

Getting Started

1

Copyright © Thomas P. Skinner

A Simple Class

```
public class Demo
{
    public int count;
    public string name = "Tom";
    public void Increment()
    {
        ++count;
    }
}
```

A Complete Program

```
public class Demo
{
    public int count;
    public string name = "Tom";
    public void Increment()
    {
        ++count;
    }
    public static void Main()
    {
        Demo demo = new Demo();
        demo.Increment();
    }
}
```

Use of Static








```
public class Demo
{
    public static int count;
    public static string name = "Tom";
    public static void Increment()
    {
        ++count;
    }
    public static void Main()
    {
        Increment();
    }
}
```

Creating a New Project

Visual Studio 2019

Open recent

Search recent (Alt+S) 🔍

	ConsoleApp1.sln C:\Users\tom\Desktop\Labs\ConsoleApp1	1/9/2020 9:41 AM
	TomcatWeb.sln C:\Users\tom\Documents\TomcatWeb	1/9/2020 9:12 AM
	WindowsFormsApp1.sln C:\Users\tom\Desktop\Labs\WindowsFormsApp1	1/8/2020 8:18 AM
	WpfApp1.sln C:\Users\tom\Desktop\labs\WpfApp1	1/8/2020 8:18 AM
	WindowsFormsApp2.sln C:\Users\tom\Desktop\labs\WindowsFormsApp2	
	TomcatWeb.sln C:\Users\tom\Documents\TomcatWeb\TomcatWeb	1/1/2020 8:58 AM
	InteractiveLaserSign.sln C:\...\pro-csharp-7-master\pro-csharp-7-master\Chapter_26\InteractiveLaserSign	12/31/2019 12:59 PM

Click

Get started



Clone or check out code

Get code from an online repository like GitHub or Azure DevOps



Open a project or solution

Open a local Visual Studio project or .sln file



Open a local folder

Navigate and edit code within any folder



Create a new project

Choose a project template with code scaffolding to get started

[Continue without code →](#)

Select Project Type

Create a new project

Recent project templates

- Console App (.NET Framework) C#
- WPF App (.NET Framework) C#
- Windows Forms App (.NET Framework) C#
- Windows Forms App (.NET Core) C#
- ASP.NET Web Application (.NET Framework) C#
- EC512 empty web application
- Empty Project C++
- Empty Project (.NET Framework) C#
- Blank Solution
- ASP.NET Web Application (.NET Framework) Visual Basic
- ASP.NET Core Web Application C#

Search for templates (Alt+S)

C# Windows Console

Set

Console App (.NET Core)
A project for creating a command-line application that can run on .NET Core on Windows, Linux and MacOS.
C# Linux macOS Windows Console

Console App (.NET Framework)
A project for creating a command-line application
C# Windows Console

Select

Not finding what you're looking for?
[Install more tools and features](#)

