Getting Started
1

A Simple Class

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

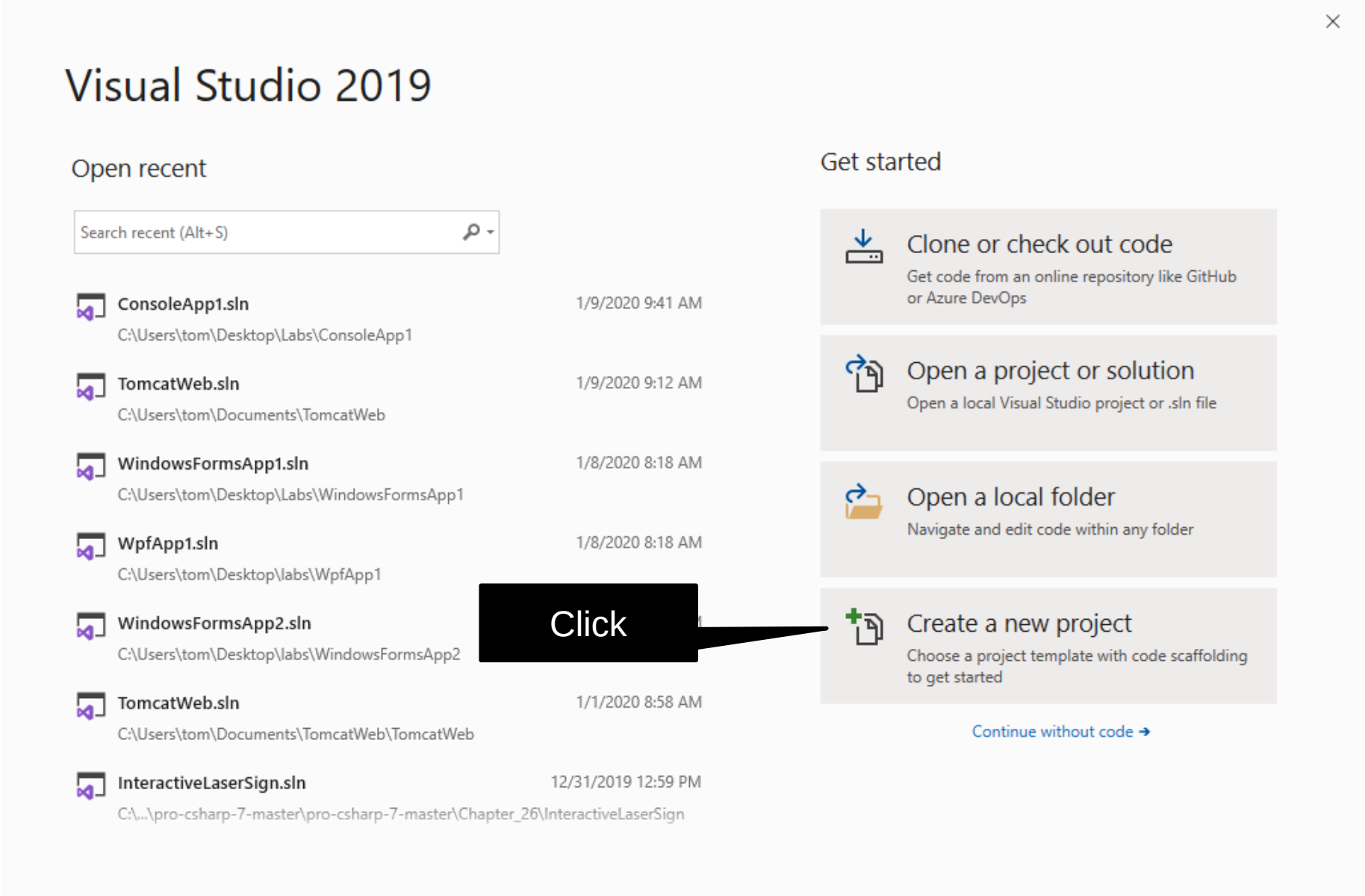
A Complete Program

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

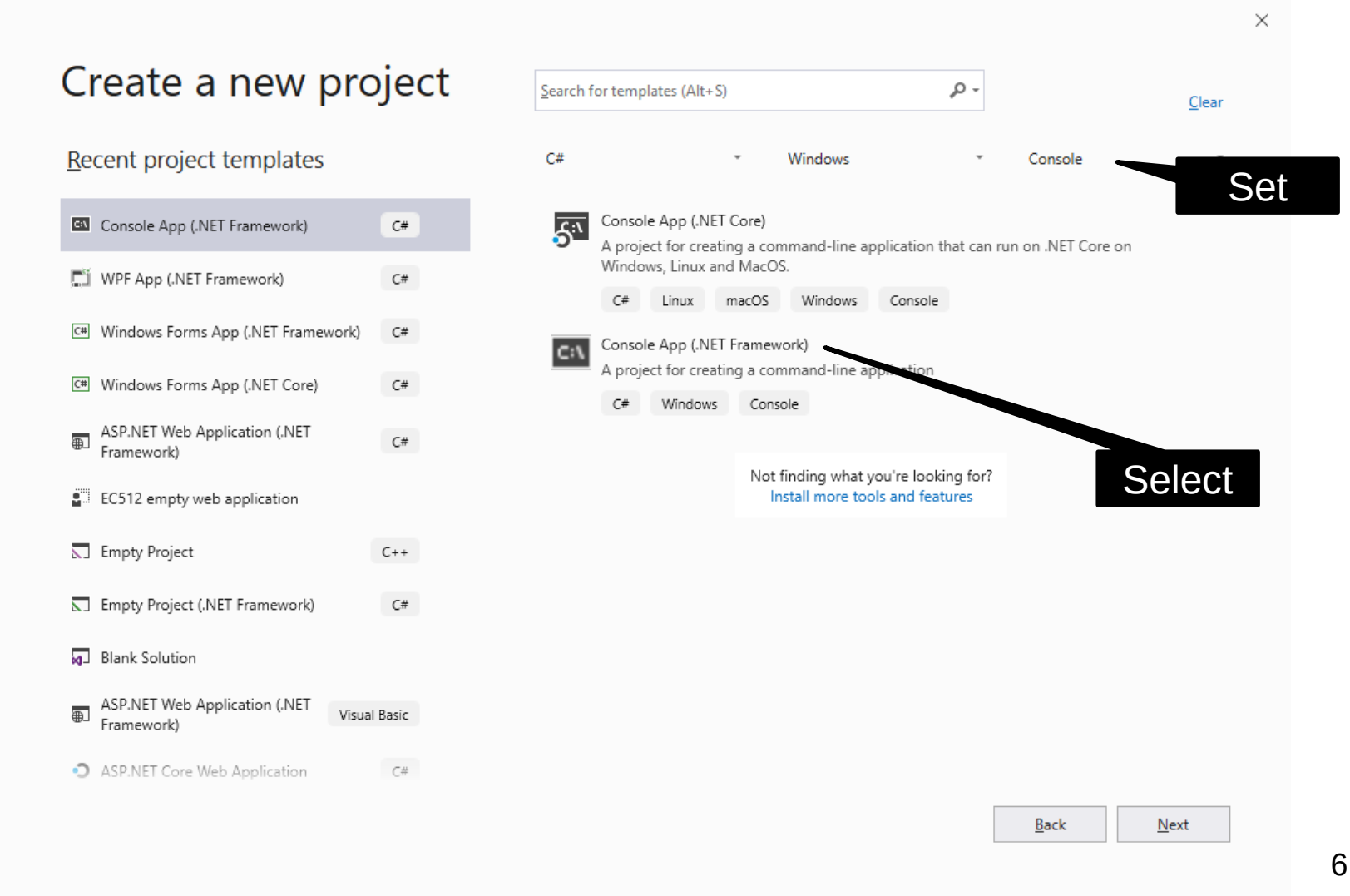
Use of Static

