TrabHCI
Technologies to Reduce the Access Barrier in Human Computer Interaction
Erasmus Intensive Programme IP29588-1-1731-10

.NET Framework, C# and a little bit of WPF

2

Ivan Bernabucci
University Roma TRE
i.bernabucci@uniroma3.it

.NET Framework, C# and a little bit of WPF

OVERVIEW

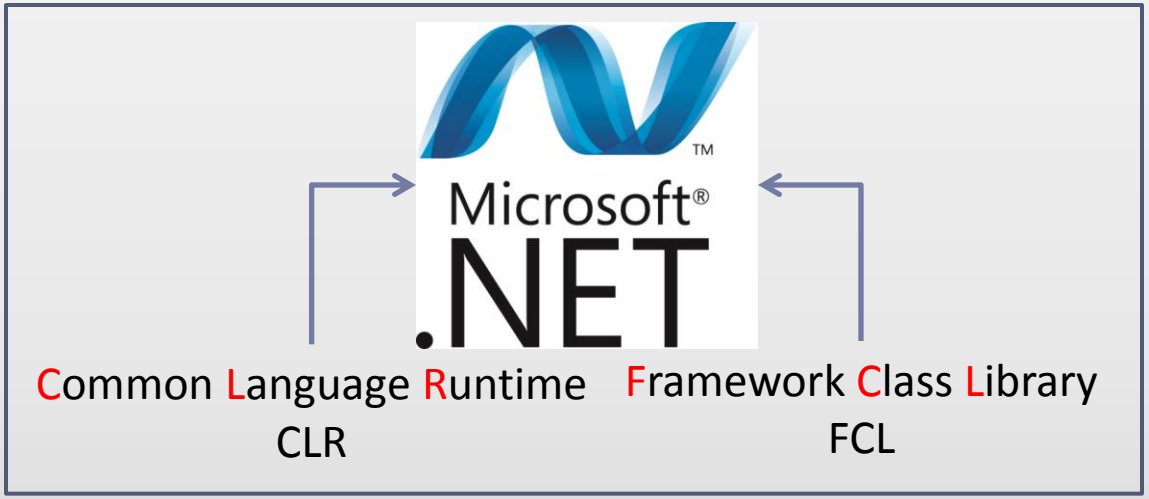
- What is .NET?
- What is FCL?
- What is CLR?
- What is C#?
- Basic Expressions and Operators
- Creating Project with Visual Studio
- Exploring WPF

.NET Framework, C# and a little bit of WPF

.NET

- Generic Software Framework

APPLICATION



- Like JAVA Virtual Machine MUST be installed

CLR

Execution Environment for Windows Applications

Responsible for:

- Bring application to life
- Manage it while it is executing
- Tear down the application when it is finished or has unrecoverable error

Services provided during application management:

- Memory Management
- Security
- Operating system and hardware indipendence
- Language Indipendence

.NET Framework, C# and a little bit of WPF

Memory Management

The CLR actively track all the object running and requested by the application, not like (original) C++. It will close everything and free memory cause it will know when you have done with a particular resource

Security

In some cases the application has a very restricted sandbox and cannot access file system areas. And ensures that the application does not read and write memory that does not belongs to the application

Operating system and hardware indipendence

It's a like a virtual machine cause it virtualize the execution environment, abstracting the operating system, the number of processors

Language Indipendence

This means that there is a common runtime engine that all the .NET languages share together, and that is possible to exploits components of other languages (F#, Visual Basic, Delphi.Net, IronPython.NET, J#)

FLC

Library of functionalities to build applications

It contains thousands and thousands of CLASSES

Base Class Library inside the .NET Framework that handles low-level operations such as:

- Database access
- File I/O
- Threading
- …

.NET Framework, C# and a little bit of WPF

Designing and developing Desktop Applications

Designing and developing Web Applications

Designing and developing Web Services

.NET Framework, C# and a little bit of WPF

Like the *Virtual Machine* in Java, the .NET Frameworks compiler provides an intermediate layer between the programming language and the assembly code: *Intermediate Language* (like the *bytecode*) which will be then used by the .NET framework in run-time execution.

This IL is the managed code (it can be .dll or .exe).

.NET Framework, C# and a little bit of WPF

An example of IL:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            Calc c = new Calc();
            Console.WriteLine("3 + 5 is {0}",c.Add(3,5));
        }
    }

    class Calc
    {
        public int Add(int x, int y)
        {
            return (x + y);
        }
    }
}
```

.NET Framework, C# and a little bit of WPF

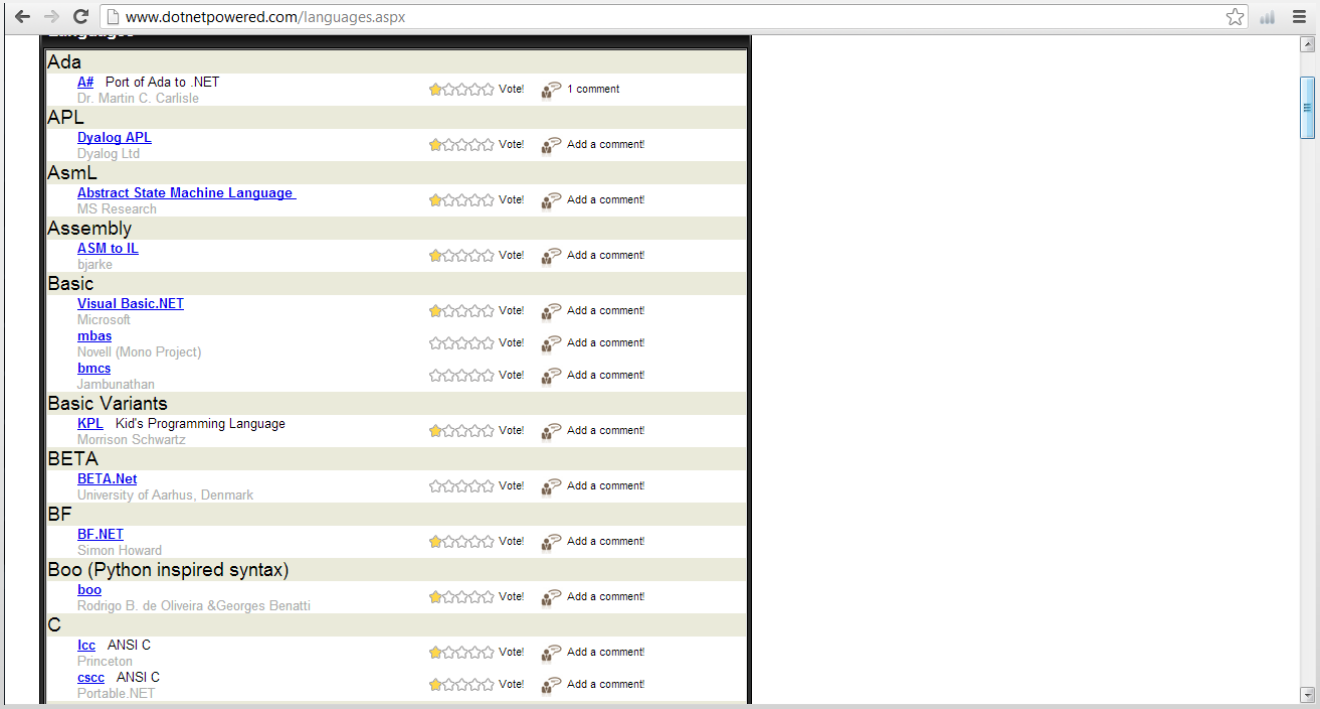
If we use the ildasm.exe and we open the Add method of the calculator this is what we see -> non platform-specific instructions

