Cairo University
Faculty of Engineering
Computer Engineering Department

Database Management Systems
Lab1 – Introduction to C#

By / Eng Mohamed Sayed

Agenda Items

- Introduction (.NET framework)
- C# overview
- Windows form overview (GUI)

Introduction (.NET framework)

What is framework ?

A framework is like **a ready-made toolkit** for building software; it offers **pre-built components**, **rules**, and guidance to make the development process **faster** and more **organized**.

We use framwork to:

- Save time
- Ensure consistency
- Leverage best practices

Introduction (.NET framework)

What is .NET framework ?

.NET Framework is a software development framework developed by **Microsoft** for building and running applications **on Windows.**

To find additional details regarding the clarification of .NET versions, including .NET, .NET Core, .NET Standard, .NET Framework, and additional information, check this.

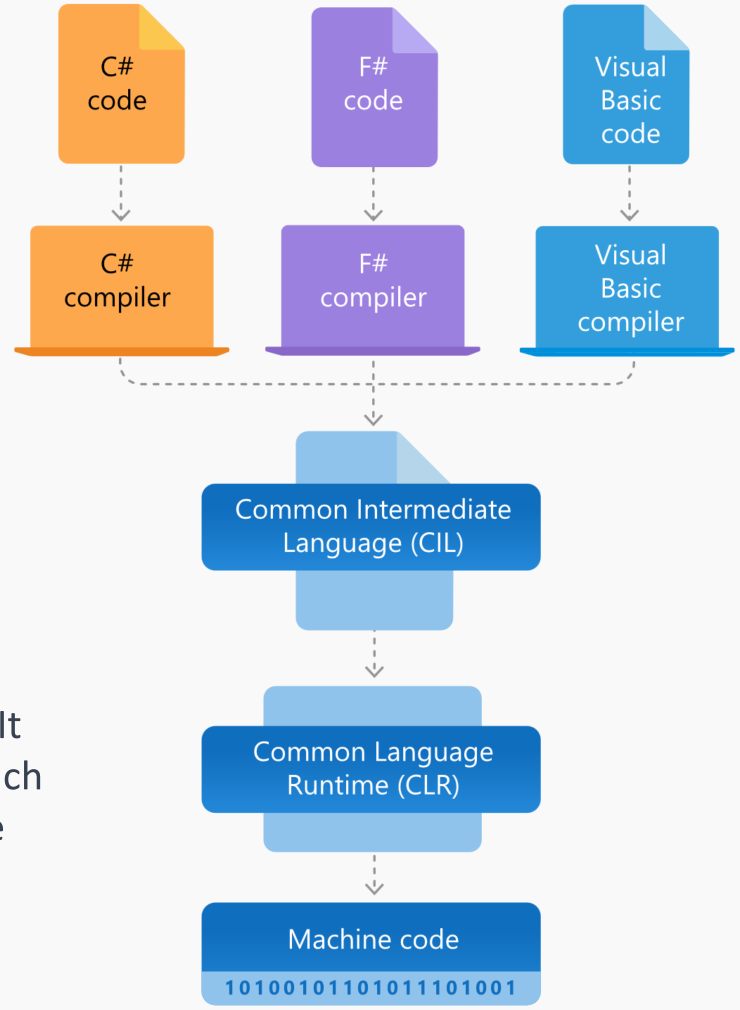
For additional details regarding the version history of .NET check this.

Introduction (.NET framework)

Architecture of .NET Framework

- **Framework Class Library (FCL):** collection of reusable classes, interfaces.

- **Common Language Runtime (CLR)** : provides many services to .NET applications, including memory management and exception handling. It also provides Just-In-Time (JIT) compilation, which compiles the CIL code into machine code on the fly as the program runs.



C# overview

It **is object-oriented** programming language developed by **Microsoft**. It was first introduced in 2000 as part of .NET framework.

Some key features and aspects of C#:

Object-Oriented: it encourages the use of objects and classes to structure code.

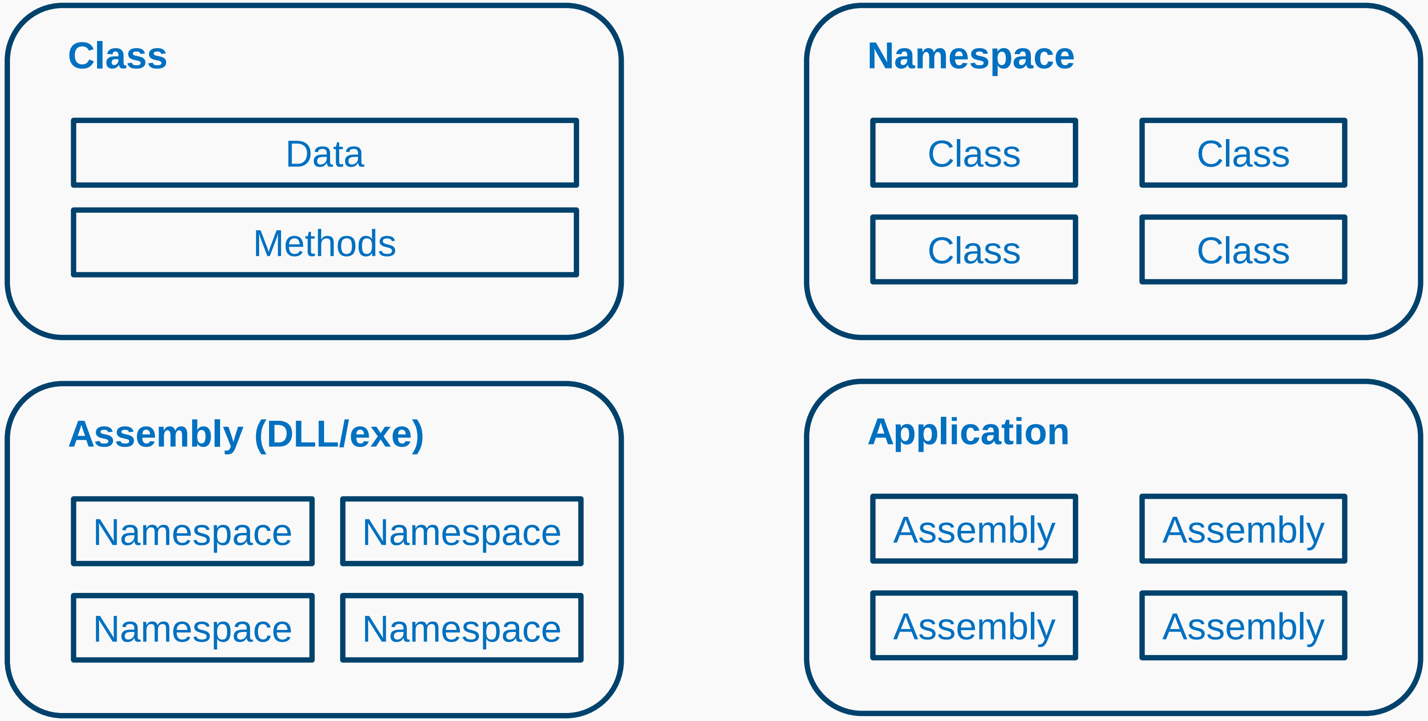
Type-Safe: it is a statically typed language, which means that variable types are checked at compile-time.

Garbage Collection: it features automatic memory management via a garbage collector for efficient memory handling.

Used in Various Domains: it is used for various types of applications, including desktop applications, web applications (via ASP.NET), mobile app development (via Xamarin), game development (with Unity).

