Clases estáticas.

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 el operador **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 `CheckClass` que tiene un método estático público denominado `MethodCheckDni`, llame al método tal como se muestra en el ejemplo siguiente:

```
        CheckClass.MethodCheckDni();
```

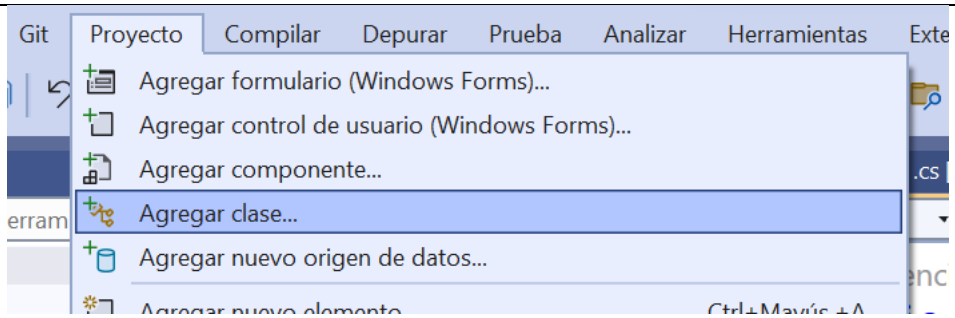
Es posible usar una clase estática **como un contenedor adecuado para conjuntos de métodos** que solo funcionan en parámetros de entrada y que no tienen que obtener ni establecer campos de instancias internas.

Por ejemplo, en la biblioteca de clases .NET, 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));           // 3.14
Console.WriteLine(Math.Floor(dub));         // -4
Console.WriteLine(Math.Round(Math.Abs(dub)));// 3
```

Para entender cómo se crearía una clase estática vamos a hacerlo creándola y añadiendo a la misma algunas de las funciones que hemos creado en nuestros ejercicios anteriores.

En primer lugar, debemos crear una nueva clase tal y como vimos en los apartados anteriores:



En la clase que se crea debemos poner el término static, así como a las distintas funciones que tengamos:

```
static class Funciones
{
    private static int numMenor(int n1, int n2)
    {
        int menor;

        if (n1 < n2)
            menor = n1;
        else
            menor = n2;

        return menor;
    }

    // Función que obtiene el máximo común divisor.
    // Empezamos desde 1 y vamos subiendo hasta el número menor de los dos
    // Nos quedamos con el último número que divide a ambos.
    public static int calcularMCD(int n1, int n2)
    {
        int mcd = 1, menor;

        menor = numMenor(n1, n2);

        for (int i = 1; i <= menor; i++)
        {
            // Si el número es divisor de ambos pasa a ser el MCD
            if (n1 % i == 0 && n2 % i == 0)
                mcd = i;
        }

        return mcd;
    }
```

```
    public static int potencia(int bbase, int exp)
    {
        int res = 1;

        for(int i = 1; i <= exp; i++)
            res *= bbase;

        return res;
    }

}
```

Después, podremos utilizar esas funciones llamándolas con: clase.nombrefuncion:

```
        private void btnMCD_Click(object sender, EventArgs e)
        {
            int num1, num2;
            int mcd;

            num1 = int.Parse(txtNum1.Text);
            num2 = int.Parse(txtNum2.Text);

            // Llamamos a la función
            mcd = Funciones.calcularMCD(num1, num2);

            MessageBox.Show("El MCD es : " + mcd);
        }

        private void btnPotencia_Click(object sender, EventArgs e)
        {
            int num1, num2;
            int pot;

            num1 = int.Parse(txtNum1.Text);
            num2 = int.Parse(txtNum2.Text);

            pot = Funciones.potencia(num1, num2);

            MessageBox.Show("La potencia: " + pot);
        }
```

Una de las ventajas de tener las funciones de este modo, es que, si tuviéramos distintos formularios, podríamos utilizarlas en ellos, y no solo en el formulario en el que estén declaradas…