“Visual Studio .NET enables quick, drag-and-drop construction of form-based applications…”

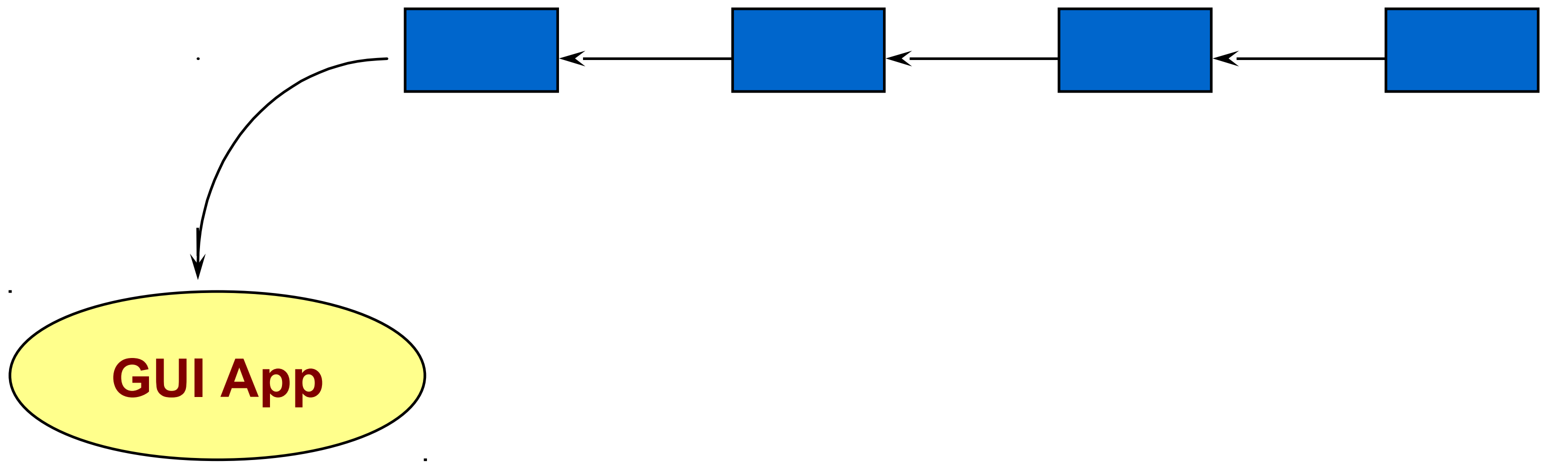
- **Event-driven, code-behind programming**
- **Visual Studio .NET**
- **WinForms**
- **Controls**

Part 1

- Event-driven, code-behind programming…

Event-driven applications

- **Idea is very simple:**
  - individual user actions are translated into “events”
  - events are passed, 1 by 1, to application for processing



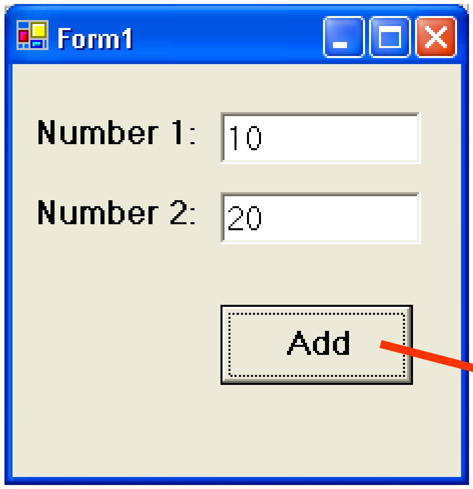
  - this is how most GUIs are programmed…

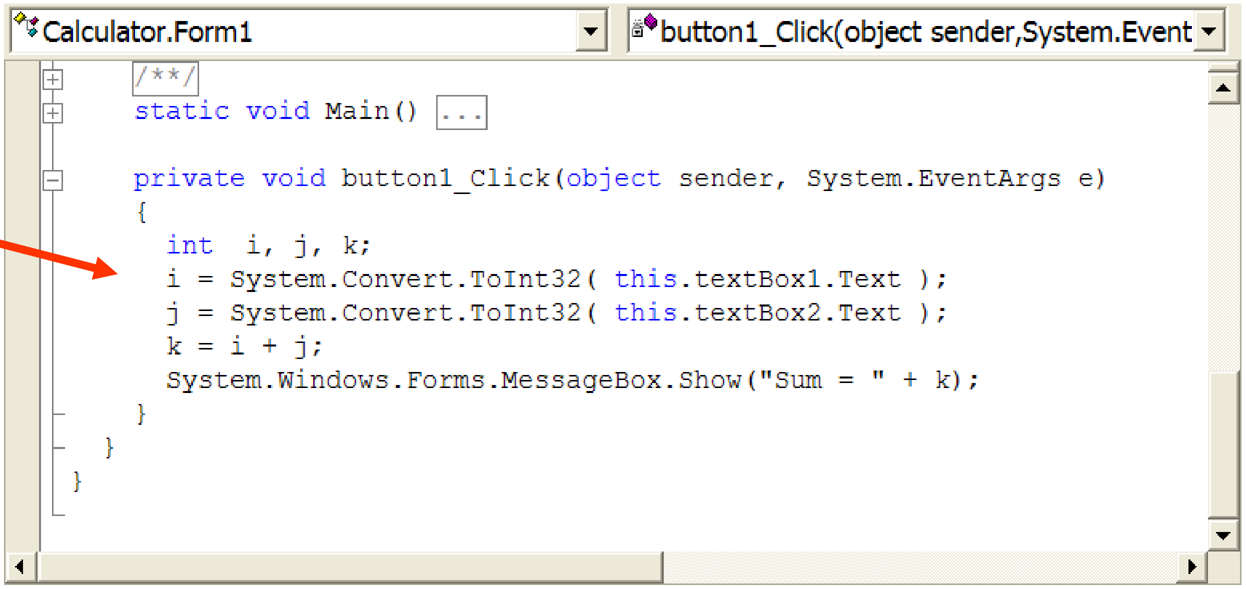
GUI-based events

- Mouse move
- Mouse click
- Mouse double-click
- Key press
- Button click
- Menu selection
- Change in focus
- Window activation
- etc.

