HNDIT1012 Visual Application Programming

Week 1

What is a program?

- A computer program is a sequence or set of instructions in a programming language for a computer to execute.
- Computer programs are one component of software.

Programming languages

- A computer programming language is a language used to write computer programs, which involves a computer performing some kind of computations.
- Thousands of different programming languages have been created, and more are being created every year.
- Many programming languages are written in an imperative form (i.e., as a sequence of operations to perform) while other languages use the declarative form (i.e. the desired result is specified, not how to achieve it).
- Eg: C++, C#, VisualBasic, Java etc..

Source code

- A computer program in its human-readable form is called source code.
- Source code needs another computer program to execute because computers can only execute their native machine instructions.
- Therefore, source code may be translated to machine instructions using the language's translators (compiler / Interpreter)

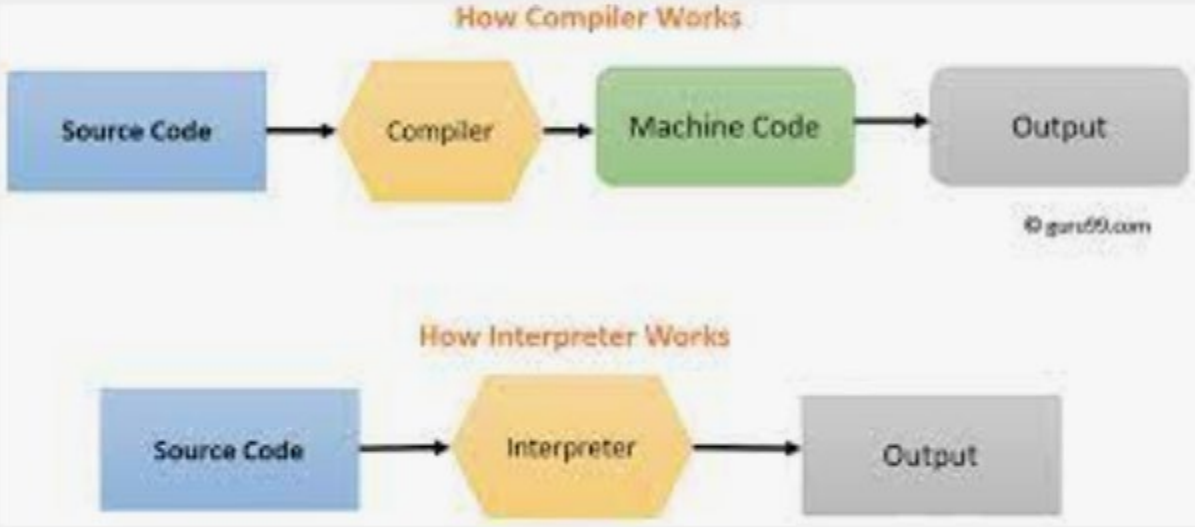
```
1   using System;
2
3   namespace CSharp_If_Statement
4   {
5       class Program
6       {
7           static void Main(string[] args)
8           {
9               string name;
10              Console.WriteLine("Enter a name:");
11              name = Console.ReadLine();
12              if (name == "John");
13              {
14                  Console.WriteLine("Hi John!");
15              }
16              Console.ReadKey();
17          }
18      }
19  }
```

Executable file

- An EXE file is an executable program you can run in Microsoft Windows. Most EXE files contain either Windows applications or application installers.

Language Translators

- Language translators allow computer programmers to write sets of instructions in specific programming languages. These instructions are converted by the language translator into machine code. The computer system then reads these machine code instructions and executes them.
  - Compiler
  - Interpreter


How Compiler Works
Source Code
Compiler
Machine Code
Output
How Interpreter Works
Source Code
Interpreter
Output

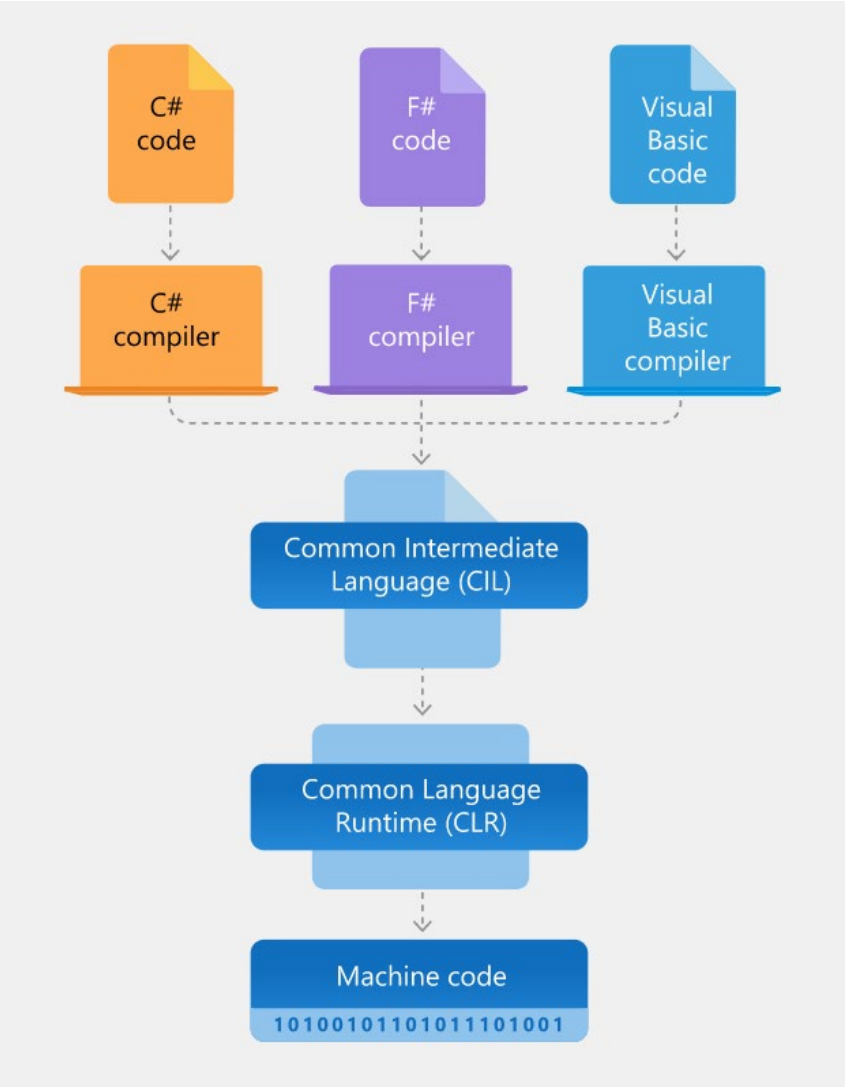
NET Framework

- NET Framework is a software development framework for building and running applications on Windows. . NET Framework is part of the . NET platform, a collection of technologies for building apps for Linux, macOS, Windows, iOS, Android, and more.

Architecture of .NET Framework ..

- .NET applications are written in the C#, F#, or Visual Basic programming language. Code is compiled into a language-agnostic Common Intermediate Language (CIL).
- Compiled code is stored in assemblies—files with a .dll or .exe file extension.
- When an app runs, the CLR takes the assembly and uses a just-in-time compiler (JIT) to turn it into machine code that can execute on the specific architecture of the computer it is running on.

Architecture of .NET Framework

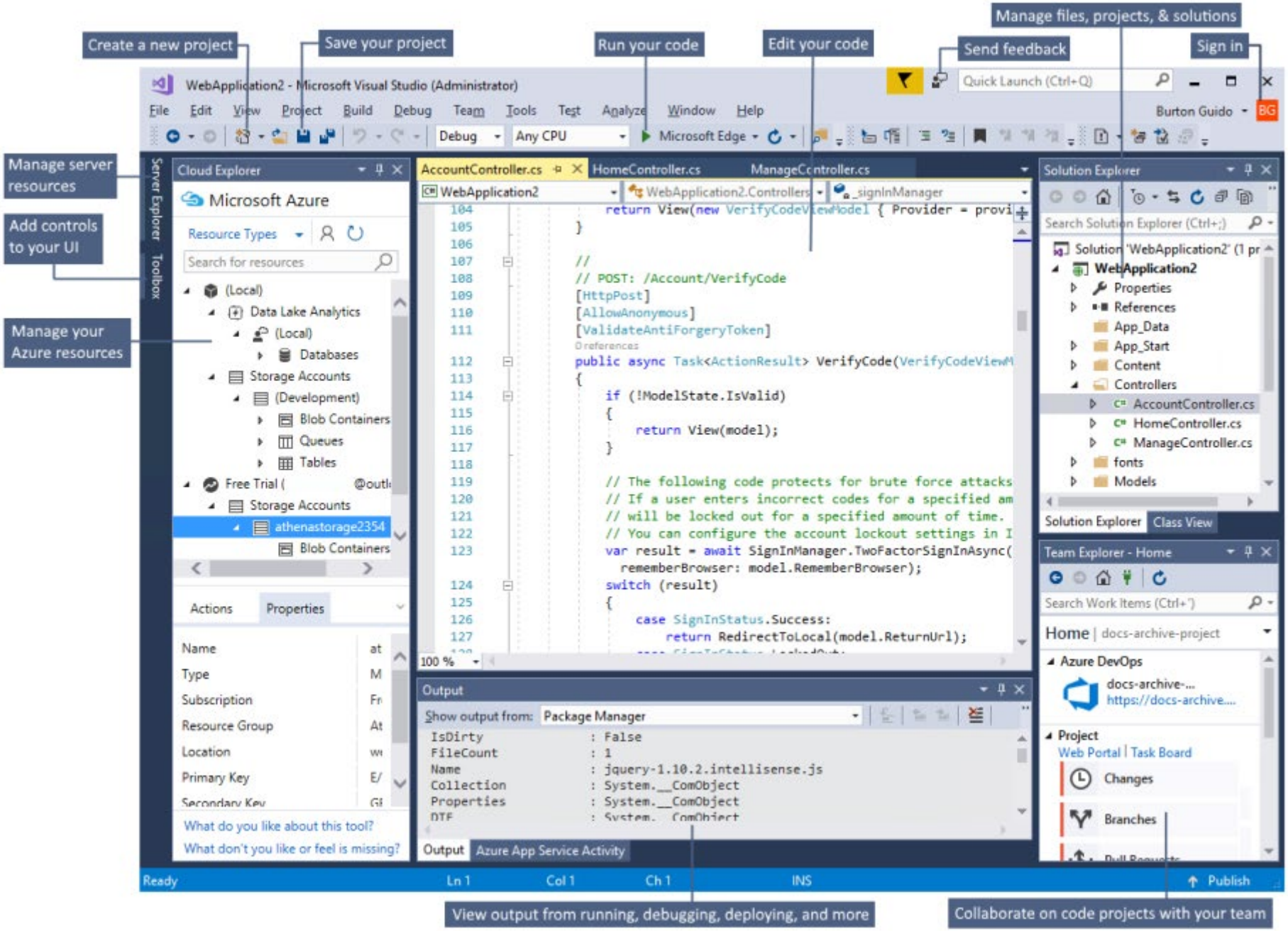


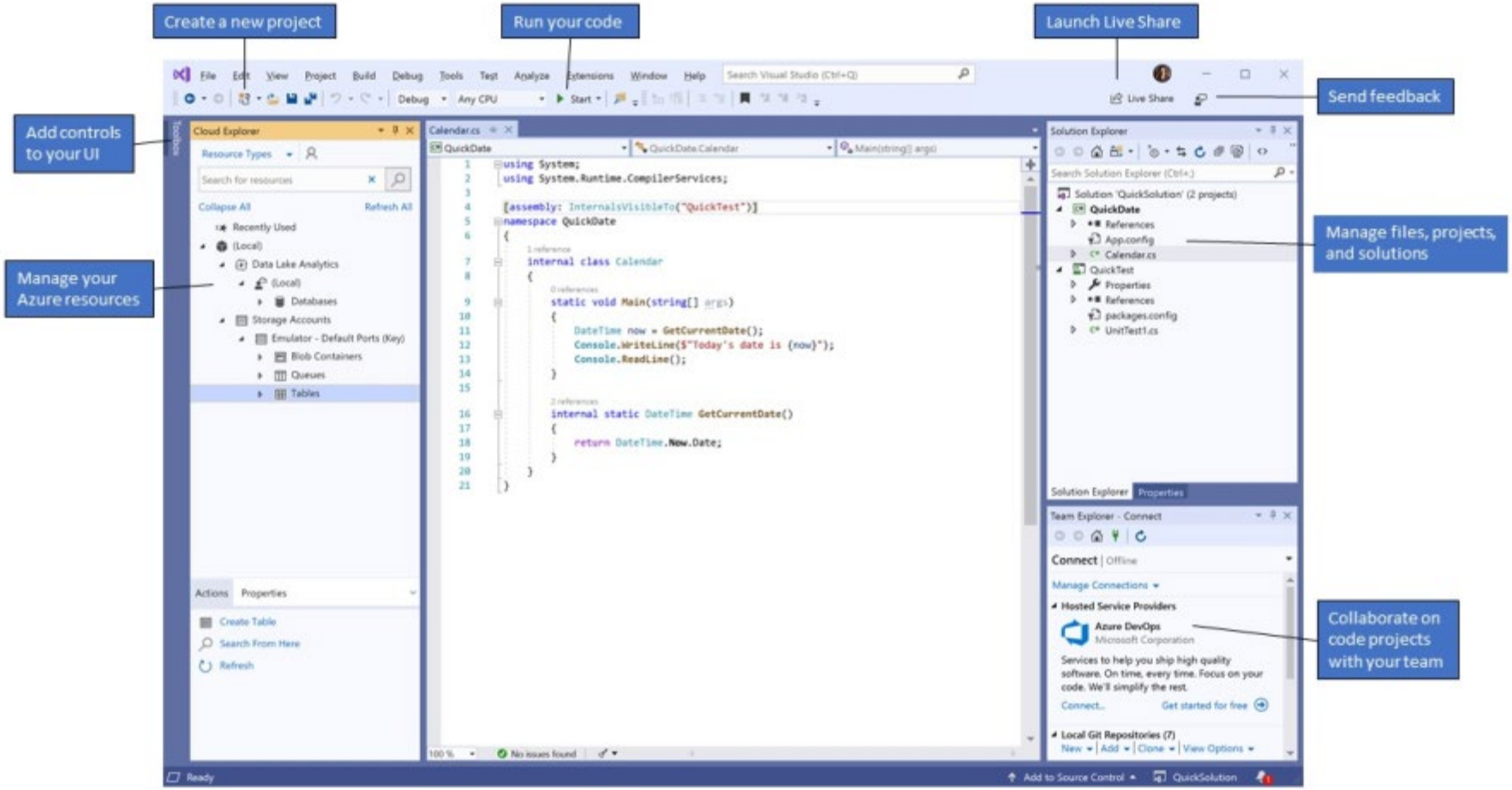
Introduction to IDE (Visual studio 2022)

- An integrated development environment (IDE) is software for building applications that combines common developer tools into a single GUI.
- Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop computer programs, as well as websites, web apps, web services and mobile apps.

