

C#.Net

Acceso a campos privados

- Implementar una clase `Cuadrado` con un campo privado de tipo `double` llamado `lado`
- Sin cambiar su modificador de acceso y sin definir constructores ¿Qué mecanismo podríamos utilizar que nos permita asignar el campo `lado` desde fuera de la clase `Cuadrado`?
- Defina el método `SetLado`
- ¿Cómo podríamos obtener su valor desde fuera de la clase `Cuadrado` ?
- Defina los métodos `GetLado` y `GetArea` e implemente el siguiente programa

Acceso a campos privados

```
class Cuadrado{  
    private double lado=0;  
    public void SetLado(double value) {  
        lado=value;  
    }  
    public double GetLado() {  
        return lado;  
    }  
    public double GetArea() {  
        return lado*lado;  
    }  
}
```

Acceso a campos privados

```
class Program
{
    static void Main() {
        Cuadrado c=new Cuadrado();
        c.SetLado(0.5);
        Console.WriteLine("Lado={0} Área={1}",
                           c.GetLado(), c.GetArea());
        Console.ReadKey(true);
    }
}
```

Propiedades

- Una **propiedad** es una mezcla entre el concepto de **campo** y el concepto de **método**.
- Externamente es accedida como si de un campo normal se tratase, pero internamente es posible asociar código a ejecutar en cada asignación o lectura de su valor
- Una propiedad no almacena datos, sino sólo se utiliza como si los almacenase

Propiedades

Sintaxis

```
<tipoPropiedad> <nombrePropiedad>
{
    set
    {
        <códigoEscritura>
    }

    get
    {
        <códigoLectura>
    }
}
```


Propiedades

- Dentro del bloque de código **set** se puede hacer referencia a un parámetro especial del mismo tipo de dato que la propiedad llamado **value**
- Dentro del código **get** se ha de devolver siempre un objeto del tipo de dato de la propiedad.
- Una propiedad que sólo tenga el bloque **get** será una propiedad de **sólo lectura**.
- Una propiedad que sólo tenga el bloque **set** será una propiedad de **sólo escritura**)

Propiedad (Ejemplo)

```
class Cuadrado{  
    private double lado=0;  
    public double Lado{  
        get{  
            return lado;  
        }  
        set{  
            lado=value;  
        }  
    }  
    public double Area{  
        get{  
            return lado*lado;  
        }  
    }  
}
```

Propiedad de
lectura/escritura

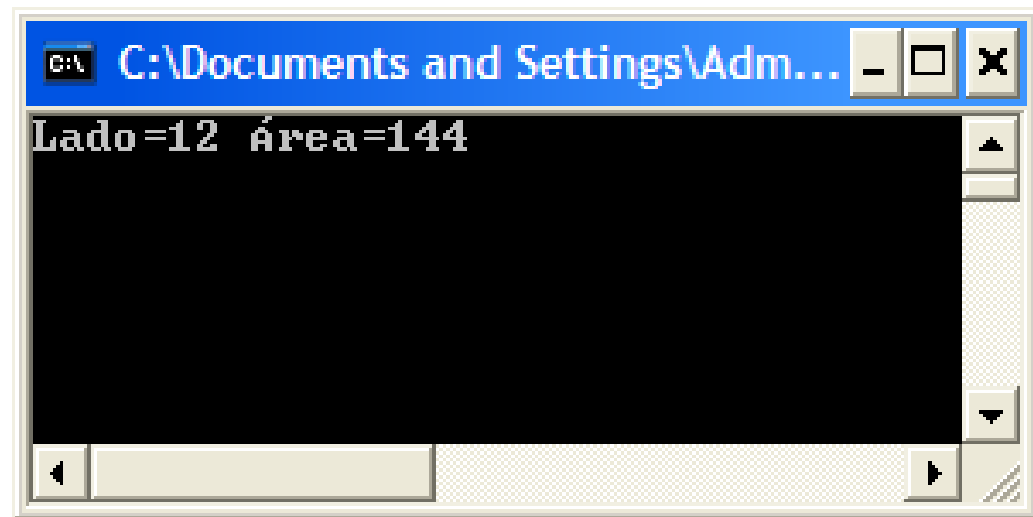


Propiedad de
sólo lectura



Propiedade (Ejemplo)

```
class programa{  
    public static void Main(){  
        Cuadrado c=new Cuadrado();  
        c.Lado=12;  
        Console.WriteLine("Lado={0} Área={1}",  
                           c.Lado,c.Area);  
        Console.ReadKey();  
    }  
}
```



Propiedad (Ejemplo)

- Cambiamos la representación interna de la clase Cuadrado, sin embargo el programa anterior sigue funcionando de la misma manera

```
class Cuadrado{  
    private double area=0;  
    public double Lado{  
        get{  
            return Math.Sqrt(area);  
        }set{  
            area=value*value;  
        }  
    }  
    public double Area{  
        get{  
            return area;  
        }  
    }  
}
```

Indizadores

- Un **indizador** es una definición de cómo aplicar el operador (**[]**) a los objetos de una clase.
- A diferencia de los arreglos, los **índices** que se les pase entre corchetes no están limitados a los enteros, pudiéndose definir varios indizadores en una misma clase siempre y cuando cada uno tome un número o tipo de índices diferente (**sobrecarga**).