```
.method public hidebysig instance int32  Add(int32 x,
                                             int32 y) cil managed
{
  // Code size       9 (0x9)
  .maxstack  2
  .locals init ([0] int32 CS$1$0000)
  IL_0000:  nop
  IL_0001:  ldarg.1
  IL_0002:  ldarg.2
  IL_0003:  add
  IL_0004:  stloc.0
  IL_0005:  br.s       IL_0007
  IL_0007:  ldloc.0
  IL_0008:  ret
} // end of method Calc::Add
```

.NET Framework, C# and a little bit of WPF

There are many .NET compilers for different languages like Smlltalk, Cobol, Pascal

http://www.dotnetpowered.com/languages.aspx



.NET Framework, C# and a little bit of WPF

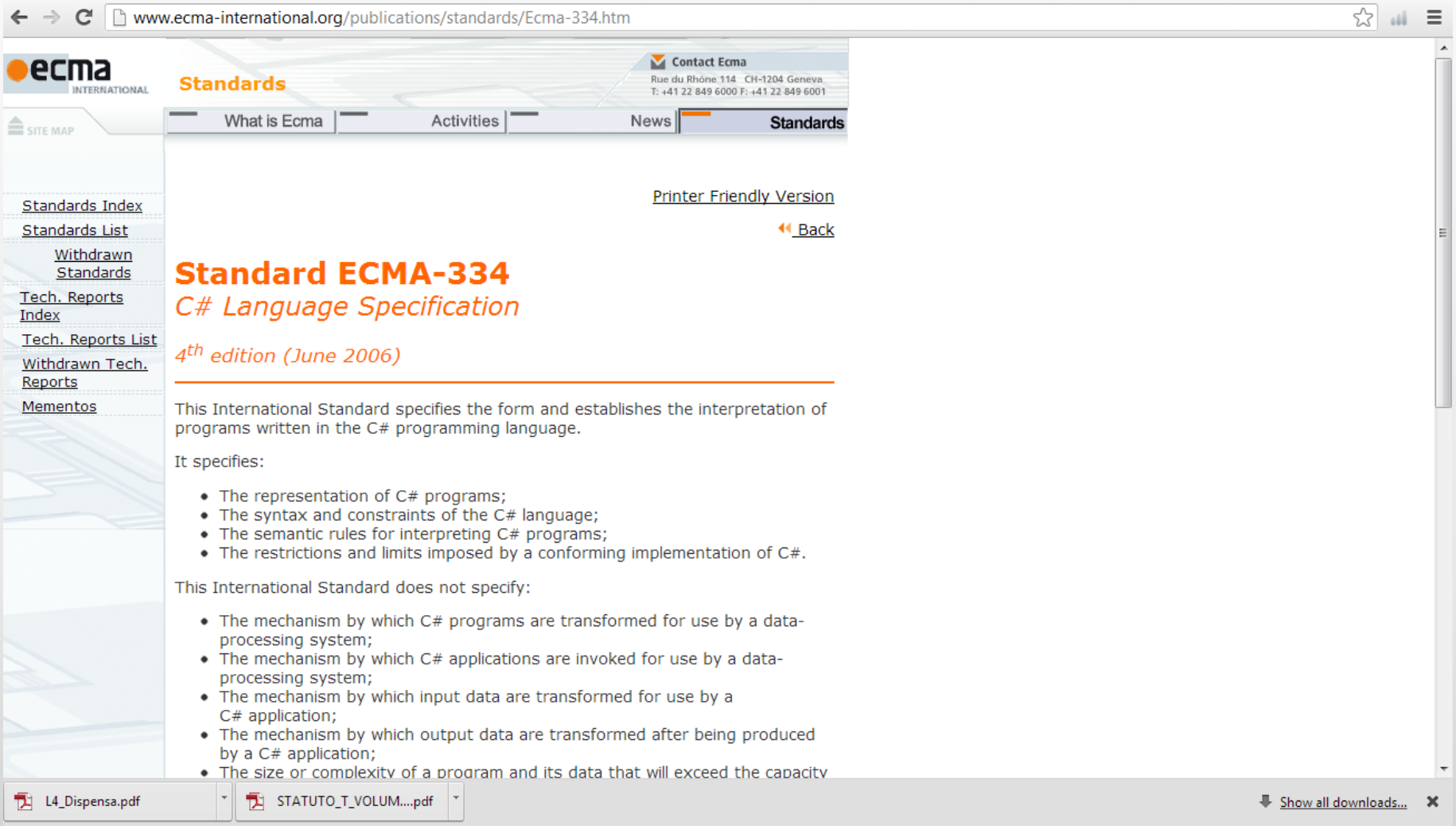
What is C#?

A standardized language to create .NET components

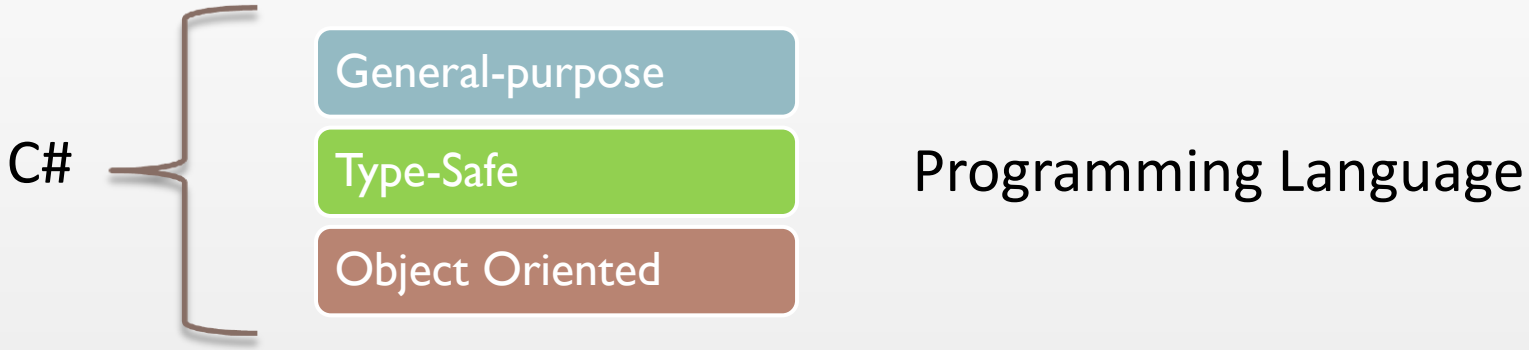
A. Standardized by ECMA
B. Create applications, services, reusable libraries
C. Syntax is similar to C++ and Java

A -> Microsoft took the semantic rules and the syntax and registered them in ECMA international (an international standard organization)

.NET Framework, C# and a little bit of WPF



.NET Framework, C# and a little bit of WPF



C# has some features in his structure:

- Unified type system: all types derive from a base type
- There are different types: objects, interfaces, structures, enumerations (like Java) and delegates!
- Function members: methods, events, properties

.NET Framework, C# and a little bit of WPF

The C# command line compiler

- Transform C# code in Microsoft Intermediate Language (MSIL)
- Produces an assembly (*.dll, *.exe)

.NET Framework, C# and a little bit of WPF

C# derives, like Java, the main features of C++ simplifying several aspects:

- No pointers required
- Automatic memory management through Garbage Collection
- Use of collections (List, Queue, …)
- Lambda expressions

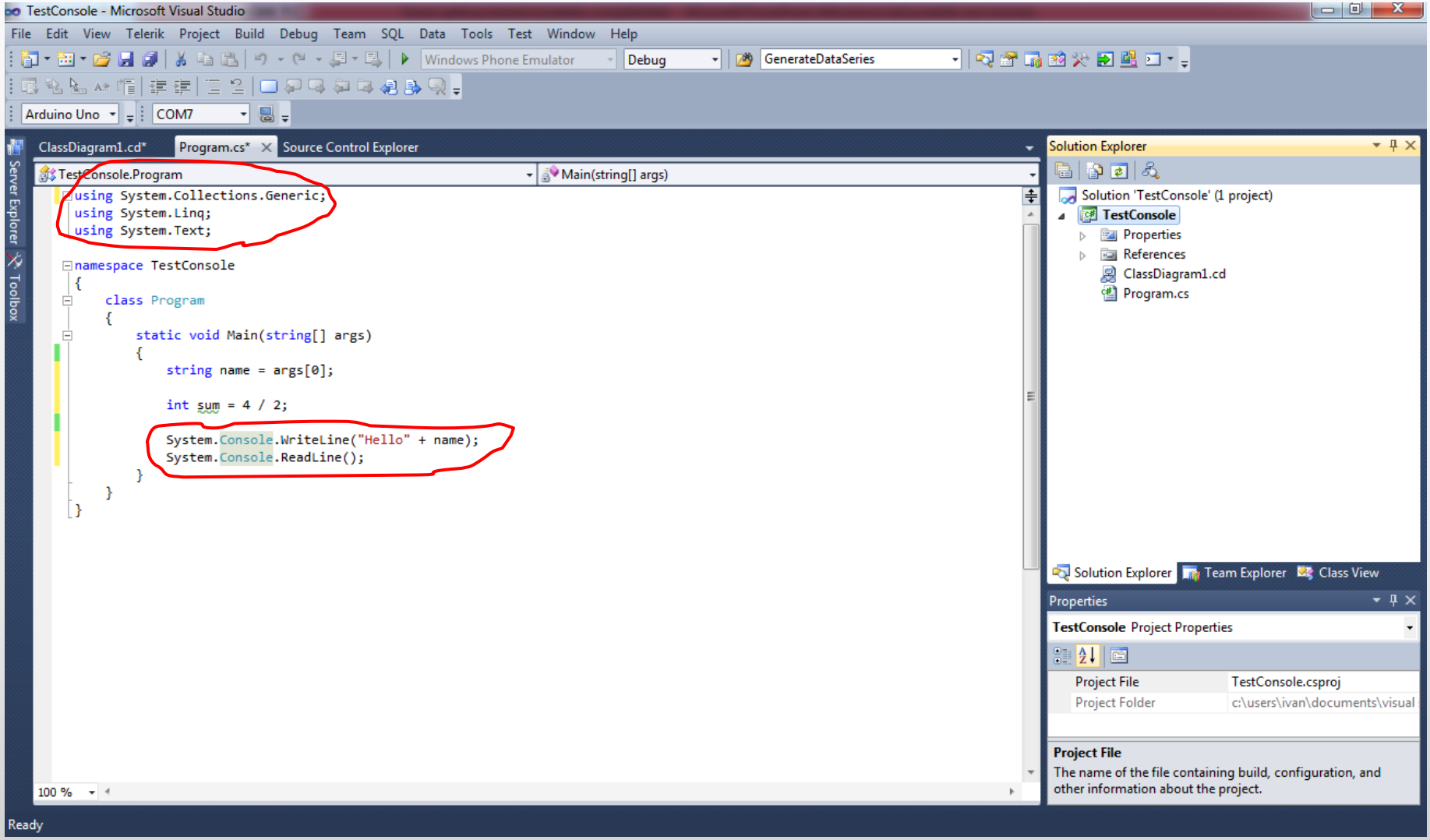
.NET Framework, C# and a little bit of WPF

Build Applications with Visual Studio 2010

Even if it's possible to write in Notepad and compile with the prompt command csc.exe

(ex: csc /target:exe Car.cs)

.NET Framework, C# and a little bit of WPF



VISUAL STUDIO

Integrated Development Environment (IDE)

- Edit C# (and other supporting) files
- Runs the C# compiler
- Debugging
- Testing

The IDE Visual Studio offer some advantages:

- Support for visual design
- Intellisense

SOLUTION EXPLORER WINDOW

Contains at least one project

- Contains one or more source code files
- Each project produces an assembly

Projects organized under a solution

- Manages multiple applications or libraries

TYPES

C# is strongly typed

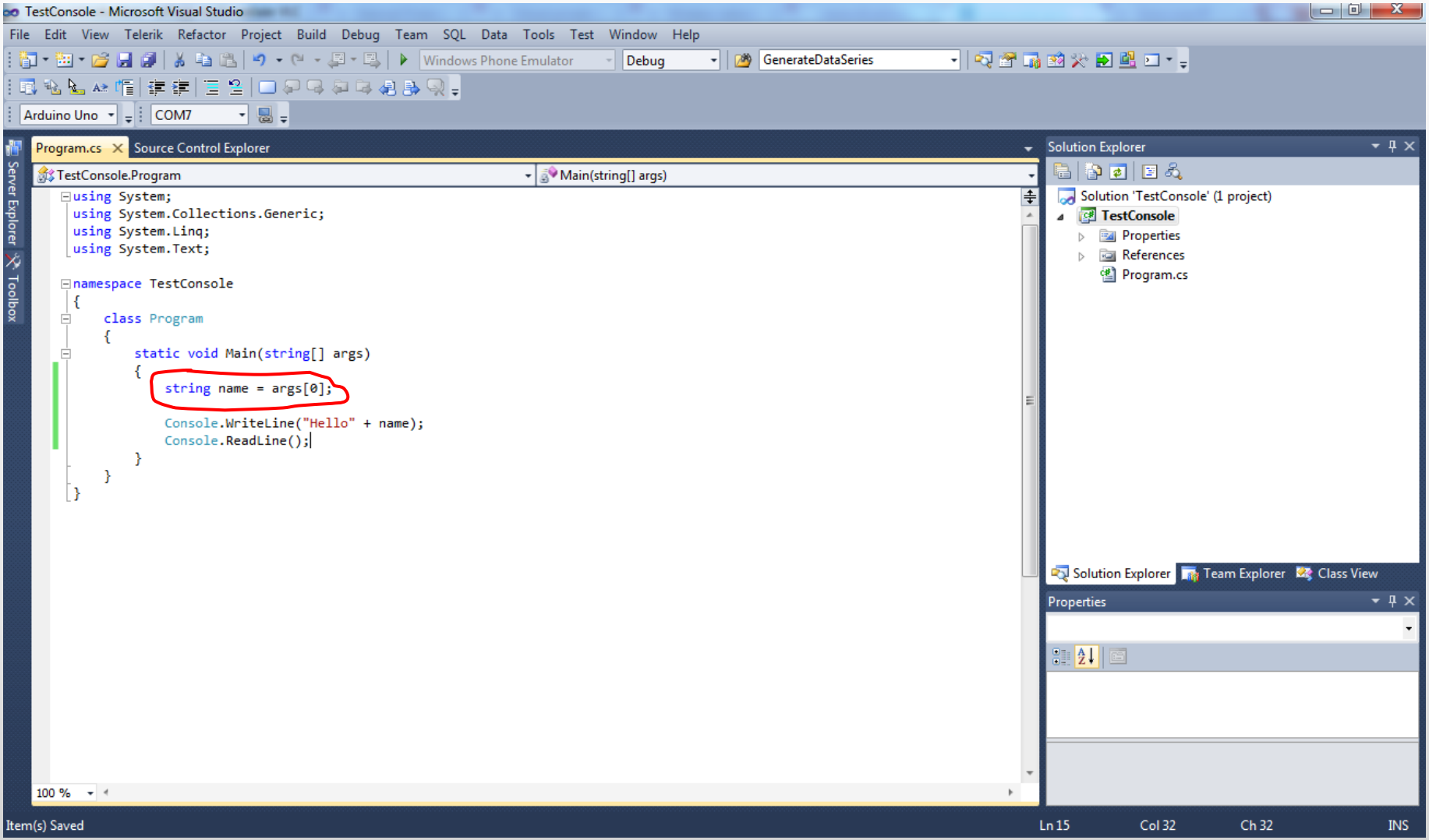
C# has a *unified type system* -> all types ultimately share a common base type. This means that all types, whether they represent business objects or are primitive types such as numbers, share the same basic set of functionality.
For example, any type can be converted to a string by calling its **ToString()** method.

- One way to define objec is to write class
- Many types are built into .NET Framework
- You can define your own custom type

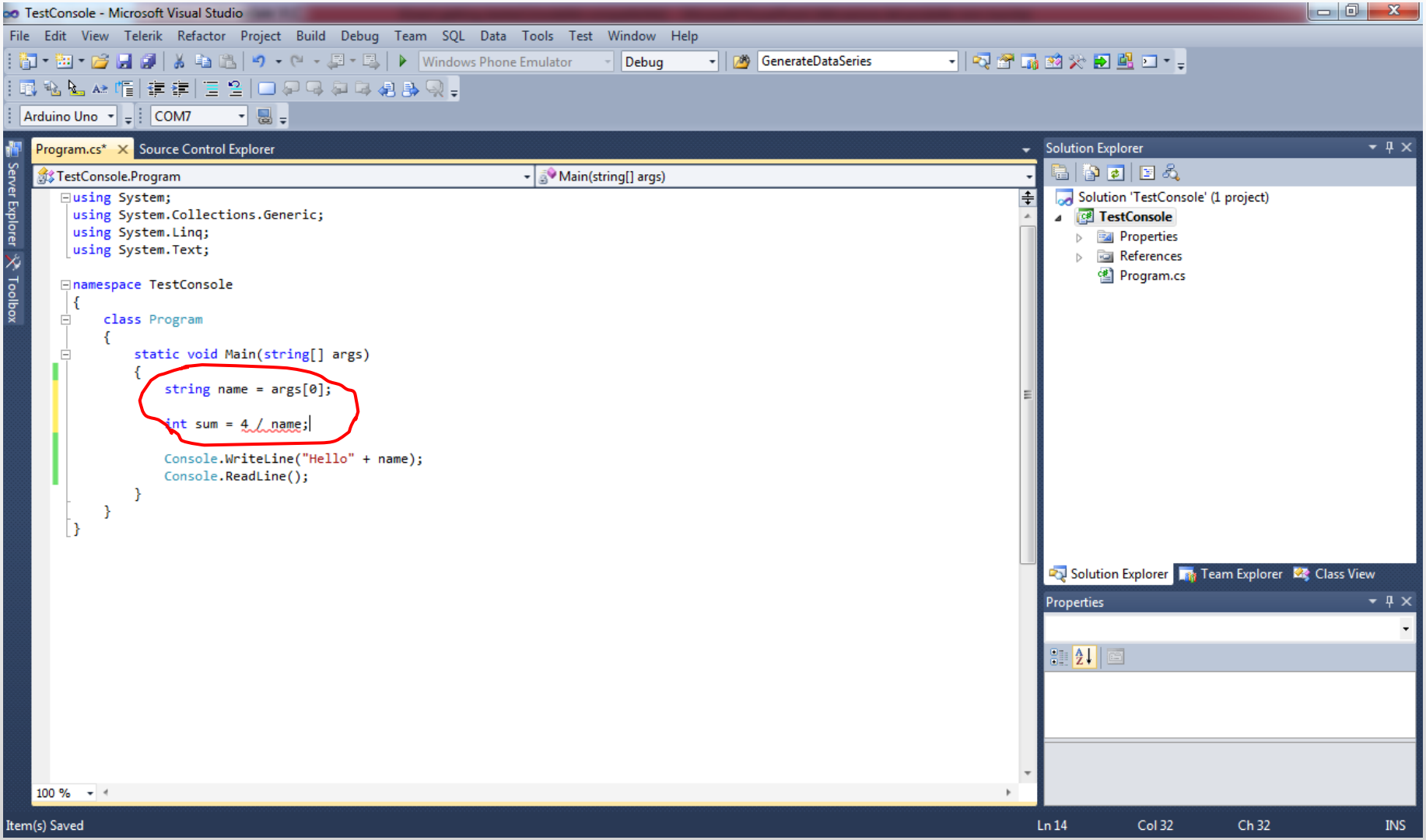
Code you want to execute must live inside a type

- Placing the code inside a method

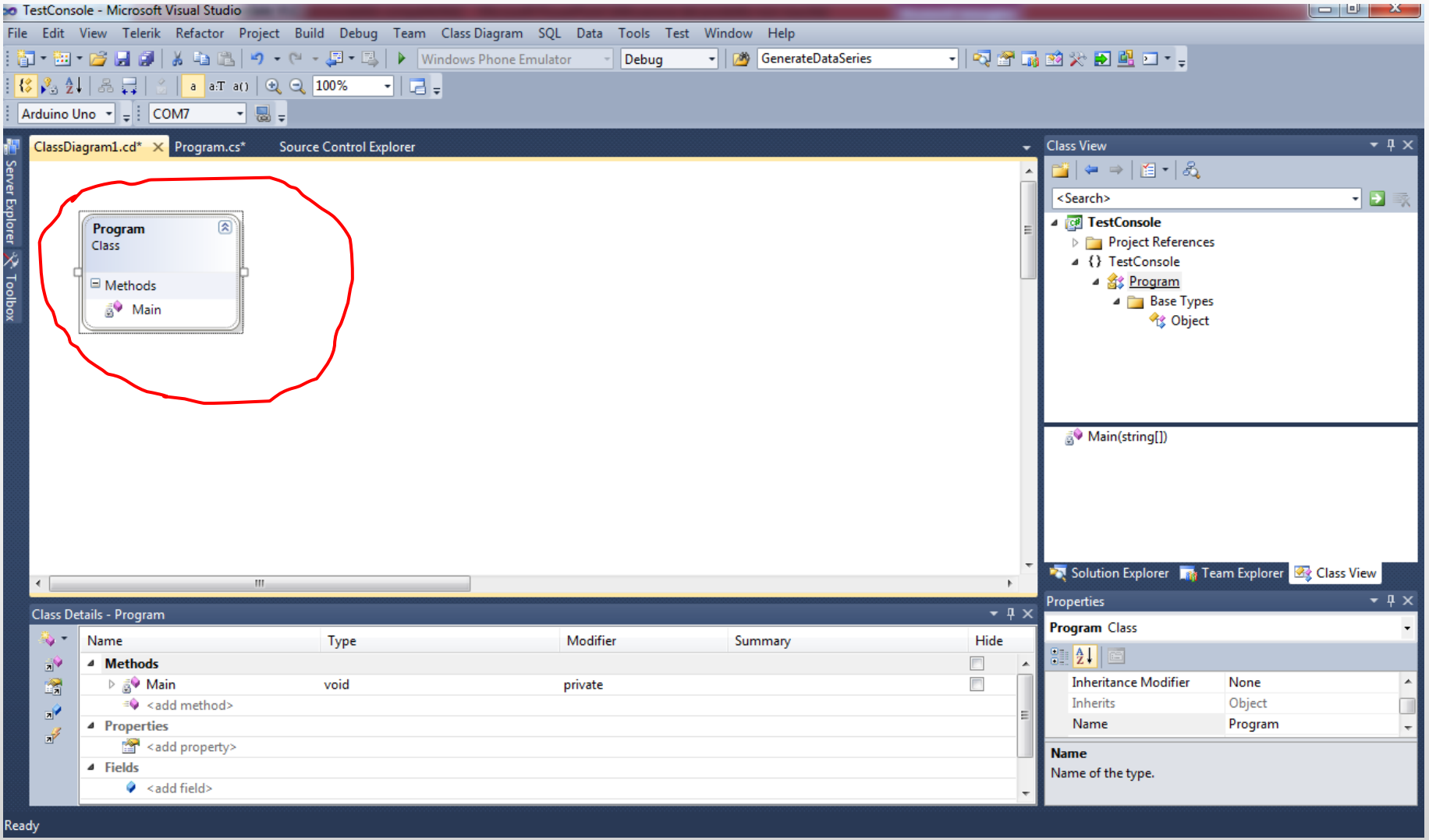
.NET Framework, C# and a little bit of WPF



.NET Framework, C# and a little bit of WPF



.NET Framework, C# and a little bit of WPF



.NET Framework, C# and a little bit of WPF

Primitive Types

Name	Description
Int32 (or int)	32 bit integer
Int64 (or long)	64 bit integer
Boolean (or bool)	true or false
Float (or float)	Single precision floating point
Double (or double)	Double precision floating point
Decimal (or decimal)	Fixed precision (financial)
DateTime	An instant in time (to 100 ns)
String (or string)	Text (as Unicode characters)

Lowest level building blocks of programming

Namespaces

Namespaces organize types

- Avoid type name collision
- Can define namespace in one or more places

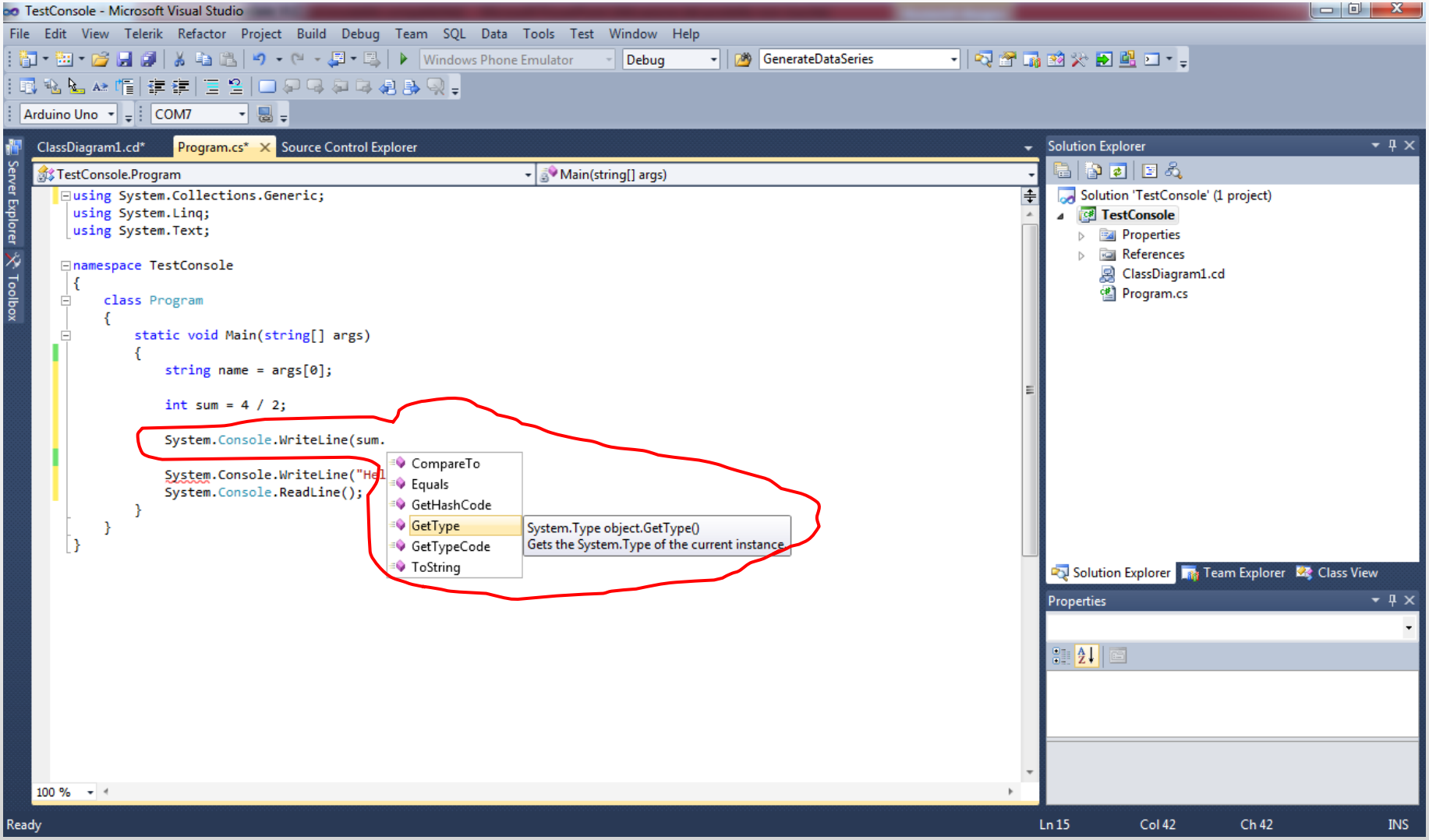
Fully qualified type names

- Include the assembly name
- Include the namespace
- Include the type name

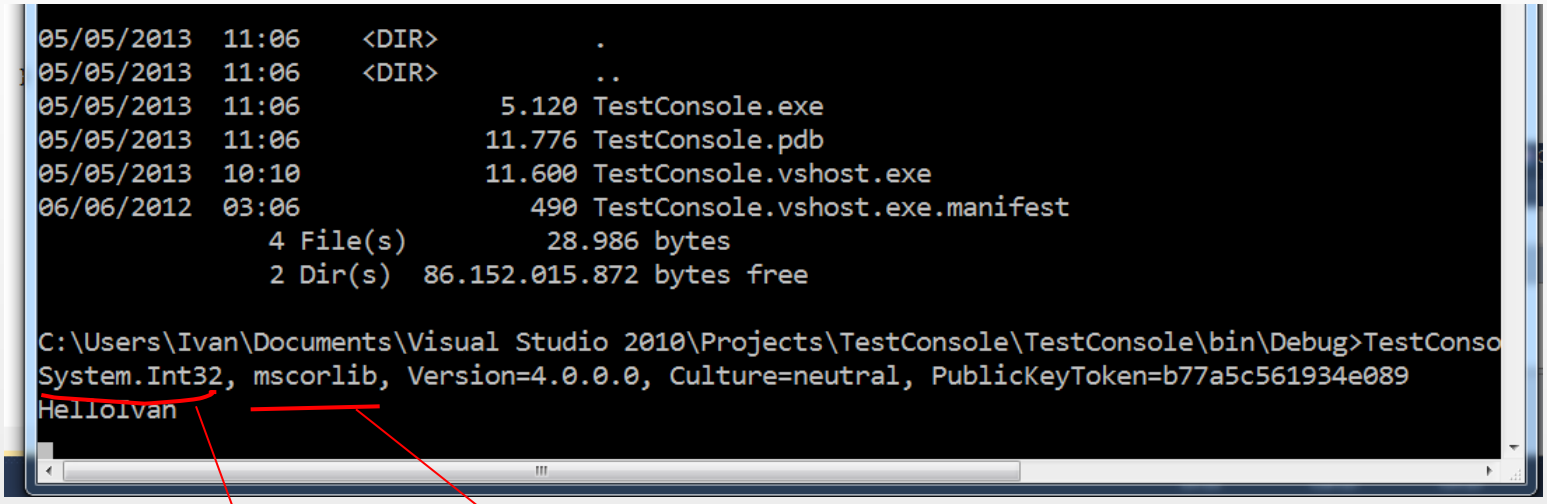
Using directive

- Include the assembly name
- Include the namespace
- Include the type name

.NET Framework, C# and a little bit of WPF



.NET Framework, C# and a little bit of WPF



Name of the assembly containing the namespace and the type

The namespace and the type

VARIABLES

Variables hold a value

- Variables always have type
- Should assign a value before you use a variable
- C# will be sure types are compatible during assignment

