string text)

{

File.WriteAllText(path, text);

}

}

public class Facade

{

private Translator \_translator;

private Normalizer \_normalizer;

private FileOperator \_fileOperator;

public Facade(FileOperator fileOperator, Translator translator, Normalizer normalizer)

{

\_fileOperator = fileOperator;

\_translator = translator;

\_normalizer = normalizer;

}

public string Translate(string translateToLang, string sourceFile = null, string sourceText = null)

{

var translatedText = "";

if (sourceFile == null && sourceText != null)

translatedText = \_translator.Translate(translateToLang, sourceText);

else if (sourceFile != null && sourceText == null)

{

sourceText = \_fileOperator.ReadFromFile(sourceFile);

translatedText = \_translator.Translate(translateToLang, sourceText);

}

var normalizedText = \_normalizer.FirstCapitalOtherNot(translatedText);

return normalizedText;

}

public void TranslateAndWrite(string translateToLang, string endfile,

string sourceFile = null,

string sourceText = null)

{

var translatedText = "";

var normalizedText = "";

if (sourceFile == null && sourceText != null)

{

translatedText = \_translator.Translate(translateToLang, sourceText);

normalizedText = \_normalizer.FirstCapitalOtherNot(translatedText);

\_fileOperator.WriteToFile(endfile, normalizedText);

}

else if (sourceFile != null && sourceText == null)

{

sourceText = \_fileOperator.ReadFromFile(sourceFile);

translatedText = \_translator.Translate(translateToLang, sourceText);

normalizedText = \_normalizer.FirstCapitalOtherNot(translatedText);

\_fileOperator.WriteToFile(endfile, normalizedText);

}

}

}

static class Program

{

private static readonly string SourcePath = @"..\\..\\..\\test.txt";

private static readonly string EndPath = @"..\\..\\..\\translated.txt";

private static readonly string EndPath2 = @"..\\..\\..\\translated2.txt";

private static readonly string SourceText = @"hi tHeRe! how aRe You? my naME is Maks. and whaT is your Name?";

public static void Main()

{

Facade facade = new(new FileOperator(), new Translator(), new Normalizer());

var german = facade.Translate(LanguageCodes.German, null, SourceText);

Console.WriteLine(german);

var polish = facade.Translate(LanguageCodes.Polish, null, german);

Console.WriteLine(polish);

facade.TranslateAndWrite(LanguageCodes.French, EndPath, null, polish);

facade.TranslateAndWrite(LanguageCodes.Polish, EndPath2, SourcePath, null);

}

}

}