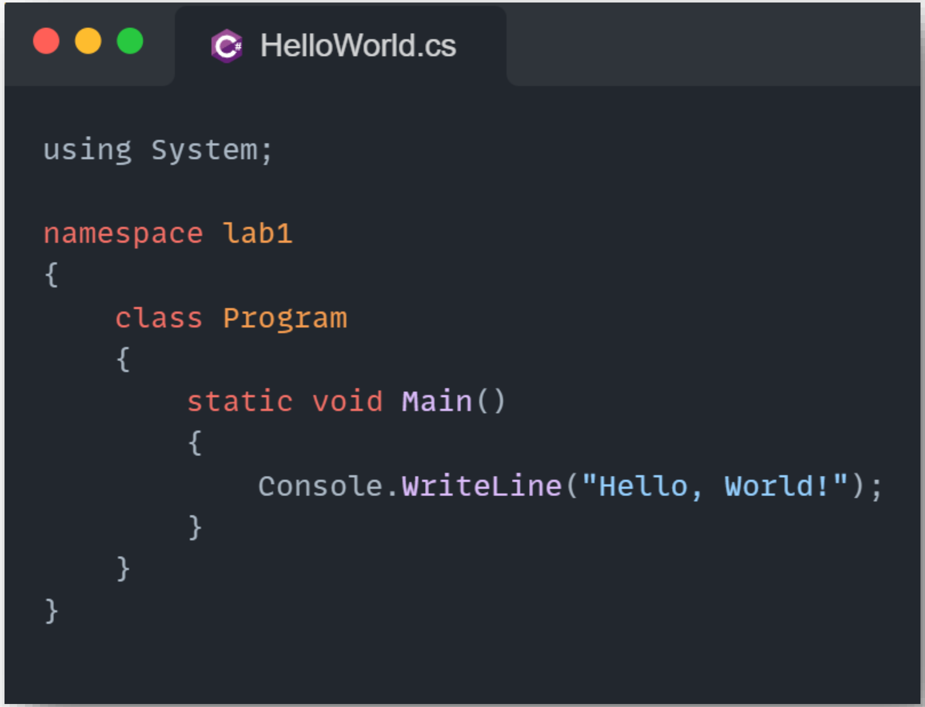
C# overview

Architecture of C # application



C# overview

HelloWorld program


HelloWorld.cs

```
using System;

namespace lab1
{
    class Program
    {
        static void Main()
        {
            Console.WriteLine("Hello, World!");
        }
    }
}
```

C# overview (Data Types)

Value Types

Numeric Types

Int, short, long, byte, float, decimal, double.

Boolean Type
bool

Character Type
char

Enumeration Type
enum

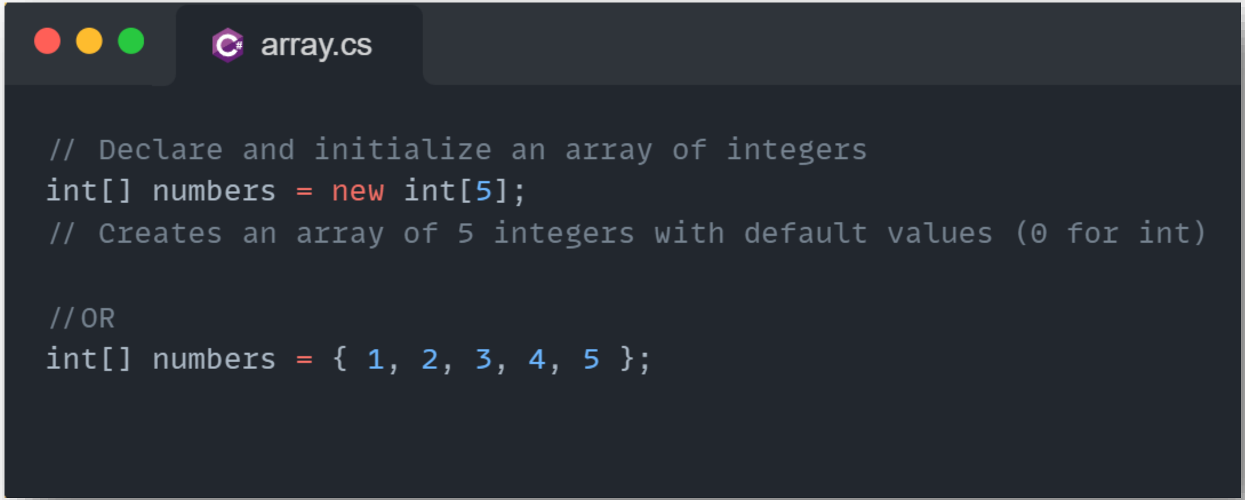
Nullable Type
Int?

Structure Type
struct

C# overview (Data Types)

Reference Types (Array)

Single-Dimensional Array:



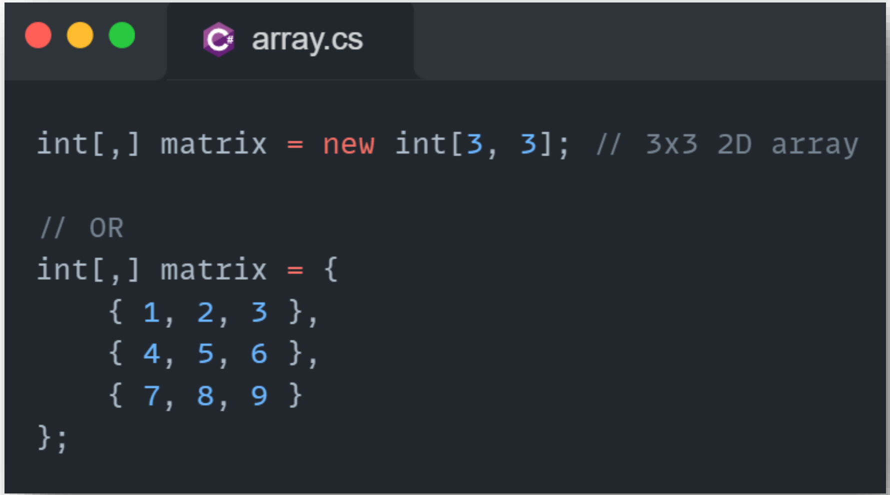
array.cs

```
// Declare and initialize an array of integers
int[] numbers = new int[5];
// Creates an array of 5 integers with default values (0 for int)

//OR
int[] numbers = { 1, 2, 3, 4, 5 };
```

C# overview (Data Types)

Reference Types (Array)

Multi-Dimensional Array:

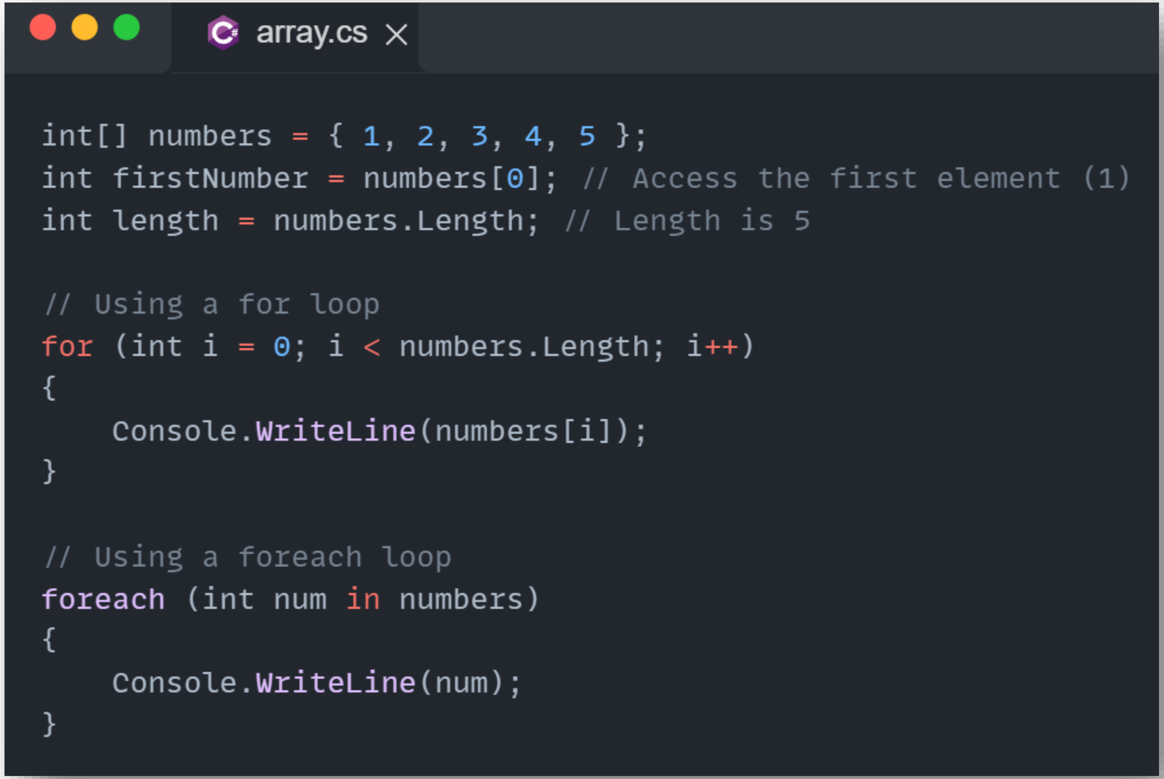


```
int[,] matrix = new int[3, 3]; // 3x3 2D array

// OR
int[,] matrix = {
    { 1, 2, 3 },
    { 4, 5, 6 },
    { 7, 8, 9 }
};
```