```
class Program
{
    static void Main(string[] args)
    {
        string name = args[0];

        int sum = name;

        System.Con                                    QualifiedName);

        System.Con
        System.Con
    }
```

(local variable) string name

Error:
Cannot implicitly convert type 'string' to 'int'

OPERATORS

Specify an operation to perform on one or more variables

- Mathematical operators (+ , - , * , /)
- Relational operators (< , > , <= , >=)
- Equality operators (== , =!)
- Conditional operators (&& , ||)
- Assignment operators (+= , -= , = , *=)

```
int d = 10;
int f = 20;
if (d != f)
{
    d = f;
}
else
{
    d++;
}
```

STATEMENTS AND EXPRESSIONS

A statement is an instruction

- A method is a series of statements
- Statements end with semicolons;
- Statements are executed in the order they appear

```
InitializeKinect();
StartEngine();
StopEngine();
```

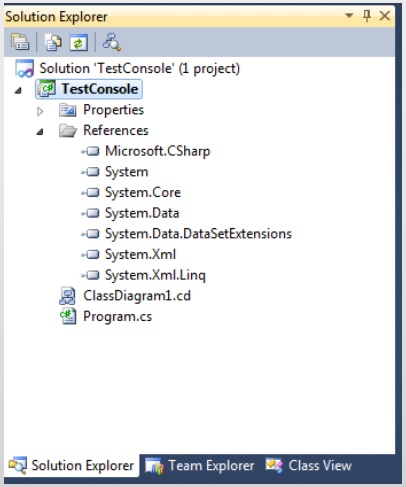
Expressions are statements that produce values

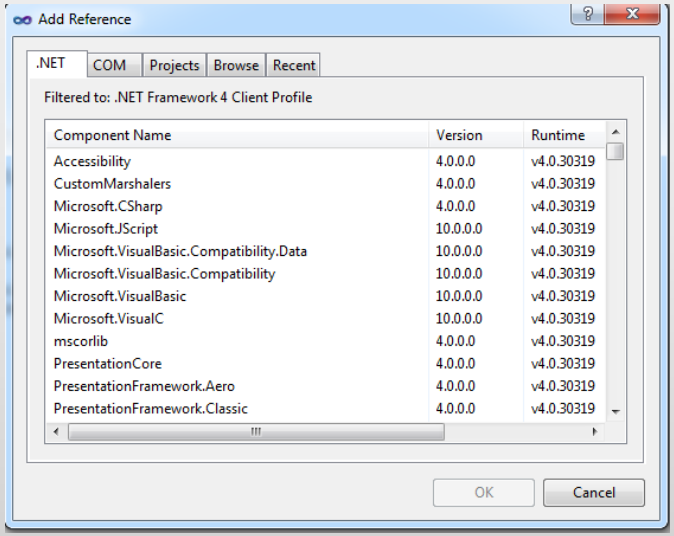
```
int d = 10;
int f = 20;
int result = d + f;
```

REFERENCES

Allow to use types in other assemblies

- Object browser is one way to examine assemblies we want to use
- Reference other assemblies in the FLC
- Reference 3° party assemblies
- Reference other assemblies in the solution





CLASSES AND OBJECTS

Classes define types

- State
- Behavior
- Access

Object are instances of a type

- You can create multiple instances
- Each instance holds a different state
- Each instance has some behavior

CONSTRUCTOR

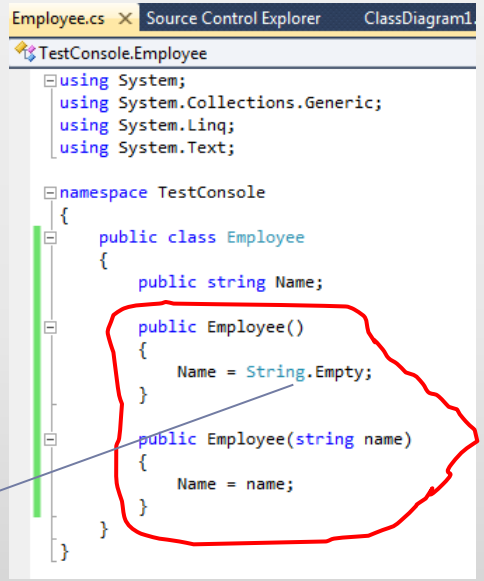
When you write '*new*' you call special methods to create objects

- Set default values

Multiple constructors allowed

- Overloaded methods must take different argument
- Factory Method

- Never returns a type
- Matches the name of the class

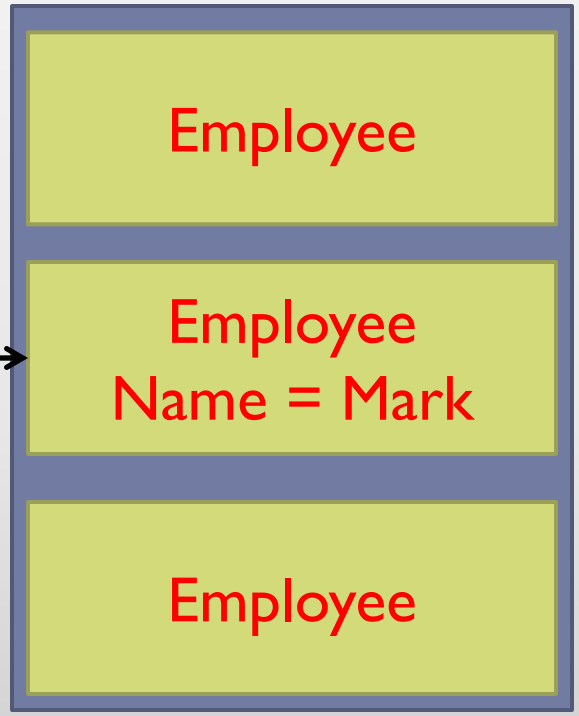


REFERENCE TYPES

Classes definition creates reference types

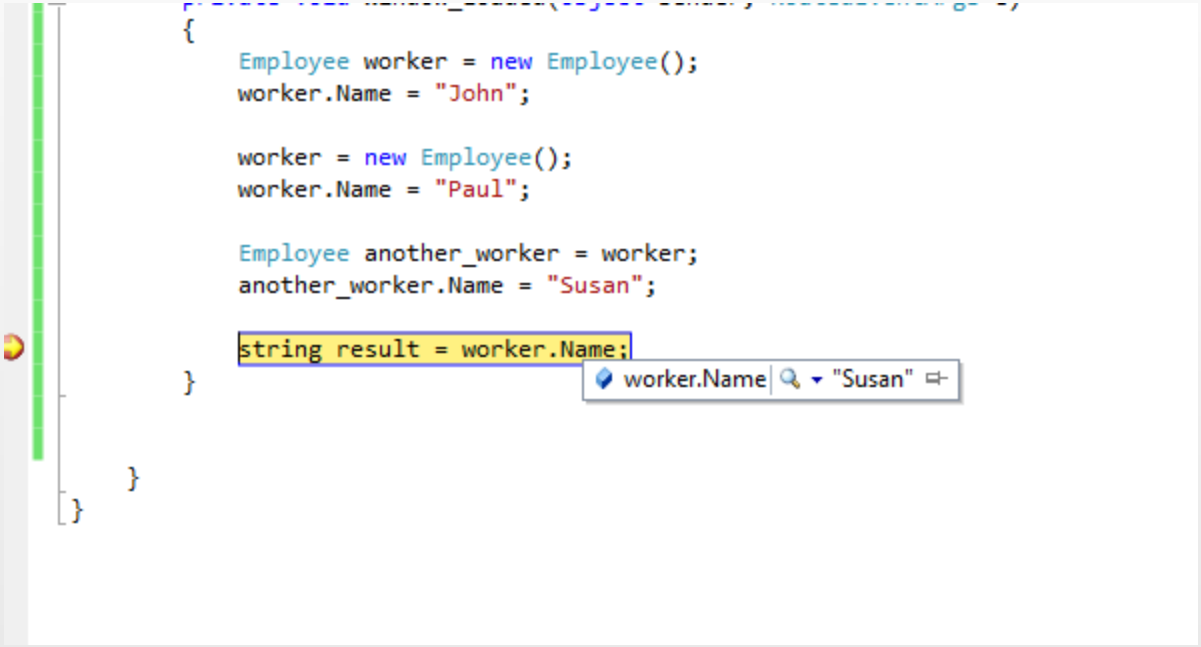
- Objects is stored in the «heap»
- Variables reference the object instance

```
Employee worker = new Employee();
worker.Name = "Mark";
```



- Multiple variables can point to the same object
- Single variable can point to multiple objects during it's lifetime

.NET Framework, C# and a little bit of WPF



VALUE TYPES

Variables hold values

- No pointers or references
- No object allocated on the heap : lightweight

Many built-in primitives are value types

- Int32, DateTime, Double

METHODS

Define behavior

Every method has a return type

Every method has zero or more parameters

Every method has a signature

```
public void WriteAsBytes(int value)
{
    byte[] bytes = BitConverter.GetBytes(value);

    foreach (byte b in bytes)
    {
        string tempValue = String.Format("0x{0:X2}", b);
        Debug.Write(tempValue);
        Debug.Write(" ");
    }


}
```

FIELDS

Fields are variables of a class

- Static fields
- Instance fields

Readonly fields

- Can only assign values in the declaration or in the constructor

```
public class Animal
{
    private readonly string _name;

    public Animal(string name)
    {
        _name = name;
    }
}
```

PROPERTIES

Like fields but they don't denote a storage location

- Every property defines a get and/or a set accessor
- Access level for set and get are indipendent
- With «prop» snippet a field is automatically created

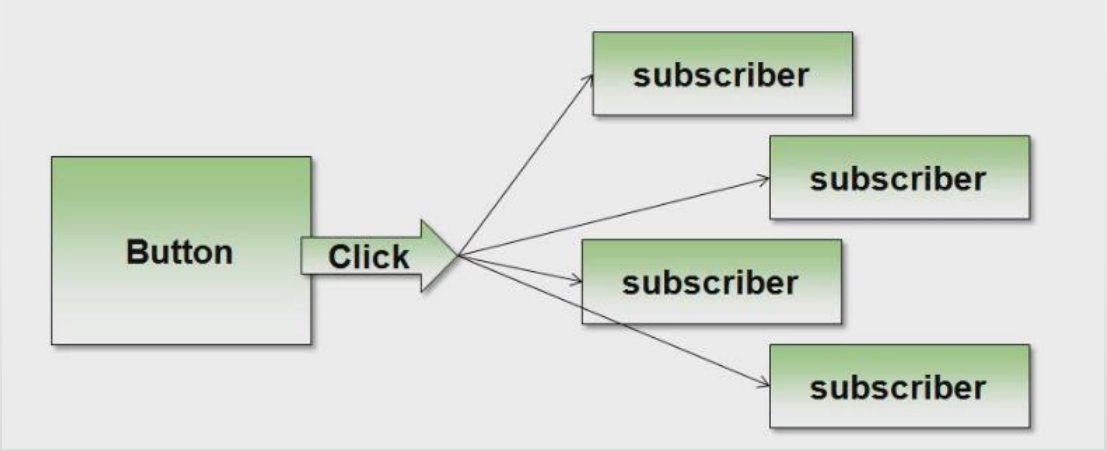
It gives more control on the check of the internal fields

