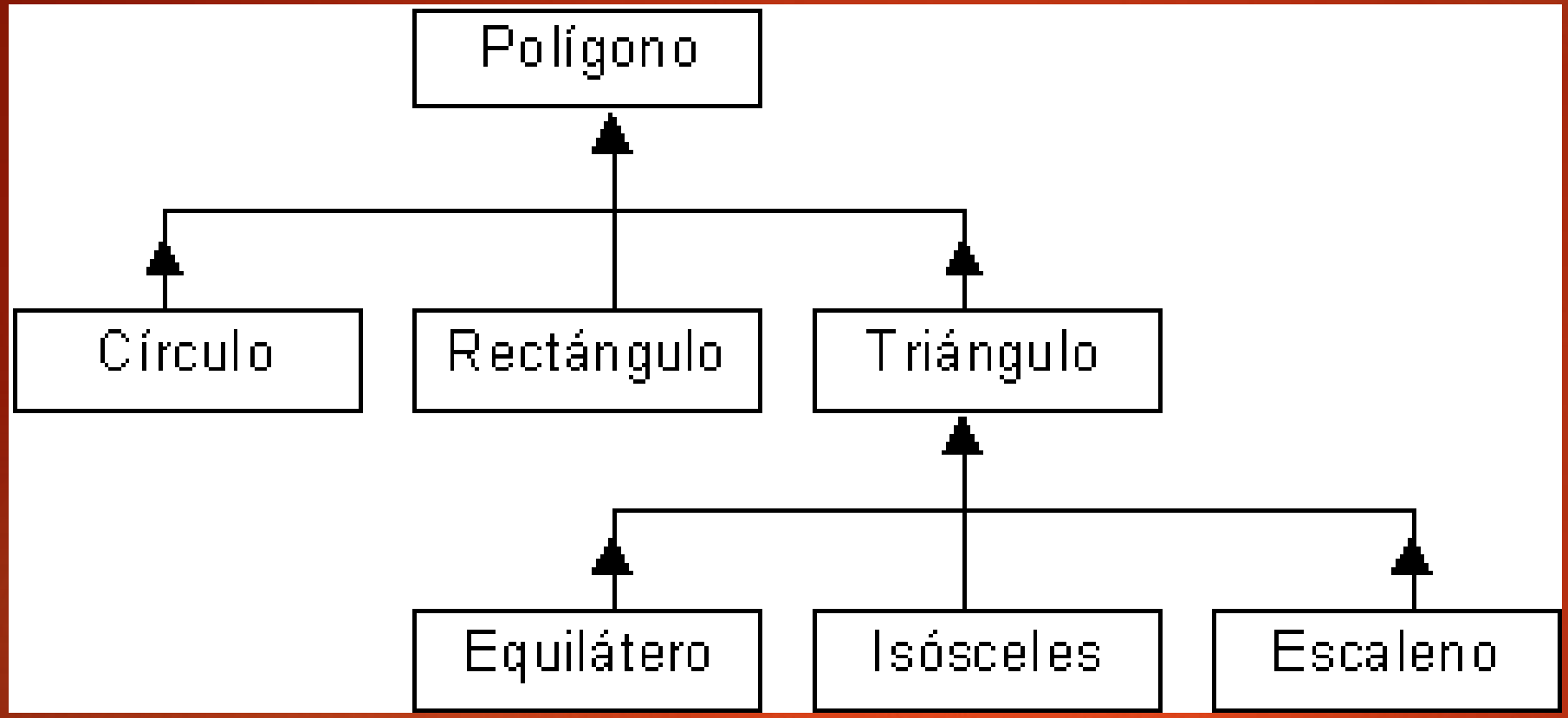
Programació orientada a objectes

Sonia Sánchez Sierra

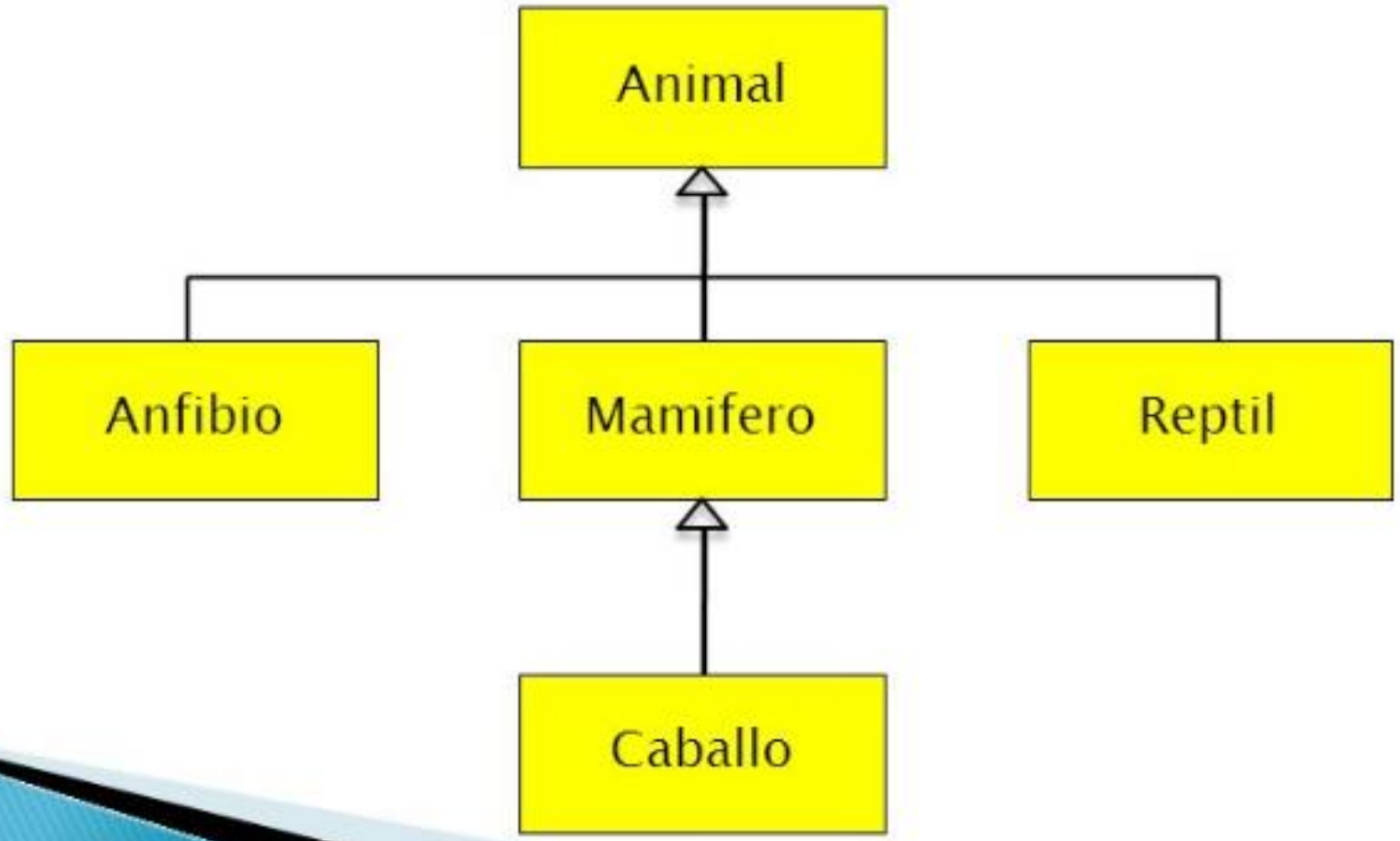
Herència

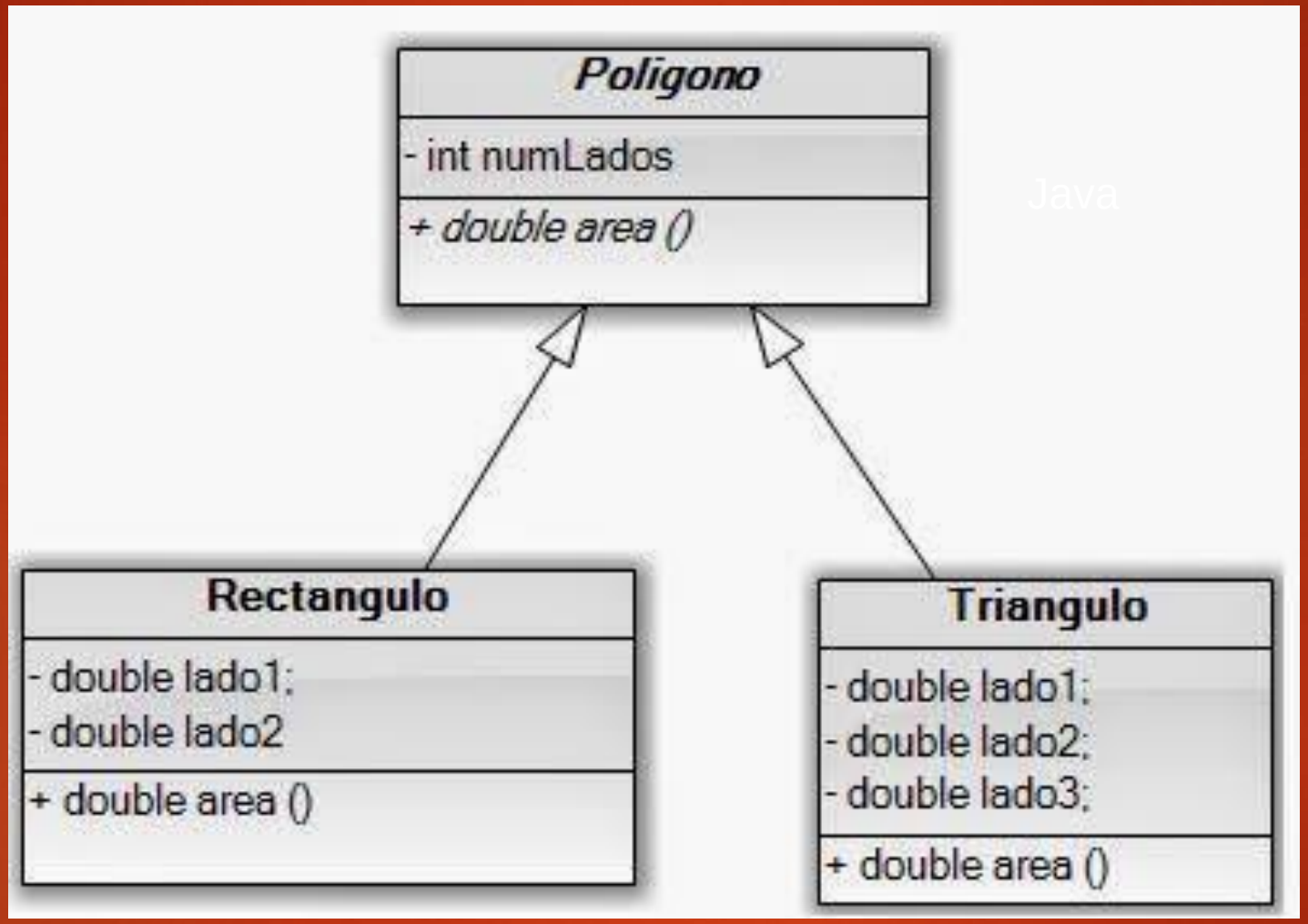
La herència és el concepte que permet que es poden definir **noves classes basades en classes existents**, amb l'objectiu de re-utilizar codi previàment desenvolupat generant una una jerarquía de classes dins de l'aplicació. Llavors, si una classe es deriva d'un altre , **aquest hereda els seus atributs i mètodes.** La classe derivada pot afegir nous atributs i mètodes i/o redefinir els atributs i mètodes heredats. Per a que un atribut i mètode poden ser heredats, és necessari que la seva visibilitat sigui "protected".

Protected: només es poden accedir per la classe i les subclasses (herència). No es poden accedir des d'una instància.



Herència simple(S'hereda d'una única super classe)
no hi ha herència múltiple (heredar de més d'una super clase, es pot fer simular amb interfícies.)


Animal
Anfibio
Mamifero
Reptil
Caballo


Poligono
- int numLados
+ double area ()
Rectangulo
- double lado1;
- double lado2
+ double area ()
Triangulo
- double lado1;
- double lado2;
- double lado3;
+ double area ()

