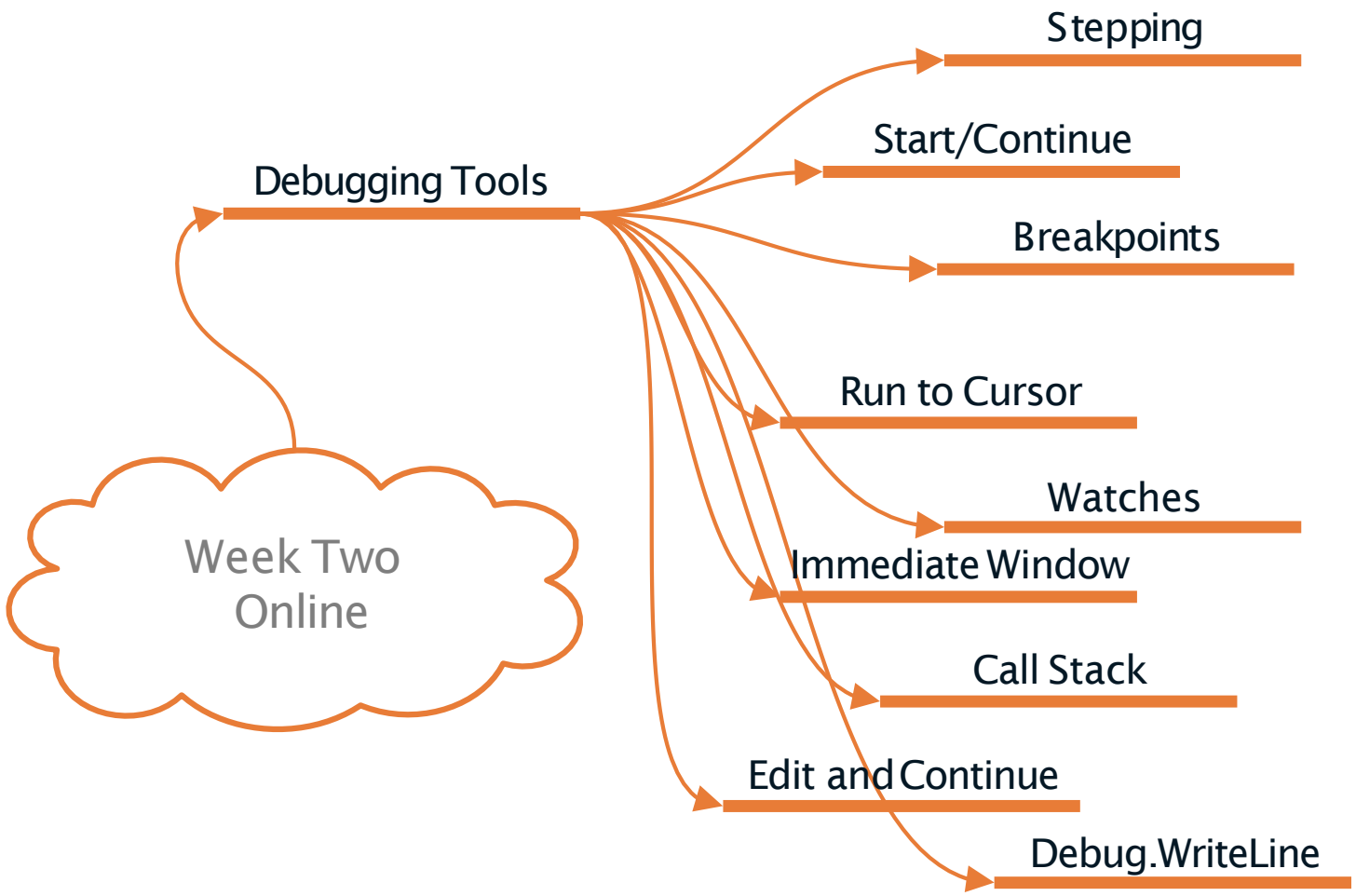
COMP 3602

C# Application Development

Week Two - Online

This Week's Learning Outcomes



Debugging Introduction

Debugging is one of the most important skill a developer has

Debugging is the process of inspecting your code *as it is running* in order to see what is happening - line by line.

Debugging allows you to not only more easily figure out problems and issues with your code, but also allows you to "workshop" code and figure out solutions much more easily than compiling and running the program after each change.