Back Next

Name & Location

Configure your new project

Console App (.NET Framework) C# Windows Console

Project name
MyApp

Location
C:\Users\tom\Desktop\labs

Solution name ⓘ
MyApp

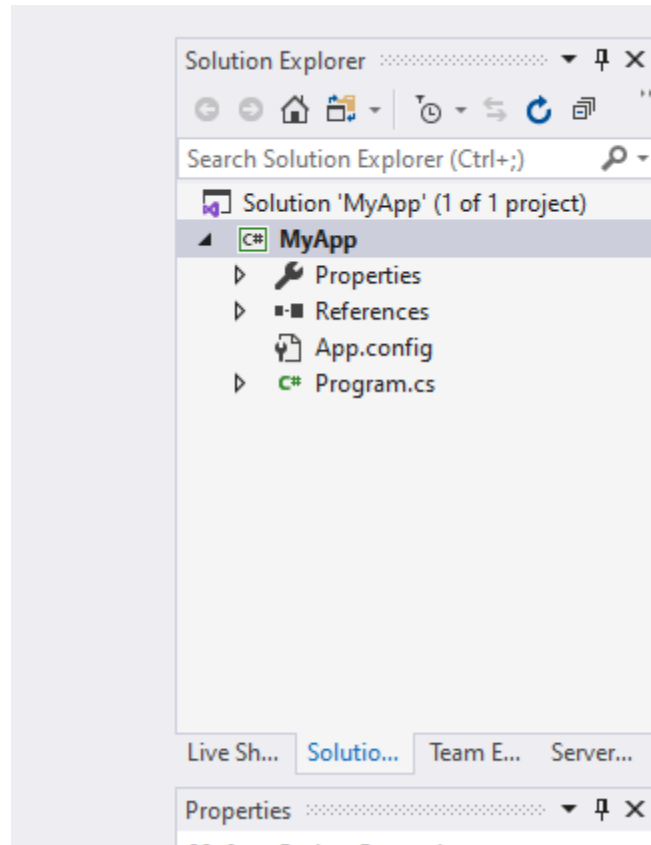
☐ Place solution and project in the same directory