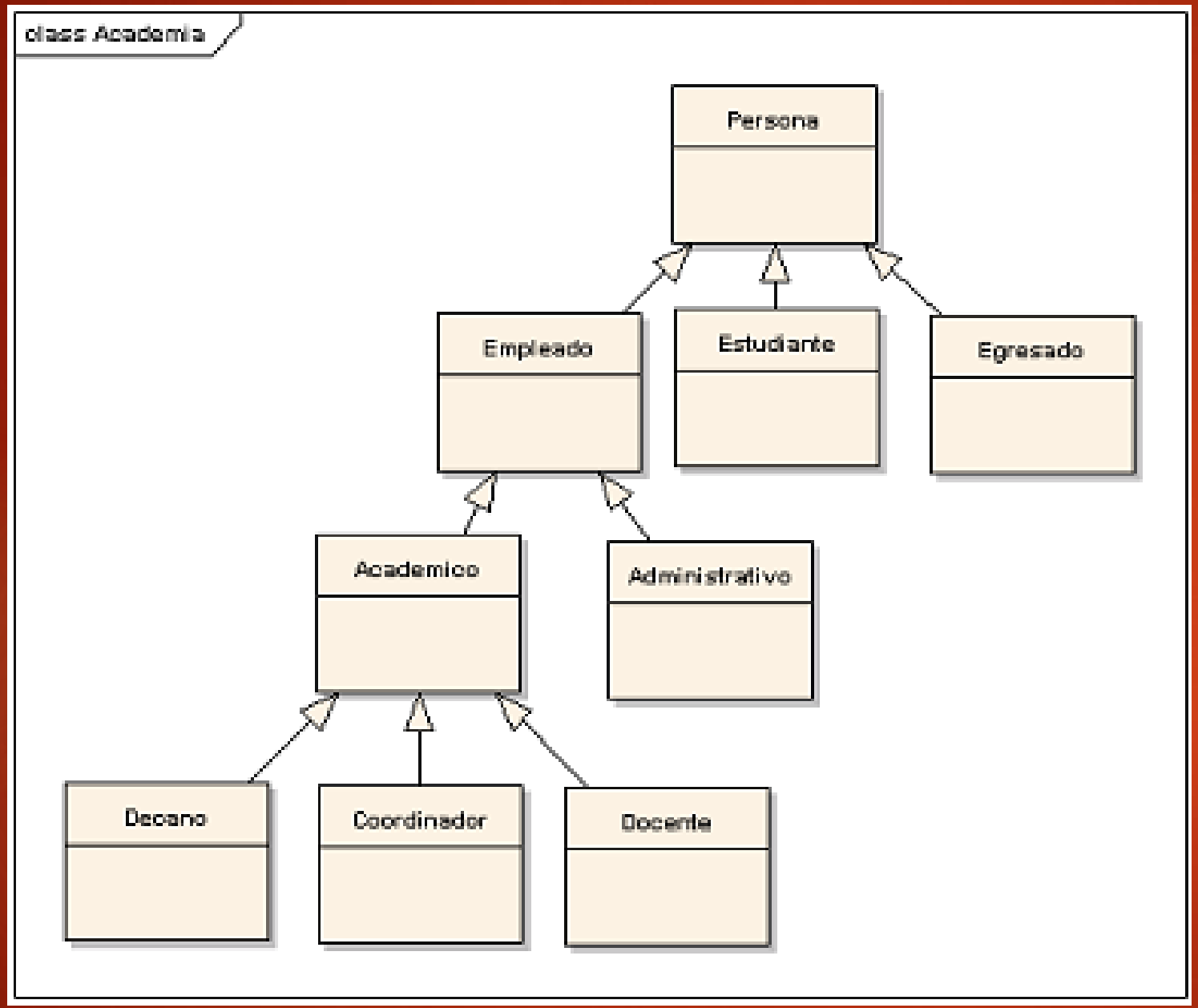
Java

Exercici

Pensar el diagrama de herència possible de…

Membres d'una institució d'educació:

La institución está formada per personas, pero cada persona tiene un rol dentro de la institución, que podría ser: empleado, estudiante, egresado.
Así mismo un empleado se puede clasificar en: académico, administrativo.
De académico se puede derivar, decano, coordinador, docente.



Sol

Molt important un bon diagrama de classes inicial
Amb una herència ben definida.
Pregunta :"Es un..."

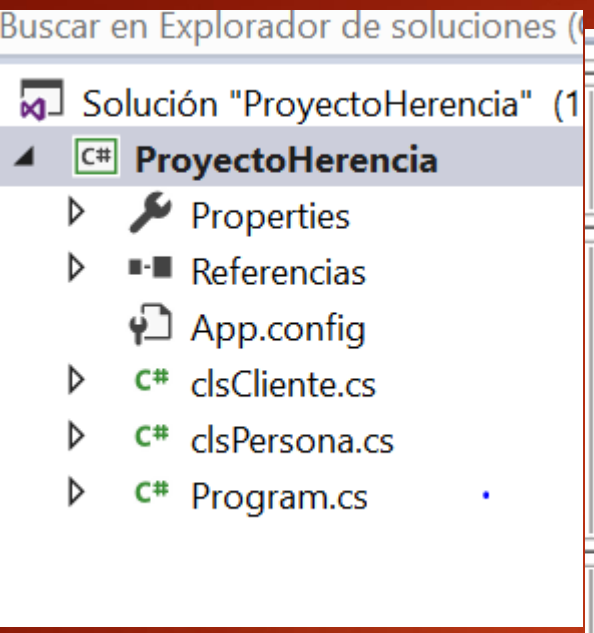
Herència en C#:

Els dos punts :

Una classe hereda d'una altra

Paraula clau **base**

```
public clsCliente(string name,string apell,string dni,int i,string e,string c):base(name,apell,dni)
{
```




```
namespace ProyectoHerencia
{
    4 referencias
    class clsPersona
    {
        private string _Nombre;
        private string _Apellidos;
        private string _DNI;
        2 referencias
        public clsPersona(string n,string a,string d)
        {
            Nombre = n;
            Apellidos = a;
            DNI = d;

        }
        2 referencias
        public string Nombre { get => _Nombre; set => _Nombre = value; }
        1 referencia
        public string Apellidos { get => _Apellidos; set => _Apellidos = value; }
        1 referencia
        public string DNI { get => _DNI; set => _DNI = value; }
    }
}
```

```
namespace ProyectoHerencia
{
    3 referencias
    internal class clsCliente:clsPersona
    {
        private int _IdCliente;
        private string _Empresa;
        private string _Ciudad;
        1 referencia
        public clsCliente(string name,string apell,string dni,int i,string e,string c):base
          (name,apell,dni)
        {
            IdCliente=i;
            Empresa = e;
            Ciudad = c;
        }
        1 referencia
        public int IdCliente { get => _IdCliente; set => _IdCliente = value; }
        1 referencia
        public string Ciudad { get => _Ciudad; set => _Ciudad = value; }
        1 referencia
        public string Empresa { get => _Empresa; set => _Empresa = value; }
    }
}
```

```
namespace ProyectoHerencia
{
    0 referencias
    class Program
    {
        0 referencias
        static void Main(string[] args)
        {
            clsPersona persona1 = new clsPersona("Laura","Lopez","24345345T");
            clsCliente cliente1 = new clsCliente("Pedro","Martinez","45635454E",2345,"Coritel","BCN");
            Console.WriteLine("Bienvenid/a:"+cliente1.Nombre);


        }
    }
}
```

protected

Un element protected és accesible dins de las seva classe i per part de les instàncies de classes derivades.

Una propietat protected d'una classe és accessible des d'una classe derivada només si l'accés es produeix mitjançant la clase derivada.

Si posem una propietat protected

```
protected string _Nombre;
private string _Apellidos;
private string _DNI;
```

Accés a la propietat des de…
Instància de la classe Persona: No
Instància de la classe derivada: No
Des de la pròpia clase Persona: Sí
Des de la propia classe Cliente: Sí

```
private string _Nombre;
private string _Apellidos;
private string _DNI;
2 referencias
public clsPersona(string n,string a,string d)
{
    Nombre = n;
    Apellidos = a;
    DNI = d;

}
2 referencias
protected string Nombre { get => _Nombre; set => _Nombre = value; }
1 referencia
```

Propietat private Mètode(get/Set) protected

Instància de la classe Persona: No
Instància de la classe derivada: No
Des de la pròpia clase Persona: Sí
Des de la propia classe Cliente: No

```
protected string _Nombre;
private string _Apellidos;
private string _DNI;
2 referencias
public clsPersona(string n,string a,string d)
{
    Nombre = n;
    Apellidos = a;
    DNI = d;
}
1 referencia
protected string Nombre { get => _Nombre; set => _Nombre = value; }
1 referencia
```

Propietat protected
Mètode(get/Set) protected

Instància de la classe Persona: No i mètode tampoc.
Instància de la classe derivada: No i mètode tampoco.
Des de la pròpia clase Persona: Sí
Des de la propia classe Cliente: Sí

