


# Static Classes and Static Class Members (C# Programming Guide)

 docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/static-classes-and-static-class-members  
Manish\_Nautiyal

[Skip to main content](#)

Microsoft



- 07/20/2015
- 5 minutes to read
- Contributors

A static class is basically the same as a non-static class, but there is one difference: a static class cannot be instantiated. In other words, you cannot use the new keyword to create a variable of the class type. Because there is no instance variable, you access the members of a static class by using the class name itself. For example, if you have a static class that is named `UtilityClass` that has a public method named `MethodA`, you call the method as shown in the following example:

C#

```
UtilityClass.MethodA();
```

A static class can be used as a convenient container for sets of methods that just operate on input parameters and do not have to get or set any internal instance fields. For example, in the .NET Framework Class Library, the static `System.Math` class contains methods that perform mathematical operations, without any requirement to store or retrieve data that is unique to a particular instance of the `Math` class. That is, you apply the members of the class by specifying the class name and the method name, as shown in the following example. 7

C#

```
double dub = -3.14;  
Console.WriteLine(Math.Abs(dub));  
Console.WriteLine(Math.Floor(dub));  
Console.WriteLine(Math.Round(Math.Abs(dub)));
```

As is the case with all class types, the type information for a static class is loaded by the .NET Framework common language runtime (CLR) when the program that references the class is loaded. The program cannot specify exactly when the class is loaded. However, it is guaranteed to be loaded and to have its fields initialized and its static constructor called

before the class is referenced for the first time in your program. A static constructor is only called one time, and a static class remains in memory for the lifetime of the application domain in which your program resides. 1

#### Note

To create a non-static class that allows only one instance of itself to be created, see [Implementing Singleton in C#](#).

The following list provides the main features of a static class:

- Contains only static members.
- Cannot be instantiated.
- Is sealed.
- Cannot contain [Instance Constructors](#).

#### 1

Creating a static class is therefore basically the same as creating a class that contains only static members and a private constructor. A private constructor prevents the class from being instantiated. The advantage of using a static class is that the compiler can check to make sure that no instance members are accidentally added. The compiler will guarantee that instances of this class cannot be created.

Static classes are sealed and therefore cannot be inherited. They cannot inherit from any class except [Object](#). Static classes cannot contain an instance constructor; however, they can contain a static constructor. Non-static classes should also define a static constructor if the class contains static members that require non-trivial initialization. For more information, see [Static Constructors](#).

## Example

---

Here is an example of a static class that contains two methods that convert temperature from Celsius to Fahrenheit and from Fahrenheit to Celsius:

C#

```
public static class TemperatureConverter
{
    public static double CelsiusToFahrenheit(string temperatureCelsius)
    {
        double celsius = Double.Parse(temperatureCelsius);

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

        return fahrenheit;
    }
}
```

```

public static double FahrenheitToCelsius(string temperatureFahrenheit)
{

    double fahrenheit = Double.Parse(temperatureFahrenheit);

    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;
        }

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

```

## Static Members

---

A non-static class can contain static methods, fields, properties, or events. The static member is callable on a class even when no instance of the class has been created. The static member is always accessed by the class name, not the instance name. Only one

copy of a static member exists, regardless of how many instances of the class are created. Static methods and properties cannot access non-static fields and events in their containing type, and they cannot access an instance variable of any object unless it is explicitly passed in a method parameter.

It is more typical to declare a non-static class with some static members, than to declare an entire class as static. Two common uses of static fields are to keep a count of the number of objects that have been instantiated, or to store a value that must be shared among all instances. 2

Static methods can be overloaded but not overridden, because they belong to the class, and not to any instance of the class. 1

Although a field cannot be declared as `static const`, a `const` field is essentially static in its behavior. It belongs to the type, not to instances of the type. Therefore, const fields can be accessed by using the same `ClassName.MemberName` notation that is used for static fields. No object instance is required.

C# does not support static local variables (variables that are declared in method scope). 4

You declare static class members by using the `static` keyword before the return type of the member, as shown in the following example:

C#

```
public class Automobile
{
    public static int NumberOfWheels = 4;
    public static int SizeOfGasTank
    {
        get
        {
            return 15;
        }
    }
    public static void Drive() { }
    public static event EventType RunOutOfGas;
}
```

Static members are initialized before the static member is accessed for the first time and before the static constructor, if there is one, is called. To access a static class member, use the name of the class instead of a variable name to specify the location of the member, as shown in the following example:

C#

```
Automobile.Drive();
int i = Automobile.NumberOfWheels;
```

If your class contains static fields, provide a static constructor that initializes them when the class is loaded.

A call to a static method generates a call instruction in Microsoft intermediate language (MSIL), whereas a call to an instance method generates a `callvirt` instruction, which also checks for a null object references. However, most of the time the performance difference between the two is not significant.

## C# Language Specification

---

For more information, see the [C# Language Specification](#). The language specification is the definitive source for C# syntax and usage.

## See Also

---

[C# Programming Guide](#)

[static](#)

[Classes](#)

[class](#)

[Static Constructors](#)

[Instance Constructors](#)

### Note

The feedback system for this content will be changing soon. Old comments will not be carried over. If content within a comment thread is important to you, please save a copy. For more information on the upcoming change, [we invite you to read our blog post](#)