University of Anbar
College of Computer Science
and Information Technology
Computer Network Systems Department

Visual Programming I

Lecture Seven

Third Stage

First Course 2021 - 2022

Seddiq Qais Abd Al-Rahman

MSc Computer Science

co.sedeikaldossary@uoanbar.edu.iq

Visual Programming

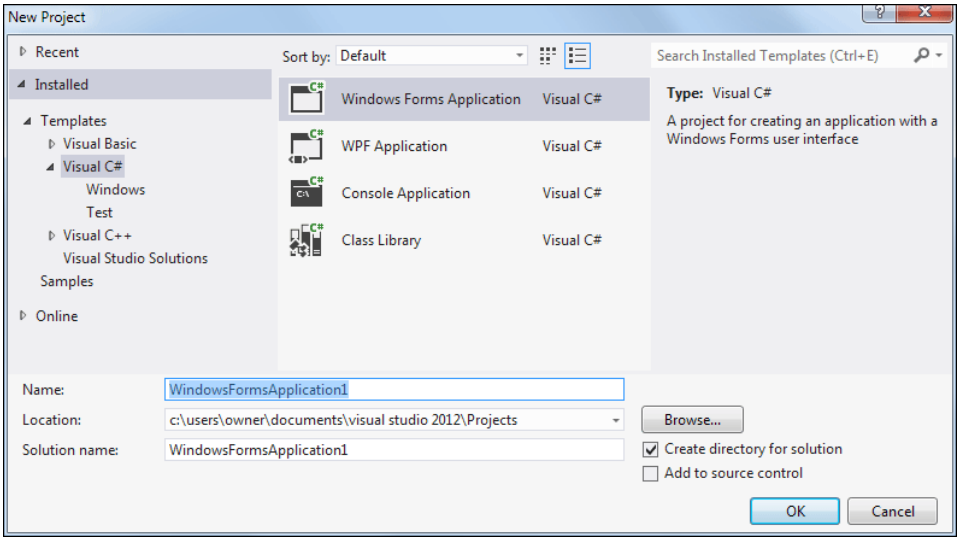
Creating Windows Applications Project

From now on, we're going to be creating Windows Applications, rather than Console Applications. Windows Applications make use of something called a **Form**. The Form is blank at first. You then add control to your form, things like buttons, text boxes, menus, check boxes, radio buttons, etc. To get your first look at a Windows Form, do the following.

If you still have your Console Application open from the previous section, click **File** from the menu bar at the top of Visual C# Express. From the File menu, click on **Close Solution**

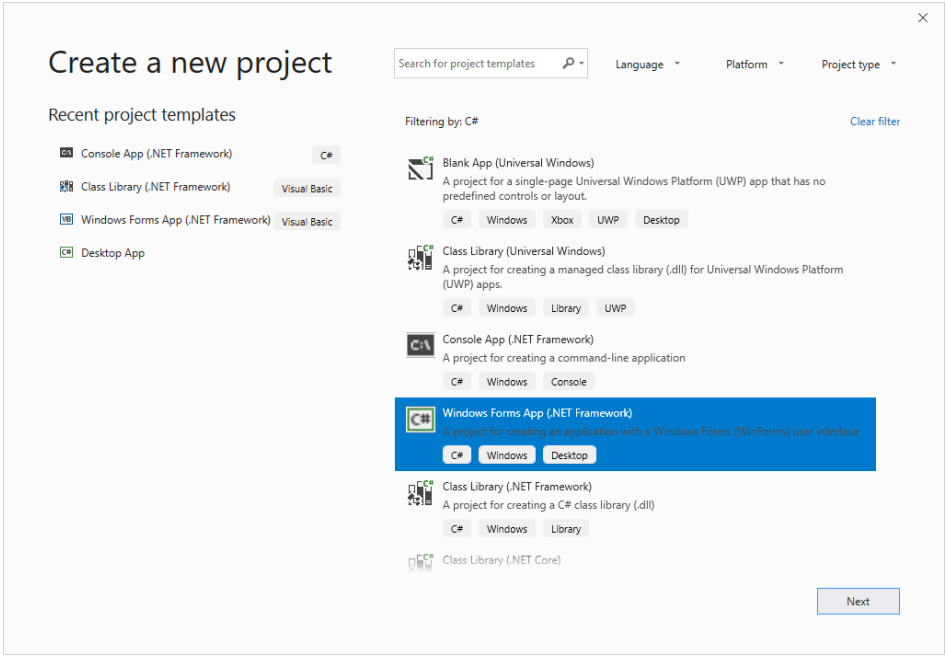
To create your first Windows form project, click the File menu again. This time, select **New Project** from the menu. When you do, you'll see the New Project dialogue box again.

If you have Visual Studio 2015 or 2017, click Visual C#, under Templates on the left:



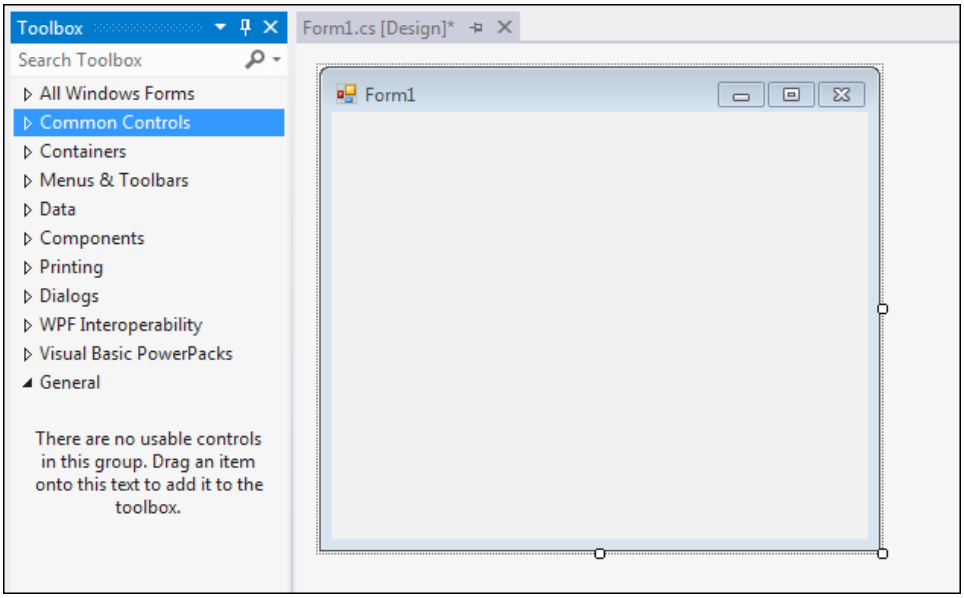
From the available templates, select **Windows Forms Application** or **Windows Forms App**. Keep the Name on the default of **WindowsFormsApplication1** (or **WindowsFormsApp1** for 2017 users) and then click OK.

