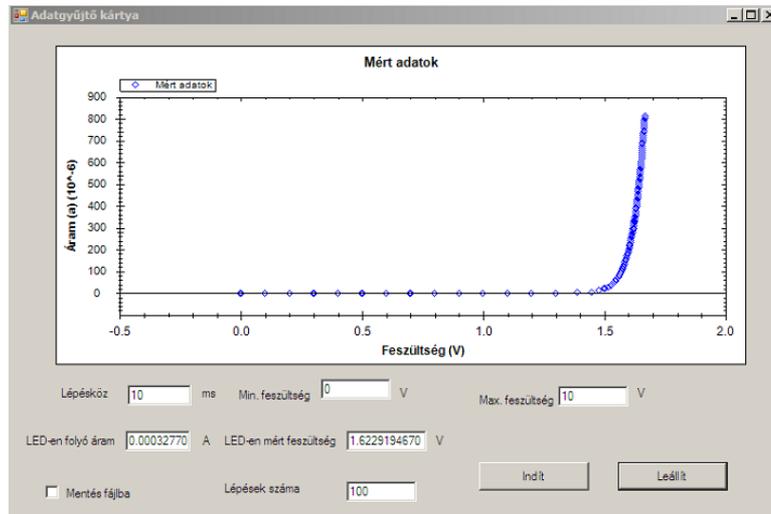


# Computer-controlled measurements



Instructor:

Gergő Fülöp

BME TTK, Department of Physics  
2023/2024 Spring semester



# Outline

## Day 1

- Setting the goal
- Basics of C# programming
  - Making a GUI
  - Syntax, flow control, etc.
  - Debugging in Visual Studio
  - File I/O

## Day 2

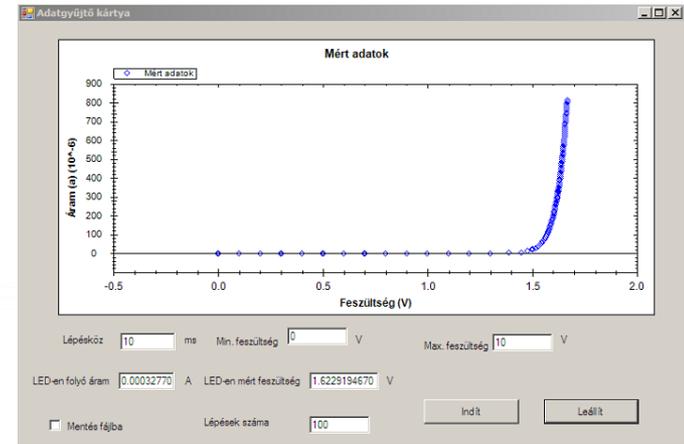
- Connecting instruments via serial port
  - Communication with instruments
  - Plot acquired data
- Solving complex measurement control and data analysis tasks
  - NI myDAQ data acquisition device

# Computer-controlled measurements

Tasks: Automated measurement, data acquisition, real-time data analysis

Devices: **Instrument:** measures a physical quantity;

**Computer:** data acquisition, display and analysis



# Computer-controlled measurements

Communication protocols:

- RS-232



- USB



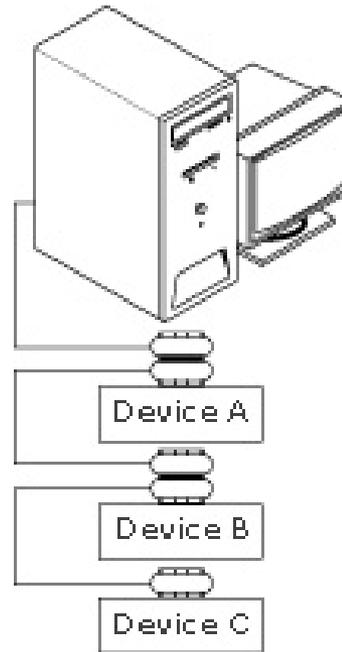
- LPT



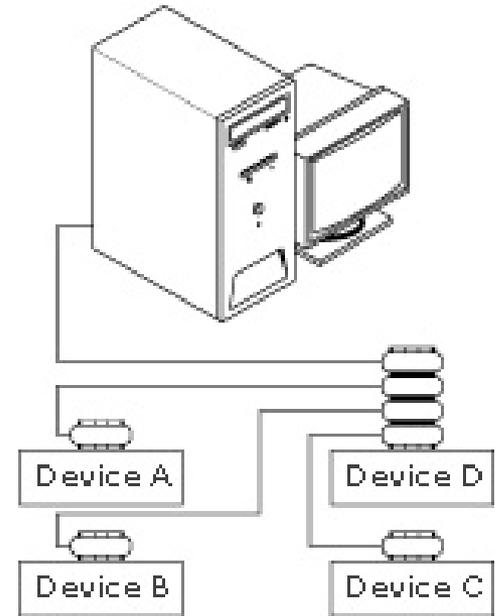
- GPIB



...



a. Linear Configuration



b. Star Configuration

Properties:

- Bandwidth
- Cost
- Connectivity, ...

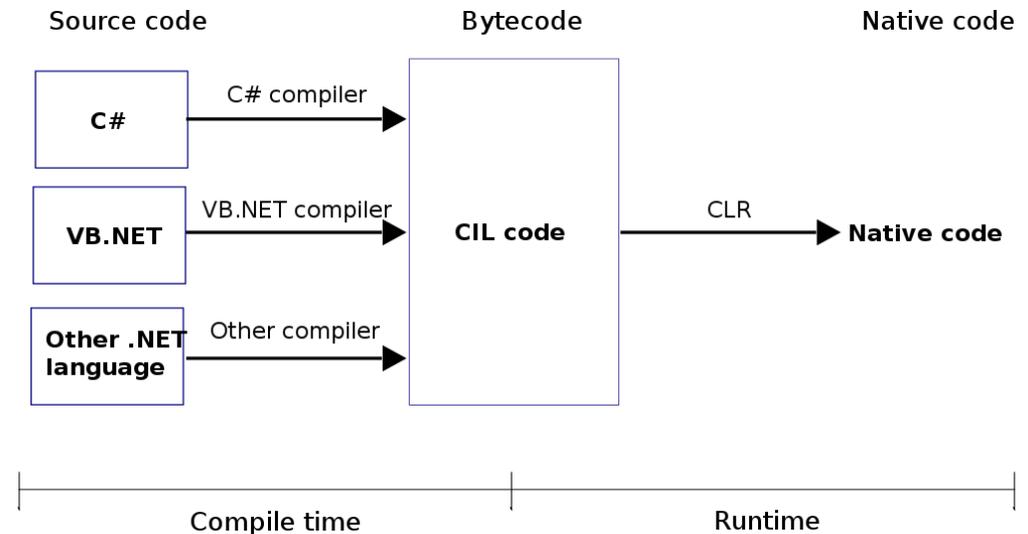
# C#

- Object-oriented, general purpose
- High-level
- .NET Framework



- Class Library
- CLR – Common Language Runtime

- MONO (Linux)