Visual studio 2022..

- The Visual Studio integrated development environment is a creative launching pad that you can use to edit, debug, and build code, and then publish an app.
- An integrated development environment (IDE) is a feature-rich program that can be used for many aspects of software development.
- Visual Studio includes compilers, code completion tools, graphical designers, and many more features to ease the software development process.


Create a new project
Save your project
Run your code
Edit your code
Send feedback
Manage files, projects, & solutions
Sign in
Manage server resources
Add controls to your UI
Manage your Azure resources
View output from running, debugging, deploying, and more
Collaborate on code projects with your team
WebApplication2 - Microsoft Visual Studio (Administrator)
File Edit View Project Build Debug Team Tools Test Analyze Window Help
Quick Launch (Ctrl+Q)
Burton Guido
Debug
Any CPU
Microsoft Edge
Server Explorer
Toolbox
Cloud Explorer
Microsoft Azure
Resource Types
Search for resources
(Local)
Data Lake Analytics
(Local)
Databases
Storage Accounts
(Development)
Blob Containers
Queues
Tables
Free Trial (
Storage Accounts
athenastorage2354
Blob Containers
Actions
Properties
Name
Type
Subscription
Resource Group
Location
Primary Key
What do you like about this tool?
What don't you like or feel is missing?
AccountController.cs
HomeController.cs
ManageController.cs
WebApplication2
WebApplication2.Controllers
_signInManager
return View(new VerifyCodeViewModel { Provider = provi
}
//
// POST: /Account/VerifyCode
[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
0 references
public async Task<ActionResult> VerifyCode(VerifyCodeViewM
{
if (!ModelState.IsValid)
{
return View(model);
}
// The following code protects for brute force attacks
// If a user enters incorrect codes for a specified am
// will be locked out for a specified amount of time.
// You can configure the account lockout settings in I
var result = await SignInManager.TwoFactorSignInAsync(
rememberBrowser: model.RememberBrowser);
switch (result)
{
case SignInStatus.Success:
return RedirectToLocal(model.ReturnUrl);
100 %
Output
Show output from: Package Manager
IsDirty : False
FileCount : 1
Name : jquery-1.10.2.intellisense.js
Collection : System.__ComObject
Properties : System.__ComObject
Output
Azure App Service Activity
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'WebApplication2' (1 pr
WebApplication2
Properties
References
App_Data
App_Start
Content
Controllers
AccountController.cs
HomeController.cs
ManageController.cs
fonts
Models
Solution Explorer
Class View
Team Explorer - Home
Search Work Items (Ctrl+')
Home | docs-archive-project
Azure DevOps
docs-archive-...
https://docs-archive....
Project
Web Portal | Task Board
Changes
Branches
Ready
Ln 1
Col 1
Ch 1
INS
Publish


Create a new project
Run your code
Launch Live Share
Send feedback
Add controls to your UI
Manage your Azure resources
Manage files, projects, and solutions
Collaborate on code projects with your team

Visual studio 2022..

- Solution Explorer (top right) lets you view, navigate, and manage your code files. Solution Explorer can help organize your code by grouping the files into solutions and projects.
- The editor window (center), where you'll likely spend a majority of your time, displays file contents. This is where you can edit code or design a user interface such as a window with buttons and text boxes.
- The Output window (bottom center) is where Visual Studio sends notifications such as debugging and error messages, compiler warnings, publishing status messages, and more. Each message source has its own tab.

Installing VisualStudio 2022

Before you begin installing Visual Studio:

- Check the system requirements. These requirements help you know whether your computer supports Visual Studio 2022.
- Apply the latest Windows updates. These updates ensure that your computer has both the latest security updates and the required system components for Visual Studio.
- Reboot. The reboot ensures that any pending installs or updates don't hinder your Visual Studio install.
- Free up space. Remove unneeded files and applications from your system drive by, for example, running the Disk Cleanup app.

System Requirements for VisualStudio 2022 - Hardware

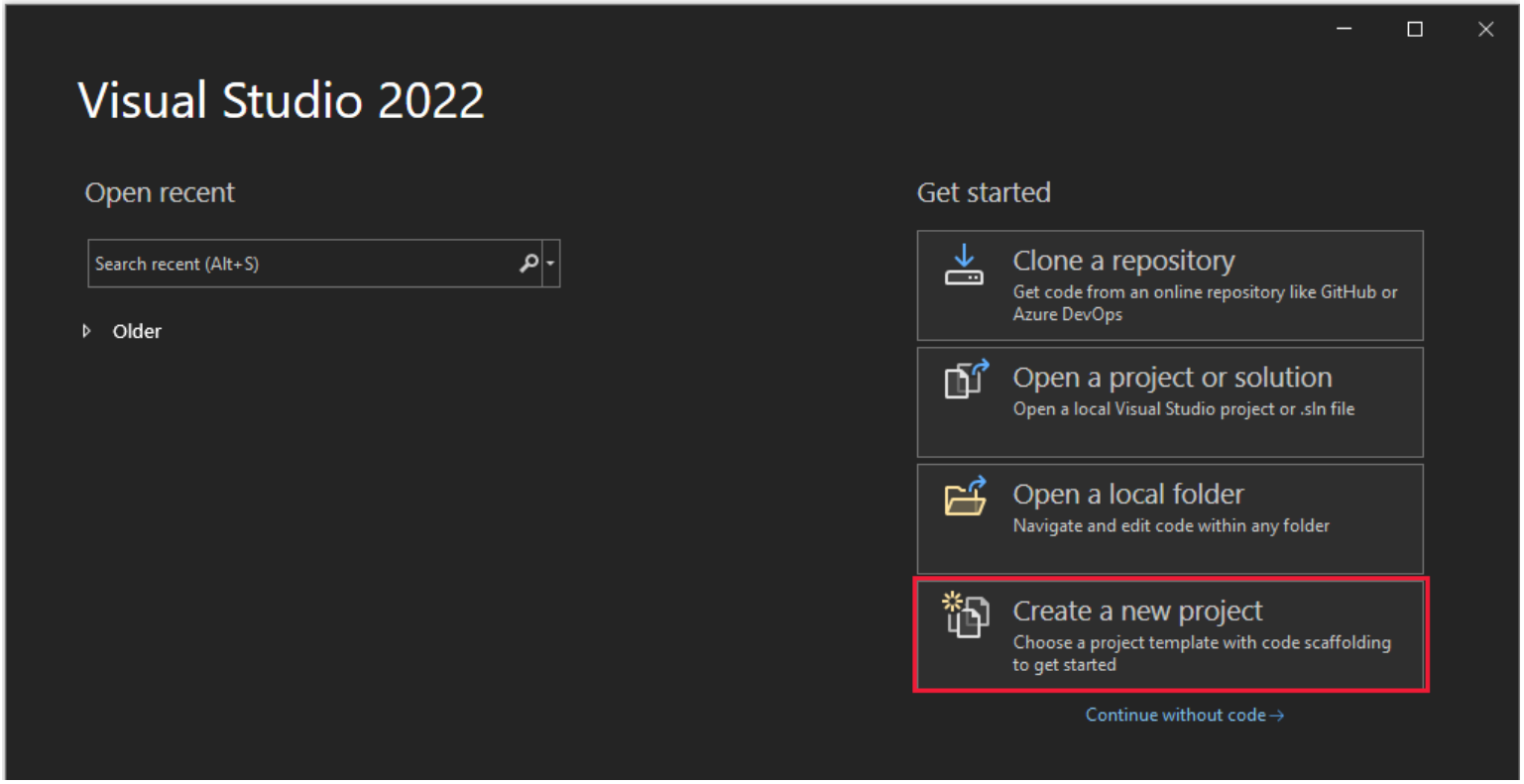
- 1.8 GHz or faster 64-bit processor; Quad-core or better recommended. ARM processors are not supported.
- Minimum of 4 GB of RAM. Many factors impact resources used; we recommend 16 GB RAM for typical professional solutions.
- Windows 365: Minimum 2 vCPU and 8 GB RAM. 4 vCPU and 16 GB of RAM recommended.
- Hard disk space: Minimum of 850 MB up to 210 GB of available space, depending on features installed; typical installations require 20-50 GB of free space.
- We recommend installing Windows and Visual Studio on a solid-state drive (SSD) to increase performance.
- Video card that supports a minimum display resolution of WXGA (1366 by 768); Visual Studio will work best at a resolution of 1920 by 1080 or higher.

Download Visual Studio

- Download link https://visualstudio.microsoft.com/downloads
- Follow the instructions in the following link. https://docs.microsoft.com/en-us/visualstudio/install/install-visual-studio?view=vs-2022

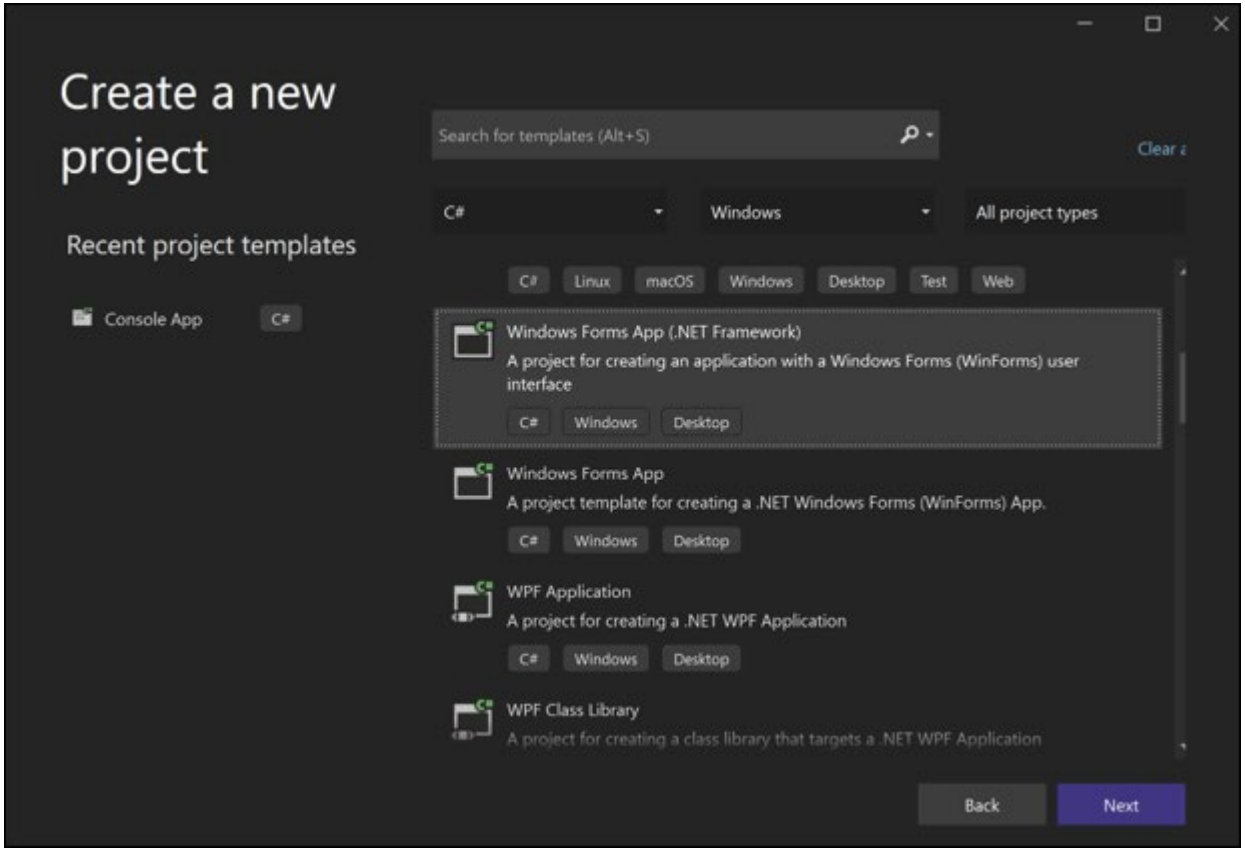
Creating a window based project.

- Open Visual Studio.
- On the start window, choose Create a new project.



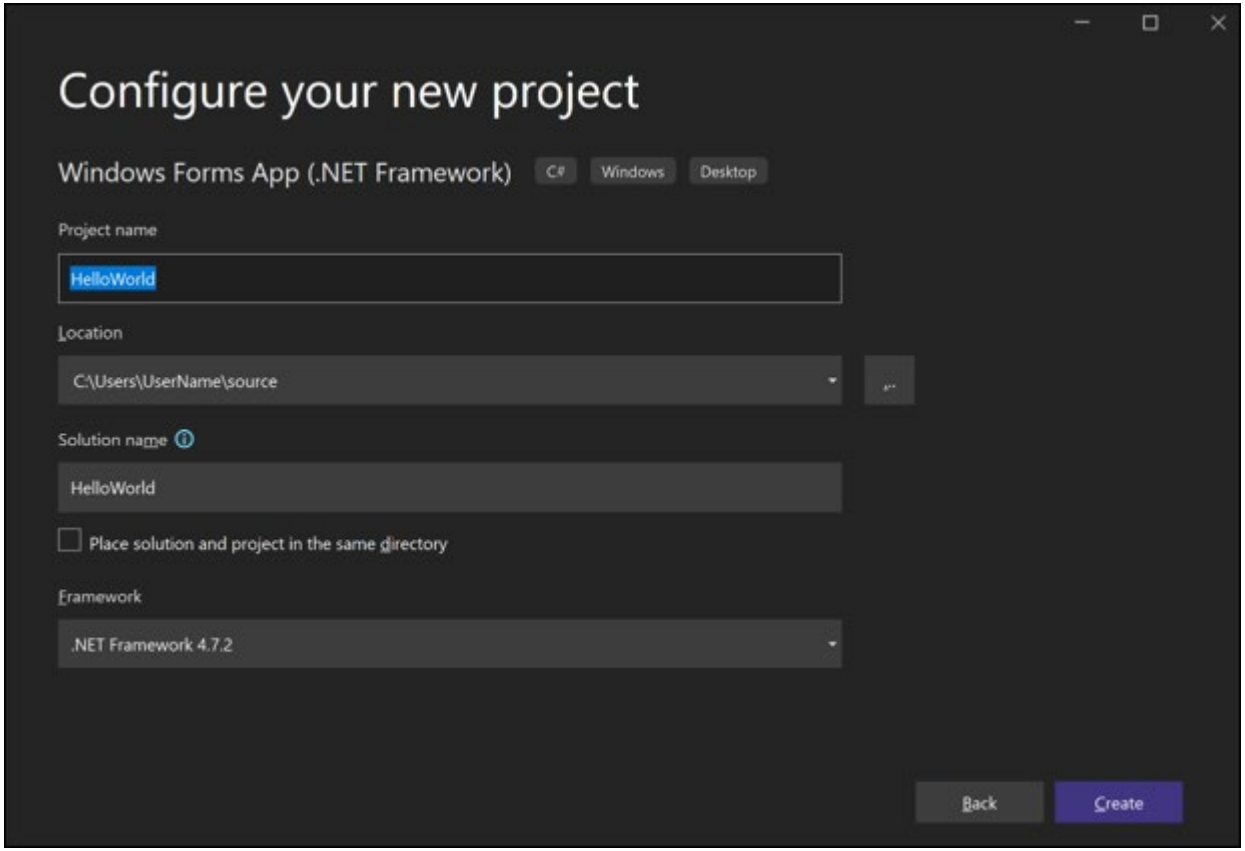
Creating a window based project…

- On the Create a new project window, choose the Windows Forms App (. NET Framework) template for C#. …



Creating a window based project…

- In the Configure your new project window, type or enter HelloWorld in the Project name box. Then, choose Create.

