

# Programación Frontend y Backend

*BLOQUE JAVA*

Objetos II

01

## Herencia



GOBIERNO  
DE ESPAÑA

MINISTERIO  
DE INDUSTRIA, COMERCIO  
Y TURISMO



Escuela de  
organización  
industrial



**Unión Europea**  
**Fondo Social Europeo**  
**Iniciativa de Empleo Juvenil**  
El FSE invierte en tu futuro





Público

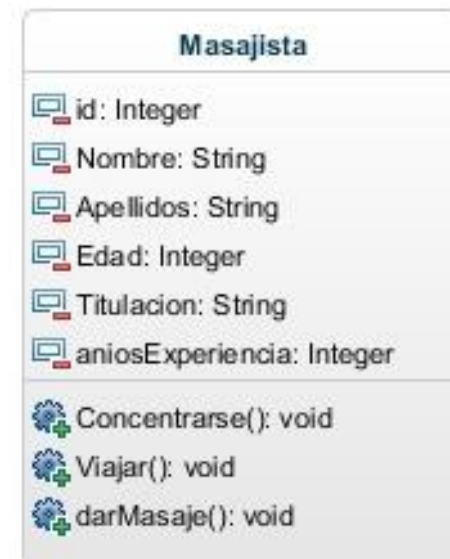


Protegido



Privado

## Herencia





Público

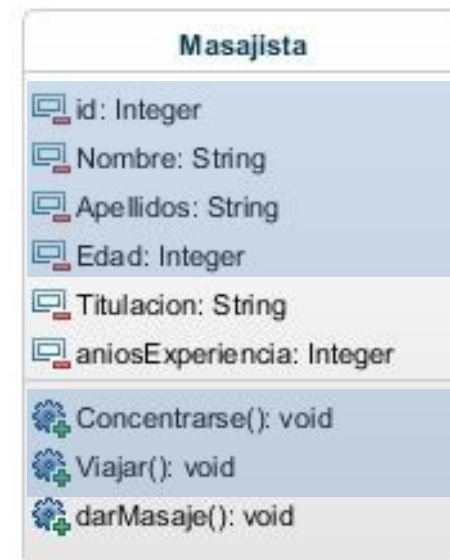
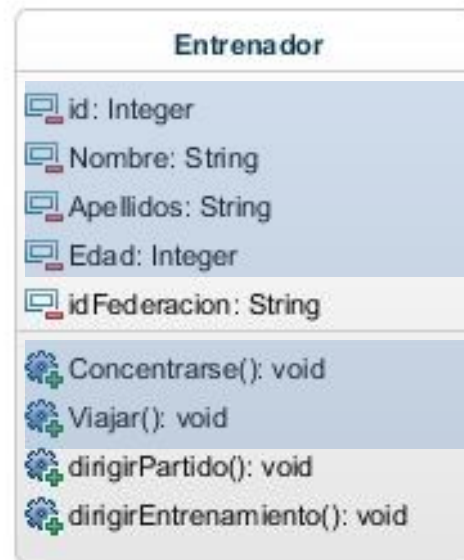


