Sintaxis C#

En el capítulo anterior, creamos un archivo en C# llamado Program.cs, y usamos el siguiente código para imprimir "Hello World" en la pantalla:

Program.cs

```
using System;


namespace HelloWorld
{
  class Program
  {
    static void Main(string[] args)
    {
      Console.WriteLine("Hello World!");
    }
  }
}
```

Resultado:

```
Hello World!
```

Ejemplo explicado

Línea 1: significa que podemos usar clases del espacio de nombres.using SystemSystem

Línea 2: Una línea en blanco. C# ignora los espacios en blanco. Sin embargo, varias líneas hacen que el código sea más legible.

Línea 3: se usa para organizar tu código, y es un contenedor para clases y otros espacios de nombres.namespace

Línea 4: Los brackets rizados marcan el principio y el final de un bloque de código.{}

Línea 5: es un contenedor para datos y métodos, que aporta funcionalidad a tu programa. Cada línea de código que se ejecuta en C# debe estar dentro de una clase. En nuestro ejemplo, llamamos al Programa de la clase.class

No te preocupes si no entiendes cómo funciona. Solo piensa en ello como algo que (casi) siempre aparece en tu programa, y que aprenderás más sobre ellos en un capítulo posterior.using Systemnamespaceclass

Línea 7: Otra cosa que siempre aparece en un programa de C# es El método. Cualquier código dentro de sus corchetes curlados será ejecutado. No tienes que entender las palabras clave antes y después de Main. Vas a llegar Conocelos poco a poco mientras lees este tutorial.Main{}

Línea 9: es una clase del espacio de nombres, que tiene un método que se utiliza para salir/imprimir texto. En nuestro ejemplo, dará "¡Hola Mundo!". ConsoleSystemWriteLine()

Si omites la línea, tendrías que escribir para imprimir/sacar texto.using SystemSystem.Console.WriteLine()

Nota: Toda afirmación C# termina con punto y coma .;

Nota: C# es sensible a mayúsculas y mayúsculas; "MiClase" y "MiClase" have significado diferente.

Nota: A diferencia de Java, el nombre del archivo C# no tiene que coincidir con el nombre de la clase, pero a menudo sí lo hacen (para una mejor organización). Al guardar el archivo, guárdalo usando un nombre propio y añade ".cs" al final de El nombre del archivo. Para ejecutar el ejemplo anterior en tu ordenador, asegúrate de que C# sea correctamente instalado: Ve al capítulo Empezar para ver cómo instalar C#. La salida debería ser:

```
Hello World!
```

Salida en C#

Para generar valores o imprimir texto en C#, puedes usar el método:WriteLine()

Ejemplo Consigue tu propio servidor C#

```
Console.WriteLine("Hello World!");
```

Puedes añadir tantos métodos como quieras. Ten en cuenta que añadirá una nueva línea para cada método:WriteLine()

Ejemplo

```
Console.WriteLine("Hello World!");

Console.WriteLine("I am Learning C#");

Console.WriteLine("It is awesome!");
```

También puedes generar números y realizar cálculos matemáticos:

Ejemplo

```
Console.WriteLine(3 + 3);
```

El método de escritura

También existe un método, que es similar a .Write()WriteLine()

La única diferencia es que no inserta una nueva línea al final de la salida:

Ejemplo

```
Console.Write("Hello World! ");

Console.Write("I will print on the same line.");
```

Ten en cuenta que añadimos un espacio extra cuando es necesario (después de "¡Hola Mundo!" en el ejemplo anterior), para mejorar la legibilidad.