This presentation introduces some of the main tools available for debugging in Visual Studio.

Debugging Tools - Stepping

Stepping is the act of executing a single line of code at a time. It gives you a chance to see every variable's value and think about the code logic before proceeding to the next line. It's invaluable for finding logic errors.

```
64          Console.WriteLine("{0}\n{1}", title, new string('-', 40));
65          if (menuItems != null) ≤1ms elapsed
66          {
67              for (int count = 0; count < menuItems.Length && menuItems[count] != null; ++count)
68              {
69                  item = menuItems[count];
70                  Console.WriteLine("{0, 3} - {1}", count + 1, item.DisplayName);
71              }
72          }
```

```
0 references
10  class Program
11  {
12      // Let's get this baby off the ground
        0 references
13      static void Main(string[] args)
14      {
15          Menu menu = new MainMenu();
16          menu.DoMenu(false);
17      }
18  }
```

You can start debugging from the first line of your program by doing a Step Over (F10) or Step Into (F11).

Program is currently in Break mode (paused at the yellow line). The yellow line will execute when you step or continue.

Step Over: F10
Executes a single line all at once, even when it contains method calls. Use this when you know for sure that the methods called on this line do not contain the bug.

Step Into: F11
Executes a single line, but steps into any method calls on that line.

Debugging Tools – Start/Continue

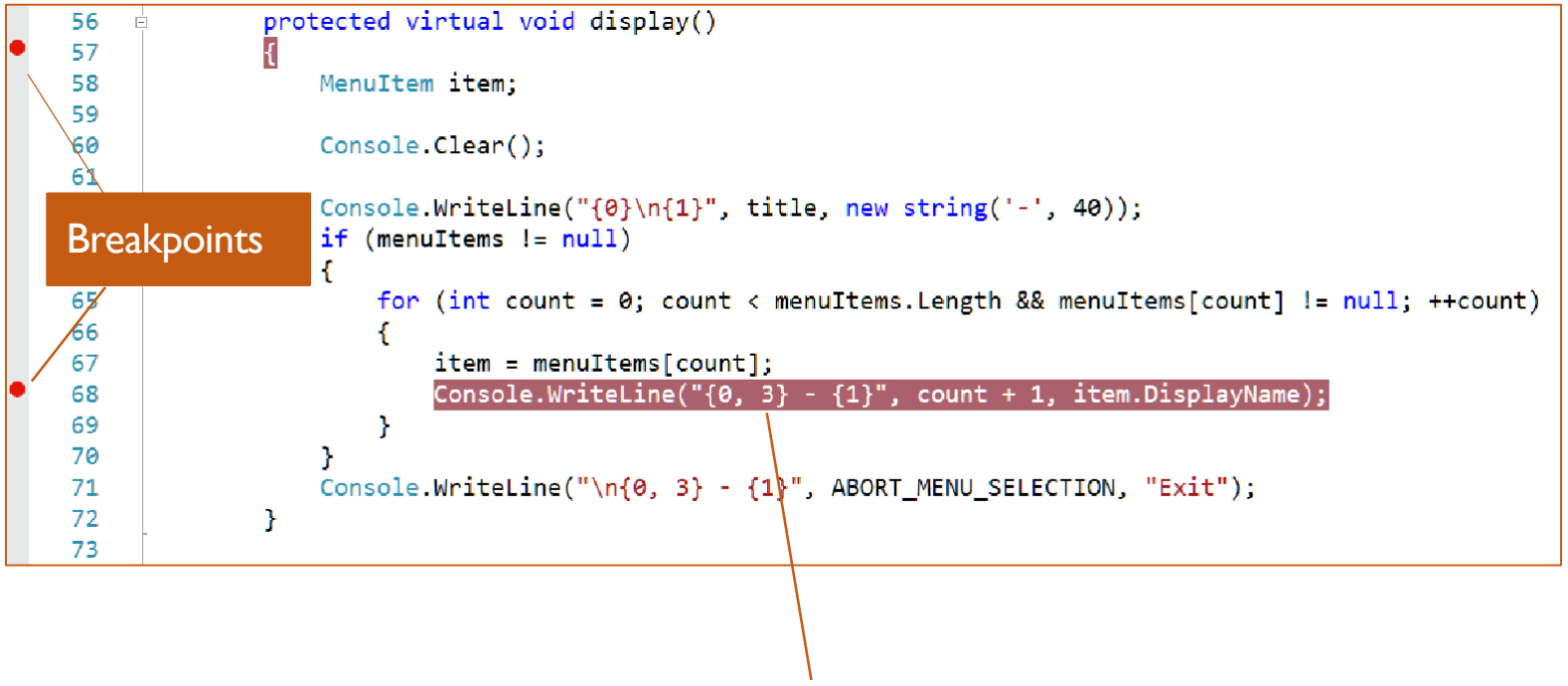
Start/Continue: F5

Starts program execution in the debugger if currently in Design mode.

Resumes program execution if currently in Break mode (program is paused at the yellow line).

Note: You can also start a program by stepping, which pauses on the first line of Main. **(F10)**

Debugging Tools - Breakpoints



Toggle Breakpoint: F9

Turns a breakpoint on or off at the line with the cursor.

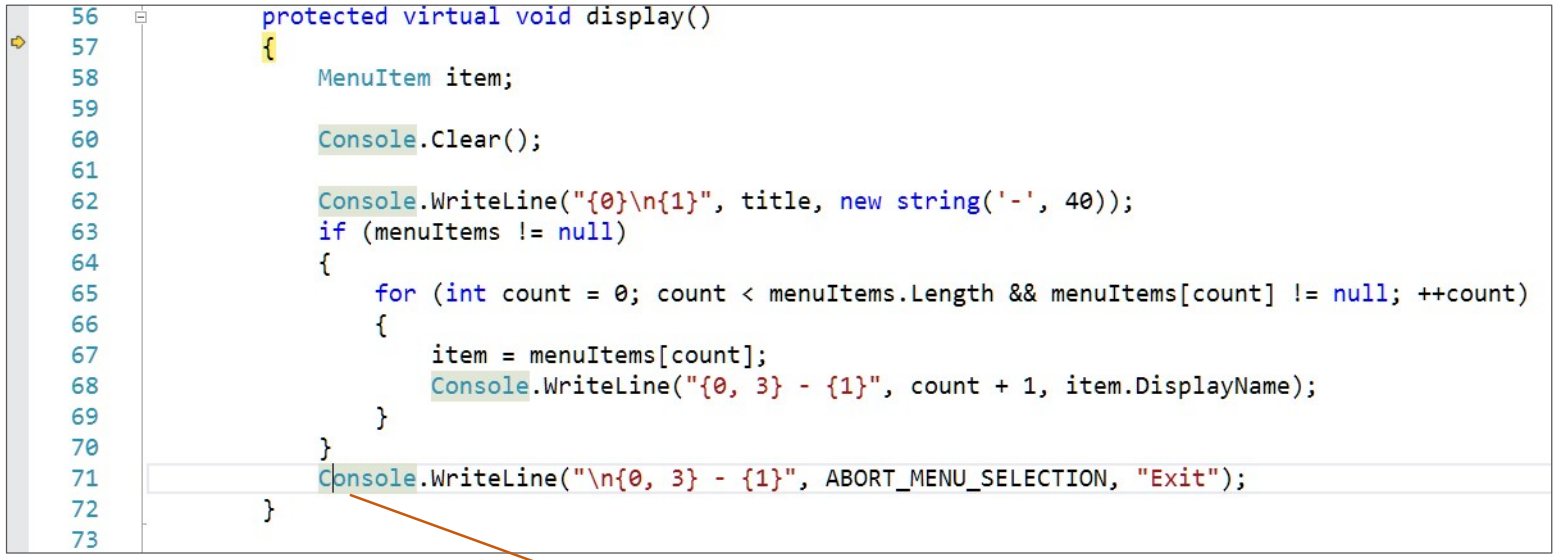
Click in the grey column to toggle breakpoints with the mouse.

A breakpoint is a "flag" that you can turn on, on almost any line of code.

Instead of stepping over every line of the program, you can run it at normal speed, and every time it encounters a line with a breakpoint, it pauses.

When the program pauses, you can: examine variables, step, continue at normal speed until the next breakpoint, etc.

