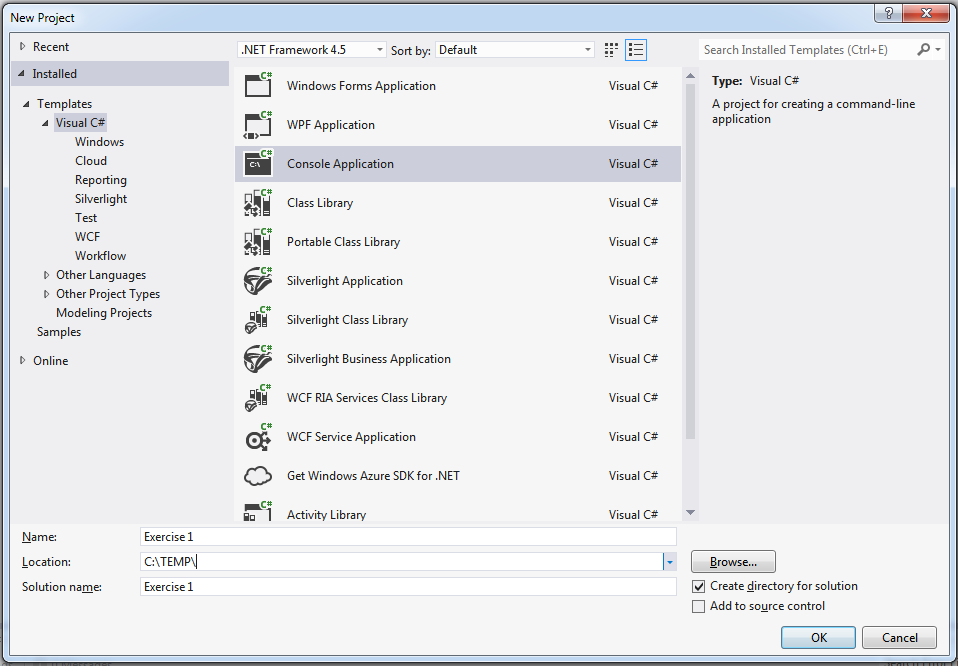
**Exercise 1. Compiling a C# program in Visual Studio**

The purpose of these lab sessions is to make sure you can write and compile simple C# programs in Visual Studio.

Copies of these exercises WITH ANSWERS will appear on Moodle on later. If you’re stuck and want to see an answer sheet / explanation now, give me or Stefan a shout.

1. Load Visual Studio 2012
2. From the start page select new project
3. Select New Console application in the Visual C# templates section and name it HelloWorld or similar. The Solution name can be the same (One solution can contain many projects – visible in the Solution Explorer)
4. 
   1. **Note about choosing a location**: Last year projects had to be saved to either to something like c:\TEMP or USB thumbdrive because permissions prevented the projects compiling correctly over a network. If you encounter this, work on the project in one of these locations, then if you want to keep it, copy the folder to your networked drive when you’re done.
5. **As this may be your first C# program, lets start with “Hello, World”** !
   1. Type in the following, into the generated “Program.cs” source file as required. You can replace what is already in there, or just add lines from the example below:

// A Hello World! program in C#.

using System;

namespace HelloWorld

{

class Hello

{

static void Main()

{

Console.WriteLine("Hello World!");

// Keep the console window open in debug mode.

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

Console.ReadKey();

}

}

}

* 1. Press F5 to compile
     1. **Note**: to run a program click on Debug->Start Debugging (Or Start Without Debugging to run the ‘release mode’ version, which is often faster but does not include any debugging options (e.g. you cannot pause execution, or break into it). Note the F5 (with debugging) and CTRL-F5 (without debugging) **shortcuts** to run the program.

If the console window disappears straight away after running: try starting without debugging (CTRL+F5).

1. **Writing a simple program using the .NET library**: Display the current time.
   1. We are going to **write a simple program to display the current time**.
   2. Desired output: . With the current time (obviously ☺!)