# Basics of Object-Oriented Programming

Object: abstract unit

Example: **Employee (class)**

Properties:

- First name
- Last name
- Position
- Salary
- Email address



Data

Methods:

- Increase salary
- Fire
- Promote

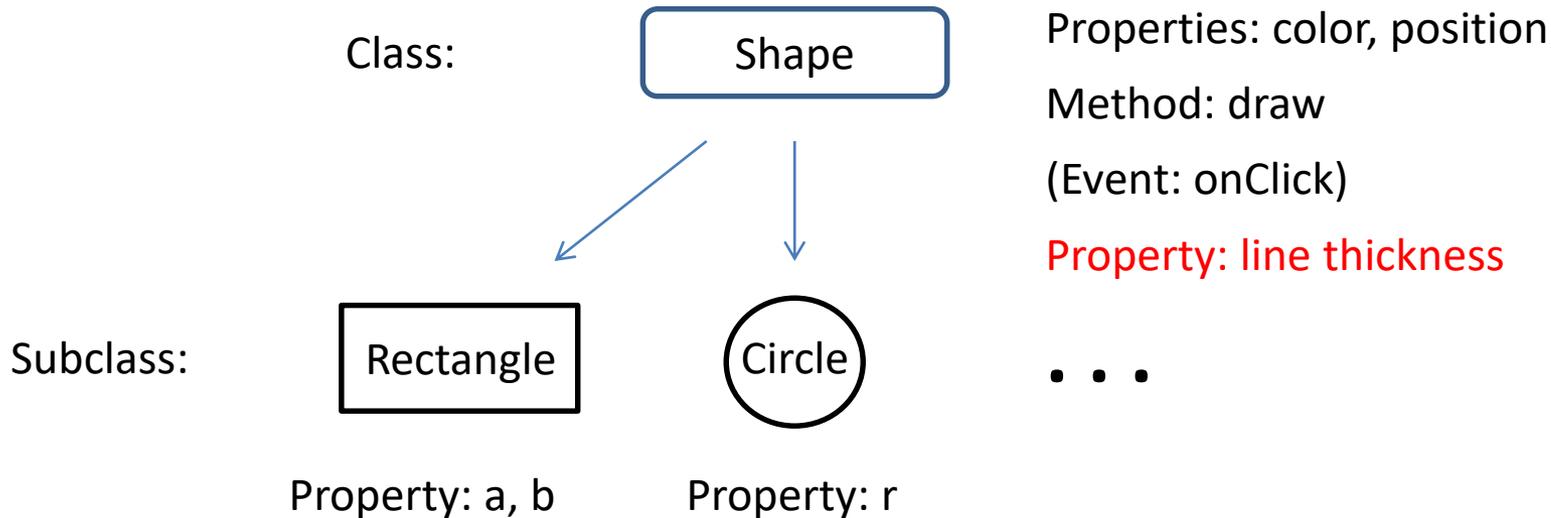


Logic/procedure/behavior

```
Employee Bill = new Employee("Bill Smith");  
Employee John = new Employee("John Doe");
```

# Basics of Object-Oriented Programming

Example: drawing program

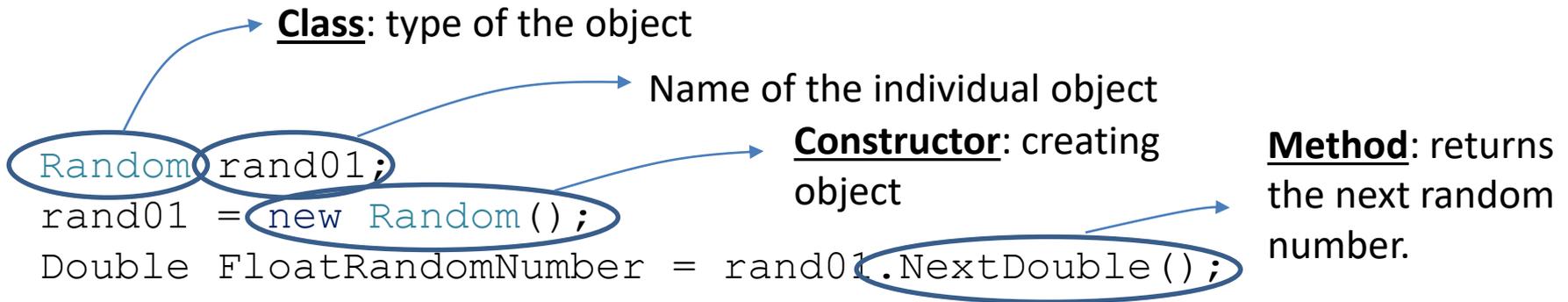


Summary:

- **Class**: definitions for the data format and available procedures;
- **Object**: instance of a class;
- **Method**: procedure associated with a message and an object;
- **Property**: data stored in the object;
- **Event**: an action or occurrence recognized by software – like user input.

# Basics of Object-Oriented Programming

Example: generating random numbers



```
Int32 IntRandomNumber = rand01.Next(MaxRandomNumber);
```

Overloading example:

`int Random.Next(int minValue, int maxValue)` (+ 2 overloads)  
Returns a random integer that is within a specified range.

Returns:  
A 32-bit signed integer greater than or equal to `minValue` and less than `maxValue`; that is, the range of return values includes `minValue` but not `maxValue`. If `minValue` equals `maxValue`, `minValue` is returned.