C# overview (Data Types)

Reference Types (Array)

Access Array (Single):

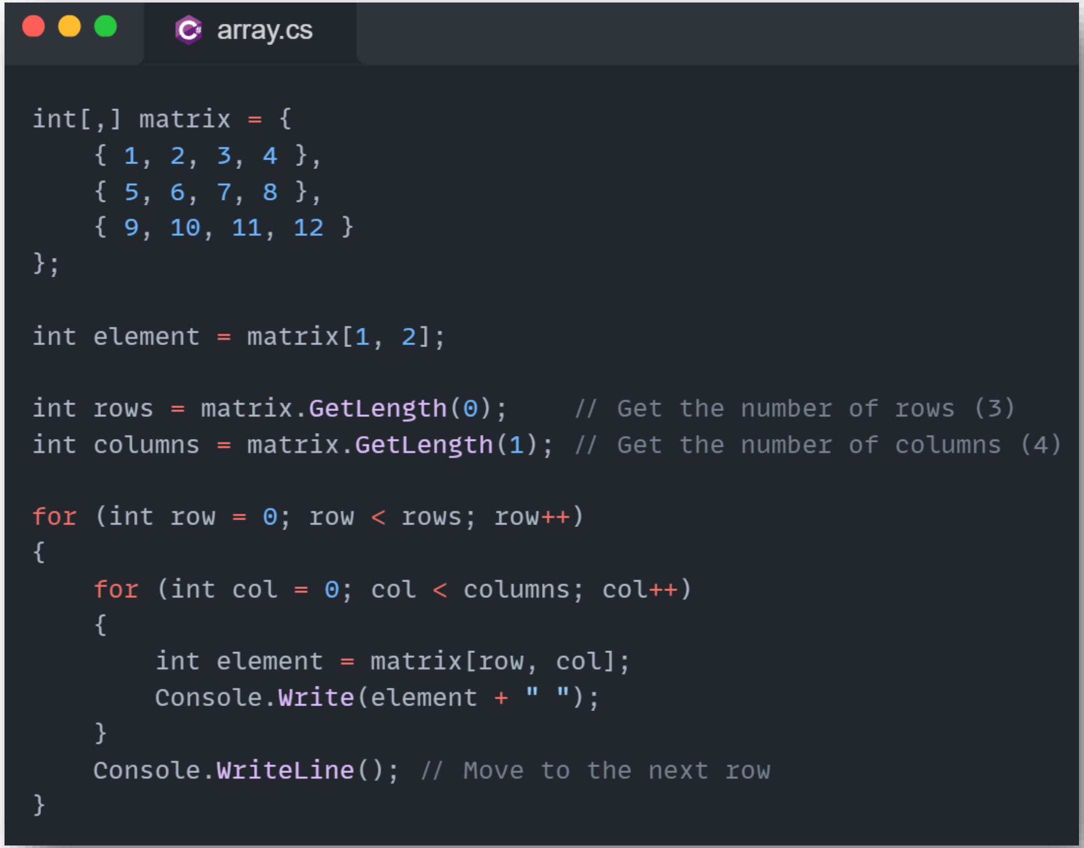

array.cs

```
int[] numbers = { 1, 2, 3, 4, 5 };
int firstNumber = numbers[0]; // Access the first element (1)
int length = numbers.Length; // Length is 5

// Using a for loop
for (int i = 0; i < numbers.Length; i++)
{
    Console.WriteLine(numbers[i]);
}

// Using a foreach loop
foreach (int num in numbers)
{
    Console.WriteLine(num);
}
```

C# overview (Data Types)

Reference Types (Array)

Access Array (Multi):

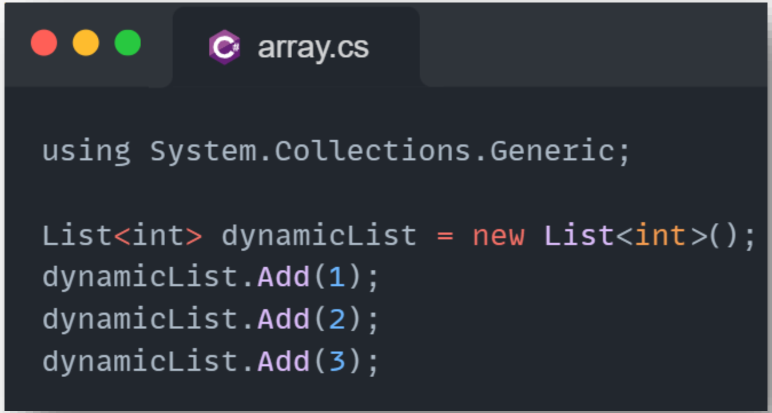

array.cs

```
int[,] matrix = {
    { 1, 2, 3, 4 },
    { 5, 6, 7, 8 },
    { 9, 10, 11, 12 }
};

int element = matrix[1, 2];

int rows = matrix.GetLength(0);    // Get the number of rows (3)
int columns = matrix.GetLength(1); // Get the number of columns (4)

for (int row = 0; row < rows; row++)
{
    for (int col = 0; col < columns; col++)
    {
        int element = matrix[row, col];
        Console.Write(element + " ");
    }
    Console.WriteLine(); // Move to the next row
}
```

C# overview (Data Types)

Reference Types (Array)

Dynamic Array:



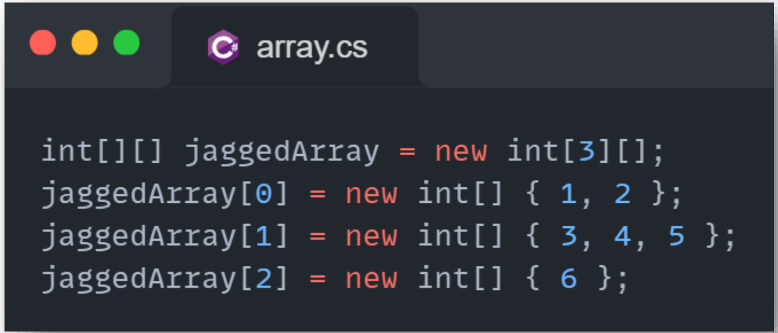
```
using System.Collections.Generic;

List<int> dynamicList = new List<int>();
dynamicList.Add(1);
dynamicList.Add(2);
dynamicList.Add(3);
```

C# overview (Data Types)

Reference Types (Array)

Jagged Arrays:

In addition to rectangular 2D arrays, C# also supports jagged arrays. A jagged array is an **array of arrays**, where each "sub-array" can have a different length.



```
int[][] jaggedArray = new int[3][];
jaggedArray[0] = new int[] { 1, 2 };
jaggedArray[1] = new int[] { 3, 4, 5 };
jaggedArray[2] = new int[] { 6 };
```

C# overview (Data Types)