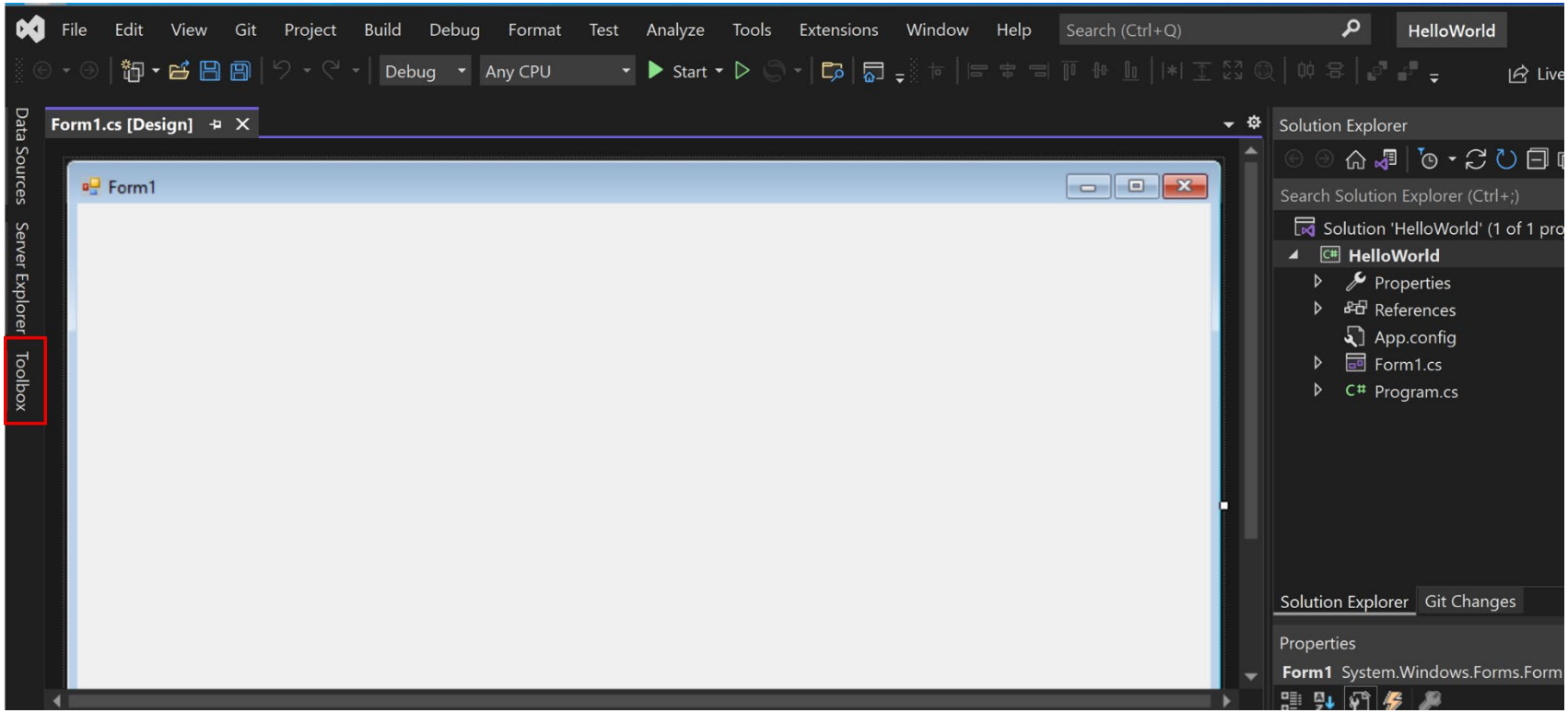


Create the application

- After you select your C# project template and name your file, Visual Studio opens a form for you. A form is a Windows user interface. We'll create a "Hello World" application by adding controls to the form, and then we'll run the app.

Create the application..

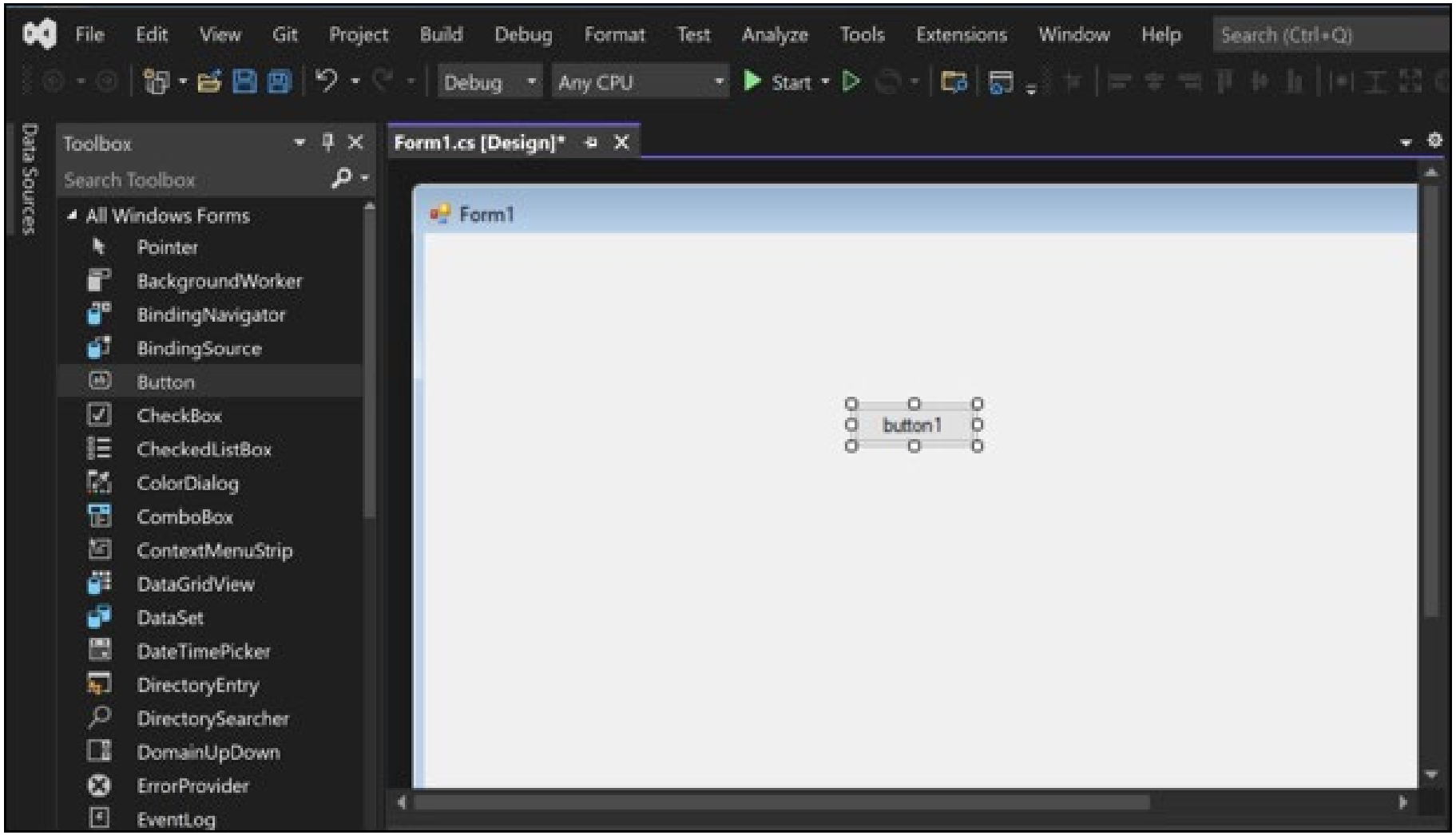
- Select Toolbox to open the Toolbox fly-out window.



Create the application..

- Select the Pin icon to dock the Toolbox window.
- Select the Button control and then drag it onto the form.
- In the Properties window, locate Text, change the name from button1 to Click this, and then press Enter.
- In the Design section of the Properties window, change the name from button1 to btnClickThis, and then press Enter.

Create the application..



Create the application..

- Select the Label control from the Toolbox window, and then drag it onto the form and drop it beneath the Click this button.

- In either the Design section or the (DataBindings) section of the Properties window, change the name of label1 to lblHelloWorld, and then press Enter.

Add code to the form

- In the Form1.cs [Design] window, double-click the Click this button to open the Form1.cs window.
- (Alternatively, you can expand Form1.cs in Solution Explorer, and then choose Form1.)
- In the Form1.cs window, after the private void line, type or enter lblHelloWorld.Text = "Hello World!";

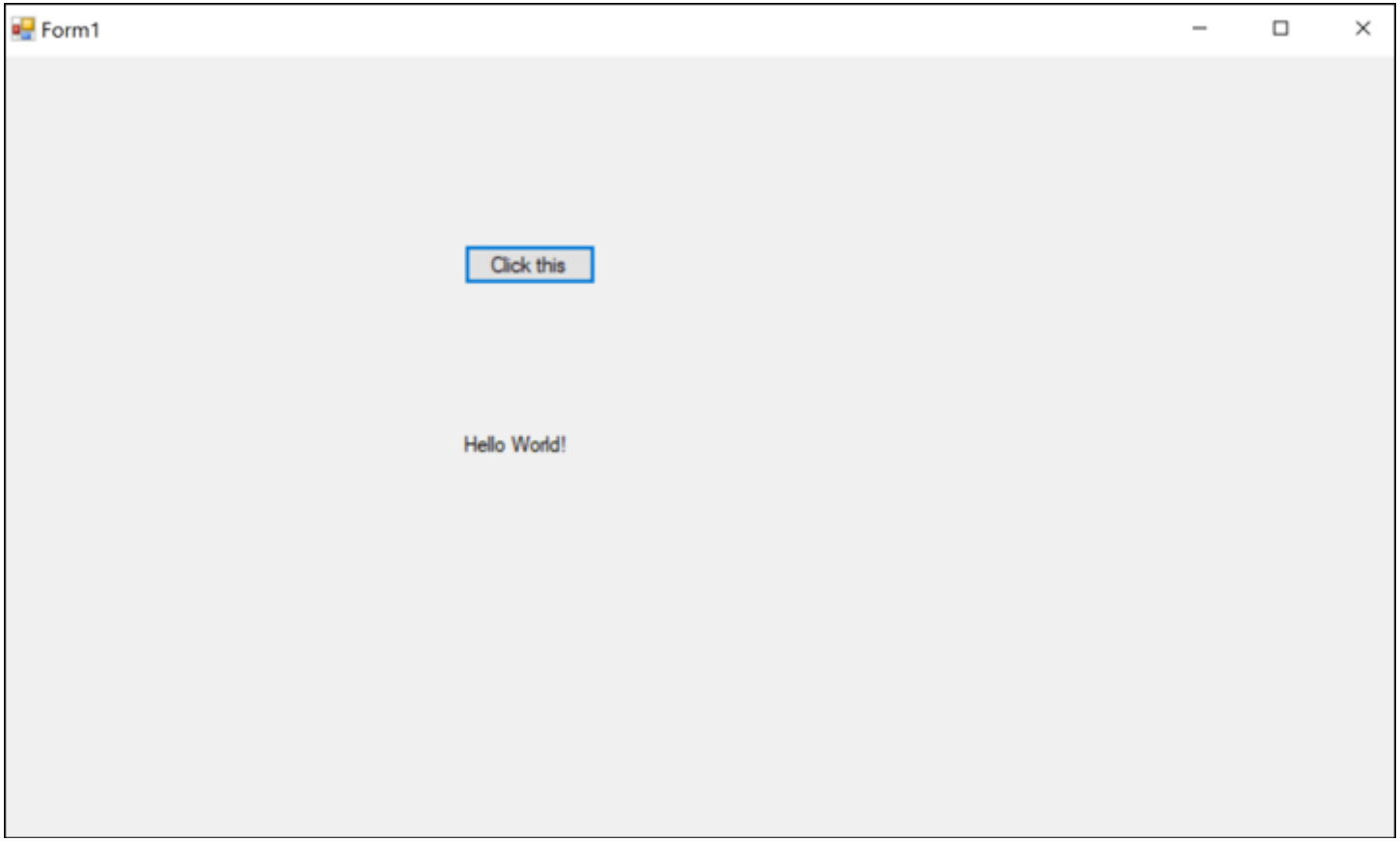
Run the application

- Select the Start button to run the application.

- Several things will happen. In the Visual Studio IDE, the Diagnostics Tools window will open, and an Output window will open, too. But outside of the IDE, a Form1 dialog box appears. It will include your Click this button and text that says label1.

- Select the Click this button in the Form1 dialog box. Notice that the label1 text changes to Hello World!.

