This course, C# Fundamentals, uses a version of Visual Studio as an instructive tool.

In order for me to follow and mimic what the instructor is doing, I built a C# solution/project. I chose “Console Application - .NET Framework”. I used “GradeBook” as the name of the solution and project. I encountered two problems.

Problem 1

I could not see what was written as a consequence of statements such as

Console.WriteLine(“Hello!”);

Either (1) use the Visual Studio pull-down menu

**Project > GradeBook Properties…**

or (2) right-click the project name (**GradeBook**) in the Solution Explorer, and then select **Properties** in the pop-up menu.

In Response Visual Studio displays a dialog with an array of tabs in a column on the left edge – “Application”, “Build”, . . . “Code Analysis”. Select the “Application” tab. The **Output type:** drop-down list displays in the top center of the dialog. Using the down arrow change the setting from “Console Application” to “Windows Application”.

As a consequence text produced by Console.WriteLine() will begin appearing in the **Output** window under the **Output** tab when **Show output from:** is set to “Debug”.

Problem 2

During debugging I want to provide values to the command-line arguments array – which conventionally uses the name args[] (e.g. static void Main(string[], args).

Follow the 1st steps under Problem 1 to bring up the properties dialog for project **GradeBook**. Then select the “Debug” tab. The **Command line arguments:** box displays near the center of the dialog. Type the desired arguments, separated by spaces.

Shortcuts

When you want to create a new function and use it,

* Write the function call into your code, e.g.

var book1 = GetBook(“Book 1”);

* But the function GetBook() is not yet defined.
* Set the mouse cursor on GetBook(), and type <ctrl>. Visual studio offers to create the skeleton of a new function GetBook() into your code; accept the offer.

To get information about a type (built-in, from a system-defined class, or user-defined Class) . . .

Set the mouse cursor on the name of the type, and type the F12 function key. Visual Studio displays a pop-up dialog containing metadata about the type; for a user-defined class it displays the c# code of the class.

Unit Tests

When the instructor started setting up for doing unit tests, I tried to mimic by setting up a unit-test project in the GradeBook solution.

* Right-click **Solution (Gradebook)**. Visual Studio responds by displaying a pop-up menu.
* Select **Add > NewProject…**  Visual Studio responds by displaying a pop-up dialog **Add a New Project**.
* Type “unit test” into the search box at the top of the window. Visual Studio responds by expanding the list of templates on the right half of the dialog.
* Select **C# Unit Test Project (.NET Framework);** then click **Next**. Visual Studio responds by displaying a dialog **Configure your new project**
* I typed “Gradebook.Tests” into the **Project Name** text box. Then click **Create**. Visual Studio added Gradebook.Tests to the solution.

The type of Unit test that the instructor prefers is called xUnit. But xUnit is not installed by default in a Visual Studio test project. As a remedy . . .

* Right-click the project **Gradebook.Tests**. In response Visual Studio displays a pop-up menu.
* Click **Manage NuGet Packages…** from the pop-up menu. In response Visual studio displays a dialog **NuGet Package Manager: Gradebook Tests**.
* Type “xunit” in the Search box. In response Visual Studio displays test projects that pertain to xunit testing,.
* Select the package with the name “xunit”. Then click the **Install** button in the upper-right corner of the dialog. In response Visual Studio installs **xunit**, and it adds several packages to the list of References.Analyzeers.