Protegido



Privado

## Herencia



## Herencia



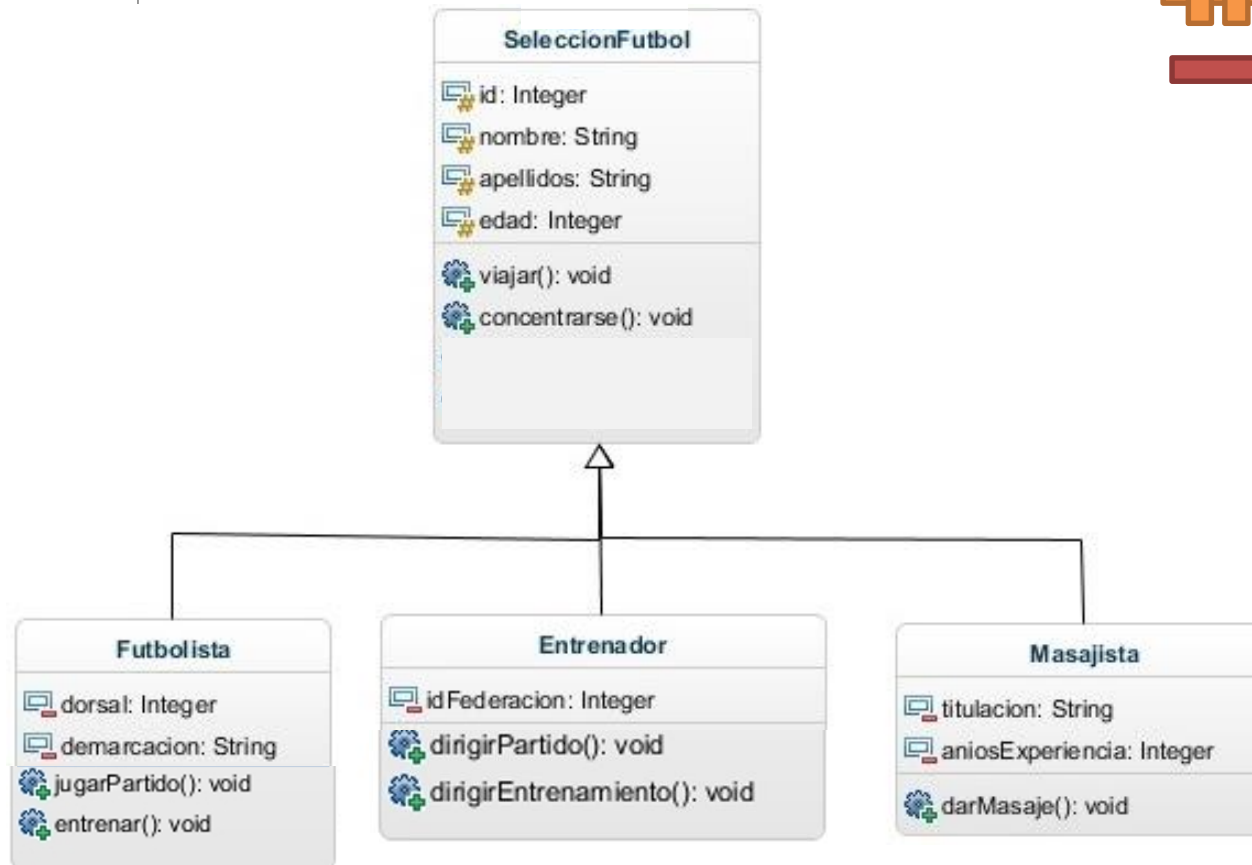
Público



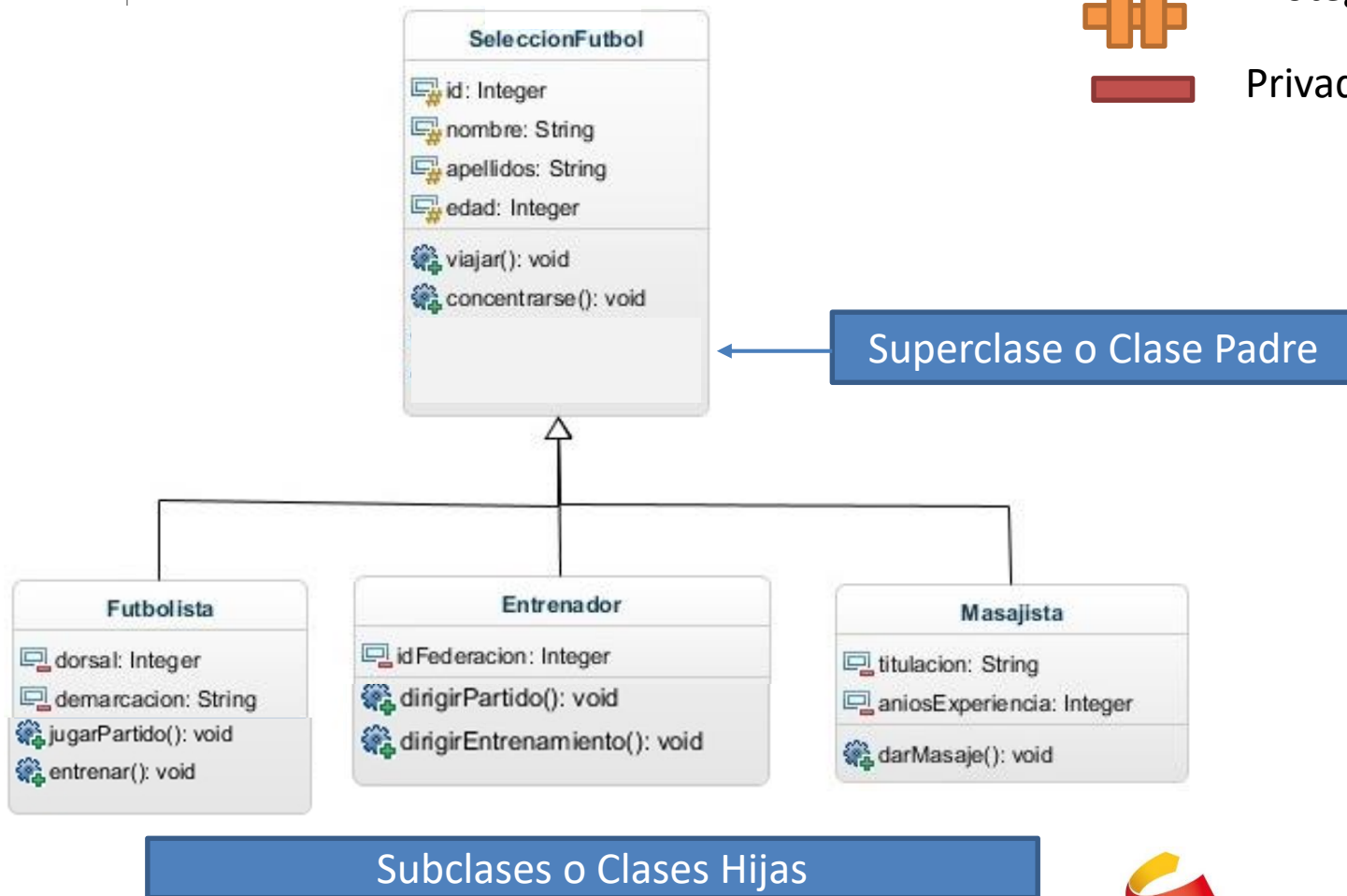
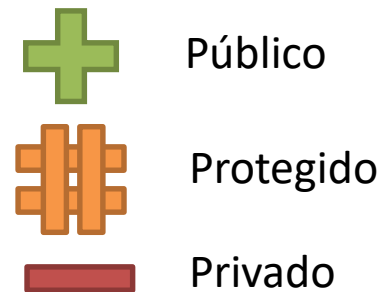
Protegido



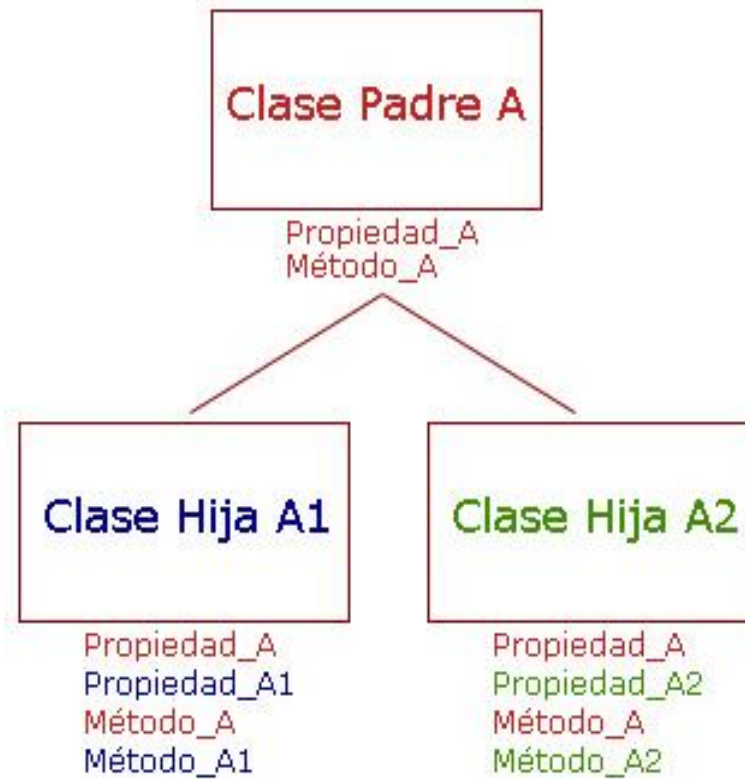
Privado



## Herencia



## Herencia



Público



Protegido



Privado



## Herencia

*class* **SeleccionFutbol**

Superclase o Clase Padre

*class* **Futbolista** *extends* **SeleccionFutbol**

*class* **Entrenador** *extends* **SeleccionFutbol**

*class* **Masajista** *extends* **SeleccionFutbol**

Subclases o Clases Hijas



GOBIERNO  
DE ESPAÑA

MINISTERIO  
DE INDUSTRIA, COMERCIO  
Y TURISMO



Escuela de  
organización  
industrial



Unión Europea  
Fondo Social Europeo  
Iniciativa de Empleo Juvenil  
El FSE invierte en tu futuro





## Herencia

- La herencia es un aspecto fundamental del paradigma orientado a objetos.
- Toda clase Java por defecto hereda de la clase ***Object***.

