CSCI 1301 - Lab 02

Clément Aubert

January 29, 2018

Deadlines: No deadline for this lab, but it would be preferable to finish it all before lab #3.

Dependencies: No part of the lab can be achieved without achieving the previous parts first.

1 Part I - Structure of a Project

Do one of the following:

a.
- Navigate to the the folder for this class you created during lab 01.
- Create a folder for this second lab, let's name it `02_lab`.
- In the examples from the textbook you downloaded and extracted during lab 01, find the folder `ch03\fig03_01`.

or

b.
- Create a folder for this class, and a folder named `02_lab` in it.
- Download `Welcome1.zip` and extract it.

- Copy the `Welcome1` folder located in it to `02_lab`. We will work with this copy instead of with the original files from this point.
- Enter the `Welcome1` folder in `02_lab` and examine its structure and content. We can draw its structure as a tree:

```
└──Welcome1
    │   Welcome1.sln
    │
    └──Welcome1
        │   App.config
        │   Welcome1.cs
        │   Welcome1.csproj
        │
        ├──bin
        │   ├──Debug
        │   │       Welcome1.exe
        │   │       Welcome1.exe.config
```

```
|   |       Welcome1.pdb
|   |       Welcome1.vshost.exe
|   |       Welcome1.vshost.exe.config
|   |       Welcome1.vshost.exe.manifest
|   |
|   └───Release
└───Properties
        AssemblyInfo.cs
```

1. How many different *extensions* are they?
2. Using a website like https://fileinfo.com/, give a brief description of those extensions.

2 Part II - Finding Answer

All the documentation for Visual Studio is at https://docs.microsoft.com/en-us/visualstudio/. The documentation for C# is at https://docs.microsoft.com/en-us/dotnet/csharp/. To get started, have a look at https://docs.microsoft.com/en-us/visualstudio/ide/creating-solutions-and-projects, and answer the following:

1. What is a solution?
2. What is a project?
3. Which one contain the other: the solution, or the project?
4. How are the templates organized?

Before starting the next part, have a look at https://docs.microsoft.com/en-us/visualstudio/ide/compiling-and-building-in-visual-studio.

3 Part III - Compiling and Executing Your First Program

a. Go to `02_lab/Welcome1` and open `Welcome1.sln` with Visual Studio. Discard the security warning.
b. In the Solution Explorer, to the right, expand all the items that can be expanded by clicking on the ▷ symbol.
c. Answer the following:

- What is the name of the solution?
- What is the name of the project?
- Describe the connections between what is shown in the Solution Explorer and the content of the directory you explored in part I. Can you find items in the Solution Explorer that do not correspond to a file? Files that are not present in the Solution Explorer?

d. In the Solution Explorer, double-click on `Welcome1.cs`. This is the *source code* of the application you are actually considering.
e. Let's compile this program, using `Build` ⟶ `Build solution`. What happened?
f. Let's run this program, using `Debug` ⟶ `Start without Debugging`. What happened?

You will **extensively** compile and run programs in this class. Instead of having to click twice, I highly recommend that you start now memorizing shortcuts:

- Use Ctrl + Shift + B to build the solution,
- Use Ctrl + F5 to start the program without debugging.

With Alt + F4, that makes 3 shortcuts already!

4 Part IV - Breaking and Fixing

We will now try to break (i.e., make the compilation impossible) the program in various ways, and revert the changes that actually broke the program.

For each of the modification indicated below, do the following:

a. Modify `Welcome1.cs` according to the instructions.
b. Compile the program.

- If compilation fails, an error will be reported in the lower part of VS. It will end with something like

  ========== Build: 0 succeeded, 1 failed, 0 up-to-date, 0 skipped ==========

Observe the error message, and infer what broke your program. Revert the change.

- If the compilation suceed, you should see a message that ends with

  ========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========

Run the program, and infer why was this change innocuous to your program.

Here are the modifications:

- Remove the first line
- Remove the semicolon after `using Sytem`
- Replace `Welcome1` with `WelProj`
- Remove the two `//` before `Main method begins execution of C# app`
- Replace `"Welcome to C# Programming!"` with `"This is my program!"`
- Remove the brace (or "curly bracket", i.e., the `}` symbol) before `// end Main`
- Add three new lines at the end of the file
- Replace `Console.WriteLine` with `CONSOLE.WriteLine`
- Replace `Console.WriteLine` with `Console.WRITELINE`
- Add a new line between `Console.` and `WriteLine`
- Add a new line between `WriteLine` and `(`
- Add a new line between `Write` and `Line`
- Replace `Main()` with `Method()`
- Remove the indentation (i.e., the space between the beginning of the line and the first character of the instruction) on all lines

5 Part V - Renaming Projects, Solutions, and Files

a. Lets rename the solution and project *within VS*:

- In the Solution Explorer, right-click on `Welcome1.cs` and select "Rename". Rename the file to `WelProj.cs`.
- Right-click on `Solution 'Welcome1' (1 project)`, select "Rename", rename it to `WelSol`.
- What changes do you notice? Can you still build and debug your program?

b. Now, exit Visual Studio, and make a copy of your `Welcome1` folder, that you name `WelSol`. Open `WelSol/WelSol.sln` in VS: is everything all right? Exit VS.

c. Inside the `WelSol` folder, rename the `Welcome1` folder to `WelProj`. Open the project in VS. Is everything ok? Not really. Exit VS, and rename `WelProj` to `Welcome1`. Make sure you can open and build that project correctly anew before continuing.

d. In the `Welcome1` folder, rename `Welcome1.csproj` to `Welproj.csproj`. Open the project in VS: is everything all right? Not really. Exit VS, and revert this change.

What can you conclude on renaming projects, solutions, folders and files, inside and outside of VS?

6 Part (Optional) - Pushing Further

The following are two independent tasks, to widen your understanding of this class, and to prepare you for the next labs, respectively.

a. To get a better insight on the variety of programming languages, have a look at http://cs.lmu.edu/~ray/notes/pltypes/

b. To get ready for the coming lab, do the following. Taking inspiration from https://docs.microsoft.com/en-us/visualstudio/debugger/debugger-feature-tour, set a breakpoint in your program and start the debugger. Examine the execution flow of your program, and advance the debugger, using F10.