ATRIBUTOS ESTÁTICOS (Ámbito a nivel de Clase) y MÉTODOS ESTÁTICOS

La definición formal de los elementos estáticos (o miembros de clase) nos dice que son aquellos que pertenecen a la clase, en lugar de pertenecer a un objeto en particular. Recuperando conceptos básicos de orientación a objetos, sabemos que tenemos:

- Clases: definiciones de elementos de un tipo homogéneo.
- Objetos: concreción de un ejemplar de una clase.

En las clases defines que tal objeto tendrá tales atributos y tales métodos, sin embargo, para acceder a ellos o darles valores necesitas construir objetos de esa clase. Una casa tendrá un número de puertas para entrar, en la clase tendrás definida que una de las características de la casa es el número de puertas, pero solo concretarás ese número cuando construyas objetos de la clase casa. Un coche tiene un color, pero en la clase solo dices que existirá un color y hasta que no construyas coches no les asignarás un color en concreto. En la clase cuadrado definirás que el cálculo del área es el "lado elevado a dos", pero para calcular el área de un cuadrado necesitas tener un objeto de esa clase y pedirle que te devuelva su área.

Ese es el comportamiento normal de los miembros de clase. Sin embargo, los métodos estáticos son un poco distintos. Son elementos que existen dentro de la propia clase y para accederlos no necesitamos haber creado ningún objeto de esa clase. Ósea, en vez de acceder a través de un objeto, podemos acceder a través del nombre de la clase.

Un atributo estático es aquel es igual en todas las clases, es decir, que aunque creemos distintos objetos en todos será igual.

Se definen utilizando el modificador **static** en la definición del método o del atributo (después del acceso -que puede ser público o privado-).

Para acceder a un método o atributo estático (también depende de su visibilidad), no es necesario crear un objeto, para ello escribimos **nombre_clase.metodo_estatico(parámetros);** otro ejemplo: **Empleado.setSalarioBase(salario);**

Debemos tener en cuenta que, si un método es estático, los atributos que estén dentro del método deben ser estáticos también, si hay alguno que no lo es, se producirá un error.

De momento así dicho queda un tanto abstracto. Pero antes de analizar un ejemplo concreto de programación, tratemos de explicar estos conceptos con situaciones de la vida real.

Por ejemplo, pensemos en los autobuses de tu ciudad. No sé si es el caso, pero generalmente en España todos los autobuses metropolitanos tienen la misma tarifa. Yo podría definir como un atributo de la clase AutobúsMetropolitano su precio. En condiciones normales, para acceder al precio de un autobús necesitaría instanciar un objeto autobús y luego consultar su precio. ¿Es esto práctico? quizás solo quiero saber su precio para salir de casa con dinero suficiente para pagarlo, pero en el caso de un atributo normal necesariamente debería tener instanciado un autobús para preguntar su precio.

Pensemos en el número "Pi". Sabemos que necesitamos ese número para realizar cálculos con circunferencias. Podría tener la clase Circunferencia y definir como atributo el número Pi. Sin embargo, igual necesito ese número para otra cosa, como pasar ángulos de valores de grados a radianes. En ese caso, en condiciones normales sin atributos de clase, necesitaría instanciar cualquier círculo para luego preguntarle por el valor de "Pi". De nuevo, no parece muy práctico.

Tenemos dos ejemplos de situaciones en las que me pueden venir bien tener atributos "static", o de clase, porque me permitirían consultar esos datos sin necesidad de tener una instancia de un objeto de esa clase.

En cuanto a métodos, pensemos por ejemplo en la clase Fecha. Puedo intentar construir fechas con un día, un mes y un año, pero puede que no necesite una fecha en un momento dado y solo quiera saber si una fecha podría ser válida. En situaciones normales debería intentar construir esa fecha y esperar a ver si el constructor me arroja un error o si la fecha que construye es válida. Quizás sería más cómodo tener un método vinculado a la clase, en el que podría pasarle un mes, un día y un año y que me diga si son válidos o no, sin necesidad de instanciar un objeto de la clase Fecha.