# Basics of Object-Oriented Programming

Example: button



**Property**: Text displayed on the button.

```
StartButton.Text="Start!"  
private void StartButton_Click(object sender, EventArgs e)  
{  
  
}
```

**Event**: Function, called when the user clicks on the StartButton.

# Your first C# program

## Create a new project

### Recent project templates

Windows Forms App (.NET Framework) C#

Search for templates (Alt+S)



[Clear all](#)

C#

Windows

Desktop



JUnit Test Project (.NET Core)

A project that contains NUnit tests that can run on .NET Core on Windows, Linux and MacOS.

C#

Linux

macOS

Windows

Desktop

Test

Web



Windows Forms App (.NET Framework)

A project for creating an application with a Windows Forms (WinForms) user interface

C#

Windows

Desktop



WPF App (.NET Framework)

Windows Presentation Foundation client application

C#

XAML

Windows

Desktop



WPF App (.NET)

Windows Presentation Foundation client application

C#

XAML

Windows

Desktop



WPF Custom Control Library (.NET)

Windows Presentation Foundation custom control library

C#

XAML

Windows

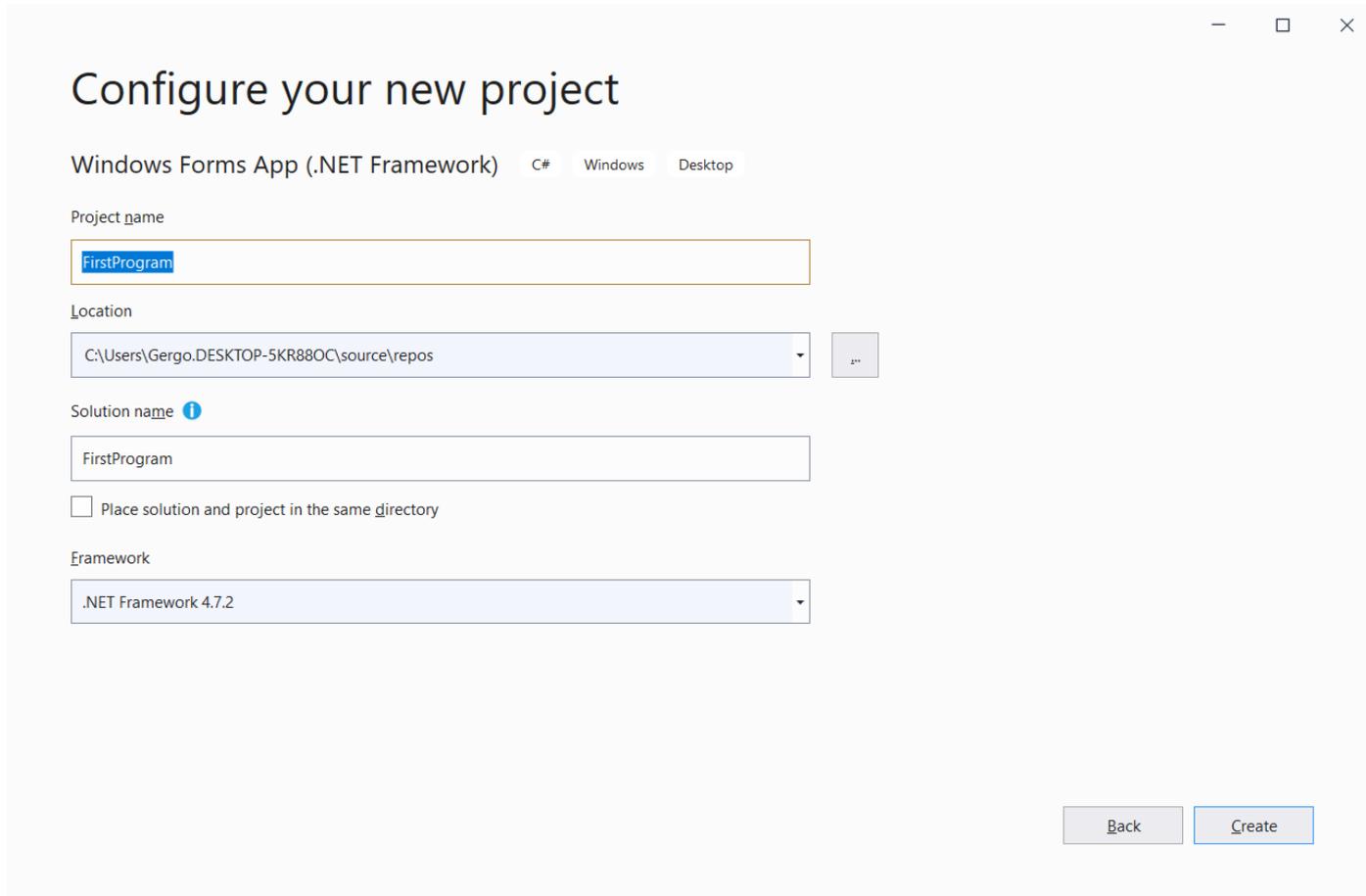
Desktop

Library

[Back](#)

[Next](#)

# Your first C# program



The image shows a screenshot of the 'Configure your new project' dialog box in Visual Studio. The dialog is titled 'Configure your new project' and has standard window controls (minimize, maximize, close) in the top right corner. Below the title, the project type is set to 'Windows Forms App (.NET Framework)' with sub-options for 'C#', 'Windows', and 'Desktop'. The 'Project name' field contains 'FirstProgram'. The 'Location' field shows the path 'C:\Users\Gergo.DESKTOP-5KR88OC\source\repos' with a browse button ('...'). The 'Solution name' field contains 'FirstProgram' and has an information icon. There is an unchecked checkbox for 'Place solution and project in the same directory'. The 'Framework' dropdown is set to '.NET Framework 4.7.2'. At the bottom right, there are 'Back' and 'Create' buttons.

