Programació orientada a objectes

Sonia Sánchez Sierra

Sobreescriure mètodes

Classe cuadrado

Lado:int
.....
+ Método CalcularArea()

Classe cubo

.....
+Método CalcularArea()

Cuadrado

Cubo

Classe cuadrado

ClsCuadrado.cs | Program.cs

juras | ProyectoFiguras.ClsCuadrado | Ca

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

namespace ProyectoFiguras
{
    4 referencias
    class ClsCuadrado
    {
        protected int lado;
        2 referencias
        public ClsCuadrado(int l) { this.lado = l; }

        0 referencias
        protected int Lado { get => lado; set => lado = value; }
        1 referencia
        public double CalcularArea() {
                return Math.Pow(this.lado, 2);
            }
    }
}
```

Classe cubo

ClsCubo.cs | ClsCuadrado.cs | Program.cs

ProyectoFiguras | ProyectoFiguras.ClsCubo

```
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6
7   namespace ProyectoFiguras
8   {
        3 referencias
9       class ClsCubo: ClsCuadrado
10      {
            1 referencia
11          public ClsCubo(int lado):base(lado) { }
            1 referencia
12          public new double CalcularArea()
13          {
14              return Math.Pow(this.lado, 3);
15          }
16      }
17  }
18
```

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

namespace ProyectoFiguras
{
    0 referencias
    class Program
    {
        0 referencias
        static void Main(string[] args)
        {
            ClsCuadrado cuadrado1 = new ClsCuadrado(5);
            ClsCubo cubo1 = new ClsCubo(5);
            Console.WriteLine("El área del cubo es:"+cubo1.CalcularArea()+ " y el del cuadrado
              es:"+cuadrado1.CalcularArea());
            Console.ReadKey();
        }
    }
}
```

Sobreescritura de mètodes

```
public class ClaseBase
{
      public int a;
      public int b;
      public virtual void  Imprimir()
      {
            Console.WriteLine("a={0},b={1}",a,b);
      }
}

public class ClaseDerivada:ClaseBase
{
      public int c;
      public override void  Imprimir()
      {
            Console.WriteLine("a={0},b={1},c={2}",a,b,c);
      }
}
class HerenciaApp
```

Imaginemos que tenemos que crear dos clases, una para los empleados y otra para los jefes.

Exercici

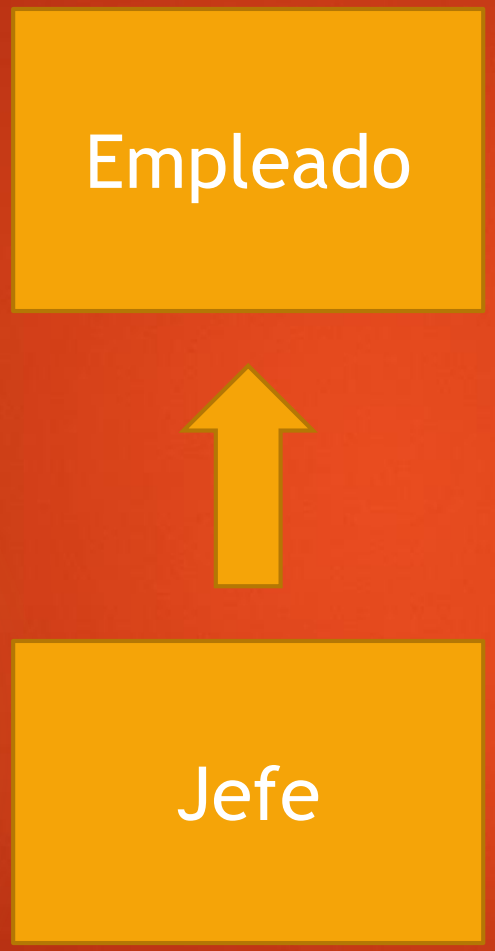
De los empleados se guarda el nombre, el sueldo, departamento.

Los jefes tienen los mismo que los empleados pero también pueden tener un incentivo.

Pensar “Es un…”

Un empleado es un jefe…(siempre)?
Un jefe es un empleado…(siempre)?

Sol



Empleado

Jefe

Classe Empleado

```
2 referencias
class ClsEmpleado
{
    protected string nombre;
    protected string departamento;
    protected float sueldo;

    1 referencia
    protected string Nombre { get => nombre; set => nombre = value; }
    1 referencia
    protected string Departamento { get => departamento; set => departamento = value; }
    3 referencias
    protected float Sueldo { get => sueldo; set => sueldo = value; }

    1 referencia
    public ClsEmpleado(string n, string d,float s) {
        this.Nombre = n;
        this.Departamento = d;
        this.Sueldo = s;

    }
    0 referencias
    public float GetSueldo() { return this.Sueldo; }
}
}
```

Classe Jefe

```
namespace ProyectoEmpleadosJEfes
{
    3 referencias
    class ClsJefe:ClsEmpleado
    {   private float incentivo;
        1 referencia
        public ClsJefe(string n,string d,float s, float i):base(n,  d,  s) { Incentivo = i; }

        1 referencia
        public float Incentivo { get => incentivo; set => incentivo = value; }


        1 referencia
        public new float GetSueldo() { return this.Sueldo; }

        //public  float GetSueldoConIncentivo() { return this.Sueldo + this.Incentivo; }
    }
}
```

Principal

```
namespace ProyectoEmpleadosJEfes
{
    0 referencias
    class Program
    {
        0 referencias
        static void Main(string[] args)
        {
            ClsEmpleado empleado1 = new ClsEmpleado("Pedro","marqueting",980.78f);
           Console.WriteLine( "el sueldo del empleado es: "+empleado1.GetSueldo());
            ClsJefe jefe1 = new ClsJefe("Laura","administracion",980.45f,150.00f);
            Console.WriteLine("el sueldo del empleado es: " + jefe1.GetSueldo());
            Console.ReadKey();

        }
    }
}
```

Definir un altre mètode per tenir només el sou.

Classe Jefe

```
namespace ProyectoEmpleadosJEfes
{
    3 referencias
    class ClsJefe:ClsEmpleado
    {   private float incentivo;
        1 referencia
        public ClsJefe(string n,string d,float s, float i):base(n,  d,  s) { Incentivo = i; }

        2 referencias
        public float Incentivo { get => incentivo; set => incentivo = value; }

        //public new float GetSueldo() { return this.Sueldo; }

        1 referencia
        public  float GetSueldoConIncentivo() { return this.Sueldo + this.Incentivo; }
    }
}
```

Principal

```
class Program
{
    0 referencias
    static void Main(string[] args)
    {
        ClsEmpleado empleado1 = new ClsEmpleado("Pedro","marqueting",980.78f);
       Console.WriteLine( "el sueldo del empleado es: "+empleado1.GetSueldo());
        ClsJefe jefe1 = new ClsJefe("Laura","administracion",980.45f,150.00f);
        Console.WriteLine("el sueldo del jefe sin incentivo es: " + jefe1.GetSueldo());
        Console.WriteLine("el sueldo del jefe con incentivo es: " + jefe1.GetSueldoConIncentivo());

        Console.ReadKey();

    }
}
```