Code-behind

- **Events are handled by methods that live behind visual interface**
  - known as "code-behind"
  - our job is to program these methods…

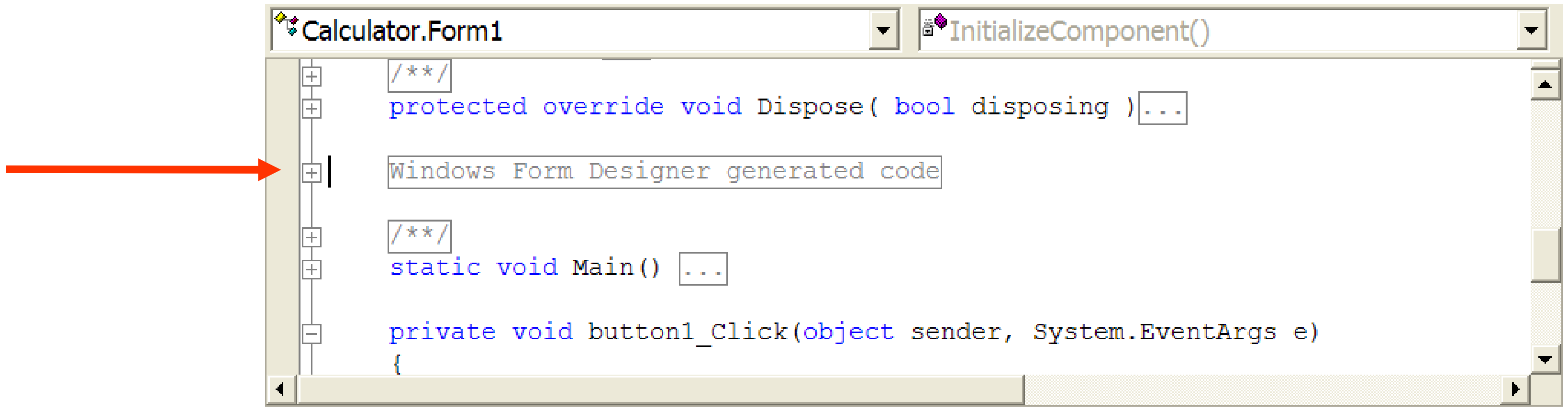




```
    /**/
    static void Main() ...

    private void button1_Click(object sender, System.EventArgs e)
    {
      int  i, j, k;
      i = System.Convert.ToInt32( this.textBox1.Text );
      j = System.Convert.ToInt32( this.textBox2.Text );
      k = i + j;
      System.Windows.Forms.MessageBox.Show("Sum = " + k);
    }
  }
}
```

Call-backs

- **Events are a *call* from object *back* to us…**
- **How is connection made?**
  - setup by code auto-generated by Visual Studio



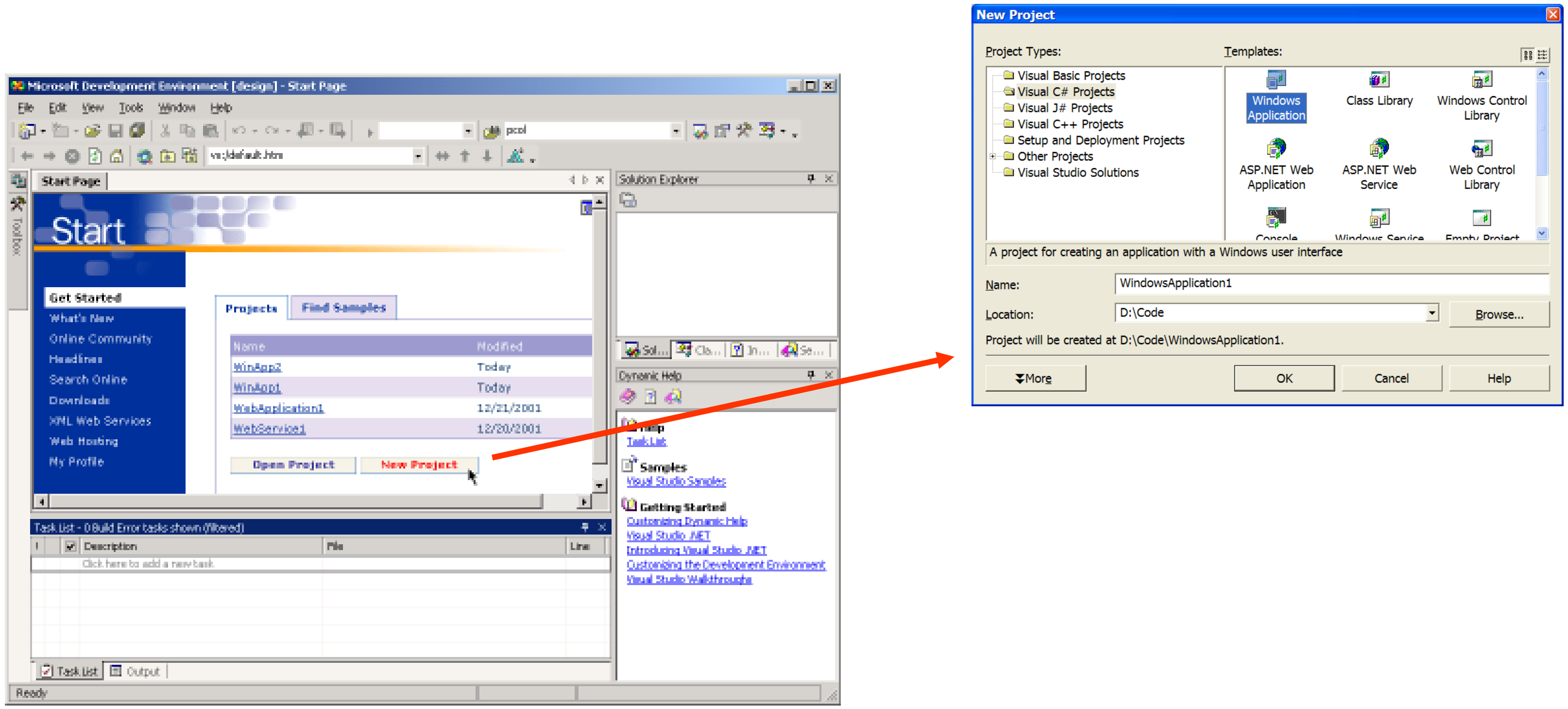
```
/**/
protected override void Dispose( bool disposing )...

Windows Form Designer generated code

/**/
static void Main() ...

private void button1_Click(object sender, System.EventArgs e)
{
```

Part 2

- **Visual Studio .NET…**

Visual Studio .NET (VS.NET)

- **A single IDE for all forms of .NET development**
  - from class libraries to form-based apps to web services
  - and using C#, VB, C++, J#, etc.



Name	Modified
WinApp2	Today
WinApp1	Today
WebApplication1	12/21/2001
WebService1	12/20/2001

Task List - 0 Build Error tasks shown (filtered)

!	✓	Description	File	Line
		Click here to add a new task		

A project for creating an application with a Windows user interface

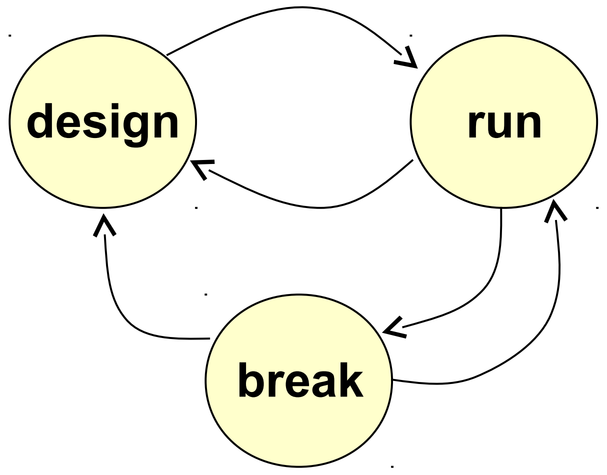
Name: WindowsApplication1

Location: D:\Code

Project will be created at D:\Code\WindowsApplication1.