Debugging Tools – Run to Cursor



```
56          protected virtual void display()
57          {
58              MenuItem item;
59
60              Console.Clear();
61
62              Console.WriteLine("{0}\n{1}", title, new string('-', 40));
63              if (menuItems != null)
64              {
65                  for (int count = 0; count < menuItems.Length && menuItems[count] != null; ++count)
66                  {
67                      item = menuItems[count];
68                      Console.WriteLine("{0, 3} - {1}", count + 1, item.DisplayName);
69                  }
70              }
71              Console.WriteLine("\n{0, 3} - {1}", ABORT_MENU_SELECTION, "Exit");
72          }
73
```

Run to Cursor: Ctrl+F10

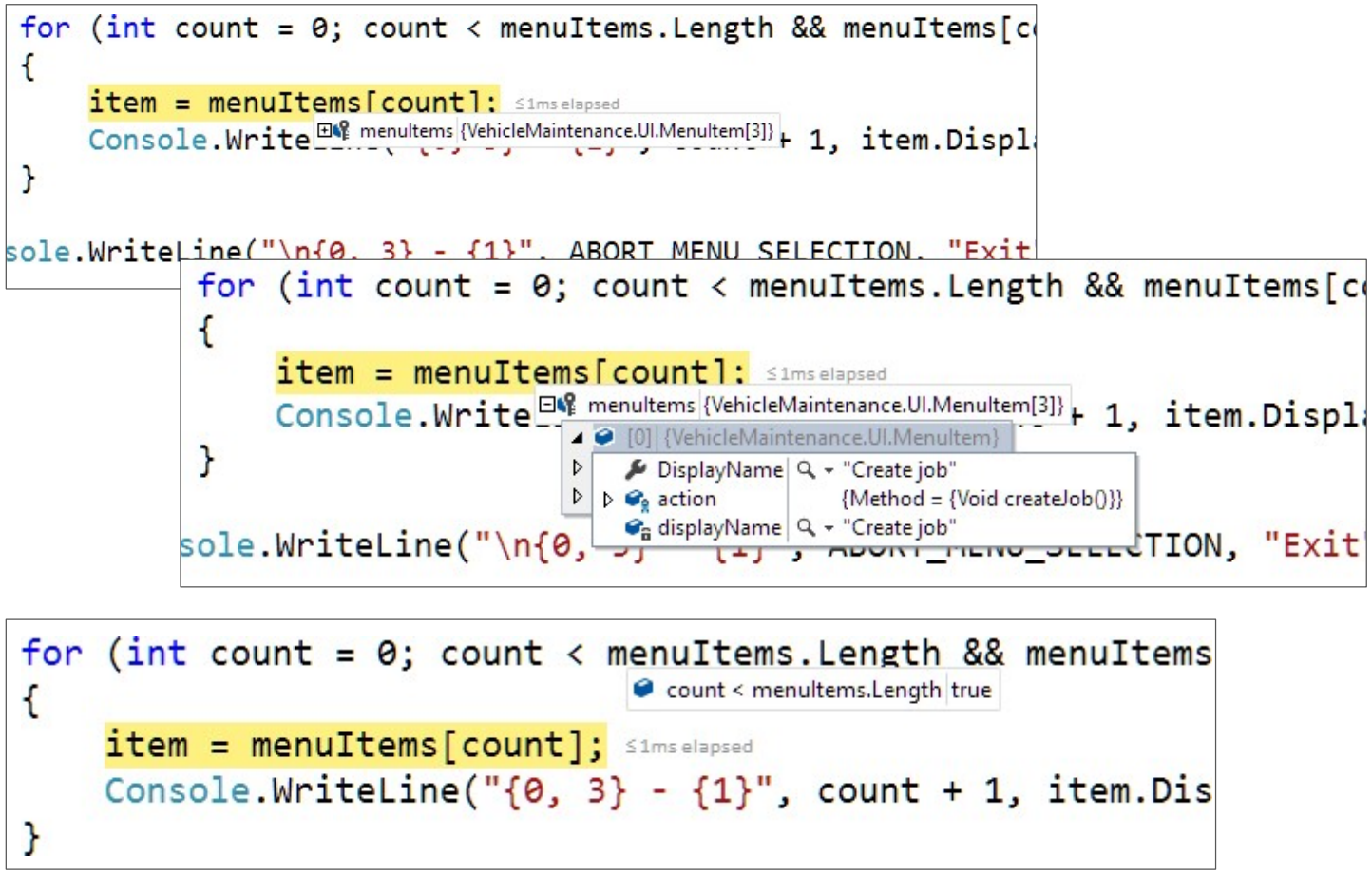
Place your cursor on the desired line and press Ctrl+F10, or right click the desired line and use the right-click popup menu

Run to Cursor is like turning on a temporary breakpoint, then running at normal speed, then turning off the breakpoint when the program encounters it.