Configure your new project

Windows Forms App (.NET Framework) C# Windows Desktop

Project name

FirstProgram

Location

C:\Users\Gergo.DESKTOP-5KR88OC\source\repos

Solution name ⓘ

FirstProgram

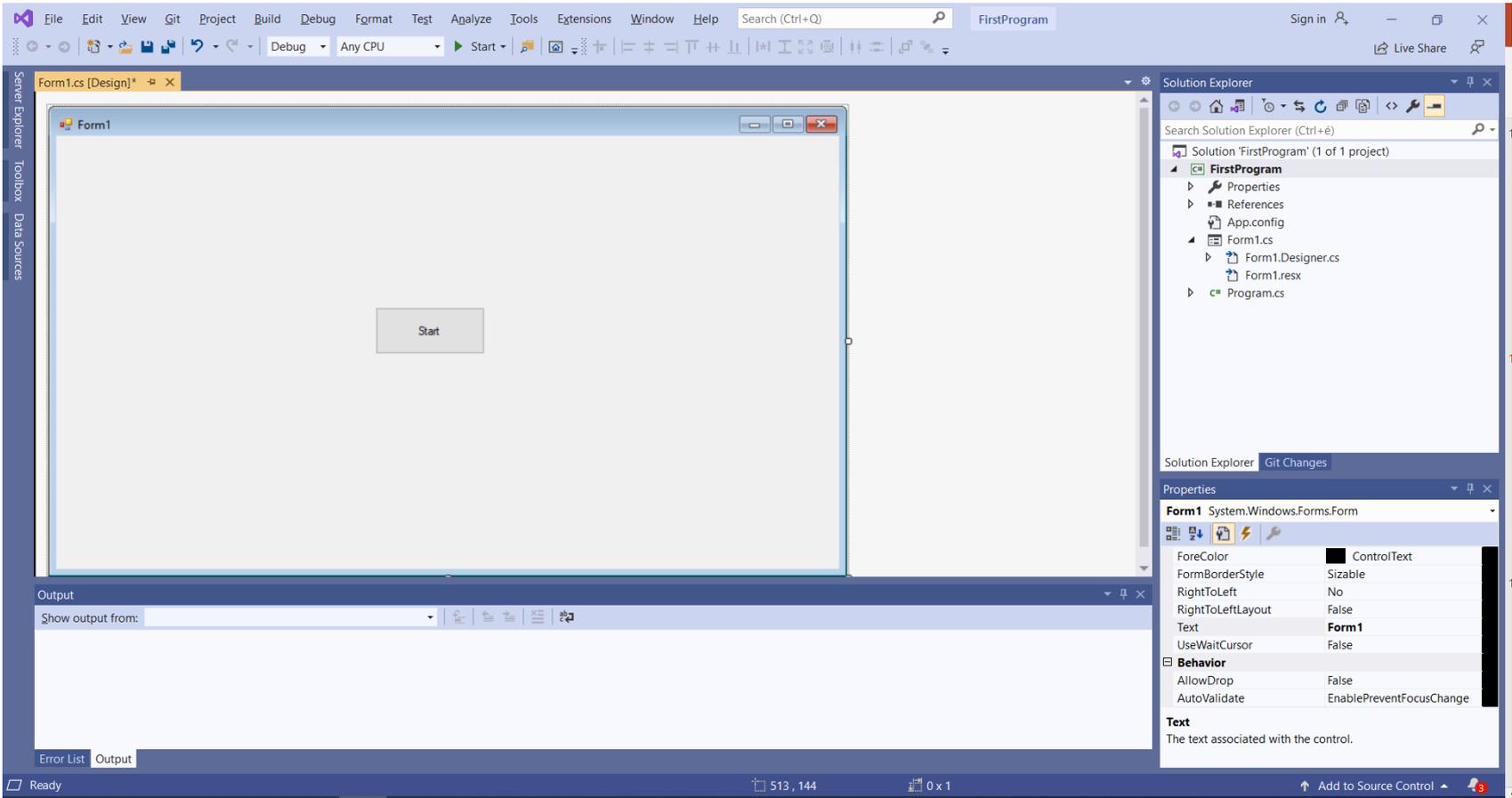
Place solution and project in the same directory

