ЖИТОМИРСЬКА ПОЛІТЕХНІКА
ДЕРЖАВНИЙ УНІВЕРСИТЕТ

.NET Framework

Лекція 01

Організаційні питання

- *Лекції 32 години (16)*
- *Лабораторні роботи 48 годин (24)*
- *Використання GitLab*
- *Курсова робота*

Посилання на курс:
https://learn.ztu.edu.ua/course/view.php?id=1628

План

1. Парадигма програмування
2. Платформа .Net
3. Середовище Visual Studio
4. Огляд мови C#
5. Найпростіша програма

ПАРАДИГМА ПРОГРАМУВАННЯ

ПАРАДИГМА (дав.-гр. παράδειγμα, paradeigma - приклад, зразок) – це фундаментальний підхід до структурування та організації програмного коду. Це сукупність теоретичних основ, методів та принципів, які лежать в основі створення програм.

Об'єктно-орієнтоване програмування (ООП) — одна з парадигм програмування, яка розглядає програму як множину «об'єктів», що взаємодіють між собою.

ООП

Об'єктно-орієнтоване програмування (ООП) — одна з парадигм програмування, яка розглядає програму як множину об'єктів, що взаємодіють між собою.

Об'єкти можуть представляти реальні сутності (люди, автомобілі) або абстрактні поняття (числа, рахунки в банку).

Кожен об'єкт має свої ***властивості*** (атрибути) та ***поведінку*** (методи).

ООП парадигма з'явилась в 1960-тих роках і не мала широкого застосування до 1990-тих.

На сьогодні багато мов програмування (C#, C++, VB, Python, PHP, Ruby, Objective-C тощо) підтримують ООП.

ОСНОВНІ принципи ООП

Інкапсуляція. Об'єднує дані (властивості) та методи, що працюють з цими даними, в єдине ціле, що дозволяє приховати внутрішню реалізацію об'єкта від зовнішнього світу, забезпечуючи захист даних та спрощуючи використання об'єкта.

Успадкування. Дозволяє створювати нові класи (підкласи) на основі вже існуючих (базових класів). Підкласи успадковують властивості та методи базових класів і можуть додавати свої власні. Це забезпечує повторне використання коду та створення ієрархії класів.

Поліморфізм. Дозволяє об'єктам різних типів відповідати на один і той же виклик методу по-різному. Це досягається за допомогою перевизначення методів у підкласах, що робить код більш гнучким і адаптивним.

ПЕРЕВАГИ ООП

1. Модульність і повторне використання коду. ООП дозволяє розбити програму на маленькі, незалежні блоки, які можна використовувати багато разів.

2. Гнучкість і масштабованість. ООП дозволяє створювати масштабовані системи, які можуть легко розширюватися і розвиватися.

3. Простота супроводу. Інтуїтивно зрозумілий інтерфейс. Кожен об'єкт містить необхідну інформацію, забезпечуючи швидкий пошук та усунення помилок. Це значно спрощує роботу як для розробників, так і для тестувальників.

4. Безпека. Інкапсуляція коду підвищує рівень безпеки програмного забезпечення.

5. Поліпшення продуктивності. Оптимізація коду = підвищення швидкодії.

6. Легка інтеграція. ООП дозволяє швидко інтегрувати різні компоненти програми, створюючи об'єкти, які взаємодіють один з одним. Це спрощує розробку складних систем.

НЕДОЛІКИ ООП

1. Складніша крива навчання. Перехід на ООП потребує більше часу та зусиль для поглибленого розуміння.

2. Можливість дублювання коду і надмірності. Неправильне проектування класів і об'єктів може призвести до дублювання коду, що виявляється у повторному використанні ідентичних або дуже схожих фрагментів коду в різних частинах системи. Це створює надлишковість і ускладнює підтримку програмного продукту.

3. Перевантаження пам’яті та обробки даних. Хоча об'єктний підхід надає великі можливості для створення складних і гнучких програм, він вимагає додаткових ресурсів пам'яті. Це пов'язано з тим, що класи і об'єкти містять не тільки дані, але й асоційовані з ними функціональні можливості.

.NET FRAMEWORK

ПЛАТФОРМА - це середовище виконання програм, яка визначає особливості розробки та виконання програмного коду – парадигми програмування, мови програмування, тощо.

.NET Framework програмна технологія, запропонована фірмою Microsoft як платформа для створення як звичайних програм, так і веб-застосунків

.NET Framework- платформа містить компоненти: the common language runtime загальномовне середовище виконання (CLR) and the .NET Framework Class Library бібліотека класів (.NET FCL).