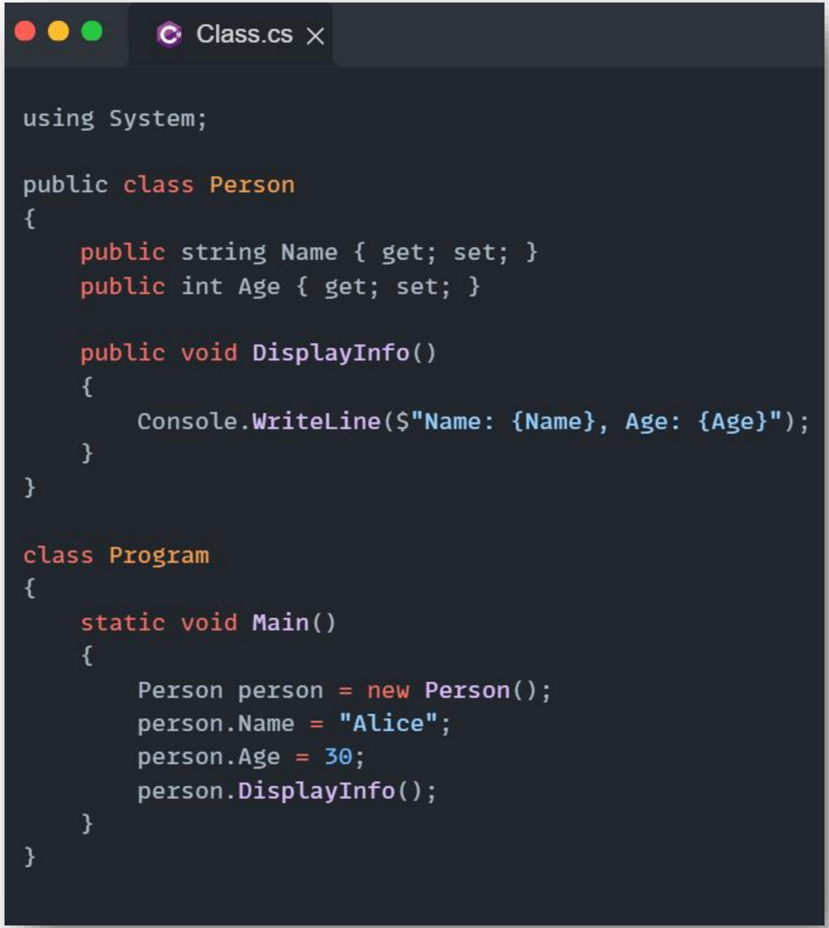
Reference Types (Class)

- Default access modifiers for the class members is **private**.

- Class objects must be created using **new** operator.

- A Class can only inherit **one class**, but can itself be inherited by **many classes**.

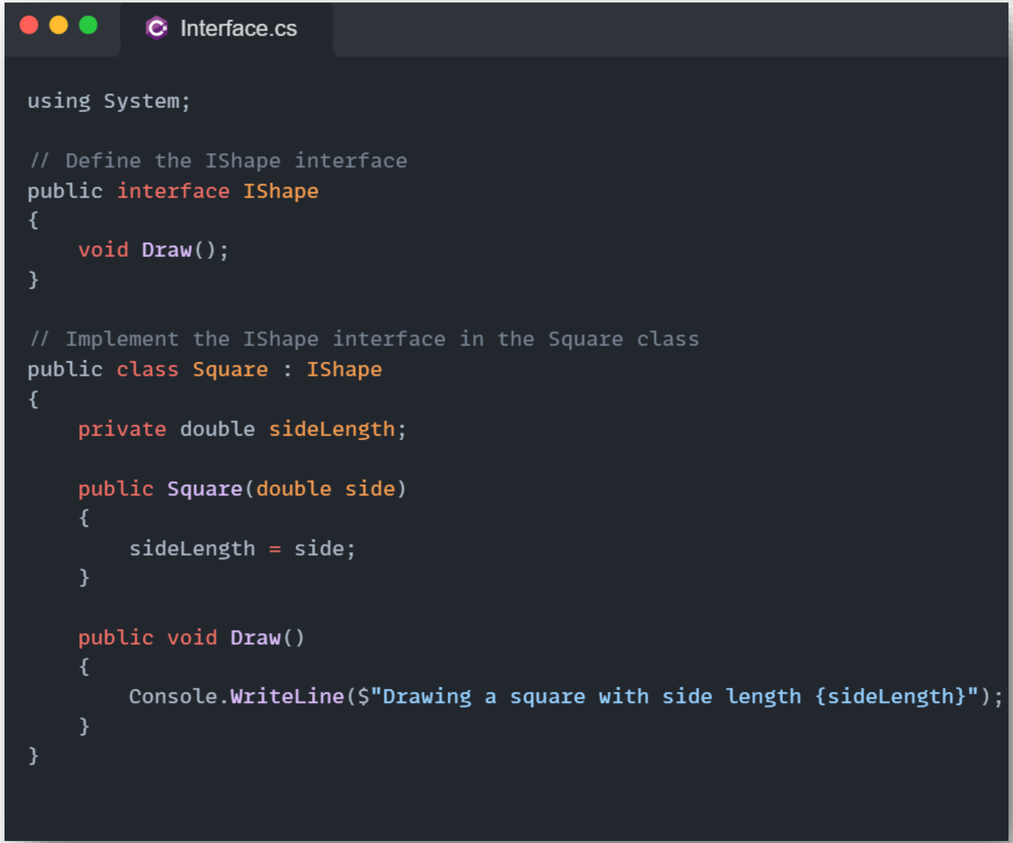
- C# allows **public** inheritance only.



```
using System;

public class Person
{
    public string Name { get; set; }
    public int Age { get; set; }

    public void DisplayInfo()
    {
        Console.WriteLine($"Name: {Name}, Age: {Age}");
    }
}

class Program
{
    static void Main()
    {
        Person person = new Person();
        person.Name = "Alice";
        person.Age = 30;
        person.DisplayInfo();
    }
}
```

C# overview (Data Types)

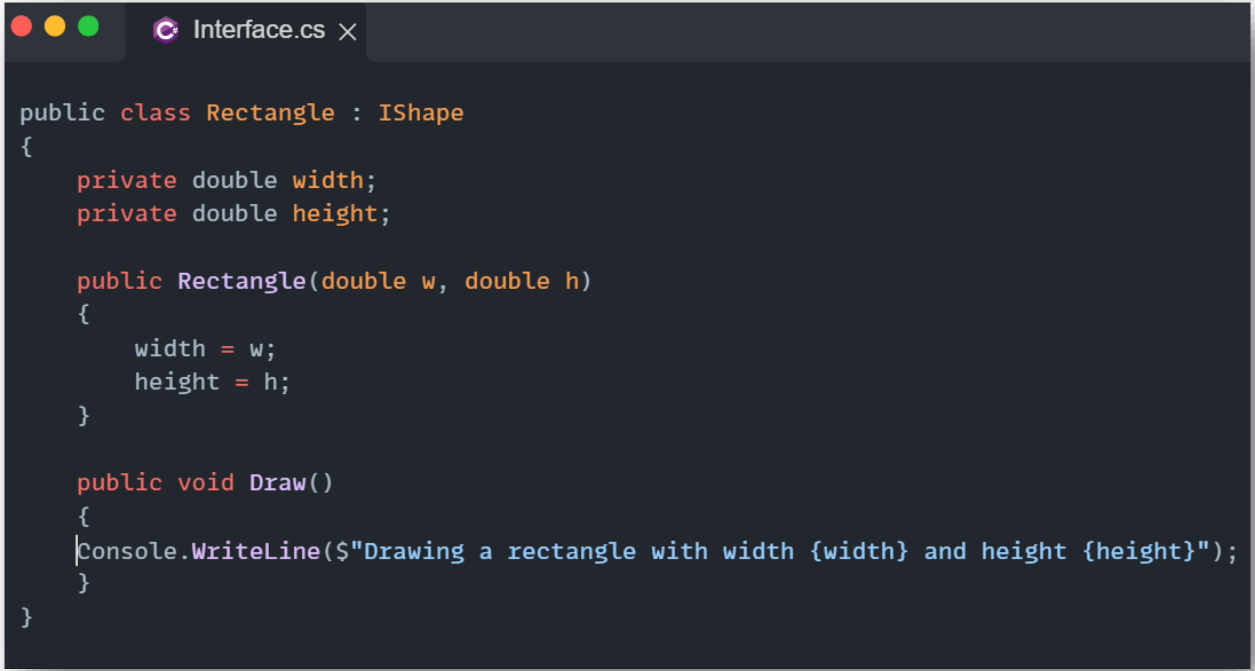
Reference Types (Interface)



```
using System;

// Define the IShape interface
public interface IShape
{
    void Draw();
}

// Implement the IShape interface in the Square class
public class Square : IShape
{
    private double sideLength;

    public Square(double side)
    {
        sideLength = side;
    }

    public void Draw()
    {
        Console.WriteLine($"Drawing a square with side length {sideLength}");
    }
}
```

