

# Programación orientada a objetos en Java

Ricardo Pérez López

IES Doñana, curso 2020/2021



Generado el 12 de febrero de 2021 a las 14:09:00

1. Uso básico de objetos
2. Clases y objetos básicos en Java

# 1. Uso básico de objetos

1.1 Instanciación

1.2 Referencias

1.3 Comparación de objetos

1.4 Destrucción de objetos

## 1.1. Instanciación

1.1.1 `new`

1.1.2 `getClass()`

1.1.3 `instanceof`

## new

- ▶ La operación **new** permite instanciar un objeto a partir de una clase.

## getClass()

- ▶ El método `getClass()` devuelve la clase de la que es instancia el objeto sobre el que se ejecuta.
- ▶ Lo que devuelve es una instancia de la clase `java.class.Class`.
- ▶ Para obtener una cadena con el nombre de la clase, se puede usar el método `getSimpleName()` definido en la clase `Class`:

```
jshell> String s = "Hola";  
s ==> "Hola"  
  
jshell> s.getClass()  
$2 ==> class java.lang.String  
  
jshell> s.getClass().getSimpleName()  
$3 ==> "String"
```

## instanceof

- ▶ El operador **instanceof** permite comprobar si un objeto es instancia de una determinada clase.
- ▶ Por ejemplo:

```
jshell> "Hola" instanceof String  
$1 ==> true
```

- ▶ Sólo se puede aplicar a referencias, no a valores primitivos:

```
jshell> 4 instanceof String  
| Error:  
| unexpected type  
|   required: reference  
|   found:    int  
| 4 instanceof String  
| ^
```

## 1.2. Referencias

### 1.2.1 `null`



## Referencias

- ▶ Los objetos son accesibles a través de **referencias**.
- ▶ Las referencias se pueden almacenar en variables de **tipo referencia**.
- ▶ Por ejemplo, `String` es una clase, y por tanto es un tipo referencia. Al hacer la siguiente declaración:

```
String s;
```

estamos declarando `s` como una variable que puede contener una referencia a un valor de tipo `String`.

# null

- ▶ El tipo `null` sólo tiene un valor: la referencia nula, representada por el literal `null`.
- ▶ El tipo `null` es compatible con cualquier tipo referencia.
- ▶ Por tanto, una variable de tipo referencia siempre puede contener la referencia nula.
- ▶ En la declaración anterior:

```
String s;
```

la variable `s` puede contener una referencia a un objeto de la clase `String`, o bien puede contener la referencia nula `null`.

- ▶ La referencia nula sirve para indicar que la variable no apunta a ningún objeto.

## 1.3. Comparación de objetos

1.3.1 `equals`

1.3.2 `compareTo`

1.3.3 `hashCode`

## Comparación de objetos

- ▶ El operador `==` aplicado a dos objetos (valores de tipo referencia) devuelve `true` si ambos son **el mismo objeto**.
- ▶ Es decir: el operador `==` compara la **identidad** de los objetos para preguntarse si son **idénticos**.
- ▶ Para usar un mecanismo más sofisticado, hay que usar el método `equals`.

## equals

- ▶ El método `equals` compara dos objetos para preguntar si son iguales.
- ▶ Debería usarse siempre en sustitución del operador `==`, que sólo comprueba si son idénticos.

```
jshell> String s = new String("hola");  
s ==> "hola"  
  
jshell> String w = new String("hola");  
w ==> "hola"  
  
jshell> s == w  
$3 ==> false  
  
jshell> s.equals(w)  
$4 ==> true
```

## compareTo

- Un método parecido es `compareTo`, que compara dos objetos de forma que la expresión `a.compareTo(b)` devuelve un entero:
  - `-1` si `a < b`.
  - `0` si `a == b`.
  - `1` si `a > b`.

## 1.4. Destrucción de objetos

### 1.4.1 Recolección de basura

## 2. Clases y objetos básicos en Java

2.1 Clases envolventes (*wrapper*)

2.2 Cadenas

2.3 Arrays



## 2.1. Clases envolventes (*wrapper*)

### 2.1.1 *Boxing y Unboxing*

## 2.2. Cadenas

2.2.1 Inmutables

2.2.2 Mutables

2.2.3 Conversión a `String`

2.2.4 Concatenación de cadenas

2.2.5 Comparación de cadenas

2.2.6 Diferencias entre literales cadena y objetos `String`

## 2.3. Arrays

2.3.1 De tipos primitivos

2.3.2 `.length`

2.3.3 De objetos

2.3.4 Subtipado entre *arrays*

2.3.5 `java.util.Arrays`

2.3.6 Copia y redimensionado de arrays

2.3.7 Comparación de *arrays*

2.3.8 Arrays multidimensionales

## Arrays.deepEquals()

Gosling, James, Bill Joy, Guy L. Steele, Gilad Bracha, and Alex Buckley. 2014. *The Java® Language Specification*. Java SE 8 edition. Upper Saddle River, NJ: Addison-Wesley.