.NET FRAMEWORK

CLR - це набір служб, необхідних для виконання збірки. При цьому програмний код збірки може бути як керованим (код, під час виконання якого CLR, активізує систему управління пам'яттю), так і некерованим («старий» програмний код).

Загальна система типів CTS (Common Type System) описує всі можливі типи даних та всі програмні конструкції, що підтримуються CLR. Крім того, у CTS показано, як ці сутності можуть взаємодіяти один з одним, і як вони представлені у форматі метаданих .NET

CLS (Common Language Specification) - загальна специфікація мов програмування. Мови, що відповідають CLS (Visual C#, Visual Basic, Visual C++), можуть інтегруватися одна з одною. CLS - це основа міжмовної взаємодії в рамках платформи .

FCL (.NET Framework Class Library) - відповідна CLS специфікації об'єктно-орієнтована бібліотека класів, інтерфейсів і системи типів (типів-значень), які включаються до складу платформи Microsoft .NET. Ця бібліотека забезпечує доступ до функціональних можливостей системи і призначена в якості основи при розробці .NET додатків, компонент, елементів керування.

ЗАГАЛЬНА СТРУКТУРА ПЛАТФОРМИ .NET FRAMEWORK

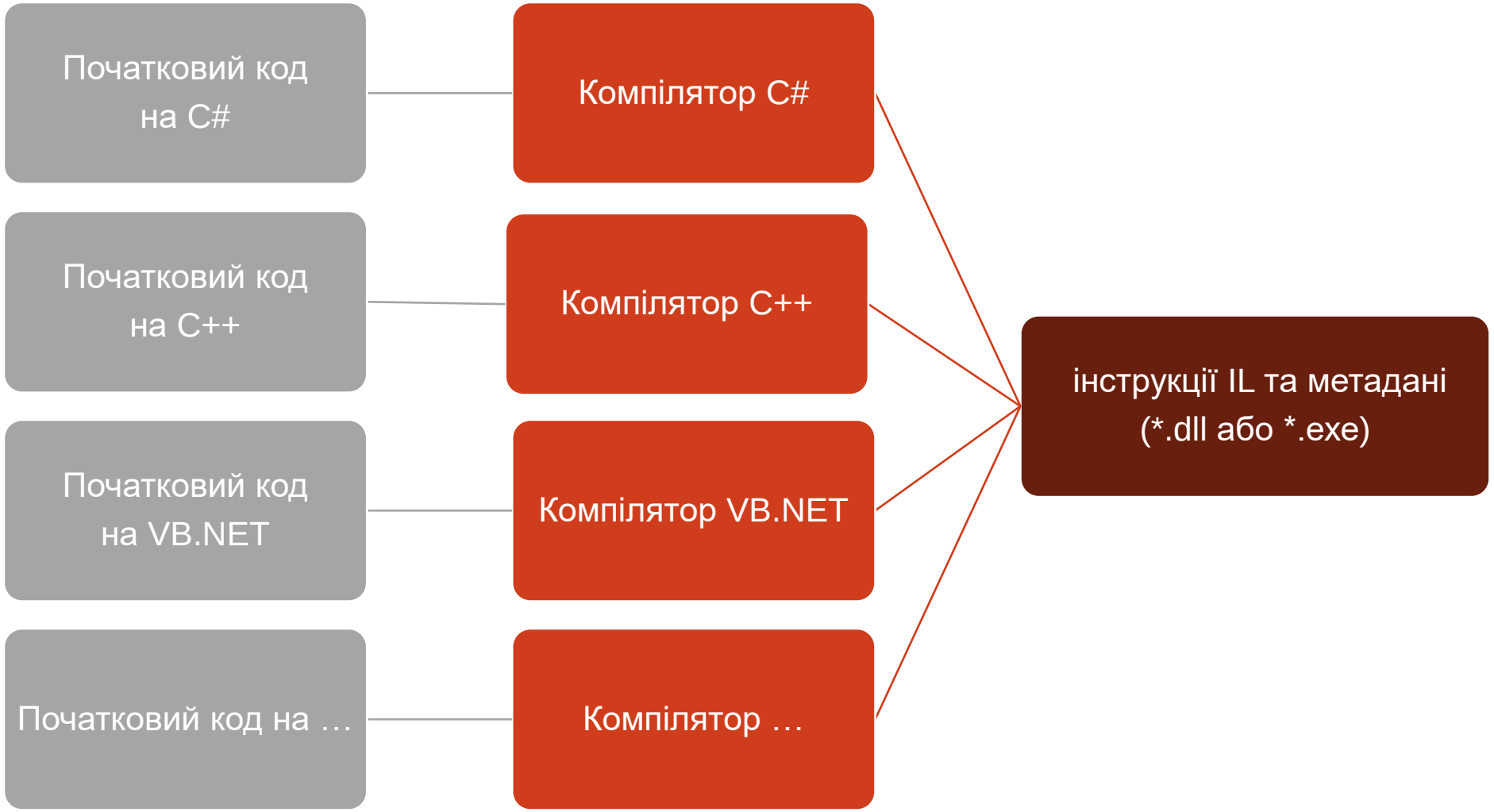
Бібліотека базових класів

- ASP.NET
- ASP.NET Web Forms
- ASP.NET MVC
- Windows Forms
- Windows Presentation Foundation