C# overview (Data Types)

Reference Types (Interface)



Interface.cs

```
public class Rectangle : IShape
{
    private double width;
    private double height;

    public Rectangle(double w, double h)
    {
        width = w;
        height = h;
    }

    public void Draw()
    {
    Console.WriteLine($"Drawing a rectangle with width {width} and height {height}");
    }
}
```

C# overview (Data Types)

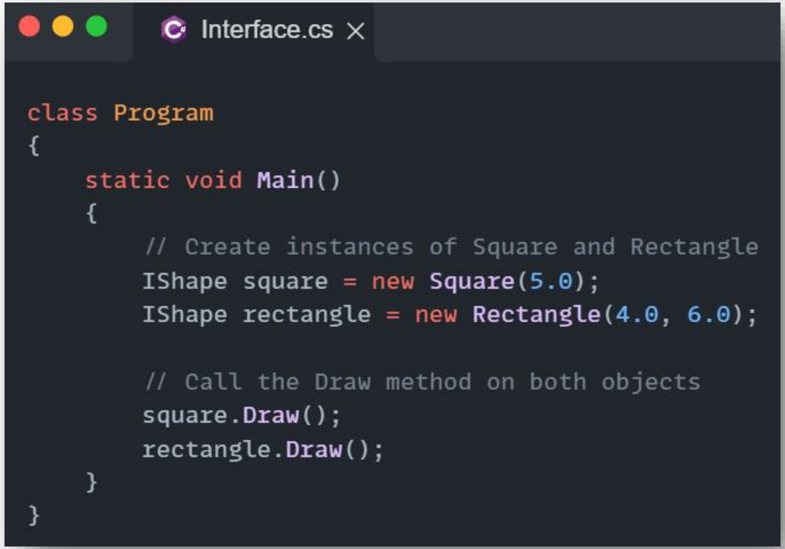
Reference Types (Interface)

- An interface contains only the signatures of methods without implementation.

- A class can implement multiple interfaces.

- A class that implements the interface must implement all the members of the interface specified in the interface definition

- Interfaces may **derive from** other interfaces



```
class Program
{
    static void Main()
    {
        // Create instances of Square and Rectangle
        IShape square = new Square(5.0);
        IShape rectangle = new Rectangle(4.0, 6.0);

        // Call the Draw method on both objects
        square.Draw();
        rectangle.Draw();
    }
}
```

Looping & Conditional Statements

- While, do while, for, if, else → all are the same as C++

- forEach was mentioned in array

Console I/O

To write something to console

```
Console.WriteLine(“Hello”);
```

To read something to console

```
Console.ReadLine();
```

Windows form overview (GUI)

GUI (Event-based) Programming

- A C# program's user interface typically consists of one or more forms, each containing multiple controls.

- **Forms** and **controls** provide events that can be handled.

- Common events include actions like **clicking** a mouse button, **typing** a character on the keyboard, or **selecting** an item from a combobox.

- There are three essential components for GUI programming, which are:
- Forms
- Controls
- Events

Windows form overview (GUI)

GUI Programming (forms)

- Forms are inherited from **System.Windows.Forms.Form** class

- A form represents a window that contains controls for user input and displaying outputs

- You can create them either using a GUI or through code (typically, the GUI approach is more common)

➢Using GUI:

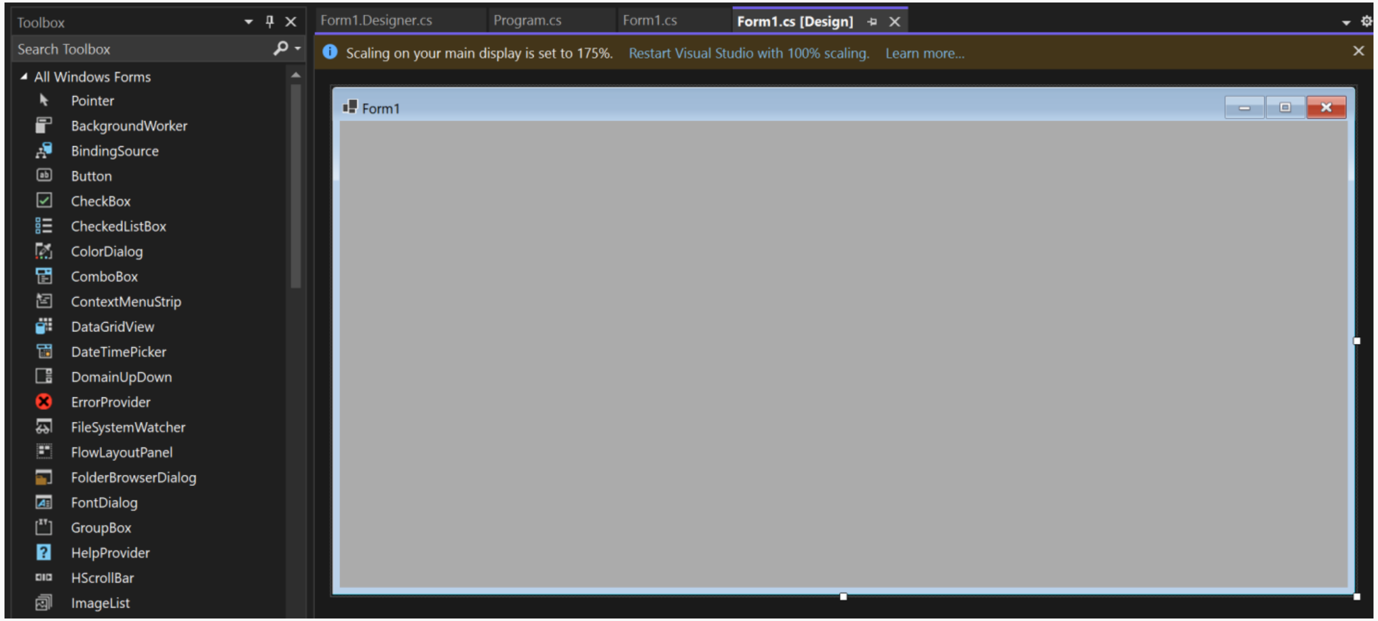
In the Solution Explorer, right-click the solution, select 'Add,' choose 'New Item,' and then pick 'Windows Form.

➢Using Code:

```
Form f = new Form(); // create a new form
f.Show();          //to show the form
```

Windows form overview (GUI)