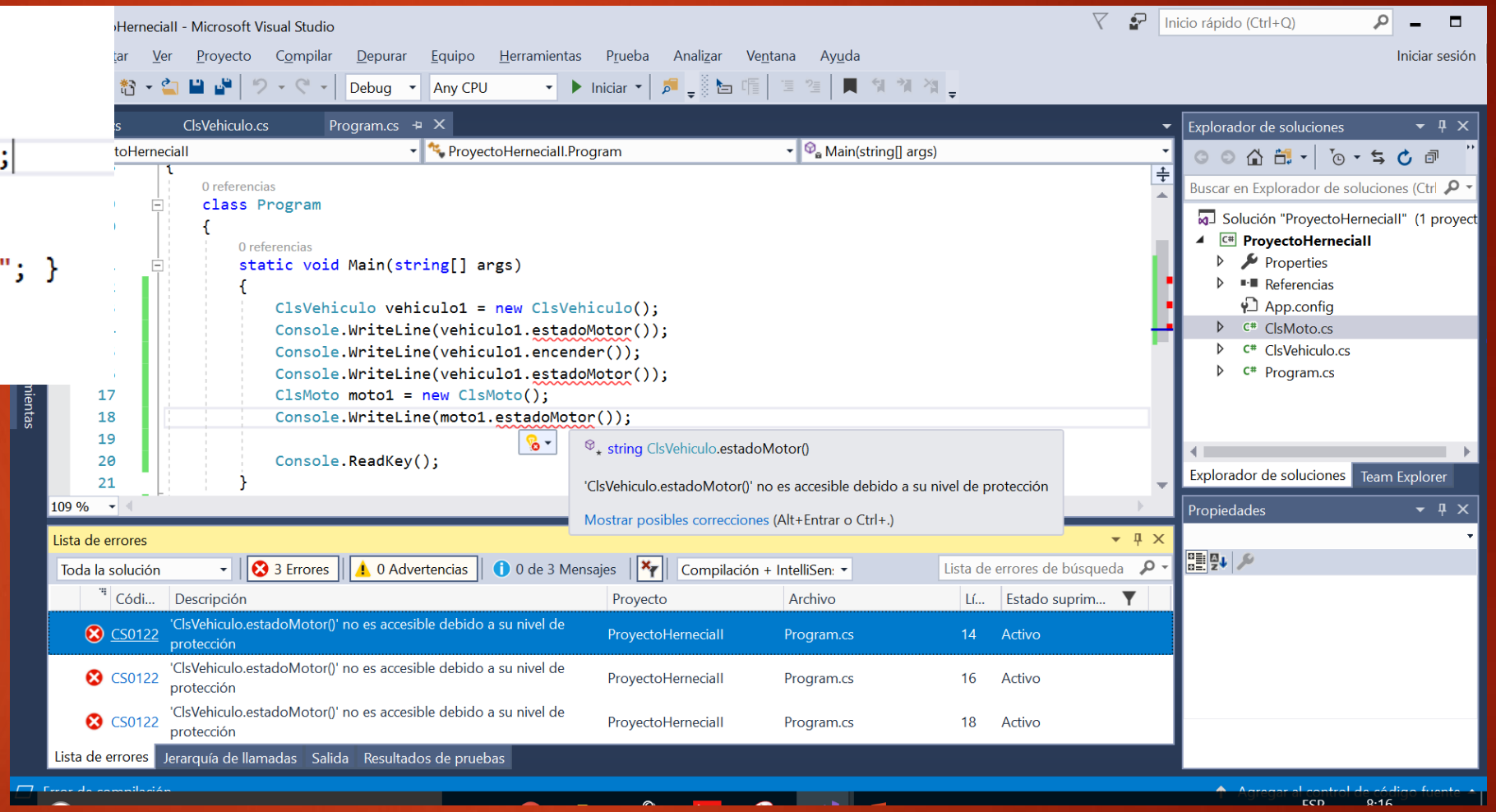
Exemple (protected)

ProyectoHErenciaII / ClsVehiculo/ClsMoto

```
class ClsVehiculo
{
    protected bool motor=false;
    protected string marca;
    protected string color;

    0 referencias
    public bool Motor { get => motor; set => motor = value; }
    0 referencias
    public string Marca { get => marca; set => marca = value; }
    0 referencias
    public string Color { get => color; set => color = value; }

    //constructor por defecto no definimos

    3 referencias
    public string estadoMotor() {
        string cadena = "";
        if (this.motor) {
            cadena = "Estoy encendido";


            }
        else { cadena = "Estoy apagado"; }

        return cadena; }
    1 referencia
    public string encender()
    {
        string cadena = "";
        if (this.motor) { cadena = "Estoy encendiendo el coche!"; }
        else { cadena = "El coche ya está encendido!"; this.motor = true; }
        return cadena;
    }
```