### Ejemplo:

|  |               |
|--|---------------|
| public class SeleccionFutbol {}                    | (Clase Padre) |
| public class Futbolista extends SeleccionFutbol {} | (Clase Hija)  |
| public class Entrenador extends SeleccionFutbol {} | (Clase Hija)  |
| public class Masajista extends SeleccionFutbol {}  | (Clase Hija)  |



Público

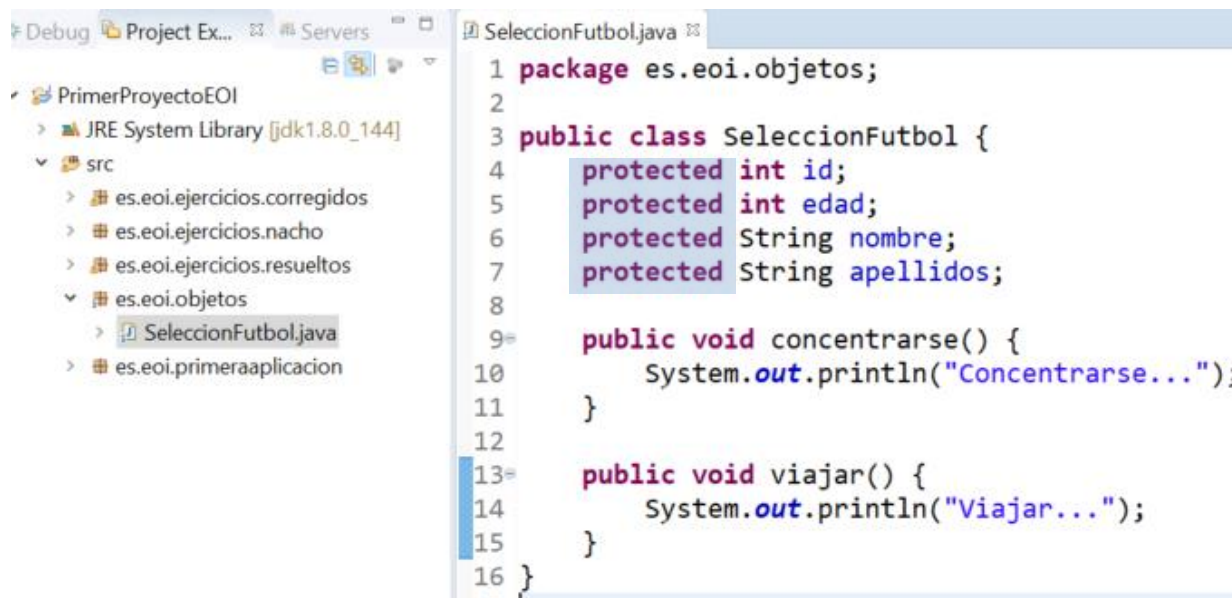


Protegido



Privado

## Herencia



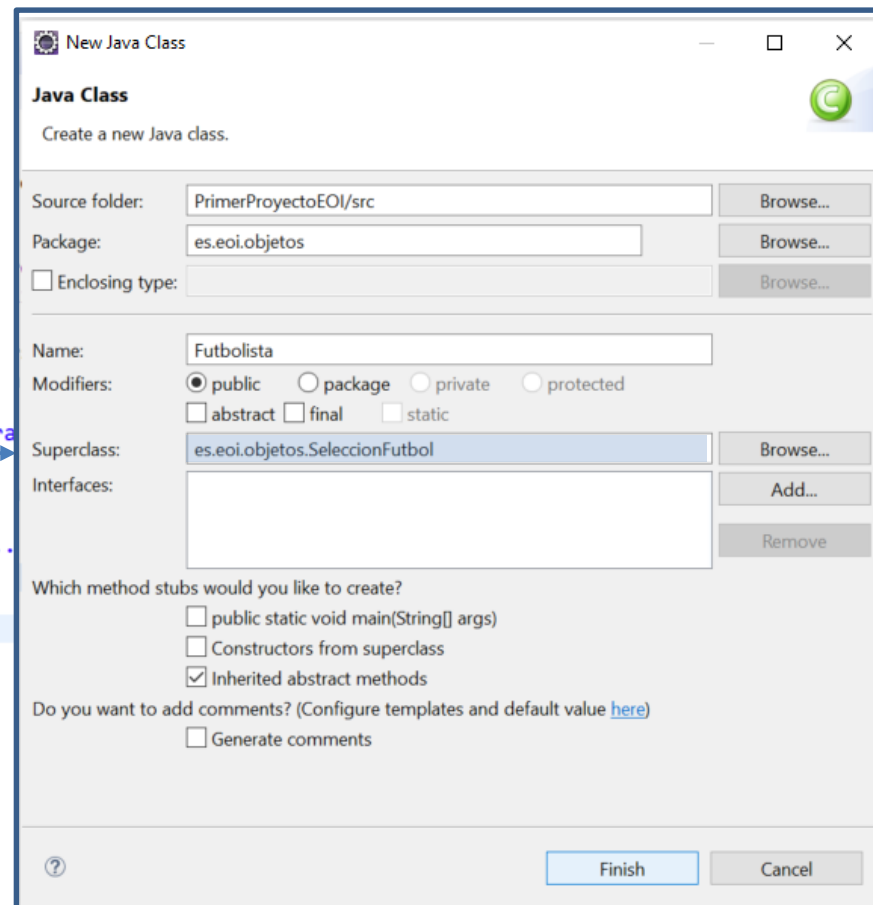
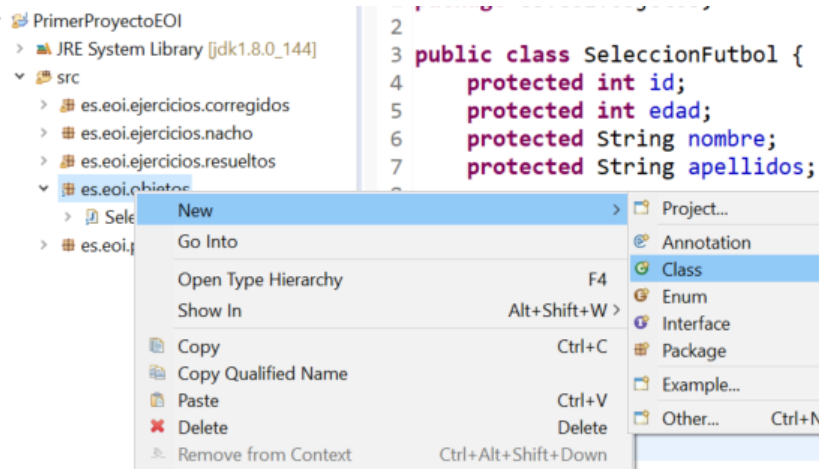
The screenshot shows an IDE with a project named 'PrimerProyectoEOI'. The project structure on the left includes a 'src' folder with subfolders 'es.eoi.ejercicios.corregidos', 'es.eoi.ejercicios.nacho', 'es.eoi.ejercicios.resueltos', 'es.eoi.objetos', and 'es.eoi.primeraplicacion'. The 'es.eoi.objetos' folder is expanded, showing the file 'SeleccionFutbol.java'. The code in this file is as follows:

```
1 package es.eoi.objetos;
2
3 public class SeleccionFutbol {
4     protected int id;
5     protected int edad;
6     protected String nombre;
7     protected String apellidos;
8
9     public void concentrarse() {
10         System.out.println("Concentrarse...");
11     }
12
13     public void viajar() {
14         System.out.println("Viajar...");
15     }
16 }
```

