**静态类和静态类成员（C# 编程指南）**

## 静态类

[静态](https://msdn.microsoft.com/zh-CN/library/98f28cdx.aspx)类与非静态类基本相同，但存在一个区别：静态类不能实例化。

也就是说，不能使用 [new](https://msdn.microsoft.com/zh-CN/library/51y09td4.aspx) 关键字创建静态类类型的变量。因为没有实例变量，所以要使用类名本身访问静态类的成员。例如，如果名为 UtilityClass 的静态类有一个名为 MethodA 的公共方法，则按下面的示例所示调用该方法：

UtilityClass.MethodA();

对于只对输入参数进行运算而不获取或设置任何内部实例字段的方法集，静态类可以方便地用作这些方法集的容器。例如，在 .NET Framework 类库中，静态类 [System.Math](https://msdn.microsoft.com/zh-CN/library/system.math.aspx) 包含的方法只执行数学运算，而无需存储或检索特定 [Math](https://msdn.microsoft.com/zh-CN/library/system.math.aspx) 类实例特有的数据。就是说，通过指定类名称和方法名称来应用类成员，如下示例所述。

double dub = -3.14;

Console.WriteLine(Math.Abs(dub));

Console.WriteLine(Math.Floor(dub));

Console.WriteLine(Math.Round(Math.Abs(dub)));

// Output:

// 3.14

// -4

// 3

和所有类类型一样，当加载引用静态类的程序时，.NET Framework 公共语言运行时 (CLR) 将加载该静态类的类型信息。程序不能指定加载静态类的确切时间。但是，可以保证在程序中首次引用该类前加载该类，并初始化该类的字段并调用其静态构造函数。静态构造函数仅调用一次，在程序驻留的应用程序域的生存期内，静态类一直保留在内存中。

| **https://wizardforcel.gitbooks.io/msdn-csharp/content/img/note.jpg 注意** |
| --- |
| 若要创建仅允许创建一个自身实例的非静态类，请参见 [Implementing Singleton in C#](http://go.microsoft.com/fwlink/?LinkID=100567)（在 C# 中实现单一实例）。 |

下表介绍静态类的主要特性：

* 仅包含静态成员。
* 无法实例化。
* 是密封的。
* 不能包含[实例构造函数](https://msdn.microsoft.com/zh-CN/library/k6sa6h87.aspx), 但可以包含静态构造函数

因此，创建静态类与创建仅包含静态成员和私有构造函数的类基本相同。私有构造函数阻止类被实例化。使用静态类的优点在于，编译器能够执行检查以确保不致偶然地添加实例成员。编译器将保证不会创建此类的实例。

静态类是密封的，因此不可被继承。它们不能从除 [Object](https://msdn.microsoft.com/zh-CN/library/system.object.aspx) 外的任何类中继承。静态类不能包含实例构造函数，但可以包含静态构造函数。如果非静态类包含需要进行重要的初始化的静态成员，也应定义静态构造函数。有关更多信息，请参见 [静态构造函数（C# 编程指南）](https://msdn.microsoft.com/zh-CN/library/k9x6w0hc.aspx)。

下面是一个静态类的示例，它包含两个在摄氏温度和华氏温度之间执行来回转换的方法：

public static class TemperatureConverter

{

public static double CelsiusToFahrenheit(string temperatureCelsius)

{

// Convert argument to double for calculations.

double celsius = Double.Parse(temperatureCelsius);

// Convert Celsius to Fahrenheit.

double fahrenheit = (celsius \* 9 / 5) + 32;

return fahrenheit;

}

public static double FahrenheitToCelsius(string temperatureFahrenheit)

{

// Convert argument to double for calculations.

double fahrenheit = Double.Parse(temperatureFahrenheit);

// Convert Fahrenheit to Celsius.

double celsius = (fahrenheit - 32) \* 5 / 9;

return celsius;

}

}

class TestTemperatureConverter

{

static void Main()

{

Console.WriteLine("Please select the convertor direction");

Console.WriteLine("1\. From Celsius to Fahrenheit.");

