# Reflexión

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

- Conocer el qué consiste la reflexión

# Contenidos

# Reflexión

- Capacidad de un programa para manipularse a sí mismo y comprender sus propias estructuras en tiempo de ejecución
- Mecanismos
  - Introspección

    Habilidad del programa para observar y razonar sobre su mismo estado (objetos y clases) en tiempo de ejecución
  - Modificación

    Habilidad del programa para cambiar su estado (objetos y clases) durante la ejecución
    - ★ Normalmente solo soportado por lenguajes interpretados

# Reflexión en Java

- Debido a la estructura de metaclases desarrollada por Java, el nivel de reflexión que se permite es de introspección
- Toda la funcionalidad para ello está definida en la clase Class de Java

## Java: Ejemplos

```
1 //Ejemplos
2 MiClase obj = new MiClase();
3 Class clase = obj.getClass() //método definido Object
4 Field[] varInstancia = clase.getFields();
5 Constructor[] construct = clase.getConstructors();
6 Method[] metodosInstancia = clase.getMethods();
7 String nombreClase = clase.getSimpleName();
```

# Reflexión en Ruby

- Debido a la estructura de metaclases desarrollada por Ruby, el nivel de reflexión que se permite es de introspección y de modificación.
- En ejecución se puede:
  - Consultar y modificar una clase
  - Consultar y modificar la estructura y funcionalidad de un objeto haciéndolo distinto de los demás de la misma clase

# Ejemplo en Ruby

## **Ruby:** Modificando la clase. Afecta a todas las instancias

```ruby
class Libro
  def initialize(titulo)
    @titulo = titulo
  end
end

libro1 = Libro.new("El señor de los anillos")

# Se modifica la clase y afecta a todas las instancias
Libro.class_eval do def publicacion(añopublicacion)
                      @añoPublicacion = añopublicacion
                    end
end

libro1.publicacion(1997) # Se invoca el nuevo método
puts libro1.inspect      # Ahora tiene un atributo adicional
```

# Ejemplo en Ruby

## **Ruby:** Modificando una única instancia

```ruby
# Se modifica solo una instancia
libro1.instance_eval do def autor(autor)
                          @autor = autor
                        end
end

libro1.autor("J. R. R. Tolkien")
puts libro1.inspect

libro2 = Libro.new("Cien años de soledad")
libro2.autor("G. García Márquez") # undefined method `autor'
```

# Ejemplo en Ruby

## **Ruby:** Ejemplos de introspección

```ruby
puts  Libro.instance_methods(false) # publicacion
# El parámetro indica si queremos solo los métodos de esa clase (false)
# o también los heredados (true)

puts libro1.instance_variables
                # @autor
                # @titulo
                # @añoPublicacion

puts libro2.instance_variables          # @titulo
puts libro1.instance_of?(Libro)         # true
```

# Reflexión

Dpto. Lenguajes y Sistemas Informáticos
Universidad de Granada

Programación y Diseño Orientado a Objetos