***Кафедра комп’ютерної та програмної інженерії***

**ЛАБОРАТОРНА РОБОТА № 3**

**з курсу**

**«Архітектура програмної платформи .NET»**

**Студент: Бондарєва Є.Ю.**

**Група: № 42 ІПЗ**

**Факультет: Математики, інформатики**

**та фізики**

**Перевірив: Січкаренко В.О.**

Тема: Використання типів даних з загальної системи типів (CTS).

Мета: Ознайомитись з загальною системою типів (CTS). Застосування CTS в процесі розробки міжплатформного програмного забезпечення на платформі .NET.Загальні завдання

1. Ознайомитися з теоретичною частиною.

2. Обрати тему з переліку запропонованих або визначити власну;

3. Перейти в локальний репозиторій з лабораторними роботами;

4. Створити окрему гілку (git checkout -b) в системі контролю версій Git для поточної лабораторної роботи;

5. Створити новий консольний проект на мові програмування C#;

6. Реалізувати 3 класи (на вибір студента) з обраної теми;

7. Під час реалізації програми використовувати типи даних визначені в просторі імен System;

8. В функції Main продемонструвати використання створених класів;

9. Продемонструвати роботу програми;

10. Результат роботи програми зберегти у вигляді скріншоту (png або jpeg) в репозиторії з проектом;

11. Індексувати проект (git add);

12. Зафіксувати зміни (git commit);

13. Надіслати зміни у віддалений репозиторій (git push);

14. Створити та виконати запит на зміни (pull request);

15. Надіслати посилання на поточну лабораторну роботу у віддаленому репозиторії в GitHub.

**Хід роботи:**

Тема ообраної мною роботи: Текстовий редактор.

Для цієї лабораторної роботи було створено окрему гілку в Git – laba3 і новий консольний проект на мові програмування C#.

A computer screen with text

Description automatically generated

Рисунок 1 - Створення гілки і новий консольний проект на мові програмування C#

A screenshot of a computer

Description automatically generated

Рисунок 2 – Створена гілка і проект в провіднику в папці репозиторія

A screenshot of a computer

Description automatically generated

Рисунок 3 – Вікно GitHub з новою гілкою

Далі я відкриваю створений проект в середовищі Visual Studio для написання коду програми для реалізації 3 класів (Program, Document, TextEditor) з обраної теми «Текстовий редактор»; під час реалізації програми використовуватимемо типи даних визначені в просторі імен System; в функції Main продемонструємо використання створених класів. Рішення буде наведено в лістингу коду 1.

Лістинг коду 1:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Xml.Linq;

namespace ConsoleApp1

{

class Program

{

static void Main(string[] args)

{

// Створення об'єкта текстового редактора

TextEditor editor = new TextEditor();

bool exit = false;

while (!exit)

{

Console.OutputEncoding = System.Text.Encoding.Unicode;

Console.WriteLine("\nОберіть дію:");

Console.WriteLine("1. Створити новий документ");

Console.WriteLine("2. Додати текст до документа");

Console.WriteLine("3. Очистити документ");

Console.WriteLine("4. Показати вміст документа");

Console.WriteLine("5. Вийти");

string choice = Console.ReadLine();

switch (choice)

{

case "1":

Console.Write("Введіть назву документа: ");

string title = Console.ReadLine();

editor.CreateDocument(title);

break;

case "2":

if (editor.CurrentDocument != null)

{

Console.Write("Введіть текст для додавання: ");

string text = Console.ReadLine();

editor.AddTextToDocument(text);

}

else

{

Console.WriteLine("Документ ще не створено.");

}

break;

case "3":

editor.ClearDocument();

break;

case "4":

editor.DisplayDocument();

break;

case "5":

exit = true;

Console.WriteLine("Вихід з програми...");

break;

default:

Console.WriteLine("Невірний вибір. Спробуйте ще раз.");

break;

}

}

}

}

class Document

{

public string Title { get; set; }

public string Content { get; private set; }

public Document(string title)

{

Title = title;

Content = string.Empty;

}

public void AddText(string text)

{

Content += text + "\n";

}

public void ClearText()

{

Content = string.Empty;

}

public void Display()

{

Console.WriteLine($"--- {Title} ---");