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

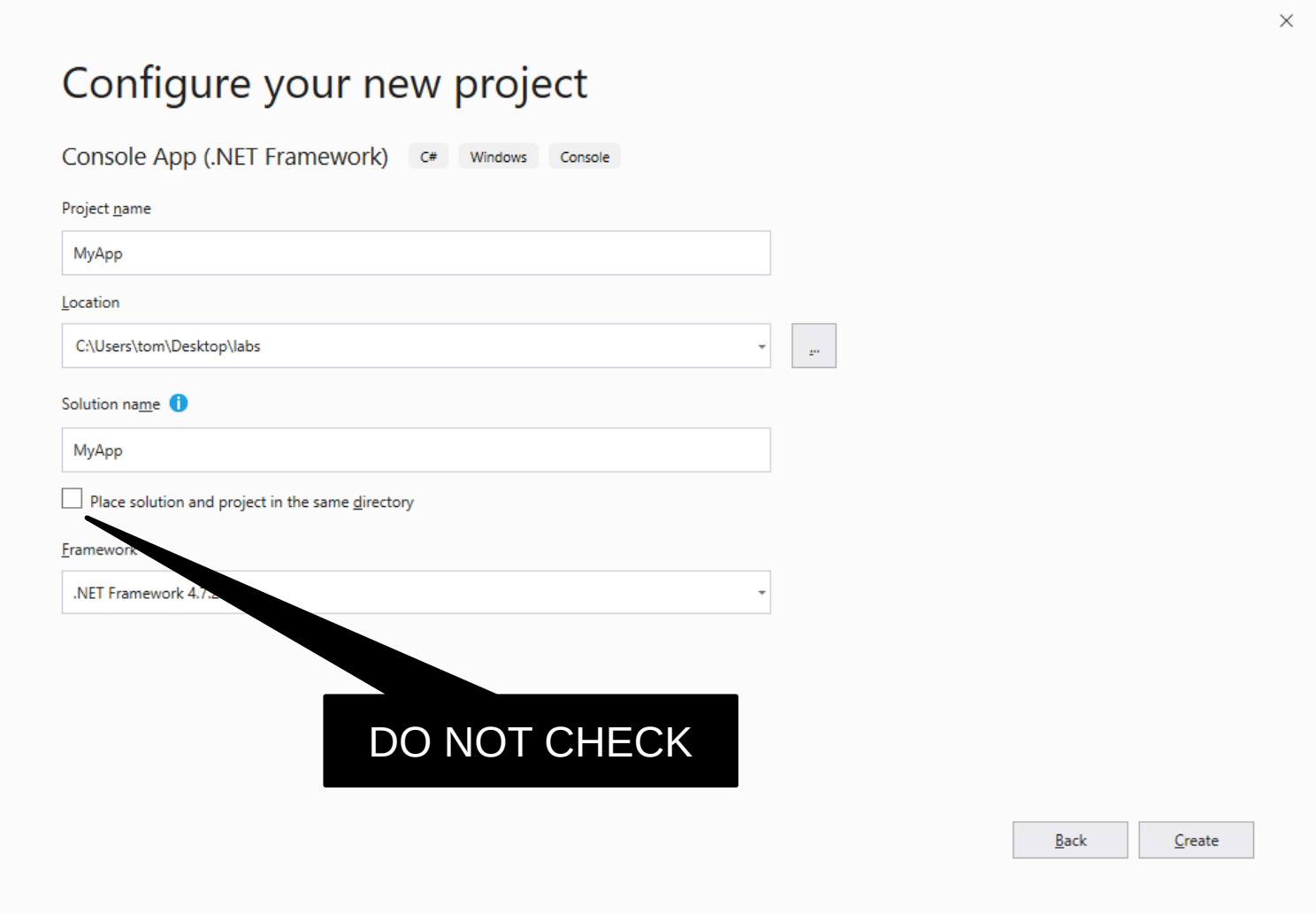
Creating a New Project



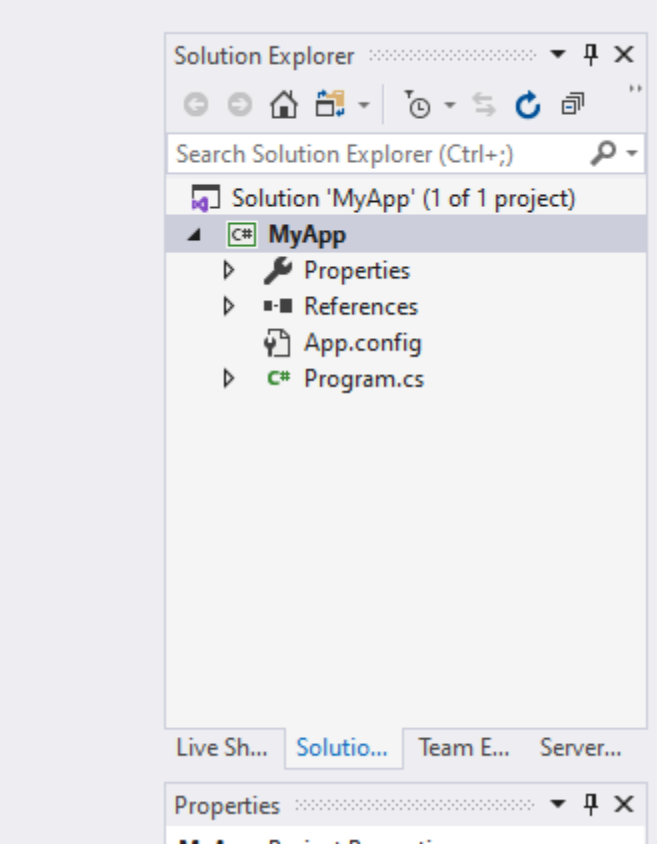
Select Project Type



Name & Location



Solution Explorer



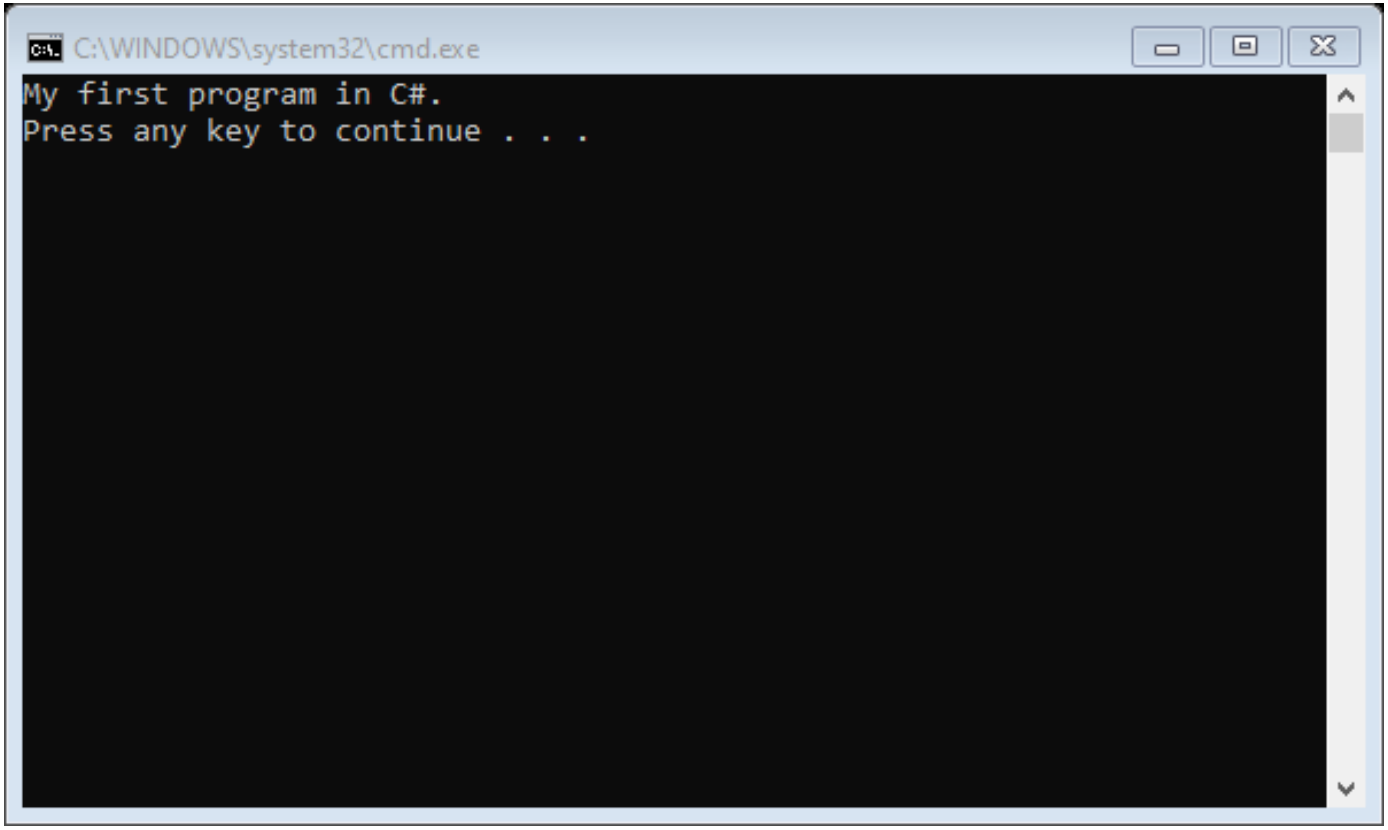
