Hello World Lab

https://csci-1301.github.io/about#authors

May 19, 2021 (11:48:42 PM)

Contents

1 Your First Program

1.1 Opening Your First Program

1. Download HelloWorld.zip from the same place where you downloaded this instructions file and save it on your computer.
2. Unzip the program file.
   - on Windows, right click and choose "Extract all"
   - on macOS: double-click the .zip file
   - on Linux: right click and choose "Extract" or "Open with Ark"[1]
3. Go in the HelloWorld folder that was created.
4. Double-click on `HelloWorld.sln` file
5. Clicking the `.sln` file should launch your default C# IDE that you installed previously.
   - If you get prompted for which application to use, choose the IDE you installed previously.
   - If the IDE does not launch or launches in a different program, right click on the `.sln` file and look for an option to `open with`, then select to open it with the IDE you installed previously.

[1] https://www.wikihow.com/Unzip-Files-in-Linux

1.2 Compiling and Executing Your First Program

1. Within the IDE, first locate `Program.cs`.

   This file will be visible in panel called "Solution Explorer" or "Explorer", depending on the IDE. If you do not see such panel right away, explore the IDE menus to find and open it.

2. After you have located `Program.cs` double click on it. This is the *source code* of the application you are actually considering.

3. Let's compile this program. Look for an option to `Build solution` and click on it. What happened?

4. Let's run this program. Look for a menu option `Build` > `Start without Debugging` or `Run` > `Run Project` and click on it. What happened?

5. 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. You should study your IDE to see the exact shortcuts for your IDE for compiling and running a program. Here are the *usual* shortcuts for different operating systems:

   Windows/Linux

   - Build solution: Ctrl + Shift + B
   - Run: Ctrl + F5

   Mac

   - Build solution: Shift + Command + B
   - Run: F5

With Alt + F4 (to exit any program), that makes 3 shortcuts already! You can find a complete list at http://visualstudioshortcuts.com/ and on this page[2] of the Microsoft docs. I will try to introduce more useful shortcuts as we progress.

1.3 Backups

Now we need to make sure you know how to save your work and access it. This is especially important if you are using the computer lab rooms, as **you can not store files permanently on the lab's computer, you will have to store them either online in your cloud storage or on a USB drive**.

1.3.1 Finding The Right Tool

You can save your project:

- On your hard drive, if you are using your own computer.
- On an external/removable data storage: USB flash drive, external hard disk drive, or any kind of USB mass storage device.
- On a server: the University has a partnership[3] with box.com[4], and you can follow this tutorial[5] to get started, but any service (Google Drive[6], Dropbox[7], OneDrive[8], etc.) would do.

[2] https://docs.microsoft.com/en-us/visualstudio/ide/default-keyboard-shortcuts-in-visual-studio?view=vs-2019
[3] https://www.augusta.edu/its/box/
[4] https://box.com/
[5] https://www.augusta.edu/its/box/quickstart.php
[6] https://www.google.com/drive/
[7] https://www.dropbox.com/
[8] https://onedrive.live.com/

Having *two* backups is generally recommended.

If you chose the "virtual" option (i.e., using a server) and you are in a computer lab, **do not** try to install a synchronization program (like Google Drive and Sync[9], Box's app[10], etc.) on the lab computer: it will likely not work, due to University rules[11]. Instead, create the structure/project/files on the computer during the lab and upload them (using the web-interface) at the end of the lab. Make sure to always upload your files before logging out of the computer.

1.4 Making Sure You Have the Right Files

After selecting where you want to store your backup files

1. Create a folder for this class (`CSCI1301`)
2. Create subfolder for the `HelloWorld` lab. Put all of the files related to the "HelloWorld" solution in this folder.
3. Explore your backup. Check that you have multiple folders. Specifically check that your backup contains at least the following files:

   - `HelloWorld.sln` - this is called a solution file. It tells the IDE how to load your C# source code in the IDE.
   - `HelloWorld.csproj` - this a project file. Every C# solution contains at least 1 project. Dividing code into multiple project is useful for very large programs and allows integrating projects written in different languages under the same solution.
   - `Program.cs` - this is the actual source code of your program and where you write code.

   It is useful to understand the purpose and role of these different files (`.sln`, `.csproj`, `.cs`) when you want to backup or share (i.e. turn in) C# programs.

1.5 How Was the Backup?

Once you are done, test that you performed the backup properly.

1. Re-download or transfer the files you just saved (the whole `HelloWorld` folder) on the computer
2. make sure you can still open the project in your IDE
3. Do you remember…
   1. how to build the solution
   2. start the program without debugging?
   3. Use the shortcuts?

If not, go have another look back at the "Compiling and Executing Your First Program" section.

If your backup went wrong (you can't open the project, it won't compile, …), try to understand what happened. Then, re-download the `HelloWorld.zip`, unzip it, and make sure you can build the solution and run the program.

[9] https://www.google.com/drive/download/
[10] https://app.box.com/services/browse/official
[11] https://augusta.policytech.com/dotNet/documents/?docid=5702

1.6 Orientation

IDEs have many features and require practice. Explore your IDE and try to complete the following tasks:

1. If you currently have a solution open in the IDE, close that solution
2. Use the IDE file menu to locate and re-open HelloWorld program
3. Build the program
4. Try to find `Clean solution`. Cleaning is the action of removing all generated files.
5. Close `Solution Explorer` or `Explorer` and make sure you can re-open it
6. Try to change the font size of your editor window.

1.7 Breaking Your Program

If you followed the instructions carefully, your were able to build the solution and start the program without debugging after each step. As you know, C# has precise rules and not respecting them can prevent your solution from being built by the IDE.

In this exercise, you are asked to do the following:

1. Change the program so that it violates one of the syntax rules of `C#`.
2. Build the solution and note that an error is reported. The IDE will report a build error similar to the following:

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

   If you do not see an error, look in different tabs. If you still cannot see an error, open build output view from the IDE menu. It is usually labelled as `View > Output`, `View > Tool Windows > Build`, or similar. Then retry building the program to see the error.
3. Make sure you understand the meaning of the error message.
4. Undo your change by pressing

   - Windows, Linux: Ctrl + z
   - MacOS: Cmd + z
5. Make sure you can build the solution without a new error message.
6. Break your program three times, in order to identify three different error messages, and three ways of breaking C#'s rules.

If you have time or need ideas, you can try with the following, and see which one(s) make the building impossible (do not forget to undo your change after):

- Remove the semicolon after `using System`
- Replace `class Program` with `class TestOne`
- Remove the brace (or "curly bracket", i.e., the `}` symbol) at the last line.
- 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