Superclase o Clase Padre

GOBIERNO  
DE ESPAÑAMINISTERIO  
DE INDUSTRIA, COMERCIO  
Y TURISMOEscuela de  
organización  
industrialUnión Europea  
Fondo Social Europeo  
Iniciativa de Empleo Juvenil  
El FSE invierte en tu futuro

## Herencia



# Herencia

1º

```
1 package es.eoi.objetos;
2
3 public class Futbolista extends SeleccionFutbol {
4
5 }
6
```

2º

```
1 package es.eoi.objetos;
2
3 public class Masajista extends SeleccionFutbol {
4
5 }
6
```

3º

```
1 package es.eoi.objetos;
2
3 public class Entrenador extends SeleccionFutbol {
4
5 }
6
```

Mismo Paquete



Público



Protegido



Privado

## Tipos de Modificadores de Acceso

| MODIFICADOR            | CLASE | PACKAGE | SUBCLASE | TODOS |
|------------------------|-------|---------|----------|-------|
| <b>public</b>          | Sí    | Sí      | Sí       | Sí    |
| <b>protected</b>       | Sí    | Sí      | Sí       | No    |
| <b>No especificado</b> | Sí    | Sí      | No       | No    |
| <b>private</b>         | Sí    | No      | No       | No    |

GOBIERNO  
DE ESPAÑAMINISTERIO  
DE INDUSTRIA, COMERCIO  
Y TURISMOEscuela de  
organización  
industrial**Unión Europea**  
**Fondo Social Europeo**  
**Iniciativa de Empleo Juvenil**  
El FSE invierte en tu futuroSISTEMA NACIONAL DE  
**GARANTÍA**  
**JUVENIL**



Público



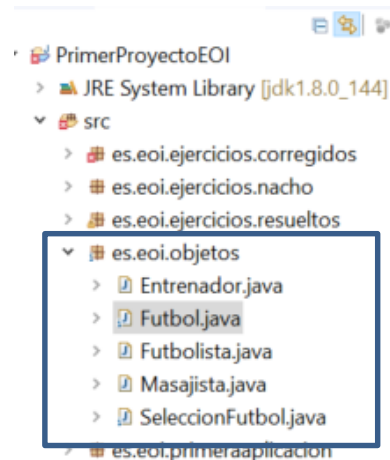
Protegido



Privado

## Tipos de Modificadores de Acceso

Mismo Paquete



```
1 package es.eoi.objetos;
2
3 public class Futbol {
4
5     public static void main(String[] args) {
6
7         SeleccionFutbol seleccion = new SeleccionFutbol();
8         Futbolista futbolista1 = new Futbolista();
9         Entrenador entrenador1 = new Entrenador();
10        Masajista masajista1 = new Masajista();
11
12        System.out.println(seleccion.nombre);
13        System.out.println(futbolista1.nombre);
14        System.out.println(entrenador1.nombre);
15        System.out.println(masajista1.nombre);
16    }
```



Público



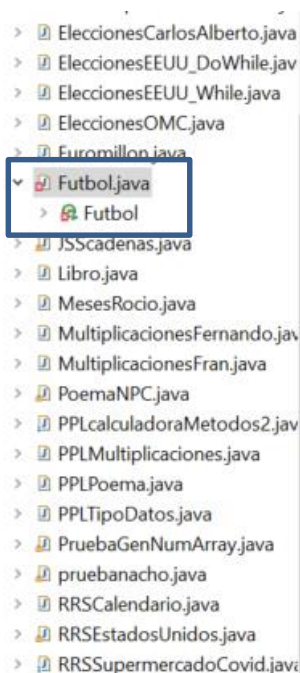
Protegido



Privado

## Tipos de Modificadores de Acceso

### Paquetes Distintos



```
5 import es.eoi.objetos.masajista;  
6 import es.eoi.objetos.SeleccionFutbol;  
7  
8 public class Futbol {  
9  
10     public static void main(String[] args) {  
11  
12         SeleccionFutbol seleccion = new SeleccionFutbol();  
13         Futbolista futbolista1 = new Futbolista();  
14         Entrenador entrenador1 = new Entrenador();  
15         Masajista masajista1 = new Masajista();  
16  
17         System.out.println(seleccion.nombre);  
18         System.out.println(futbolista1.nombre);  
19         System.out.println(entrenador1.nombre);  
20         System.out.println(masajista1.nombre);  
21  
22     }  
23  
24 }  
25 }
```