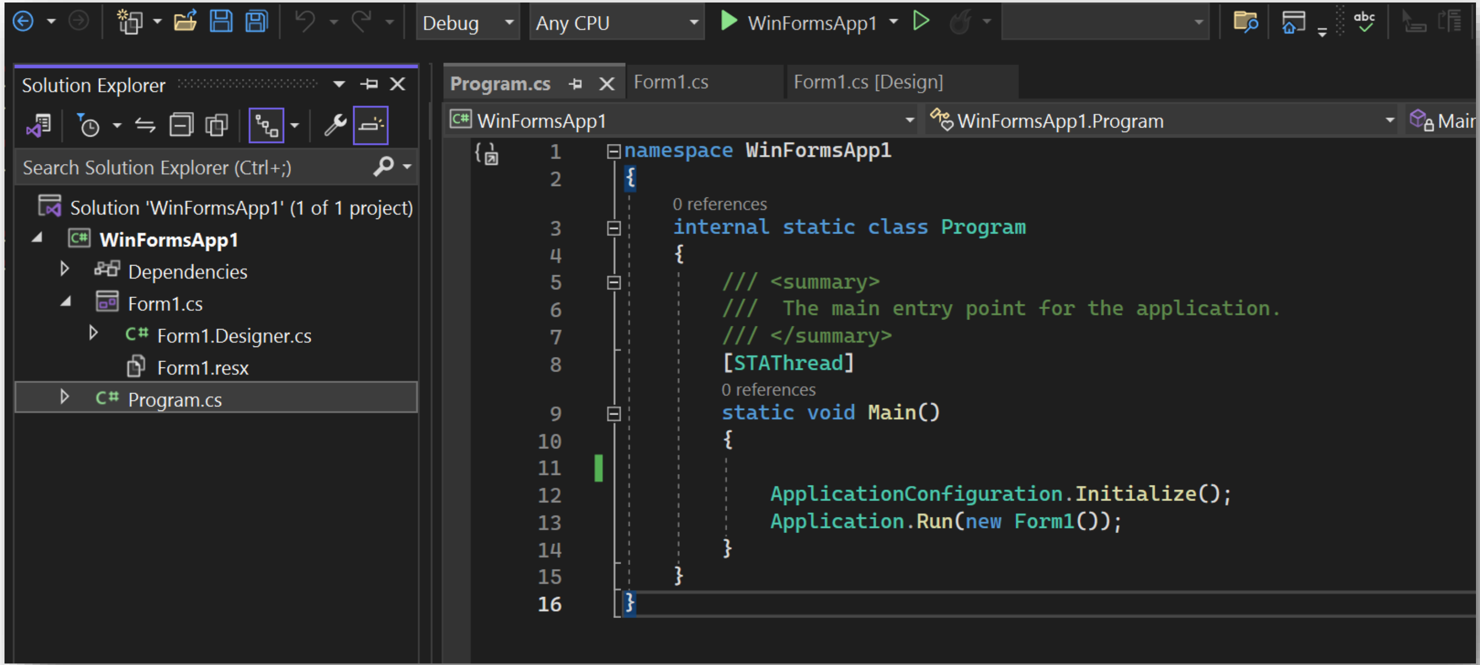
GUI Programming (forms)



Scaling on your main display is set to 175%. Restart Visual Studio with 100% scaling. Learn more...

Windows form overview (GUI)

GUI Programming (forms)



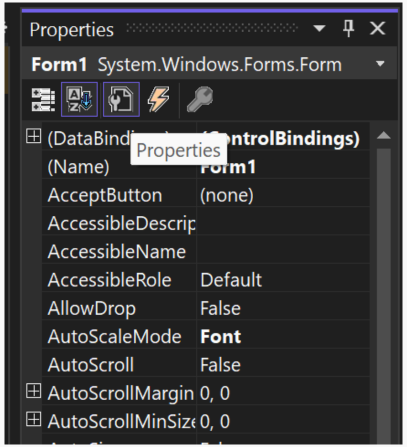
```
namespace WinFormsApp1
{
    0 references
    internal static class Program
    {
        /// <summary>
        ///  The main entry point for the application.
        /// </summary>
        [STAThread]
        0 references
        static void Main()
        {

            ApplicationConfiguration.Initialize();
            Application.Run(new Form1());
        }
    }
}
```

Windows form overview (GUI)

GUI Programming (forms)

Properties:

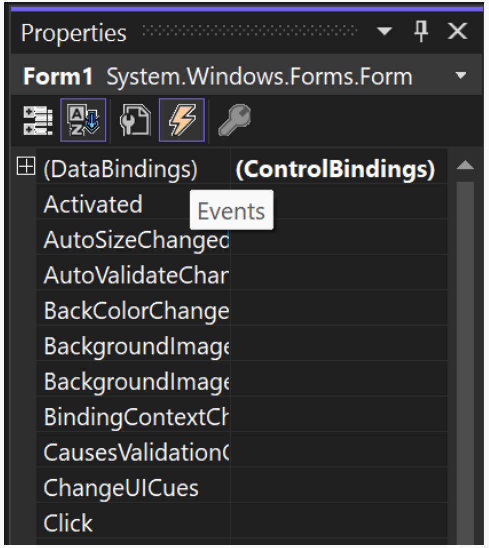
- Name, FormBorderStyle, Text, BackColor, BackImage,...etc


Properties
Form1 System.Windows.Forms.Form

(DataBindings)	(ControlBindings)
(Name)	Form1
AcceptButton	(none)
AccessibleDescrip	
AccessibleName	
AccessibleRole	Default
AllowDrop	False
AutoScaleMode	Font
AutoScroll	False
AutoScrollMargin	0, 0
AutoScrollMinSize	0, 0

Events:

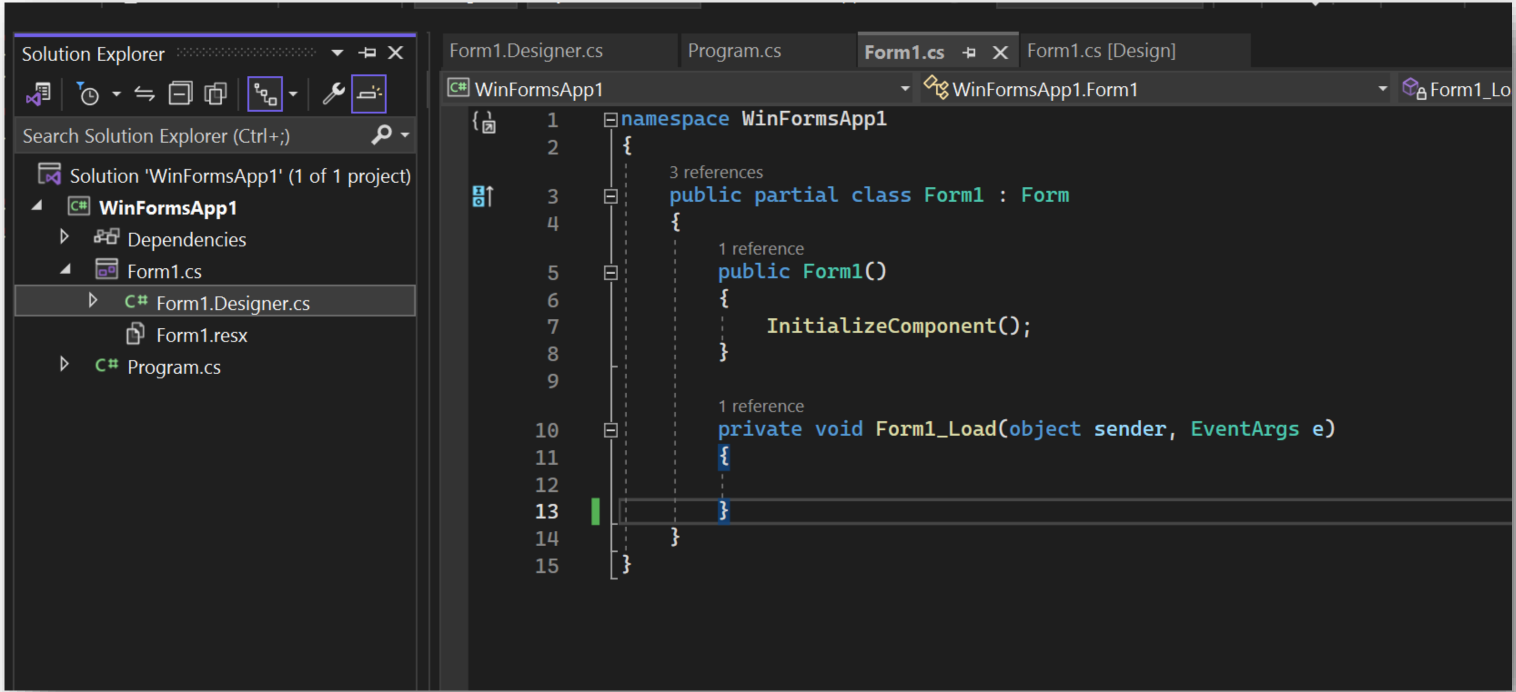
- Load, FormClosing, FormClosed
- GotFocus, LostFocus
- MouseHover, Click, DoubleCLick


Properties
Form1 System.Windows.Forms.Form

(DataBindings)	(ControlBindings)
Activated	
AutoSizeChanged	
AutoValidateChar	
BackColorChange	
BackgroundImage	
BackgroundImage	
BindingContextCh	
CausesValidation(	
ChangeUICues	
Click	

Windows form overview (GUI)

GUI Programming (forms)



```
namespace WinFormsApp1
{
    3 references
    public partial class Form1 : Form
    {
        1 reference
        public Form1()
        {
            InitializeComponent();
        }

        1 reference
        private void Form1_Load(object sender, EventArgs e)
        {

        }
    }
}
```

Windows form overview (GUI)

GUI Programming (forms)

- You can set form properties in constructor.
- Initialization code can be set also in the **Form_Load** event.
- To make a form a start up form for your application pass an object of the form to **Application.Run** :

Example:

- Application.Run(new Form1()); when you run the application.