Run the application…



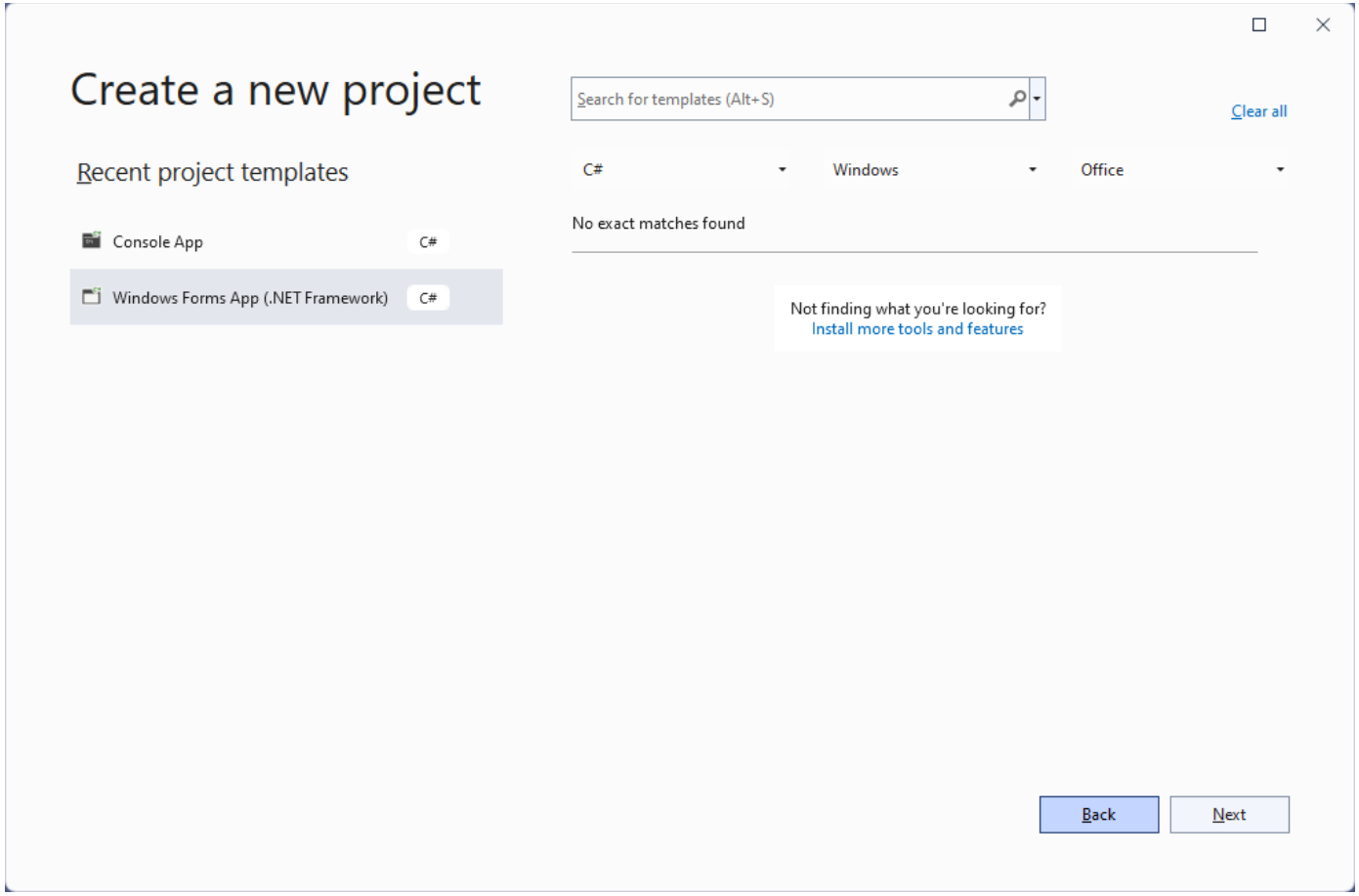
Close the Form1 dialog box to stop running the app.

Thank You

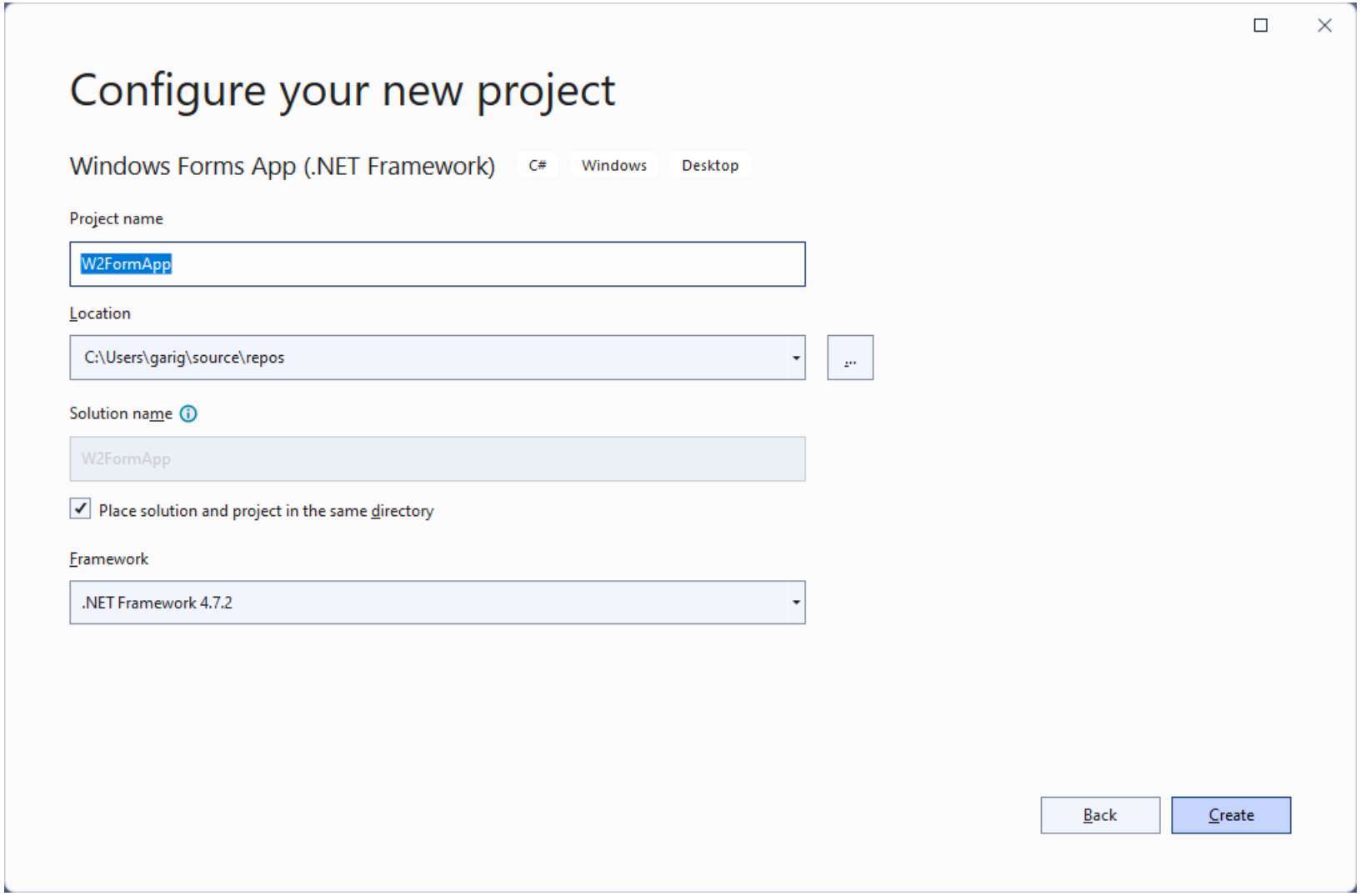
HNDIT1012 Visual Application Programming

Week 2

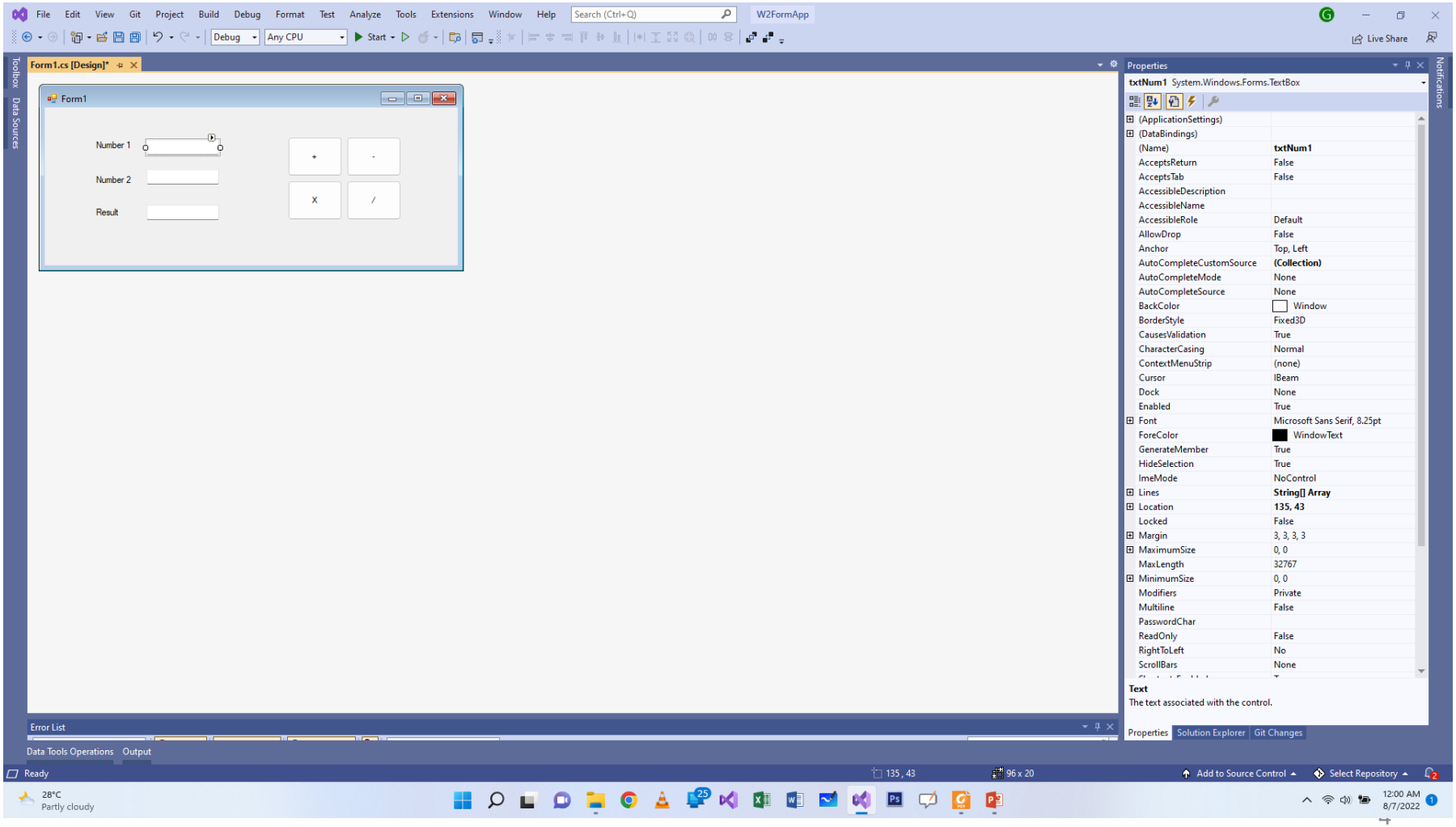
Windows Application - Step 1



Step 2


Configure your new project
Windows Forms App (.NET Framework)
C#
Windows
Desktop
Project name
W2FormApp
Location
C:\Users\garig\source\repos
Solution name
W2FormApp
Place solution and project in the same directory
Framework
.NET Framework 4.7.2
Back
Create

Designing Form – Step 3

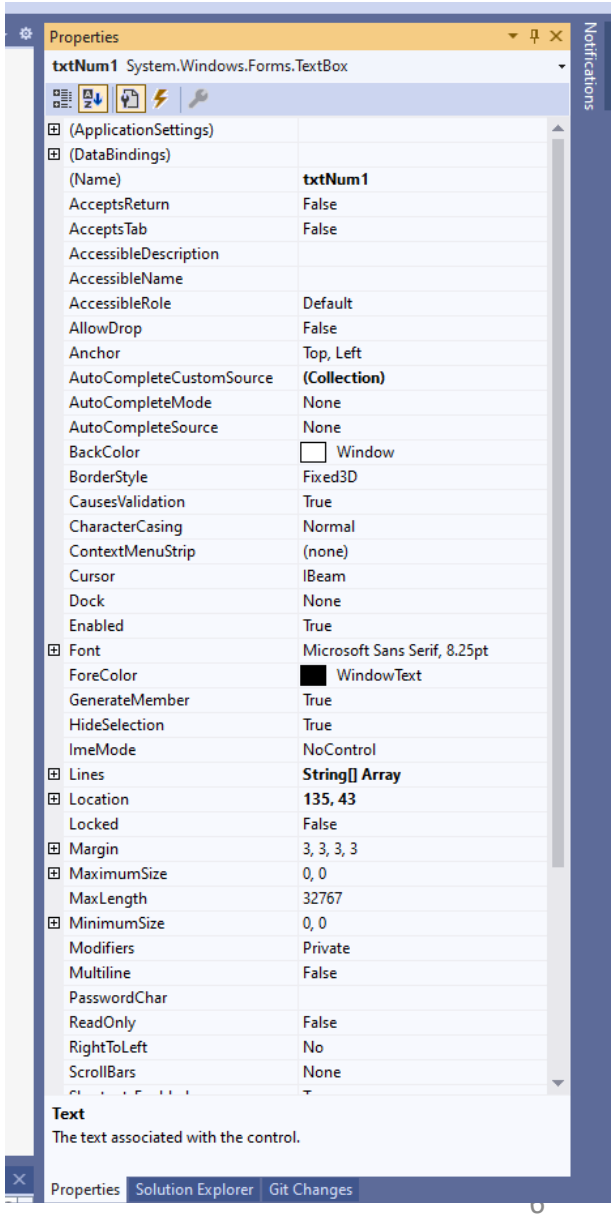


Adding Controls

- Design the form as shown above.
- The controls used in the form are:
  - TextBox :Text box controls allow entering text on a form at runtime. By default, it takes a single line of text, however, you can make it accept multiple texts and even add scroll bars to it. Let's create a text box by dragging a Text Box control from the Toolbox and dropping it on the form.
  - Button : Button control is used to perform a click event in Windows Forms, and it can be clicked by a mouse or by pressing Enter keys.
  - Label : The Label control represents a standard Windows label. It is generally used to display some informative text on the GUI which is not changed during runtime.

Property Window

- You can find Properties Window on the View menu. You can also open it by pressing F4 or by typing Properties in the search box. The Properties window displays different types of editing fields, depending on the needs of a particular property.



Setting the properties – Step 4

- Select the first textbox and set the name as txtNum1 on the property window.
- Similarly set the names of other two textboxes as txtNum2 and txtResult respectively.
- For Labels, Set the text property to Number 1, Number 2 and Result respectively
- For Buttons, set the properties as shown in the given table.