The field SeleccionFutbol.nombre is not visible  
2 quick fixes available:  
• Change visibility of 'nombre' to 'public'  
• Create getter and setter for 'nombre'...  
Press 'F2' for focus

GOBIERNO  
DE ESPAÑAMINISTERIO  
DE INDUSTRIA, COMERCIO  
Y TURISMOEscuela de  
organización  
industrialUnión Europea  
Fondo Social Europeo  
Iniciativa de Empleo Juvenil  
El FSE invierte en tu futuro





Público



Protegido



Privado

## Herencia

```
SeleccionFutbol seleccion = new SeleccionFutbol();  
Futbolista futbolista1 = new Futbolista();  
Entrenador entrenador1 = new Entrenador();  
Masajista masajista1 = new Masajista();
```

```
seleccion.concentrarse();  
futbolista1.concentrarse();  
entrenador1.concentrarse();  
masajista1.concentrarse();
```

```
Console  
<terminated> Futbol [Java Application] (  
Concentrarse...  
Concentrarse...  
Concentrarse...  
Concentrarse...
```

```
public class SeleccionFutbol {  
    protected int id;  
    protected int edad;  
    protected String nombre;  
    protected String apellidos;  
  
    public void concentrarse() {  
        System.out.println("Concentrarse...");  
    }  
}
```

GOBIERNO  
DE ESPAÑAMINISTERIO  
DE INDUSTRIA, COMERCIO  
Y TURISMOEscuela de  
organización  
industrial

Unión Europea  
Fondo Social Europeo  
Iniciativa de Empleo Juvenil  
El FSE invierte en tu futuro





Público



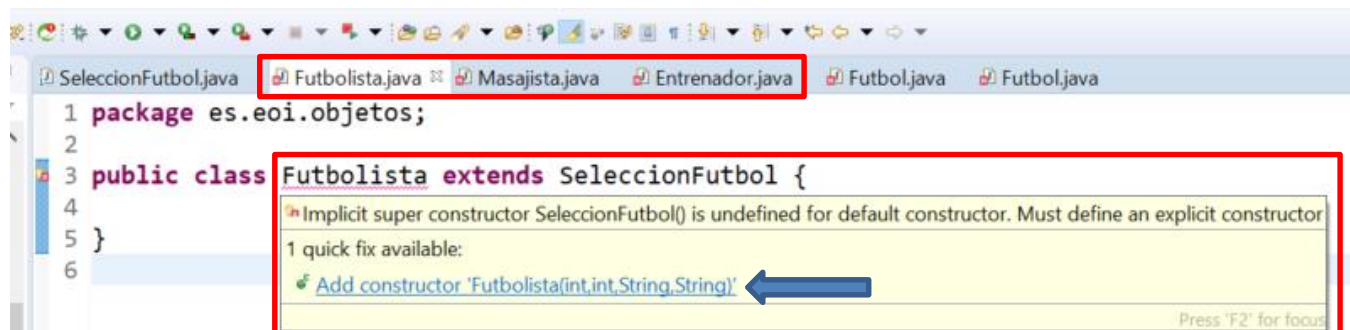
Protegido



Privado

## Herencia

```
public class SeleccionFutbol {  
    protected int id;  
    protected int edad;  
    protected String nombre;  
    protected String apellidos;  
  
    public SeleccionFutbol(int id, int edad, String nombre, String apellidos) {  
        this.id = id;  
        this.edad = edad;  
        this.nombre = nombre;  
        this.apellidos = apellidos;  
    }  
}
```





Público



Protegido



Privado

## Herencia

```
public class Futbolista extends SeleccionFutbol {  
    public Futbolista(int id, int edad, String nombre, String apellidos) {  
        super(id, edad, nombre, apellidos);  
    }  
}  
  
public class Masajista extends SeleccionFutbol {  
    public Masajista(int id, int edad, String nombre, String apellidos) {  
        super(id, edad, nombre, apellidos);  
    }  
}  
  
public class Entrenador extends SeleccionFutbol {  
    public Entrenador(int id, int edad, String nombre, String apellidos) {  
        super(id, edad, nombre, apellidos);  
    }  
}
```

Super = Superclass

GOBIERNO  
DE ESPAÑAMINISTERIO  
DE INDUSTRIA, COMERCIO  
Y TURISMOEscuela de  
organización  
industrialUnión Europea  
Fondo Social Europeo  
Iniciativa de Empleo Juvenil  
El FSE invierte en tu futuro



Público



Protegido



Privado

## Herencia

```
SeleccionFutbol seleccion = new SeleccionFutbol(1, 0, null, null);  
Futbolista futbolista1 = new Futbolista(2, 30, "Leo", "Messi");  
Futbolista futbolista2 = new Futbolista(2, 30, "Cristiano", "Ronaldo");  
Entrenador entrenador1 = new Entrenador(3, 50, "Ronald", "Koeman");  
Entrenador entrenador2 = new Entrenador(3, 50, "Zinedine", "Zidane");  
Masajista masajista1 = new Masajista(4, 25, "Ángel", "Mur");
```



¿Nombre y  
Apellidos de  
SeleccionFutbol?

GOBIERNO  
DE ESPAÑAMINISTERIO  
DE INDUSTRIA, COMERCIO  
Y TURISMOEscuela de  
organización  
industrial

Unión Europea  
Fondo Social Europeo  
Iniciativa de Empleo Juvenil  
El FSE invierte en tu futuro





Público



Protegido



Privado

## Herencia

```

public class SeleccionFutbol {
    protected int id;
    protected int edad;
    protected String nombre;
    protected String apellidos;

    public SeleccionFutbol(int id, int edad, String nombre, String apellidos) {
        this.id = id;
        this.edad = edad;
        this.nombre = nombre;
        this.apellidos = apellidos;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public int getEdad() {
        return edad;
    }

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

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getApellidos() {
        return apellidos;
    }
}

