Especificadores de acceso (Visibilidad)

Dpto. Lenguajes y Sistemas Informáticos
Universidad de Granada

Programación y Diseño Orientado a Objetos

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

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

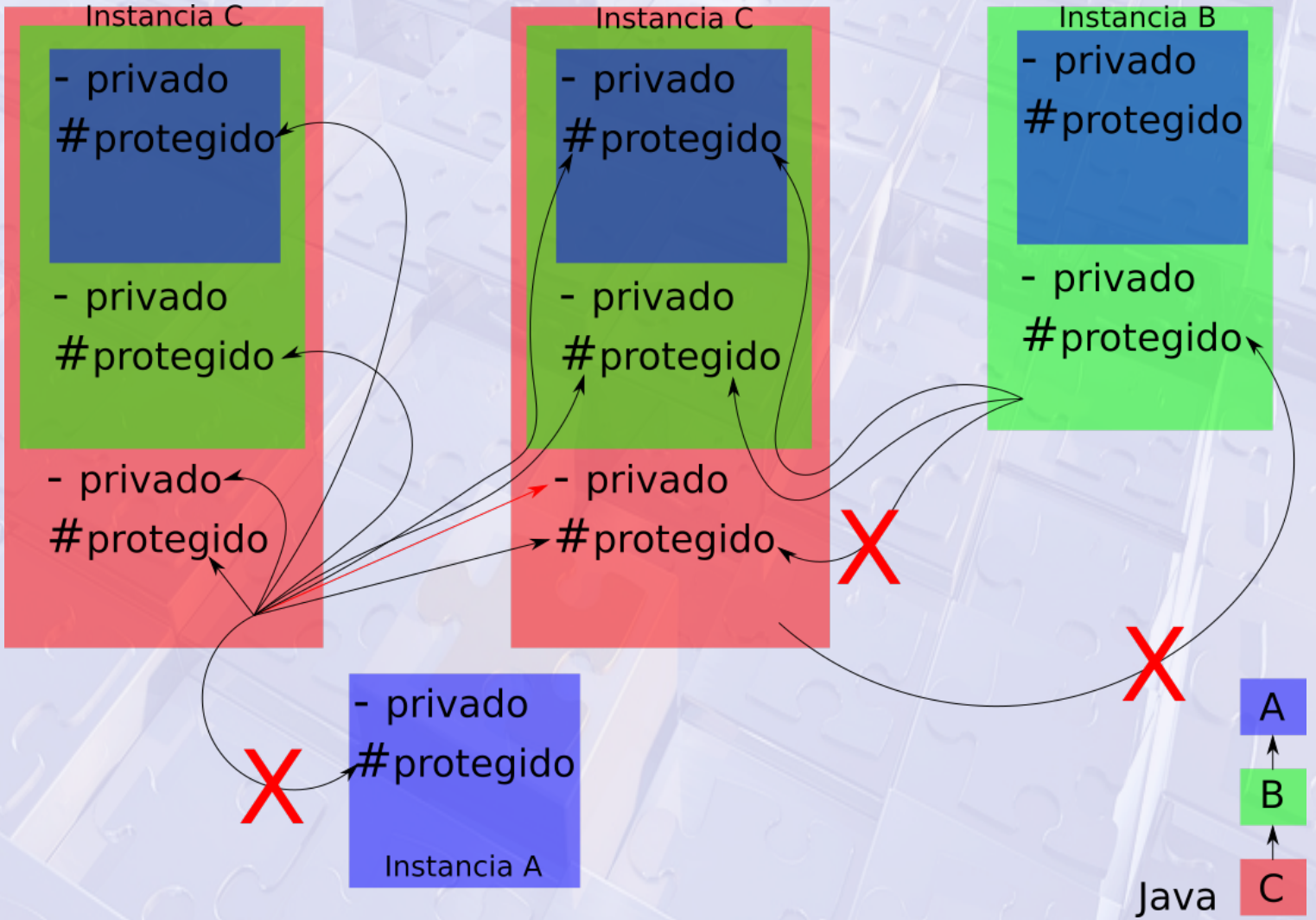
- Las propias clases Java podrán ser:
  - Públicas: `public`

