Clases Estaticas

Una clase estática es básicamente lo mismo que una clase no estática, con la diferencia de que no se pueden crear instancias de una clase estática. En otras palabras, no puede usar la palabra clave new para crear una variable del tipo de clase. Dado que no hay ninguna variable de instancia, para tener acceso a los miembros de una clase estática, debe usar el nombre de la clase. Por ejemplo, si tiene una clase estática denominada UtilityClass que tiene un método estático público denominado MethodA, llame al método tal como se muestra en el ejemplo siguiente:

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

Es posible usar una clase estática como un contenedor adecuado para conjuntos de métodos que solo funcionan en parámetros de entrada y que no tienen que obtener ni establecer campos de instancias internas. Por ejemplo, en la biblioteca de clases .NET Framework, la clase estática System.Math contiene métodos que realizan operaciones matemáticas, sin ningún requisito para almacenar o recuperar datos que sean únicos de una instancia concreta de la clase Math. Es decir, para aplicar los miembros de la clase, debe especificar el nombre de clase y el nombre de método, como se muestra en el ejemplo siguiente.

```
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
```

Como sucede con todos los tipos de clase, Common Language Runtime (CLR) de .NET Framework carga la información de tipo de una clase estática cuando se carga el programa que hace referencia a la clase. El programa no puede especificar exactamente cuándo se carga la clase, pero existe la garantía de que se cargará, de que sus campos se inicializarán y de que se llamará a su constructor estático antes de que se haga referencia a la clase por primera vez en el programa. Solo se llama una vez a un constructor estático, y una clase estática permanece en memoria durante la vigencia del dominio de aplicación en el que reside el programa.

La siguiente lista contiene las características principales de una clase estática:

- Solo contiene miembros estáticos.
- No se pueden crear instancias de ella.
- Está sellada.
- No puede contener constructores de instancias.

Por lo tanto, crear una clase estática es básicamente lo mismo que crear una clase que contiene solo miembros estáticos y un constructor privado. Un constructor privado impide que se creen instancias de la clase. La ventaja de usar una clase estática es que el compilador puede comprobar que no se agregue accidentalmente ningún miembro de instancia. El compilador garantizará que no se creen instancias de esta clase.

Las clases estáticas están selladas y, por lo tanto, no pueden heredarse. No pueden heredar de ninguna clase excepto Object. Las clases estáticas no pueden contener un constructor de instancias, pero pueden contener un constructor estático. Las clases no estáticas también deben definir un constructor estático si la clase contiene miembros estáticos que requieren inicialización no trivial. Para obtener más información, vea Constructores estáticos.

Ejemplo

A continuación se muestra un ejemplo de una clase estática que contiene dos métodos que convierten la temperatura de grados Celsius a grados Fahrenheit y viceversa:

```
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.
 */
```

Miembros estáticos

Una clase no estática puede contener métodos, campos, propiedades o eventos estáticos. El miembro estático es invocable en una clase, incluso si no se ha creado ninguna instancia de la clase. Siempre se tiene acceso al miembro estático con el nombre de clase, no con el nombre de instancia. Solo existe una copia de un miembro estático, independientemente del número de instancias de la clase que se creen. Los métodos y las propiedades estáticos no pueden tener acceso a campos y eventos no estáticos en su tipo contenedor, y tampoco pueden tener acceso a una variable de instancia de un objeto a menos que se pase explícitamente en un parámetro de método.

Es más habitual declarar una clase no estática con algunos miembros estáticos que declarar toda una clase como estática. Dos usos habituales de los campos estáticos son llevar la cuenta del número de objetos de los que se han creado instancias y almacenar un valor que se debe compartir entre todas las instancias.

Los métodos estáticos se pueden sobrecargar pero no invalidar, ya que pertenecen a la clase y no a una instancia de la clase.

Aunque un campo no se puede declarar como `static const`, el campo const es básicamente estático en su comportamiento. Pertenece al tipo, no a las instancias del tipo. Por lo tanto, se puede tener acceso a campos const mediante la misma notación `ClassName.MemberName` que se usa para los campos estáticos. No se requiere ninguna instancia de objeto.

C# no admite variables locales estáticas (variables que se declaran en el ámbito del método).

Para declarar miembros de clases estáticas, use la palabra clave `static` antes del tipo de valor devuelto del miembro, como se muestra en el ejemplo siguiente:

```
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...
}
```

Los miembros estáticos se inicializan antes de que se obtenga acceso por primera vez al miembro estático y antes de que se llame al constructor estático, en caso de haberlo. Para tener acceso a un miembro de clase estática, use el nombre de la clase en lugar de un nombre de variable para especificar la ubicación del miembro, como se muestra en el ejemplo siguiente:

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

Si la clase contiene campos estáticos, proporcione un constructor estático que los inicialice al cargar la clase.

Una llamada a un método estático genera una instrucción de llamada en Lenguaje Intermedio de Microsoft (MSIL), mientras que una llamada a un método de instancia genera una instrucción `callvirt`, que también comprueba si hay referencias a un objeto NULL, pero la mayoría de las veces la diferencia de rendimiento entre las dos no es significativo.