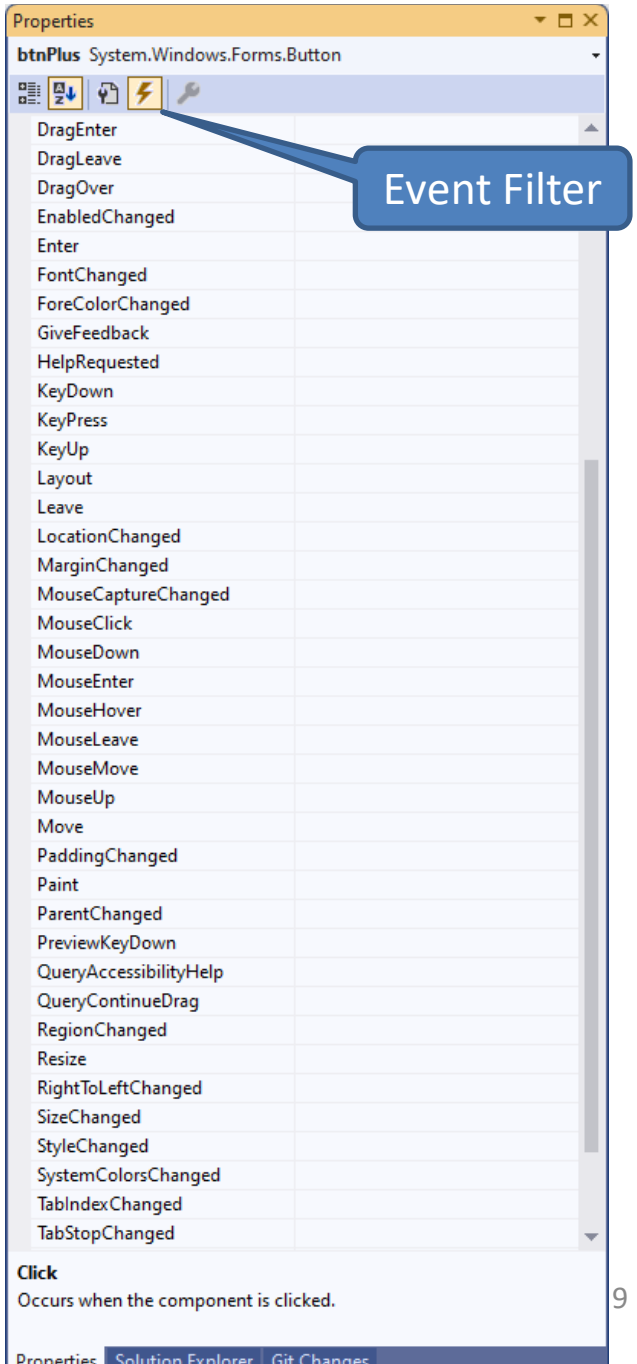
Name	Text
btnPlus	+
btnMinus	-
btnMul	X
btnDiv	/

Events

- An event is a signal that informs an application that something important has occurred.
- For example, when a user clicks a control on a form, the form can raise a Click event and call a procedure that handles the event. Events also allow separate tasks to communicate.
- Eg: click, double click, Key Press etc..

List of Events of a Button

- You can filter List of events on property window by clicking the event filter



Adding Click events – Step 5

- Double click on plus button to open code editor for click event and type the following code.

```
1 reference
private void btnPlus_Click(object sender, EventArgs e)
{
    txtResult.Text =txtNum1.Text + txtNum2.Text;
}
```

Running the application

- Input values for Number1 and Number 2 and check the Result by clicking the plus button.
- For eg:

Number 1	Number 2	Result
Com	puter	Computer
12	6	126
I am	Kumar	I amKumar

Addition Vs Concatenation

- In the above example '+' operator works as a string Concatenation operator, not an addition operator. Because both operands are string.
- If you need addition operation, you have to convert both operands to numeric values such as integer, double etc..

- Change the code of the click event as shown below to do arithmetic operations like
- +, -, * and /

```
1 reference
private void btnPlus_Click(object sender, EventArgs e)
{
    txtResult.Text = (Int32.Parse(txtNum1.Text) + Int32.Parse(txtNum2.Text)).ToString();

}
```

- Run the application and input integer values to Number 1 and Number 2

Int32.Parse Method

- Converts the string representation of a number to its 32-bit signed integer equivalent.

toString Method

- It converts an object to its string representation so that it is suitable for display.

C# Data types

C# type keyword	.NET type
bool	System.Boolean
byte	System.Byte
sbyte	System.SByte
char	System.Char
decimal	System.Decimal
double	System.Double
float	System.Single
int	System.Int32
uint	System.UInt32
nint	System.IntPtr
nuint	System.UIntPtr
long	System.Int64
ulong	System.UInt64
short	System.Int16
string	System.String

- Add click events to other buttons and type the code as given below:

```
1 reference
private void btnMinus_Click(object sender, EventArgs e)
{
    txtResult.Text = (Int32.Parse(txtNum1.Text) - Int32.Parse(txtNum2.Text)).ToString();

}

1 reference
private void btnMul_Click(object sender, EventArgs e)
{
    txtResult.Text = (Int32.Parse(txtNum1.Text) * Int32.Parse(txtNum2.Text)).ToString();
}

1 reference
private void btnDiv_Click(object sender, EventArgs e)
{
    txtResult.Text = (Int32.Parse(txtNum1.Text) / Int32.Parse(txtNum2.Text)).ToString();
}
```

- / operator performs an integer division if both operands are integers.
- Eg:
  - ❖3/4 = 0
  - ❖4/4 =1
  - ❖5/4=1

C# Arithmetic Operators

Operator	Name	Description	Example
+	Addition	Adds together two values	x + y
-	Subtraction	Subtracts one value from another	x - y
*	Multiplication	Multiplies two values	x * y
/	Division	Divides one value by another	x / y
%	Modulus	Returns the division remainder	x % y
++	Increment	Increases the value of a variable by 1	x++
--	Decrement	Decreases the value of a variable by 1	x--

C# assignment Operators

Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
&=	x &= 3	x = x & 3
|=	x |= 3	x = x | 3
^=	x ^= 3	x = x ^ 3
>>=	x >>= 3	x = x >> 3
<<=	x <<= 3	x = x << 3

C# Comparison Operators

Operator	Name	Example
==	Equal to	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

C# Logical Operators

Operator	Name	Description	Example
&&	Logical and	Returns true if both statements are true	x < 5 && x < 10
||	Logical or	Returns true if one of the statements is true	x < 5 || x < 4
!	Logical not	Reverse the result, returns false if the result is true	!(x < 5 && x < 10)

Bitwise Operators

Bitwise operator works on bits and perform bit by bit operation. The truth tables for
& - AND,
| - OR, and
^ - XOR are as follows –

p	q	p & q	p | q	p ^ q
0	0	0	0	0
0	1	0	1	1
1	1	1	1	0
1	0	0	1	1

Bitwise Operators

Assume if A = 60; and B = 13; then in the binary format they are as follows
A = 0011 1100 B = 0000 1101

Operator	Description	Example
&	Binary AND Operator copies a bit to the result if it exists in both operands.	(A & B) = 12, which is 0000 1100
|	Binary OR Operator copies a bit if it exists in either operand.	(A | B) = 61, which is 0011 1101
^	Binary XOR Operator copies the bit if it is set in one operand but not both.	(A ^ B) = 49, which is 0011 0001
~	Binary Ones Complement Operator is unary and has the effect of 'flipping' bits.	(~A) = -61, which is 1100 0011 in 2's complement due to a signed binary number.
<<	Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand.	A << 2 = 240, which is 1111 0000
>>	Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right operand.	A >> 2 = 15, which is 0000 1111

Operator Precedence in C#

Category	Operator	Associativity
Postfix	() [] -> . ++ - -	Left to right
Unary	+ - ! ~ ++ - - (type)* & sizeof	Right to left
Multiplicative	* / %	Left to right
Additive	+ -	Left to right
Shift	<< >>	Left to right
Relational	< <= > >=	Left to right
Equality	== !=	Left to right
Bitwise AND	&	Left to right
Bitwise XOR	^	Left to right
Bitwise OR	|	Left to right
Logical AND	&&	Left to right
Logical OR	||	Left to right
Conditional	?:	Right to left
Assignment	= += -= *= /= %=>>= <<= &= ^= |=	Right to left
Comma	,	Left to right

Thank You

HNDIT1012 Visual Application Programming

Week 3

Events in C#

There are several categories of events in C#. We are going to focus on some events in the following categories:

1. Mouse Events
2. Keyboard Events
3. Windows Form Events
4. Timer Events

Some Mouse Events

Event	Description
Click	This event occurs when the mouse button is released, typically before the MouseUp event
MouseClick	This event occurs when the user clicks the control with the mouse.
DoubleClick	This event occurs when the control is double-clicked.
MouseDown	This event occurs when the mouse pointer is over the control and the user presses a mouse button.
MouseEnter	This event occurs when the mouse pointer enters the border or client area of the control, depending on the type of control
MouseMove	This event occurs when the mouse pointer moves while it is over a control.
MouseUp	This event occurs when the mouse pointer is over the control and the user releases a mouse button.
MouseWheel	This event occurs when the user rotates the mouse wheel while the control has focus.

MouseEventArgs Class

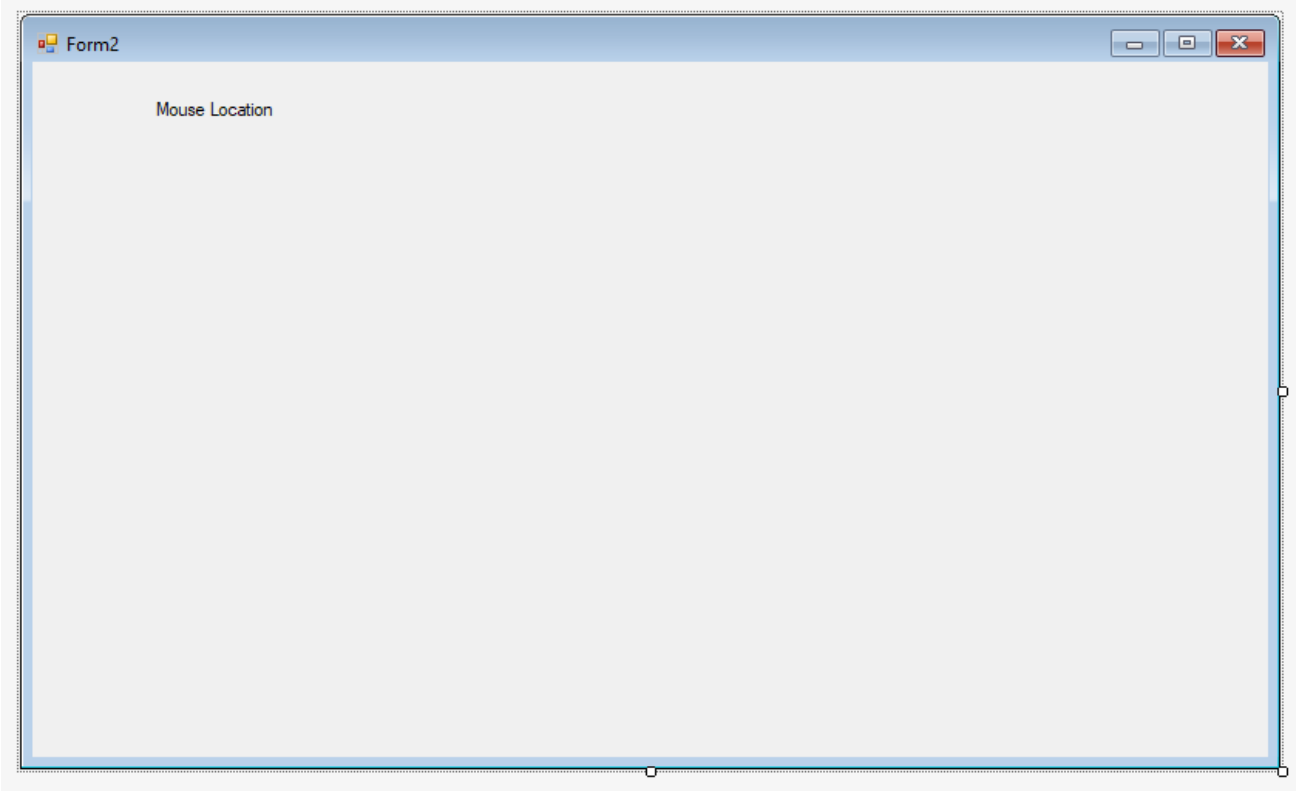
- Provides data for the MouseUp, MouseDown, and MouseMove events.
- A MouseEventArgs specifies which mouse button is pressed, how many times the mouse button was pressed and released, the coordinates of the mouse, and the amount the mouse wheel moved.

Some Properties of MouseEventArgs

Button	Gets which mouse button was pressed.
Clicks	Gets the number of times the mouse button was pressed and released.
Location	Gets the location of the mouse during the generating mouse event.
X	Gets the x-coordinate of the mouse during the generating mouse event.
Y	Gets the y-coordinate of the mouse during the generating mouse event.

Example

- Create a windows form application and place a label on the form as shown below.



- Open the property window and select the label to view the properties.
- Change the name of the label to “mousePos”.
- Select the mouse move event of the form and type the code as shown in the next slide.

```
1 reference
private void Form2_MouseMove(object sender, MouseEventArgs e)
{
    // This method display the cordinates of the mouse pointer
    // on the label box
    mousePos.Text = String.Format("({0}, {1})", e.X, e.Y);
}
```

Run the application and view the label while moving the mouse over the form.

This method display the current coordinates of the mouse pointer on the label box.

The MouseEventArgs object e contains all the information about the event such as x and y coordinates, button pressed etc..

- Add the mouseclick event for the form as shown below

```
1 reference
private void Form2_MouseClick(object sender, MouseEventArgs e)
{
    if (e.Button == MouseButtons.Left)

            MessageBox.Show("You clicked left mouse button");
    else
        if (e.Button == MouseButtons.Right)
            MessageBox.Show("You clicked Right mouse button");

}
```

Run the project and click on the form by left or right mouse button to view the message box.
MessageBox is used to display any message through a model Dialog.

Keyboard Events

Event	Description
KeyDown	This event is raised when a user presses a physical key. The KeyDown event occurs once.
KeyPress	This event is raised when the key or keys pressed result in a character. The KeyPress event, which can occur multiple times when a user holds down the same key.
KeyUp	This event is raised when a user releases a physical key. The KeyUp event occurs once when a user releases a key.

