МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра интеллектуальных информационных технологий

Отчет по лабораторной работе №6

По дисциплине «Современные платформы программирования»

Специальность ПО-8

Выполнил:

Серко А.С.

студент группы ПО-8

Проверил:

ст. преп. кафедры ИИТ,

«\_\_»\_\_\_\_\_\_\_\_\_\_\_2024 г.

**Цель работы:** приобрести навыки применения паттернов проектирования при решении практических задач с использованием языка C#

**Вариант 19**

• Прочитать задания, взятые из каждой группы.

• Определить паттерн проектирования, который может использоваться при реализации задания.

Пояснить свой выбор.

• Реализовать фрагмент программной системы, используя выбранный паттерн.

• Реализовать все

необходимые дополнительные классы.

**Задание 1**

Один из подходящих паттернов проектирования для данной задачи - это Паттерн Строитель (Builder). Этот паттерн позволяет создавать сложные объекты пошагово, разделяя процесс конструирования объекта от его представления, таким образом, что один и тот же процесс конструирования может создавать разные представления.

Применение этого паттерна в задаче с бургер-закусочной позволит нам управлять процессом создания заказа, разделяя его на отдельные шаги - добавление типа бургера, напитка, упаковки и т.д. Мы можем создать строителя заказа, который позволит нам последовательно добавлять каждый компонент заказа и в итоге получить готовый заказ с итоговой стоимостью.

Выполнение

**Код программы**

public abstract class MenuItem

{

public string Name { get; protected set; }

public decimal Price { get; protected set; }

public MenuItem(string name, decimal price)

{

Name = name;

Price = price;

}

}

public class Burger : MenuItem

{

public BurgerCategory Category { get; private set; }

public Burger(string name, decimal price, BurgerCategory category) : base(name, price)

{

Category = category;

}

}

public class Drink : MenuItem

{

public DrinkCategory Category { get; private set; }

public Drink(string name, decimal price, DrinkCategory category) : base(name, price)

{

Category = category;

}

}

public class Order

{

private List<MenuItem> items = new List<MenuItem>();

public void AddItem(MenuItem item)

{

items.Add(item);

}

public decimal CalculateTotal()

{

decimal total = 0;

foreach (var item in items)

{

total += item.Price;

}

return total;

}

}

public enum BurgerCategory

{

Vegan,

Meat

}

public enum DrinkCategory

{

Cold,

Hot

}

class Program

{

static void Main(string[] args)

{

Order order = new Order();

Burger veganBurger = new Burger("Black beans & Mushrooms Burger", 5.99m, BurgerCategory.Vegan);

Drink coldDrink = new Drink("Pepsi", 1.99m, DrinkCategory.Cold);

order.AddItem(veganBurger);

order.AddItem(coldDrink);

foreach (var item in order.items)

{

Console.WriteLine($"{item.Name} {item.Price}");

}

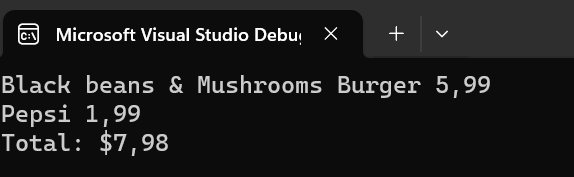
Console.WriteLine("Total: $" + order.CalculateTotal());

}

}

**Спецификация вывода**

**Пример**



**Задание 2.**

9) Проект «Часы». В проекте должен быть реализован класс, который дает возможность пользоваться часами со стрелками так же, как и цифровыми часами. В классе «Часы со стрелками» хранятся повороты стрелок.

Шаблон "Адаптер" позволяет преобразовать интерфейс одного класса в интерфейс другого класса, не изменяя его исходного кода. В случае проекта "Часы", где есть два типа часов - часы со стрелками и цифровые часы, мы можем использовать адаптер, чтобы обеспечить совместимость между ними.

Выполнение

**Код программы**

interface IDigitalClock

{

void DisplayTime(int hour, int minute);

}

class AnalogClock

{

private int hour;

private int minute;

public AnalogClock(int hour, int minute)

{

this.hour = hour;

this.minute = minute;

}

private double CalculateHourHandAngle()

{

return (hour % 12 + minute / 60.0) \* 360 / 12;

}

private double CalculateMinuteHandAngle()

{

return minute \* 360 / 60;

}

public void DisplayTime()

{

Console.WriteLine($"Analog clock shows the time: {hour}:{minute}");

Console.WriteLine($"Angle of the hour hand: {CalculateHourHandAngle()} degrees");

Console.WriteLine($"Angle of the minute hand: {CalculateMinuteHandAngle()} degrees");

}

}

class AnalogToDigitalAdapter : IDigitalClock

{

private AnalogClock analogClock;

public AnalogToDigitalAdapter(AnalogClock analogClock)

{

this.analogClock = analogClock;

}

public void DisplayTime(int hour, int minute)

{

analogClock.DisplayTime();

}

}

class Program

{

static void Main(string[] args)

{

AnalogClock analogClock = new AnalogClock(10, 30);

IDigitalClock adapter = new AnalogToDigitalAdapter(analogClock);

adapter.DisplayTime(10, 30);

}

}

**Спецификация ввода**

**Пример**

AnalogClock analogClock = new AnalogClock(10, 30);

**Спецификация вывода**

**Пример**

**Изображение выглядит как текст, Шрифт, снимок экрана, типография

Автоматически созданное описание**

**Задание 3**

9) Шифрование текстового файла. Реализовать класс-шифровщик текстового файла с поддержкой различных алгоритмов шифрования. Возможные варианты шифрования: удаление всех гласных букв из текста, изменение букв текста на буквы, получаемые фиксированным сдвигом из алфавита (например, шифром буквы а будет являться буква д для сдвига 4 и т.д.), применение операции исключающее или с заданным ключом.