It is great for when you've stepped into a big long loop and you want to finish the loop and pause afterwards.

Also see Step Out (F11) to finish executing a method that you didn't mean to step into.

Debugging Tools - Watches



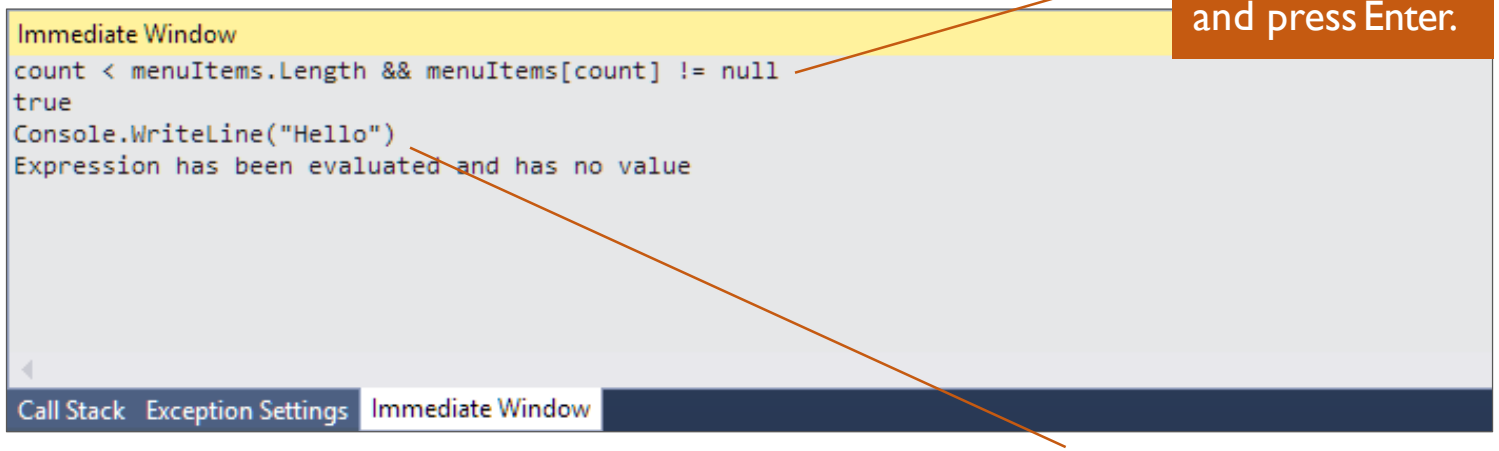
A watch is a display that shows you the value of a variable or expression.

Watches are available only when the program is running in the debugger.

Watches show a value only when the variable/expression is within scope.

Debugging Tools – Immediate Window

```
for (int count = 0; count < menuItems.Length && menuItems[count] != null; ++count)
{
    item = menuItems[count];
    Console.WriteLine("{0, 3} - {1}", count + 1, item.DisplayName);
}
```



Type (or paste) any expression and press Enter.