Framework

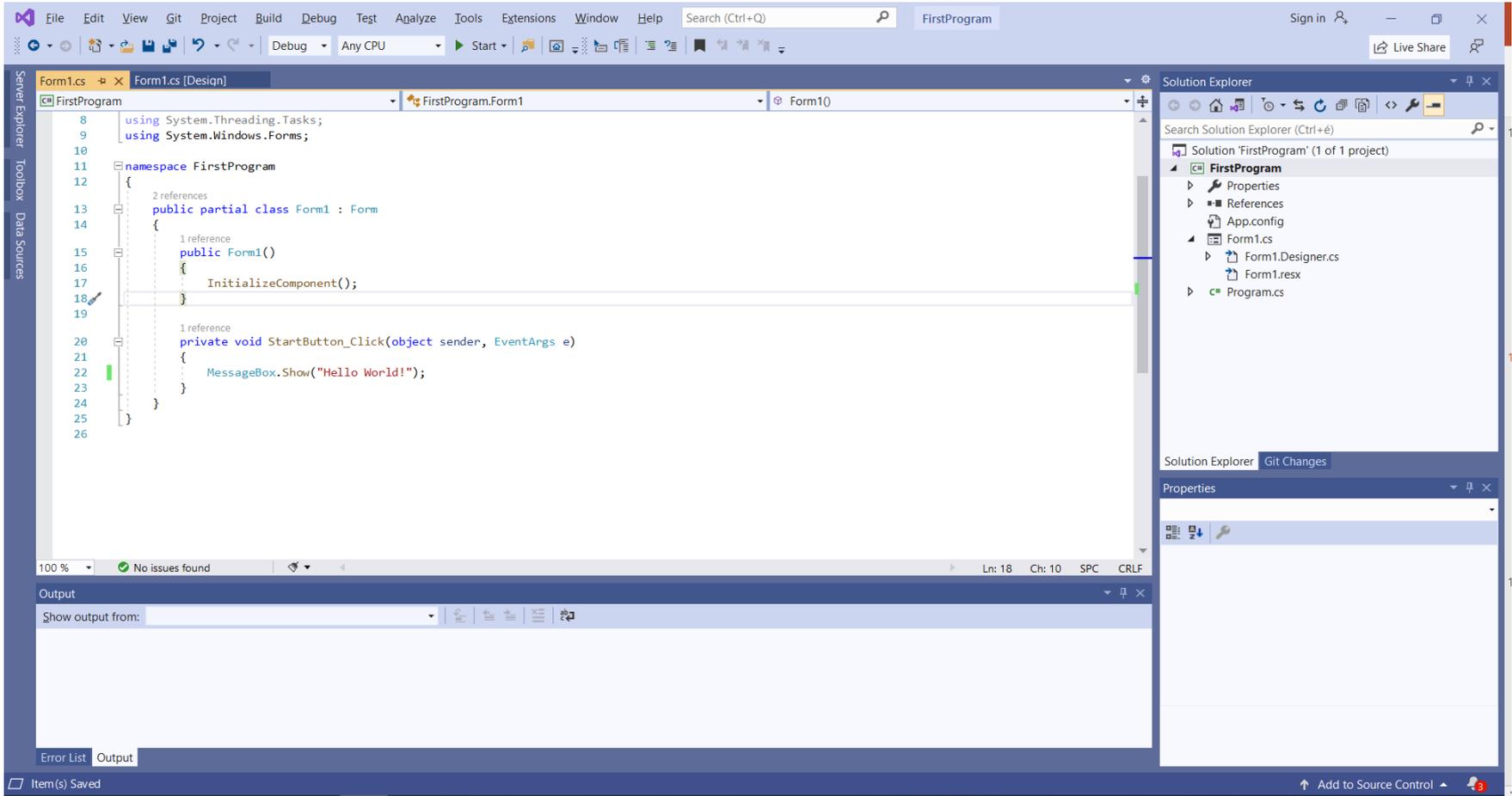
.NET Framework 4.7.2

Back Create

# Your first C# program



# Your first C# program



# Basics of Object-Oriented Programming

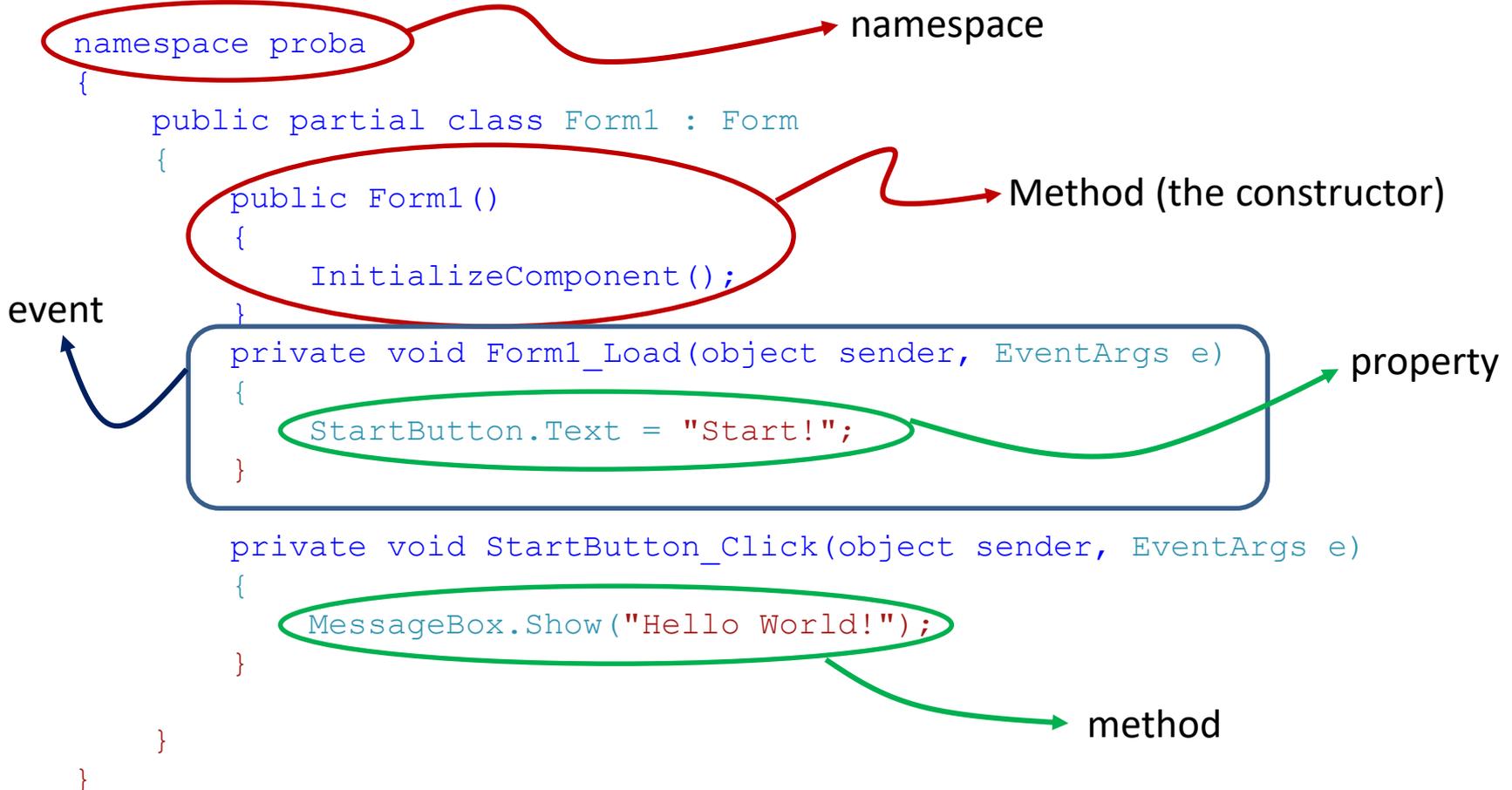
„Hello World!“ program:

```
namespace proba
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        private void Form1_Load(object sender, EventArgs e)
        {
            StartButton.Text = "Start!";
        }

        private void StartButton_Click(object sender, EventArgs e)
        {
            MessageBox.Show("Hello World!");
        }
    }
}
```

# Basics of Object-Oriented Programming

„Hello World!“ program:



# Button

```
using System.Windows.Forms;
```



## Properties

Name	Identifier of the object
Text	Text displayed on the button

## Methods

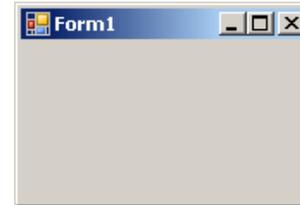
Hide	Hides Button on the user interface
Show	Shows Button on the user interface

## Events

Click	Function, called when user clicks on the button
-------	---

# Form

```
using System.Windows.Forms;
```



## Properties

Name	Identifier of the object
Text	Text displayed in the header of the Form

## Methods

Show	Opens Form
Close	Closes Form (Exits the program)

## Events

Load	Function, called when the program opens
FormClosing	Function, called when the program exits

# TextBox



```
using System.Windows.Forms;
```

## Properties

Name	Identifier of the object
Text	Text displayed in the textbox

## Methods

Hide	Hides the TextBox on user interface
Show	Shows the TextBox on the user interface

## Events

TextChanged	Function, called when the input text is changed
Click	Function, called when the user clicks on the TextBox

Typically, used in combination with a Button.