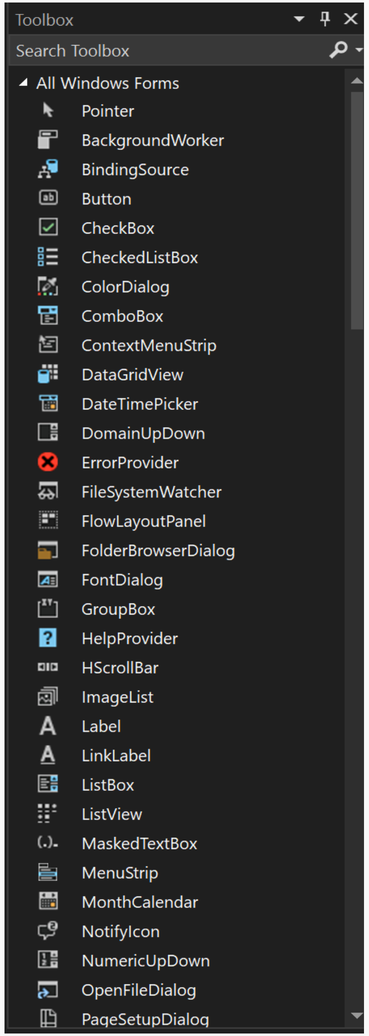
Windows form overview (GUI)

GUI Programming (controls)

A control is a **component** that provides user-interface (UI) capabilities.

Examples:

- TextBox
- Label
- Button
- DataGrid
- RadioButton
- CheckBox
- ComoBox
- …etc



Toolbox

Search Toolbox

All Windows Forms
- Pointer
- BackgroundWorker
- BindingSource
- Button
- CheckBox
- CheckedListBox
- ColorDialog
- ComboBox
- ContextMenuStrip
- DataGridView
- DateTimePicker
- DomainUpDown
- ErrorProvider
- FileSystemWatcher
- FlowLayoutPanel
- FolderBrowserDialog
- FontDialog
- GroupBox
- HelpProvider
- HScrollBar
- ImageList
- Label
- LinkLabel
- ListBox
- ListView
- MaskedTextBox
- MenuStrip
- MonthCalendar
- NotifyIcon
- NumericUpDown
- OpenFileDialog
- PageSetupDialog

Windows form overview (GUI)

GUI Programming (controls)

How to add control to form?

1. Make sure that the toolbox is visible (you can show it from view menu).
2. From toolbox window, drag a button.
3. From properties window, edit some of them.

Windows form overview (GUI)

GUI Programming (Events)

- An event is an **action** that can be **detected** and acted upon in code, such as responding to a **button click** or text change

- In event-driven programming:

- User actions are translated into **events**.
- These events are then passed to the application for processing.

- Each form and control provides a predefined set of events.

Windows form overview (GUI)

GUI Programming (Events)

How to add event to control?

- Pick the control on which you wish attach an event.
- Select the properties menu.
- Click on event button. ⚡
- Select the event and double-click it.

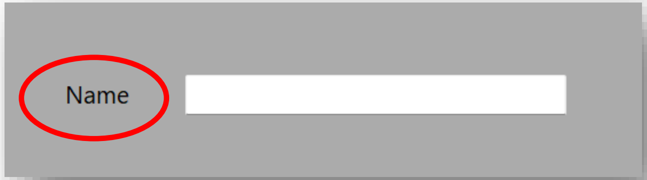
Windows form overview (GUI)

GUI Programming (control) examples:

Label

Usage:
- For displaying of text used to label other things on the form

Properties:
- Text
- Font
- ..etc



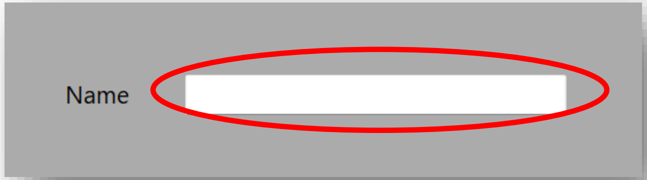
Windows form overview (GUI)

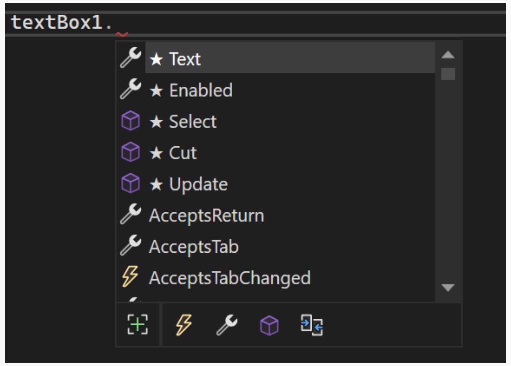
GUI Programming (control) examples:

TextBox

Usage:
- To accept user input

Properties:
- Text
- Enabled
- ..etc





```
textBox1.
```

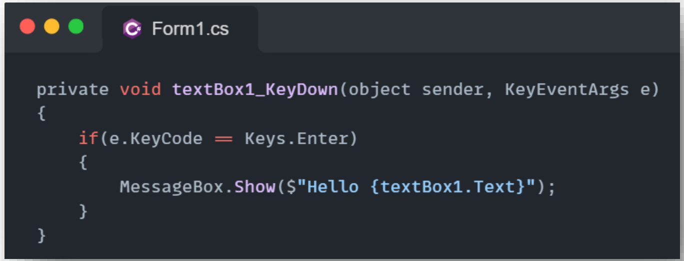
Windows form overview (GUI)

GUI Programming (control) examples:

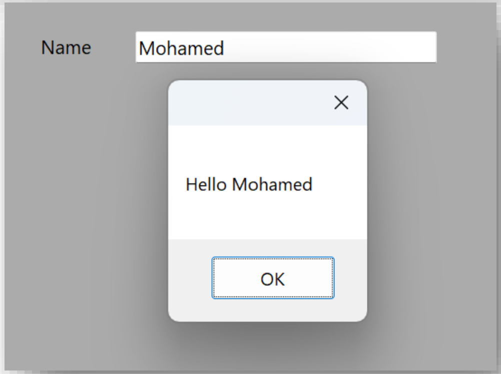
TextBox (example)

1. Create TextBox to accept user’s name
2. Add Label to the input field
3. Attach KeyDown event so when user press

Enter this message appear “hello+ {name of user}”



```
private void textBox1_KeyDown(object sender, KeyEventArgs e)
{
    if(e.KeyCode == Keys.Enter)
    {
        MessageBox.Show($"Hello {textBox1.Text}");
    }
}
```



Hello Mohamed

Windows form overview (GUI)

GUI Programming (control) examples:

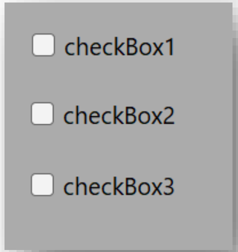
CheckBox

Usage:

- To allow user to choose multiple selections between number of options

Properties:

- Text
- Checked (true/false)
- ..etc



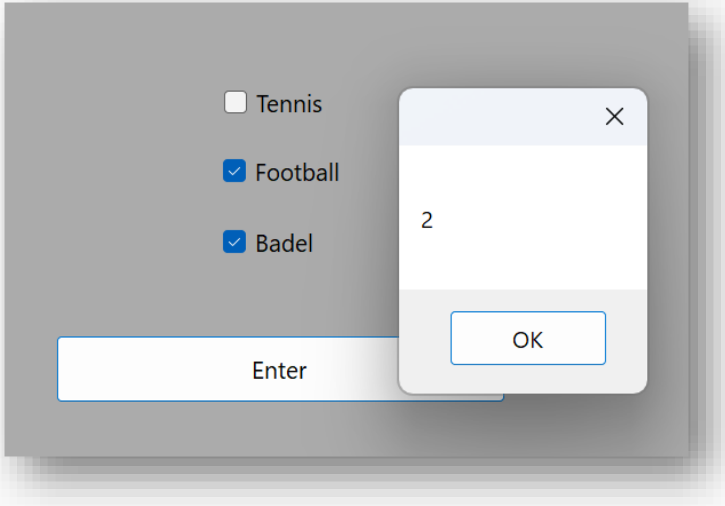
Windows form overview (GUI)

GUI Programming (control) examples:

CheckBox (example)

- Design a simple form asking the user about his/her favorite sports.

