Debugging

What is Debugging?

- Debugging is the process of finding bugs in code
- A software bug is an error or failure in a computer program
- Where did the term bug come from?
  - Grace Hopper coined the term ‘bug’ when a moth short-circuited a computer in 1945

Why Should You Debug?

- The ability to debug code and use debug tools is critical in becoming a successful software developer
- Any time you come across an issue (i.e. bug, error) in your code, your first instinct must be to place a breakpoint and begin stepping through your code to debug the issue
- While the application is in debug mode, you can hover over variables, fields, properties, etc. to see the values change as you step through.

Breakpoint

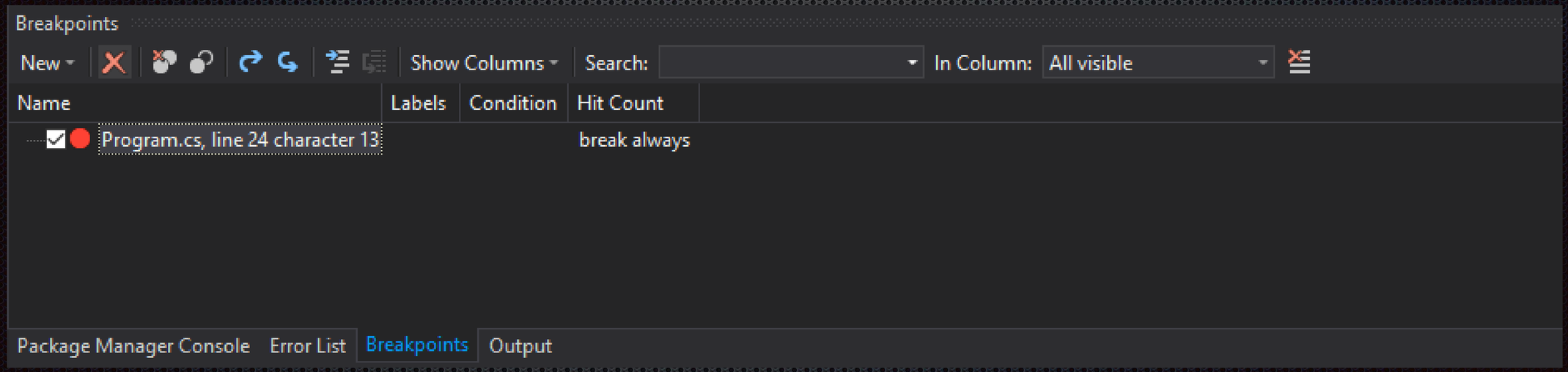
- Drop a breakpoint on a line of code to halt the program when that line is executed
- Hover mouse pointer over a variable or object to see the values stored in that variable or object
- Breakpoint window shows information about all of the breakpoints in your application as well as allows the ability to enable/disable them

Breakpoint

```
0 references
static void Main(string[] args)
{
    double wages;
    double hoursWorked;
    double totalPay;

    Console.WriteLine("Please enter your pay rate:");
    string payRateResult = Console.ReadLine();
    wages = Convert.ToDouble(payRateResult);

    Console.WriteLine("Please enter your hours worked:");
    string hoursWorkedResult = Console.ReadLine();
    hoursWorked = double.Parse(hoursWorkedResult);

    totalPay = wages * hoursWorked;
    Console.WriteLine("Your total pay was: " + totalPay);
    Console.ReadLine();

}
```

Breakpoint Window

- Allows a developer to see all of the breakpoints in an application
- In Visual Studio 2017: Menu Bar → Debug → Windows → Breakpoints



Stepping with Breakpoint

- Step Into (F11)
  - Step into a method call to step through the code block located inside the method
- Step Over (F10)
  - Step over a method call without having to step through each individual line of code inside the method. This typically happens when you know code works.
- Step Out (Shift + F11)
  - Step out a block of code after you have already stepped into it.
- Continue
  - Set up strategic breakpoints and continue past clean code to come across a bug quicker

ContentCreation - Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Test Analyze Window Help

Debug | Any CPU | Start

Can.cs | Program.cs

ContentCreation | ContentCreation.Program | Main(string[] args)

```
using ContentCreation.Classes;
using ContentCreation.DependencyInversion;
using ContentCreation.DependencyInversion.AnotherExample;
using ContentCreation.ErrorHandling;
using ContentCreation.Functions;
using ContentCreation.Lists;
using ContentCreation.Projects;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ContentCreation
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {

            Can mountainDew = new Can("yellow-green", false, 100);
            mountainDew.Open();
        }
    }
}
```