# Label



```
using System.Windows.Forms;
```

## Properties

Name	Identifier of the object
Text	Text displayed on the Label

## Methods

Hide	Hides Label on the user interface
Show	Shows Label on the user interface

## Events

VisibleChanged	Function, called when the label becomes Hidden/Shown
Click	Function, called when the user clicks on the label

# File IO

StreamWriter, StreamReader:

```
using System.IO;
```

## Constructor (Opening the file)

```
StreamWriter FileWriter = new StreamWriter("File Name");
```

```
StreamReader FileReader = new StreamReader("File Name");
```

## Methods

Write("Text")

Writes string to the file

WriteLine("Text")

Writes string ending with a new line character to the file

Read()

Reads next character from the file

ReadLine()

Reads next line from the file

Close()

Closes file

## Properties

EndOfStream

Bool value, signals when the end of the file is reached.

# File Dialogs

OpenFileDialog, SaveFileDialog :



```
using System.Windows.Forms;
```

## Methods

ShowDialog()	Opens the FileDialog window
Reset()	Clears the settings of the object

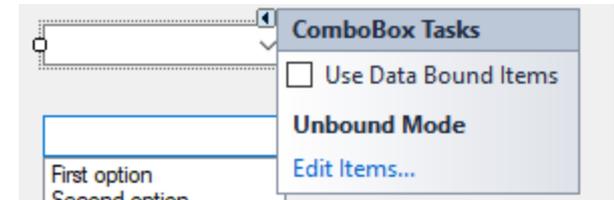
## Properties

FileName	Path and name of the chosen file.
Title	Header of the dialog window
InitialDirectory	Default directory
DefaultExt	Default file extension

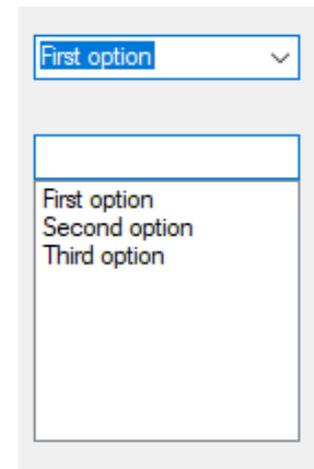
Usage: drop them on the GUI first  
See example code at the file I/O examples

# ComboBox

- Populate the options:
  - Through the GUI, or
  - Items Collection (e.g. `comboBox1.Items`)



- `DropDownStyle`
  - `DropDown`
  - `Simple`
- Get selected item



```
string selected = comboBox1.GetItemText(comboBox1.SelectedItem);
```

# NumericUpDown

- Properties

- Value
- Minimum
- Maximum
- DecimalPlaces
- Increment



- Note: type is decimal

# Examples

Measure time:

```
using System.Diagnostics;  
// ...
```

```
Stopwatch sw = new Stopwatch();
```

```
sw.Start();
```

```
// ...
```

```
sw.Stop();
```

```
double elapsed_time = sw.Elapsed.TotalMilliseconds;
```

Wait (makes the program hang):

```
System.Threading.Thread.Sleep(5000); // time in ms
```

# Examples

Generating random numbers:

```
// This code executes once  
Random rand01;  
rand01 = new Random();
```

```
•  
•  
•
```

```
// This code executes whenever a new random number  
needs to be generated  
Double RandomNumber = rand01.NextDouble();
```

# Read from File

Example with StreamReader:

```
using System.IO;
```

```
...
```

```
StreamReader reader = new StreamReader("filename.txt");  
string line;
```

```
while ((line = reader.ReadLine()) != null)  
{  
    TextBox1.AppendText(line);  
}  
reader.Close();
```

# Write to File

StreamWriter, SaveFileDialog example:

```
using System.IO;
```

```
...
```

```
// The SaveFileDialog object must be added in the designer
```

```
...
```

```
if(sfDialog.ShowDialog() == DialogResult.OK)
```

```
{
```

```
    StreamWriter writer = new StreamWriter(sfDialog.FileName);
```

```
    writer.WriteLine("your text");
```

```
    writer.Close();
```

```
}
```

# C# basics

Declaration:

```
int i;
```

Initialization:

```
i = 5;
```

Int32 ↔ int, Int64 ↔ long

Declaration + Initialization:

```
double j = 1.5;
```

## Array

- Sequence of elements
- Elements must be the same type
- Fixed size
- Refer to them by their index (zero-based)

```
double[] data = new double[16];  
data[0] = 1.5;  
data[15] = 2.3;
```

```
string[] daysOfWeek = { "Monday", "Tuesday", ..., "Sunday" };
```

Sorting: `Array.Sort(data)` // sorts in-place

# C# basics

Functions:

```
private Int32 Function(arglist)
{
    ...
}
```

private: can only be accessed within the class  
public: can be accessed outside of the class

Calling a function:

```
Int32 x = Function(arglist);
```

Type conversion:

```
x = Convert.ToDouble(Object);
string = Convert.ToString(Object);
i = Convert.ToInt(Object);
...
```

String:

```
string Text = "Hello";
int length = Text.Length;
string Part = Text.Substring(start, length);
int index = Text.IndexOf(char);
Text = Object.ToString("Format");
```

# C# basics

## string manipulation

```
string Text = " apple ";
```

## Length of a string:

```
int length = Text.Length;
```

Trim(): removes white space characters from the beginning and end of the string

TrimStart(), TrimEnd(): removes whitespace characters only from the beginning or end of the string

```
newText = Text.Trim(); // "apple"  
newText = Text.TrimStart(); // "apple "  
newText = Text.TrimEnd(); // " apple"
```

Substring():

```
// Text.Substring(start, length);  
newText = Text.Substring(0, 4) // " ap"
```

Split():

```
string text = "6+3";  
string[] numbers = text.Split('+');  
// numbers[0] = "6"  
// numbers[1] = "3"
```

# C# basics

string manipulation

Concatenating strings: "Hello World!" = "Hello" + " " + "World!"