The Prpgram Class

Program.cs*

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   {
        0 references
9       class Program
10      {
            0 references
11          static void Main(string[] args)
12          {
13              Console.WriteLine("My first program in C#.");
14          }
15      }
16  }
17
```

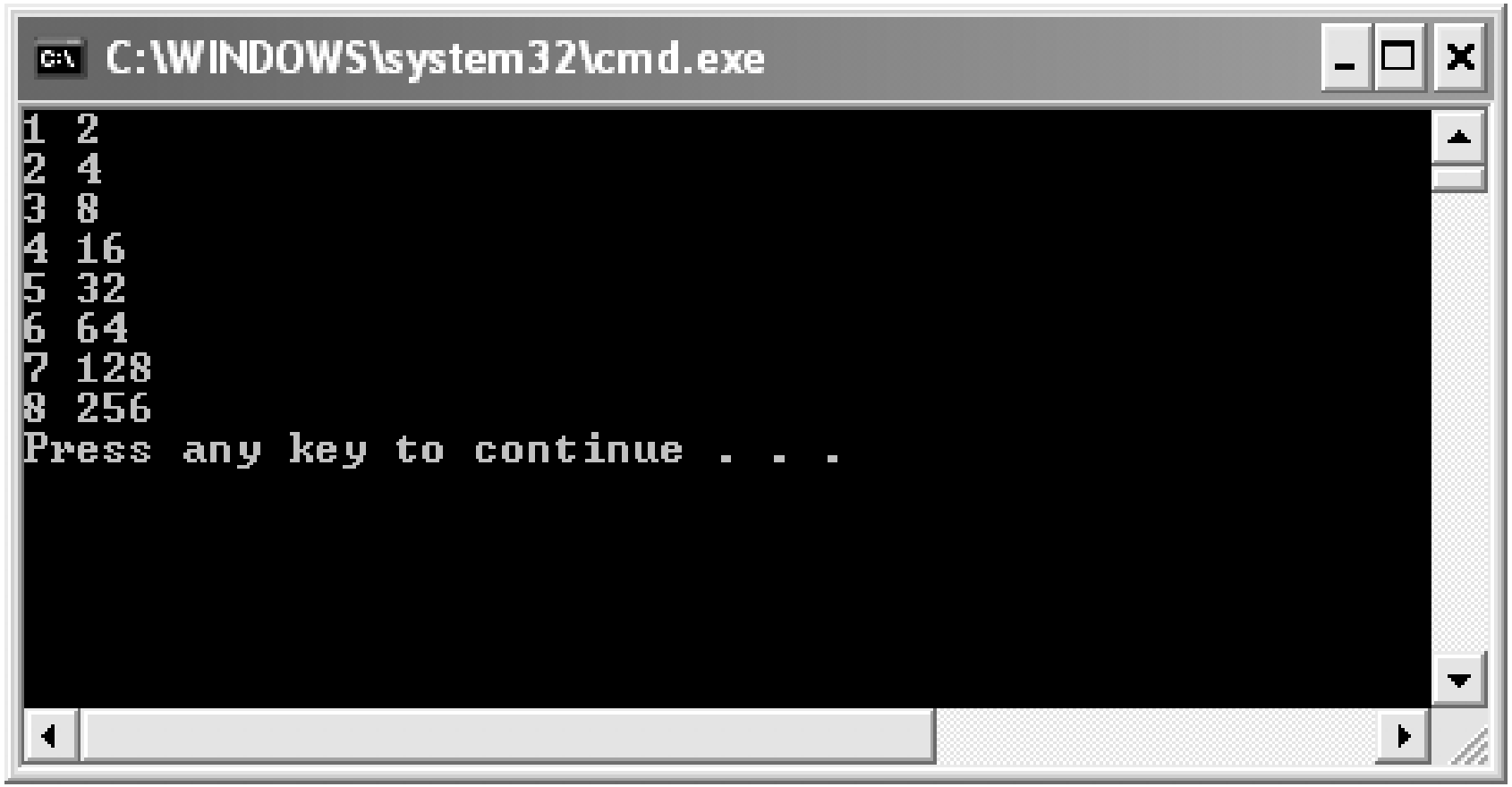
Select Debug/Start without debugging from the menu



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



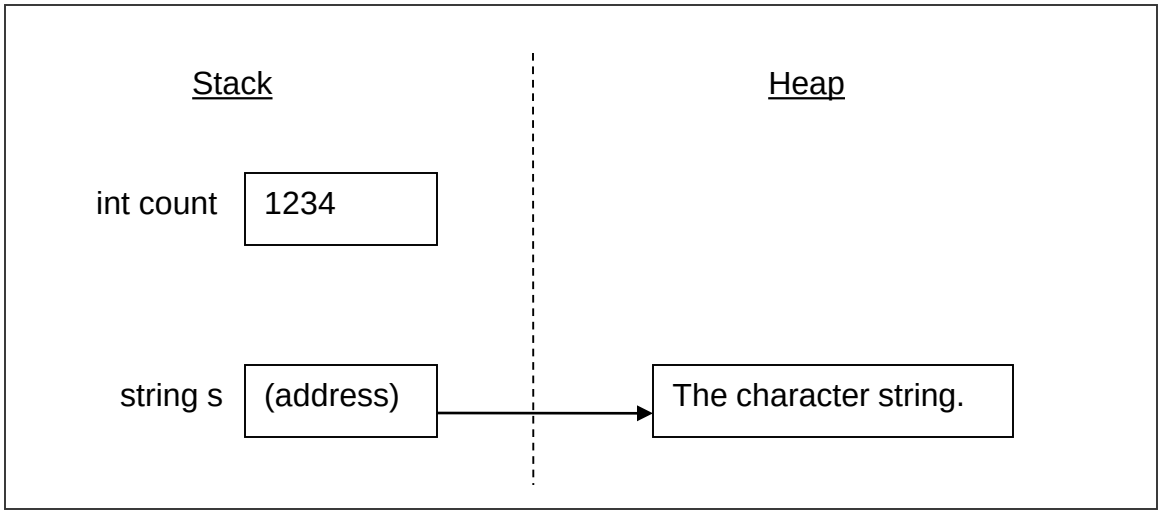