You can build this into your exisiting Hello World program. Just add the necessary lines. Use Concole.WriteLine again to display the output.

**You’ll need to use** the system-defined **DateTime** structure. If you type DateTime, then put the mouse inside the word, then press F1 you will bring up online MSDN help. Scroll down and click on DateTime Members to view the properties you can access. This context-sensitive help can be used on methods, properties etc. as well

**TIP**  if this appears:



Select No and click do not show this again. Its is rarely if ever useful to want to run the last successful build if there are bugs in the current one!

1. **Waiting for the Escape key**
   1. Now write a loop at the end of program to force the system to wait until the escape key is pressed. This prevents the software closing before the user is ready.  
      
2. **Change the format of the time string to hours, minutes, seconds, then AM/PM**
   1. Use the ToString method of the DateTime object return by the Now property :

**TIP** You can look up the details of this by pressing F1 with the cursor inside the word ToString..

Example: 

**Exercise 2. C# Forms program example**

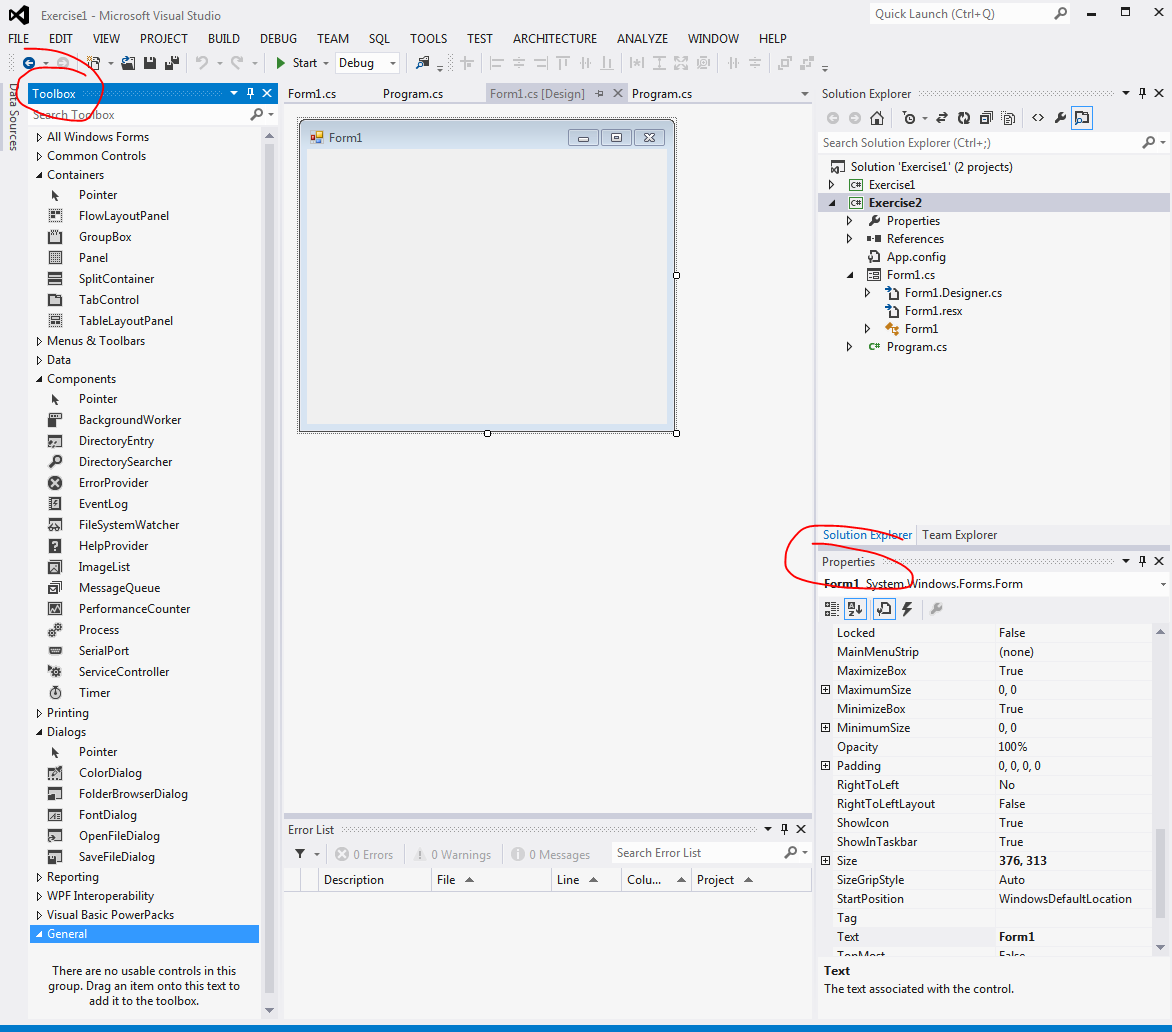
**To use a GUI in c#, a simple place to start in using Forms.**

