

Reflexión

Prof. Francisco Velasco Anguita

Dpto. Lenguajes y Sistemas Informáticos
Universidad de Granada

Programación y Diseño Orientado a Objetos

Doble Grado en Ingeniería Informática
y Administración y Dirección de Empresas
(Curso 2024-2025)

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

- 1 Reflexión
- 2 Reflexión en Java
- 3 Reflexión en Ruby

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 metaclasses 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 en 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 metaclasses 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

```
1 class Libro
2   def initialize(titulo)
3     @titulo = titulo
4   end
5 end
6
7 libro1 = Libro.new("El señor de los anillos")
8
9 # Se modifica la clase y afecta a todas las instancias
10 Libro.class_eval do
11   def publicacion(año_publicacion)
12     @año_publicacion = año_publicacion
13   end
14 end
15
16 libro1.publicacion(1997) # Se invoca el nuevo método
17 puts libro1.inspect      # Ahora tiene un atributo adicional
```


Ejemplo en Ruby

Ruby: Modificando una única instancia

```
1 # Se modifica solo una instancia
2 libro1.instance_eval do
3   def autor(autor)
4     @autor = autor
5   end
6 end
7
8 libro1.autor("J. R. R. Tolkien")
9 puts libro1.inspect
10
11 libro2 = Libro.new("Cien años de soledad")
12 libro2.autor("G. García Márquez") # ERROR: undefined method 'autor'
```

Ejemplo en Ruby

Ruby: Ejemplos de introspección

```
1 puts Libro.instance_methods(false) # publicacion
2 # El parámetro indica si queremos solo los métodos de esa clase (false)
3 # o también los heredados (true)
4
5 puts libro1.instance_variables
6     # @autor
7     # @titulo
8     # @añoPublicacion
9
10 puts libro2.instance_variables      # @titulo
11 puts libro1.instance_of?(Libro)    # true
```

Reflexión

Prof. Francisco Velasco Anguita

Dpto. Lenguajes y Sistemas Informáticos
Universidad de Granada

Programación y Diseño Orientado a Objetos

Doble Grado en Ingeniería Informática
y Administración y Dirección de Empresas
(Curso 2024-2025)