Console.WriteLine(Content == string.Empty ? "Документ порожній" : Content);

}

}

class TextEditor

{

public Document CurrentDocument { get; private set; }

public void CreateDocument(string title)

{

CurrentDocument = new Document(title);

Console.WriteLine($"Документ '{title}' створено.");

}

public void AddTextToDocument(string text)

{

if (CurrentDocument != null)

{

CurrentDocument.AddText(text);

Console.WriteLine("Текст додано до документа.");

}

else

{

Console.WriteLine("Документ не створено.");

}

}

public void ClearDocument()

{

if (CurrentDocument != null)

{

CurrentDocument.ClearText();

Console.WriteLine("Документ очищено.");

}

else

{

Console.WriteLine("Документ не створено.");

}

}

public void DisplayDocument()

{

if (CurrentDocument != null)

{

CurrentDocument.Display();

}

else

{

Console.WriteLine("Документ не створено.");

}

}

}

}

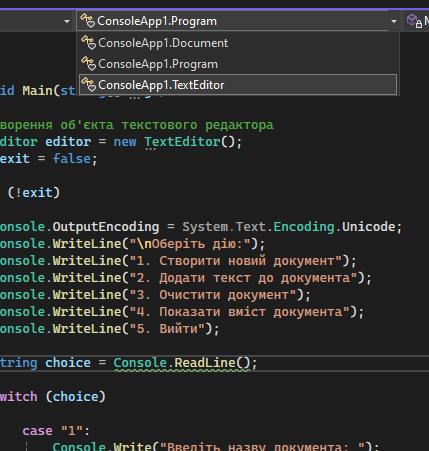


Рисунок 4 – Відображення створених класів у проекті в Visual Studio

A computer screen shot of a black screen

Description automatically generated

Рисунок 5 – Компіляція коду для виявлення помилок

Помилок виявлено не було результат виконання програми рис.9-10. Для фіксації проекту на гіті відкриємо його в Visual Studio Code і за допомогою терміналу зафіксуємо на гілці.

A screen shot of a computer program

Description automatically generated

Рисунок 6 – код в Visual Studio Code

Фіксуємо зміни в Git (рис.7).

A screenshot of a computer program

Description automatically generated

Рисунок 7 – Фіксація змін

Перегляд створеного проекту для перевірки в консольному застосуванні як повідомлялось на початку лабораторної роботи.

A screenshot of a computer program

Description automatically generated

Рисунок 8 – Демонстрація роботи пректу

Поєднуємо код з запиту на зміни з головною гілкою(рис.9).

A screenshot of a computer program

Description automatically generated

Рисунок 9 – Демонстрація роботи проекту

A screenshot of a computer

Description automatically generated

Рисунок 10 – Файли в репозитарії

Надіслати викладачу посилання на створений репозиторій в Git:

<https://github.com/Lizokbondarieva/labs/tree/laba3/laba3>

Висновки

В результаті виконання лабораторної роботи мною було створено нову гілку на репозиторій створено новий проект по темі «Текстовий редактор». В програмі наявні : 3 класи (Program, Document, TextEditor) з обраної теми ; під час реалізації програми використовували типи даних визначені в просторі імен System; в функції Main продемонстровано використання створених класів.

В ході виконання лабораторної роботи помилок та труднощів не виникло та було досягнуто поставленого результату.

Контрольні запитання

1. Що таке клас? Основні складові класу

Клас — це шаблон або модель, яка описує структуру та поведінку об'єктів певного типу в програмуванні. Клас визначає поля (дані) і методи (поведінку), які будуть доступні для об'єктів цього класу.

Основні складові класу:

- Поля (fields) — змінні, що зберігають стан об'єкта.

- Методи (methods) — функції або процедури, які визначають поведінку об'єкта.

- Властивості (properties) — забезпечують доступ до полів класу з можливістю перевірки або обмеження доступу.

- Конструктори (constructors) — спеціальні методи для ініціалізації об'єктів.

- Події (events) — механізми для реагування на певні дії.

- Деструктори (destructors) — методи, що викликаються при знищенні об'єктів.

2. Що таке CTS, CLS?

- CTS (Common Type System) — загальна система типів у .NET, яка визначає всі типи даних, що використовуються в різних мовах, які підтримують .NET. CTS гарантує, що типи даних можуть бути сумісними між різними мовами.