Загальномовне середовище виконання (CLR)

- Загальна система типів (CTS)
- Загальномовна специфікація (CLS)

ЗАГАЛЬНОМОВНЕ СЕРЕДОВИЩЕ ВИКОНАННЯ CLR



ЗАГАЛЬНА СХЕМА ВИКОНАННЯ ПРОГРАМИ У .NET

Бібліотека класів платформи .NET

Простір імен - це спосіб організації системи типів у єдину групу.

Існує загальномовна бібліотека базових класів. І концепція простору імен забезпечує ефективну організацію та навігацію в цій бібліотеці. Незалежно від мови програмування доступ до певних класів забезпечується завдяки їхньому угрупованню в межах загальних просторів імен.

Бібліотека класів Framework Class Library (FCL) містить понад 7000 типів (класів, структур, інтерфейсів, перелічених типів і делегатів), які поділено між "просторами імен", кожен з яких відповідає за служби з певної області.

Бібліотека класів платформи .NET

Простір імен System – це базовий простір, який містить класи для обробки виняткових ситуацій, типів даних, забезпечення введення/виведення даних, збирання сміття тощо.

Доступ до простору імен можливий через директиву:

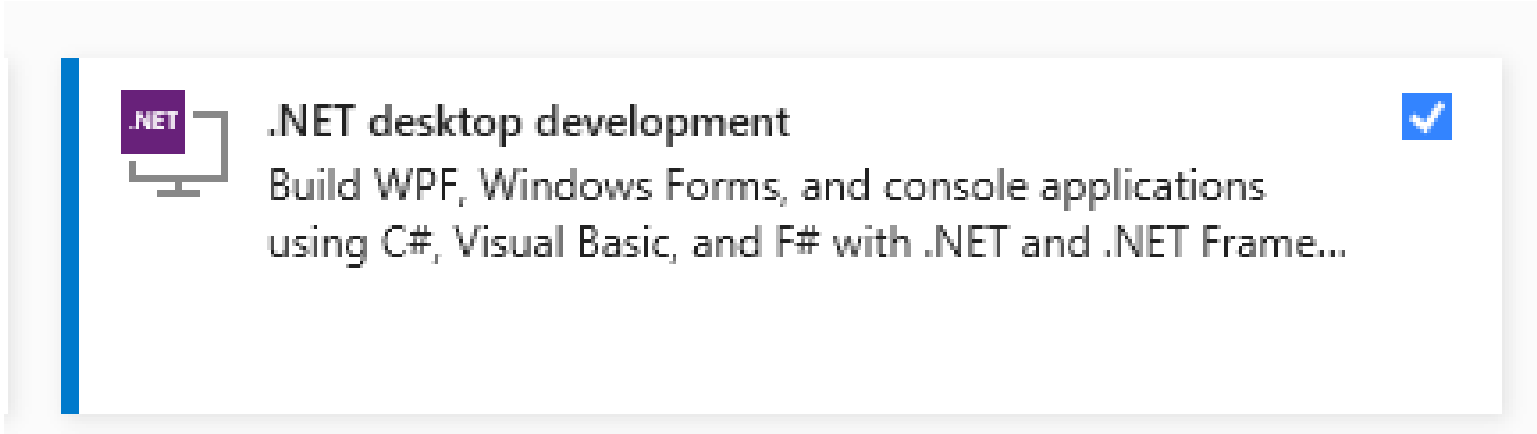
using [aliasname =] namespace.element

де: *aliasname* – ідентифікатор, за допомогою якого можна посилатися усередині модуля на зазначений простір імен; *namespace* – назва імпортованого простору; *element* – назва елемента простору (клас, простір назв тощо).

