2. MIS PRIMEROS PROGRAMAS Y OBJETOS

2.1 Mi primer programa. Hola Mundo.

Para seguir la tradición de los libros de programación, veamos el clásico programa de bienvenida con el que comienzan la mayoría de los libros y manuales de programación (Listado 2.1). Este código nos servirá para introducir algunas de las características básicas de C#, así como de su sintaxis y notación.

```
1  namespace WEBOO.Programacion
2  {
3    class PrimerPrograma
4    {
5      static void Main(string[] args)
6      {
7        //Escribir mensaje de saludo
8        System.Console.WriteLine("Bienvenido a la Programación con C#");
9      }
10   }
11 }
12
```

Listado 2.1. Primer Programa

Si se compila y ejecuta este código, se mostrará en la ventana de salida por consola el texto:

```
Hola, Bienvenido a C#
```

Este programa consta de una única **clase** a la que se le ha dado el nombre `PrimerPrograma` (línea 3). Para ello se ha utilizado la palabra reservada `class` seguida del nombre que quiere dar a la clase. Una **palabra reservada** o palabra clave[1] es una palabra con un uso específico dentro del lenguaje y que no puede utilizarse para otro fin, en este ejemplo `static` y `void` son también palabras reservadas. Las palabras reservadas se irán introduciendo a lo largo del texto según se utilicen para expresar los diferentes recursos de programación de C# que se utilizarán. Si usted está escribiendo el código con Visual Studio (como será el caso de los ejemplos que se usarán en este libro), una de las opciones de visualización es mostrar las palabras claves en color azul.

La definición de una clase va encerrada entre las llaves `{` y `}`. Dentro de las llaves va el código de la clase, que en este caso consiste solo en la definición de un método de nombre `Main` (línea 5).

[1] Por el término en inglés *keyword*.

Note que el nombre `Main` está precedido de las palabras `static` y `void`. El significado de la palabra reservada `static` se explicará más adelante. La palabra reservada `void` significa que este método `Main` no devuelve de manera explícita un resultado. La definición de un método (como `Main` en este caso) va seguido de los paréntesis `(` y `)` que agrupan a los "parámetros" del método. Una aplicación que contenga una clase con un método `static` y `Main` significa que la ejecución de la aplicación comenzará por invocar a dicho método. Por convenio, el compilador de C# dejará el resultado de la compilación de la tal clase en un fichero con extensión `.exe`.

A su vez, el código de la implementación del método (lo que se conoce también como *cuerpo del método*) va también encerrado entre las dos llaves `{` y `}`. La línea 7

```
//Escribir mensaje de saludo
```

es lo que se denomina un *comentario* (Visual Studio los suele destacar en color verde). Todo texto que comience con un par de diagonales (*slash*) `//` se considera un comentario. Un comentario es un texto para ser leído por los programadores dando información sobre el código que lo incluye pero que no tiene incidencia directa con el funcionamiento del programa, ya que el compilador de C# no lo traduce a ninguna instrucción ejecutable.

Un comentario que comienza con `//` termina en el final de la línea. De modo que

```
//Si esto pretende ser un comentario será un error ya que el compilador de C#
interpreta esta segunda línea como una instrucción de C#.
```

Si un comentario ocupa más de una línea puede escribirse

```
// Este comentario ocupa
// dos líneas
```

Hay una forma más cómoda para expresar comentarios de más de una línea. Se empieza el comentario con los símbolos `/*` y se termina con `*/`

```
/* Este comentario ocupa más de una línea y
   es correcto */
```

La siguiente línea del método `Main` es la instrucción

```
System.Console.WriteLine("Bienvenido a la Programación con C#");
```

Esta instrucción es una llamada al método `WriteLine`, que es un método estático definido dentro de la clase `Console` que está en el espacio de nombres `System`. Para dar mayor claridad y legibilidad a la hora de lidiar con gran cantidad de nombres, .NET permite agrupar las clases en lo que se denomina un **espacio de nombres** (en inglés **namespace**); en este ejemplo se está usando el espacio de nombres `System`, que es un espacio de nombres que forma parte de la biblioteca BCL ofrecida por el marco de trabajo (*framework*) de .NET. En este espacio de nombres se agrupan muchas clases de utilidad general para la programación, como es en este ejemplo el caso de la clase `Console`. La clase `Console` ofrece funcionalidades para hacer entrada y salida de texto a través de la ventana de consola; esta clase incluye entre sus métodos al método `WriteLine`, que es el que se ha invocado aquí para escribir el mensaje de bienvenida.