Framework
.NET Framework 4.7.2

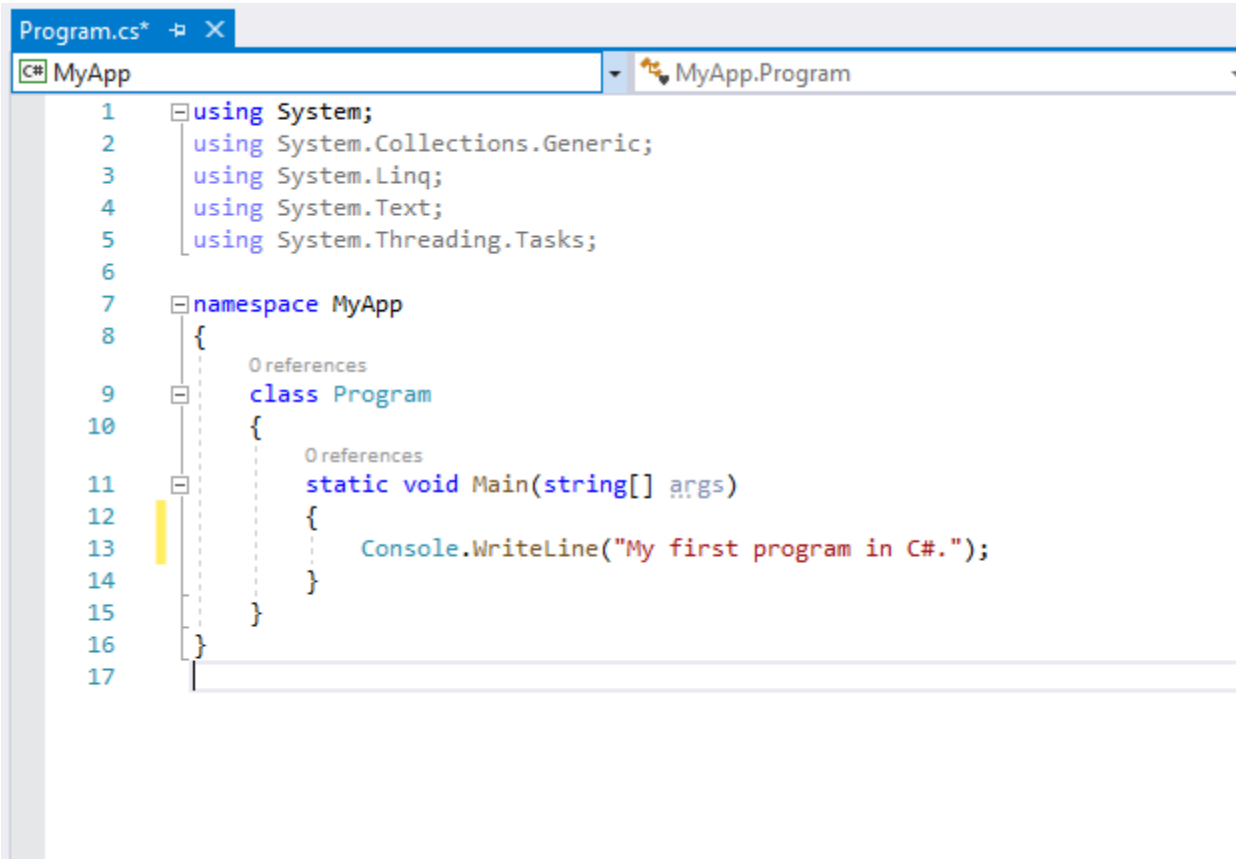
Back Create

DO NOT CHECK

Solution Explorer

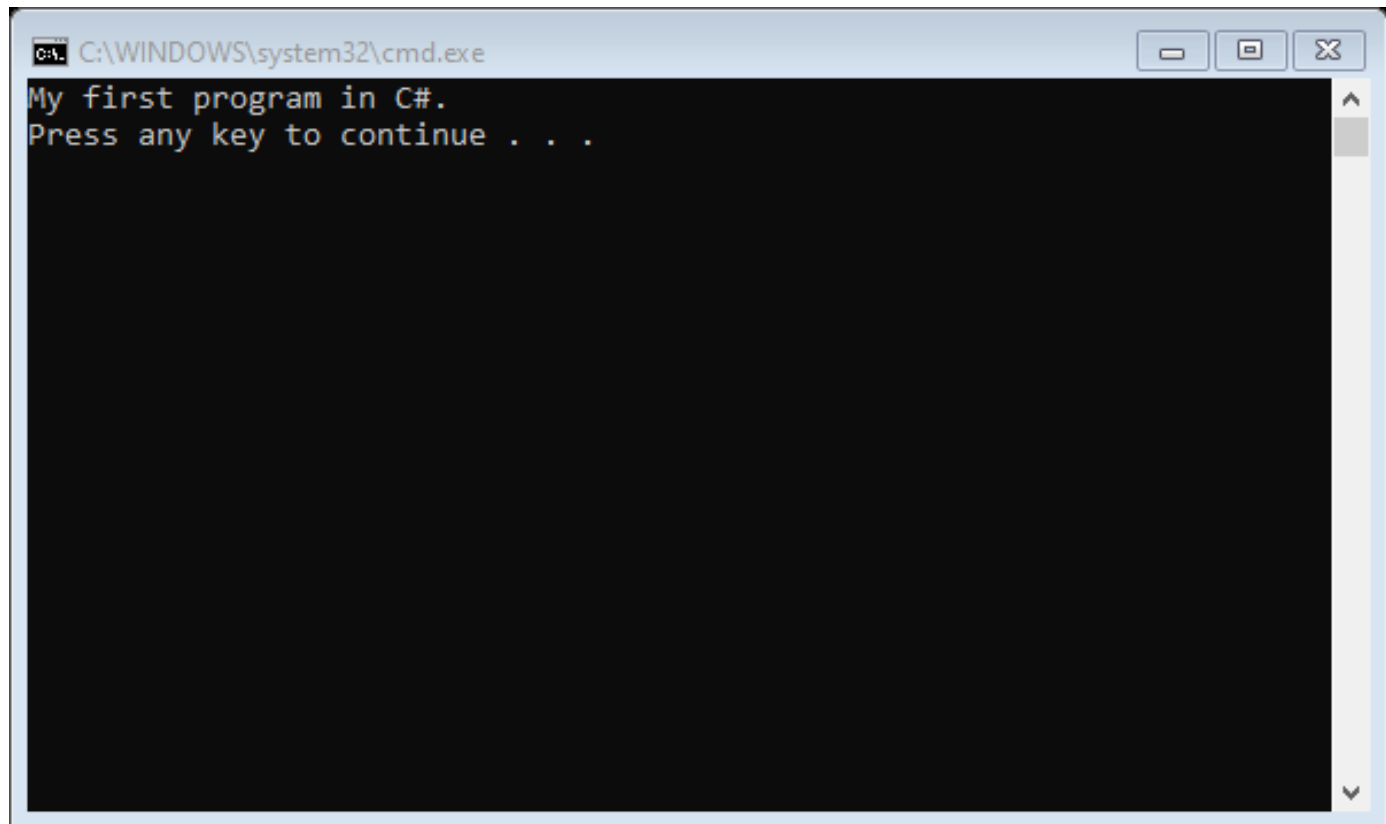


The Program Class



```
Program.cs*  + X
C# MyApp MyApp.Program
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace MyApp
8  {
9      class Program
10     {
11         static void Main(string[] args)
12         {
13             Console.WriteLine("My first program in C#.");
14         }
15     }
16 }
17
```

Select Debug/Start without debugging from the menu



Powers of Two

```
static void Main(string[] args)
{
    int n=2;
    for (int i = 1; i <= 8; ++i)
    {
        Console.WriteLine("{0} {1}", i, n);
        n *= 2;
    }
}
```

Program Output

A screenshot of a Windows command prompt window. The title bar at the top reads "C:\WINDOWS\system32\cmd.exe" and includes standard minimize, maximize, and close buttons. The command prompt area has a black background with white text. It displays a list of eight lines, each containing a number followed by a space and then a power of two: "1 2", "2 4", "3 8", "4 16", "5 32", "6 64", "7 128", and "8 256". Below this list, the text "Press any key to continue . . ." is displayed. The window features a vertical scrollbar on the right side and a horizontal scrollbar at the bottom, both of which are currently at the top and left positions respectively, indicating no scrolling is needed.

```
C:\WINDOWS\system32\cmd.exe  
1 2  
2 4  
3 8  
4 16  
5 32  
6 64  
7 128  
8 256  
Press any key to continue . . .
```

