CONSTRUCTORES

INTRODUCCIÓN A LA ORIENTACIÓN A OBJETOS

OpenWebinars

CONSTRUCCIÓN DE OBJETOS

- En Java, todo objeto debe ser construido antes de ser usado.
- Construcción → creación a partir de la plantilla (clase).
- Construcción ≡ instanciación.
- Un mismo objeto se puede construir de diferentes formas.

CONSTRUCTORES DE UNA CLASE

- Una clase puede definir **cero**, **uno** o **más** de un **constructor**.
- Método especial que *se llama como la propia clase*.
- Por ahora, siempre **public**.

```
public Coche(String marca, String modelo, int anio) {
    this.marca = marca;
    this.modelo = modelo;
    this.anio = anio;
}
```

CONSTRUCTOR POR DEFECTO

- **Si una clase no define ningún constructor, Java genera uno por defecto.**
- Constructor sin argumentos.
  - No recibe ningún valor entre paréntesis.
  - Instancia un objeto “vacío”.

```
Coche coche = new Coche();
```

CONSTRUCTOR POR DEFECTO

- ¡OJO! Si definimos un constructor por nuestra cuenta, Java ya no genera el constructor por defecto.
- Si necesitamos un constructor sin argumentos, lo tenemos que definir nosotros.

```
public Coche() {}

public Coche(String marca, String modelo, int anio) {
    this.marca = marca;
    this.modelo = modelo;
    this.anio = anio;
}
```

CONSTRUCTORES QUE USAN CONSTRUCTORES

- Un constructor puede usar a otro constructor como parte de su proceso de construcción del objeto.
- Así reutilizamos código.
- Se invoca con **this(...)**

```
public Coche(String marca, String modelo, int anio) {
    this.marca = marca;
    this.modelo = modelo;
    this.anio = anio;
}


public Coche(String marca, String modelo) {
    this(marca, modelo, 2020);
}
```

LIMITACIONES EN LOS CONSTRUCTORES

- Podemos tener tantos constructores como queramos.
- Restricción: todos deben tener alguna diferencia
  - Diferente número de argumentos.
  - A mismo número de argumentos, diferente tipo.

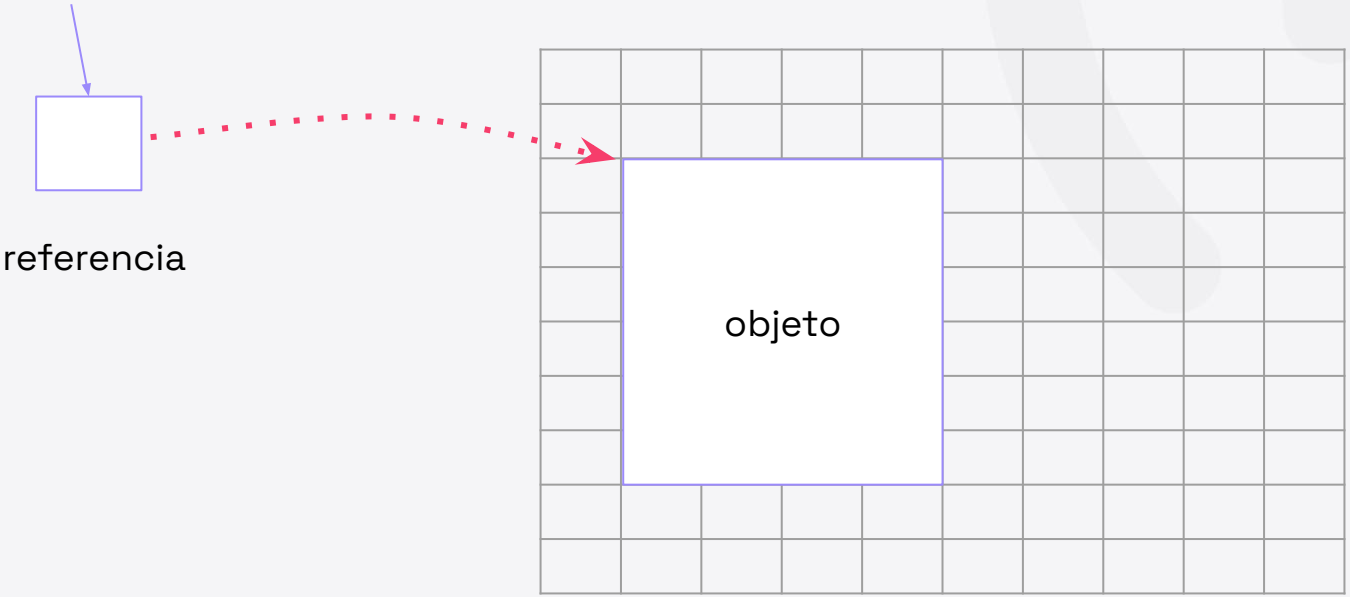
```
public Coche(String marca, String modelo) {
    this(marca, modelo, 2020);
}

public Coche(String marca, int anio) {
    this.marca = marca;
    this.anio = anio;
}
```

OBJETOS Y MEMORIA

- Java gestiona automáticamente la memoria.
  - Gran diferencia con C, C++ donde el programador se encarga de reservar, ampliar, liberar, ...
- Los objetos se guardan en una zona llamada *heap*.
- Cuando creamos un objeto...
  - La referencia no es realmente el objeto.
  - Es un *acceso directo* al objeto en memoria.

OBJETOS Y MEMORIA

Coche coche = new Coche();



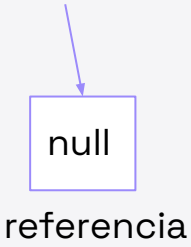
NULL

- Al igual que con los tipos primitivos, **se puede declarar un objeto sin instanciarlo**.
- Entonces, se dice que su referencia es nula o apunta a **null**.
- Se puede hacer indicar explícitamente.

```
Coche coche;
Coche coche = null;
```

NULL

Coche coche;



referencia

Todavía no hay nada en el *heap*.

INFERENCIA DE TIPO DE DATO

- Se puede usar ***var*** para declarar una referencia a un objeto.

```
var coche2 = new Coche("Seat", "Ibiza", 2021);
coche2.arrancar();
```

GENERACIÓN DE CONSTRUCTORES

- Eclipse tiene un asistente para generar los constructores de una clase.