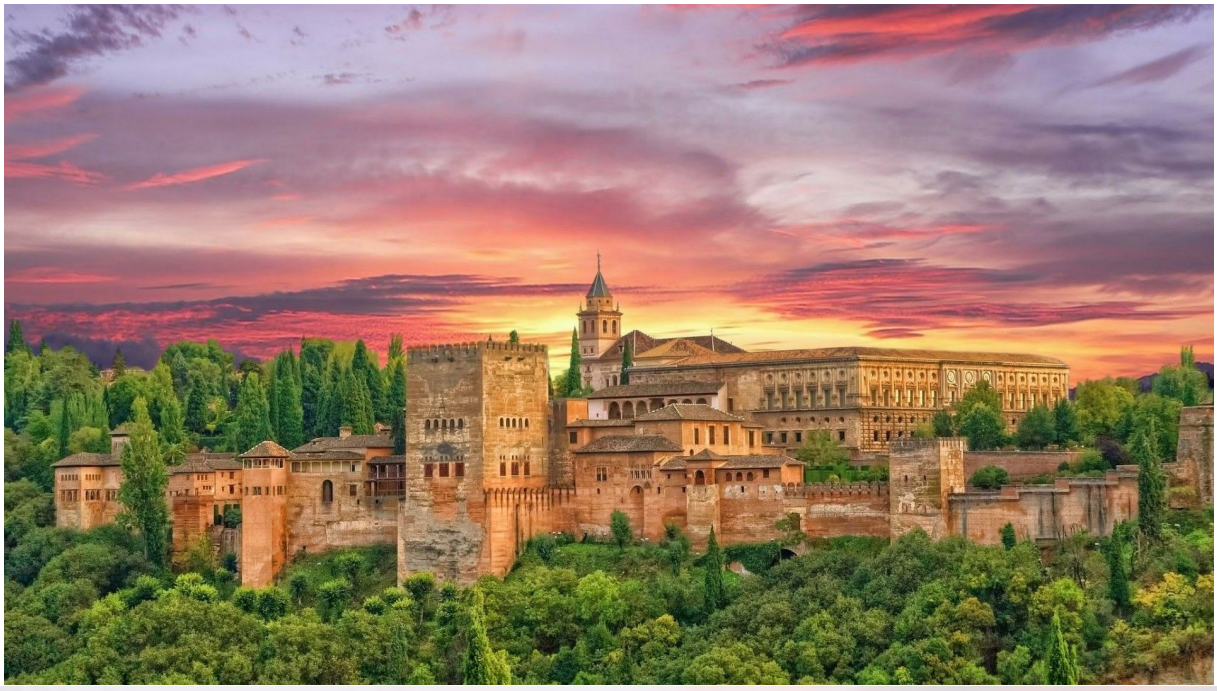
ETSIIT
Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

Ingeniería Informática + ADE

Universidad de Granada (UGR)

Autor: Ismael Sallami Moreno

Asignatura: Apuntes Herencia (PDOO)



Índice

1 Ejemplo 1

```ruby
# Definición de la clase Persona
class Persona
  # Método initialize para inicializar la variable de instancia @nombre
  def initialize(n)
    @nombre = n # Se asigna el valor pasado al nombre de la persona
  end

  # Método andar que simula la acción de caminar
  def andar
    "Ando como una persona" # Devuelve una cadena describiendo la acción
  end

  # Método hablar que simula la acción de hablar
  def hablar
    "Hablo como una persona" # Devuelve una cadena describiendo cómo habla
  end
end

# Definición de la clase Profesor que hereda de Persona
class Profesor < Persona
  # Sobrescribe el método hablar
  def hablar
    tmp = "Estimados alumnos:\n" # Cadena inicial específica para el profesor
    tmp += "Me llamo #{@nombre}\n" # Usa @nombre de la clase base Persona
    tmp += super # Llama al método hablar de la clase base
    tmp # Retorna la cadena completa
  end
end

# Crear una instancia de Profesor y llamar al método hablar
puts Profesor.new("Jaime").hablar
# Si Profesor no tiene un método initialize, usa el de Persona.
# En este caso, @nombre es inicializado por el método initialize heredado.
```

- **¿En qué momento ha tomado valor `@nombre`?** La variable de instancia `@nombre` toma su valor al llamar al método `initialize` de la clase base `Persona` cuando se crea la instancia `Profesor.new("Jaime")`.
- **Si `Profesor` no tiene `initialize`, ¿qué va a ocurrir?** Al no definir un método `initialize` propio, `Profesor` hereda el método `initialize` de `Persona`. Por lo tanto, `@nombre` será inicializado correctamente con el valor pasado al crear la instancia, en este caso, `"Jaime"`.

