




Métodos Avanzados de Programación Científica y Computación

M^a Luisa Díez Platas

Tema 2. Relaciones entre clases (I)

¿Cómo estudiar este tema?

| IDEAS CLAVE | LO + RECOMENDADO | + INFORMACIÓN | TEST |
|--|---|--|---|
| <p>¿Cómo estudiar este tema?</p> <p>Abstracción y herencia</p> <p>Conceptos avanzados de herencia</p> <p>Polimorfismo</p> <p>Composición y agregación</p> <p><code>This</code> y <code>super</code></p> <p>Complejidad de un algoritmo</p> | <p>No dejes de leer...</p> <p>Herencia</p> <p>Paquetes en Java</p> <p>Herencia en Java</p> <p>No dejes de ver...</p> <p> Proyecto en ArgoUML</p> <p> Herencia en Java</p> <p> Polimorfismo en Java</p> | <p>A fondo</p> <p>Ejemplo de herencia en Java</p> <p>Ejemplo de polimorfismo en Java</p> <p>UML Distilled</p> <p>Programación orientada a objetos usando Java</p> |  |

- Abstracción y herencia
- Polimorfismo
- Agregación
- Conceptos relacionados con los métodos

Características de los métodos de las clases. This

- **This**, argumento oculto de todo métodos no estático de una clase
- Es una referencia al objeto que va a soportar la operación

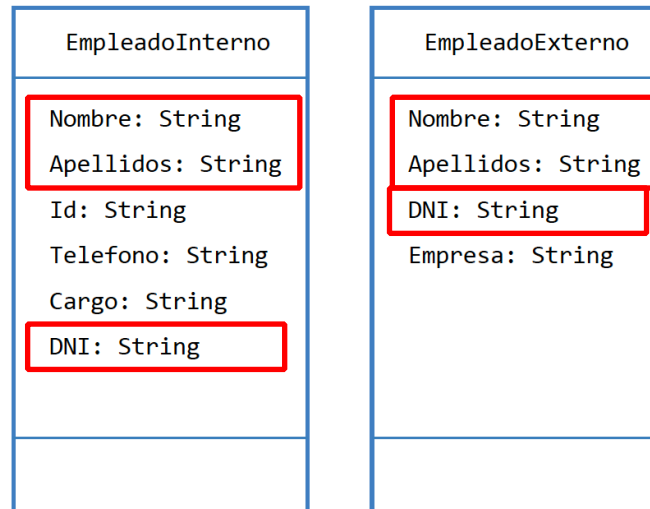
```
class Persona{  
    //definicion de atributos  
    private int edad;  
    //definicion de métodos  
    public Persona(){this.edad=0;}  
    public Persona(int ed;){this. edad=ed;}  
    public getEdad(){return this.edad;}  
    public setEdad(int ed){this.edad=ed;}  
}
```