Basic operation

- **Visual Studio operates in one of 3 modes:**
  1) design
  2) run
  3) break


$$\begin{tikzcd}
\text{design} \arrow[rr, bend left] & & \text{run} \arrow[ll, bend left] \arrow[dl, bend left] \\
& \text{break} \arrow[ul, bend left] \arrow[ur, bend right] &
\end{tikzcd}$$

- **When in doubt, check the title bar of VS…**

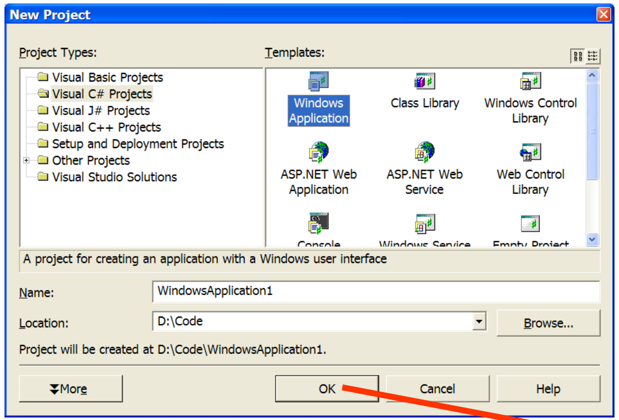
Example: a windowing application

- **GUI apps are based on the notion of <u>forms</u> and <u>controls</u>…**
  - a form represents a window
  - a form contains 0 or more controls
  - a control interacts with the user

- **Let's create a GUI app in a series of steps…**

Step 1

- **Create a new project of type “Windows Application”**
  - a form will be created for you automatically…

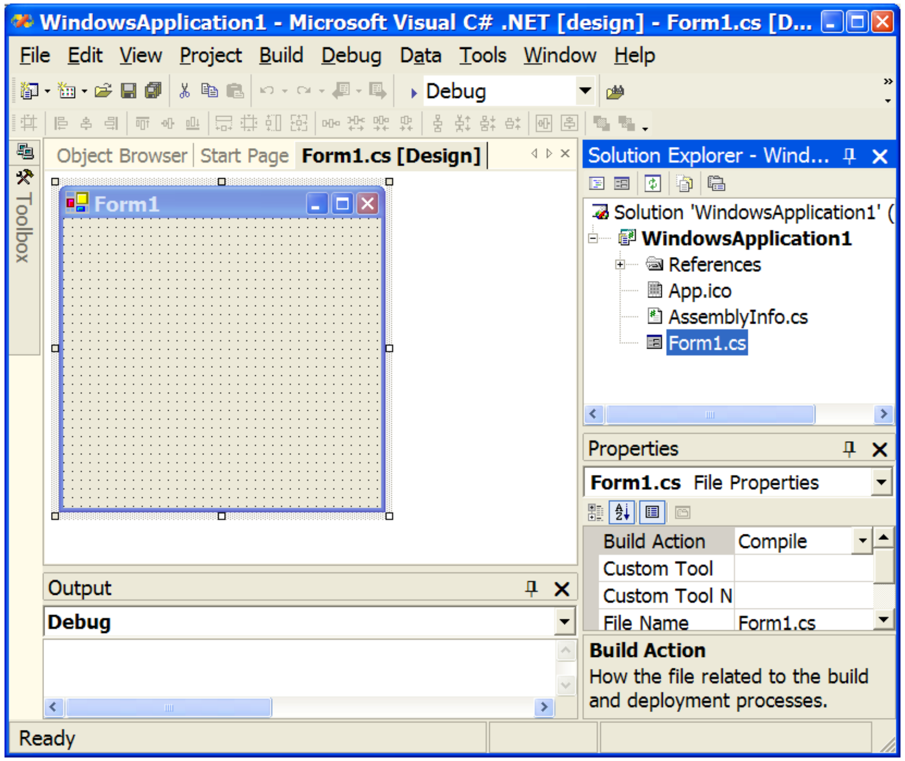


A project for creating an application with a Windows user interface

Name: WindowsApplication1

Location: D:\Code

Project will be created at D:\Code\WindowsApplication1.



Solution 'WindowsApplication1' (
WindowsApplication1
References
App.ico
AssemblyInfo.cs
Form1.cs

Debug

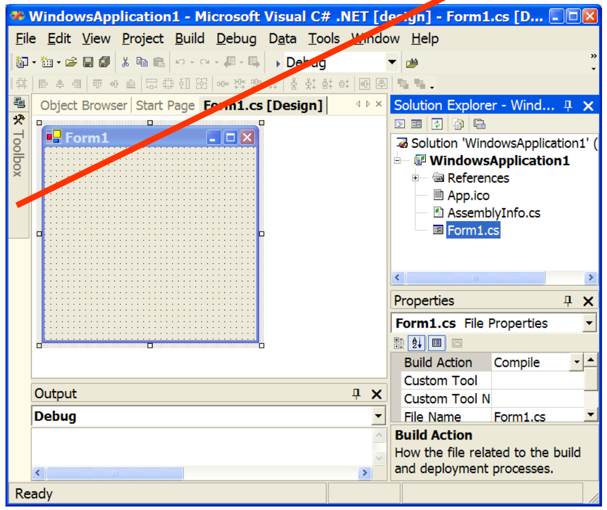
Form1.cs File Properties

Build Action	Compile
Custom Tool	
Custom Tool N	
File Name	Form1.cs

Build Action
How the file related to the build and deployment processes.

Step 2 — GUI design

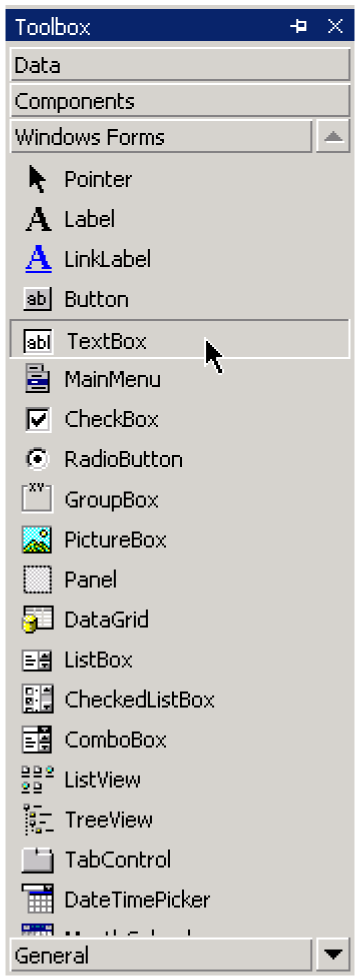
- **Select desired controls from toolbox…**
  - hover mouse over toolbox to reveal
  - drag-and-drop onto form
  - position and resize control