2 Ejemplo 2

```ruby
# Definición de la clase A
class A
  def initialize(a) # Método initialize con un argumento
    puts "Creando A" # Imprime un mensaje indicando que se crea A
    @a = a # Asigna el valor pasado a la variable de instancia @a
  end
end

# Definición de la clase B que hereda de A
class B < A
end

# Crear una instancia de A con un argumento válido
A.new(77) # Correcto: Se ejecuta initialize de A

# Crear una instancia de B sin argumentos
B.new # Incorrecto: No se pasa el argumento esperado por initialize de A

# Crear una instancia de B con un argumento
B.new(88) # Correcto: Se ejecuta initialize heredado de A con el
   argumento
```

- Una de las 3 últimas líneas es errónea. ¿Cuál? ¿Por qué? La línea `B.new` es errónea porque la clase `B` hereda el método `initialize` de `A`, que requiere un argumento. Sin ese argumento, se genera un error.

3 Ejemplo 3

```ruby
# Definición de la clase C
class C
  def initialize(c) # Método initialize con un argumento
    puts "Creando C" # Imprime un mensaje indicando que se crea C
    @c = c # Asigna el valor pasado a la variable de instancia @c
  end
end

# Definición de la clase D que hereda de C
class D < C
  def initialize # Redefine el método initialize sin argumentos
    puts "Creando D" # Imprime un mensaje indicando que se crea D
    @d = 88 # Asigna 88 a la variable de instancia @d
  end
end

# Crear una instancia de C con un argumento válido
C.new(99) # Correcto: Se ejecuta initialize de C con el argumento

# Crear una instancia de D usando su initialize redefinido
d = D.new # Correcto: Se ejecuta initialize de D

# Mostrar el estado del objeto d
puts d.inspect # Muestra las variables de instancia de d
```

- ¿Qué ocurre en la línea 16? Se crea una instancia de `D` y se ejecuta el método `initialize` redefinido en `D`. No se llama al método `initialize` de `C`.
- ¿Cuál es el resultado de la línea 17? Muestra las variables de instancia del objeto `d`: $"@d => 88$. La variable `@c` no existe porque no se llamó a `initialize` de `C`.

4 Ejemplo 4

```ruby
# Definición de la clase E
class E
  def initialize(e) # Método initialize con un argumento
    puts "Creando E" # Imprime un mensaje indicando que se crea E
    @e = e # Asigna el valor pasado a la variable de instancia @e
  end
end

# Definición de la clase F que hereda de E
class F < E
  def initialize # Redefine initialize
    puts "Creando F" # Imprime un mensaje indicando que se crea F
    @f = 88 # Asigna 88 a la variable de instancia @f
    super(99) # Llama al initialize de E explícitamente con un argumento
  end
end

# Crear una instancia de E con un argumento válido
E.new(99) # Correcto: Se ejecuta initialize de E

# Crear una instancia de F usando su initialize redefinido
f = F.new # Correcto: Se ejecuta initialize de F y también el de E

# Mostrar el estado del objeto f
puts f.inspect # Muestra las variables de instancia de f
```

- ¿Qué ocurre en la línea 18? Se crea una instancia de `F`, ejecutando primero el `initialize` de `F`, luego se llama explícitamente a `initialize` de `E` con `super(99)`.
- ¿Cuál es el resultado de la línea 19? Muestra las variables de instancia del objeto `f`: $"@f" => 88, "@e" => 99$. Ambas inicializaciones se completaron correctamente.

```ruby
# Definición de la clase G
class G
  def initialize # Método initialize sin argumentos
    puts "Creando G" # Imprime un mensaje indicando que se crea G
    @g = 66 # Asigna 66 a la variable de instancia @g
  end
end

# Definición de la clase H que hereda de G
class H < G
  def initialize # Redefine initialize sin llamar a super
```

```ruby
12          puts "Creando H" # Imprime un mensaje indicando que se crea H
13          @h = 88 # Asigna 88 a la variable de instancia @h
14        end
15      end
16
17      # Crear una instancia de G
18      G.new # Correcto: Se ejecuta initialize de G
19
20      # Crear una instancia de H usando su initialize redefinido
21      h = H.new # Correcto: Se ejecuta initialize de H
22
23      # Mostrar el estado del objeto h
24      puts h.inspect # Muestra las variables de instancia de h
```

- ¿Qué ocurre en la línea 16? Se crea una instancia de `H`, ejecutando solo el método `initialize` redefinido en `H`. No se llama a `initialize` de `G`.
- ¿Cuál es el resultado de la línea 17? Muestra las variables de instancia del objeto `h`: $"@h" => 88$. La variable `@g` no existe porque no se llamó a `initialize` de `G`.