

# **Especificadores de acceso (Visibilidad)**

Dpto. Lenguajes y Sistemas Informáticos  
Universidad de Granada

**Programación y Diseño Orientado a Objetos**

(Curso 2019-2020)

# Créditos

- Las siguientes imágenes e ilustraciones son libres y se han obtenido de:
  - ▶ Emojis, <https://pixabay.com/images/id-2074153/>
- El resto de imágenes e ilustraciones son de creación propia, al igual que los ejemplos de código

# Objetivos

- Entender el propósito de los especificadores de acceso
- Comprender cómo afectan los especificadores de acceso a métodos y atributos
- Saber usarlos en Java y Ruby
  - ▶ (y no confundirse con las diferencias existentes en cada lenguaje)

# Contenidos

- 1 **Propósito de los especificadores de acceso**
- 2 **Especificadores de acceso en Java**
  - Ejemplos
- 3 **Especificadores de acceso en Ruby**
  - Ejemplos

# Propósito de los especificadores de acceso

- Permiten **restringir el acceso** a atributos y métodos
- **Ocultan detalles de la implementación** para que los objetos sean usados a través de una interfaz concreta
- Suele ser aconsejable **usar el nivel más restrictivo posible**  
→ *Diseño* ←
- Dependiendo del lenguaje también pueden ser aplicados a otros elementos como las clases

# Especificadores de acceso habituales

- Los especificadores de acceso habituales son:
  - ▶ Privado
  - ▶ Protegido
  - ▶ Público
- Según el lenguaje pueden existir otros, por ejemplo:
  - ▶ Java añade un especificador más: Paquete
  - ▶ Smalltalk solo tiene Público y Protegido
- **Atención:** Hay diferencias importantes en su significado dependiendo del lenguaje

# Especificadores de acceso en Java

- Permite establecerlos a atributos y métodos
  - ▶ Cada elemento debe incluir el suyo
- Particularidades del especificador **private**
  - ▶ Solo es **accesible desde código de la propia clase** (ya sea desde **ámbito de instancia o de clase**)
    - ★ Desde el ámbito de instancia se puede acceder a elementos de clase privados de la misma clase
  - ▶ Se puede acceder a elementos privados de otra **instancia distinta** si es **de la misma clase** (tanto desde ámbito de instancia como de clase)
    - ★ Esa otra instancia distinta ha podido recibirse como parámetro en un método (de instancia o de clase)



# Especificadores de acceso en Java

- Particularidades del especificador *de paquete*
  - ▶ No poner ningún especificador significa visibilidad de paquete
  - ▶ Estos elementos son **públicos dentro del paquete**
  - ▶ y **privados respecto al exterior del paquete**



# Especificadores de acceso en Java

- Particularidades del especificador **protected**
  - ▶ Estos elementos son **públicos dentro del mismo paquete**
    - ★ Son accesibles desde el mismo paquete  
(con independencia de la relación de herencia que exista (o no) entre las clases involucradas)
  - ▶ También son **accesibles desde subclases de otros paquetes**
    - ★ Dentro de una misma instancia, se podrá acceder a elementos protegidos definidos en cualquiera de sus superclases  
(con independencia del paquete en el que estén las clases involucradas)

# Especificadores de acceso en Java

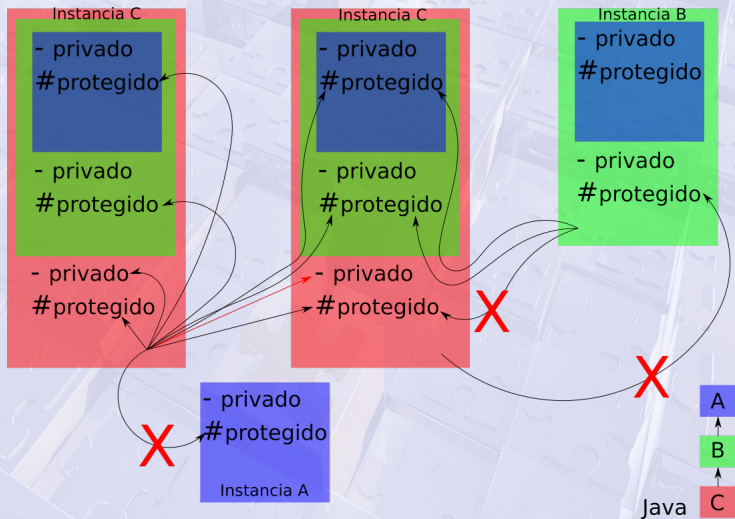
- Particularidades del especificador **protected** (continuación)
- Para poder acceder a **elementos protegidos de una instancia distinta:**
  - (tanto desde ámbito de clase como de instancia)
  - ▶ Esa instancia tiene que ser de la misma clase que la propietaria del código desde el que se realiza el acceso o de una subclase de la misma
    - ★ Es decir, *esa instancia debe ser-un yo*
  - ▶ El elemento accedido tiene que estar declarado en la clase propietaria del código desde el que se realiza el acceso o en una superclase de la misma
    - ★ Es decir, *el elemento debe ser visible por mí*
  - ▶ **Recordar:** Si las clases involucradas están en el mismo paquete, los elementos protegidos son accesibles siempre