IndexOf() : index of the searched character (returns -1 if character is not found).

```
int index = Text.IndexOf('p'); // index=4
```

ToString() :

```
// Text = Object.ToString("Format");  
double num = 5.0133;  
Text = num.ToString("0.00"); // fixed 2 decimal places, 5.01
```

# C# basics

Character:

```
char c = 'g';  
c = (char)103;           //ASCII 'g' character
```



Special characters:

```
char c;  
c = '\t';           // tabulator  
c = '\n';          // new line  
c = '\r';          // carriage return  
c = '\\';          // backslash  
c = '\'';          // quotation marks  
c = '\"';          // double quotation marks
```

# C# basics

## if branching:

```
int seconds = 0;
int minutes = 0;
...
if (seconds == 59)
{
    seconds = 0;
    minutes++;
}
else
    seconds++;
```

## Multiple branches:

```
if (day == 0)
    dayName = "Sunday";
else if (day == 1)
    dayName = "Monday";
...
else if (day == 6)
    dayName = "Saturday";
else
    dayName = "unknown";
```

## switch branching:

```
switch (day)
{
    case 0 :
        dayName = "Sunday";
        break;
    case 1 :
        dayName = "Monday";
        break;
    case 2 :
        dayName = "Tuesday";
        break;
    ...
    default :
        dayName = "Unknown";
        break;
}
```

- Only for internal datatypes (pl. int, string)
- Value has to be compared to a constant

# C# basics

while loop:

```
int i = 0;
while (i < 10)
{
    MessageBox.Show(i.ToString());
    i++;
}
```

“Do while” loop:

```
int i = 0;
do
{
    MessageBox.Show(i.ToString());
    i++;
}
while (i < 10);
```

for loop:

```
for (int i = 0; i < 10; i++)
{
    MessageBox.Show(i.ToString());
}
```

Initializing multiple variables:

```
for (int i = 0, j = 10; i <= j; i++, j--)
{
    ...
}
```

# C# basics

Math class

## Mathematical functions:

Math.Cos(rad) : cosine function

Math.Sin(rad) : sine function

Math.Min(num1, num2) : returns the smaller number

Math.Abs(num) : absolute value

Math.Pow(base, exponent) : Exponential function

Math.Exp(x) :  $e^x$

Math.Round(num, int decimals) : rounding numbers

Math.Sqrt(num) : square root

...

## Mathematical constants:

Math.PI:  $\pi$

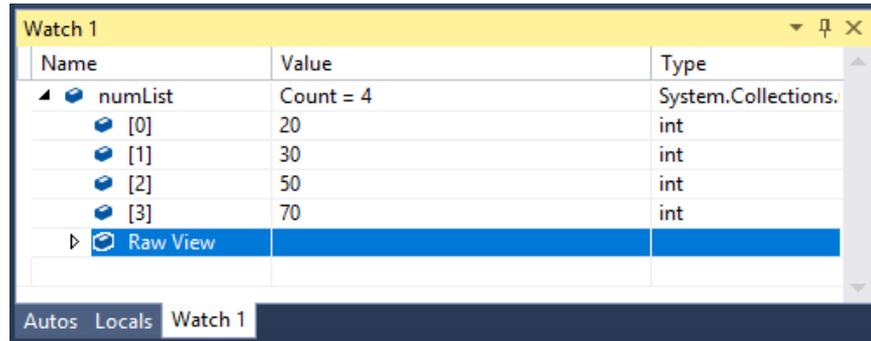
Math.E: e

# Lists in C#: <list>

Lists: dynamic arrays in C#

Example: list containing integer numbers

```
List<int> numList = new List<int>();  
numList.Add(20);  
numList.Add(30);  
numList.Add(50);  
numList.Add(70);
```



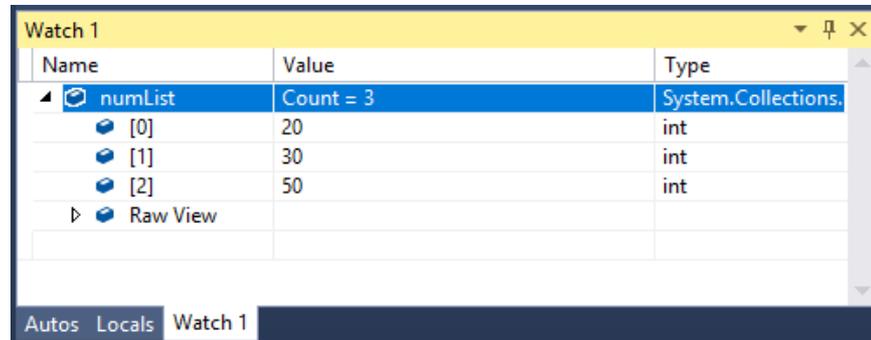
Watch 1

Name	Value	Type
numList	Count = 4	System.Collections.IList
[0]	20	int
[1]	30	int
[2]	50	int
[3]	70	int
Raw View		

Autos Locals Watch 1

```
int length = numList.Count; // length = 4  
int item = numList[2]; // item = 50
```

```
numList.RemoveAt(3);
```



Watch 1

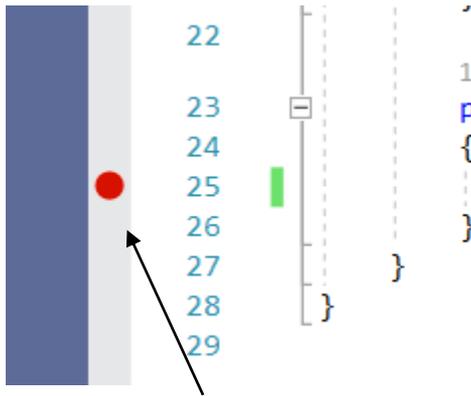
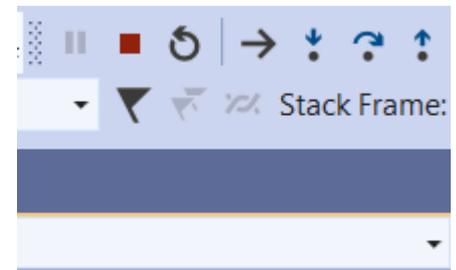
Name	Value	Type
numList	Count = 3	System.Collections.IList
[0]	20	int
[1]	30	int
[2]	50	int
Raw View		

Autos Locals Watch 1

Further methods: Clear(), Find(), Sort() ...