WindowsApplication1 - Microsoft Visual C# .NET [design] - Form1.cs [D...
File Edit View Project Build Debug Data Tools Window Help
Debug
Toolbox
Object Browser Start Page Form1.cs [Design]
Form1
Solution Explorer - Wind...
Solution 'WindowsApplication1' (
WindowsApplication1
References
App.ico
AssemblyInfo.cs
Form1.cs
Properties
Output
Debug
Ready

Form1.cs File Properties

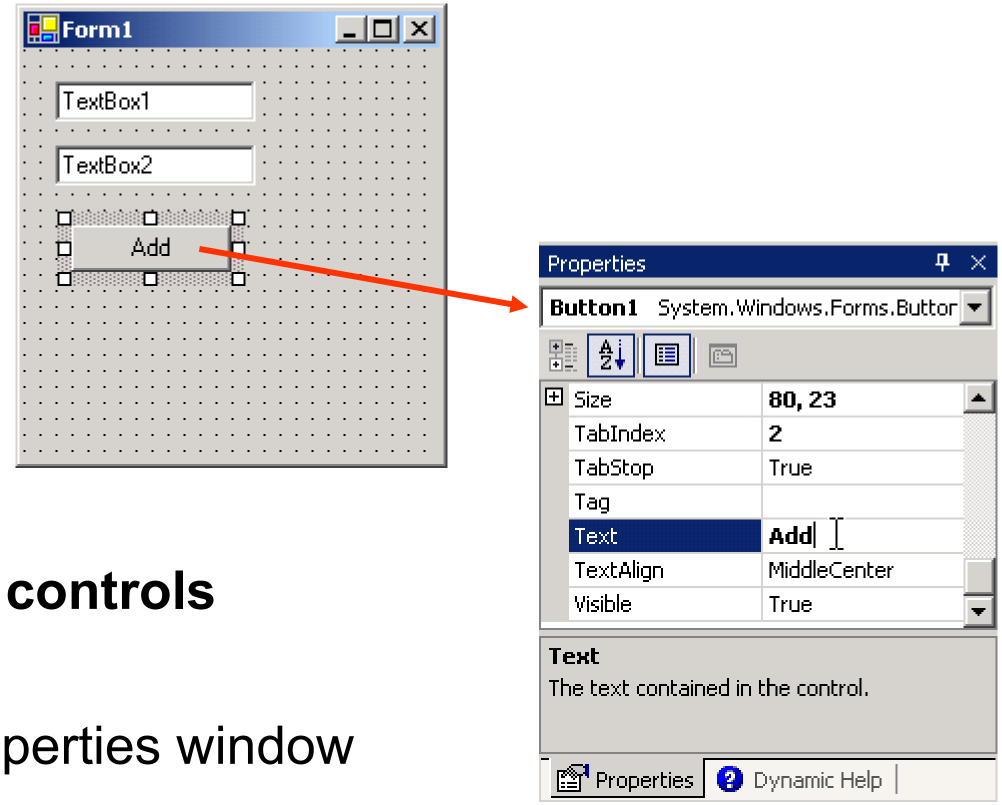
Build Action	Compile
Custom Tool	
Custom Tool N	
File Name	Form1.cs

Build Action
How the file related to the build and deployment processes.



GUI design cont'd…

- **A simple calculator:**



Button1 System.Windows.Forms.Button

Size	80, 23
TabIndex	2
TabStop	True
Tag	
Text	Add
TextAlign	MiddleCenter
Visible	True

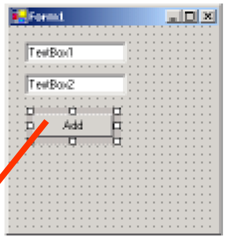
Text
The text contained in the control.

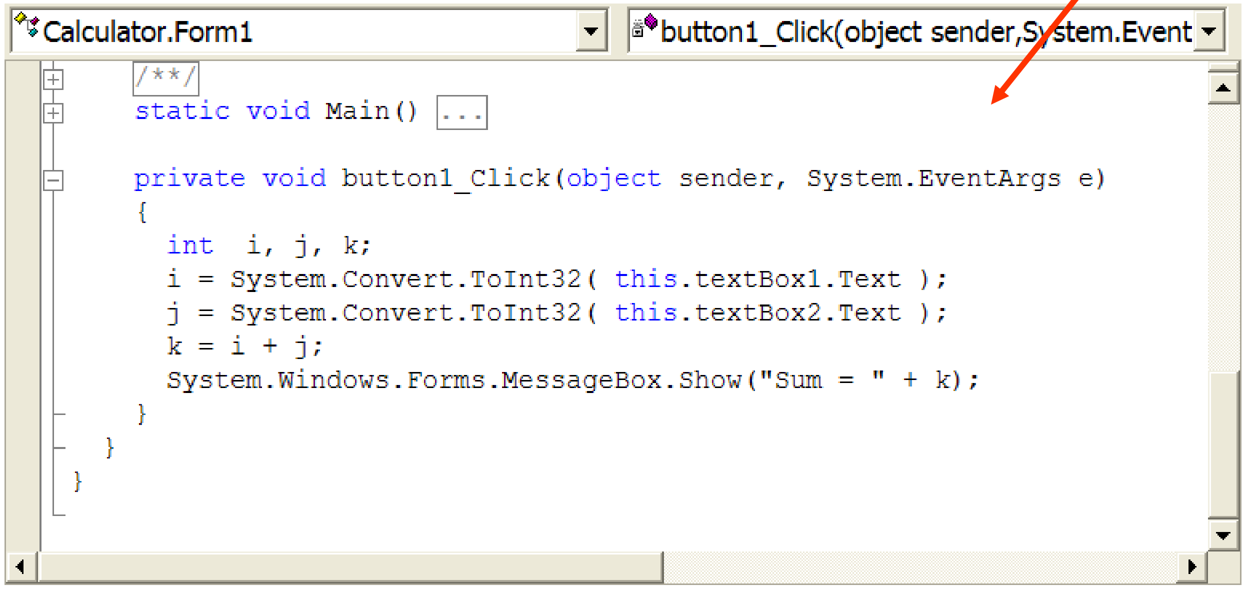
Properties | Dynamic Help

- **Position and configure controls**
  - click to select
  - set properties via Properties window

Step 3 — code design

- "Code behind" the form…
- **Double-click the control you want to program**
  - reveals coding window