(lo acabamos de ver en la página anterior)

# Especificadores de acceso de las clases Java

- Las propias clases Java podrán ser:
  - ▶ Públicas: `public`  
Son utilizables desde cualquier sitio
  - ▶ De paquete: *no se indica ningún especificador de acceso*  
Son solo utilizables dentro del paquete en las que se definen

# Especificadores de acceso en Java: Resumen



# Especificadores de acceso en Java: Ejemplos

## Java: Acceso a elementos de otra instancia de la misma clase

```
1 package unPaquete;  
2  
3 public class Padre {  
4     private int privado;  
5     protected int protegido;  
6     int paquete;  
7     public int publico;  
8  
9     public void testInstanciaPadre (Padre o) {  
10        System.out.println (o.privado);  
11        System.out.println (o.protegido);  
12        System.out.println (o.paquete);  
13        System.out.println (o.publico);  
14    }  
15  
16    public static void testClasePadre (Padre o) {  
17        System.out.println (o.privado);  
18        System.out.println (o.protegido);  
19        System.out.println (o.paquete);  
20        System.out.println (o.publico);  
21    }  
22 }
```

★ ¿Qué ocurre en cada línea? ¿Algún error?

# Especificadores de acceso en Java: Ejemplos

## Java: Acceso a instancia de la superclase desde el mismo paquete

```
1 package unPaquete;  
2  
3 public class HijaPaquete extends Padre{  
4  
5     public void testInstanciaHijaPaquete (Padre o) {  
6         System.out.println (privado);  
7         System.out.println (o.privado);  
8  
9         System.out.println (protegido);  
10        System.out.println (o.protegido);  
11  
12        System.out.println (o.paquete);  
13        System.out.println (o.publico);  
14    }  
15  
16    public static void testClaseHijaPaquete (Padre o) {  
17        System.out.println (o.privado);  
18  
19        System.out.println (o.protegido);  
20  
21        System.out.println (o.paquete);  
22        System.out.println (o.publico);  
23    }  
24 }
```

★ ¿Qué ocurre en cada línea? ¿Algún error?

# Especificadores de acceso en Java: Ejemplos

## Java: Acceso a instancia de la superclase desde otro paquete

```
1 package otroPaquete;  
2  
3 public class HijaOtroPaquete extends Padre {  
4  
5     public void testInstanciaHijaOtroPaquete (Padre o){  
6         // Acceso a elementos heredados  
7         System.out.println (privado);  
8         System.out.println (paquete);  
9         System.out.println (protegido);  
10  
11         // Acceso a elementos de otra instancia  
12         System.out.println (o.privado);  
13         System.out.println (o.protegido);  
14         System.out.println (o.paquete);  
15         System.out.println (o.publico);  
16     }  
17  
18     public static void testClaseHijaOtroPaquete (Padre o){  
19         // Acceso a elementos de otra instancia  
20         System.out.println (o.privado);  
21         System.out.println (o.protegido);  
22         System.out.println (o.paquete);  
23         System.out.println (o.publico);  
24     }  
25 }
```

★ ¿Qué ocurre en cada línea? ¿Algún error?



# Especificadores de acceso en Java: Ejemplos

**Java:** Acceso a instancia de la misma clase,  
los elementos heredados se declaran en otro paquete

```
1 package otroPaquete;  
2  
3 public class HijaOtroPaquete extends Padre {  
4  
5     // El mismo código cambiando solo el tipo del parámetro  
6  
7     public void testInstanciaHijaOtroPaquete (HijaOtroPaquete o) {  
8         System.out.println(o.privado);  
9         System.out.println(o.protegido);  
10        System.out.println(o.paquete);  
11        System.out.println(o.publico);  
12    }  
13  
14    public static void testClaseHijaOtroPaquete (HijaOtroPaquete o) {  
15        System.out.println(o.privado);  
16        System.out.println(o.protegido);  
17        System.out.println(o.paquete);  
18        System.out.println(o.publico);  
19    }  
20 }
```

★ ¿Qué ocurre en cada línea? ¿Algún error?