If you have Visual Studio 2019, you'll see the **Create a new project** dialog box again. Select C# as the language again. This time, scroll down and select **Windows Forms App**:



When you click OK a new Windows Application project will be created for you. In 2019, you'll have another screen, **Configure your new project**. You can leave everything on the default (unless you need to change the save location) and just click **Create**.

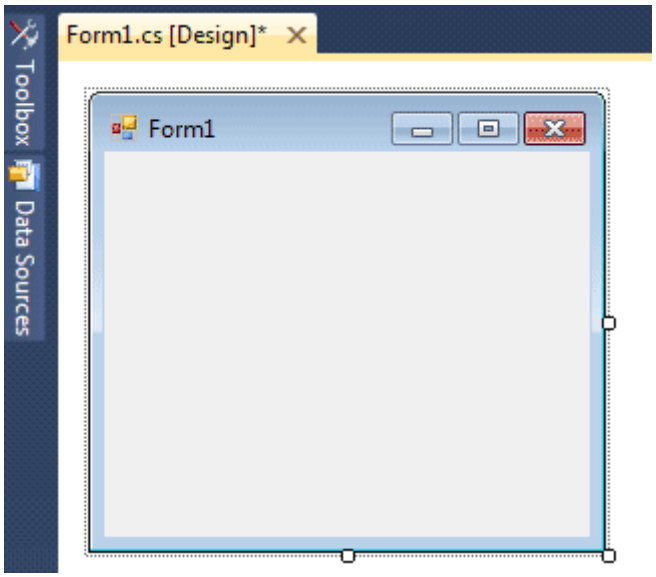
All users will then get a new project:



1. Details Windows Project

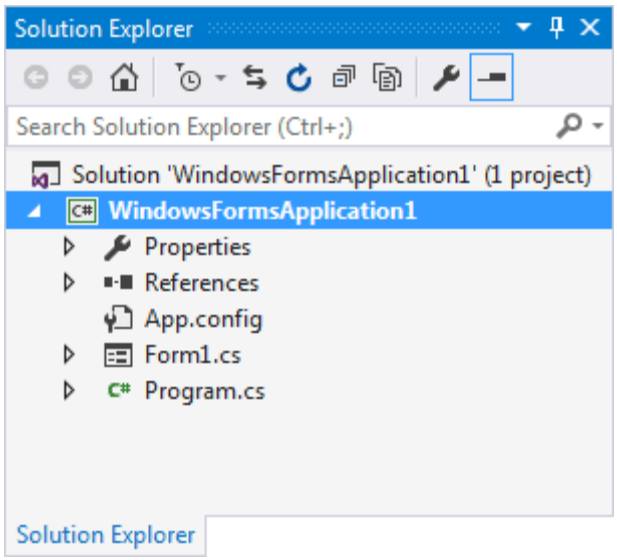
The obvious difference from the Console Application you created in the previous section is the blank Form in the main window. Notice the **Toolbox**, though, on the left hand side. We'll be adding controls from the Toolbox to that blank **Form1** you can see in the image above.

If you can't see the Toolbox, you may just see the Tab, as in the following image (the Community 2015 and 2017 editions have an extra tab called Server Explorer)::



If your screen looks like the one above, move your mouse over to the Toolbox tab. It will expand to look like the first one. If you want to permanently display the Toolbox, click on the pin symbol in the middle:

Notice the Solution Explorer on the right side of your screen. (If you can't see the Solution Explorer, click its entry on the **View** menu at the top of Visual Studio Express.) If you compare it with the Solution Explorer when you created your Console Application, you can see the similarities:



2. Program.cs File

Both projects have sections for Properties, References, and a Program.cs file. Double click the Program.cs file to open it, and you'll see some familiar code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsFormsApplication1
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```

And here's the code from the Console Application:

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

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
        }
    }
}
```

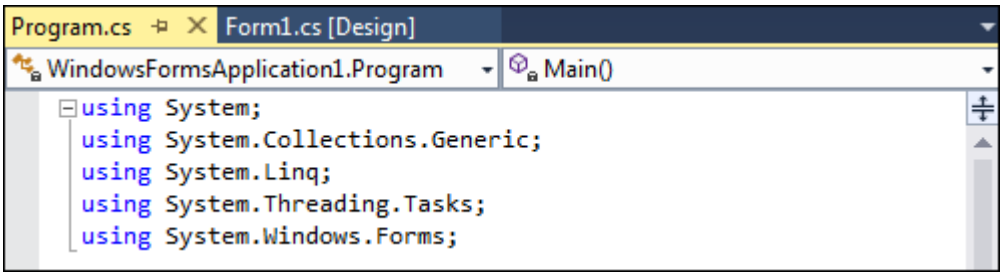
Both have the same **using** lines, a **namespace**, a class called Program, and a **Main** Method.

The Main Method is the entry point for your programme. The code between the curly brackets of Main will get executed when the programme first starts. The last line in the **WindowsApplication1** code above is the one that Runs Form1 when the Application starts.

You can do other things here. For example, suppose you had a programme that connects to a server. If it finds a connection then it loads some information from a database. In the Main Method, you could check that the server connection is OK. If it's not, display a second form; if it's OK, then display the first form.

But don't worry if all that code has you scratching your head. The thing to bear in mind here is that a method called **Main** starts your programme. And **Program.cs** in the Solution Explorer on the right is where the code for Main lives.

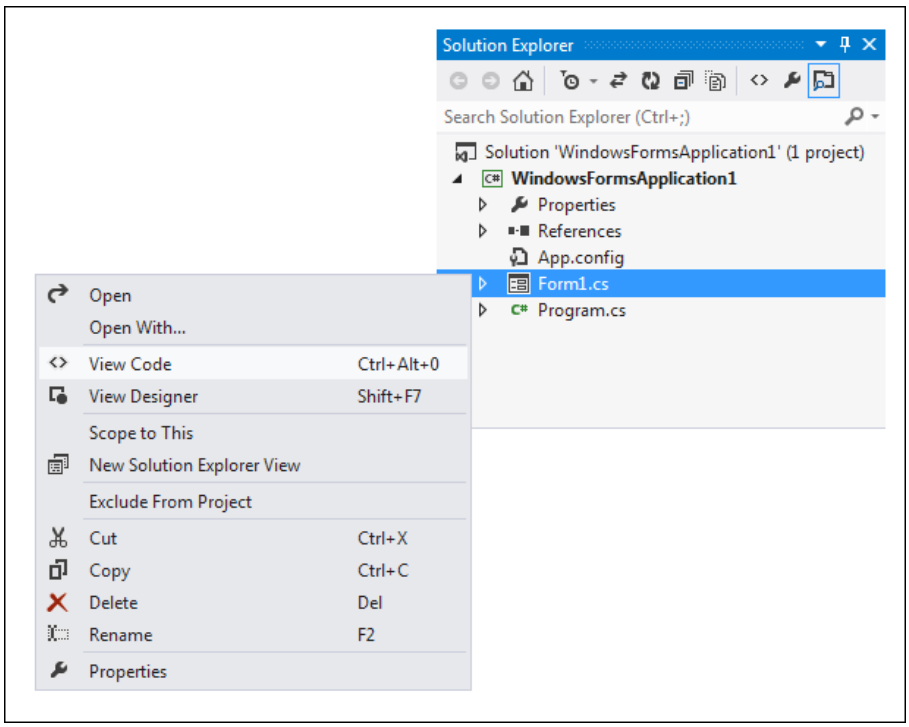
But we won't be writing code in the Program.cs file, so we can close it. Have a look near the top of the coding window, and you'll some tabs:



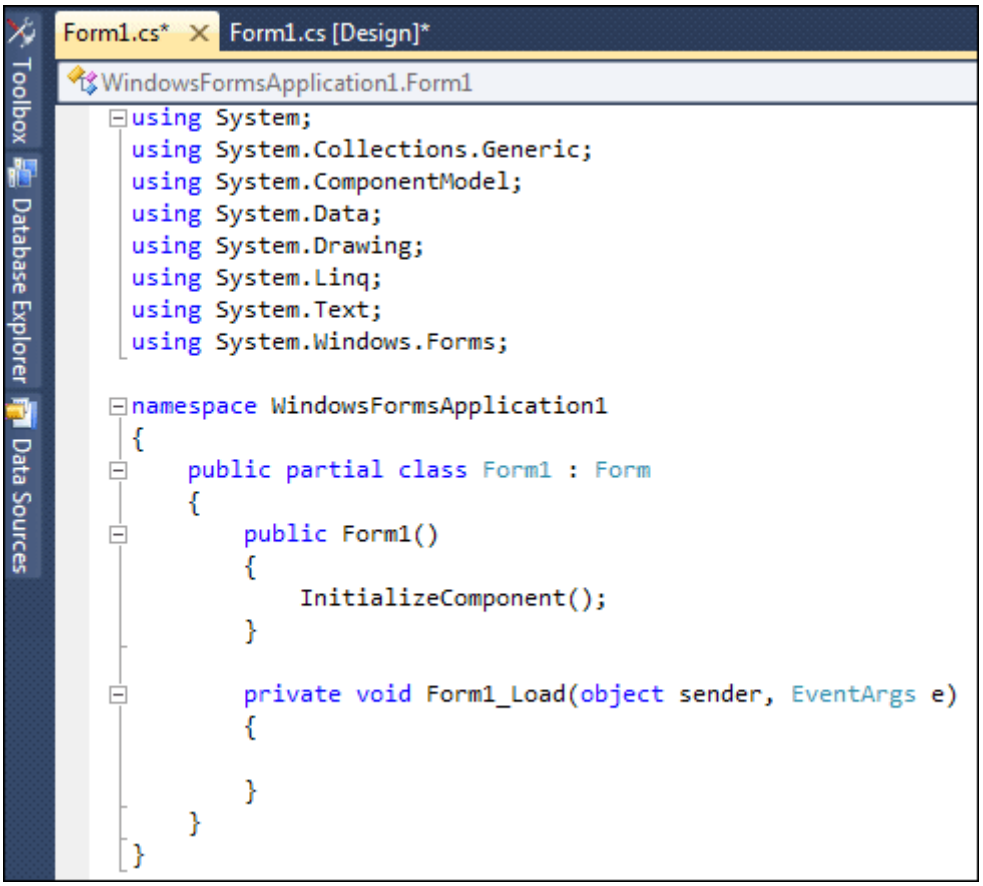
Click the X to close the tab. You should now see your form again (you may have a Start tab as well. You can close this, if you want).

3. Form Code and Design

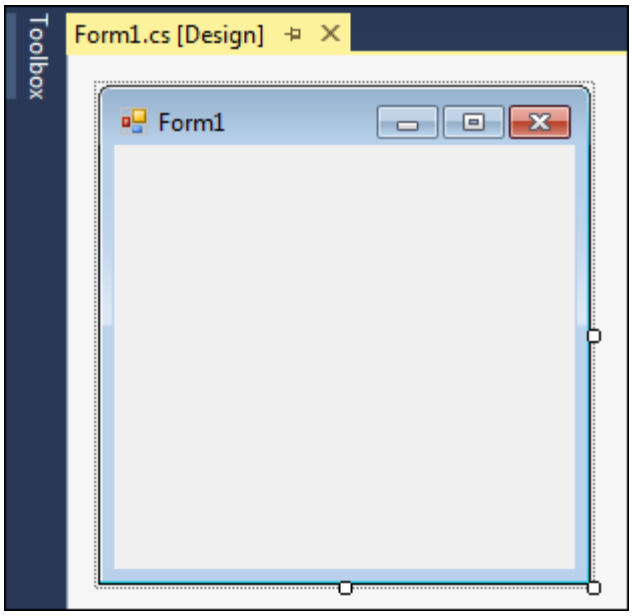
To see the window where you'll write most of your code, right click **Form1.cs** in the Solution Explorer:



The menu has options for **View Code** and **View Designer**. The Designer is the Form you can see at the moment. Click View Code from the menu to see the following window appear (you can also press the F7 key on your keyboard):

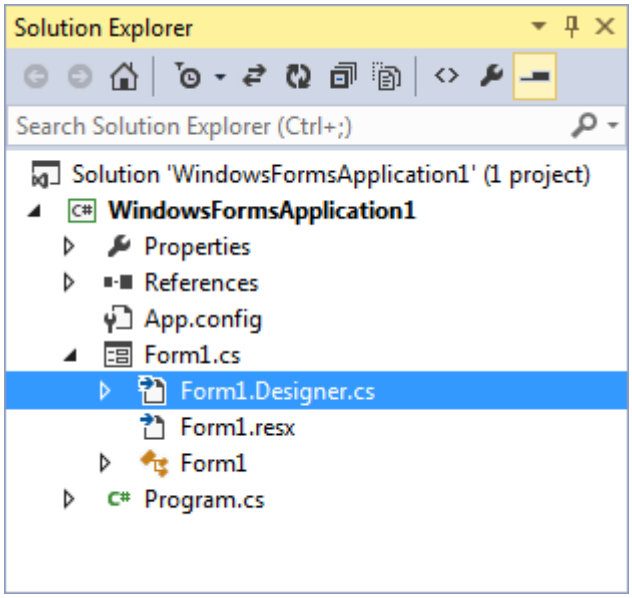


This is the code for the Form itself (ignore the Form1_Load lines as you may not have them). This Form:



The code has a lot more **using** statements than before. Don't worry about these for now. They just mean "using some code that's already been written".

The code also says **partial class Form1**. It's partial because some code is hidden from you. To see the rest of it (which we don't need to alter), click the arrow symbol next to Form1.cs in the Solution Explorer:



Now double click **Form1.Designer.cs**. You'll see the following code:

```
namespace WindowsApplication1
{
    partial class Form1
    {
        /// <summary> ...
        private System.ComponentModel.IContainer components = null;

        /// <summary> ...
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        Windows Form Designer generated code
    }
}
```

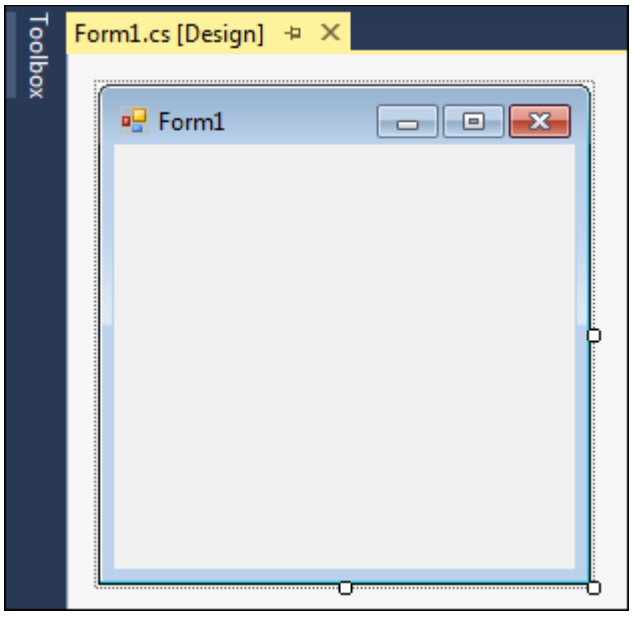
Again, you see **partial class Form1**, which is the rest of the code. Click the plus symbol next to **Windows Form Designer generated code**. You'll see the following:

```
#region Windows Form Designer generated code

/// <summary>
/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
    this.components = new System.ComponentModel.Container();
    this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
    this.Text = "Form1";
}