    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   public void test() {
7     A a = new A();
8     a.protegidoA = 666;
9     B b = new B();
10    b.protegidoB = 666;
11    C c = new C();
12    c.protegidoA = 555;
13    D d = new D();
14    d.protegidoA = 555;
15    D d2 = new D();
16    d2.protegidoB = 555;
17    d2.protegidoD = 555;
18    this.protegidoA = 777;
19  }
20 }
```

★ ¿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
    A partir de Ruby 2.7 (diciembre 2019) sí se permite `self` como receptor de mensaje explícito.
  - 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

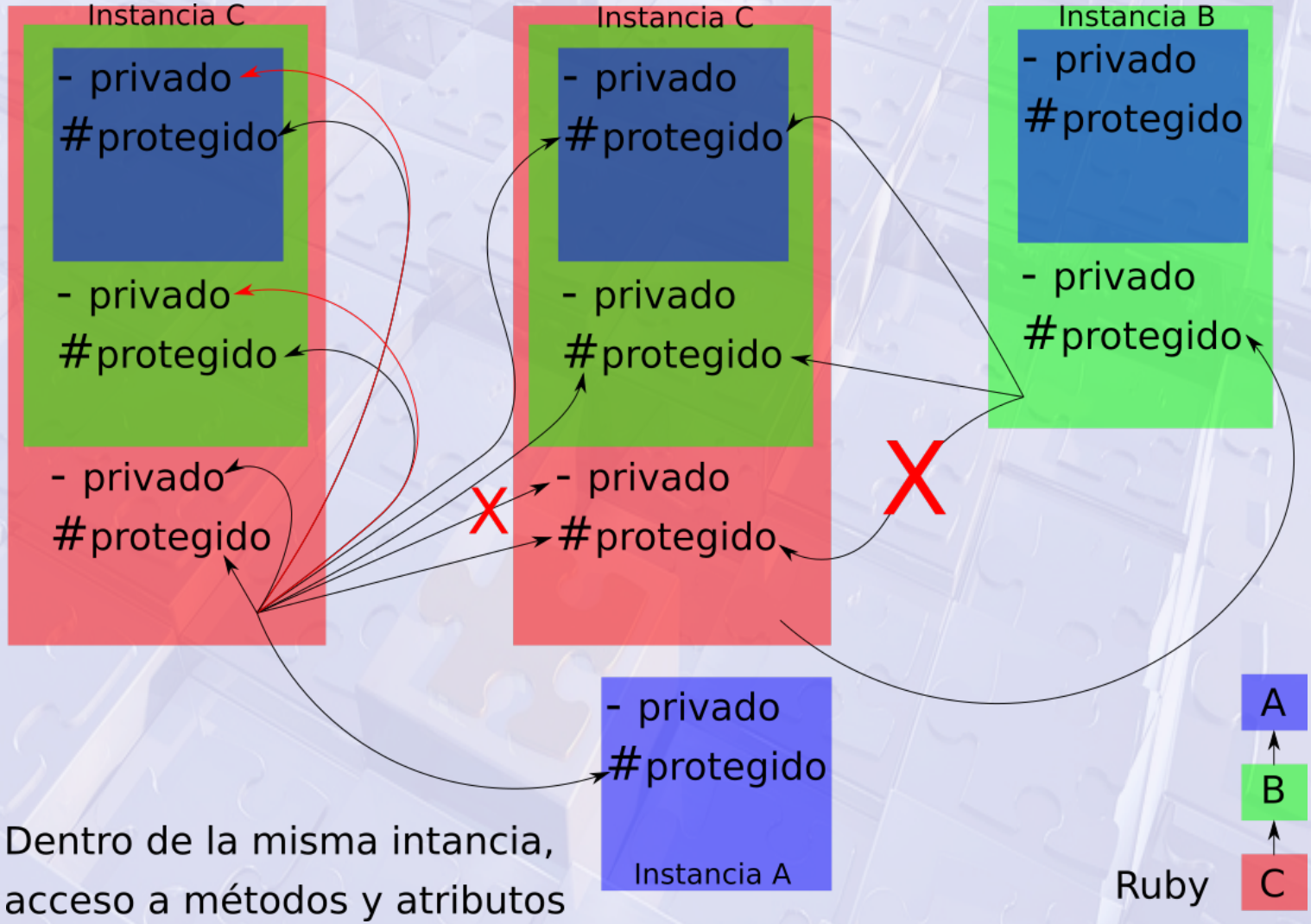
Especificadores de acceso en Ruby

- 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