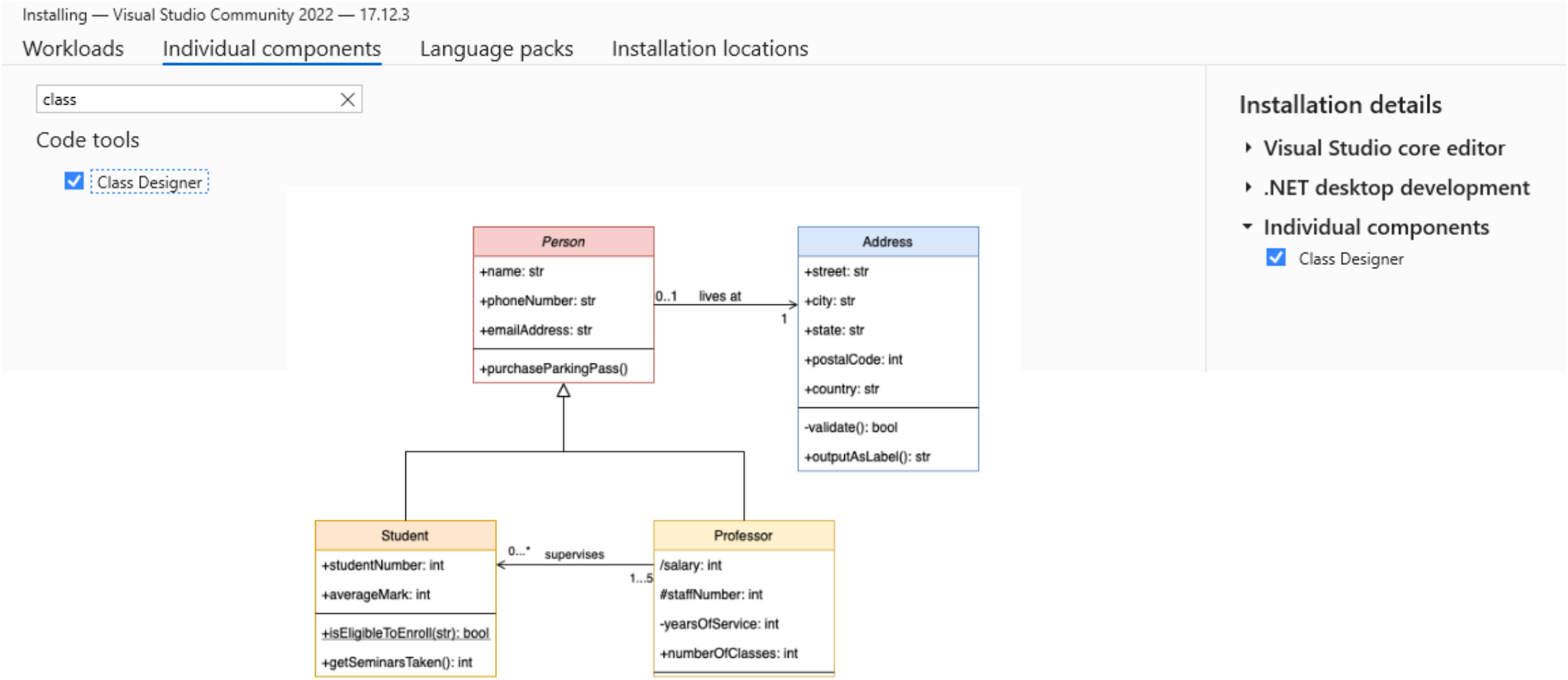
Простір імен .NET

Простір імен	Опис
System	Містить основні типи і класи
System.Collections	Кешовані таблиці, динамічні масиви та інші контейнери
System.Data	Класи ADO.NET для роботи з базами даних
System.Drawing	Класи для генерування графіки (GDI+)
System.IO	Класи для введення/виведення у файли і потоки
System.NET	Класи, що є програмною реалізацією мережевих протоколів
System.Reflection	Класи для читання/запису метаданих
System.ServiceProcess	Класи для створення NET-сервісів
System.Threading	Класи для створення і керування процесами
System.Web	Класи для підтримки
System.Web.Ser-vices	Класи для розробки веб-сервісів
System.Web.Services.Protocols	Класи для розробки клієнтів веб-сервісів
System.Web.UI	Основні класи технології ASP.NET
System.Web.UI.WebControls	Серверні контролери ASP.NET
System.Windows.Forms	Класи для GUI-програм
System.Xml	Класи для читання і запису даних у форматі XML