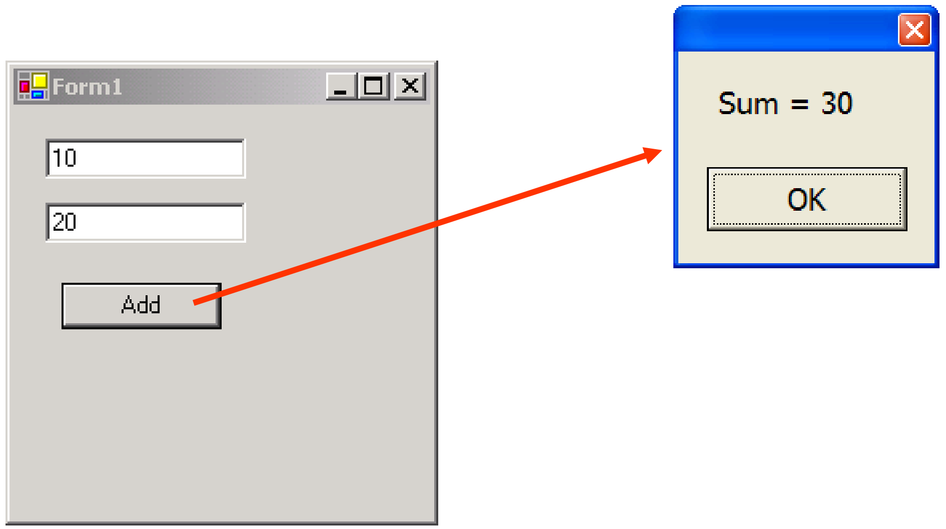




```
/**/
static void Main() ...

private void button1_Click(object sender, System.EventArgs e)
{
  int  i, j, k;
  i = System.Convert.ToInt32( this.textBox1.Text );
  j = System.Convert.ToInt32( this.textBox2.Text );
  k = i + j;
  System.Windows.Forms.MessageBox.Show("Sum = " + k);
}
  }
}
```

Step 4 — run mode

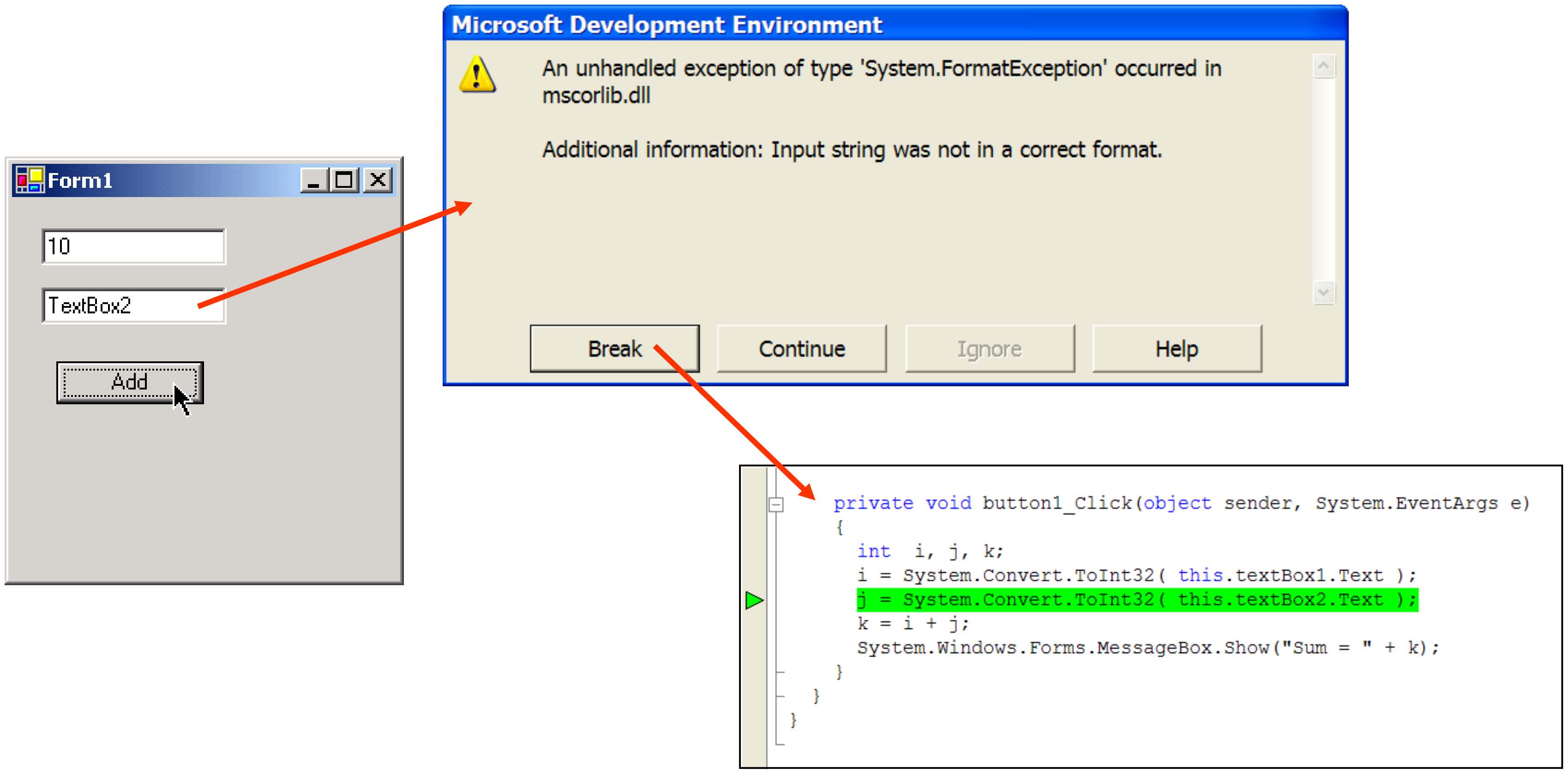
- **Run!**



Sum = 30

Break mode?

- **Easily triggered in this application via invalid input…**



An unhandled exception of type 'System.FormatException' occurred in mscorlib.dll

Additional information: Input string was not in a correct format.

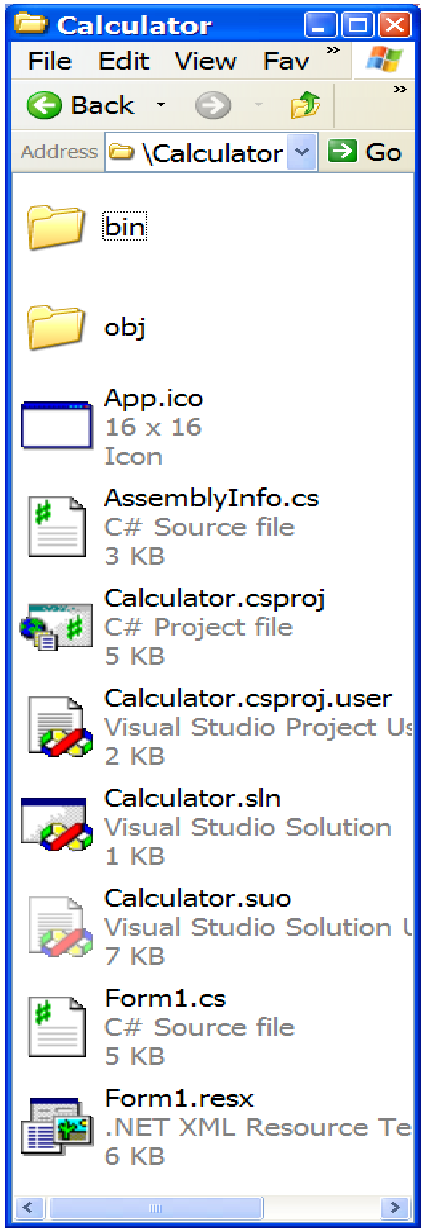
```
private void button1_Click(object sender, System.EventArgs e)
{
  int  i, j, k;
  i = System.Convert.ToInt32( this.textBox1.Text );
  j = System.Convert.ToInt32( this.textBox2.Text );
  k = i + j;
  System.Windows.Forms.MessageBox.Show("Sum = " + k);
}
}
}
```