#endregion
```

InitializeComponent is code (a Method) that is automatically generated for you when you create a new Windows Application project. As you add things like buttons and text boxes to your form, more code will be added here for you.

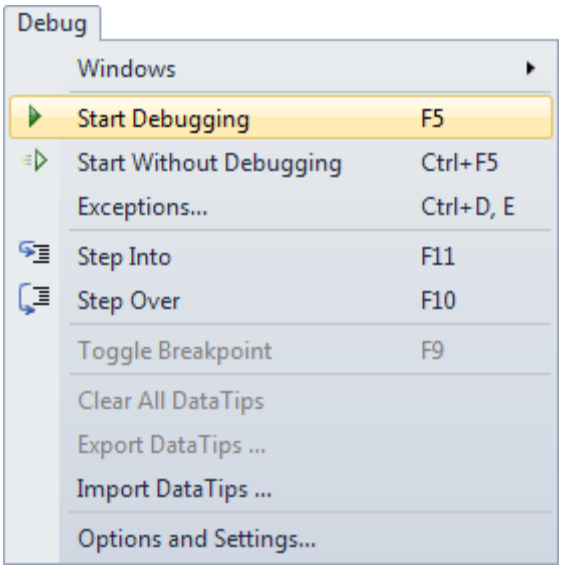
But you don't need to do anything in this window, so you can right click the **Form1.Designer.cs** tab at the top, and click **Close** from the menu. Or just click the X.

Click back on the **Form1.cs** tab at the top to see you form again. If the tab is not there, right click Form1.cs in the Solution Explorer on the right. From the menu, select **View Designer**. Here's what you should be looking at:



4. Run Project (Debugging)

It's in Designer view that we'll be adding things like buttons and text boxes to our form. But you can run this programme as it is. From the **Debug** menu at the top, click **Start Debugging** (Or you can just press the F5 key on your keyboard.):



When you click Start Debugging, Visual C# will Build the programme first, and then run it, if it can. If it can't run your programme you'll see error messages.

But you should see your form running on top of Visual Studio. It will have its own Red X, and it's own minimize and maximize buttons. Click the Red X to close your programme, and to return to Visual C# Express.

From now on, when we say Run your programme, this is what we mean: either press F5, or click **Debug > Start Debugging**. You can also select **Debug > Start Without Debugging**.

Run your picture viewer app

When you create a Windows Forms App project, you actually build a program that runs. In this tutorial, your picture viewer app doesn't do much yet—although it will. For now, it displays an empty window that shows **Form1** in the title bar.

Here's how to run your app.

1. Choose one of the following methods:

- Choose the **F5** key.
- On the menu bar, choose **Debug** > **Start Debugging**.
- On the toolbar, choose the **Start Debugging** button, which appears as follows:

Start Debugging *toolbar button*

2. Visual Studio runs your app, and a window called **Form1** appears. The following screenshot shows the app you just built. The app is running, and you'll soon add to it.

Windows Forms App, *running*

3. Go back to the Visual Studio integrated development environment (IDE), and then look at the new toolbar. Additional buttons appear on the toolbar when you run an application. These buttons let you do things like stop and start your app, and help you track down any errors (bugs) it may have. For this example, we're using it to start and stop the app.

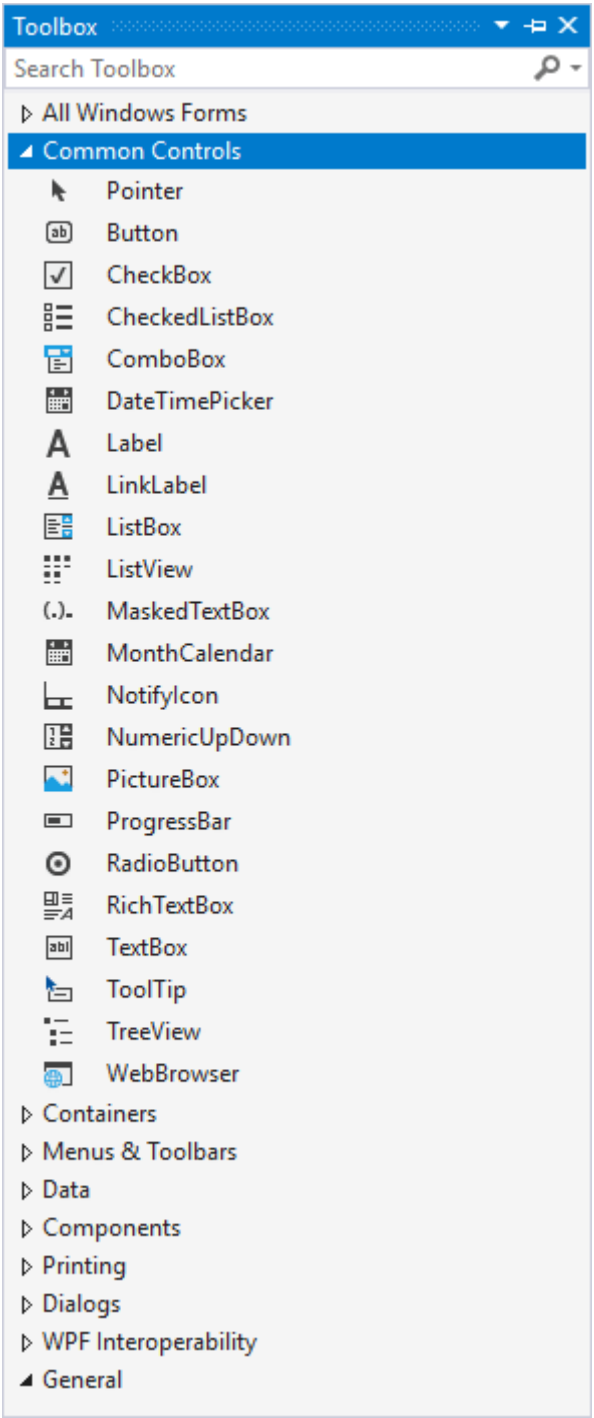
Debugging *toolbar*

4. Use one of the following methods to stop your app:
   - On the toolbar, choose the **Stop Debugging** button.
   - On the menu bar, choose **Debug** > **Stop Debugging**.
   - Use your keyboard and press **Shift**+**F5**.
   - Choose the **X** button in the upper corner of the **Form1** window.

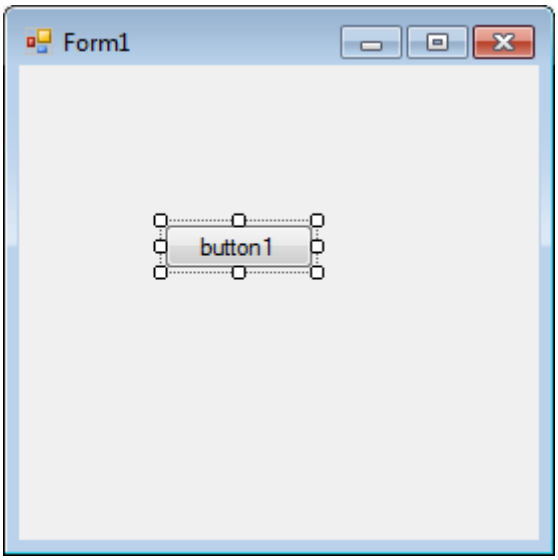
Learn about The Visual Studio Interface

Adding Controls To A Blank C# Form

The first thing we'll do is to add a button to the blank form. We'll then write a single line of code, so that you can see how things work. If you want to add a control to a form, you can use the Toolbox on the left of Visual Studio. Move your mouse over to the Toolbox, and click the arrow symbol next to Common Controls. You should see the following list of things that you can add to your form:



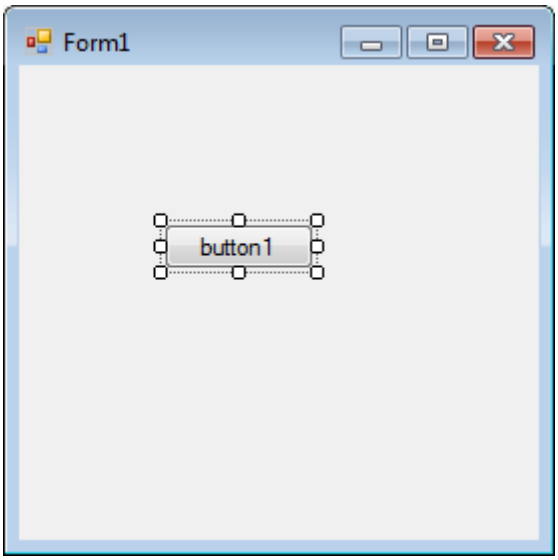
Click the Button item under the Common Controls heading. This will select it. Now click once anywhere on your form. A button will be drawn for you, and your Form will look like this:



(You can also hold down your left mouse button and drag out a button to the size you want it.) A button is something you want people to click on. When they do, the code you write gets executed. The text on the button, which defaults to "button1", can be changed. You can add anything you like here, but it should be something that's going to be useful for your users, such as "Open a text file", or "Calculate Now". We're going to display a simple message box when the button is clicked. So we need to add some text to the button. You do this by changing something called a property. This is quite easy, and you'll see how to do it in the next part of this lesson.

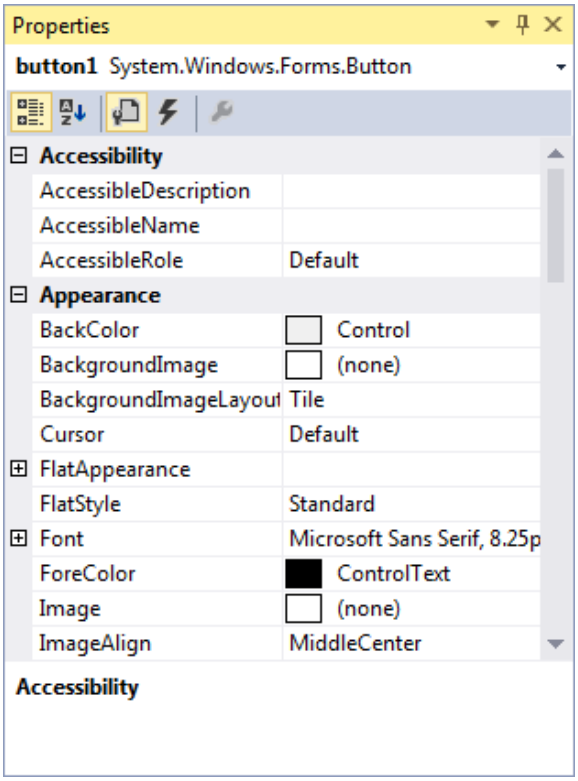
Properties Of A C# Control

The controls you add to a form have something called Properties. A property of a control is things like its Height, its Width, its Name, its Text, and a whole lot more besides. To see what properties are available for a button, make sure the button is selected, as in the image below:

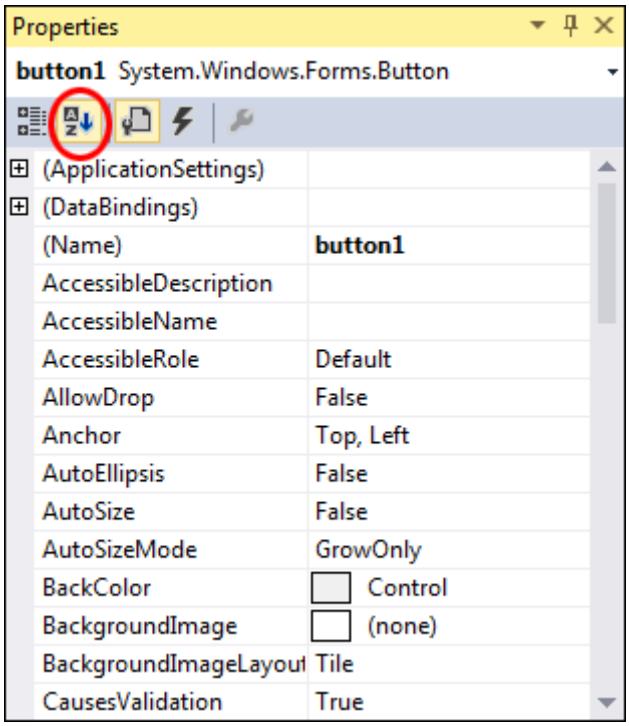


If a control is selected, it will have white squares surrounding it. If your button is not selected, simply click it once.

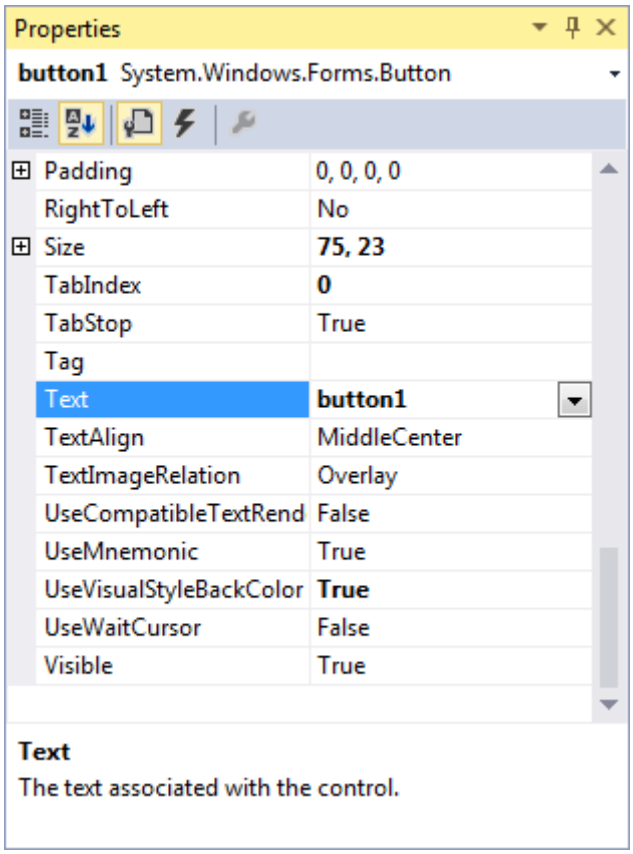
Now look in the bottom right of Visual C#, just below the Solution Explorer. You should see the Properties Window (if it's not there, select it from the View menu at the top):



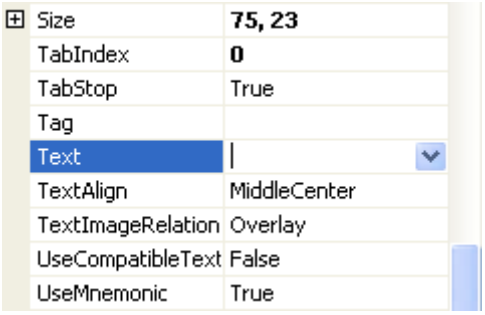
To view the list of Properties in alphabetical order, click the AZ symbol at the top, circled in red in the image below:



As you can see, there's a lot of Properties for a button. Scroll down to the bottom and locate the Text Property:



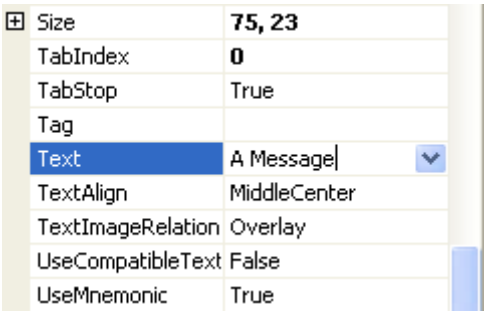
The Text Property, as its name suggests, is the Text you want to appear on the button. At the moment, it says button1. Click inside of the text area of button1. Delete the default text:



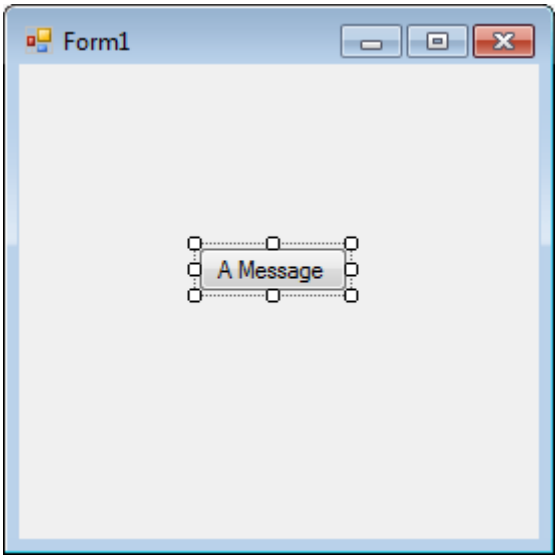
Now type the following:

A Message

The Text part of your Properties Window will then look like this:



Now press the enter key on your keyboard. Have a look at your Form, and the Text on the button should have changed:



There's a few more Properties we can change before we get to the code.

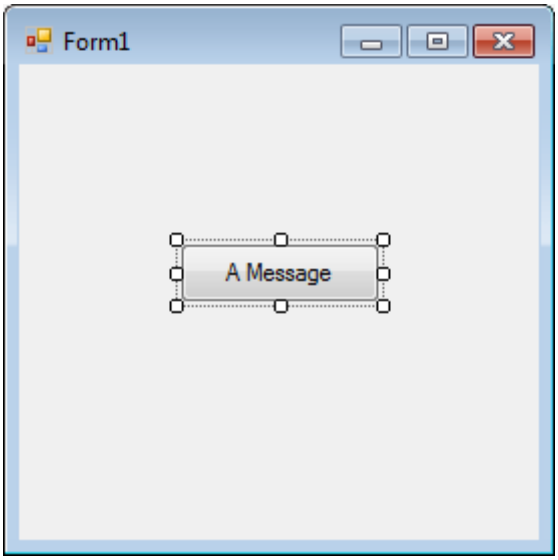
Locate Size in the Properties Window:

⊞ Padding	0, 0, 0, 0
RightToLeft	No
⊞ Size	**75, 23**
TabIndex	**0**
TabStop	True

The first number, 75, is the width of the button. The second number, 23, is the height of the button. The two numbers are separated by a comma. Change the numbers to 100, 30:

Modifiers	Private
⊞ Padding	0, 0, 0, 0
RightToLeft	No
⊞ Size	**100, 30**
TabIndex	**0**
TabStop	True
Tag	
Text	**A Message**

Press the enter key again, and the size of your button will change:



You can move your button around the Form by clicking it with the left mouse button to select it. Hold down you left mouse button and drag your button around the form. Let go of your left mouse button when you're happy with the new location.

Exercise
You can also move a button by changing its Location property. Use the Properties Window to change the Location of your button to a position on the form of 100, 50. (100 means 100 units from the left edge of the form; 50 means 50 units down from the top of the form.)

Exercise
A Form also has lots of Properties. Click away from the button and on to the Form itself. The Properties for the form will appear in the Properties Window. Change the Text Property of the Form to A First Message.

Exercise

The Form, like the button, also has a Size Property. Change the Size of the Form to 300, 200.

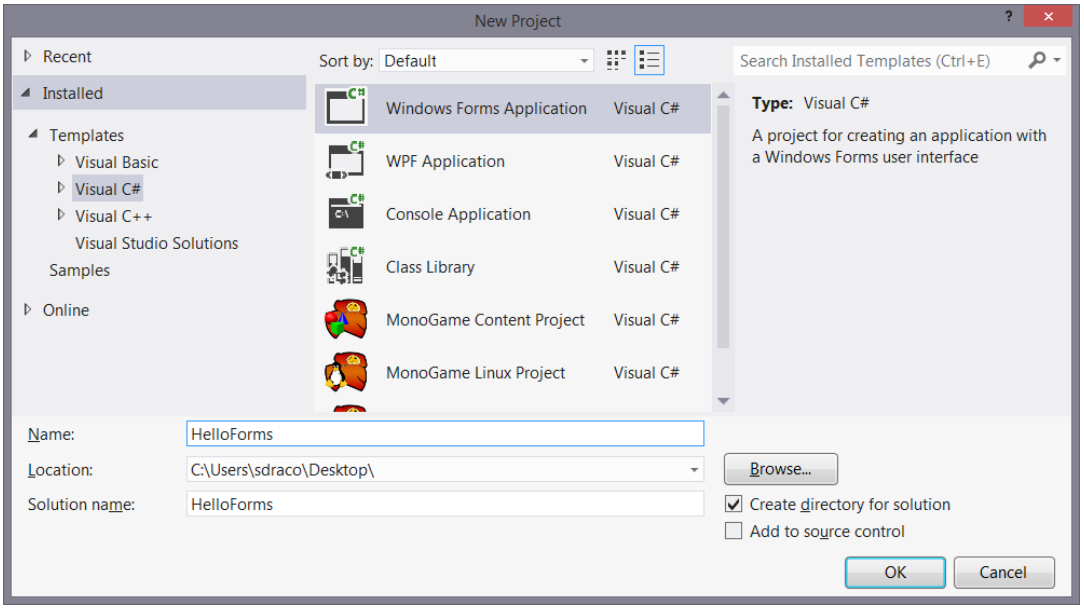
After you complete the three exercises above, your Form should look like this:

When you changed the Text property of the Form, you changed the text that runs across the blue bar at the top, the text in white. You can type anything you like here, but it should be something that describes what the form is all about or what it does. Often, you'll see the name of the software here, like Microsoft Word, or Microsoft Visual Studio.

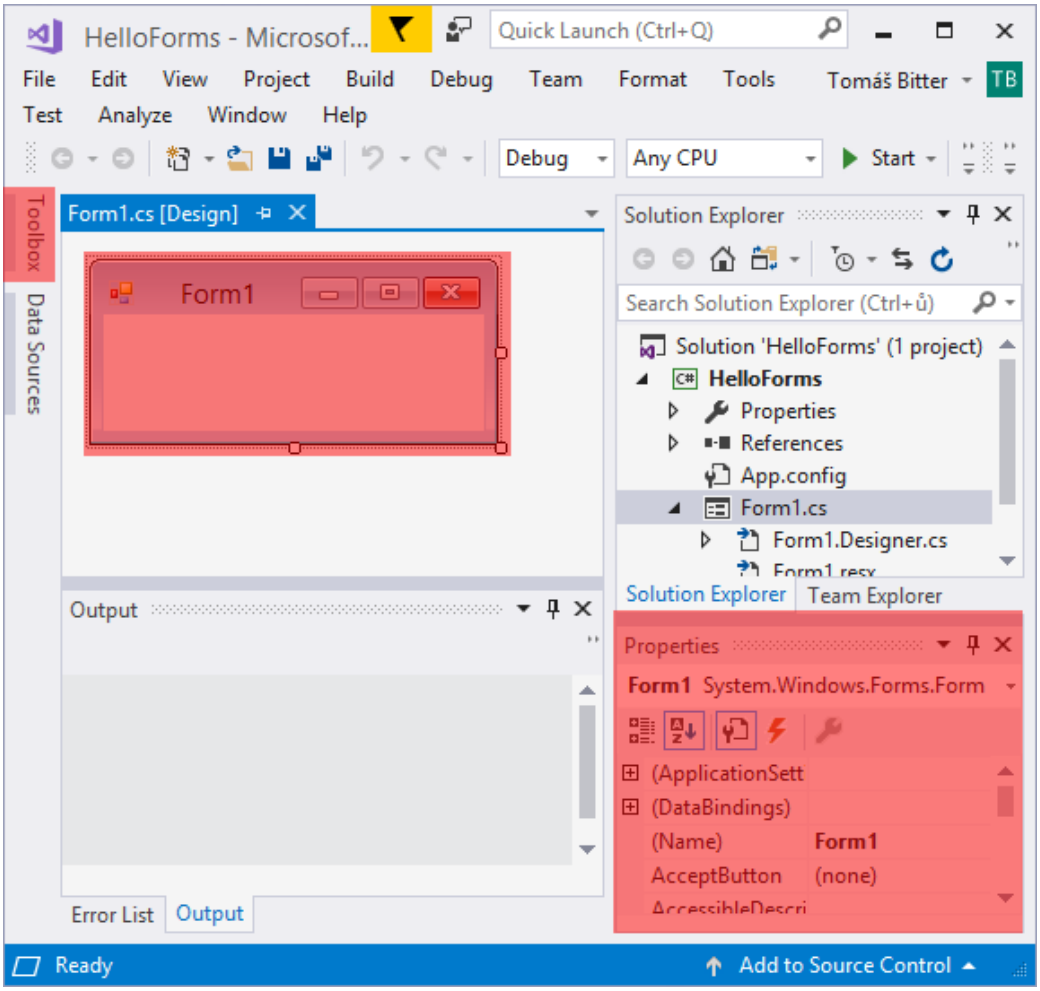
Sample Running

We won't start with anything other than the classic Hello World application, this time in forms If you haven't read local course, let me repeat that it's a simple application that does nothing but write some text.

Create a new project, select Windows Forms Application as the project type. Type HelloForms as the project name:



Your Visual Studio window should now look something like this:



Let's describe its individual parts, which we'll use to develop form applications. Important parts are highlighted in red in the image above.

Designer - In the Designer we see what the form looks like. So far, it's just an empty window.

Properties - In the Properties window, we can see the properties of the currently selected element on the form. If you don't see the window, enable it in View -> Properties Window.

Toolbox - Toolbox is a sliding window that serves as a palette with individual controls that we can add to the form.

Setting control properties

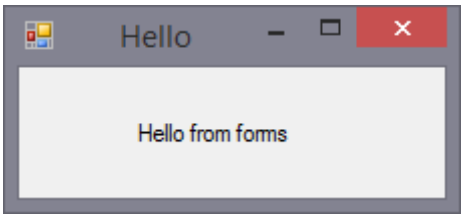
When we select an element on the form or the form itself, we can change the properties of that element in the Properties window.

Since we got no element on the form, it's the form what is selected. We'll change the title to "Hello". Find the Text property and enter Hello in it. The result is reflected in the designer immediately. This way we'll set the properties of all elements on the form.

Adding Controls to the Form

Now we'll open the Toolbox and select the Label control, which is a text label. We'll insert it into the form either by double-clicking on it or by dragging it. Reduce the form size and simply drag the label in the middle. In the Properties window, set the label to "Hello from forms".

As always, you can start your first window application with the green Play button or the F5 key. You should get a similar result:



Under the Hood

Let's explain how the application works inside. The form itself is of course an object (how else). It's declared as the Form1 class which we can find in the Form1.cs file. Of course you can rename the file in Solution Explorer, the class will also be renamed. For our application, the form could be named HelloForm, rename it so you can navigate in it better.

Visual Studio displays either a visual preview of the form or its source code. We can switch between these modes either by right-clicking on the form (resp. on the code) and selecting View Code (resp. View Designer). It's useful to know the keyboard shortcuts Shift + F7 to move to the designer and Ctrl + Alt + 0 to move to the code. It must be the zero on the alphanumeric keyboard (the left one).

Move to form code that looks like this (I omitted the initial using statements):

