

# Herencia en el Ámbito de Clase

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 las diferencias existentes entre Java y Ruby relacionadas con la herencia en el ámbito de clase

# Contenidos

1 Java

2 Ruby

# Java

- No permite la redefinición de métodos de clase al mismo nivel que de instancia
- Aunque pueden existir métodos de clase con el mismo nombre en una jerarquía de clases, no se obtienen los mismos resultados que a nivel de instancia

# Ejemplo

## Java: Ejemplo de herencia en el ámbito de clase

```

1 class Padre {
2     public static final int DECLASE = 1;
3     public static int getDECLASE() { return DECLASE; }
4 }
5
6 class Hija extends Padre {
7     public static final int DECLASE = 2; // Variable shadowing
8 }
9
10 class Nieta extends Hija{
11     public static int getDECLASE() { // No es una redefinición
12         // super.getDECLASE() No permitido
13         return DECLASE;
14     }
15 }
16
17 public static void main(String[] args) {
18     System.out.println (Padre.DECLASE); // 1
19     System.out.println (Hija.DECLASE); // 2
20     System.out.println (Nieta.DECLASE); // 2
21     System.out.println (Padre.getDECLASE()); // 1
22     System.out.println (Hija.getDECLASE()); // 1
23     //porque "redefine" el método de clase
24     System.out.println (Nieta.getDECLASE()); // 2
25 }

```

# Ejemplo

## Java: Ejemplo de herencia en el ámbito de clase

```
1 public static void main(String[] args) {  
2  
3     // El tipo estático de las instancias influye  
4  
5     // Aunque Java lo permite, no se debe invocar a métodos de clase así  
6     // Lo digo en serio  
7  
8     Padre p=new Padre();  
9     System.out.println (p.getDECLASE()); // 1  
10  
11     p = new Nieta();  
12     System.out.println (p.getDECLASE()); // 1  
13  
14     Nieta n = new Nieta();  
15     System.out.println (n.getDECLASE()); // 2  
16 }
```

# Ruby

- Las clases son *first class citizens* y en el ámbito de clase todo funciona como es de esperar



# Ejemplo

## Ruby: Ejemplo de herencia en el ámbito de clase

```

1 class Padre
2   @atributo_clase1 = 1
3   @atributo_clase2 = 2
4   @@atributo_clase3 = 5
5
6   def self.salida
7     puts @atributo_clase1+1
8     puts @atributo_clase2+1 unless @atributo_clase2.nil?
9     puts @@atributo_clase3+1
10  end
11
12  def self.salida2
13    salida
14  end
15 end
16
17 Padre.salida # 2 3 6
18
19 class Hija < Padre
20   @atributo_clase1 = 3
21   @@atributo_clase3 = 7
22
23   def self.salida2
24     super # Las clases son "first class citizens"
25     puts @atributo_clase1+1
26   end
27 end
28
29 Padre.salida # 2 3 8
30 Hija.salida # 4 8
31 Padre.salida2 # 2 3 8
32 Hija.salida2 # 4 8 4

```

# Herencia en el Ámbito de Clase

Dpto. Lenguajes y Sistemas Informáticos  
Universidad de Granada

Programación y Diseño Orientado a Objetos