# Especificadores de acceso en Java: Ejemplos

**Java:** Acceso a instancia de subclase  
los elementos heredados se declaran en otro paquete

```
1 package otroPaquete;  
2  
3 public class HijaOtroPaquete extends Padre {  
4  
5     // El mismo código. El tipo del parámetro es subclase.  
6  
7     public void testInstanciaHijaOtroPaquete (NietaOtroPaquete o) {  
8         System.out.println(o.privado);  
9         System.out.println(o.protegido);  
10        System.out.println(o.paquete);  
11        System.out.println(o.publico);  
12    }  
13  
14    public static void testClaseHijaOtroPaquete (NietaOtroPaquete o) {  
15        System.out.println(o.privado);  
16        System.out.println(o.protegido);  
17        System.out.println(o.paquete);  
18        System.out.println(o.publico);  
19    }  
20 }  
21  
22 // NietaOtroPaquete deriva de HijaOtroPaquete (ambas están en otroPaquete)
```

★ ¿Qué ocurre en cada línea?    ¿Algún error?

# Especificadores de acceso en Java: Ejemplos

**Java:** Acceso a instancia de la superclase  
los elementos heredados se declaran en otro paquete

```
1 package otroPaquete;  
2  
3 public class NietaOtroPaquete extends HijaOtroPaquete {  
4  
5     // Ahora probamos con un parámetro de la superclase  
6  
7     public void testInstanciaNietaOtroPaquete (HijaOtroPaquete o) {  
8         System.out.println (o.privado);  
9         System.out.println (o.protegido);  
10        System.out.println (o.paquete);  
11        System.out.println (o.publico);  
12    }  
13  
14    public static void testClaseNietaOtroPaquete (HijaOtroPaquete o) {  
15        System.out.println (o.privado);  
16        System.out.println (o.protegido);  
17        System.out.println (o.paquete);  
18        System.out.println (o.publico);  
19    }  
20 }
```

★ ¿Qué ocurre en cada línea? ¿Algún error?

# Especificadores de acceso en Java: Ejemplos

**Java:** Acceso a instancia de la superclase  
los elementos heredados se declaran en otro paquete

```
1 package otroPaquete;  
2  
3 public class NietaOtroPaquete extends HijaOtroPaquete{  
4  
5     public void testInstanciaNietaOtroPaquete (NietaOtroPaquete o) {  
6         System.out.println (o.privado);  
7         System.out.println (o.protegido);  
8         System.out.println (o.paquete);  
9         System.out.println (o.publico);  
10    }  
11  
12    public static void testClaseNietaOtroPaquete (NietaOtroPaquete o) {  
13        System.out.println (o.privado);  
14        System.out.println (o.protegido);  
15        System.out.println (o.paquete);  
16        System.out.println (o.publico);  
17    }  
18 }
```

★ ¿Qué ocurre en cada línea? ¿Algún error?

# Especificadores de acceso en Java: Ejemplos

## Java: Acceso a protegidos

```
1 package base;
2
3 public class A {
4     protected int protegidoA = 0;
5 }
6
7 // *****
8
9 public class B extends A{
10     protected int protegidoB = 1;
11 }
12
13 // *****
14
15 import base2.C;
16
17 public class D extends C {
18     protected int protegidoD = 3;
19 }
20 }
```

## Java: Acceso a protegidos

```
1 package base2;
2 import base.*;
3
4 public class C extends B{
5     protected int protegidoC = 2;
6
7     public void test() {
8         A a = new A();
9         a.protegidoA = 666;
10        B b = new B();
11        b.protegidoB = 666;
12        C c = new C();
13        c.protegidoA = 555;
14        D d = new D();
15        d.protegidoA = 555;
16        D d2 = new D();
17        d2.protegidoB = 555;
18        d2.protegidoD = 555;
19        this.protegidoA = 777;
20    }
21 }
```

★ ¿Qué ocurre en cada línea? ¿Algún error?