public setEdad(int edad){this.edad=edad;}

Necesario resolver

Herencia

- La herencia implica un proceso de abstracción
- Generalización/especialización

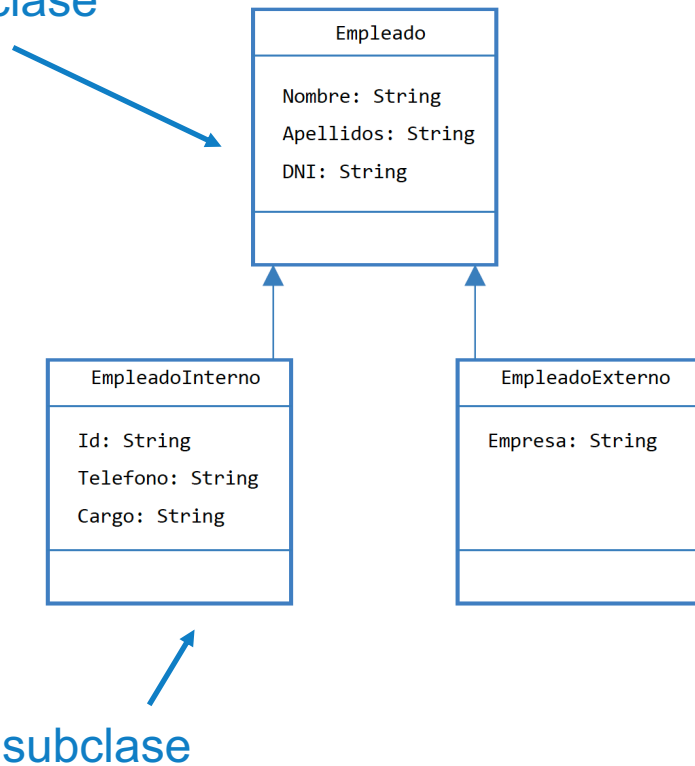


Característica para definir relaciones jerárquicas → proceso por el que una subclase recibe todos los atributos y métodos de una clase superior

Herencia

- La clase base contiene los atributos comunes a las clases derivadas

superclase



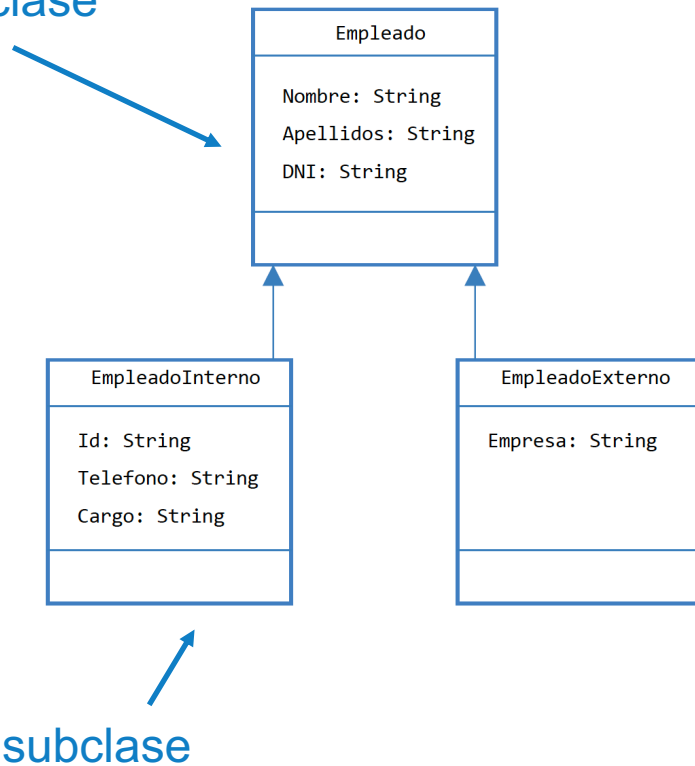
```
class Empleado{
String Nombre;
String Apellidos;
String DNI;
}
```

```
class EmpleadoInterno extends Empleado{
String Id;
String Telefono;
String Cargo;
}
class EmpleadoExterno extends Empleado{
String Empresa;
}
```

Herencia

- La clase base contiene los atributos comunes a las clases derivadas

superclase



```
class Empleado{
    private String Nombre;
    private String Apellidos;
    private String DNI;
    public String getNombre(){return Nombre;}
}
```

```
}
```

```
class EmpleadoInterno extends Empleado{
    String Id;
    String Telefono;
    String Cargo;
    public String obtenerNombre(){return getNombre();}
}
```

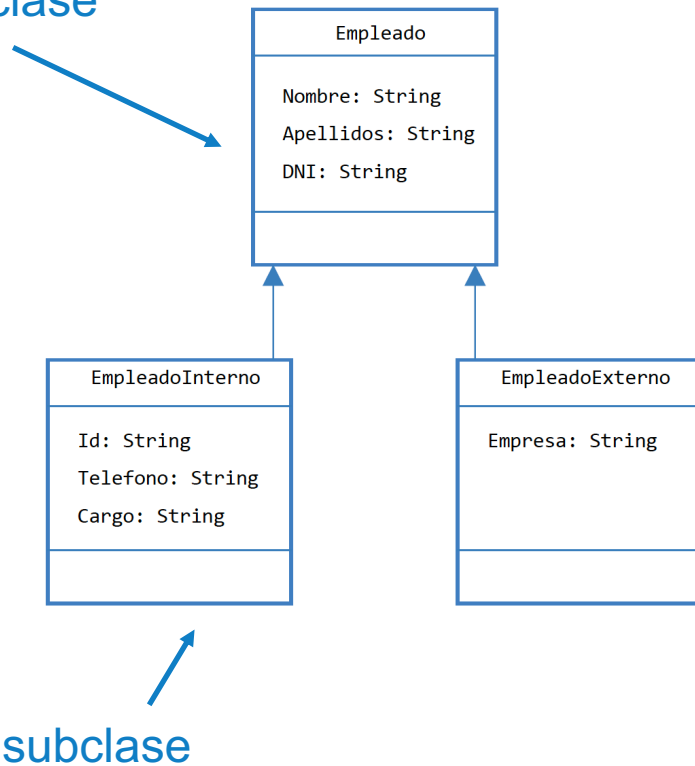
```
}
```

```
class EmpleadoExterno extends Empleado{
    String Empresa;
}
```