Паттерн проектирования, который может быть полезен для реализации задачи шифрования текстового файла, - это стратегия (Strategy).

Стратегия позволяет определить семейство алгоритмов, инкапсулировать каждый из них и делать их взаимозаменяемыми. Таким образом, мы можем легко добавлять новые алгоритмы шифрования и изменять их независимо от кода, который использует эти алгоритмы.

В данном случае, каждый вариант шифрования (удаление гласных, сдвиг по алфавиту, операция XOR с ключом) может быть реализован как отдельная стратегия. Это позволит легко добавлять новые методы шифрования без изменения существующего кода.

Выполнение

**Код программы**

using System.Text;

using System.Text.RegularExpressions;

interface IEncryptionStrategy

{

string Encrypt(string text);

}

class RemoveVowelsEncryptionStrategy : IEncryptionStrategy

{

public string Encrypt(string text)

{

return Regex.Replace(text, "[aeiouAEIOU]", "");

}

}

class ShiftEncryptionStrategy : IEncryptionStrategy

{

private int shift;

public ShiftEncryptionStrategy(int shift)

{

this.shift = shift;

}

public string Encrypt(string text)

{

StringBuilder encryptedText = new StringBuilder();

foreach (char c in text)

{

if (char.IsLetter(c))

{

char shiftedChar = (char)(c + shift);

if ((char.IsLower(c) && shiftedChar > 'z') || (char.IsUpper(c) && shiftedChar > 'Z'))

{

shiftedChar = (char)(c - (26 - shift));

}

encryptedText.Append(shiftedChar);

}

else

{

encryptedText.Append(c);

}

}

return encryptedText.ToString();

}

}

class XOREncryptionStrategy : IEncryptionStrategy

{

private string key;

public XOREncryptionStrategy(string key)

{

this.key = key;

}

public string Encrypt(string text)

{

StringBuilder encryptedText = new StringBuilder();

for (int i = 0; i < text.Length; i++)

{

encryptedText.Append((char)(text[i] ^ key[i % key.Length]));

}

return encryptedText.ToString();

}

}

class TextFileEncryptor

{

private IEncryptionStrategy encryptionStrategy;

public TextFileEncryptor(IEncryptionStrategy encryptionStrategy)

{

this.encryptionStrategy = encryptionStrategy;

}

public void EncryptFile(string inputFile, string outputFile)

{

string text = File.ReadAllText(inputFile);

string encryptedText = encryptionStrategy.Encrypt(text);

File.WriteAllText(outputFile, encryptedText);

}

}

class Program

{

static void Main(string[] args)

{

IEncryptionStrategy strategy = new RemoveVowelsEncryptionStrategy();

TextFileEncryptor encryptor = new TextFileEncryptor(strategy);

encryptor.EncryptFile("input.txt", "output2.txt");

}

}

**Спецификация ввода**

**Пример**

**Input.txt**

I recently went to the grocery store and saw a sign that said 'Limit two per customer' on the toilet paper aisle. I guess people are getting ready for another round of panic buying. Who knew that the key to surviving a pandemic was having a stockpile of Charmin?

**Спецификация вывода**

**Пример**

ShiftEncryptionStrategy(4)

**output.txt**

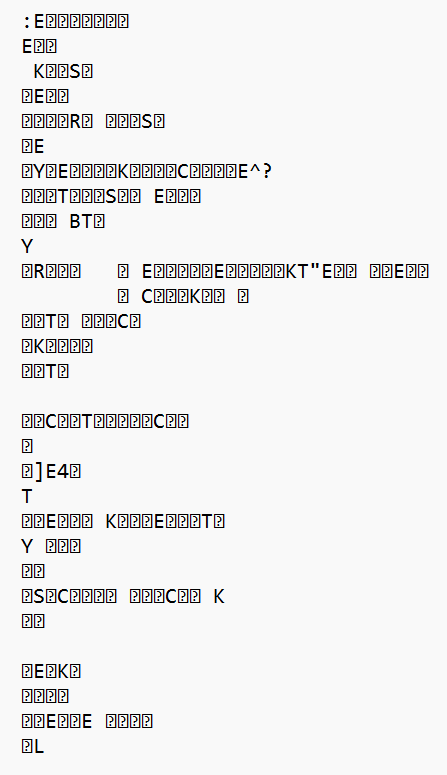
L uhfhqwob zhqw wr wkh jurfhub vwruh dqg vdz d vljq wkdw vdlg 'Olplw wzr shu fxvwrphu' rq wkh wrlohw sdshu dlvoh. L jxhvv shrsoh duh jhwwlqj uhdgb iru dqrwkhu urxqg ri sdqlf exblqj. Zkr nqhz wkdw wkh nhb wr vxuylylqj d sdqghplf zdv kdylqj d vwrfnsloh ri Fkduplq?

RemoveVowelsEncryptionStrategy

**output2.txt**

rcntly wnt t th grcry str nd sw sgn tht sd 'Lmt tw pr cstmr' n th tlt ppr sl. gss ppl r gttng rdy fr nthr rnd f pnc byng. Wh knw tht th ky t srvvng pndmc ws hvng stckpl f Chrmn?

**output2.txt**

****

**Вывод**

Приобрели навыки применения паттернов проектирования при решении практических задач с использованием языка C#.