```

```

SeleccionFutbol seleccion = new SeleccionFutbol(1, 0, null, null);
Futbolista futbolista1 = new Futbolista(2, 30, "Leo", "Messi");
Futbolista futbolista2 = new Futbolista(2, 30, "Cristiano", "Ronaldo");
Entrenador entrenador1 = new Entrenador(3, 50, "Ronald", "Koeman");
Entrenador entrenador2 = new Entrenador(3, 50, "Zinedine", "Zidane");
Masajista masajista1 = new Masajista(4, 25, "Ángel", "Mur");

System.out.println(seleccion.getNombre() + " " + seleccion.getApellidos());
System.out.println(futbolista1.getNombre() + " " + futbolista1.getApellidos());
System.out.println(futbolista2.getNombre() + " " + futbolista2.getApellidos());
System.out.println(entrenador1.getNombre() + " " + entrenador1.getApellidos());
System.out.println(entrenador2.getNombre() + " " + entrenador2.getApellidos());
System.out.println(masajista1.getNombre() + " " + masajista1.getApellidos());

```

```

<terminated> Futbol [Java Application]
null null
Leo Messi
Cristiano Ronaldo
Ronald Koeman
Zinedine Zidane
Ángel Mur

```

GOBIERNO  
DE ESPAÑAMINISTERIO  
DE INDUSTRIA, COMERCIO  
Y TURISMOEscuela de  
organización  
industrial

Unión Europea  
Fondo Social Europeo  
Iniciativa de Empleo Juvenil  
El FSE invierte en tu futuro





Público



Protegido



Privado

## Override

SeleccionFutbol.java Futbolista.java Masajista.java Entrenador.java Futbol.java

```
55
56= @Override
57 public String toString() {
58     return "¡Soy una Selección Futbol!";
59 }
```

SeleccionFutbol.java Futbolista.java Masajista.java Entrenador.java Futbol.java

```
9= @Override
10 public String toString() {
11     return "¡Soy un Futbolista y me llamo:"+this.nombre+" "+this.apellidos+"!";
12 }
```

SeleccionFutbol.java Futbolista.java Masajista.java Entrenador.java Futbol.java

```
9= @Override
10 public String toString() {
11     return "Hola, soy un Masajista y tengo:"+this.edad+" años.";
12 }
```

SeleccionFutbol.java Futbolista.java Masajista.java Entrenador.java Futbol.java

```
9= @Override
10 public String toString() {
11     return "Soy un Entrenador y me llamo:"+this.nombre+" "+this.apellidos+", gracias.";
12 }
```





Público



Protegido



Privado

## Override

```
SeleccionFutbol seleccion = new SeleccionFutbol(1, 0, null, null);  
Futbolista futbolista1 = new Futbolista(2, 30, "Leo", "Messi");  
Futbolista futbolista2 = new Futbolista(2, 30, "Cristiano", "Ronaldo");  
Entrenador entrenador1 = new Entrenador(3, 50, "Ronald", "Koeman");  
Entrenador entrenador2 = new Entrenador(3, 50, "Zinedine", "Zidane");  
Masajista masajista1 = new Masajista(4, 25, "Ángel", "Mur");
```

```
System.out.println(seleccion);  
System.out.println(futbolista1);  
System.out.println(futbolista2);  
System.out.println(entrenador1);  
System.out.println(entrenador2);  
System.out.println(masajista1);
```

Console

```
<terminated> Futbol [Java Application] C:\ej_ico_0\Java\jdk1.8.0_144\bin\javaw.exe (14 nc  
¡Soy una Selección Futbol!  
¡Soy un Futbolista y me llamo:Leo Messi!  
¡Soy un Futbolista y me llamo:Cristiano Ronaldo!  
Soy un Entrenador y me llamo:Ronald Koeman, gracias.  
Soy un Entrenador y me llamo:Zinedine Zidane, gracias.  
Hola, soy un Masajista y tengo:25 años.
```

GOBIERNO  
DE ESPAÑAMINISTERIO  
DE INDUSTRIA, COMERCIO  
Y TURISMOEscuela de  
organización  
industrial

Unión Europea  
Fondo Social Europeo  
Iniciativa de Empleo Juvenil  
El FSE invierte en tu futuro





02

## Clase Abstracta



GOBIERNO  
DE ESPAÑA

MINISTERIO  
DE INDUSTRIA, COMERCIO  
Y TURISMO



Escuela de  
organización  
industrial

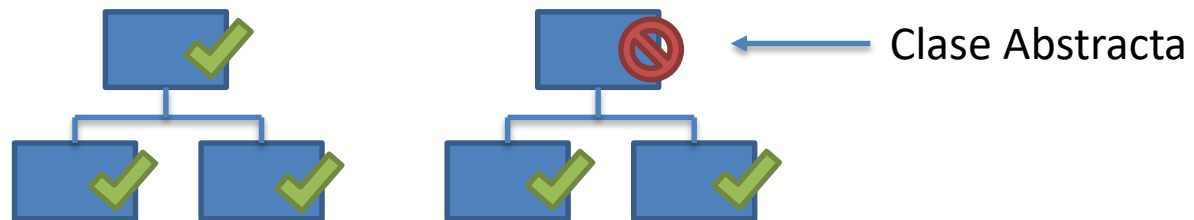


**Unión Europea**  
**Fondo Social Europeo**  
**Iniciativa de Empleo Juvenil**  
El FSE invierte en tu futuro



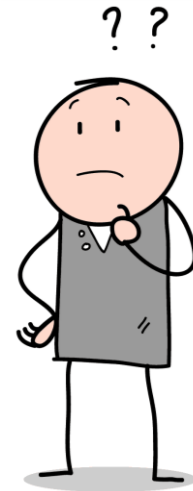
## Clase Abstracta

- Hemos comprobado de primera mano que la herencia nos puede ayudar a crear clases que comparten ciertos atributos o incluso ciertos métodos, lo que nos permite ahorrarnos código.
- En ocasiones nos puede interesar que únicamente se puedan instanciar las clases hijas, y prohibir la instanciación de las clases padre, un motivo puede ser que la clase padre únicamente sirve para agrupar atributos y/o métodos y no nos interese tener objetos del tipo de la clase padre.