# Especificadores de acceso en Ruby

- Los **atributos** son **siempre privados**. No se puede cambiar
- Los métodos son por defecto públicos, aunque esto se puede modificar mediante especificadores de acceso
- El método **initialize** es **siempre privado**. No se puede cambiar
- Cuando se utiliza un especificador de acceso, este afecta a todos los elementos declarados a continuación

# Especificadores de acceso en Ruby

- Particularidades del especificador **private**
  - ▶ Un método privado nunca puede ser utilizado mediante un receptor de mensaje explícito (recordar que el receptor implícito siempre es `self`)
  - ▶ Solo se puede utilizar un método privado de la propia instancia
  - ▶ Si B hereda de A
    - ★ Desde un ámbito de instancia de B se puede llamar a métodos de instancia privados de A
    - ★ Desde un ámbito de clase de B se puede llamar a métodos de clase privados de A
  - ▶ No se puede acceder a métodos privados de clase desde el ámbito de instancia
  - ▶ No se puede acceder a métodos privados de instancia desde el ámbito de clase



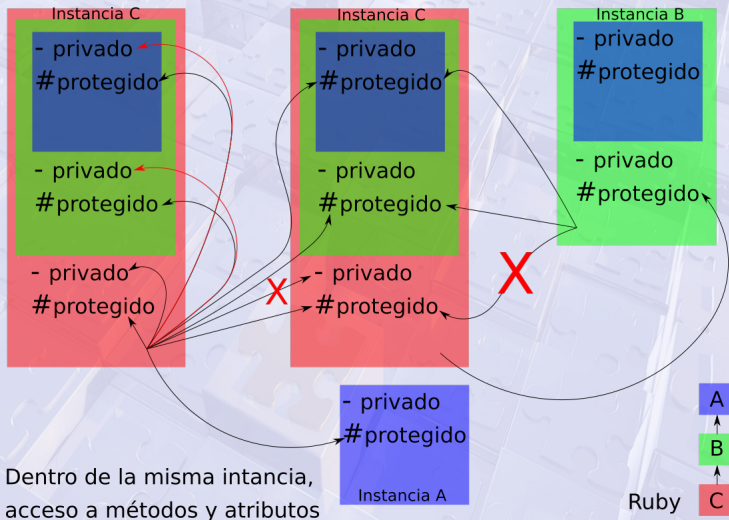
# Especificadores de acceso en Ruby

- Particularidades del especificador **protected**
  - ▶ Los métodos protegidos sí pueden ser invocados con un receptor de mensaje explícito
    - ★ La clase del código que invoca debe ser la misma, o una subclase, de la clase donde se declaró dicho método
  - ▶ No existen métodos protegidos de clase

# Especificadores de acceso en Ruby

- En general, recordar que en Ruby las clases también son objetos a todos los efectos
  - ▶ Una clase y sus instancias **no son de la misma clase**
  - ▶ Como objetos que son, son instancias de clases distintas
- Los atributos de clase (`@@atributo_de_clase`) sí pueden ser accedidos directamente desde el ámbito de instancia

# Especificadores de acceso en Ruby: Resumen



# Especificadores de acceso en Ruby: Ejemplos

## Ruby: Acceso a privados y protegidos

```
1  class Padre
2    private
3    def privado
4    end
5
6    protected
7    def protegido
8    end
9
10   public
11   def publico
12   end
13
14   def test(p)
15     privado
16     self.privado
17     p.privado
18     protegido
19     self.protegido
20     p.protegido
21   end
22 end
```

★ ¿Qué ocurre en cada línea? ¿Algún error?

# Especificadores de acceso en Ruby: Ejemplos

## Ruby: Acceso a privados y protegidos

```
1  # Fuera de cualquier clase
2
3  Padre.new.test(Padre.new)
4  p=Padre.new
5
6  # Receptor explícito fuera de la clase o subclases
7
8  p.privado
9  p.protegido
10 p.publico
```

★ ¿Qué ocurre en cada línea? ¿Algún error?

# Especificadores de acceso en Ruby: Ejemplos

## Ruby: Acceso a privados y protegidos con relación de herencia

```
1  class Hija < Padre
2    def test(p)
3      privado
4      self.privado
5      p.privado
6      protegido
7      self.protegido
8      p.protegido
9      publico
10     self.publico
11     p.publico
12   end
13 end
14
15 # Fuera de cualquier clase
16
17 Hija.new.test(Hija.new)
18 Hija.new.test(Padre.new)
19 h=Hija.new
20 h.privado
21 h.protegido
22 h.publico
```

★ ¿Qué ocurre en cada línea? ¿Algún error?