Visual Studio Intellisense

- Visual Studio can identify some of your errors in real time.

```
Console.WriteLine("Enter Desired Temperature:");
int response = Console.ReadLine();
```

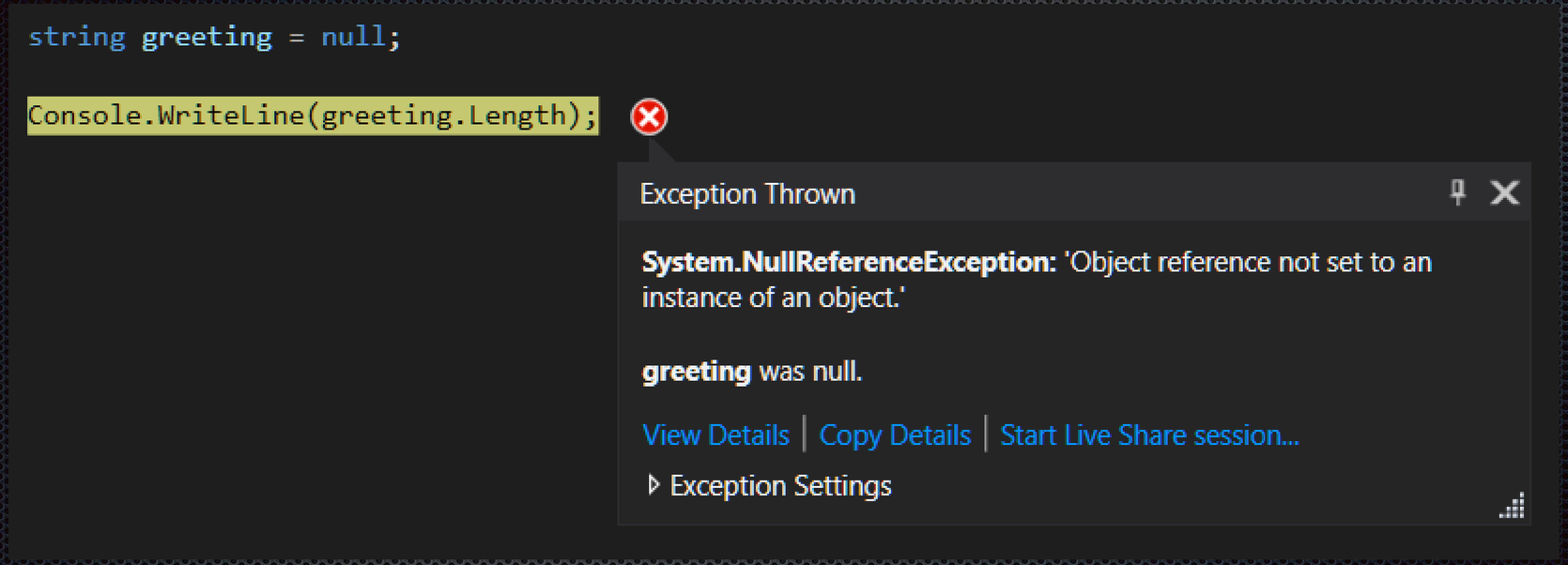
- Any information given about errors is important and needs to be investigated so you can resolve the error. Don't let them stack up!

```
Console.WriteLine("Enter Desired Temperature:");
int response = Console.ReadLine();
```

Cannot implicitly convert type 'string' to 'int'

Exceptions

- Exceptions are errors that slip past intellisense and cause the application to "crash" when encountered while executing.



- Exceptions can be "intercepted" by dropping a breakpoint before the line where it occurs to allow for step-by-step debugging until it is encountered.

Dealing with Errors & Exceptions

- Any time you hit an error or exception, remember: you are not the first person to experience this error. Other developers have solved this in the past, and you will be able to, too!
- If you are unfamiliar with an error or exception you encounter, the first thing to do is research it and try to understand why your code is causing it. Stepping through with a breakpoint is essential.
- There are many common errors you will get to know well. Learning how to deal with them will pay off dividends!