Vamos a ver un ejemplo con programación. Tenemos una clase Empleado con un atributo salarioBase que es **estático:**

Ejemplo de uso:

Clase Program

```
using System;
using System.Collections.Generic;

namespace Atribstatic
{
    class Program
    {
        static void Main(string[] args)
        {
       // Creamos una lista de 3 empleados. Salario base por defecto es 890.
            List<Empleado> listaEmpleado = new List<Empleado>();
            for (int i = 0; i < 3; i++)
            {
                listaEmpleado.Add(new Empleado());
            }
       // Visualizamos el salario base de cada empleado
            foreach (Empleado emp in listaEmpleado)
            {
                Console.WriteLine("El salario base de {0} es {1}", emp.Nombre, emp.SalarioBase);
            }
       // Modifico el salario base de un empleado.
       // Al ser un atributo static (ámbito de clase) lo modifico para todos
       // los objetos de la clase (utilizo su propiedad).
            listaEmpleado[0].SalarioBase = 1200;

       // Para acceder a un método o atributo estático (también depende de su visibilidad),
       // no es necesario crear un objeto, escribimos nombre_clase.metodo_estatico(parámetros);
       // Piensa, por ejemplo cómo accedemos al método WriteLine de la Clase Console...
       // Efectivamente WriteLine es método static
       // En nuestro ejemplo podríamos hacer:
            Empleado.setSalarioBase(1400);   // el método SetSalarioBase está definido como static.

            Console.WriteLine("\nNuevo salario Base: {0}", listaEmpleado[0].SalarioBase);

       // Vuelvo a visualizar el salario base de cada empleado y podemos
       // ver que todos los empleados tienen el mismo salario base (ha cambiado para todos).
            foreach (Empleado emp in listaEmpleado)
            {
                emp.VerDatosEmpleado();
            }

       // Aumento en una cantidad (pedida por teclado) el salarioBase de los empleados.
            Console.Write("\nVamos a aumentar el SalarioBase de los empleados. Cantidad a incr.: ");
            double cantAumento = double.Parse(Console.ReadLine());
       // Invoco al método AumentoPlus utilizando la clase Empleado (y no un objeto).
       // No utilizo el retorno: nuevo salarioBase
            Empleado.AumentoPlus(cantAumento);

       // Vuelvo a visualizar el salario base de cada empleado y podemos
       // ver que todos los empleados tienen el mismo salario base (ha cambiado para todos).
            foreach (Empleado emp in listaEmpleado)
            {
                emp.VerDatosEmpleado();
            }
            Console.ReadKey();
        }
    }
}
```

Clase Empleado

```
using System;

namespace Atribstatic
{
```

```
    class Empleado
    {
        // Atributos

        private string nombre;                  // nombre de empleado
        private string apellido;                // apellido del empleado
        private int edad;                       // edad del empleado
        private static double salarioBase = 890; // salario base (atributo static -ámbito de clase-)

       // Cuando creamos un objeto el salarioBase es de 890.
       // El salarioBase afectará a todos los objetos. Tiene ámbito de Clase.
       // Si cambiamos el salario base afectará a todos los objetos de la clase.

        // Propiedades para acceso a atributos
        public string Nombre { get { return nombre; } set { nombre = value; } }
        public string Apellido { get { return apellido; } set { apellido = value; } }
        public int Edad { get { return Edad; } set { edad = value; } }
        public double SalarioBase { get { return salarioBase; } set { salarioBase = value; } }

       // método de acceso al atributo salarioBase (es complementario a la propiedad Salario)
       // Es un método estático para poder ser llamado desde la Clase Empleado (directamente).
       // Cuando se asigna un salario base se cambia para todos los empleados que se hayan
       // instanciado de la clase.
        public static void setSalarioBase(double sbase)
        {
            salarioBase = sbase;
        }

        // Constructores
        public Empleado(string pnom, string papell, int pedad)
        {
            this.nombre = pnom;
            this.apellido = papell;
            this.edad = pedad;
        }

        public Empleado()
        {
            Console.Write("\nNombre del empleado: ");
            this.nombre = Console.ReadLine();
            Console.Write("Apellido del empleado: ");
            this.apellido = Console.ReadLine();
            Console.Write("Edad del empleado: ");
            this.edad = int.Parse(Console.ReadLine());
        }

       // métodos públicos

        // Suma un plus al salario base y devuelve el nuevo salarioBase.
        // Hacemos el Método estático de tal forma que no sea necesario instanciar
        // un objeto para incrementar el salario base de todos los empleados.
        public static double AumentoPlus (double sueldoPlus)
        {
            salarioBase += sueldoPlus;
            return salarioBase;
        }

        public void VerDatosEmpleado ()
        {
            Console.Write("\nEmpleado: {0} {1}",nombre, apellido);
            Console.Write("\nEdad: {0} años.", edad);
            Console.Write("\nSalario base: {0:N} euros", salarioBase);
            Console.ReadKey();
        }

    }
}
```