1. Go to File->Add->New Project. Add a Windows Forms Application to your solution workspace, called Exercise 2. “Adding” it means it is added to the existing solution (the workspace). You can now switch between Exercise 1 and 2 in the Solution Explorer window.

**TIP**: Right-clicking the project name and selecting “Set as Start-up project” lets you choose which project to compile by default

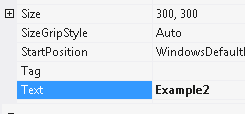
1. Run the program. A default empty window should appear.
2. Close the Form1 program. Click on Form1.cs[Design]. This is the **visual designer** for Windows Forms Graphical User Interfaces.

**TIP**: You can switch between the visual designer and the **code** by right clicking and selecting View Code or View Designer.

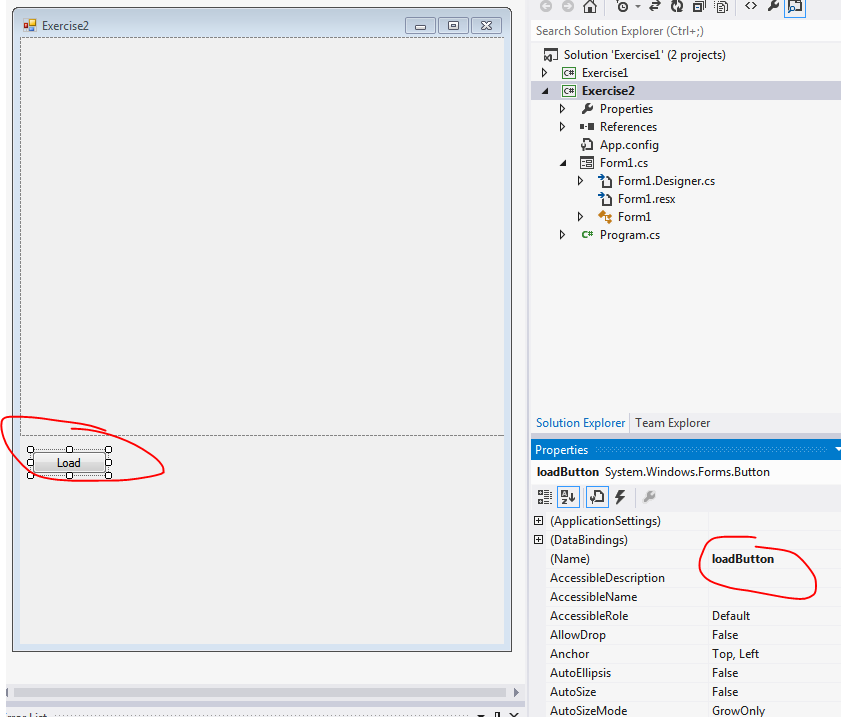


The toolbox (left in image above) lets you add controls to a Form. The Properties box (right in image) lets you set properties, like window names, names of controls, visibility, whether something is enabled/disabled etc. Events are also listed in this window when you press the Lightning icon.

1. Change the name that appears on the title bar of the form to “Example2” by changing the Text property in the Properties box.