```
private int number;
public int Number
{
    get
    {
        return number;
    }
    set
    {
        if (value > 0)
            number = value;
    }
}
```

EVENTS

Allow a class to send notification to other classes or objects

- Publisher raises the event
- One or more subscribers process the event



.NET Framework, C# and a little bit of WPF

Implementation of the object-oriented paradigm

Inheritance	The ability to define a class that has the same (inherits) behavior and state of another class. For reuse code
Encapsulation	The ability to hide inner details of inner working code in the class. Reduces the complexity and secures some special states.
Polymorphism	Works together with inheritance. The ability to extends the capabilities of a class with the implementation of behaviors and state COMMON TO other classes

.NET Framework, C# and a little bit of WPF

INHERITANCE

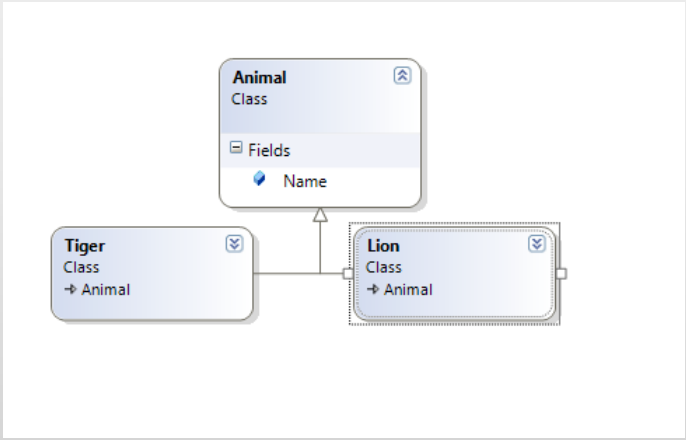
Create classes to extends other classes

- Classes inherits from System.Object by default
- Gain all the state and behavior of the base class

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace TestWPF_Application
{
    public class Animal
    {
        public string Name;
    }

    public class Lion : Animal
    {
    }

    public class Tiger : Animal
    {
    }
}
```

100 %

When not specified derives form System.Object



C#, WPF and the .NET Framework

C# Class:

Formally a class is composed by:

- Field data (the member variables)
- Members that operate on these data (constructor, properties, methods, events)

```
class ECG
    {
        // Thes state of the object
        private string patientName;
        private int samplingFrequency;
        private List<double> dataSamples;

        public int MeanValue()
        {
            return (int)(dataSamples.Sum() / dataSamples.Count);
        }
    }
```

```
class Program
    {
        static void Main(string[] args)
        {
            ECG myECG = new ECG();

            myECG.patientName = "Mario Rossi";
            myECG.samplingFrequency = 1000;
        }
    }
class ECG
    {
        // The state of the object
        public string patientName;
        public int samplingFrequency;
        public List<double> dataSamples;
         // Methos of the object
        public int MeanValue()
        {
            return (int)(dataSamples.Sum() / dataSamples.Count);
        }
    }
```

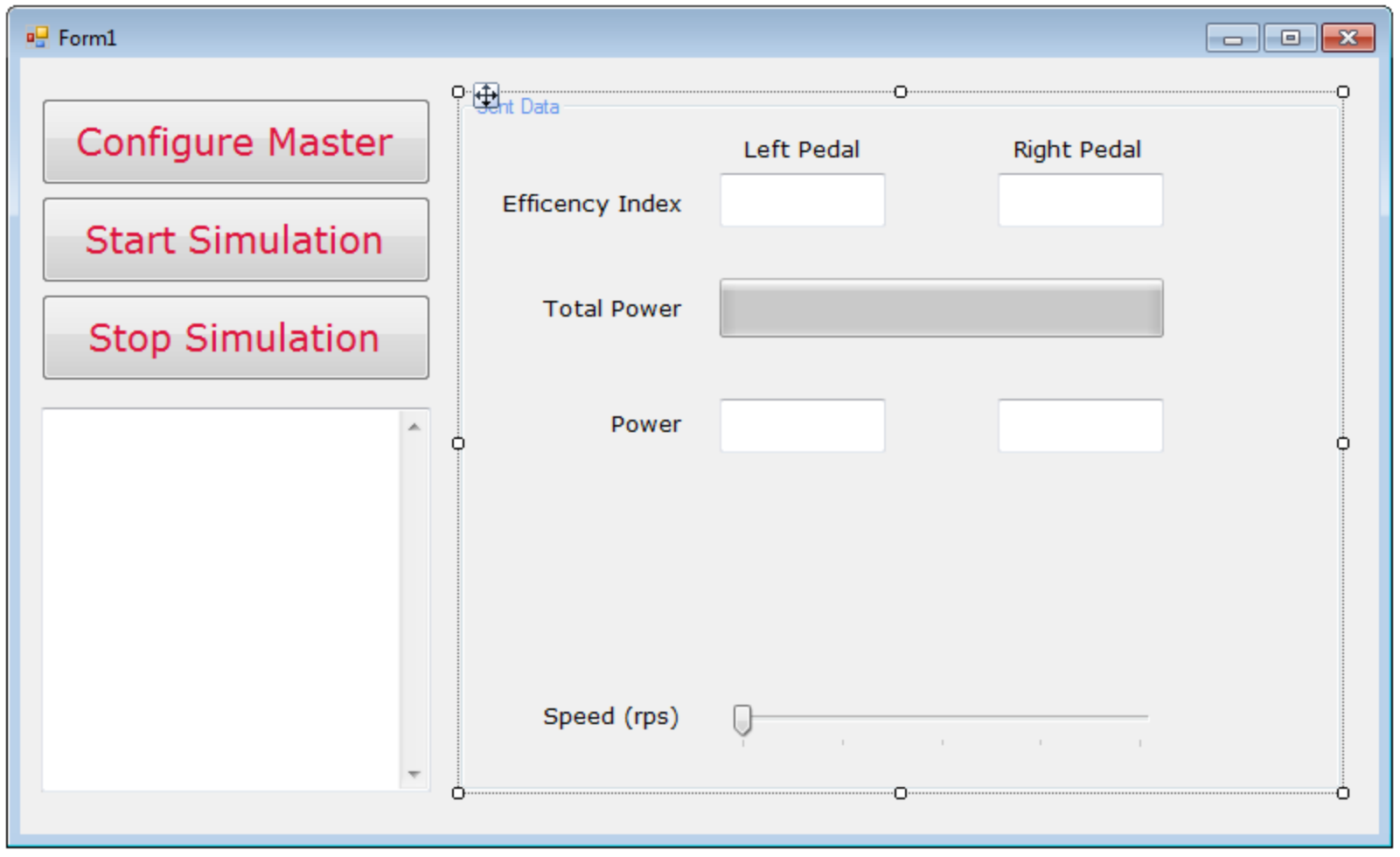
C#, WPF and the .NET Framework

The Windows Presentation Foundation is a graphical display system for Windows.

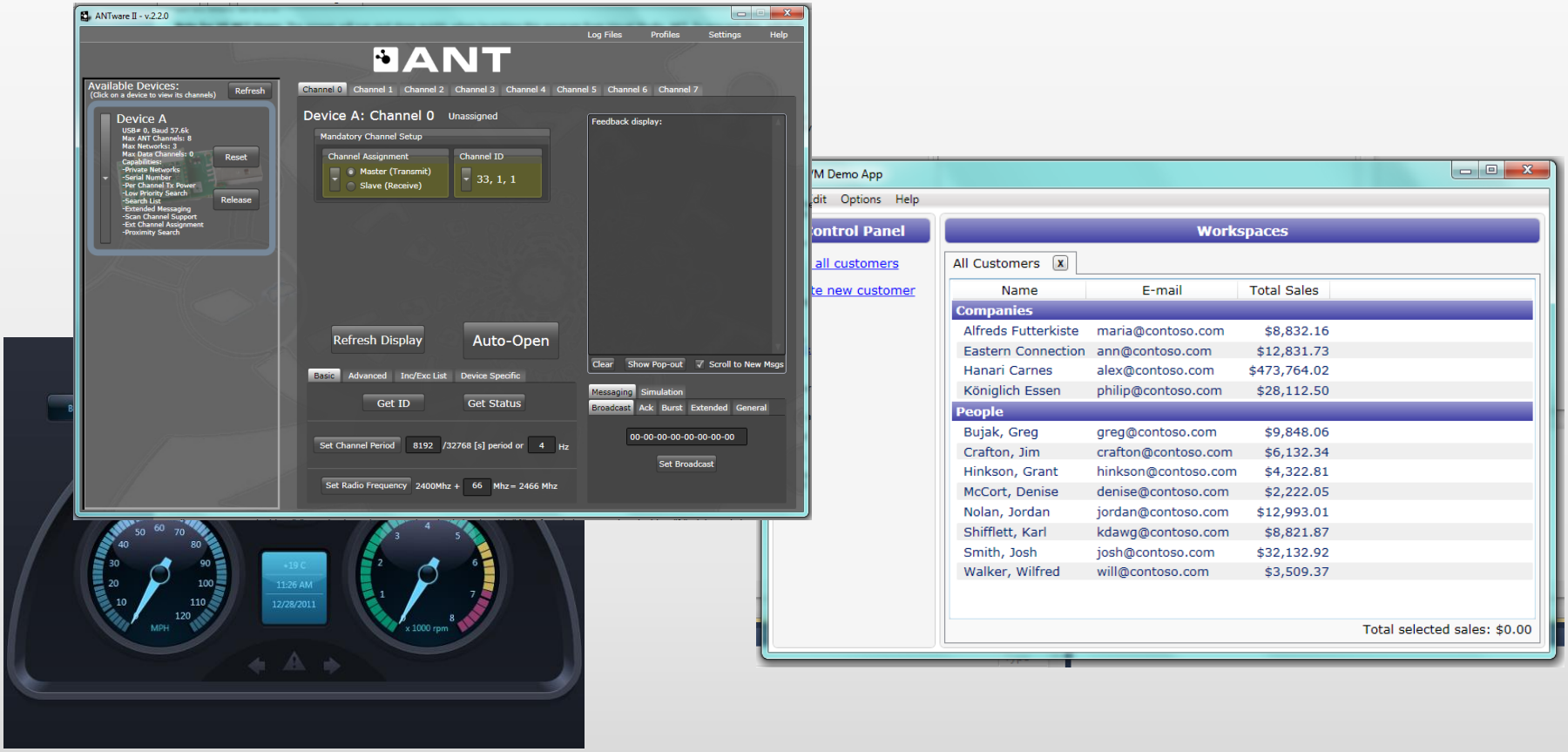
Windows left the GDI/GDI+ (used for more than 10 years) system to embrace the DirectX libraries (best performance)

- WPF enables automatically video card optimization
- and when the video card is too old,..
- ..it automatically optimizes the software (DirectX functions)

C#, WPF and the .NET Framework



C#, WPF and the .NET Framework



C#, WPF and the .NET Framework

http://archive.msdn.microsoft.com/wpfsamples

WPF allows the design of stylish and high-performant application (the programmer should work with a real designer!!):

- Web Layout Model (flexibility)
- Rich Drawing Model (transparent, shapes, graphical layers)
- Animation and timeline
- Support for Audio and Video (Windows Media Player)
- Styles and Template