Working with Visual Studio

- **In Visual Studio, you work in terms of source files, projects & solutions**

- **Source files contain code**
  - end in .cs, .vb, etc.
- **Project files represent 1 assembly**
  - used by VS to keep track of source files
  - all source files must be in the same language
  - end in .csproj, .vbproj, etc.

- **Solution (*.sln) files keep track of projects**
  - so you can work on multiple projects



bin

obj

App.ico
16 x 16
Icon

AssemblyInfo.cs
C# Source file
3 KB

Calculator.csproj
C# Project file
5 KB

Calculator.csproj.user
Visual Studio Project Us
2 KB

Calculator.sln
Visual Studio Solution
1 KB

Calculator.suo
Visual Studio Solution U
7 KB

Form1.cs
C# Source file
5 KB

Form1.resx
.NET XML Resource Te
6 KB

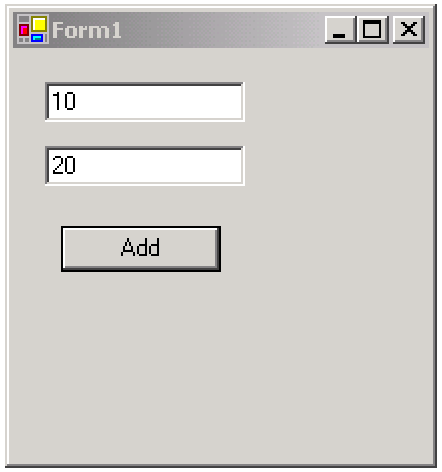
Part 3

- WinForms…

WinForms

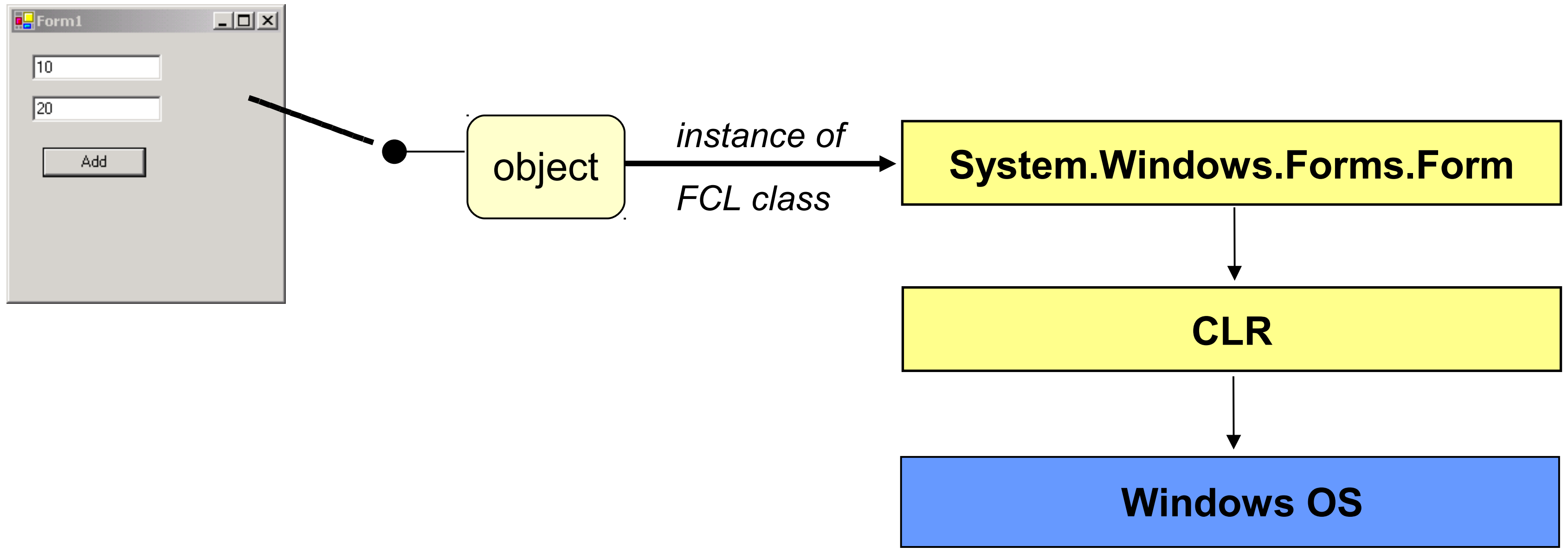
- **Another name for traditional, Windows-like GUI applications**
  - vs. WebForms, which are web-based

- **Implemented using FCL**
  - hence portable to any .NET platform



Abstraction

- **FCL acts as a layer of abstraction**
  - separates WinForm app from underlying platform



Form properties

- **Form properties typically control visual appearance:**

```
Form1  form;
form = new Form1();
form.WindowState = FormWindowState.Maximized;
form.Show();
```

  - `AutoScroll`
  - `BackgroundImage`
  - `ControlBox`
  - `FormBorderStyle` (sizable?)
  - `Icon`
  - `Location`
  - `Size`
  - `StartPosition`
  - `Text` (i.e. window's caption)
  - `WindowState` (minimized, maximized, normal)

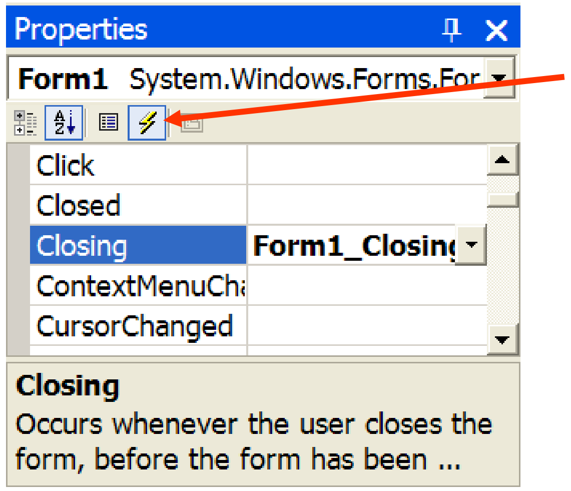
Form methods

```
form.Hide();
 .
 .
 .
form.Show();
```