- CLS (Common Language Specification) — загальна специфікація мови, яка визначає набір правил, що гарантують сумісність між мовами .NET. CLS описує мінімальний набір функцій, які повинен підтримувати будь-який компілятор .NET.

3. Типи даних представлених в CTS

CTS класифікує типи даних на дві категорії:

1. Значимі типи (Value Types):

- Наприклад: `int`, `float`, `double`, `char`, `bool`, `struct`.

- Ці типи зберігають свої значення безпосередньо в змінній.

2. Посилальні типи (Reference Types):

- Наприклад: `class`, `interface`, `delegate`, `array`, `string`.

- Ці типи зберігають посилання на об'єкт, який міститься в пам'яті.

4. Типи значень та типи посилань. Особливості роботи з цими типами даних

- Значимі типи (Value Types):

- Зберігають значення безпосередньо в змінній.

- При передачі таких типів у метод створюється копія значення.

- Вони зберігаються в стеку (stack), що робить доступ до них швидшим.

- Наприклад: `int`, `double`, `bool`, `struct`.

- Посилальні типи (Reference Types):

- Зберігають посилання на об'єкт в пам'яті.

- При передачі цих типів у метод передається посилання на об'єкт, а не копія.

- Вони зберігаються в купі (heap), і доступ до них трохи повільніший.

- Наприклад: `class`, `string`, масиви.

5. Що таке «магічні числа» та як їх уникати?

Магічні числа — це числові значення, які використовуються в коді без пояснень їхнього значення або призначення. Вони ускладнюють читання та розуміння коду.

Як уникати магічних чисел:

- Використовувати константи або перерахування (enum) для надання імені числам.

- Наприклад, замість `if (age > 18)` краще використовувати `if (age > LEGAL\_AGE)`.

6. Оголошення та використання констант

Константи використовуються для збереження незмінних значень. Їх оголошують за допомогою ключового слова `const`.

```csharp

const double PI = 3.14159;

```

Константи не можуть змінювати своє значення після ініціалізації.

7. Тип даних описаний за допомогою struct

`struct` — це значимий тип, що використовується для зберігання невеликих наборів даних. Він не підтримує наслідування, але може реалізовувати інтерфейси.

```csharp

struct Point

{

public int X;

public int Y;

public Point(int x, int y)

{

X = x;

Y = y;

}

}

```

8. Тип даних описаний за допомогою class

`class` — це посилальний тип, що може включати поля, методи, властивості, конструктори та інші елементи. Класи підтримують наслідування і поліморфізм.

```csharp

class Person

{

public string Name;

public int Age;

public Person(string name, int age)

{

Name = name;

Age = age;

}

public void SayHello()

{

Console.WriteLine($"Hello, my name is {Name}.");

}

}

```

9. Різниця між struct та class

1. Тип даних:

- `struct` — це значимий тип.

- `class` — це посилальний тип.

2. Місце зберігання:

- `struct` зберігається в стеку (stack).

- `class` зберігається в купі (heap).

3. Наслідування:

- `struct` не підтримує наслідування, але може реалізовувати інтерфейси.

- `class` підтримує наслідування і поліморфізм.

4. Швидкодія:

- Операції з `struct` виконуються швидше, оскільки вони зберігаються в стеку і мають меншу вартість при копіюванні.

- `class` більш гнучкий, але працює трохи повільніше через використання купи.

10. Специфікатори доступу до класів, структур та їх частин

У C# є наступні специфікатори доступу:

1. public — відкритий доступ для будь-якого коду.

2. private — доступ тільки всередині того самого класу або структури.

3. protected — доступ всередині того самого класу та похідних класів.

4. internal — доступ тільки всередині поточної збірки.

5. protected internal — доступ всередині поточної збірки та похідних класів.

6. private protected — доступ всередині поточного класу або похідних класів у межах тієї ж збірки.

Наприклад:

```csharp

public class Car

{

private string model;

protected int year;

public string Make { get; set; }

public Car(string model, int year, string make)

{

this.model = model;

this.year = year;

Make = make;

}

}

```

**Використані джерела інформації:**

1. https://moodle.fmif.udu.edu.ua/mod/assign/view.php?id=31274