Console.WriteLine("2\. From Fahrenheit to Celsius.");

Console.Write(":");

string selection = Console.ReadLine();

double F, C = 0;

switch (selection)

{

case "1":

Console.Write("Please enter the Celsius temperature: ");

F = TemperatureConverter.CelsiusToFahrenheit(Console.ReadLine());

Console.WriteLine("Temperature in Fahrenheit: {0:F2}", F);

break;

case "2":

Console.Write("Please enter the Fahrenheit temperature: ");

C = TemperatureConverter.FahrenheitToCelsius(Console.ReadLine());

Console.WriteLine("Temperature in Celsius: {0:F2}", C);

break;

default:

Console.WriteLine("Please select a convertor.");

break;

}

// Keep the console window open in debug mode.

Console.WriteLine("Press any key to exit.");

Console.ReadKey();

}

}

/\* Example Output:

Please select the convertor direction

1\. From Celsius to Fahrenheit.

2\. From Fahrenheit to Celsius.

:2

Please enter the Fahrenheit temperature: 20

Temperature in Celsius: -6.67

Press any key to exit.

\*/

**静态成员**

非静态类可以包含静态的方法、字段、属性或事件。即使没有创建类的实例，也可以调用该类中的静态成员。始终通过类名而不是实例名称访问静态成员。无论对一个类创建多少个实例，它的静态成员都只有一个副本。静态方法和属性不能访问其包含类型中的非静态字段和事件，并且不能访问任何对象的实例变量（除非在方法参数中显式传递）。

更常见的做法是声明具有一些静态成员的非静态类，而不是将整个类声明为静态类。静态字段有两个常见的用法：一是记录已实例化对象的个数，二是存储必须在所有实例之间共享的值。

静态方法可以被重载但不能被重写，因为它们属于类，不属于类的任何实例。

虽然字段不能声明为 static const，但 [const](https://msdn.microsoft.com/zh-CN/library/e6w8fe1b.aspx) 字段的行为在本质上是静态的。这样的字段属于类型，不属于类型的实例。因此，可以同对待静态字段一样使用 ClassName.MemberName 表示法来访问 const 字段。不需要对象实例。

C# 不支持静态局部变量（在方法范围内声明的变量）。

通过在成员的返回类型之前使用 **static** 关键字可以声明静态类成员，如下面的示例所示：

public class Automobile

{

public static int NumberOfWheels = 4;

public static int SizeOfGasTank

{

get

{

return 15;

}

}

public static void Drive() { }

public static event EventType RunOutOfGas;

// Other non-static fields and properties...

}

静态成员在第一次被访问之前并且在调用静态构造函数（如有存在）之前进行初始化。若要访问静态类成员，应使用类名而不是变量名来指定该成员的位置，如下面的示例所示：

Automobile.Drive();

int i = Automobile.NumberOfWheels;

如果类包含静态字段，请提供在加载类时初始化这些字段的静态构造函数。

对静态方法的调用以 Microsoft 中间语言 (MSIL) 生成调用指令，而对实例方法的调用生成 **callvirt** 指令，该指令还检查 null 对象引用。但是，两者之间的性能差异在大多数时候并不明显。

**C# 语言规范**

有关详细信息，请参阅 [C# 语言规范](https://msdn.microsoft.com/zh-CN/library/ms228593.aspx)。该语言规范是 C# 语法和用法的权威资料。

**请参阅**

[C# 编程指南](https://msdn.microsoft.com/zh-CN/library/67ef8sbd.aspx)

[static（C# 参考）](https://msdn.microsoft.com/zh-CN/library/98f28cdx.aspx)

[类（C# 编程指南）](https://msdn.microsoft.com/zh-CN/library/x9afc042.aspx)

[class（C# 参考）](https://msdn.microsoft.com/zh-CN/library/0b0thckt.aspx)

[静态构造函数（C# 编程指南）](https://msdn.microsoft.com/zh-CN/library/k9x6w0hc.aspx)

[实例构造函数（C# 编程指南）](https://msdn.microsoft.com/zh-CN/library/k6sa6h87.aspx)