Classes / Mètodes abstractes

Sonia Sánchez Sierra

Imaginem el context:

Classe Profesor de la que hereden ProfesorInterino i ProfesorTitular

Però es dóna la situació que o ets interí o ets titular, amb lo qual mai serà necessari fer instàncies de la classe Profesor.

La superclasse professor té l'objectiu d' unificar camps i mètodes de les subclasses, evitant la repetició de codi i unificant processos.

A aquest tipus de classes se'ls anomena com a classe abstracta.

Per tant: Una classe abstracta és una classe de la que **mai es faran instàncies**: simplement serveixen com a superclasse va a servir com a superclasse a altres classes. No es pot fer servir la paraula clau new aplicada a classes abstractes.

Les classes abstractes :

- Solen contenir mètodes abstractes.
- Poden incloure propietats i mètodes no abstractes.
- No es poden definir constructors abstractes o mètodes estàtics abstractes.

Un mètode o propietat abstracte ha d'incloure en la seva signatura (declaració) la paraula clau abstract.
I compleix les següents característiques:

- Finalitza amb ;
- Els mètodes només posen la signatura public abstract void Encencer();
- Només poden estar dins d'una classe abstracta.
- Forzosament s'ha de sobreescriure (implementar) en les subclasses, sinó la subclasse es tornaria abstracta.

Exemple 1

Classe abstracta

Fer una instància no es pot

ClsClientes.cs | Program.cs*

ProyectoAbstracta | ProyectoAbstracta.ClsClientes | Id

```
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6
7   namespace ProyectoAbstracta
8   {
        2 referencias
9       public abstract class ClsClientes
10      {   //propiedad abstracta
            0 referencias
11          public abstract int Id { get; set; }
            0 referencias
12          public abstract string Nombre { get; set; }
            0 referencias
13          public abstract int Codigo { get; set; }
14      }
15  }
16
```

```
namespace ProyectoAbstracta
{
    0 referencias
    class Program
    {
        0 referencias
        static void Main(string[] args)
        {
            ClsClientes cliente = new ClsClientes();
        }
    }
}
```

Implementació de les propietats per la clase que hereda

```
namespace ProyectoAbstracta
{
    2 referencias
    class ClsBaseClientes:ClsClientes
    {
        private int _Id;
        private string _Nombre;
        private int  _Codigo;


        1 referencia
        public override  int Codigo{ get=>_Codigo; set => _Codigo = value; ; }
        1 referencia
        public override string Nombre  { get => _Nombre; set => _Nombre = value; }
        1 referencia
        public override int Id { get => _Id; set => _Id = value; }
    } }
```

Sí es pot fer una instància

```
namespace ProyectoAbstracta
{
    0 referencias
    class Program
    {
        0 referencias
        static void Main(string[] args)
        {
            ClsBaseClientes cliente = new ClsBaseClientes();

        }
    }
}
```

Incluim constructors

```
ProyectoAbstracta.ClsBaseClientes                    ClsBaseClientes()
    private string _Nombre;
    private int  _Codigo;

    1 referencia
    public ClsBaseClientes()
    {
        Id = 0;
        Nombre = string.Empty;
        Codigo = 0;
    }
    0 referencias
    public ClsBaseClientes(int pId,string pNombre,int pCodigo)
    {
        Id = pId;
        Nombre = pNombre;
        Codigo = pCodigo;
    }
    3 referencias
    public override  int Codigo{ get=>_Codigo; set => _Codigo = value; ;
    3 referencias
```

Exemple:

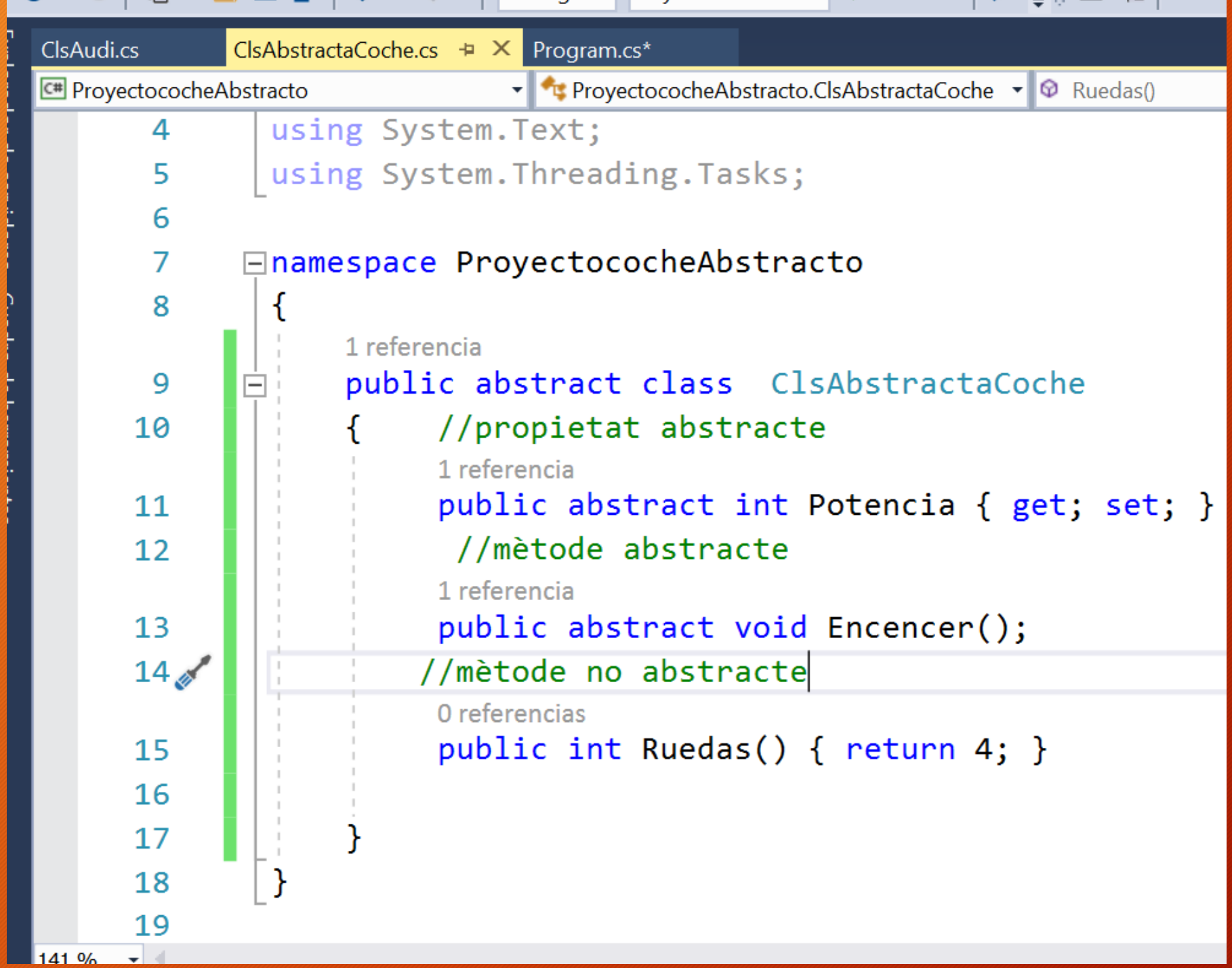
Classe abstracta : ClsAbstractaCoche

Propietat sencera potencia abstracta

Mètode abstracte Encender() que escriurà “Me enciendo!”

Mètode Ruedas no abstracte que retorna el número de rodes (4)

Classe que hereda: ClsAudi



Classe abstracta
ClsAbstractaCoche

ClsAudi.cs | ClsAbstractaCoche.cs | Program.cs

ProyectococheAbstracto | ProyectococheAbstracto.ClsAudi | ClsAudi(int pPotencia)

```
6
7   namespace ProyectococheAbstracto
8   {
        3 referencias
9       class ClsAudi:ClsAbstractaCoche
10      {
11          private int _Potencia;
            3 referencias
12          public override int Potencia { get=>_Potencia; set=>_Potencia=value; }
            1 referencia
13          public override void Encender() { Console.WriteLine("Me enciendo!!"); }
14
            1 referencia
15          public ClsAudi(int pPotencia) {
16              Potencia = pPotencia;
17          }
18
19      }
20  }
21
```

141 %

ClsAbstractaCoche.cs | Program.cs

ococheAbstracto | ProyectococheAbstracto.Program | Main(string[] args)

```
4       using System.Text;
5       using System.Threading.Tasks;
6
7       namespace ProyectococheAbstracto
8       {
            0 referencias
9           class Program
10          {
                0 referencias
11              static void Main(string[] args)
12              {
13                  ClsAudi miAudi = new ClsAudi(500);
14                  Console.WriteLine("Mi Audi tiene..."+miAudi.Ruedas()+" ruedas"+ "y
                      tiene una potencia de:"+miAudi.Potencia);
15              }
16          }
17      }
18
```

Exercici

Classe Abstracta Remolque amb una propietat pública peso=500 i tres mètodes abstractes enganchar() , arrancar() i soltar().

Classe Coche que hereda de Remolque

Crear una instancia de la clase Coche

Solució

ClsRemolque.cs | Program.cs

olqueCoche | ProyectoRemolqueCoche.ClsRemolque

```
using System.Threading.Tasks;

namespace ProyectoRemolqueCoche
{
    0 referencias
    public abstract  class ClsRemolque
    {
        public int peso=500;
        0 referencias
        public abstract void Arrancar();
        0 referencias
        public abstract void  Enganchar();
        0 referencias
        public abstract void Soltar();


    }
}
```

```
namespace ProyectoRemolqueCoche
{
    0 referencias
    class ClsCoche:ClsRemolque
    {
        1 referencia
        public override void Arrancar() { Console.WriteLine("Arrancamos!"); }
        1 referencia
        public override void Enganchar(){Console.WriteLine("Me engancho");}
        1 referencia
        public override void Soltar(){Console.WriteLine("Soltamos!");}
    }
}
```

```
namespace ProyectoRemolqueCoche
{
    0 referencias
    class Program
    {
        0 referencias
        static void Main(string[] args)
        {
            ClsCoche Coche1 = new ClsCoche();
            Coche1.Enganchar();
            Coche1.Arrancar();
            Console.WriteLine("Y peso.."+Coche1.peso);
            Console.ReadKey();



        }
    }
```

Si posem la propietat protected:

```
1 referencia
public abstract  class ClsRemolque
{
    private int peso = 500;

    1 referencia
    protected int Peso { get => peso; set => peso = value; }


    2 referencias
    public abstract void Arrancar();
```

```
1 referencia
public override void Soltar(){Console.WriteL
1 referencia
public int DamePeso() { return Peso; }
```

S'ha de definir un mètode per poder acceder a la propietat

I la fem servir des de la instància

```
Coche1.Arrancar();
Console.WriteLine("Y peso.."+Coche1.DamePeso());
Console.ReadKey();
```