- **Actions you can perform on a form:**
  - `Activate`: give this form the focus
  - `Close`: close & release associated resources
  - `Hide`: hide, but retain resources to show form later
  - `Refresh`: redraw
  - `Show`: make form visible on the screen, & activate
  - `ShowDialog`: show modally

Form events

- **Events you can respond to:**
  - bring up properties window
  - double-click on event name



Properties

Form1 System.Windows.Forms.For

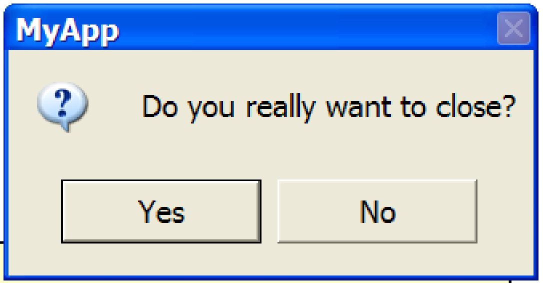
Click	
Closed	
Closing	Form1_Closing
ContextMenuCh	
CursorChanged	

Closing
Occurs whenever the user closes the form, before the form has been ...

  - `Load`: occurs just before form is shown for first time
  - `Closing`: occurs as form is being closed (ability to cancel)
  - `Closed`: occurs as form is definitely being closed
  - `Resize`: occurs after user resizes form
  - `Click`: occurs when user clicks on form's background
  - `KeyPress`: occurs when form has focus & user presses key

Example

- **Ask user before closing form:**


MyApp

Do you really want to close?

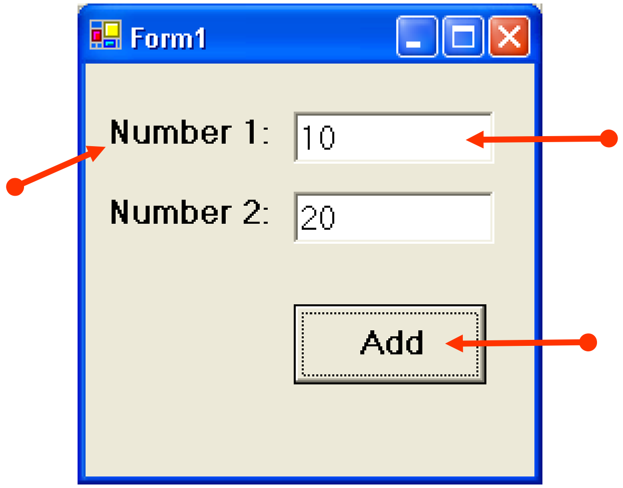
```
private void Form1_Closing(object sender,
                           System.ComponentModel.CancelEventArgs e)
{
   DialogResult  r;

   r = MessageBox.Show("Do you really want to close?",
                       "MyApp",
                       MessageBoxButtons.YesNo,
                       MessageBoxIcon.Question,
                       MessageBoxDefaultButton.Button1);
   if (r == DialogResult.No)
     e.Cancel = true;
}
```

Part 4

- **Controls…**

Controls

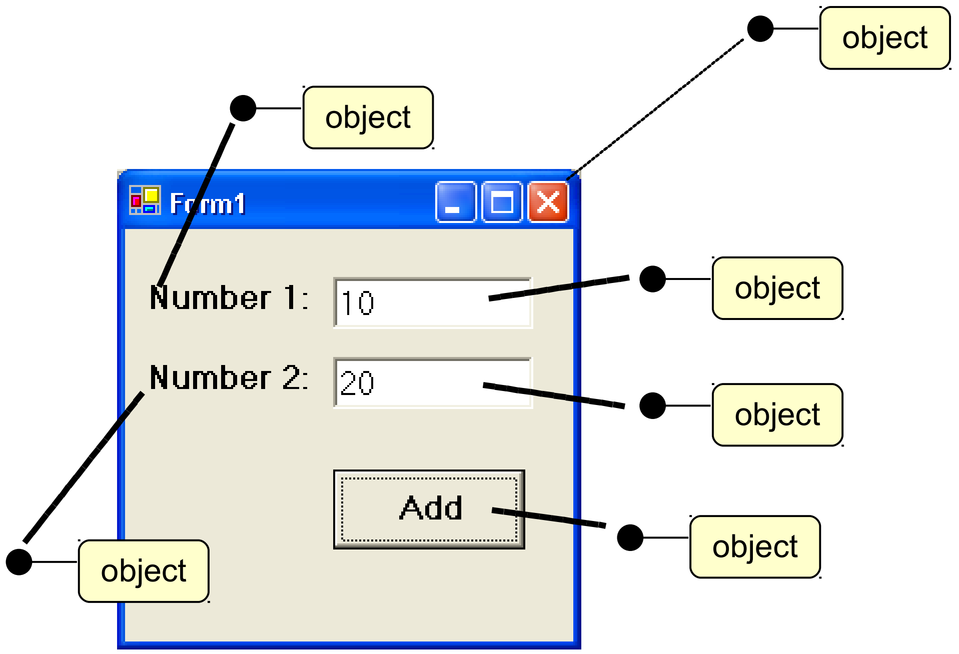
- **User-interface objects on the form:**
  - labels
  - buttons
  - text boxes
  - menus
  - list & combo boxes
  - option buttons
  - check boxes
  - etc.



Abstraction

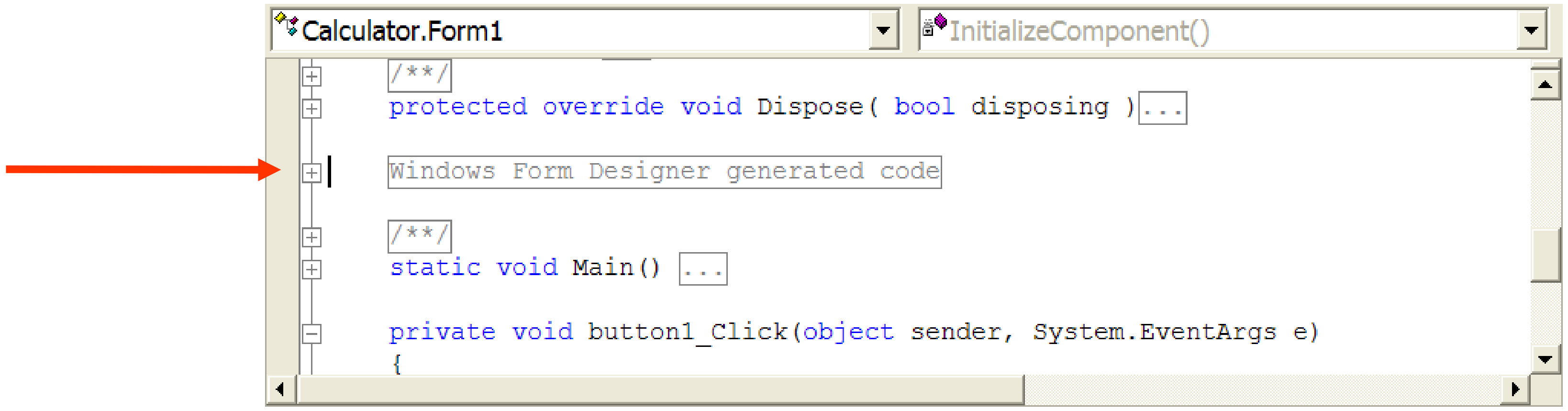
- **Like forms, controls are based on classes in the FCL:**
  - System.Windows.Forms.Label
  - System.Windows.Forms.TextBox
  - System.Windows.Forms.Button
  - etc.