KeyEventArgs Class

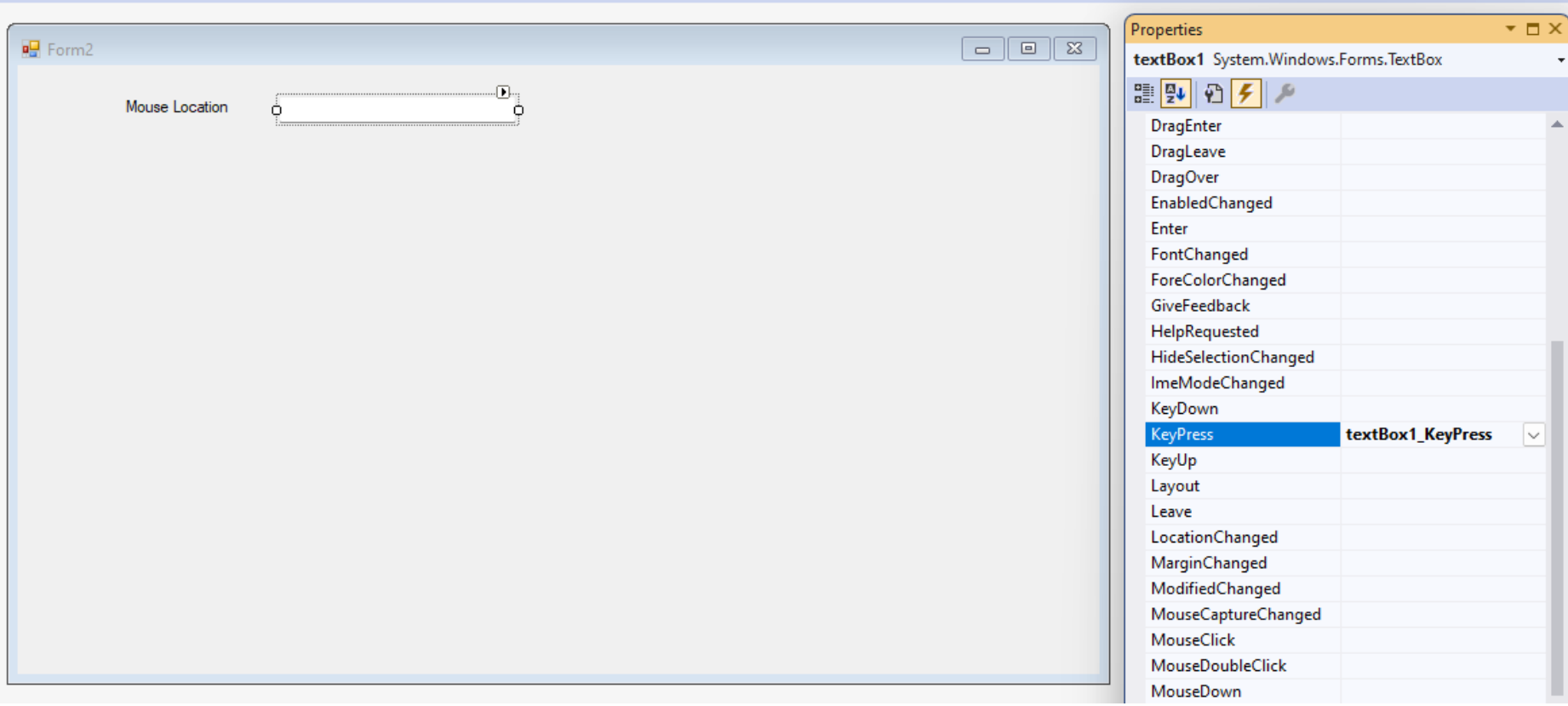
- Provides data for the KeyDown or KeyUp event.
- A KeyEventArgs, which specifies the key the user pressed and whether any modifier keys (CTRL, ALT, and SHIFT) were pressed at the same time, is passed with each KeyDown or KeyUp event.

KeyPressEventArgs Class

- Provides data for the KeyPress event.
- A KeyPressEventArgs specifies the character that is composed when the user presses a key. For example, when the user presses SHIFT + K, the KeyChar property returns an uppercase K.

Example

- Insert a text box to the form as shown below



- Select the keyPressed method of the textbox1.
- Type the code as shown below

```
1 reference
private void textBox1_KeyPress(object sender, KeyPressEventArgs e)
{
    e.KeyChar = '*';
}
```

- Whatever the key user pressed inside the textbox, the character will be replaced by “*”
- Run the solution and check the result by typing some text in the text box.

Some Windows Form Events

Event	Description
Form Load	Occurs whenever the user load the form
Form Close	Occurs when the form is closed.
Form Shown	Occurs whenever the form is first displayed.

Timer Event

Event	Description
Tick	Occurs when the specified timer interval has elapsed and the timer is enabled.
Elapsed	Occurs when the interval elapses.

Timer.Interval Property

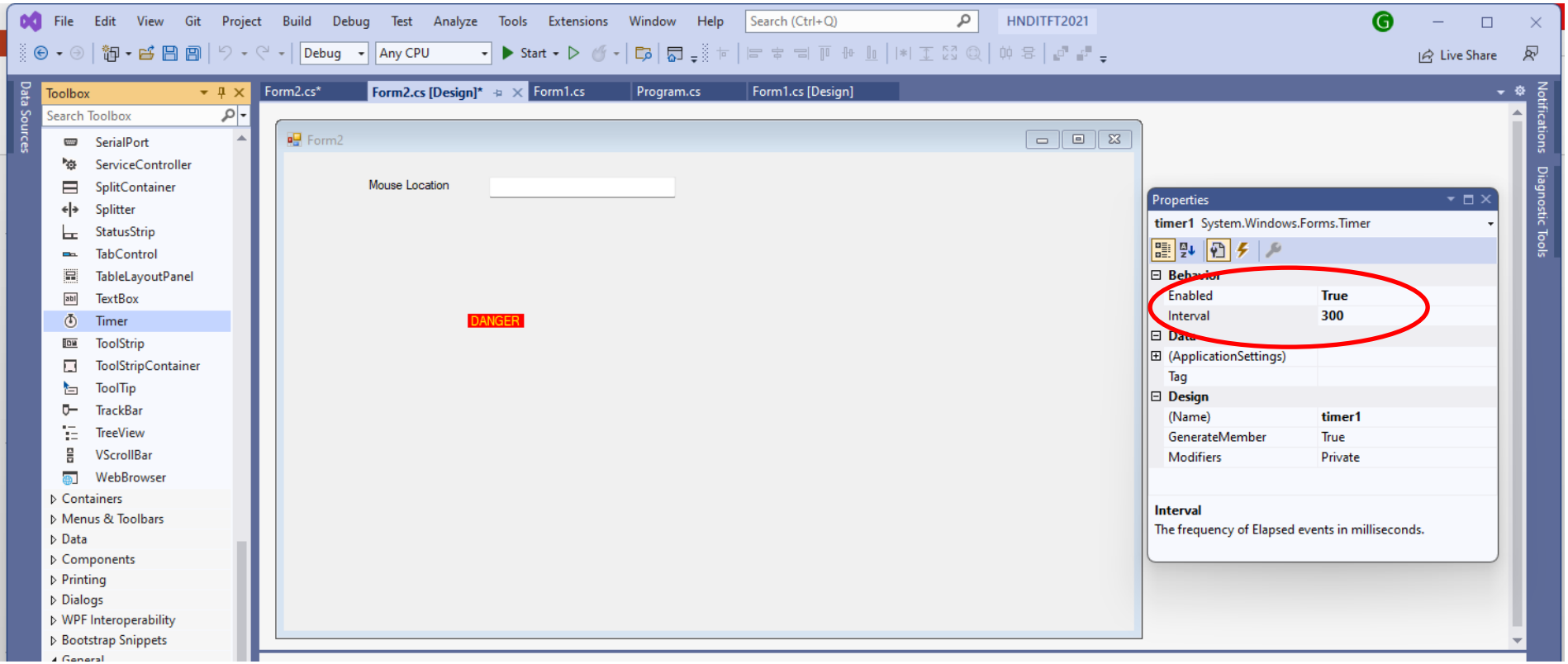
The time, in milliseconds, between Elapsed events. The value must be greater than zero, and less than or equal to Int32.MaxValue. The default is 100 milliseconds.

Timer.Enabled Property

Gets or sets a value indicating whether the Timer should raise the Elapsed event. True if the Timer should raise the Elapsed event; otherwise, false. The default is false.

Example

- Insert another label to the form and change the properties as shown below:
  - Name : lblDanger
  - Text : DANGER
  - Background color : Red
  - Foreground color : Yellow
- Insert a timer and set the timer properties as shown in the next slide.



Double click on timer1 to open the tick event and type the code as shown below:

```
1 reference
private void timer1_Tick(object sender, EventArgs e)
{
    lblDanger.Visible = !lblDanger.Visible;
}
```

- Run the programe to view the blinking DANGER label.
- Visible property will be complemented for every 300 milliseconds. (visible property will switch the values true and false for every 300 milliseconds)

Thank You

HNDIT1012 Visual Application Programming

Week 4

C# Variables

Variables are containers for storing data values.

Declaring (Creating) Variables

To create a variable, you must specify the type and assign it a value:

Syntax

type variableName = value;

Where type is a C# type (such as int or string), and variableName is the name of the variable (such as x or name). The equal sign is used to assign values to the variable.

Eg:-

```
int myNum = 5;
double myDoubleNum = 5.99D;
char myLetter = 'D';
bool myBool = true;
string myText = "Hello";
```

```
int x = 5, y = 6, z = 50;
```

Declare Many Variables

To declare more than one variable of the **same type**, use a comma-separated list:

```
int x = 5, y = 6, z = 50;
```

C# Identifiers

All C# variables must be identified with unique names.

These unique names are called identifiers.

Identifiers can be short names (like x and y) or more descriptive names (age, sum, totalVolume).

It is recommended to use descriptive names in order to create understandable and maintainable code:

The general rules for naming variables

- Names can contain letters, digits and the underscore character (_)
- Names must begin with a letter
- Names should start with a lowercase letter and it cannot contain whitespace
- Names are case sensitive ("myVar" and "myvar" are different variables)
- Reserved words (like C# keywords, such as int or double) cannot be used as names

escaping character

C# includes escaping character \ (backslash) before these special characters to include in a string

Use backslash \ before double quotes and some special characters such as \,\n,\r,\t, etc. to include it in a string.

Eg:

string text = "This is a \"string\" in C#.";

string str = "xyzdef\\rabc";

string path = "\\\\mypc\\ shared\\project";

Escape Sequence	Represents
\a	Bell (alert)
\b	Backspace
\f	Form feed
\n	New line
\r	Carriage return
\t	Horizontal tab
\v	Vertical tab
\'	Single quotation mark
\"	Double quotation mark
****	Backslash
\?	Literal question mark
**** *ooo*	ASCII character in octal notation
\x *hh*	ASCII character in hexadecimal notation
\x *hhhh*	Unicode character in hexadecimal notation if this escape sequence is used in a wide-character constant or a Unicode string literal. For example, WCHAR f = L'\x4e00' or WCHAR b[] = L"The Chinese character for one is \x4e00".

Verbatim string

Verbatim string in C# allows a special characters and line brakes. Verbatim string can be created by prefixing @ symbol before double quotes.

```
string str = @"xyzdef\rabc";
string path = @"\\mypc\shared\project";
string email = @"test@test.com";
```

Control Structures in C#

The if Statement

Use the if statement to specify a block of C# code to be executed if a condition is True.

Syntax

```
if (condition)
{
  // block of code to be executed if the condition is True
}
```

Eg:

```
int x = 20;
int y = 18;
if (x > y)
{
  Console.WriteLine("x is greater than y");
}
```

If --- else --

Use the else statement to specify a block of code to be executed if the condition is False.

Syntax

```
if (condition)
{
  // block of code to be executed if the condition is True
}
else
{
  // block of code to be executed if the condition is False
}
```

```
int time = 20;
if (time < 18)
    {
      Console.WriteLine("Good day.");
    }
else
    {
      Console.WriteLine("Good evening.");
    }

// Outputs "Good evening."
```

The else if Statement

Use the else if statement to specify a new condition if the first condition is False.

Syntax

```
if (condition1)
{
  // block of code to be executed if condition1 is True
}
else if (condition2)
{
  // block of code to be executed if the condition1 is false and condition2 is True
}
else
{
  // block of code to be executed if the condition1 is false and condition2 is False
}
```

```
int time = 22;
if (time < 10)
    {
      Console.WriteLine("Good morning.");
    }
else if (time < 20)
        {
          Console.WriteLine("Good day.");
        }
    else
        {
          Console.WriteLine("Good evening.");
        }
// Outputs "Good evening."
```

Short Hand If...Else (Ternary Operator)

There is also a short-hand if else, which is known as the ternary operator because it consists of three operands. It can be used to replace multiple lines of code with a single line. It is often used to replace simple if else statements:

Syntax

variable = (condition) ? expressionTrue : expressionFalse;

Eg:

```
int time = 20;
string result = (time < 18) ? "Good day." : "Good evening.";
Console.WriteLine(result);
```

Thank You

HNDIT1012 Visual Application Programming

Week 4

Switch Statement

Switch is a selection statement that chooses a single switch section to execute from a list of candidates based on a pattern match with the match expression.

Eg:

```
int caseSwitch = 1;
switch (caseSwitch)
{
        case 1:
                Console.WriteLine("Case 1");
                break;
        case 2:
                Console.WriteLine("Case 2");
                break;
        default:
                Console.WriteLine("Default case");
                break;
}
```

Example

```
class Switch
{
static void Main()
{
        Console.Write("Enter your selection (1, 2, or 3): ");
        string s = Console.ReadLine();
        int n = Int32.Parse(s);
        switch (n)
        {
        case 1:
                Console.WriteLine("Current value is 1");
                break;
        case 2:
                Console.WriteLine("Current value is 2");
                break;
        case 3:
                Console.WriteLine("Current value is 3");
                break;
        default:
                Console.WriteLine("Sorry, invalid selection.");
                break;
        }

Console.WriteLine("Press any key to exit.");
Console.ReadKey();
}
```

Loops / Iterations

Loops are used to execute a block of statements several times. In C# following types of loops are discussed here.

for loop

while loop

do while loop

for loop

- The for statement executes a statement or a block of statements while a specified Boolean expression evaluates to true .
- At any point within the for statement block, you can break out of the loop by using the break statement, or step to the next iteration in the loop by using the continue statement. You can also exit a for loop by the goto, return, or throw statements

Structure of the for statement

- The for statement defines initializer, condition, and iterator sections:

```
for (initializer; condition; iterator)
        body
```

All three sections are optional. The body of the loop is either a statement or a block of statements. The statements in the initializer section are executed only once, before entering the loop.

The condition section, if present, must be a boolean expression. That expression is evaluated before every loop iteration. If the condition section is not present or the boolean expression evaluates to true , the next loop iteration is executed; otherwise, the loop is exited.

The iterator section defines what happens after each iteration of the body of the loop. The iterator section contains zero or statement expressions, separated by commas.