Si posem protected al mètode

```
3 referencias
protected string estadoMotor() {
    string cadena = "";
    if (this.motor) {
        cadena = "Estoy encendido";


        }
    else { cadena = "Estoy apagado"; }

    return cadena; }
1 referencia
```

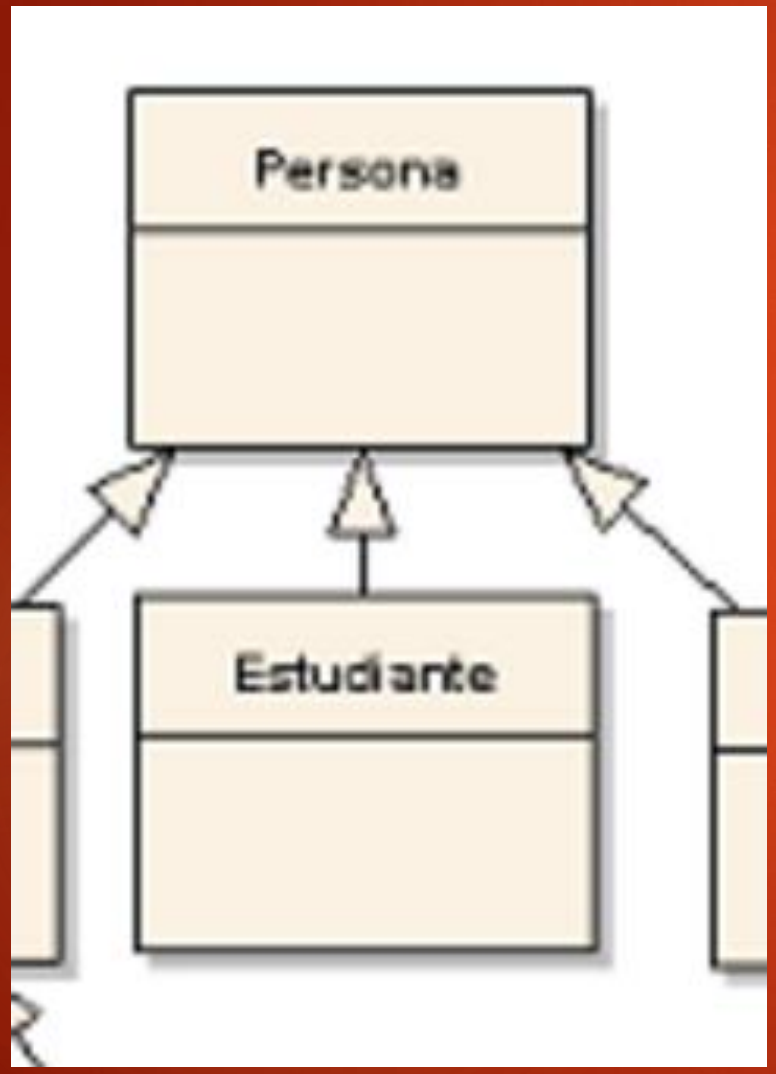


Cuando es protected no se puede acceder des de la instància

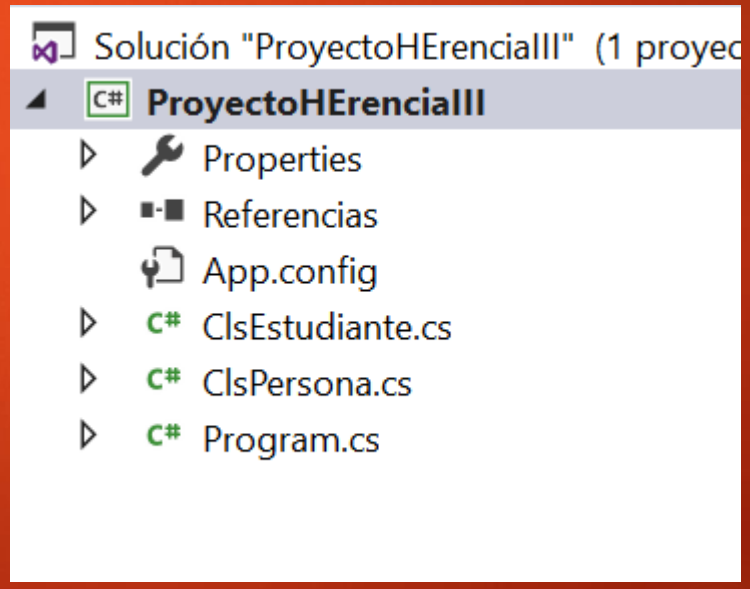
```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ProyectoHerneciaII
{
    2 referencias
    internal class ClsMoto: ClsVehiculo
    { //hereda todas las propiedades y métodos de la clase de la que hereda.

        0 referencias
        public string estadoMotorMoto()
        {
            string cadena = "";
            if (this.motor) //si fuera private en ClsVehiculo da un error de acceso
            {
                cadena = "Estoy encendido";

            }
            else { cadena = "Estoy apagado"; }

            return cadena;
        }
    }
}
```