- **Controls are instances of these classes**



Who creates all these objects?

- **Who is responsible for creating control instances?**
  - code is auto-generated by Visual Studio
  - when form object is created, controls are then created…



```
/**/
protected override void Dispose( bool disposing )...

Windows Form Designer generated code

/**/
static void Main() ...

private void button1_Click(object sender, System.EventArgs e)
{
```

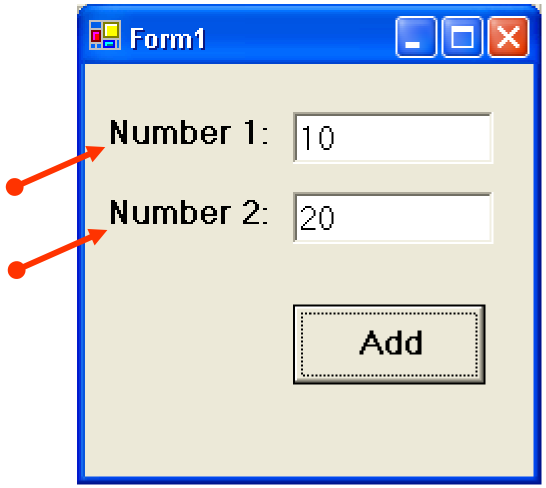
Naming conventions

- **Set control's name via Name property**
- **A common naming scheme is based on prefixes:**
  - `cmdOK` refers to a command button control
  - `lstNames` refers to a list box control
  - `txtFirstName` refers to a text box control

Labels

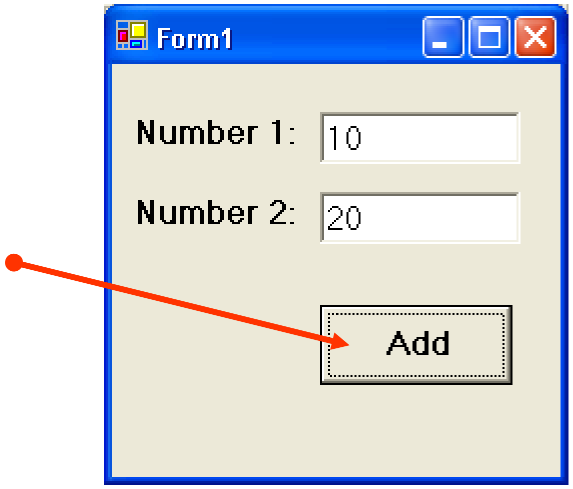
- **For static display of text**
  - used to label other things on the form
  - used to display read-only results

- **Interesting properties:**
  - `Text:` what user sees
  - `Font:` how he/she sees it



Command buttons

- **For the user to click & perform a task**
- **Interesting properties:**
  - `Text`: what user sees
  - `Font`: how he/she sees it
  - `Enabled`: can it be clicked
- **Interesting events:**
  - `Click`: occurs when button is "pressed"

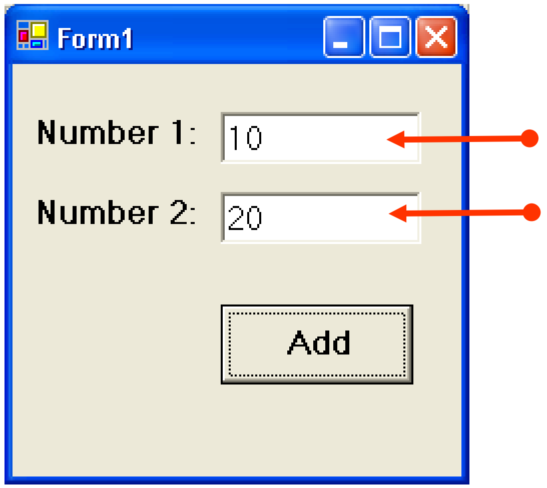


```
private void cmdAdd_Click(...)
{
   int  i, j, k;
   i = System.Convert.ToInt32( this.txtNum1.Text );
   j = System.Convert.ToInt32( this.txtNum2.Text );
   k = i + j;
   MessageBox.Show( "Sum = " + k.ToString() );
}
```

Text boxes

- **Most commonly used control!**
  - for displaying text
  - for data entry

- **Lots of interesting features…**



Text box properties

- **Basic properties:**
  - `Text`: denotes the entire contents of text box (a string)
  - `Modified`: has text been modified by user? (True / False)
  - `ReadOnly`: set if you want user to view text, but not modify

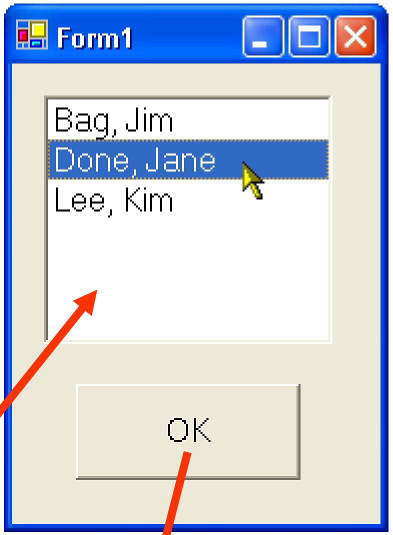
- **Do you want multi-line text boxes?**
  - `MultiLine`: True allows multiple lines of text
  - `Lines`: array of strings, one for each line in text box
  - `ScrollBars`: none, horizontal, vertical, or both

Text box events

- **Interesting events:**
  - `Enter`, `Leave`: occurs on change in focus
  - `KeyPress`: occurs on ascii keypress
  - `KeyDown`, `KeyUp`: occurs on any key combination
  - `TextChanged`: occurs whenever text is modified
  - `Validating` and `Validated`
    - Validating gives you a chance to cancel on invalid input

List Boxes

- **Great for displaying / maintaining list of data**
  - list of strings
  - list of objects (list box calls ToString() to display)



```
Customer[]  customers;
 .
 . // create & fill array with objects...
 .

// display customers in list box
foreach (Customer c in customers)
  this.listBox1.Items.Add(c);
```

```
// display name of selected customer (if any)
Customer  c;
c = (Customer) this.listBox1.SelectedItem;
if (c == null)
  return;
else
  MessageBox.Show(c.Name);
```

And much more…MSDN tutorials online