Herencia

- La clase base contiene los atributos comunes a las clases derivadas

superclase



```
class Empleado{
    protected String Nombre;
    private String Apellidos;
    private String DNI;
    public String getNombre(){return Nombre;}
}
```

```
}
```

```
class EmpleadoInterno extends Empleado{
    String Id;
    String Telefono;
    String Cargo;
    public String obtenerNombre(){return Nombre;}
}
```

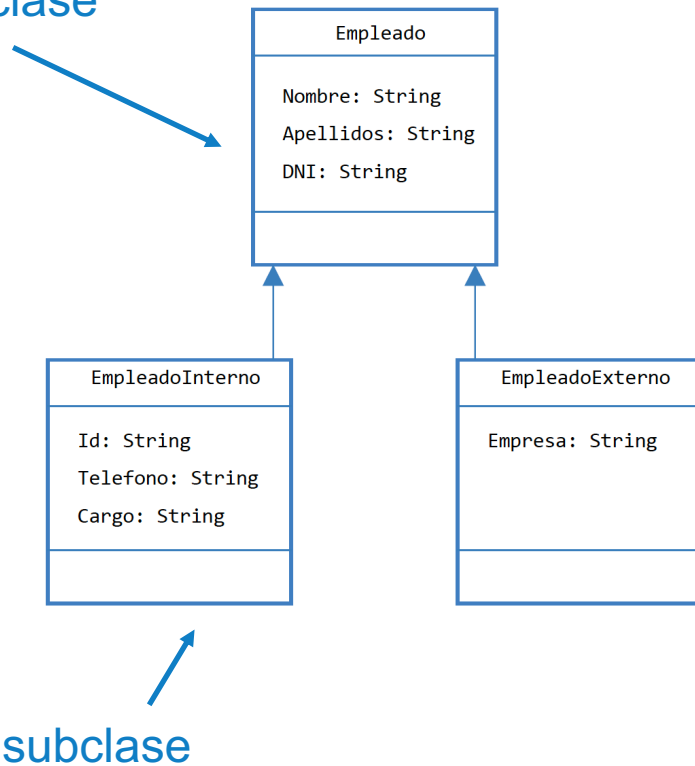
```
}
```

```
class EmpleadoExterno extends Empleado{
    String Empresa;
}
```

Herencia

- La clase base contiene los atributos comunes a las clases derivadas

superclase



```
class Empleado{
    public String Nombre;
    private String Apellidos;
    private String DNI;
    public String getNombre(){return Nombre;}
}
```

```
}
```

```
class EmpleadoInterno extends Empleado{
    String Id;
    String Telefono;
    String Cargo;
    public String obtenerNombre(){return Nombre;}
}
```