C# utiliza la secuencia de nombres separados por . (punto) para indicar el camino para llegar al recurso deseado. De modo que

```
System.Console.WriteLine
```

significa que nos estamos refiriendo al método `WriteLine` que esté en la clase `Console` que a su vez está en el espacio de nombres `System`.

En la llamada al método `WriteLine` se han encerrado entre paréntesis los parámetros (argumentos, si usamos la jerga matemática) que se le pasan al método. En este ejemplo se le está pasando como parámetro la cadena de caracteres

`"Bienvenido a la Programación con C#"`. Esta cadena de caracteres es un valor del tipo `string` (el tipo `string` se estudia en más detalle en el Capítulo 3), que es el tipo de parámetro que hay que pasar al método `WriteLine`. Toda secuencia de caracteres que se quiera indicar como un valor de tipo `string` debe encerrarse entre comillas dobles `"` y `"` (las comillas no forman parte del texto, solo sirven para que C# distinga, del resto del código, donde comienza y dónde termina la cadena).

C# es *case-sensitive*, es decir, es sensible a la diferencia entre mayúsculas y minúsculas. Esto significa que `WriteLine`, que en este caso es el nombre correcto del método, no es lo mismo que `writeLine` o `WRITEline`.

Cuando haga falta para dar mejor idea del significado de un nombre, es un buen estilo usar nombres compuestos, como es el caso de `WriteLine` Para facilita su legibilidad se suele empezar con mayúscula cada palabra que forme parte del nombre de un método. Así se ha hecho también con el nombre de la clase `MiPrimerPrograma`.

2.2 Mi segundo programa. Personalizar la bienvenida

Para personalizar el saludo de bienvenida, vamos a pedir al usuario que teclee su nombre. El programa deberá leer lo que se teclee como nombre y luego dar el mensaje de bienvenida incluyendo ese nombre (Listado 2.2).

```
1   using System;
2   namespace WEBOO.Programacion
3   {
4     class Program
5     {
6       static void Main(string[] args)
7       {
8         Console.WriteLine("Hola, teclea tu nombre por favor");
9         string nombre = Console.ReadLine();
10        Console.WriteLine("Hola " + nombre +
11            ". Bienvenido a la Programación con C#");
12      }
13    }
14  }
```

Listado 2.2. Segundo programa. Bienvenida personalizada

En este código se ha añadido el comando `using System` (línea 1); esto indica que en el código se va a usar el espacio de nombres `System` y que por tanto no hay que preceder

con la palabra `System` cuando nos referimos a un recurso de este espacio de nombres. De modo que se puede escribir (línea 8):

```
Console.WriteLine("Hola, teclea tu nombre por favor");
```

En la instrucción

```
string nombre = Console.ReadLine();
```

se está llamando a otro método de la clase `Console`, el método `ReadLine`. Este método "lee" de la consola y devuelve como valor la cadena de caracteres que se haya tecleado hasta pulsar la tecla *Enter*.

La cadena (valor de tipo `string`) leída se ha guardado en este caso como valor de la variable `nombre`, que se ha declarado como de tipo `string` (línea 9).

En la instrucción de las líneas 10 y 11:

```
Console.WriteLine("Hola "+ nombre + ". Bienvenido a la Programación con C#");
```

se escribe el mensaje de bienvenida. Note que se ha utilizado aquí el operador "+" para **concatenar** la cadena `"Hola "` con la cadena valor de la variable `nombre` al hacer

```
"Hola " + nombre
```

y que a su vez a esto se le concatena el resto del saludo

```
"Hola " + nombre + ". Bienvenido a la Programación con C#"
```

Esto provocará la siguiente salida por la ventana de consola

```
Hola, teclea tu nombre por favor
Miguel
    Hola Miguel, bienvenido a la Programación con C#
```

La clase `Console` y los métodos `WriteLine` y `ReadLine` se utilizarán a lo largo de este texto en varios ejemplos para hacer entrada y salida de datos en forma de texto.