- When the user clicks Enter button, you should write the count of sports the user prefers according to the number of checkboxes he/she marked in the textbox



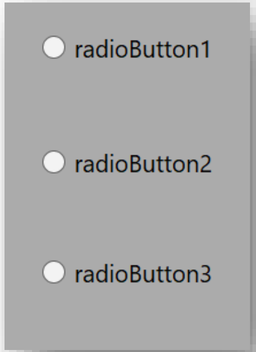
Windows form overview (GUI)

GUI Programming (control) examples:

RadioButton

Usage:
- To allow user to choose one option between number of options

Properties:
- Text
- Checked (true/false)
- ..etc



Windows form overview (GUI)

GUI Programming (control) examples:

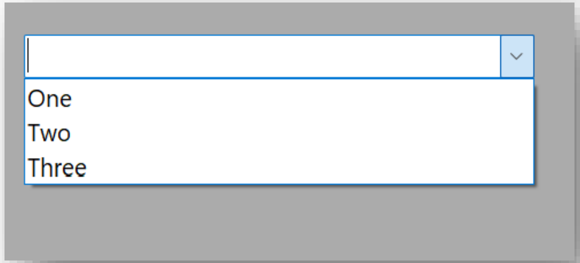
ComboBox

Usage:

- Dropdown list enable user to select one item from list

Properties:

- DataSource
- SelectedIndex
- SelectedItem
- ..etc



Windows form overview (GUI)

GUI Programming (control) examples:

How to add items into a ComboBox?

1. Add items one by one:

comboBox1.Items.Add("Sunday");

comboBox1.Items.Add("Monday ");

2. Bind ComboBox to a data source:

comboBox1.DataSource= new string[] {"Sunday ",“Monday"};

- We will use the second way when we populate combobox from a database.

- You can simply remove items from a ComboBox using **comboBox.Items.Remove(itemIndex)**

- Events: **SelectedIndexChanged()**

Windows form overview (GUI)

GUI Programming (control) examples:

ComboBox(example)

1. Create a simple form containing two ComboBoxes.

2.Initialize Combobox1 with values **“Day”** and **“Month”.**

3.Write a handler for the event “**SelectedIndexChanged**” of Combobox1 such that you set Combobox2 items according to the value selected of Combobox1.

i.e. if the user selected “Day” from combox1, combox2 should display weekdays (Saturday, Sunday) and if the user selected “Month” from combox1, combox2 should display months (January, February)

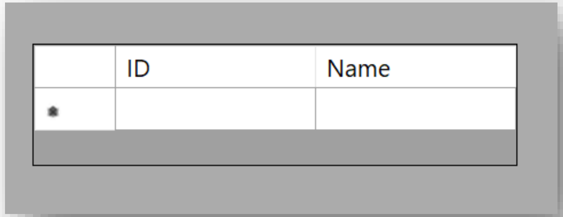
Windows form overview (GUI)

GUI Programming (control) examples:

DatagridView

Usage:
- The DataGridView control is designed to display tabular data

Properties:
- DataSource
- Columns.Add / Rows.Add
- Columns.Remove / Rows.Remove
- ..etc



Windows form overview (GUI)

GUI Programming (control) examples:

DatagridView

- To add a column to datagridview:

dataGridView1.Columns.Add(columnName, headerText);

OR using GUI

- To fill a datagridview’s rows with data, use:

string[] row=new string[] {“Soha”,”Student”};
dataGridView1.Rows.Add(row);

Windows form overview (GUI)

GUI Programming (control) examples:

DatagridView

- To access value of a certain cell in the datagridview:
String value = **dataGridView1.Rows[rowIndex].Cells[colName].Value.ToString();**
rowIndex is the row index of the cell we want to access, **colName** is the column name.

- To remove a row from DatagridView:
dataGridView1.Rows.Remove(rowIndex);

- To remove a column from DataGridview:
dataGridView1.Columns.Remove(colName);

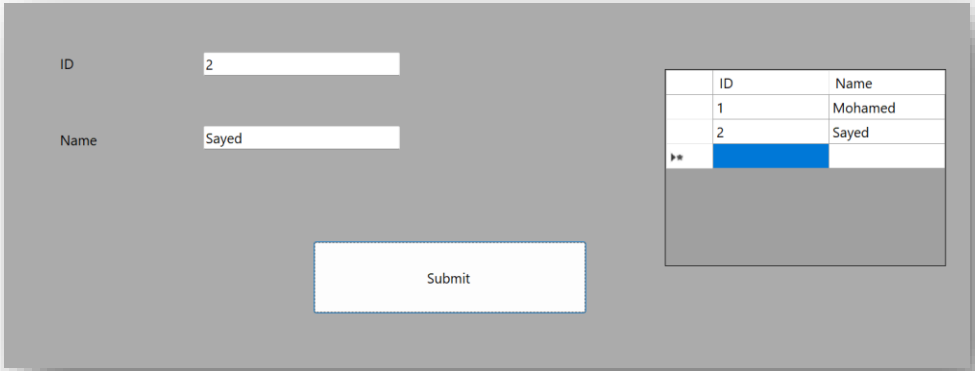
Windows form overview (GUI)

GUI Programming (control) examples:

DatagridView(example)

- Provide an event handler for **CellClickevent.**

- When a user selects a cell then the cell’s row is deleted.



	ID	Name
	1	Mohamed
	2	Sayed
▸*		

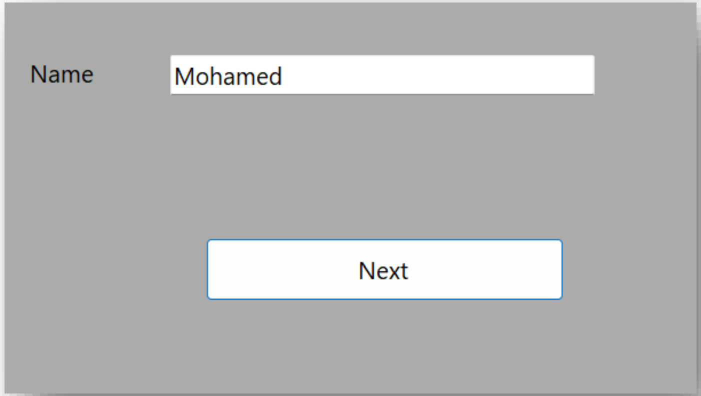
Windows form overview (GUI)

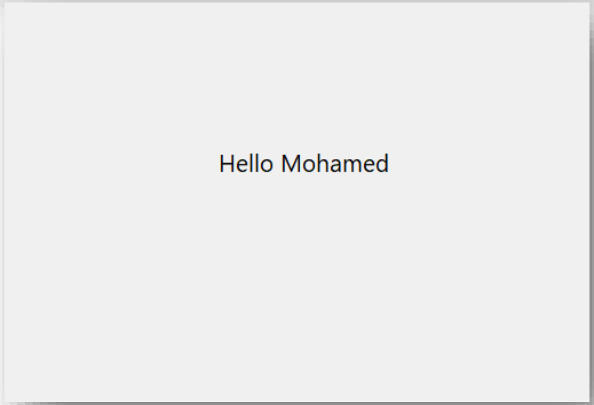
Passing Data between forms:

- Add a “Next” button to the form(Form1) that when clicked opens another form (Form2) containing a textbox for user’s name .

- Reminder, to create a new form from code:

Form2 f=new Form2();
// create a new form
f.Show();
//to show the form





Questions

Thank You 😇