# Especificadores de acceso en Ruby: Ejemplos

## Ruby: Métodos privados de instancia y de clase

```
1  class Padre
2    private
3    def privado_instancia
4    end
5
6    def self.privado_clase # Por ahora este método es público
7    end
8
9    private_class_method :privado_clase # Atención a la sintaxis
10
11   public
12   def test
13     self.class.privado_clase
14   end
15
16   def self.test(p)
17     p.privado_instancia
18   end
19 end
20
21 # Fuera de cualquier clase
22
23 Padre.new.test
24 Padre.test(Padre.new)
```

★ ¿Qué ocurre en cada línea? ¿Algún error?



# Especificadores de acceso en Ruby: Ejemplos

## Ruby: Variables de clase y de instancia de la clase

```
1  class Padre
2    @instanciaDeClase = 1
3    @duda = 2
4    @@deClase = 11
5    @@duda = 22
6
7    def initialize
8      @deInstancia = 333
9      @duda = 444
10   end
11
12   def self.salida
13     puts @instanciaDeClase+1
14     puts @duda+1 unless @duda.nil? # desde Hija?
15     puts @@deClase+1
16     puts @@duda+1
17   end
18
19   def salida
20     puts @deInstancia+1
21     puts @duda+1
22     puts @@deClase+1
23     puts @@duda+1
24   end
25 end
```

★ ¿Qué ocurre en cada línea? ¿Algún error?

# Especificadores de acceso en Ruby: Ejemplos

## Ruby: Variables de clase y de instancia de la clase

```
1  class Hija < Padre
2    @instanciaDeClase = -1
3
4    # Sobreescribimos el valor fijado anteriormente
5    # Este atributo es compartido
6    @@deClase = -11
7
8    def modifica
9      # Acceso a los atributos definidos en Padre
10     @duda += 111
11   end
12 end
13
14 # Fuera de cualquier clase
15
16 Padre.salida
17 Hija.salida # Atención a lo que ocurre con la segunda línea
18
19 p = Padre.new
20 p.salida
21 h = Hija.new
22 h.salida
23
24 h.modifica
25 h.salida
```

★ ¿Qué ocurre en cada línea? ¿Algún error?

# Especificadores de acceso en Ruby: Ejemplos

## Ruby: Relaciones de varias clases en cadena

```
1  class A
2    protected
3    def protegidoA
4    end
5  end
6
7  class B < A
8    protected
9    def protegidoB
10   end
11 end
```

★ ¿Qué ocurre en cada línea? ¿Algún error?

# Especificadores de acceso en Ruby: Ejemplos

## Ruby: Relaciones de varias clases en cadena

```
1  class C < B
2    protected
3    def protegidoC
4    end
5
6    public
7    def test
8      A.new.protegidoA
9      B.new.protegidoA
10     B.new.protegidoB
11     C.new.protegidoA
12     C.new.protegidoB
13     C.new.protegidoC
14     D.new.protegidoA
15     D.new.protegidoD
16   end
17 end
18
19 class D < C
20   protected
21   def protegidoD
22   end
23 end
24
25 C.new.test
26 C.new.protegidoC
```

★ ¿Qué ocurre en cada línea? ¿Algún error?

# Especificadores de acceso en Ruby: Ejemplos

## Ruby: Falsa seguridad

```
1  class FalsaSeguridad
2    # Un consultor puede ser muy peligroso
3    attr_reader :lista
4
5    def initialize
6      @lista = [1,2,3,4]
7    end
8
9    def info
10     puts @lista.size
11   end
12
13 end
14
15 # Fuera de cualquier clase
16 f = FalsaSeguridad.new
17 f.info #4
18
19 # Modificamos el estado interno
20 # Simplemente usando un consultor
21 # Aunque el atributo sea privado
22 # Cuidado con las referencias
23 f.lista.clear
24
25 f.info #0
```

★ ¿Qué ocurre en cada línea? ¿Algún error?

# Especificadores de acceso

→ **Diseño** ←

- Se aconseja usar el nivel más restrictivo posible
- Un atributo privado con un consultor público que devuelve una referencia puede ser modificado *“desde fuera”*
  - Como ya se comentó en el módulo de consultores y modificadores:
    - ▶ Hay que evaluar cada caso y decidir qué se devuelve (o asigna)
      - ★ Una referencia o una copia
      - ★ Dependiendo de a quién se le dé, o de dónde venga

# **Especificadores de acceso (Visibilidad)**

Dpto. Lenguajes y Sistemas Informáticos  
Universidad de Granada

**Programación y Diseño Orientado a Objetos**

(Curso 2019-2020)