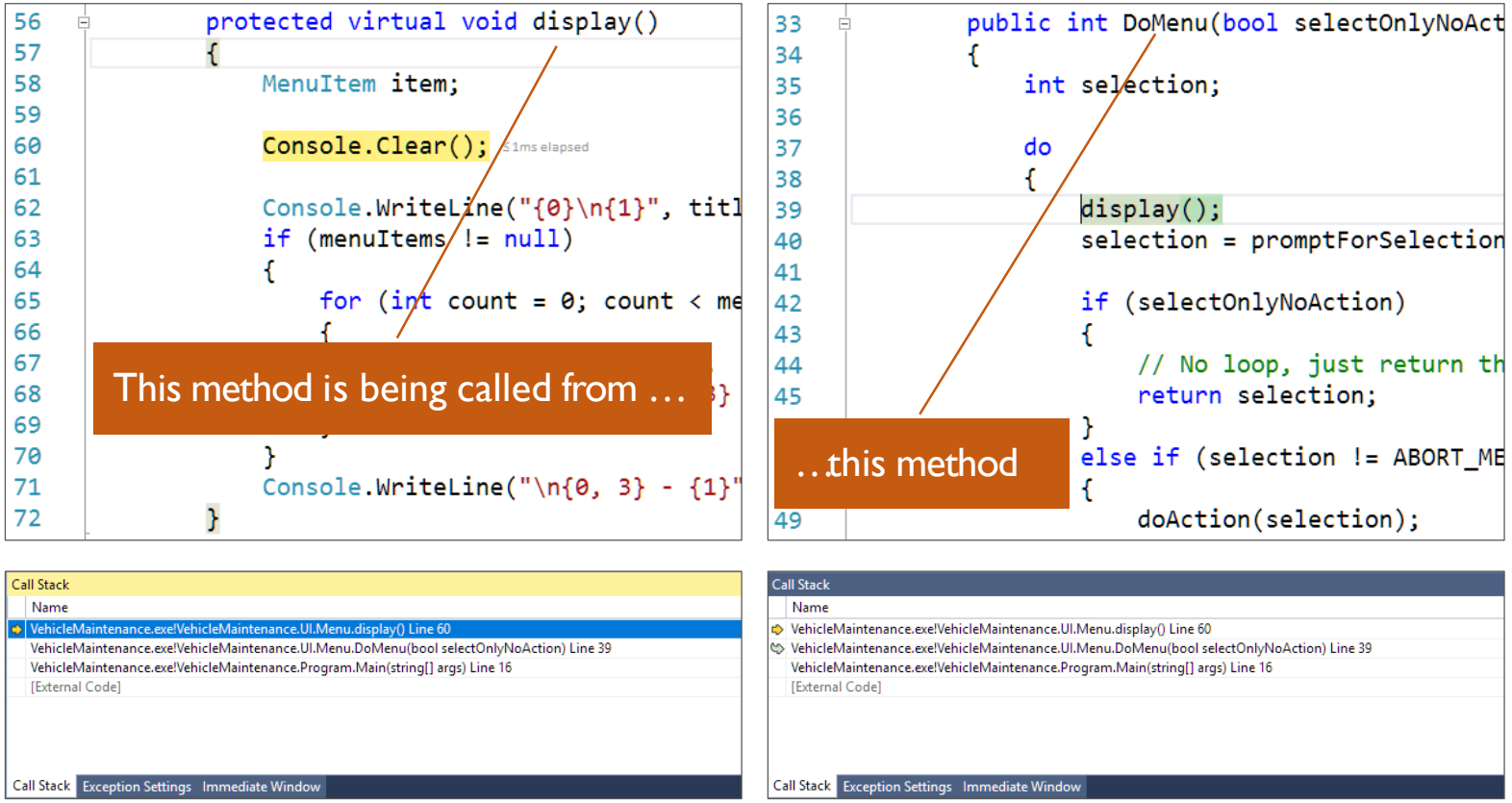
Some expressions have no value but they still execute.

In the Immediate window, you can enter an expression and it will be evaluated and its value is shown.

The Immediate window is only responsive when your program is paused in the debugger.

You may have to turn it on from the menu: Debug | Windows | Immediate

Debugging Tools – Call Stack



Every program has a call stack, which is a list of method calls, ordered from the first call to the most recent one.

Whenever your program is paused in the debugger you can view the call stack.

It shows you which method called the current one and which method called it, and so on, all the way back to Main.