Example

```
for (int i = 0; i < 5; i++)
{
        Console.WriteLine(i);
}
```

Another Example

```
int i;
int j = 10;
for (
        i = 0, Console.WriteLine($"Start: i={i}, j={j}");
        i < j;
        i++, j--, Console.WriteLine($"Step: i={i}, j={j}")
        )
{
        // Body of the loop.
}
```

Initializer Section

ndition

Iterator Section

The following example defines the infinite for loop:

```
for ( ; ; )
{
        // Body of the loop.
}
```

Break statement

The break statement terminates the closest enclosing loop or switch statement in which it appears. Control is passed to the statement that follows the terminated statement, if any.

```
for (int i = 1; i <= 100; i++)
{
        if (i == 5)
                {
                        break;
                }
        Console.WriteLine(i);
}
```

Output:

```
1
2
3
4
```

While Loop

- The while statement executes a statement or a block of statements while a specified Boolean expression evaluates to true . Because that expression is evaluated before each execution of the loop, a while loop executes zero or more times. This differs from the do loop, which executes one or more times.
- At any point within the while statement block, you can break out of the loop by using the break statement.
- You can step directly to the evaluation of the while expression by using the continue statement. If the expression evaluates to true , execution continues at the first statement in the loop. Otherwise, execution continues at the first statement after the loop.
- You can also exit a while loop by the goto, return, or throw statements

Example

```
int n = 0;
while (n < 5)
{
        n++;
}
TextBox1.Text= Convert.ToInt32(n);
```

do ... while loop

The while loop tests the condition before executing the code following the while .

The do ... while loop executes the code first, and then checks the condition. The do while loop is shown in the following code:

```
int counter = 0;
do
{
        Console.WriteLine($"Hello World! The counter is {counter}");
        counter++;
} while (counter < 10);
```

Thank You

HNDIT1012 Visual Application Programming

Week 6

Array

- Arrays are used to store multiple values in a single variable, instead of declaring separate variables for each value.
- To declare an array, define the variable type with **square brackets**:

Eg: string[] cars;

int[] marks;

Array …

To insert values to it, we can use an array literal - place the values in a comma-separated list, inside curly braces:

```
string[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
```

To create an array of integers, you could write:

```
int[] myNum = {10, 20, 30, 40};
```

Access the Elements of an Array

You access an array element by referring to the index number.

This statement assign the value of the first element in cars to the string variable mycar:

```
string mycar=cars[0];
```

Array Length

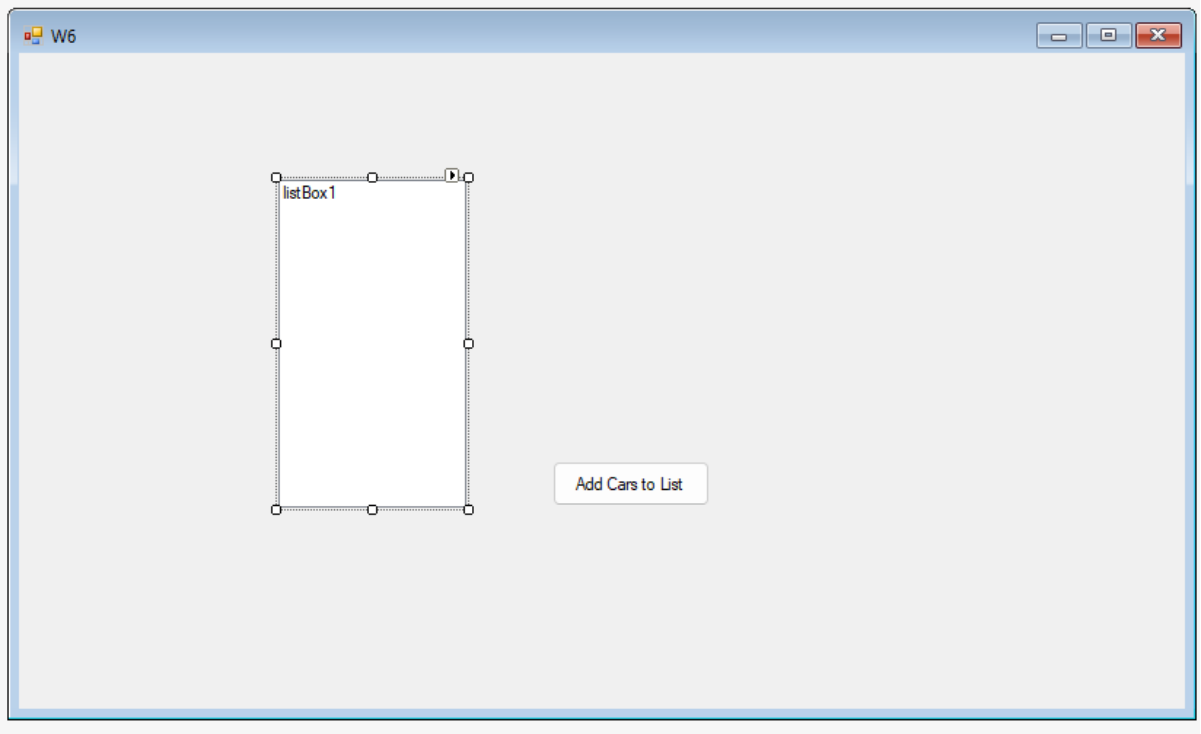
To find out how many elements an array has, use the Length property:

Eg:

```
int x=cars.Length;
```

The following example add all elements in the cars array to the listbox:

- Create a C# solution and design the form as shown below:



Add following code to Button Click event

```
1 reference
private void button1_Click(object sender, EventArgs e)
{
    string[] cars= { "Volvo", "BMW", "Ford", "Mazda" };
    for (int i = 0; i < cars.Length; i++)
        listBox1.Items.Add(cars[i]);

}
```

Above code will add the elements of car array to the listbox items.
The method Add(item) will add an item to the Items collection of the listbox1.

Same example using foreach loop

```
1 reference
private void button1_Click(object sender, EventArgs e)
{
    string[] cars= { "Volvo", "BMW", "Ford", "Mazda" };
    foreach(string car in cars)
        listBox1.Items.Add(car);


}
```

For single-dimensional arrays, the foreach statement processes elements in increasing index order, starting with index 0 and ending with index Length - 1:

Sorting an Array

There are many array methods available, for example Sort(), which sorts an array alphabetically or in an ascending order:

Eg:

```
// Sort a string
string[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
Array.Sort(cars);

// Sort an int
int[] myNumbers = {5, 1, 8, 9};
Array.Sort(myNumbers);
```

Reverse(Array)

Reverses the sequence of the elements in the entire one-dimensional Array.

Reverse(Array, Int32, Int32)
Reverses the sequence of a subset of the elements in the one-dimensional Array.

Parameters

Array
The one-dimensional Array to reverse.

index
Int32
The starting index of the section to reverse.

length
Int32
The number of elements in the section to reverse.

Example

```
1 reference
private void button1_Click(object sender, EventArgs e)
{
    string[] cars = { "Volvo", "BMW", "Toyota", "Mazda", "Suzuki" };
    Array.Sort(cars);
    Array.Reverse(cars,0,2);
    comboBox1.Items.AddRange(cars);

}
```

AddRange() method add an array to Items collection of a ListBox or ComboBox.

In the above example, the Reverse() method reverse the first 2 elements of the array. To reverse all elements, change the statement as given below:

Array.Reverse(cars,0,cars.Length);

Here the second parameter 0 indicates beginning of the array and second parameter is the total number of elements to be reversed.

Thank You

HNDIT1012 Visual Application Programming

Week 7

C# Strings

Strings are used for storing text.

A string variable contains a collection of characters surrounded by double quotes:

Example

Create a variable of type string and assign it a value:

```
string greeting = "Hello";
```

String Length

A string in C# is actually an object, which contain properties and methods that can perform certain operations on strings. For example, the length of a string can be found with the Length property:

```
string txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
int x=txt.Length;
```

String Methods

There are many string methods available, for example ToUpper() and ToLower(), which returns a copy of the string converted to uppercase or lowercase:

```
string txt = "Hello World";
string u=txt.ToUpper();
string v=txt.ToLower();
```

use the string.Concat() method to concatenate two strings:

string.Concat() method to concatenate two strings:

```
string firstName = "John ";
string lastName = "Doe";
string name = string.Concat(firstName, lastName);
```

Access Strings

You can access the characters in a string by referring to its index number inside square brackets [].

This example prints the first character in myString:

```
string myString = "Hello";
Console.WriteLine(myString[1]);
```

String methods

Methods	Description
Format()	returns a formatted string
Split()	splits the string into substring
Substring()	returns substring of a string
Compare()	compares string objects
Replace()	replaces the specified old character with the specified new character
Contains()	checks whether the string contains a substring
Join()	joins the given strings using the specified separator
Trim()	removes any leading and trailing whitespaces
EndsWith()	checks if the string ends with the given string
IndexOf()	returns the position of the specified character in the string
Remove()	removes characters from a string
ToUpper()	converts the string to uppercase
ToLower()	converts the string to lowercase
PadLeft()	returns string padded with spaces or with a specified Unicode character on the left
PadRight()	returns string padded with spaces or with a specified Unicode character on the right
StartsWith()	checks if the string begins with the given string
ToCharArray()	converts the string to a char array
LastIndexOf()	returns index of the last occurrence of a specified string

```
1 reference
private void button1_Click(object sender, EventArgs e)
{
    string myString = textBox1.Text;
    string s=myString;
    var x=myString.Split(' ');
    listBox1.Items.AddRange(x);

    myString = String.Format("First letter of your text is {0} and the last letter is {1}", myString[0], myString[myString.Length - 1]);
    MessageBox.Show(myString);

    for (int i = 0; i <= s.Length;i++)
    {
        richTextBox1.Text+=s.Substring(0,i) +"\n";
    }

    MessageBox.Show(s.Replace("p","b"));   // replace all "p" by "b"
    MessageBox.Show(s.PadRight(20,'#'));
    s = "Computer is an electronic information processing machine";
    MessageBox.Show(s.IndexOf("is").ToString()); // search from the begning of the string
    MessageBox.Show(s.IndexOf("is", 15).ToString()); // search from the 15th location
}
}
```

Thank You

HNDIT1012 Visual Application Programming

Week 8

Graphics object.

You can draw many different shapes and lines by using following methods of a Graphics object.

DrawLine

DrawArc

DrawClosedCurve

DrawPolygon

DrawRectangle

DrawEllipse.

Pen Class

Pen Defines an object used to draw lines and curves.

(Same as a drawing pen or pencil)

You need a paper and pen/pencil to draw a drawing. Similarly in C# you can imagine graphic object as a paper and you are going to draw some thing using a pen.

Pen has several properties including:

- Color and
- Width

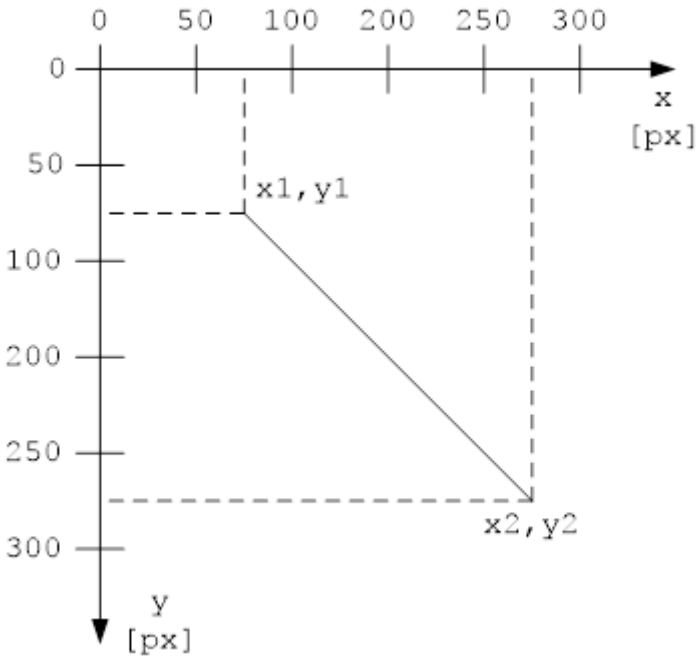
Coordinate System

To draw a line, start point and end point must be defined. Start point is set by the values of x1 and y1. End point is set by the values of x2 and y2.

In this case

x1 = y2 = 75 pixels and

x2 = y2 = 275 pixels.



DrawLine(Pen, Int32, Int32, Int32, Int32)

Draws a line connecting the two points specified by the coordinate pairs.

Parameters

pen
Pen
Pen that determines the color, width, and style of the line.

x1
Int32
The x-coordinate of the first point.

y1
Int32
The y-coordinate of the first point.

x2
Int32
The x-coordinate of the second point.

y2
Int32
The y-coordinate of the second point.

Example

The following code example is designed for use with Windows Forms, and it requires PaintEventArgs e, which is a parameter of the Paint event handler. The code performs the following actions:

Creates a black pen.

Creates the coordinates of the endpoints of the line.

Draws the line to the screen.

```
public void DrawLineInt(PaintEventArgs e)
{
    // Create a black pen with thickness 3 .
    Pen blackPen = new Pen(Color.Black, 3);
     // Create coordinates of points that define line.
    int x1 = 100;
    int y1 = 100;
    int x2 = 500;
    int y2 = 100;
    // Draw line to screen.
    e.Graphics.DrawLine(blackPen, x1, y1, x2, y2);
}
```

DrawEllipse(Pen, Int32, Int32, Int32, Int32)

Parameters

pen
Pen
Pen that determines the color, width, and style of the ellipse.
x
Int32
The x-coordinate of the upper-left corner of the bounding rectangle that defines the ellipse.
y
Int32
The y-coordinate of the upper-left corner of the bounding rectangle that defines the ellipse.
width
Int32
Width of the bounding rectangle that defines the ellipse.
height
Int32
Height of the bounding rectangle that defines the ellipse.

Example

The following code example is designed for use with Windows Forms, and it requires PaintEventArgs e, which is a parameter of the Paint event handler. The code performs the following actions:

Creates a black pen.

Creates the position and size of a rectangle to bound an ellipse.

Draws the ellipse to the screen.

```
private void DrawEllipseInt(PaintEventArgs e)
{
    // Create pen.
    Pen blackPen = new Pen(Color.Black, 3);
    // Create location and size of ellipse.
    int x = 0;
    int y = 0;
    int width = 200;
    int height = 100;
    // Draw ellipse to screen.
    e.Graphics.DrawEllipse(blackPen, x, y, width, height);
}
```

Thank You

HNDIT1012 Visual Application Programming

Week 9

FileDialog Class

Displays a dialog box from which the user can select a file. Some properties of FileDialog are:

CheckFileExists	Gets or sets a value indicating whether the dialog box displays a warning if the user specifies a file name that does not exist.
CheckPathExists	Gets or sets a value indicating whether the dialog box displays a warning if the user specifies a path that does not exist.
FileName	Gets or sets a string containing the file name selected in the file dialog box.
FileNames	Gets the file names of all selected files in the dialog box.
Filter	Gets or sets the current file name filter string, which determines the choices that appear in the "Save as file type" or "Files of type" box in the dialog box.
FilterIndex	Gets or sets the index of the filter currently selected in the file dialog box.
InitialDirectory	Gets or sets the initial directory displayed by the file dialog box.

