

## 5.C. Métodos.

### 1. Métodos.

#### 1.6. Sobrecarga de métodos.

En principio podrías pensar que un método puede aparecer una sola vez en la declaración de una clase (no se debería repetir el mismo nombre para varios métodos). Pero no tiene porqué siempre suceder así. Es posible tener varias versiones de un mismo método (varios métodos con el mismo nombre) gracias a la **sobrecarga de métodos**.



El lenguaje **Java** soporta la característica conocida como **sobrecarga de métodos**. Ésta permite declarar en una misma clase varias versiones del mismo método con el mismo nombre. La forma que tendrá el compilador de distinguir entre varios métodos que tengan el mismo nombre será mediante la lista de parámetros del método: si el método tiene una lista de parámetros diferente, será considerado como un método diferente (aunque tenga el mismo nombre) y el **analizador léxico no producirá un error** de compilación al encontrar dos nombres de método iguales en la misma clase.

Imagínate que estás desarrollando una clase para escribir sobre un lienzo que permite utilizar diferentes tipografías en función del tipo de información que se va a escribir. Es probable que necesitemos un método diferente según se vaya a pintar un número entero (**int**), un número real (**double**) o una cadena de caracteres (**String**). Una primera opción podría ser definir un nombre de método diferente dependiendo de lo que se vaya a escribir en el lienzo. Por ejemplo:

- Método `pintarEntero (int entero)`.
- Método `pintarReal (double real)`.
- Método `pintarCadena (double String)`.
- Método `pintarEnteroCadena (int entero, String cadena)`.

Y así sucesivamente para todos los casos que desees contemplar...

La posibilidad que te ofrece la sobrecarga es utilizar un mismo nombre para todos esos métodos (dado que en el fondo hacen lo mismo: pintar). Pero para poder distinguir unos de otros será necesario que siempre exista alguna diferencia entre ellos en las listas de parámetros (bien en el número de parámetros, bien en el tipo de los parámetros). Volviendo al ejemplo anterior, podríamos utilizar un mismo nombre, por ejemplo `pintar`, para todos los métodos anteriores:

- Método `pintar (int entero)`.
- Método `pintar (double real)`.
- Método `pintar (double String)`.
- Método `pintar (int entero, String cadena)`.

En este caso el compilador no va a generar ningún error pues se cumplen las normas ya que unos métodos son perfectamente distinguibles de otros (a pesar de tener el mismo nombre) gracias a que tienen listas de parámetros diferentes.

Lo que sí habría producido un error de compilación habría sido por ejemplo incluir otro método `pintar (int entero)`, pues es imposible distinguirlo de otro método con el mismo nombre y con la misma lista de parámetros (ya existe un método `pintar` con un único parámetro de tipo **int**).

También debes tener en cuenta que el **tipo devuelto** por el método no es considerado a la hora de identificar un método, así que un tipo devuelto diferente **no es suficiente para distinguir un método de otro**. Es decir, no podrías definir dos métodos exactamente iguales en nombre y lista de parámetros e intentar distinguirlos indicando un tipo devuelto diferente. El compilador producirá un error de duplicidad en el nombre del método y no te lo permitirá.

**Es conveniente no abusar de sobrecarga de métodos** y utilizarla con cierta moderación (cuando realmente puede beneficiar su uso), dado que podría hacer el código menos legible.

#### Autoevaluación

En una clase Java puedes definir tantos métodos con el mismo nombre como desees y sin ningún tipo de restricción pues el lenguaje soporta la sobrecarga de métodos y el compilador sabrá distinguir unos métodos de otros. ¿Verdadero o falso?

- ☐ Verdadero.
- ☐ Falso.