## Clase Abstracta

```
SeleccionFutbol seleccion = new SeleccionFutbol(1, 0, null, null);  
Futbolista futbolista1 = new Futbolista(2, 30, "Leo", "Messi");  
Futbolista futbolista2 = new Futbolista(2, 30, "Cristiano", "Ronaldo");  
Entrenador entrenador1 = new Entrenador(3, 50, "Ronald", "Koeman");  
Entrenador entrenador2 = new Entrenador(3, 50, "Zinedine", "Zidane");  
Masajista masajista1 = new Masajista(4, 25, "Ángel", "Mur");
```



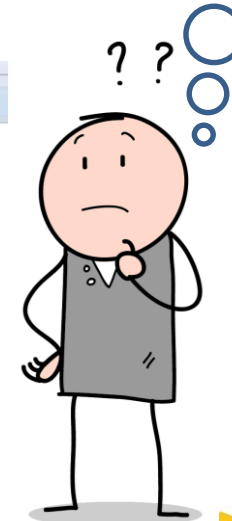
¿SeleccionFutbol  
es abstracta?

## Clase Abstracta

```
public abstract class SeleccionFutbol {  
    protected int id;  
    protected int edad;  
    protected String nombre;  
    protected String apellidos;
```

```
SeleccionFutbol.java  Futbolista.java  Masajista.java  Entrenador.java  Futbol.java x  
3 public class Futbol {  
4  
5     public static void main(String[] args) {  
6  
7         SeleccionFutbol seleccion = new SeleccionFutbol(1, 0, null, null);  
8         Futbolista futbolista1 = new Futbolista(1, 0, null, null, "Ronaldo");  
9         Futbolista futbolista2 = new Futbolista(2, 0, null, null, "Ronaldo");  
10        Entrenador entrenador1 = new Entrenador(3, 50, "Ronald", "Koeman");  
11        Entrenador entrenador2 = new Entrenador(3, 50, "Zinedine", "Zidane");  
12        Masajista masajista1 = new Masajista(4, 25, "Ángel", "Mur");
```

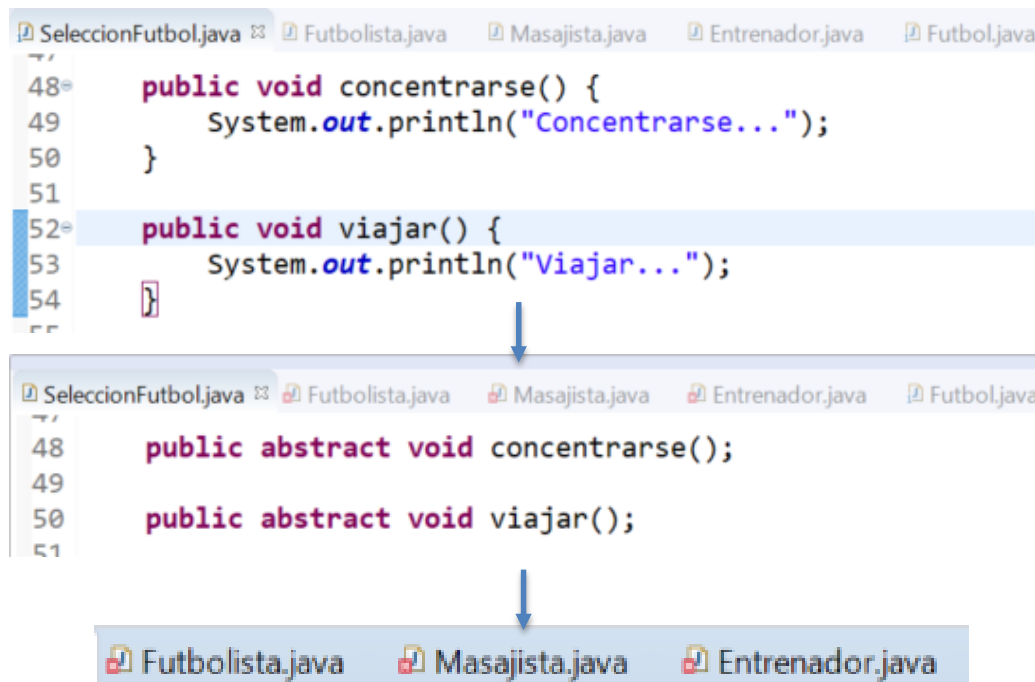
¿SeleccionFutbol  
es abstracta?



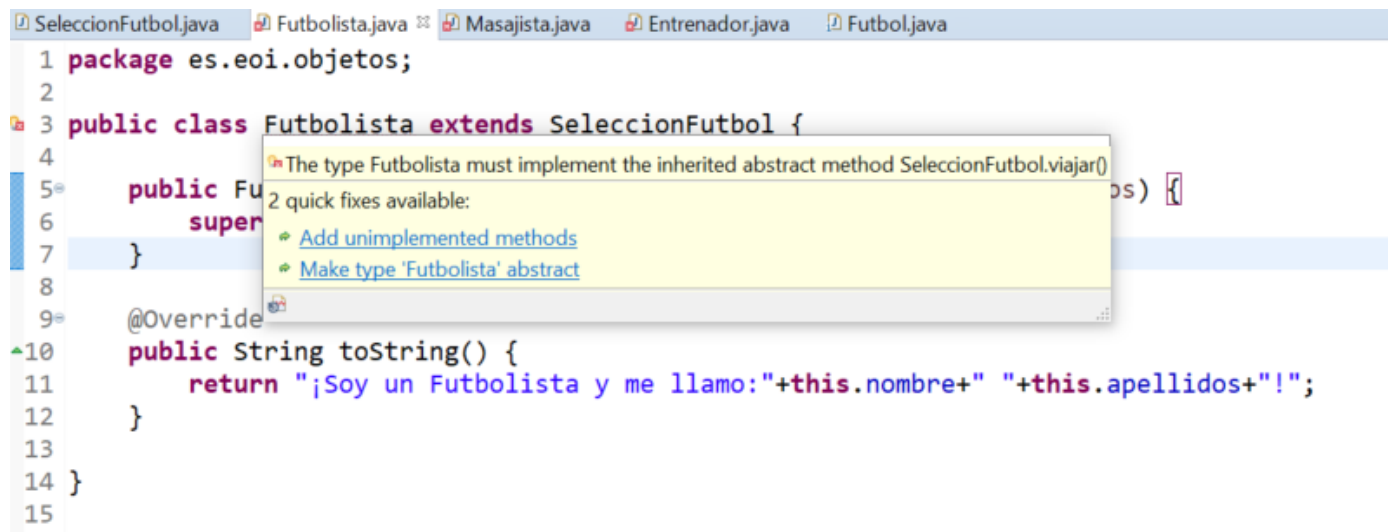
## Clase Abstracta

```
public static void main(String[] args) {  
  
    SeleccionFutbol seleccion = new SeleccionFutbol(1, 0, null, null);  
    Futbolista futbolista1 = new Futbolista(2, 30, "Leo", "Messi");  
    Futbolista futbolista2 = new Futbolista(2, 30, "Cristiano", "Ronaldo");  
    Entrenador entrenador1 = new Entrenador(3, 50, "Ronald", "Koeman");  
    Entrenador entrenador2 = new Entrenador(3, 50, "Zinedine", "Zidane");  
    Masajista masajista1 = new Masajista(4, 25, "Ángel", "Mur");  
  
    System.out.println(futbolista1);  
    System.out.println(futbolista2);  
    System.out.println(entrenador1);  
    System.out.println(entrenador2);  
    System.out.println(masajista1);  
}
```

