Classes estáticas e membros da classe estática (guia de programação do C#)

Uma classe estática é basicamente a mesma que uma classe não-estática, mas há uma diferença: uma classe estática não pode ser instanciada. Em outras palavras, você não pode usar a Palavra-chave new para criar uma variável do tipo da classe. Como não há nenhuma variável de instância, você acessa os membros de uma classe estática usando o nome de classe propriamente dito. Por exemplo, se você tiver uma classe estática denominada UtilityClass contendo um método público chamado MethodA, você chama o método conforme o exemplo seguinte:

C#
```
UtilityClass.MethodA();
```

Uma classe estática pode ser usada como um recipiente conveniente para conjuntos de métodos que só operam nos parâmetros de entrada e não é necessário obter ou definir quaisquer campos de instância interno. Por exemplo, na.NET Framework Class Library, estáticaSystem.Math classe contém métodos que executam operações matemáticas, sem qualquer necessidade de armazenar ou recuperar dados que é exclusivos para uma determinada instância da Math classe. Ou seja, aplicar os membros da classe, especificando o nome da classe e o nome do método, conforme mostrado no exemplo a seguir.

```
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
```

Tal como acontece com todos os tipos de classe, as informações do tipo para uma classe estática são carregadas pelo .NET Framework common language runtime (CLR) quando o programa que referencia a classe é carregado. O programa não pode especificar exatamente quando a classe é carregada. No entanto, ele é garantido para ser carregado e ter seus campos inicializados e seu construtor estático chamado antes que a classe é referenciada pela primeira vez no seu programa. Um construtor estático é chamado somente uma vez e uma classe estática permanece na memória durante a vida útil do domínio do aplicativo no qual reside o seu programa.

A lista a seguir fornece os principais recursos de uma classe Estática:

- Contém apenas membros estáticos.
- Não é possível criar uma instância.
- É sealed.
- Não pode conter Construtores de instância.

Criar uma classe estática é, portanto, basicamente o mesmo que criar uma classe que contém apenas membros estáticos e um construtor particular. Um construtor particular impede que a classe seja instanciado. A vantagem de usar uma classe estática é que o compilador pode verificar para certificar-se de que nenhum membro de instância acidentalmente é adicionado. O compilador garantirá que as instâncias dessa classe não podem ser criadas.

Classes estáticas são seladas e, portanto, não podem ser herdadas. Eles não podem herdar de qualquer classe, exceto Object. Classes estáticas não podem conter um construtor de instância; No entanto, eles podem conter um construtor estático. Classes não-estático também devem definir um construtor estático se a classe contém membros estáticos que requerem inicialização não trivial. Para obter mais informações, consulte Construtores estáticos (guia de programação do C#).

Exemplo

Aqui está um exemplo de uma classe estática que contém dois métodos que converter a temperatura em Celsius para Fahrenheit e Fahrenheit para Celsius:

C#

```
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.
 */
```

Membros estáticos

Uma classe non-static pode conter métodos, campos, propriedades ou eventos estáticos. O membro estático é que pode ser chamado em uma classe, mesmo quando nenhuma instância da classe foi criada. O membro

estático sempre é acessado pelo nome da classe, não o nome da instância. Apenas uma copia de um membro estático existe, independentemente de quantas instâncias da classe são criadas.

Propriedades e métodos estáticos não podem acessar campos e eventos não estáticos em seu tipo de recipiente e eles não podem acessar uma variável de instância de qualquer objeto, a menos que explicitamente, ele seja passado em um parâmetro de método.

É mais comum declarar uma classe non-static com alguns membros estáticos, que para declarar uma classe inteira como estático. Dois usos comum de campos estáticos são para manter uma conta do número de objetos que foi criada, ou para armazenar valor comum que deve ser compartilhado entre todas as instâncias.

Métodos estáticos podem ser sobrecarregados, mas não substituídos, porque eles pertencem à classe e não para qualquer instância da classe.

Embora um campo não pode ser declarado como `static const`, um const o campo é essencialmente static seu comportamento. Ele pertence ao tipo, não para instâncias do tipo. Portanto, os campos const podem ser acessados usando o mesmo`ClassName.MemberName` notação usada para campos estáticos. Nenhuma instância do objeto é necessária.

C# não oferece suporte a variáveis locais estáticas (variáveis que são declaradas no escopo do método).

Você declara a membros da classe estática usando o **static** palavra-chave antes do tipo de retorno do membro, conforme mostrado no exemplo a seguir:

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;

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

Membros estáticos são inicializados antes do membro estático é acessado pela primeira vez e antes do construtor estático, se houver um, é chamado. Para acessar um membro de classe estática, use o nome da classe em vez de um nome de variável para especificar o local do membro, conforme mostrado no exemplo a seguir:

C#

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

Se sua classe contém campos estáticos, fornece um construtor estático que inicializa-los quando a classe é carregada.

Uma chamada para um método estático gera uma instrução de chamada na Microsoft intermediate language (MSIL), enquanto que uma chamada para um método de instância gera um **callvirt** as instruções, que também procura faz referência a um objeto nulo. No entanto, na maioria das vezes a diferença de desempenho entre os dois não é significante.