```ruby
class Padre
  private
  def privado
  end

  protected
  def protegido
  end

  public
  def publico
  end

  def test(p)
    privado
    self.privado #Correcto solo a partir de Ruby 2.7
    p.privado
    protegido
    self.protegido
    p.protegido
  end
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

```ruby
class Hija < Padre
  def test(p)
    privado
    self.privado #Correcto solo a partir de Ruby 2.7
    p.privado
    protegido
    self.protegido
    p.protegido
    publico
    self.publico
    p.publico
  end
end

# Fuera de cualquier clase

Hija.new.test(Hija.new)
Hija.new.test(Padre.new)
h=Hija.new
h.privado
h.protegido
h.publico
```

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

Especificadores de acceso en Ruby: Ejemplos

Ruby: Métodos privados de instancia y de clase

```ruby
class Padre
  private
  def privado_instancia
  end

  def self.privado_clase # Por ahora este método es público
  end

  private_class_method :privado_clase # Atención a la sintaxis

  public
  def test
    self.class.privado_clase
  end

  def self.test(p)
    p.privado_instancia
  end
end

# Fuera de cualquier clase

Padre.new.test
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

```ruby
class Padre
  @instanciaDeClase = 1
  @duda = 2
  @@deClase = 11
  @@duda = 22

  def initialize
    @deInstancia = 333
    @duda = 444
  end

  def self.salida
    puts @instanciaDeClase+1
    puts @duda+1 unless @duda.nil? # desde Hija?
    puts @@deClase+1
    puts @@duda+1
  end

  def salida
    puts @deInstancia+1
    puts @duda+1
    puts @@deClase+1
    puts @@duda+1
  end
end
```

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

Especificadores de acceso en Ruby: Ejemplos

Ruby: Variables de clase y de instancia de la clase

```ruby
class Hija < Padre
  @instanciaDeClase = -1

  # Sobreescribimos el valor fijado anteriormente
  # Este atributo es compartido
  @@deClase = -11

  def modifica
    # Acceso a los atributos definidos en Padre
    @duda += 111
  end
end

# Fuera de cualquier clase

Padre.salida
Hija.salida # Atención a lo que ocurre con la segunda línea

p = Padre.new
p.salida
h = Hija.new
h.salida

h.modifica
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    public
6      def test
7        A.new.protegidoA
8        B.new.protegidoA
9        B.new.protegidoB
10       C.new.protegidoA
11       C.new.protegidoB
12       C.new.protegidoC
13       D.new.protegidoA
14       D.new.protegidoD
15     end
16 end
17 class D < C
18   protected
19   def protegidoD
20   end
21 end
22 C.new.test
23 C.new.protegidoC
```

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

Especificadores de acceso en Ruby: Ejemplos

Ruby: Falsa seguridad

```ruby
class FalsaSeguridad
  # Un consultor puede ser muy peligroso
  attr_reader :lista

  def initialize
    @lista = [1,2,3,4]
  end

  def info
    puts @lista.size
  end

end

# Fuera de cualquier clase
f = FalsaSeguridad.new
f.info #4

# Modificamos el estado interno
# Simplemente usando un consultor
# Aunque el atributo sea privado
# Cuidado con las  referencias
f.lista.clear

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