Indizadores


Sintaxis

```
<tipoIndizador> this[<índices>]  
{  
    set  
    {  
        <códigoEscritura>  
    }  
    get  
    {  
        <códigoLectura>  
    }  
}
```

- Sintaxis similar a la sintaxis de las propiedades
- El nombre es siempre **this**
- En <índices> se indica cuáles son los índices que se pueden utilizar al acceder al indizador.
- Dentro del bloque **set** se utiliza el parámetro especial **value** del mismo tipo que el indizador

Indizadores - Ejemplo

```
class Auto{  
    private string marca;  
    private int modelo;  
    public string Marca {  
        get{ return marca; }  
    }  
    public Auto(string marca, int modelo) {  
        this.modelo=modelo;  
        this.marca=marca;  
    }  
    public void Imprimir() {  
        Console.WriteLine("{0} {1}",marca,modelo);  
    }  
}
```



Codificar la clase Auto

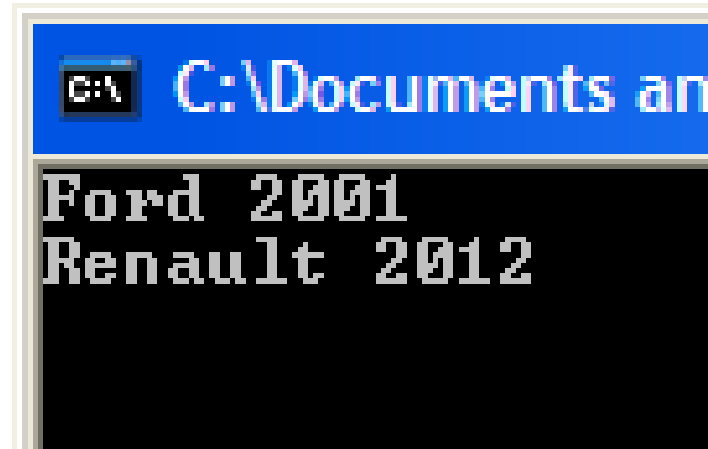
Indizadores - Ejemplo

```
class Estacionamiento{  
    Auto[] vector = new Auto[100];  
    public Auto this[int indice]{  
        get{  
            return vector[indice];  
        }set{  
            vector[indice]=value;  
        }  
    }  
    public Auto this[string marca]{  
        get{  
            foreach (Auto a in vector)  
                if (a!=null && a.Marca==marca)  
                    return a;  
            return null;  
        }  
    }  
}
```

Codificar la clase
Estacionamiento

Indizadores - Ejemplo

```
class programa{  
    public static void Main() {  
        Estacionamiento e=new Estacionamiento();  
        e[4]=new Auto("Ford",2001);  
        e[2]=new Auto("Renault",2012);  
        e[88]=new Auto("Fiat",1998);  
        e[45]=new Auto("Ferrari",2013);  
        e[4].Imprimir();  
        e["Renault"].Imprimir();  
        Console.ReadKey();  
    }  
}
```



Indizadores - Ejemplo

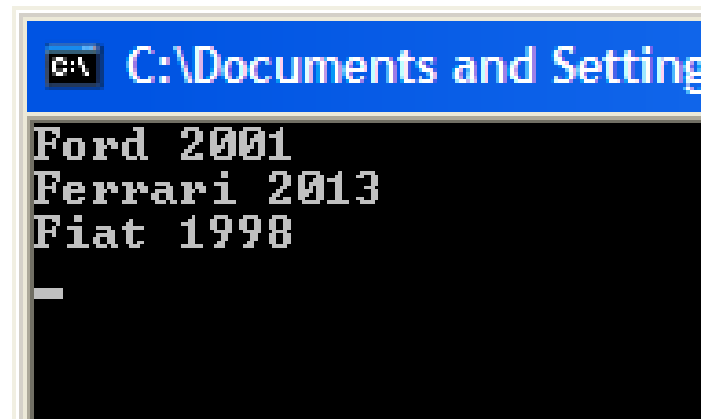
- ¿Qué devuelve el siguiente indizador de la clase Estacionamiento?

```
public ArrayList this[char car] {  
    get {  
        ArrayList aux=new ArrayList();  
        foreach (Auto a in vector) {  
            if (a != null && a.Marca[0] == car)  
                aux.Add(a);  
        }  
        return aux;  
    }  
}
```