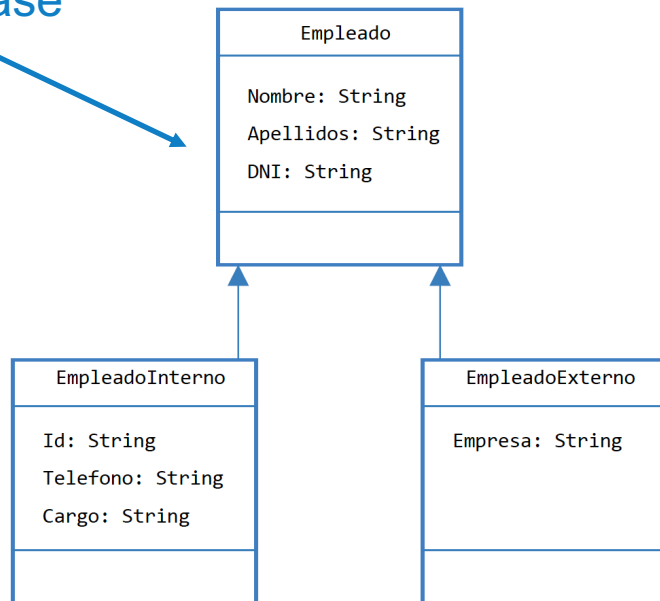
```
}
```

```
class EmpleadoExterno extends Empleado{
    String Empresa;
}
```


Herencia. super()

- La clase base contiene los atributos comunes a las clases derivadas

superclase



subclase

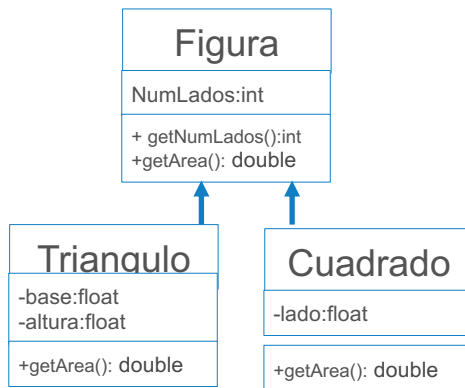
```
class Empleado{
    private String Nombre;
    private String Apellidos;
    private String DNI;
    public Empleado(String n, String a; String d){
        Nombre=n;
        Apellidos=a;
        DNI=d;
    }
    public String getNombre(){return Nombre;}
}

class EmpleadoExterno extends Empleado{
    String Empresa;
    public Empleado(String n, String a; String d,String e){
        super(n,a,d);
        Empresa=e;
    }
}
```

```
EmpleadoExterno ee1=new EmpleadoExterno ("Jose","Garcia","22222X","Aentar")
```

Herencia. Clases abstractas

- La clase base contiene los atributos comunes a las clases derivadas



```
abstract class Figura{
    private int NumLados;
    public int getNumLados(){return NumLados;}
    public abstract double getArea();
}

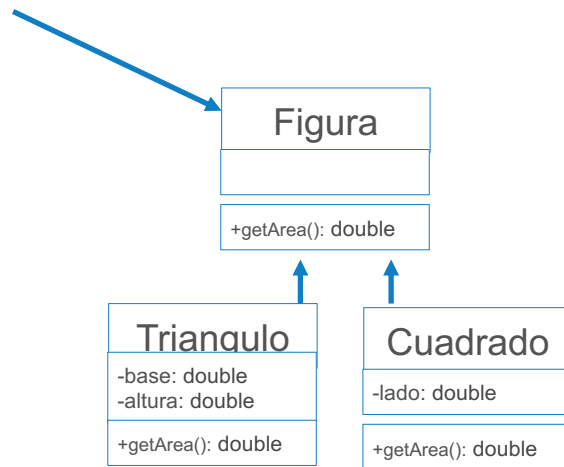
class Triangulo extends Figura{
    double base;
    double altura;
    public double getArea(){return base*altura/2;}
}
```

No tiene implementado alguno de los métodos
No se pueden crear ejemplares → new de la clase

Interfaces

- La clase base contiene los atributos comunes a las clases derivadas

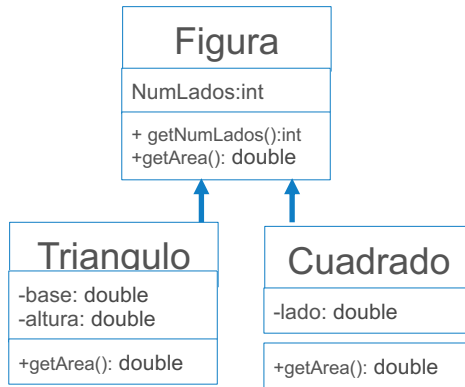
superclass



```
interface class Figura{
    public double getArea();
}
class Triangulo implements Figura{
    double base;
    double altura;
    public float getArea(){return base*altura/2;}
}
```