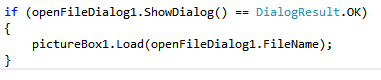
1. Add a **PictureBox** control. This is found under Common Controls in the toolbox and can be added by double-clicking. Resize to fill the top 2/3s of your form.
   1. Note in the properties box (when you click on the picture box control) the ‘pictureBox1’ name appears. This is the name used in the code to refer to this control (the variable name, if you like). You might want to rename the names of controls to something more meaningful.
2. Add a Button in the same way and move below the PictureBox. Name it loadButton and change the text to Load:



1. Double click the Load button. VS will generate a method which will be called when you click the Load button, which you will be taken to. You have added an **event handler.**
2. Go back to the Designer (right click code, then View Designer). Add an **OpenFileDialog** component by double clicking from the toolbox, under Dialogs. There’s no ‘visible’ component to this, but it will add an icon underneath the Form:



1. Add code to the method created when you clicked the Load button: This simple code will let you load a picture file and show it in our PictureBox. Double click the Load button, or go to Form1.cs to get there again. Then type this:



It should be clear what’s happening here. Even if you’ve never seen C# before, this should make sense. Notice as you’re typing that Visual Studio tries to **autocomplete** your code. This is a powerful feature of using C# in visual studio. We will come to use this more, and even get visual studio to generate code for us in future weeks.

1. Return to the designer, and look at the properties for the **OpenFileDialog** control by clicking on it . We want to add some filters to just display image files. In the Filter property, add something like:

JPEG Files (\*.jpg)|\*.jpg|PNG Files (\*.png)|\*.png|BMP Files (\*.bmp)|\*.bmp|All files (\*.\*)|\*.\*

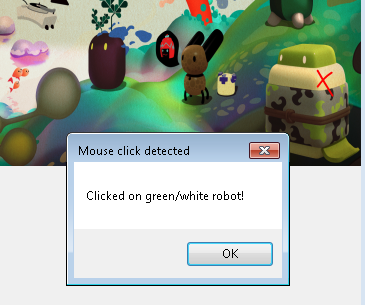
1. If you run the program and click Load you will see what this filter does. Load an image (look in e.g. the wallpaper folder C:\Windows\Web\Wallpaper\Scenes if you don’t have any others to hand). Note it may be stretched/zoomed in a strange way (see bullet points below)

Other things to try

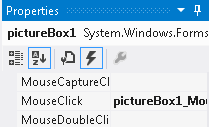
* Try renaming the load image dialog box to “Load an Image…”;
* Change the way an image is displayed so that it scales the image to fit on the space available. Clue: try changing the SizeMode property of the PictureBox.

**Exercise 3 - -a bit harder. Pop up a message when the user clicks on *regions* of the image**

This is often called an “image map”. Set up ‘regions of interest’ on the picture so that when the user clicks on them different messages are displayed, eg.:



Click lightning bolt for events list



Double-click to add event handler

You’ll need to add a MouseClick event handler to the Picturebox (to get mouse info), and check if a click is in a particular area.

To add a mouse handling event, select the PictureBox, then in the properties box press the lighting button



Double click the MouseClick event to add a method which is called when the user clicks the image somewhere.

First try setting up a rectangle area to get mouse clicks, e.g.



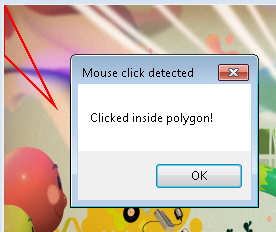
Pop up a message box using something like:

MessageBox.Show("Clicked in a box!", "Mouse click detected");

Advanced:

* You can also define irregular regions using a GraphicsPath and the isVisible method to see if the mouse is inside it….
* You will need to add a ‘using’ statement to the top of the file so C# can find the GraphicsPath method:

using System.Drawing.Drawing2D;

* You can also draw this in the picturebox Paint event if you want to see it:
  + - e.Graphics.DrawPath(new Pen(Color.Red, 2), path);