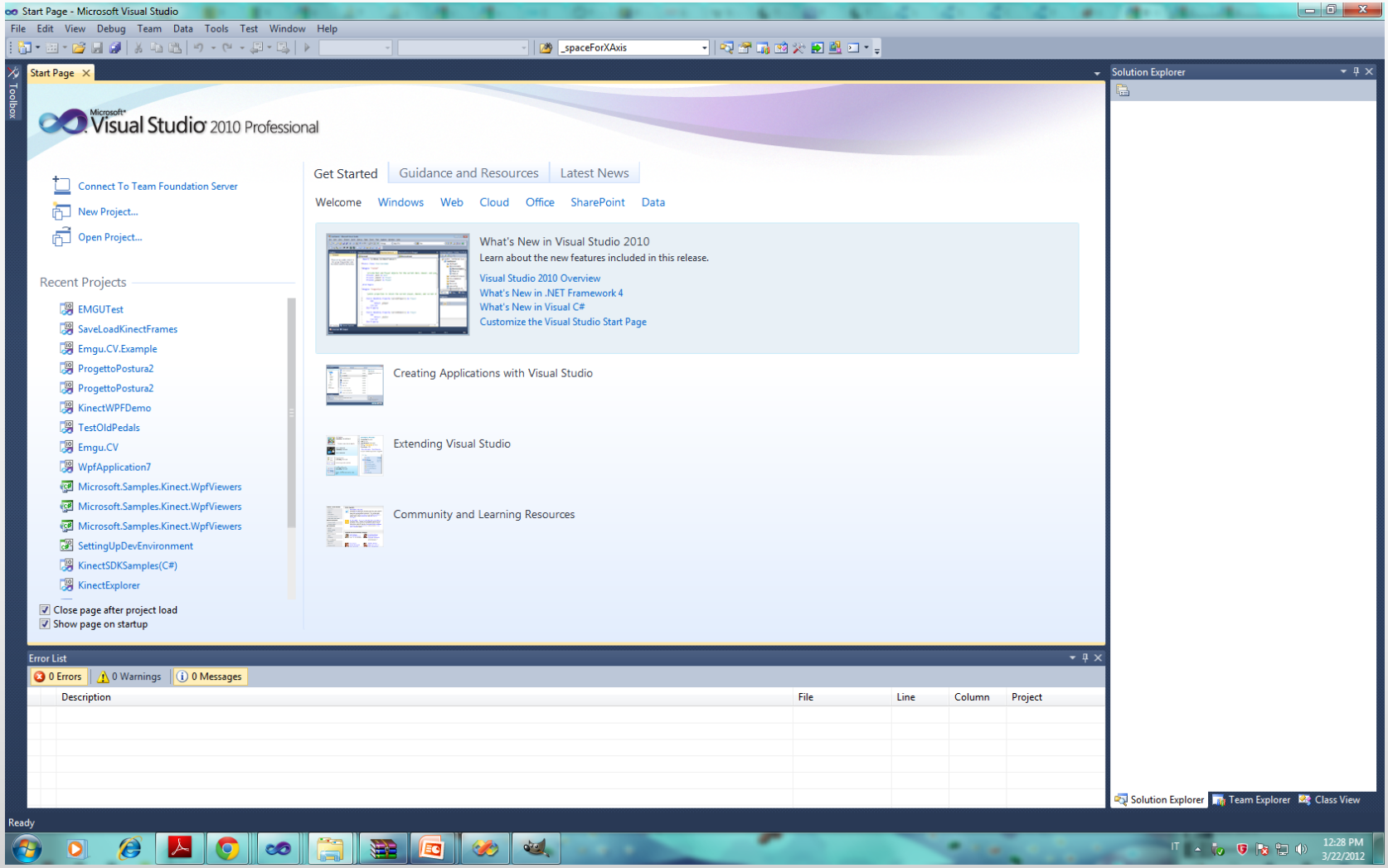
C#, WPF and the .NET Framework

WPF is based on XAML (Extensible Application Markup Language - 2009)

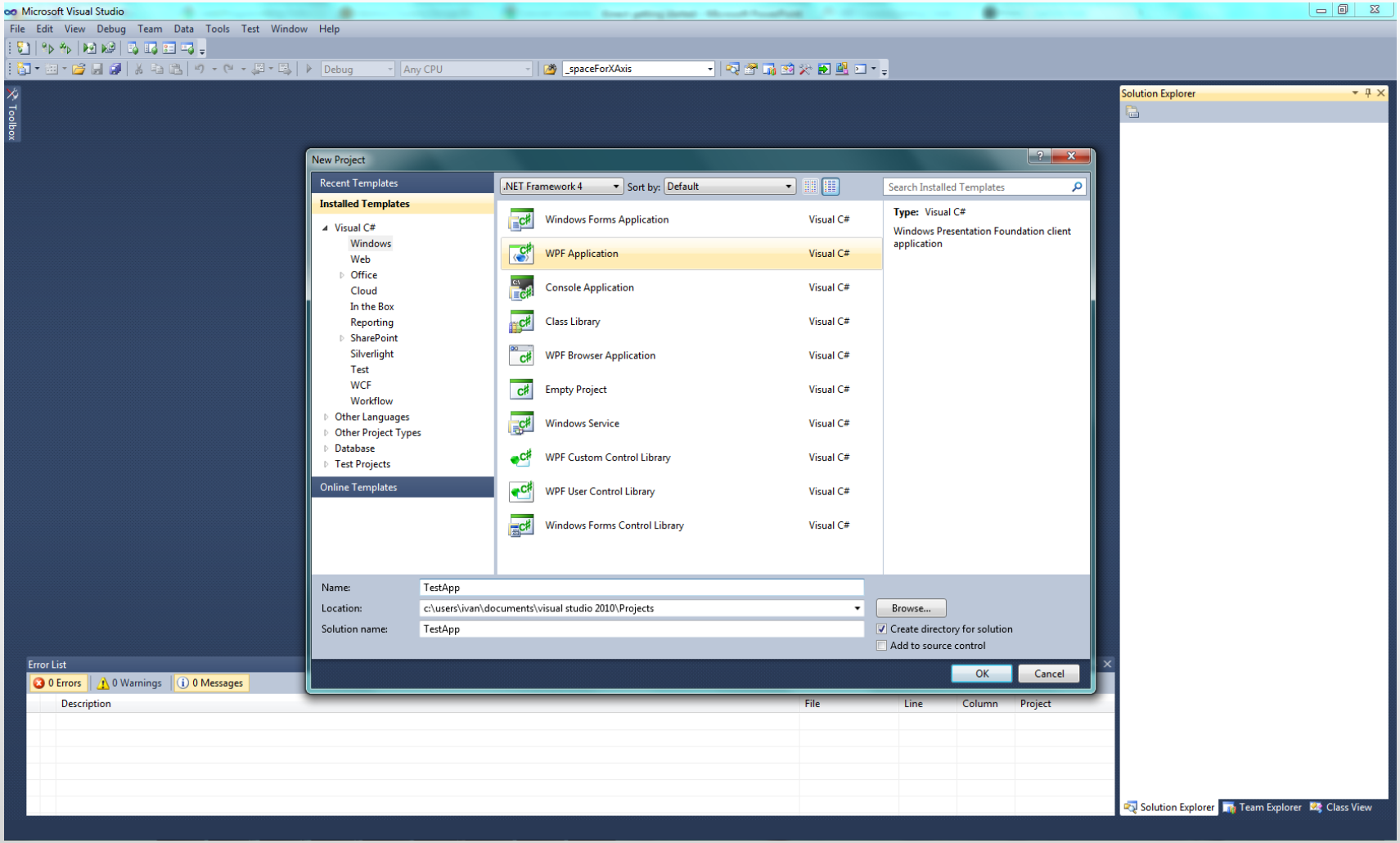
Usually XAML is not written by hand but graphically design by means of special tools (like Expression Blend or Visual Studio design section)

The idea under the XAML is to separate completely the graphic part from the coding part

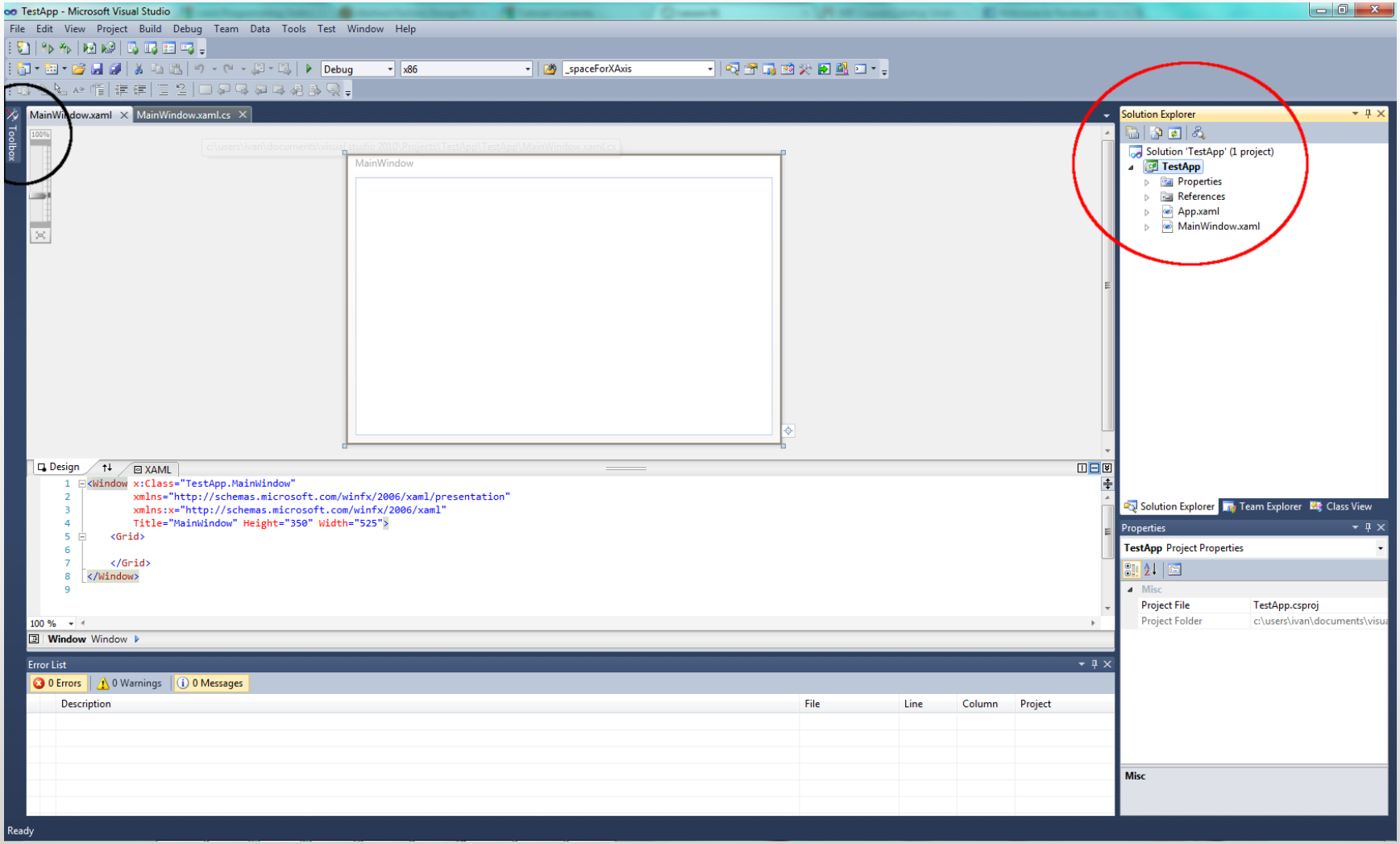
C#, WPF and the .NET Framework



C#, WPF and the .NET Framework



C#, WPF and the .NET Framework



C#, WPF and the .NET Framework

The XAML code behind the default form:

```
<Window x:Class="TestApp.MainWindow“
        xmlns=http://schemas.microsoft.com/winfx/2006/xaml/presentation
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
         Title="MainWindow" Height="350" Width="525">
    <Grid>

    </Grid>
</Window>
```

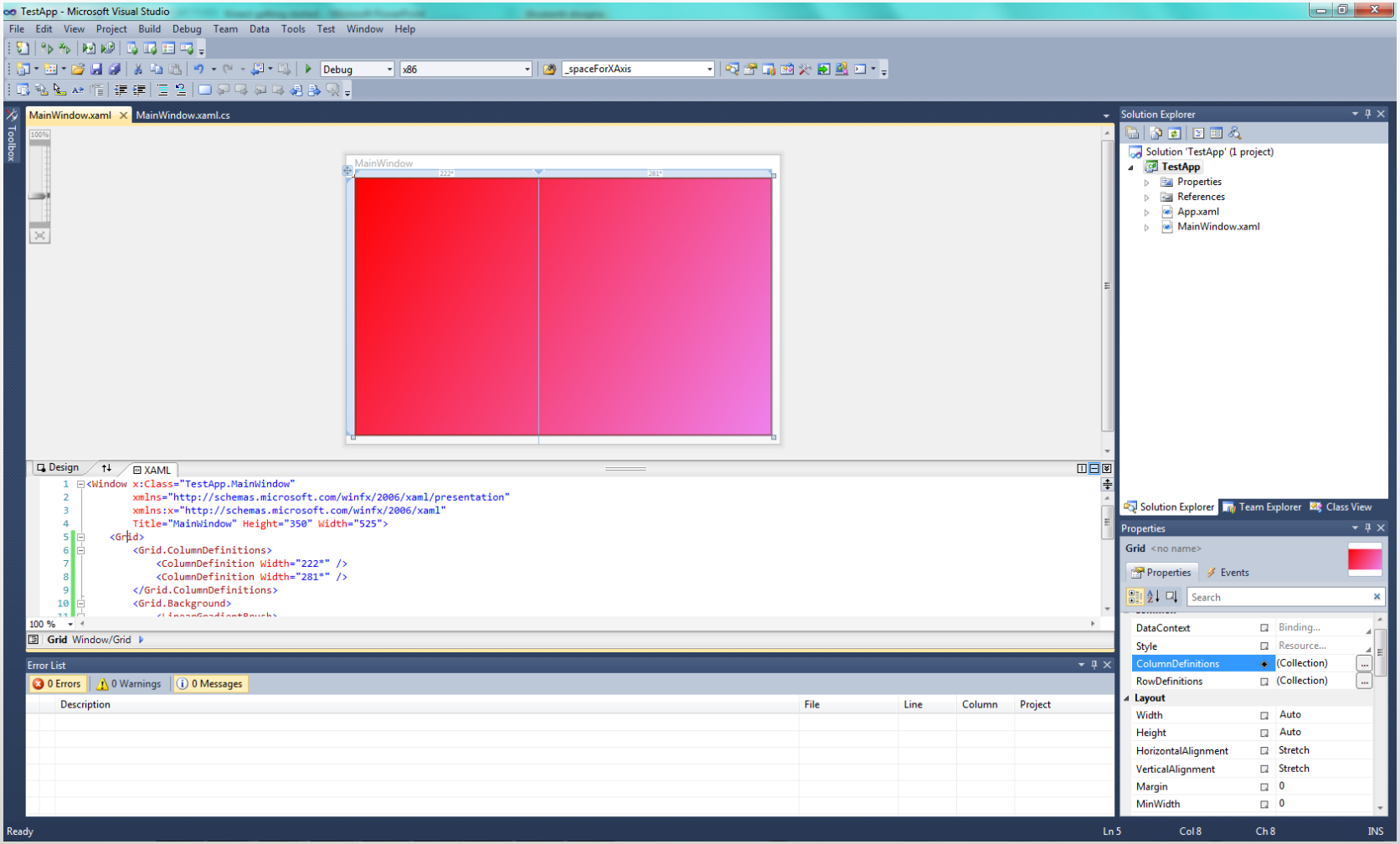
- The element in a XAML maps to instance of .NET classes. The name of the element matches the name of the class (<Grid> is a Grid Object)
- You can nest elements inside elements (same way an HTML page is structured)
- Properties are set through attributes