Встановлення



Встановлення



C#

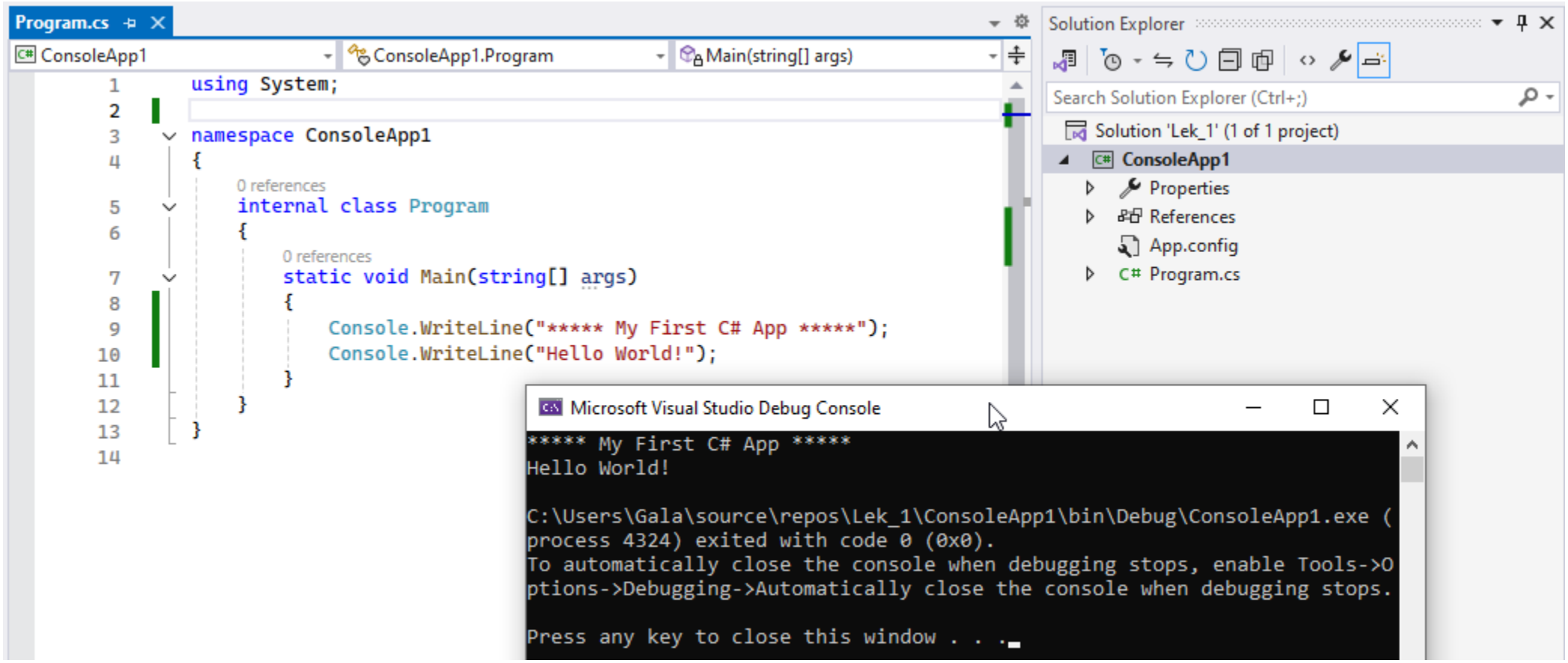
C# – сучасна універсальна високорівнева об'єктно-орієнтована мова програмування, створена у 1998-2001 рр. компанією Microsoft як засіб розробки додатків для платформи Microsoft .NET Framework.

C# відноситься до сімейства мов з С-подібним синтаксисом. На його розробників значний вплив зробили мови Java та C++

Підтримує такі можливості як:

- інкапсуляцію;
- успадкування;
- поліморфізм;
- перевантаження операторів;
- статичну типізацію;
- ітератори;
- коментарі у форматі XML;
- тощо.

C# найпростіша програма



Приклад

Оператори **using** завжди розміщуються на початку файлу програми, перед усіма іншими операторами і дозволяє вказати простір імен до якого будемо звертатися тобто звертатися до класів цього простору.

```
using System;

class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("***** My First C# App *****");
        Console.WriteLine("Hello World!");
        Console.WriteLine();
        Console.ReadLine();
    }
}
```

Допоміжна література

C# Підручник

https://w3schoolsua.github.io/cs/index.html#gsc.tab=0

C# language documentation

https://learn.microsoft.com/uk-ua/dotnet/csharp/