# Debugging

Control the execution



```
1 reference
private void StartButton_Click(object sender, EventArgs e)
{
    MessageBox.Show("Hello World!");
}
```

breakpoint

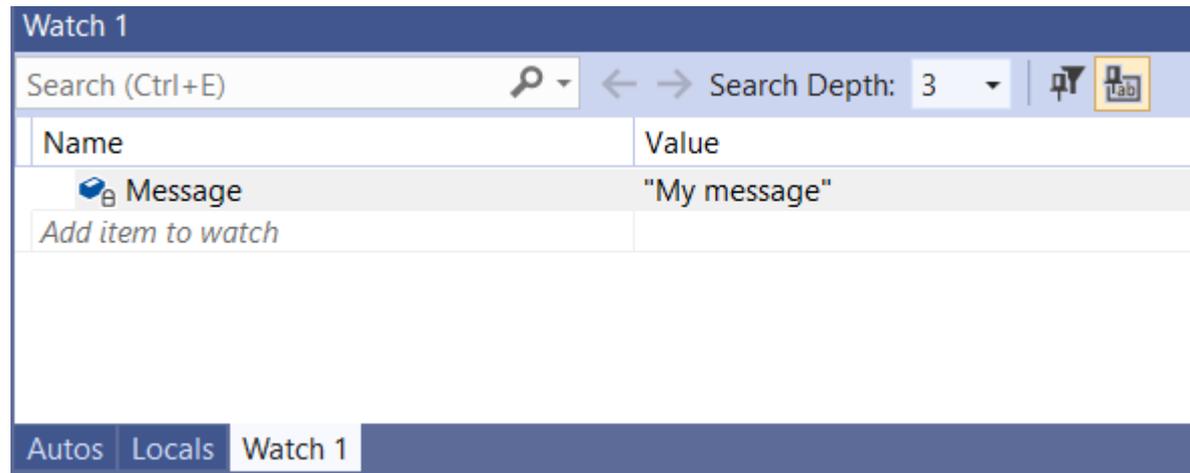
Tracking the value of a variable:  
In debug mode, right click on the variable → Add watch

See also:

- Conditions
- Actions

Watch window:

- Evaluating expressions
- Manipulating objects



# Scope of variables

```
namespace helloworld
{
    public partial class Form1 : Form
    {
        string globalText = "Hello World!";

        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            MessageBox.Show(globalText);
        }

        private void button1_Click(object sender, EventArgs e)
        {
            string text = "Message";
            MessageBox.Show(text);
        }
    }
}
```

Can be accessed by every function in the Class



Can only be accessed from the button1\_Click() function



# Good practices

"Programs must be written for people to read, and only incidentally for machines to execute." — Harold Abelson

## Naming practices, conventions

- Latin letters, English names
- Not too long, not too short
- Descriptive
- Avoid:
  - abbreviations (they are ambiguous, e.g. mon)  
acceptable: i, j, k in loops
  - dropping letters (e.g. ltrrs, nmftrls)
  - puns, jokes
  - numeric suffixes (e.g. increase1(), increase2() )
- Casing
  - camelCase (e.g. firstName, age, startIndex, lastNegativeNumberIndex)
  - PascalCase (e.g. SendEmail(), CarColor)

# Good practices

## Magic numbers

Avoid:

```
int deadline = now + 604800;
```

Instead:

```
const int SecondsPerMinute = 60;  
const int SecondsPerHour = 60 * SecondsPerMinute;  
const int SecondsPerDay = 24 * SecondsPerHour;  
const int SecondsPerWeek = 7 * SecondsPerDay;  
  
int deadline = now + SecondsPerWeek;
```

## Make it clear what units you are using

**Avoid duplicate code** (DRY: Don't Repeat Yourself) → make a function

# Pitfalls

## 1. Dividing integers:

```
double d;  
d = 4/5; // d = 0  
d = (double) 4/5; // d = 0.8
```

2. `Convert.ToDouble()` depends on the Windows locale settings (decimal separator).
3. `Textbox.TextChanged()` event is generated every time a character is changed.
4. Objects cannot be converted to a number, e.g. `Convert.ToDouble(TextBox1)`.  
Correctly: `Convert.ToDouble(TextBox1.Text)`
5. Don't copy code from the PDF file. (Special characters get messed up.)
6. When using an example code, make sure that the correct linkage exists (e.g. the function is set as an event function)

# Programming exercises

1. When the user pushes a button, change the color of a graphic object on the user interface. Try the debugging features of Visual Studio. (Setting the color: e.g. `StartButton.BackColor = Color.Blue;`)
2. Write the Fibonacci series into a text file. The user defines how many numbers to write to the file using a **TextBox** (e.g. to write the first 100 numbers of the series). Use a **SaveFileDialog** for selecting the file where the data will be written.
3. Make a program which evaluates the definite integral  $\int_a^b \sin(x) dx$  numerically. Use the trapezoidal rule:

$$\int_a^b f(x) dx \approx \sum_{k=1}^N \frac{f(x_{k-1}) + f(x_k)}{2} \Delta x_k.$$

Make GUI elements for the user to input all necessary parameters. Check your program with  $a = 0, b = \pi$  and  $a = 0, b = 2\pi$ .

# Programming exercises

4. Create a program which solves the equation  $ax + b = \sin(x)$  numerically.

Use Newton's method to find the roots of  $f(x)$ :

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Make GUI elements for the user to input all necessary parameters. Make it possible for the user to make  $N$  iterations with a single button click.

How many distinct solutions can you find for  $a = 0.5$ ,  $b = -0.1$ ?