C#, WPF and the .NET Framework

Let's modify:

```
<Window x:Class="TestApp.MainWindow“
         xmlns=http://schemas.microsoft.com/winfx/2006/xaml/presentation
         xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="MainWindow" Height="350" Width="525">
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="200"></ColumnDefinition>
            <ColumnDefinition Width="*"></ColumnDefinition>
        </Grid.ColumnDefinitions>
         <Grid.Background>
            <LinearGradientBrush>
                <LinearGradientBrush.GradientStops>
                    <GradientStop Offset="0.00" Color="Red" />
                    <GradientStop Offset="1.00" Color="Violet" />
                </LinearGradientBrush.GradientStops>
            </LinearGradientBrush>
        </Grid.Background>

    </Grid>
</Window>
```

C#, WPF and the .NET Framework

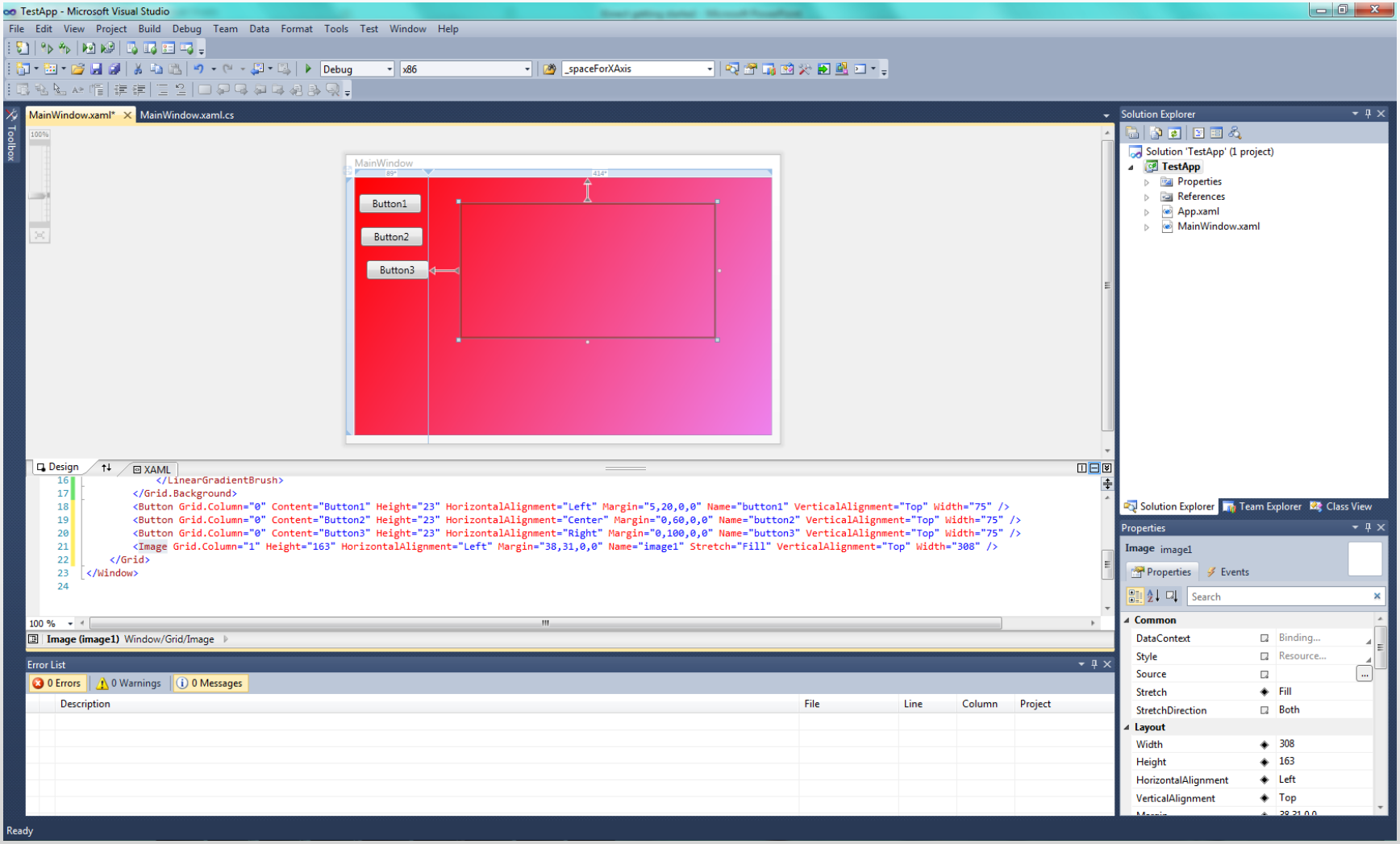


C#, WPF and the .NET Framework

Let's modify:

```
<Window x:Class="TestApp.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="MainWindow" Height="350" Width="525">
    ….[]..
        </Grid.Background>
        <Button Grid.Column="0" Content="Button1" Height="23"
HorizontalAlignment="Left" Margin="5,20,0,0" Name="button1" VerticalAlignment="Top"
Width="75" />
        <Button Grid.Column="0" Content="Button2" Height="23"
HorizontalAlignment="Center" Margin="0,60,0,0" Name="button2" VerticalAlignment="Top"
Width="75" />
        <Button Grid.Column="0" Content="Button3" Height="23"
HorizontalAlignment="Right" Margin="0,100,0,0" Name="button3" VerticalAlignment="Top"
Width="75" />
        <Image Grid.Column="1" Height="163" HorizontalAlignment="Left"
Margin="38,31,0,0" Name="image1" Stretch="Fill" VerticalAlignment="Top" Width="308"
/>
    </Grid>
</Window>
```

C#, WPF and the .NET Framework



C#, WPF and the .NET Framework

Data binding is a relationship that tells WPF to extract some information from a source object and use it to set a property in a target object.

It's perfect for design decoupled systems. The View and the Logic.

EMGU

First we talk about OpenCV!!

What is **OpenCV**?

OpenCV is an open source computer vision library (*http://SourceForge.net/projects/opencvlibrary*). The library is written in C and C++ and runs under Linux, Windows and Mac OS X. There is active development on interfaces for Python, Ruby, Matlab, and other languages.

It is highly-optimized for image processing -> Focus on real time applications

EMGU

Open CV contains over 500 functions that span many areas in vision, including:

- Medical imaging
- Security
- User interface
- Camera calibration
- Stereo vision
- Robotics

A lot of applications have been released:

- Stitching images together in satellite and web maps
- Image scan alignment
- Medical image noise reduction
- Object analysis
- Security and intrusion detection systems
- Military applications

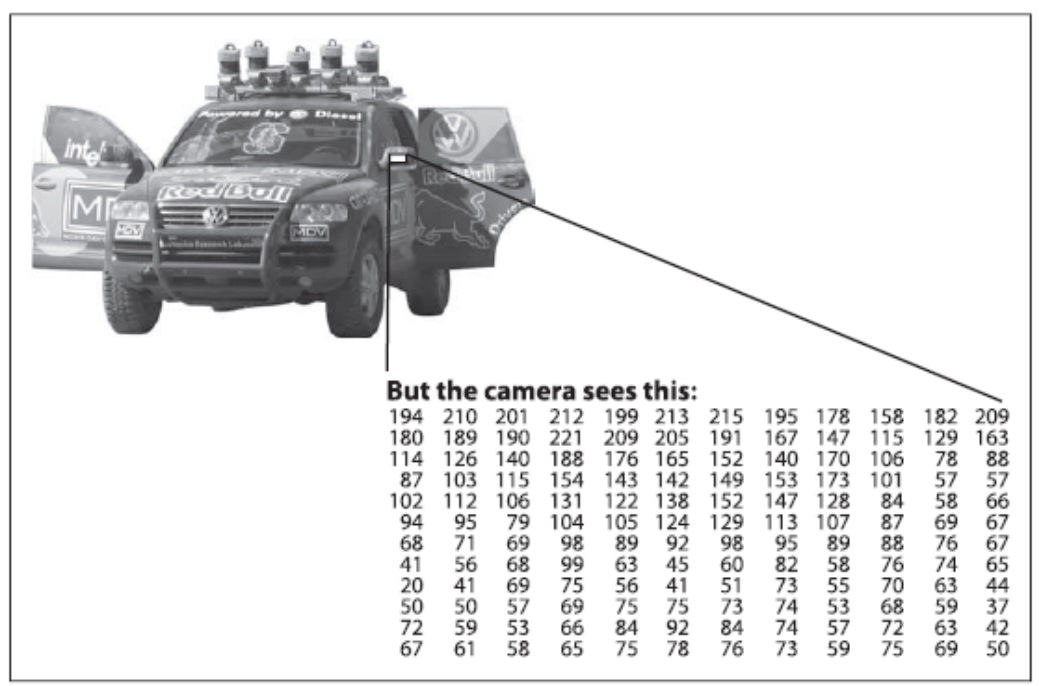
EMGU

OpenCV can be used in commercial product without problem and its community counts more than 20.000 members..!!

Many time in Computer Vision there is the **transformation** of data from a still or video camera into either a **decision** (turning a color image into a grayscale image) or a new **representation** ("there are 5 tumor cells", "the person isn't part of the group")

While the brain has an internal auto-color setting, auto focus setting and pattern recognition system…

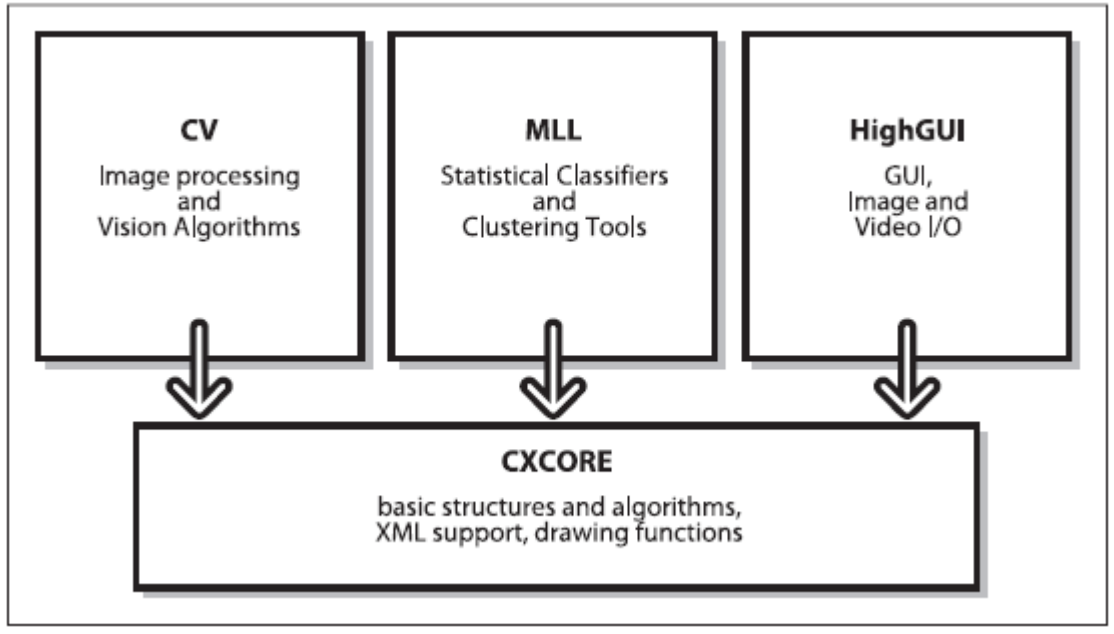
..This is what we get form a camera!!