## Métodos Abstractos



## Métodos Abstractos



The screenshot shows a Java IDE with several tabs: SeleccionFutbol.java, Futbolista.java, Masajista.java, Entrenador.java, and Futbol.java. The active file is Futbolista.java, which contains the following code:

```
1 package es.eoi.objetos;
2
3 public class Futbolista extends SeleccionFutbol {
4
5     public Futbolista() {
6         super();
7     }
8
9     @Override
10    public String toString() {
11        return "¡Soy un Futbolista y me llamo:" + this.nombre + " " + this.apellidos + "!";
12    }
13
14 }
15
```

An IDE error message is displayed over the code, stating: "The type Futbolista must implement the inherited abstract method SeleccionFutbol.viajar()". It offers two quick fixes: "Add unimplemented methods" and "Make type 'Futbolista' abstract".



Las clases o métodos abstractos funcionan como una clase que declara la existencia de métodos pero no su implementación

## Métodos Abstractos



```
SeleccionFutbol.java  Futbolista.java  Masajista.java  Entrenador.java  Futbol.java

13
14 @Override
15 public void concentrarse() {
16     // TODO Auto-generated method stub
17
18 }
19
20 @Override
21 public void viajar() {
22     // TODO Auto-generated method stub
23
24 }
```

03

## Herencia



GOBIERNO  
DE ESPAÑA

MINISTERIO  
DE INDUSTRIA, COMERCIO  
Y TURISMO



Escuela de  
organización  
industrial



**Unión Europea**  
**Fondo Social Europeo**  
**Iniciativa de Empleo Juvenil**  
El FSE invierte en tu futuro



## Herencia

- Las clases hijas o subclases, contienen implícitamente todos los métodos y propiedades de la clase Padre que sean **public** o **protected**, y en caso de ambas pertenecer al mismo paquete, también los métodos y propiedades con modificador de acceso por defecto.

```
public class Futbolista extends SeleccionFutbol {  
    private int dorsal;  
  
    public Futbolista(int id, int edad, String nombre, String apellidos) {  
        super(id, edad, nombre, apellidos);  
    }  
  
    public Futbolista(int id, int edad, String nombre, String apellidos, int dorsal) {  
        super(id, edad, nombre, apellidos);  
        this.dorsal = dorsal;  
    }  
}
```



GOBIERNO  
DE ESPAÑA

MINISTERIO  
DE INDUSTRIA, COMERCIO  
Y TURISMO



Escuela de  
organización  
industrial



Unión Europea  
Fondo Social Europeo  
Iniciativa de Empleo Juvenil  
El FSE invierte en tu futuro



## Herencia

- A diferencia de lo que ocurre con los métodos y atributos no privados, los constructores no se heredan, las subclases deberán invocar al constructor de la superclase que nos interese.
- Java automáticamente inserta llamadas al constructor por defecto, sin parámetros de la superclase en el constructor de la subclase, siempre que exista.
- Si no existiese constructor sin argumentos en la superclase, hay que invocar a algún constructor de la superclase en cada constructor de las subclases (con super) o se obtendrá un error de compilación.

## This y Super

### This

Al acceder a variables de instancia de una clase, la palabra clave `this` hace referencia a los miembros de la propia clase en el objeto actual; es decir, `this` se refiere al objeto actual sobre el que está actuando un método determinado y se utiliza siempre que se quiera hacer referencia al objeto actual de la clase.

### Super

Si se necesita llamar al método padre dentro de una clase que ha reemplazado ese método, se puede hacer referencia al método padre con la palabra clave `super`.

04

## Ejercicios



GOBIERNO  
DE ESPAÑA

MINISTERIO  
DE INDUSTRIA, COMERCIO  
Y TURISMO



Escuela de  
organización  
industrial



**Unión Europea**  
**Fondo Social Europeo**  
**Iniciativa de Empleo Juvenil**  
El FSE invierte en tu futuro



## Ejercicio I

Vamos a crear una clase `Animal.java` esta clase tendrá las propiedades `nombre`, `peso` y `altura`, crearemos las clases `Perro.java`, `Gato.java` y `Raton.java`, cada una de estas clases tendrá las tres propiedades anteriores y además el perro podrá ladrar, el gato maullar y el ratón roer.

Debemos tener en cuenta que todos los animales pueden olfatear, deberemos tener en cuenta que no queremos que se puedan crear objetos directamente de la clase `Animal`, tan solo de las clases `Perro`, `Gatos` y `Raton`.

A continuación desde una clase `main` llamada `AnimalesDomesticos.java` almacenaremos 5 perros, 5 gatos y 5 ratones en sus respectivos arrays, por último recorreremos cada array e imprimiremos el siguiente mensaje por cada animal:

**{nombre}** es un **{Perro/Gato/Ratón}** pesa **{peso}** y mide **{altura}**.



## Ejercicio II

Crearemos las clases Rectangulo.java y Cuadrado.java, en ambas clases almacenaremos altura y anchura y símbolo (carácter), debemos tener en cuenta que a la hora de crear cuadrados la altura debe ser igual a la anchura, ambas clases tendrán un método dibujar()

En una clase llamada MundoGeometrico.java definiremos un array de rectángulos y otro de cuadrados, dónde almacenaremos 5 figuras geométricas de cada clase, por último recorreremos cada elemento del array y dibujaremos la figura geométrica que corresponda por consola.

Ejemplo: Cuadrado N°1: (2x2)

##

##

\*Podemos ayudarnos de superclases si lo creemos oportuno.