```
namespace HelloForms
{
    public partial class HelloForm : Form
    {
        public HelloForm()
        {
            InitializeComponent();
        }
    }
}
```

We can see that the form is a class inherited from the Form class. We can't see any trace of anything we added or set to the form, only the constructor calls the strange InitializeComponent() method.

The class is marked as partial, meaning it's defined in multiple files. Specifically, there's also a HelloForm.Designer.cs file that contains less readable code that is automatically generated by us clicking in the designer.

This code is separated into another file on purpose, to make the form's source code clear. Never tamper with the Designer.cs file manually, you even wouldn't have to know that it exists. But let's look at its content to understand how the application works:

```
namespace HelloForms
{
   partial class HelloForm
   {
      /// <summary>
      /// Required designer variable.
      /// </summary>
      private System.ComponentModel.IContainer components = null;
      /// <summary>
      /// Clean up any resources being used.
      /// </summary>
      /// <param name="disposing">true if managed resources should be disposed;
otherwise, false.</param>
      protected override void Dispose(bool disposing)
      {
         if (disposing && (components != null))
         {
            components.Dispose();
         }
         base.Dispose(disposing);
      }
      #region Windows Form Designer generated code
      /// <summary>
      /// Required method for Designer support - do not modify
      /// the contents of this method with the code editor.
      /// </summary>
      private void InitializeComponent()
      {
         this.label1 = new System.Windows.Forms.Label();
         this.SuspendLayout();
         //
         // label1
         //
         this.label1.AutoSize = true;
         this.label1.Location = new System.Drawing.Point(75, 32);
```

