HelloWorld and HelloUser Programs
Getting to know the IDE and writing your first program!

Start Microsoft Visual C# 2010 Express

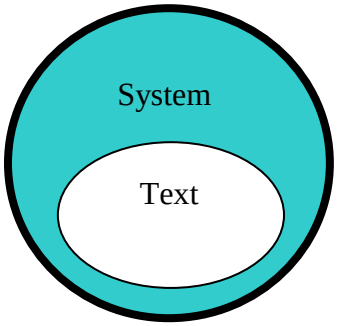
Create a new project

Project types: Visual C# > Console Application

Name: HelloWorld (no spaces)

Notice:

- There is already a bunch of code there…
  - The IDE has written a bunch of code for you.
  - This is code that will always be needed and the IDE just does it for you.
  - Often called "boilerplate" code
- 1st line: "using System;"
  - What is "System"?
    - It is a ***namespace***
    - Remember that namespace is the outermost layer of the .NET onion
- We will ignore the 2nd & 3rd lines for now…
- 4th line: "using System.Text;"
  - It is a namespace inside a namespace



- 6th line: "namespace HelloWorld"
  - It is the namespace for our program … it is our little playground. ☺
  - The IDE has created it for us
- 7th line: "{"
  - There is a left curly brace immediately below the namespace declaration
  - Also notice that on the last line is a right-curly-brace
  - Place the cursor just to the right of the right-curly-brace on the last line

- Did you notice that it and the left-curly-brace on line 7 were highlighted?
  - This is showing you that those two braces go together…
    - Together they "wrap" everything in our namespace
  - Notice that everything between them is indented…
    - This is helping us see what code is part of the namespace
- Try this with other sets of curly-braces to see what code they "wrap"
- Every set of code that needs to be considered a unique ***block*** is wrapped with curly-braces and indented

- 8th line: "class Program"
  - Remember the onion layers of .NET?
  - The layer inside the namespace layer is ***class***.
  - Another name for class is ***object***…a thing that can do something
  - This class is our program
  - You can rename this class if you like
  - Rename it to "MyFirstProgram"
- 10th line: "static void Main(string[] args)"
  - For now ignore "static" and "void"
  - Also for now, ignore what is in the parentheses: "string[] args"
  - "Main" is a ***method*** in our class
    - The layer inside the class layer is method
    - A method does something for our class
    - ***It contains code***
    - "Main" is the name our class' method
    - Our class only has one method
    - This is where all our code for this program will go

Let's write some code!

- Create a new line inside the curly-braces of our Main method.
  - Notice how the cursor auto-magically indented from the curly-braces?
- Type the capital letter C
  - Notice how a window pops up with a bunch of options in it?

- These are the possible commands that start with C.

- Type the lower case letter o
  - Notice how the options in the pop-up window changes?
  - Also notice the third option down is Console?
  - Double-click on it…the IDE fills it in for you
- Wait a second … notice there are now red up tic marks? ^^
  - These are telling you that you need to type more to complete the command
- Type a period
  - Notice the pop-up window appeared again?
  - It is giving you the options for subcommands of the Console object
- Type the capital letter W
- Double-click on WriteLine
  - You can also just hit Enter if it is selected
  - The IDE fills in the next part of the command for you
  - Again notice the ^^… there is more we need to type
- Type a left parenthesis
  - Another pop-up window appears … it says

    1 of 19 void **Console.WriteLine()**
    Writes the current line terminator to the standard output stream
  - This is telling us there are 19 different ways we can use the WriteLine method
  - If you click the down arrow after the "19" you can cycle through the options
  - The one we want to use is actually the 11th … the one that takes a string
  - You don't have to scroll through them unless you need a reminder which one to use
- Type the following including the double-quotes, right-parenthesis and semi-colon:

  "Hello World!");
- Congratulations! You have now written your first program!

It is now time to compile and run your program.

- Find the green "arrow" icon button at the top of the IDE.
  - If you hold your cursor over it, it says:

Start Debugging (F5)

- When you click this button (or hit the F5 button) the IDE will compile **and** run your program
- If you have made a typo, the compiler will likely detect it and tell you about it. You will have to correct all typos before it can run the program.

- Hit the Start Debugging button
  - What happened?
  - Assuming you had no compile errors, all you saw was the flash of a black window which immediately disappeared and returned you to the IDE
  - Did anything really happen? What was that black window?
  - It was your program running!
  - It ran:
    1. created the console window
    2. made it appear
    3. sent the message "Hello World!" to it
    4. and finished
- Hmm, that isn't quite what we wanted is it?
  - What we'd really like is for the console window to stay around until we've been able to read it.
  - Let's add something to our program.
- Place the cursor at the end of the Console.WriteLine line and hit enter
- Type the following:

  Console.ReadLine();
- Now compile and run the program again (click the Start Debugging button)
  - Whoohoo! It worked! It is just sitting there so we can read it.
  - Now hit Enter and our console window will go away
  - In other words our program completes/exits and returns us to the IDE
- The ReadLine method of the Console object collects all keyboard input until the Enter key is hit.

HelloWorld and HelloUser Programs
Getting to know the IDE and writing your first program!

The next step

- Now that we know how to collect keyboard input, let's put it to use.
- I'm going to give you a hint then turn you loose to see if you can figure the rest out.
- Keyboard input is letters, numbers, punctuation, etc.
- Individually, each letter, number, etc. is considered a "character" and can be stored in a "char" variable. One character in a char variable at a time.
- Multiple characters (a sentence or name for instance) are considered a "string" and are stored/saved in a "string" variable.
- To create a variable, you tell the compiler the type of information the variable will hold and give it a name.
- So if we are going to ask the user for their name we could create a variable to store/save it like this:

```
string userName;
```

First task

- Using what you have and what I just told you about creating a string variable, create a new project named HelloUser that asks the user for their name and then tells them hello using their name.

Second task

- What are some boundary conditions this program has?
- Is your program sensitive to them?
- What can you do to help it deal with those boundary conditions?

Third task

- Play with your HelloUser program to make it more interesting:
    - Are there fun things you can do with the output?
    - Are there other questions you want to ask the user?
    - Access the .NET documentation about the System.Console class
        - Mouse-click so that the cursor in the middle of the word "Console" in your code.

HelloWorld and HelloUser Programs
Getting to know the IDE and writing your first program!

- Hit the ***F1*** button … this is the "Help" button.
- IE will appear and navigate to the Microsoft .NET help page for the System.Console class.
- It is showing all the capabilities of the System.Console class.
- Explore what class members are available to you
- Can you "spruce-up" your console window?
- Can you change its behavior in a fun way?