

# Classes sealed

Una classe sealed és una classe de la que no es poden tenir subclasses. Per dues raons; per seguretat i per disseny.

```
sealed class nombre_clase
```

```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace ProyectoClaseFinal
```

```
{
```

2 referencias

```
class ClsRectangulo
```

```
:
```

```
{
```

```
    private int _Lado1;
```

```
    private int _Lado2;
```

1 referencia

```
    public ClsRectangulo(int l1,int l2) { Lado1 = l1;Lado2 = l2; }
```

2 referencias

```
    public int Lado1 { get => _Lado1; set => _Lado1 = value; }
```

2 referencias

```
    public int Lado2 { get => _Lado2; set => _Lado2 = value; }
```

```
}
```

```
}
```

ClsRectangulo



```
Final ProyectoClaseFinal.ClsFinalCubo ClsFi

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ProyectoClaseFinal
{
    4 referencias
    sealed class ClsFinalCubo : ClsRectangulo
    {
        1 referencia
        public ClsFinalCubo(int m1, int m2) : base(m1, m2) { }
    }
}
```

ClsfinalCuadrado  
sealed

La classe Cuadrado declarada com “final” sealed, no s’espera que es necessiti crear classes derivades.

```
ClsHeredaCubo.cs  ClsFinalCubo.cs  ClsRectangulo.cs  Program.cs
C# ProyectoClaseFinal  ProyectoClaseFinal.ClsHeredaCu

1  using System;
2      using System.Collections.Generic;
3      using System.Linq;
4      using System.Text;
5      using System.Threading.Tasks;
6
7  namespace ProyectoClaseFinal
8  {
9      class ClsHeredaCubo: ClsFinalCubo
10     {
11     }
12 }
13
```

Error si es crea classe derivada

# Mètodes i propietats estàtics



Declarar propietats o mètodes de classes com estàtics els fan accessibles sense la necessitat d'instanciar la classe.

Una variable(propietat) estàtica és compartida per tots els objectes d'aquesta classe.

Per tant per accedir a ella s'ha de fer des de la classe des de l' objecte:  
Empleados.IdSiguiente

Per exemple(Java o JS):

La constant PI de la classe Math:

NumeroPi=Math.PI; o Math.Pi(c#)

# Mètodes estàtics de classe

Un mètode d'instància és el que s'invoca **sempre** sobre una instància (objecte) d'una classe.

Per exemple : `persona1.Nombre()`;  
amb `persona1` objecte de classe `Persona`

És un mètode d'instància: per invocar-lo necessitem una instància de persona.

Un método de classe es aquell que pot invocat sense existir una instància.

Exemple de definició:

```
public static tipus Nom () { ...
```

Per invocar: `nomClasse.nomMetodoEstatic`



```
var x = Math.PI;           // Returns PI
var y = Math.sqrt(16);     // Returns the square root of 16
```

For a tutorial about the Math object, read our [JavaScript Math Tutorial](#).

## Math Object Properties

Property	Description
<u>E</u>	Returns Euler's number (approx. 2.718)
<u>LN2</u>	Returns the natural logarithm of 2 (approx. 0.693)
<u>LN10</u>	Returns the natural logarithm of 10 (approx. 2.302)
<u>LOG2E</u>	Returns the base-2 logarithm of E (approx. 1.442)
<u>LOG10E</u>	Returns the base-10 logarithm of E (approx. 0.434)
<u>PI</u>	Returns PI (approx. 3.14)
<u>SQRT1_2</u>	Returns the square root of 1/2 (approx. 0.707)

Mètodes de la classe Math: són tots estàtics.

JS

## Math Object Methods

Method	Description
<code>abs(x)</code>	Returns the absolute value of x
<code>acos(x)</code>	Returns the arccosine of x, in radians
<code>acosh(x)</code>	Returns the hyperbolic arccosine of x
<code>asin(x)</code>	Returns the arcsine of x, in radians
<code>asinh(x)</code>	Returns the hyperbolic arcsine of x

`Math.Sqrt(x);`

```
Math.sqrt(9);
```

# Característiques principals d'una classe estàtica

- Només conté membres estàtics.
- No es pot crear instàncies d'ella.
- Està 'sellada'.
- No pot contenir constructors d'instàncies.

<https://docs.microsoft.com/es-es/dotnet/csharp/programming-guide/classes-and-structs/static-classes-and-static-class-members>



# ClsTemperatura

```
namespace ProyectoStatic
{
    2 referencias
    public static class ClsTemperatura
    {
        1 referencia
        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;
        }

        1 referencia
        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;
        }
    }
}
```

# Main()

```
Console.WriteLine("Qué conversión desea?");
Console.WriteLine("1. De Celsius a Fahrenheit.");
Console.WriteLine("2. De Fahrenheit a Celsius.");
Console.Write(":");

string selection = Console.ReadLine();
double F, C = 0.0; //0

switch (selection)
{
    case "1":
        Console.Write("Entre la temperatura Celsius: ");
        F = ClsTemperatura.CelsiusToFahrenheit(Console.ReadLine());
        Console.WriteLine("Temperature in Fahrenheit: {0:F2}", F);
        break;
    case "2":
        Console.Write("Entre la temperatura Fahrenheit: ");
        C = ClsTemperatura.FahrenheitToCelsius(Console.ReadLine());
        Console.WriteLine("Temperature in Celsius: {0:F2}", C);
        break;
    default:
        Console.WriteLine("Seleccione un convertidor...");
        break;
}

Console.WriteLine("Cualquier tecla para salir... ");
Console.ReadKey();
```