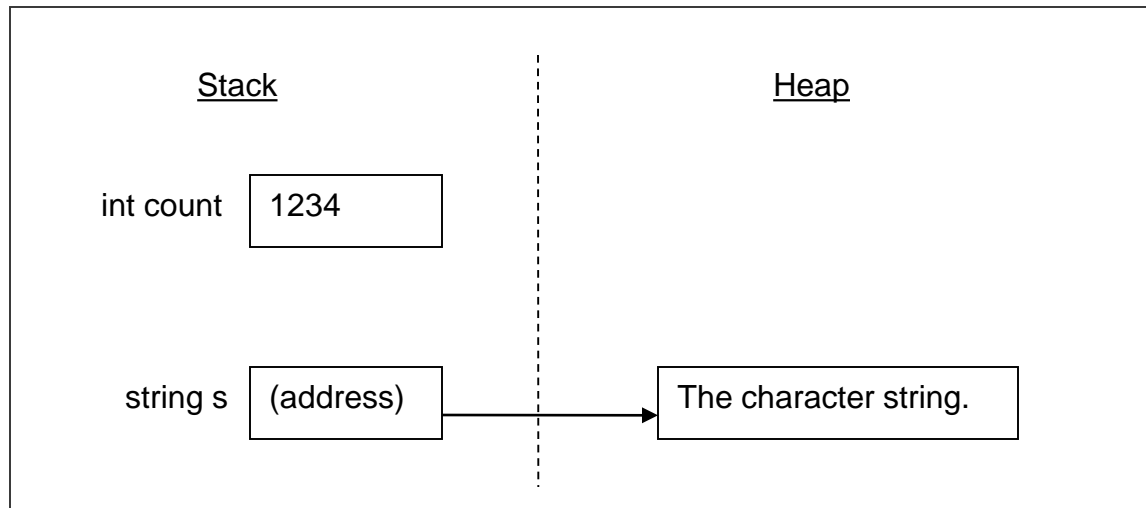
C# Data Types

<i>Type</i>	<i>Size</i>
sbyte	Signed 8-bit
byte	Unsigned 8-bit
char	Unicode 16-bit character
short	Signed 16-bit
ushort	Unsigned 16-bit
int	Signed 32-bit
uint	Unsigned 32-bit
long	Signed 64-bit
ulong	Unsigned 64-bit
decimal	128-bit decimal number (28-29 digits)
float	32-bit floating-point (7 digits)
double	64-bit floating-point (15-16 digits)

Other Data Types

<i>Type</i>	<i>Description</i>
string	A Unicode character string of arbitrary length
bool	A boolean <i>true</i> or <i>false</i> value
object	The base class for all C# classes

Value vs. Reference Types



Boxing and Unboxing

//Boxing

```
int i = 0;
```

```
object o = i;
```

//Unboxing

```
int j = (int) o;
```


Unboxing Error

//The runtime will not accept this

```
int i = 0;
```

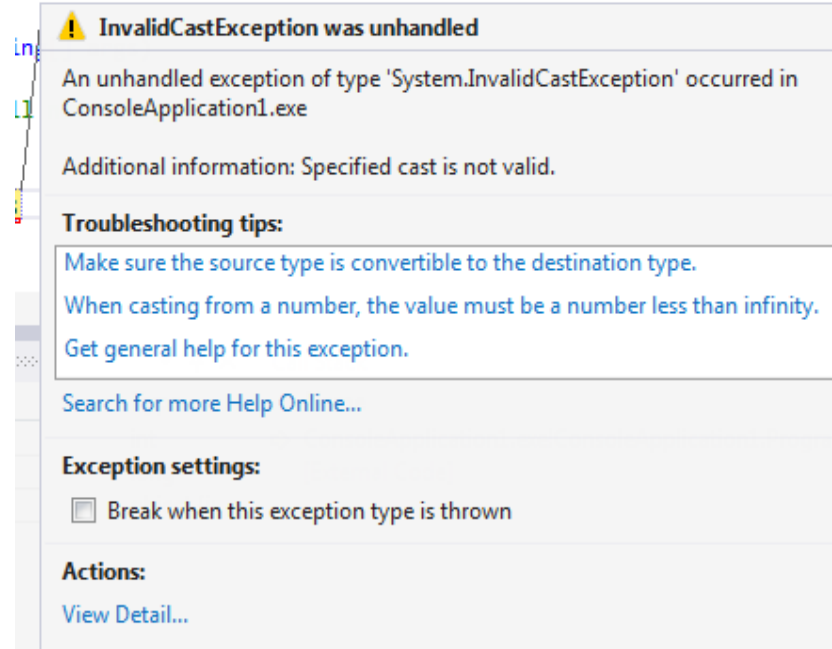
```
object o = i;
```

```
long j = (long) o;
```

//This works

```
long k = (int) o;
```

Runtime Exception



The Ref Keyword

```
static void Main(string[] args)
{
    int i=0;
    foo(ref i);
    Console.WriteLine(i);
}
static void foo(ref int i)
{
    ++i;
}
```

//The output is 1 and not 0

//Without using ref the output would be 0

The Out Keyword

- Variables must be initialized prior to being passed to a method unless “out” is used

```
//This fails even though out is used
static void foo(out int i)
{
    ++i;  //what is the value of I prior to increment?
}
```

```
//On the other hand, this would work just fine:
static void foo(out int i)
{
    i = 100;
}
```

- Why is there no "in" keyword?

Access Modifiers

<i>Keyword</i>	<i>Meaning</i>
public	Access is not restricted.
protected	Access is limited to the containing class or types derived from the containing class.
internal	Access is limited to the current assembly.
protected internal	Access is limited to the current assembly or types derived from the containing class.
private	Access is limited to the containing type.