Modificadores de acceso

En este tutorial veremos todo lo referente a los modificadores de acceso.

Índice

1 - Qué es un modificador de acceso

Un modificador de acceso es aquella cláusula de código que nos indica si podemos acceder o no a un bloque de código específico desde otra parte del programa.

Hay una gran variedad de modificadores de acceso, y estos son aplicados tanto a métodos, propiedades, o clases.

2 - Modificadores de acceso

public

Acceso no restringido que permite acceder a sus miembros desde cualquier parte del código al que se le hace referencia.

```
public class EjemploPublic
{
    public string PruebaAcceso { get; set; }
}

class Program
{
    static void Main(string[] args)
    {
        EjemploPublic ejemplo = new EjemploPublic();
        Console.WriteLine(ejemplo.PruebaAcceso); //Funciona correctamente
    }
}
```

Private

Permite acceder a los miembros exclusivamente desde la clase o struct que los contiene.

```
public class EjemploPrivate
{
    private string PruebaAcceso { get; set; }

    public EjemploPrivate(){
        PruebaAcceso = "funciona";//Funciona
    }
}
class Program
{
    static void Main(string[] args)
    {
        EjemploPrivate ejemplo = new EjemploPrivate();
        Console.WriteLine(ejemplo.PruebaAcceso); //Da un error
    }
}
```

internal

Permite acceder desde el mismo proyecto o asembly pero no desde uno externo.

Por ejemplo, si tenemos una librería, podremos acceder a los elementos internal desde la propia librería, pero si referenciamos a esa librería desde otro proyecto no podremos acceder a ellos.

```
//Libreria externa (Distinto proyecto|Asembly)
public class EjemploInternalLibreria
{
    internal string PruebaAcceso { get; set; }
}

class EjemploImprimirInternal
{
    public void Imprimir()
    {
        EjemploInternalLibreria ejemplo = new EjemploInternalLibreria();
        Console.WriteLine(ejemplo.PruebaAcceso); //Funciona
    }
}

//proyecto principal
class Program
{
    void Imprimir()
    {
        EjemploInternalLibreria ejemplo = new EjemploInternalLibreria();
        Console.WriteLine(ejemplo.PruebaAcceso); // Error. Al estar en otro proyec
```

```
    }
}
```

protected

Podremos acceder a los elementos desde la misma clase, o desde una que deriva de ella.

```
class EjemploProtected
{
    protected string PruebaAcceso { get; set; }
}

class claseHerencia : EjemploProtected //Herencia, osea clase hija.
{
    void Imprimir()
    {
        Console.WriteLine(PruebaAcceso); //Accedemos sin problemas ya que es una pr
    }
}

class Program
{
    void Print()
    {
        EjemploProtected ejemplo = new EjemploProtected();
        Console.WriteLine(ejemplo.PruebaAcceso); // Error. no podemos acceder ya qu
    }
}
```

protected internal

Combina tanto protected como internal permitiendo acceder desde el mismo proyecto o assembly o de los tipos que lo derivan.

private protected

Finalmente combinamos private y protected lo que nos permitirá acceder desde la clase actual o desde las que derivan de ella. Lo que permite referenciar métodos y propiedades en clases de las cuales heredamos.