No tiene implementado ninguno de los métodos (abstracta pura)
No se pueden crear ejemplares → new de la clase

Herencia. Polimorfismo



capacidad de ejecutar un método que cambia de comportamiento dependiendo de cómo se haya realizado la instanciación del mismo.

```
abstract class Figura{
    private int NumLados;
    public int getNumLados(){return NumLados;}
    public abstract double getArea();
}

class Triangulo extends Figura{
    double base;
    double altura;
    public Triangulo(double b, double a,int n){
        super(n);
        base=b;altura=a;
    }
    public double getArea(){return base*altura/2;}
}

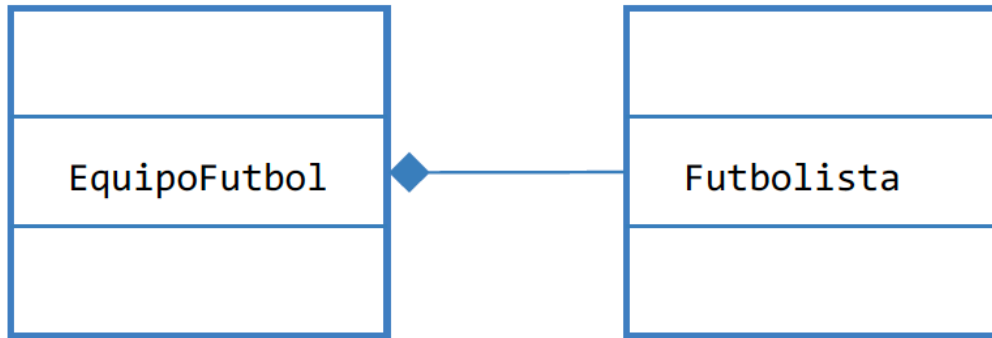
class Cuadrado extends Figura{
    double lado;
    public Cuadrado(double lint n){
        super(n);
        lado=l;altura=a;
    }
    public double getArea(){return base*altura/2;}

    public double getArea(){return lado*lado;}
}
```

```
Figura t= new Triangulo(2.0,3.0,3);
Figura c= new Cuadrado(2.0);
t.getArea();
c.getArea();
```

← **Ligadura tardía**

Agregación



```
class Futbolista{
public Futbolista(.....){//.....}
class EquipoFutbol {
Private Futbolista futbolistas[];
public EquipoFutbol(.....){
futbolistas=new Futbolista[20];
futbolistas[0]=new Futbolista(....);
}
}
```

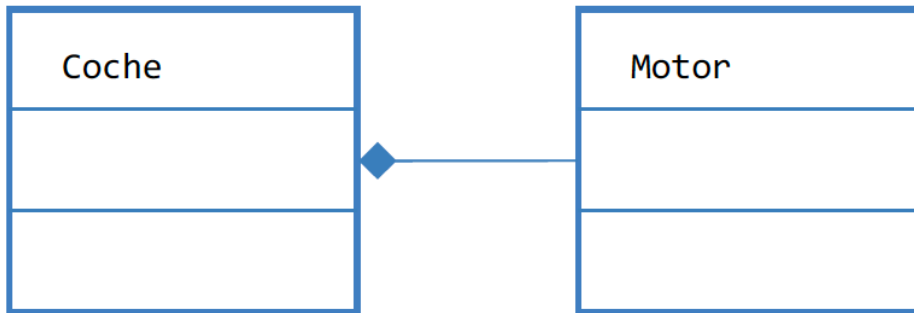
Una clase esta formada por objetos de otra

Los objetos contenidos no se crean al crear el objeto de la contenedora

La destrucción del objeto de la contenedora no implica la destrucción de los objetos contenidos

Composición débil

Composición



```
class Motor{
public Motor(.....){//.....}
class Coche {
private Motor elMotor;
public Coche(.....){
elMotor=new Motor(...);
}
}
```

Una clase esta formada por objetos de otra

Los objetos **contenidos se crean al crear el objeto de la contenedora**

La destrucción del objeto de la contenedora no implica la destrucción de los objetos contenidos

Composición débil

UNIVERSIDAD
INTERNACIONAL
DE LA RIOJA

unir

www.unir.net