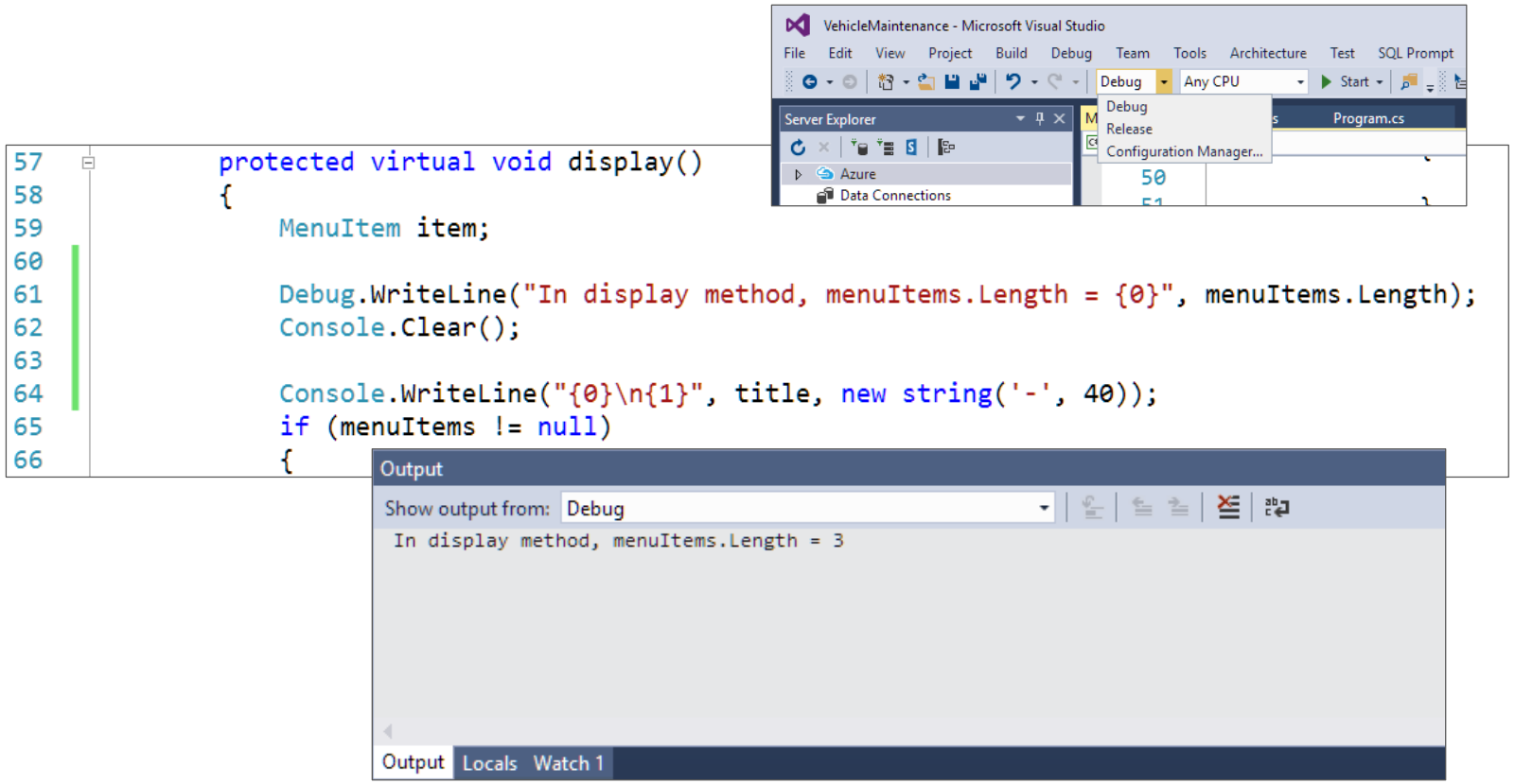
Debugging Tools – Edit and Continue

Edit and Continue is a time-saving feature that enables you to make changes to your source code while your program is in break mode (when your program is paused in the debugger). When you resume execution of the program by choosing Continue or Step, Edit and Continue automatically applies the code changes with some limitations.

This allows you to make changes to your code during a debugging session, instead of having to stop, recompile your entire program, and restart the debugging session.

You can also move/drag the current line of execution (the yellow line) to another line of code before continuing execution.

Debugging Tools – Debug.WriteLine



The Debug class lives in the System.Diagnostics namespace.

Debug.WriteLine lets you print a message to the Output window.

Use this method as often as you like; it does not get compiled when you build in Release mode.

Debugging – More Info

Here are links to Microsoft documentation on Debugging:

- How to debug for absolute beginners
- Visual Studio debugger documentation
- First look at the Visual Studio Debugger
- Navigate through code with the Visual Studio debugger
- Tutorial: Learn to debug C# code using Visual Studio

Here's a short and helpful video to introduce debugging with Visual Studio:

- https://www.youtube.com/watch?v=oZHMeCaHizY