C# Data Types

Type	*Size*
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

Type	*Description*
string	A Unicode charcter string of arbitrary length
bool	A boolean *true* or *false* value
object	The base class for all C# classes

Value vs. Reference Types



Boxing and Unboxing

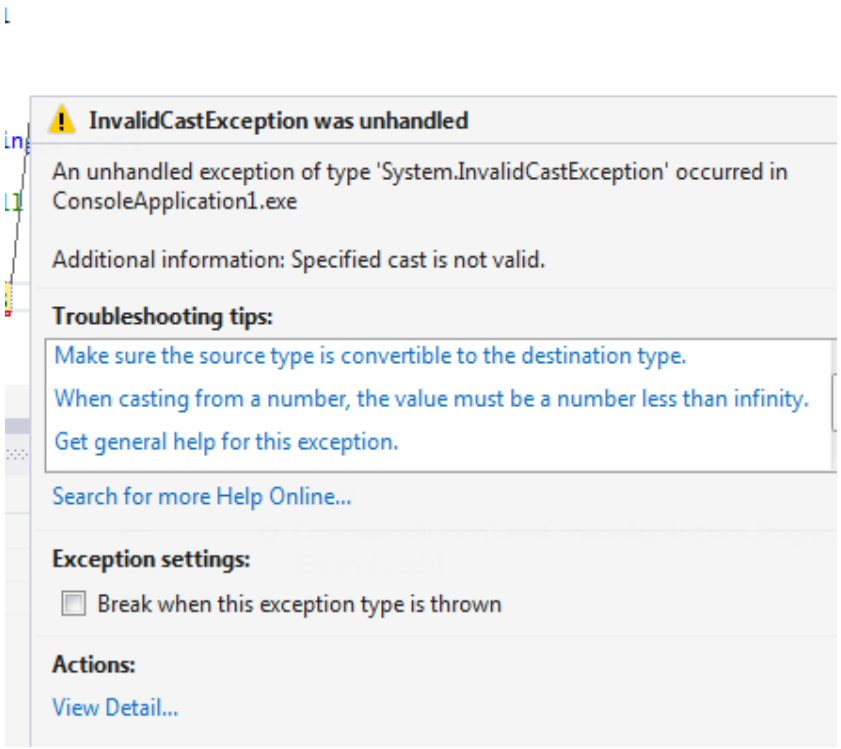
```
//Boxing
int i = 0;
object o = i;

//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

Keyword	***Meaning***
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.