System.IO.File Class

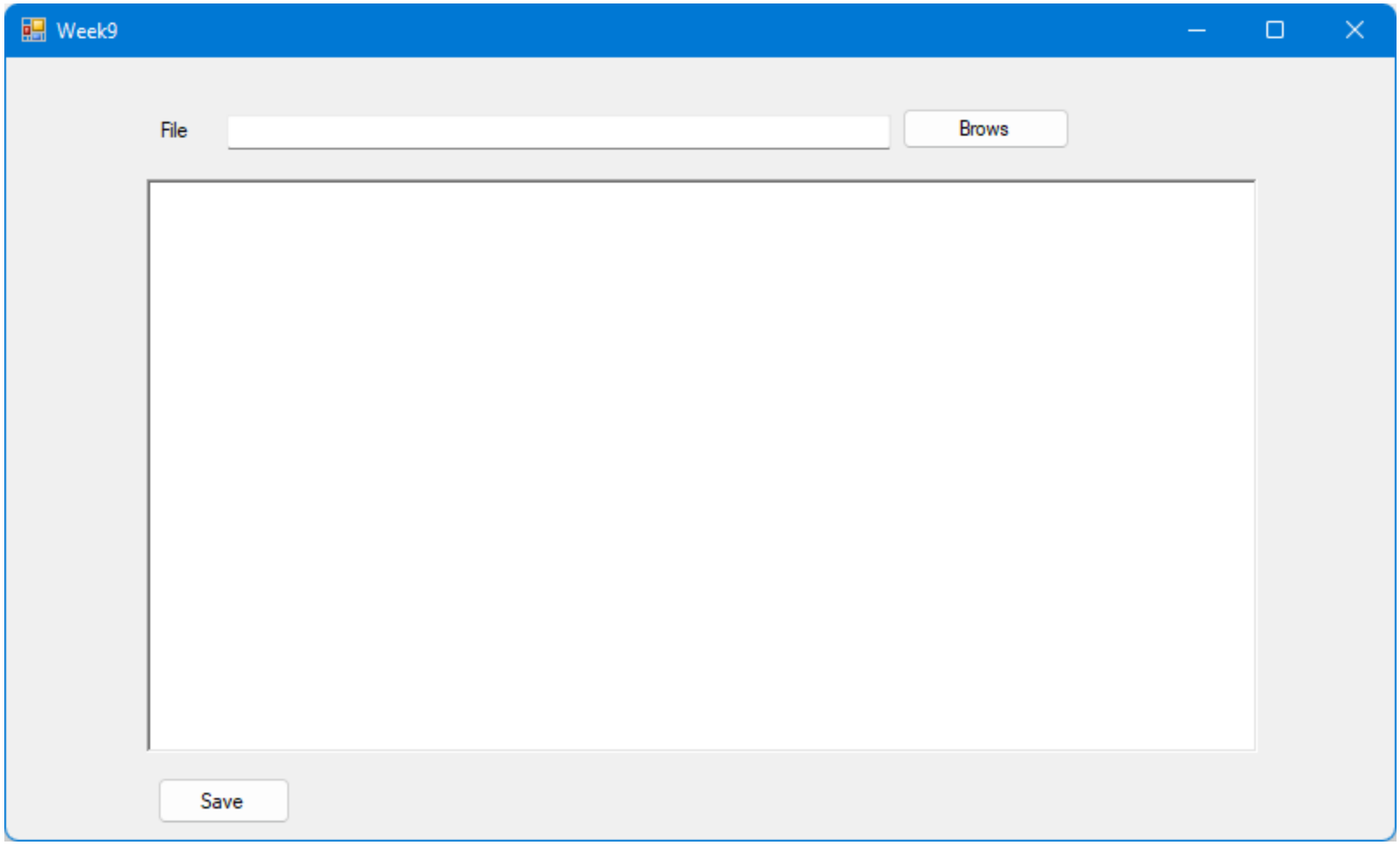
Provides static methods for the

- creation,
- copying,
- deletion,
- moving, and
- opening of a single file

Some Methods of File Class

- ReadAllText(fileName) - Opens a text file, reads all the text in the file, and then closes the file.
- ReadAllLines(fileName) - Opens a text file, reads all lines of the file, and then closes the file.
- ReadAllBytes(fileName) - Opens a binary file, reads the contents of the file into a byte array, and then closes the file.
- WriteAllText(fileName, textContent) - Creates a new file, writes the specified string to the file, and then closes the file. If the target file already exists, it is overwritten.

Example- Design a form as shown below



Reading all contents of a text file

```
private void btnBrows_Click(object sender, EventArgs e)
        {
            OpenFileDialog fileDialog = new OpenFileDialog();
            fileDialog.Filter = "txt files (*.txt)|*.txt|All files (*.*)|*.*";
            fileDialog.FilterIndex = 0;
            fileDialog.InitialDirectory = @“C:\asp c#"; // give your default directory

            // Reading all contents of a text file
            if (fileDialog.ShowDialog() == DialogResult.OK)
            {
                txtFile.Text = fileDialog.FileName;
                string readBuffer = System.IO.File.ReadAllText(fileDialog.FileName);
                richTextBox1.Text=readBuffer;
            }

        }
```

Reading a text file line by line

```
private void btnBrows_Click(object sender, EventArgs e)
        {
            OpenFileDialog fileDialog = new OpenFileDialog();
            fileDialog.Filter = "txt files (*.txt)|*.txt|All files (*.*)|*.*";
            fileDialog.FilterIndex = 0;
            fileDialog.InitialDirectory = @"E:\asp c#"; // give your default directory

            // Reading a text file line by line

            if (fileDialog.ShowDialog() == DialogResult.OK)
                {
                    txtFile.Text = fileDialog.FileName;
                    string[] readBuffer = System.IO.File.ReadAllLines(fileDialog.FileName);
                    for (int i = 0; i < readBuffer.Length; i++)
                        richTextBox1.Text = readBuffer[i] +"\n";
                }
        }
```

Reading a file byte by byte

```
private void btnBrows_Click(object sender, EventArgs e)
        {
            OpenFileDialog fileDialog = new OpenFileDialog();
            fileDialog.Filter = "txt files (*.txt)|*.txt|All files (*.*)|*.*";
            fileDialog.FilterIndex = 0;
            fileDialog.InitialDirectory = @"E:\asp c#";

            // Reading a file byte by byte

            if (fileDialog.ShowDialog() == DialogResult.OK)
            {
                txtFile.Text = fileDialog.FileName;
                byte[] readBuffer = System.IO.File.ReadAllBytes(fileDialog.FileName);
                foreach (byte b in readBuffer)
                    richTextBox1.Text += (char)b;
            }
        }
```

Saving a text File

Type the following code to the save button click event.

```
private void btnSave_Click(object sender, EventArgs e)
        {
            SaveFileDialog fileDialog = new SaveFileDialog();
            fileDialog.InitialDirectory = @"C:\Users\garig";
            fileDialog.Filter = "txt files (*.txt)|*.txt|All files (*.*)|*.*";
            //add default extension if the user omit extension
            fileDialog.AddExtension = true;
            fileDialog.DefaultExt = ".txt"; //default extension
            if (fileDialog.ShowDialog() == DialogResult.OK)
            {
                System.IO.File.WriteAllText(fileDialog.FileName,richTextBox1.Text);
            }
        }
```

Thank You

HNDIT1012 Visual Application Programming

Week 10

Introduction to DataSet in C#

DataSet is a disconnected architecture it represents the data in table structure which means the data into rows and columns. Dataset is the local copy of your database which exists in the local system and makes the application execute faster and reliable. DataSet contains collection of DataTables. DataSet works like a real database with an entire set of data which includes the constraints, relationship among tables, and so on. It will be found in the namespace “System. Data”.

How DataSet Works?

Let's understand the working procedure of DataSet in C# with example, We creating two data tables

Employee and

Salary tables

and then create data columns to add the columns into the tables and finally create data rows to add records into both the tables.

Create an Application

Example

Add two dataGrideView controls as shown in previous slide and insert a Button and labels as shown in previous slide.

Add the following code to the click event of the button.

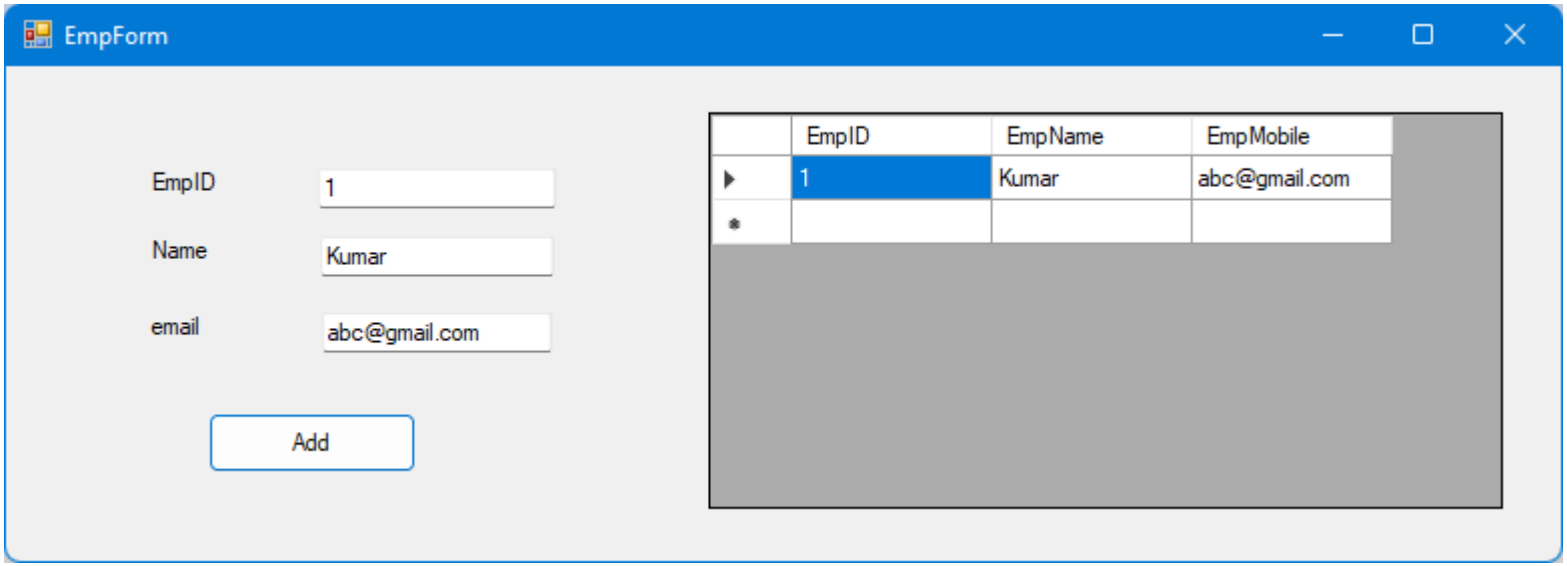
```
private void button1_Click(object sender, EventArgs e)
{
      // building the EmployeeDetails table using DataTable
      DataTable EmployeeDetails = new DataTable("EmployeeDetails");

      //to create the column and schema
      DataColumn EmployeeID = new DataColumn("EmpID", typeof(Int32));
      EmployeeDetails.Columns.Add(EmployeeID);
      DataColumn EmployeeName = new DataColumn("EmpName", typeof(string));
      EmployeeDetails.Columns.Add(EmployeeName);
      DataColumn EmployeeMobile = new DataColumn("EmpMobile", typeof(string));
      EmployeeDetails.Columns.Add(EmployeeMobile);
      //to add the Data rows into the EmployeeDetails table
      EmployeeDetails.Rows.Add(1001, "Andrew", "9000322579");
      EmployeeDetails.Rows.Add(1002, "Briddan", "9081223457");

      // to create one more table SalaryDetails
      DataTable SalaryDetails = new DataTable("SalaryDetails");
```

```
//to create the column and schema
   DataColumn SalaryId = new DataColumn("SalaryID", typeof(Int32));
   SalaryDetails.Columns.Add(SalaryId);
   DataColumn empId = new DataColumn("EmployeeID", typeof(Int32));
   SalaryDetails.Columns.Add(empId);
   DataColumn empName = new DataColumn("EmployeeName", typeof(string));
   SalaryDetails.Columns.Add(empName);
   DataColumn SalaryPaid = new DataColumn("Salary", typeof(Int32));
   SalaryDetails.Columns.Add(SalaryPaid);

   //to add the Data rows into the SalaryDetails table
   SalaryDetails.Rows.Add(10001, 1001, "Andrew", 42000);
   SalaryDetails.Rows.Add(10002, 1002, "Briddan", 30000);

   //to create the object for DataSet
   DataSet dataSet = new DataSet();

   //Adding DataTables into DataSet
   dataSet.Tables.Add(EmployeeDetails);
   dataSet.Tables.Add(SalaryDetails);
```

```
 dataGridViewEmployee.DataSource = EmployeeDetails;
   // Alternative ways
   /*
   dataGridViewEmployee.DataSource = dataSet.Tables[0];
   Or
   dataGridViewEmployee.DataSource = dataSet.Tables["EmployeeDetails"];
   */

dataGridViewSalary.DataSource = SalaryDetails;
   // Alternative ways
   /*
   dataGridViewSalary.DataSource = dataSet.Tables[1];
   Or
   dataGridViewSalary.DataSource = dataSet.Tables["SalaryDetails"];
   */
}
```

Adding data using Forms

Design a form as shown below:



Add the code as shown below

```
public partial class EmpForm : Form
    {
        DataTable EmployeeDetails;
        public EmpForm()
        {
            InitializeComponent();
            EmployeeDetails = new DataTable("EmployeeDetails");

            //to create the column and schema
            DataColumn EmployeeID = new DataColumn("EmpID", typeof(Int32));
            EmployeeDetails.Columns.Add(EmployeeID);
            DataColumn EmployeeName = new DataColumn("EmpName", typeof(string));
            EmployeeDetails.Columns.Add(EmployeeName);
            DataColumn EmployeeMobile = new DataColumn("EmpMobile", typeof(string));
            EmployeeDetails.Columns.Add(EmployeeMobile);
            dataGridView1.DataSource = EmployeeDetails;
        }

        private void btnAdd_Click(object sender, EventArgs e)
        {
            EmployeeDetails.Rows.Add(txtEmpID.Text, txtEmpName.Text, txtEmpMobile.Text);
        }
    }
```

Thank You