EMGU

OpenCV is aimed at providing the basic tools needed to solve computer vision problems.

In some cases, high-level functionalities in the library will be sufficient to solve the more complex problems in computer vision. Even when this is not the case, the basic components in the library are complete enough to enable creation of a complete solution of your own to almost any computer vision problem.



EMGU

Basic types of OpenCV (they are all simple structures):

- CvPoint
- CvSize
- CvRect

The most important class in openCV is the **IplImage**!!

It derives from the class CvMatrix (everything in OpenCV is a matrix), and this is the reason why it's possible to operate with special matrix functions and operators directly on these images!!

EMGU

Function	Description
cvAbs	Absolute value of all elements in an array
cvAbsDiff	Absolute value of differences between two arrays
cvAbsDiffS	Absolute value of difference between an array and a scalar
cvAdd	Elementwise addition of two arrays
cvAddS	Elementwise addition of an array and a scalar
cvAddWeighted	Elementwise weighted addition of two arrays (alpha blending)
cvAvg	Average value of all elements in an array
cvAvgSdv	Absolute value and standard deviation of all elements in an array
cvCalcCovarMatrix	Compute covariance of a set of *n*-dimensional vectors
cvCmp	Apply selected comparison operator to all elements in two arrays
cvCmpS	Apply selected comparison operator to an array relative to a scalar
cvConvertScale	Convert array type with optional rescaling of the value
cvConvertScaleAbs	Convert array type after absolute value with optional rescaling
cvCopy	Copy elements of one array to another
cvCountNonZero	Count nonzero elements in an array
cvCrossProduct	Compute cross product of two three-dimensional vectors
cvCvtColor	Convert channels of an array from one color space to another
cvDet	Compute determinant of a square matrix
cvDiv	Elementwise division of one array by another
cvDotProduct	Compute dot product of two vectors
cvEigenVV	Compute eigenvalues and eigenvectors of a square matrix

EMGU

cvFlip	Flip an array about a selected axis
cvGEMM	Generalized matrix multiplication
cvGetCol	Copy elements from column slice of an array
cvGetCols	Copy elements from multiple adjacent columns of an array
cvGetDiag	Copy elements from an array diagonal
cvGetDims	Return the number of dimensions of an array
cvGetDimSize	Return the sizes of all dimensions of an array
cvGetRow	Copy elements from row slice of an array
cvGetRows	Copy elements from multiple adjacent rows of an array
cvGetSize	Get size of a two-dimensional array and return as CvSize
cvGetSubRect	Copy elements from subregion of an array
cvInRange	Test if elements of an array are within values of two other arrays
cvInRangeS	Test if elements of an array are in range between two scalars
cvInvert	Invert a square matrix

EMGU

cvReduce	Reduce a two-dimensional array to a vector by a given operation
cvRepeat	Tile the contents of one array into another
cvSet	Set all elements of an array to a given value
cvSetZero	Set all elements of an array to 0
cvSetIdentity	Set all elements of an array to 1 for the diagonal and 0 otherwise
cvSolve	Solve a system of linear equations
cvSplit	Split a multichannel array into multiple single-channel arrays
cvSub	Elementwise subtraction of one array from another
cvSubS	Elementwise subtraction of a scalar from an array
cvSubRS	Elementwise subtraction of an array from a scalar
cvSum	Sum all elements of an array
cvSVD	Compute singular value decomposition of a two-dimensional array
cvSVBkSb	Compute singular value back-substitution
cvTrace	Compute the trace of an array
cvTranspose	Transpose all elements of an array across the diagonal
cvXor	Elementwise bit-level XOR between two arrays
cvXorS	Elementwise bit-level XOR between an array and a scalar
cvZero	Set all elements of an array to 0

EMGU

That was only a little part for Matrix operations… !!!

There are also special methids that can be apply directly on an image (Smooth filtering, Canny, Hough transform, etc..)

http://www.seas.upenn.edu/~bensapp/opencvdocs/ref/opencvref_cv.htm

Here comes EMGU…

"**Emgu CV** is a cross platform .Net **wrapper** to the Intel **OpenCV** image processing library and allows OpenCv functions to be called from .NET compatible languages such as C#, VB, IronPython,.."

This means that it's possible to use OpenCV methods and structure in the C# simple style…

Example: the **IplImage** is defined in EMGU as an **Image** and is described (and instantiated since it's a class) by its generic parameters: color and depth

An image with 3 channels BGR each one defined by 1 byte:

```
Image<Bgr, byte> image=new Image<Bgr, byte>(new System.Drawing.Size(640, 480));
```

(The image will be managed by the garbage collector)

The main color types are supported :

Gray
Bgr
Bgra
Hsv (Hue Saturation Value)
Hls (Hue Lightness Saturation)
Lab (CIE L*a*b*)

EMGU

One of the most important method in EMGU is the **CvInvoke,** which allows to call directly the OpenCv functions (some OpenCV functions are wrapped in EMGU methods, but not all of them)…

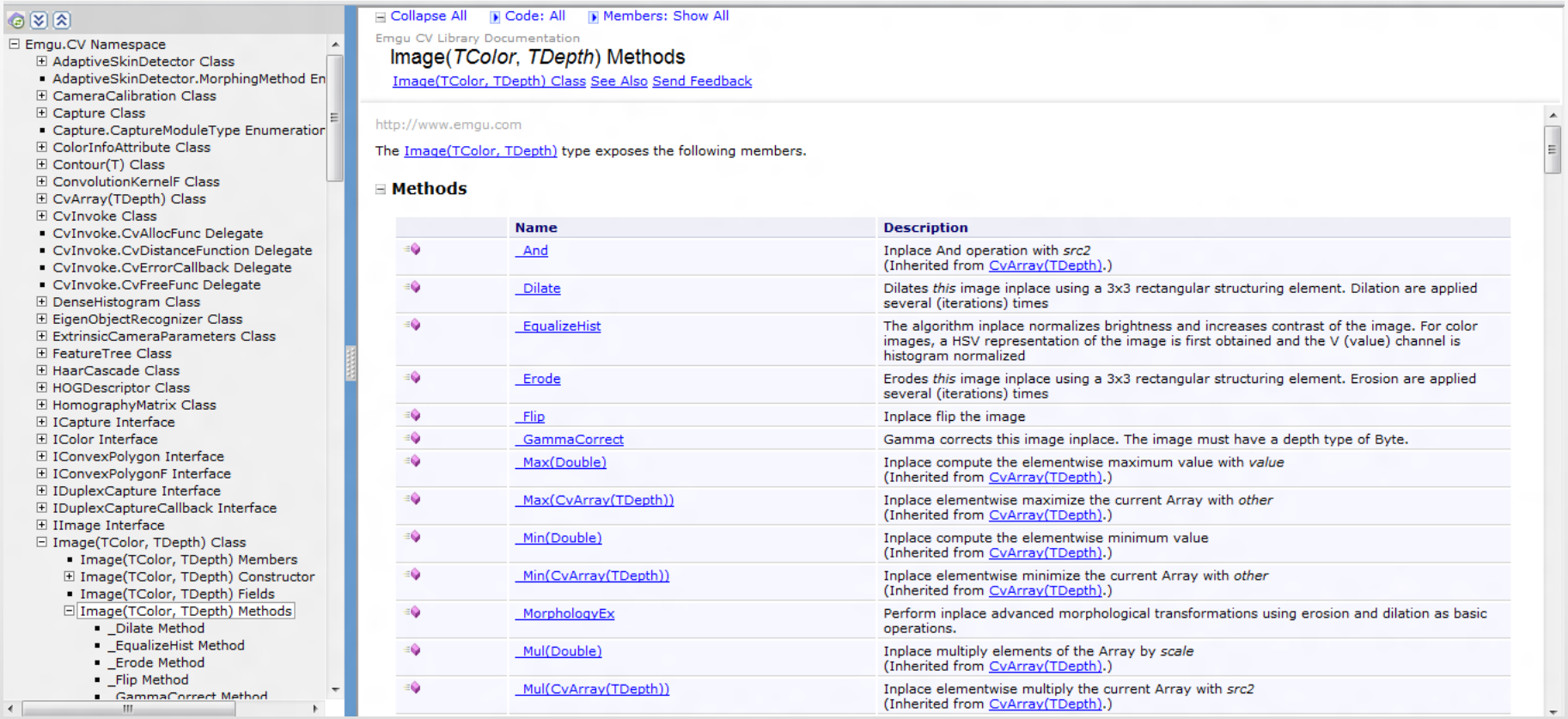
```
IntPtr image = CvInvoke.cvCreateImage(new System.Drawing.Size(200, 200),
Emgu.CV.CvEnum.IPL_DEPTH.IPL_DEPTH_8U, 1);

CvInvoke.cvDilate(ImageIn, ImageOut, myDilateElem, 1);
```

BUT…. For a basic list of methods that you can apply directly on the **Image<ColorType, Depht>** go:

http://www.emgu.com/wiki/files/2.3.0/document/Index.html

EMGU.CV.NameSpace -> Image (TColor, Tdepht) class -> Methods

EMGU


Emgu.CV Namespace
AdaptiveSkinDetector Class
AdaptiveSkinDetector.MorphingMethod En
CameraCalibration Class
Capture Class
Capture.CaptureModuleType Enumeration
ColorInfoAttribute Class
Contour(T) Class
ConvolutionKernelF Class
CvArray(TDepth) Class
CvInvoke Class
CvInvoke.CvAllocFunc Delegate
CvInvoke.CvDistanceFunction Delegate
CvInvoke.CvErrorCallback Delegate
CvInvoke.CvFreeFunc Delegate
DenseHistogram Class
EigenObjectRecognizer Class
ExtrinsicCameraParameters Class
FeatureTree Class
HaarCascade Class
HOGDescriptor Class
HomographyMatrix Class
ICapture Interface
IColor Interface
IConvexPolygon Interface
IConvexPolygonF Interface
IDuplexCapture Interface
IDuplexCaptureCallback Interface
IImage Interface
Image(TColor, TDepth) Class
Image(TColor, TDepth) Members
Image(TColor, TDepth) Constructor
Image(TColor, TDepth) Fields
Image(TColor, TDepth) Methods
_Dilate Method
_EqualizeHist Method
_Erode Method
_Flip Method
Collapse All
Code: All
Members: Show All
Emgu CV Library Documentation
Image(TColor, TDepth) Methods
Image(TColor, TDepth) Class See Also Send Feedback
http://www.emgu.com
The Image(TColor, TDepth) type exposes the following members.
Methods
Name
Description
_And
Inplace And operation with src2
(Inherited from CvArray(TDepth).)
_Dilate
Dilates this image inplace using a 3x3 rectangular structuring element. Dilation are applied several (iterations) times
_EqualizeHist
The algorithm inplace normalizes brightness and increases contrast of the image. For color images, a HSV representation of the image is first obtained and the V (value) channel is histogram normalized
_Erode
Erodes this image inplace using a 3x3 rectangular structuring element. Erosion are applied several (iterations) times
_Flip
Inplace flip the image
_GammaCorrect
Gamma corrects this image inplace. The image must have a depth type of Byte.
_Max(Double)
Inplace compute the elementwise maximum value with value
(Inherited from CvArray(TDepth).)
_Max(CvArray(TDepth))
Inplace elementwise maximize the current Array with other
(Inherited from CvArray(TDepth).)
_Min(Double)
Inplace compute the elementwise minimum value
(Inherited from CvArray(TDepth).)
_Min(CvArray(TDepth))
Inplace elementwise minimize the current Array with other
(Inherited from CvArray(TDepth).)
_MorphologyEx
Perform inplace advanced morphological transformations using erosion and dilation as basic operations.
_Mul(Double)
Inplace multiply elements of the Array by scale
(Inherited from CvArray(TDepth).)
_Mul(CvArray(TDepth))
Inplace elementwise multiply the current Array with src2
(Inherited from CvArray(TDepth).)

EMGU

Let's see an example!!