Indizadores - Ejemplo

- ¿Qué salida se produce por la consola?

```
class programa{
    public static void Main(){
        Estacionamiento e=new Estacionamiento();
        e[4]=new Auto("Ford",2001);
        e[2]=new Auto("Renault",2012);
        e[88]=new Auto("Fiat",1998);
        e[45]=new Auto("Ferrari",2013);
        ArrayList lista=e['F'];
        foreach (Auto a in lista)
            a.Imprimir();
        Console.ReadKey();
    }
}
```



Miembros estáticos (de la clase)

- Los miembros estáticos son miembros ligados a la clase como tal y no a los objetos (instancias) de la misma.
- Para definirlos basta con preceder la definición de ese miembro con la palabra reservada **static**


```
class ClaseA {
```

```
    public int X;
```


```
    public static int Y;
```

```
}
```

Variable de
instancia



Variable de
clase



Miembros estáticos (de la clase)

- La sintaxis a usar para acceder a un miembro de clase es `<nombreClase>.<miembro>`, como muestra el ejemplo donde se asigna el valor 1 al miembro **Y** de la clase **ClaseA** definida anteriormente:

```
ClaseA.Y = 1;
```

Miembros estáticos (de la clase)

- ¿Qué sentencias son incorrectas?

```
class programa{  
    public static void Main(){  
        ClaseA a= new ClaseA();  
        a.X = 2;  
        ClaseA.Y = 1;  
        a.Y = 3;  
        ClaseA.X = 5;  
        Console.ReadKey();  
    }  
}  
  
class ClaseA {  
    public int X;  
    public static int Y;  
}
```


Y es una variable de clase, no está ligada a la instancia **a**

X es una variable de instancia no puede accederse desde la clase **ClaseA**

Miembros estáticos (de la clase)

- Desde los métodos de clase no se pueden accederse a los miembros que no sean estáticos.

```
class ClaseA {  
    public int X;  
    public static int Y;  
    public static void Incrementa() {  
        X++;  
    }  
}
```



Error: **X** es miembro de instancia e **Incrementa()** es un miembro de clase.

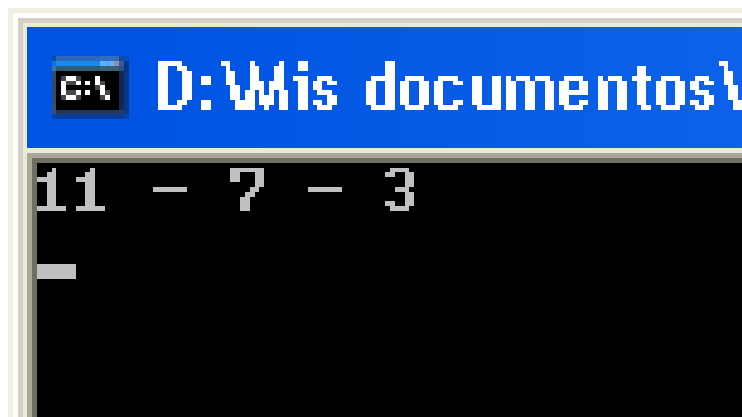
Miembros estáticos (de la clase)

- ¿Cuál será la salida por Consola?

```
using System;
class programa{
    public static void Main(){
        ClaseA a1= new ClaseA(), a2=new ClaseA();
        a1.Incrementa(5);a1.Incrementa(6);
        a2.Incrementa(7);
        Console.WriteLine("{0} - {1} - {2}",
                           a1.X, a2.X, ClaseA.Y);
        Console.ReadKey();
    }
}

class ClaseA {
    public int X=0;
    public static int Y=0;
    public void Incrementa(int valor){
        X += valor;
        ClaseA.Y++;
    }
}
```

La variable de clase **Y** se está utilizando para contar cuántas veces es invocado el método de instancia **Incrementa()** por algún objeto sin importar cuál sea éste.



Miembros estáticos (de la clase)

```
class ClaseA {  
    public int X=0;  
    public static int Y=0;  
    public void Incrementa(int valor) {  
        X += valor;  
        ClaseA.Y++;  
    }  
}
```

Los miembros estáticos pueden accederse desde la propia clase sin anteponer el nombre de la misma. Sin embargo, anteponer el nombre de la clase puede hacer el código más claro y fácil de leer.

```
class ClaseA {  
    public int X=0;  
    public static int Y=0;  
    public void Incrementa(int valor) {  
        X += valor;  
        Y++;  
    }  
}
```