En las aplicaciones actuales, la entrada y salida de información es mucho más variada que en forma de *texto simple*. Por razones de simplicidad, por ahora se seguirá usando `Console` con sus métodos `WriteLine` y `ReadLine`.

2.3 Tercer programa. Usando los primeros objetos

Vamos a complicar un poco más el código del Listado 2.2 para poder escribir también como resultado el tiempo que el usuario se ha demorado en teclear el nombre.

En el espacio de nombres `System.Diagnostics` se dispone de la clase `Stopwatch`. Esta clase nos ofrece una funcionalidad que simula el funcionamiento de un cronómetro (Listado 2.3).

```
1  using System;
2  using System.Diagnostics;
3  class Program
4  {
5    static void Main(string[] args)
6    {
7
```

```
8       //Crear un cronómetro
9       Stopwatch crono = new Stopwatch();
10      Console.WriteLine("Hola, teclea tu nombre por favor");
11      //Iniciar el cronómetro
12      crono.Start();
13      string nombre = Console.ReadLine();
14      //Parar el cronómetro
15      crono.Stop();
16      Console.WriteLine("Hola " + nombre +
17          ". Bienvenido a la Programación con C#");
18      //Usar el tiempo que ha medido el cronómetro
19      Console.WriteLine("Te has demorado " + crono.ElapsedMilliseconds +
20                        " ms en teclear tu nombre");
21    }
    }
```

Listado 2.3. Usando los primeros objetos

En la instrucción de la línea 8:

```
Stopwatch crono = new Stopwatch();
```

se **crea un objeto** (también se dice se **crea una instancia**) de tipo `Stopwatch` el cual se asigna como valor a la variable `crono`.

La clase `Stopwatch` tiene varios métodos y propiedades que se pueden invocar a través de los objetos del tipo definido por esa clase. Uno de ellos es el método `Start`, de modo que cuando se hace `crono.Start()` (línea 11) este "cronómetro" empezará a medir el tiempo, en este caso justo antes de la línea 12, que es la que precisamente hace la lectura de lo que el usuario debe teclear.

Esta forma de utilizar los recursos de la clase a través de un objeto o instancia de la clase es una de las características fundamentales del método orientado a objetos y se le conoce como **notación punto** (*dot notation*) y tiene la forma:

<nombre del objeto>.*<nombre del método>*(*<parámetros del método>*)

como en este ejemplo cuando se escribe

```
crono.Start()
```

Después de ejecutar la instrucción

```
string nombre = Console.ReadLine();
```

que lee el nombre tecleado por el usuario, entonces se debe "parar" el cronómetro, lo que se logra invocando al método `crono.Stop();` (línea 14).

El tiempo que este cronómetro estuvo andando hasta que se mandó a parar es un valor de tipo numérico, funcionalidad que el tipo `Stopwatch` ofrece mediante la propiedad `ElapsedMilliseconds`, que como su nombre indica nos da un valor entero con la cantidad de milisegundos transcurridos. Como se explica en el capítulo siguiente, hay varios tipos numéricos dentro del conjunto de los tipos básicos de .NET y en el lenguaje C#. Este valor se utiliza en la línea 18 en la que se manda a escribir entonces la cantidad de milisegundos transcurridos.

La ejecución de este código daría finalmente como resultado

```
Hola, teclea tu nombre por favor
miguel
Hola miguel, bienvenido a la Programación con C#
Te has demorado 4328 ms en teclear tu nombre
```

Claro que el lector puede decir con razón que esto no garantiza que realmente se esté tecleando correctamente el nombre. En el Capítulo 4, que trata las instrucciones condicionales, veremos cómo se puede mejorar esta implementación para dar mayor seguridad de que se ha tecleado lo esperado.