Visual Studio intro notes, 1-20-2019

**Repos page:**

**-**For each homework and assignment in cis300, you will get a link to create a private github repo accessible by the teacher, you, and the TAs

-To use repositories, clone them over to a local machine

**Solutions page:**

-All code belongs to units called solutions:

-A solution contains projects, which contains files, which contain code.

-A solution can contain any number of projects and files, and projects can belong to multiple solutions

-In CIS 300, each repo will contain one solution with the suffix .sln.

**Excerpt From Solutions for Future Perusal:**

Near the top, just under the search box, is the name of the solution with an indication of how many projects it contains. Listed under the name of the solution is each project, together with the various components of the project. One of the projects is always shown in bold face. The bold face indicates that this project is the startup project; i.e., it is the project that the debugger will attempt to execute whenever it is invoked (for more details, see the section, "[The Debugger](http://people.cs.ksu.edu/~rhowell/DataStructures/redirect/debugger)").

The solution in the above example is the initial solution created when a new Windows Forms Application is opened. The project components having a suffix of ".cs" are C# source code files. When a new Windows Forms Application is created, its project will contain the following three source code files:

* **Form1.cs:** This file contains code that you will write in order to implement the main GUI for the application. It will be discussed in more detail in "[The Code Window](http://people.cs.ksu.edu/~rhowell/DataStructures/redirect/code-window)".
* **Form1.Designer.cs:** You will need to click the triangle to the left of "Form1.cs" in the Solution Explorer in order to reveal this file name. This contains automatically-generated code that completes the definition of the main GUI. You will build this code indirectly by laying out the graphical components of the GUI in the design window (see the section, "[The Design Window](http://people.cs.ksu.edu/~rhowell/DataStructures/redirect/design-window)" for more details). Ordinarily, you will not need to look at the contents of this file.
* **Program.cs:** This file will contain something like the following:
* using System;
* using System.Collections.Generic;
* using System.Linq;
* using System.Threading.Tasks;
* using System.Windows.Forms;
* namespace Ksu.Cis300.HelloWorld
* {
* 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());
* }
* }
* }

The **Main** method is where the application code begins. The last line of this method constructs a new instance of the class that implements the GUI. The call to **[Application.Run](http://msdn.microsoft.com/en-us/library/ms157902(v=vs.110).aspx)** displays the GUI and starts a loop that processes events such as mouse clicks and keystrokes. Ordinarily, there is no need to look at this code.

One of the first things you will need to do when starting a new Windows Forms Application is to change the name of Form1.cs, as this name (without the ".cs" suffix) is also the name of the class implementing the GUI. Therefore, it will need to be changed in order to conform to the [naming convention](http://people.cs.ksu.edu/~rhowell/DataStructures/redirect/naming) for classes. To do this, right-click on its name in the Solution Explorer, and select "Rename" from the resulting popup menu. You will then be able to edit the name in the Solution Explorer. If you are working through the example begun in the previous section, rename this file to "UserInterface.cs".

**Detour: Naming Conventions:**

-Pascal Case: multiple words joined without spaces, capital letters at the beginning of each word

-Camel Case: First letter isn’t capitalized

-Namespaces: use periods between words, capitalize each word in pascal case. Name the program after its function, not anything like “homework1”. Suitable examples include “Driver”, “ErrorHandler”, “ThingManager”, etc.

-Interfaces: Put a capital i before the first letter, which should also be capitalized.

-Properties: Use Pascal Case, no abbreviations. For example, “HasIntegers”, not “hasInts”.

-Control On Forms: Use camel case, begin names with “ux” followed by a capital letter.

-Public constants: Be descriptive, use Camel Case.

-Private fields: Use camel case with an underscore preceding the first letter, “\_weight” for example

**The Design Window:**

**-**Used to simulate and build graphical components.

-Can be manipulated the way a window is manipulated.

-To add controls, use the toolbox (a tab on left edge). Controls can be manipulated the same way images or text would be manipulated in, say, Microsoft publisher

-To make an item react to stimuli, go into the properties for the object, and add an event handler script to a given event. Event handlers are scripts that perform a certain action defined by which event the handler is assigned to. Event handlers can also be removed through the properties menu.

**The Code Window:**

-a partial class has some of its definitions in userinterface.design.cs

-auto-complete is a feature; use it by pressing “enter” when an autocomplete suggestion comes up

-the code window also gives you a parameter information box when it detects you are in a place that takes parameters

-when you rename a parameter or method, you get the option to change all instances of it to the new name

**The Debugger:**

**-**the debugger attempts to save, compile, and run unsaved/compiled code.

-If everything works correctly, the app is done

-When the debugger fails to complete, it will return messages about what errors it encounters.

-When asked if you would like to run the last successful build, answer ‘no’. this will cause the compiler to list out all the errors it encountered, as well as the values of any variables that were in use at the time.

-When debugging, it is useful to pause at certain points in order to examine the variable values. To do this, set a breakpoint by clicking the left margin of the code where we want to pause execution.

-Breakpoints can also be set conditionally

**Submitting Assignments**

-Manually refresh changes tracked by git (circular arrow icon)

-Commit your changes to the local repository. (pencil icon)

-Push changes to github by clicking the up arrow at the bottom of the window

-Submit the URL of the commit you want graded (for lab assignments) using the assignment form provided in the instructions

**DO ALL THESE STEPS IN ORDER!!!**