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

namespace ProyectoHerneciaII
{
    0 referencias
    class Program
    {
        0 referencias
        static void Main(string[] args)
        {
            ClsVehiculo vehiculo1 = new ClsVehiculo();
            Console.WriteLine(vehiculo1.estadoMotor());
            Console.WriteLine(vehiculo1.encender());
            Console.WriteLine(vehiculo1.estadoMotor());
            ClsMoto moto1 = new ClsMoto();
            Console.WriteLine(moto1.estadoMotorMoto());

            Console.ReadKey();
        }
    }
}
```

Necessitarem tres classes:



Persona
Estudiante
I la principal: Program
ProyectoHerenciaIII



```
namespace ProyectoHErenciaIII
{
    2 referencias
    class ClsPersona
    { //propiedades protected
        protected string _Nombre;
        protected string _Apellido;
        protected string _Correo;

        3 referencias
        protected string Nombre { get => _Nombre; set => _Nombre = value; }
        1 referencia
        protected string Apellido { get => _Apellido; set => _Apellido = value; }
        1 referencia
        protected string Correo { get => _Correo; set => _Correo = value; }

        //constructor
        1 referencia
        public ClsPersona(string n, string a,string c) {this.Nombre=n; this.Apellido = a;this.Correo = c; }


    }
}
```

Posem protected en els dos, en la propietat i en els mètodes.

```
namespace ProyectoHErenciaIII
{
    3 referencias
    class ClsEstudiante : ClsPersona
    {   private int _Codigo;
        private string _Facultad;
        1 referencia
        public ClsEstudiante(string name, string apell, string mail, int codi, string f) : base(name,
          apell, mail)
        {
            Codigo = codi;
            Facultad= f;
        }

        1 referencia
        public int Codigo { get => _Codigo; set => _Codigo = value; }
        1 referencia
        public string Facultad { get => _Facultad; set => _Facultad = value; }

    }
}
```

Persona estudiante

Al ser protected els mètodes d'accés hauré de definir un mètode per obtenir el seu valor.
Si els posem public no tindrem problemes d'accés.

```
class Program
{
    0 referencias
    static void Main(string[] args)
    {
        ClsEstudiante estudiante1 = new ClsEstudiante("Laura","Martinez","s@gmail.com",12346,"UB");
        Console.WriteLine(estudiante1.Nombre);//da problemas por el protected
        Console.ReadKey();
    }
}
```

```
namespace ProyectoHErenciaIII
{
    3 referencias
    class ClsEstudiante : ClsPersona
    {   private int _Codigo;
        private string _Facultad;
        1 referencia
        public ClsEstudiante(string name, string apell, string mail, int codi, string f) : base(name,
          apell, mail)
        {
            Codigo = codi;
            Facultad= f;
        }

        1 referencia
        public int Codigo { get => _Codigo; set => _Codigo = value; }
        1 referencia
        public string Facultad { get => _Facultad; set => _Facultad = value; }
        1 referencia
        public string SaberNombre() { return this.Nombre; }
    }
}
```

```
Console.WriteLine(estudiante1.SaberNombre());
Console.ReadKey();
```

```
class ClsPersona
{ //propiedades protected
    protected string _Nombre;
    protected string _Apellido;
    protected string _Correo;

    3 referencias
    public string Nombre { get => _Nombre; set => _Nombre = value; }
    1 referencia
    protected string Apellido { get => _Apellido; set => _Apellido = value; }
    1 referencia
    protected string Correo { get => _Correo; set => _Correo = value; }

    //constructor
    1 referencia
    public ClsPersona(string n, string a,string c) {this.Nombre=n; this.Apellido =
      a;this.Correo = c; }


}
```

O posem el mètode public

```
Console.WriteLine(estudiante1.Nombre);/
```