```
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(135, 17);
            this.label1.TabIndex = 0;
            this.label1.Text = "Hello from forms";
            //
            // Form1
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(8F, 16F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(282, 80);
            this.Controls.Add(this.label1);
            this.Name = "Form1";
            this.Text = "Helloz";
            this.ResumeLayout(false);
            this.PerformLayout();
        }
        #endregion
        private System.Windows.Forms.Label label1;
    }
}
```

In the code, we see the hidden InitializeComponent() method, which does nothing but create all the elements on the form one by one and set the appropriate properties we've chosen. Here we can see how our label is created and its properties set. The method is then called in the constructor to initialize the form. An unaware programmer is then completely encapsulated from the code that the designer generates. Of course, this is mainly to prevent them from breaking it However, it's important to know how it works to be able to, for example, add controls at runtime or fix errors in the designer file.

References:

- Tony Gaddis, “Starting out with Visual C#.”, Fourth edition, Boston, Pearson Inc., 2017.
- Salvatore A. Buono, “C# and Game Programming: A Beginner's Guide.” Second Edition, Boca Raton, CRC Press Inc., 2019.
- Eric Butow and Tommy Ryan, "C#: Your Visual Blueprint for Building .NET Applications.", Hungry Minds Inc., New York, 2002.
- Faraz Rasheed, “Programmer's Heaven: C# School.”, First Edition, Fuengirola, Synchron Data, 2006.