FUNDACIÓN ACCENTURE

accenture

JAVA I

3. LOS OBJETOS EN JAVA

Conoce qué es un objeto en Java y aprende a crearlos con sus diferentes métodos.

TABLA DE CONTENIDOS

JAVA I

3.1. ¿QUÉ ES UN OBJETO?

¿QUÉ ES UN OBJETO EN JAVA?

Los objetos en Java, al igual que los objetos del mundo real, son elementos (nuestro coche, el coche de mi amigo) que se crean a partir de una clase (Coche).

Los objetos en Java constan de:

- **Identidad:** propiedad que permite diferenciarse entre otros objetos (Ej. Nombre coche)
- **Estado:** atributos y sus valores en un momento dado (Ej. Color, longitud, año de fabricación)
- **Comportamiento:** son las operaciones que puede realizar el objeto (Ej. Acelerar, frenar)

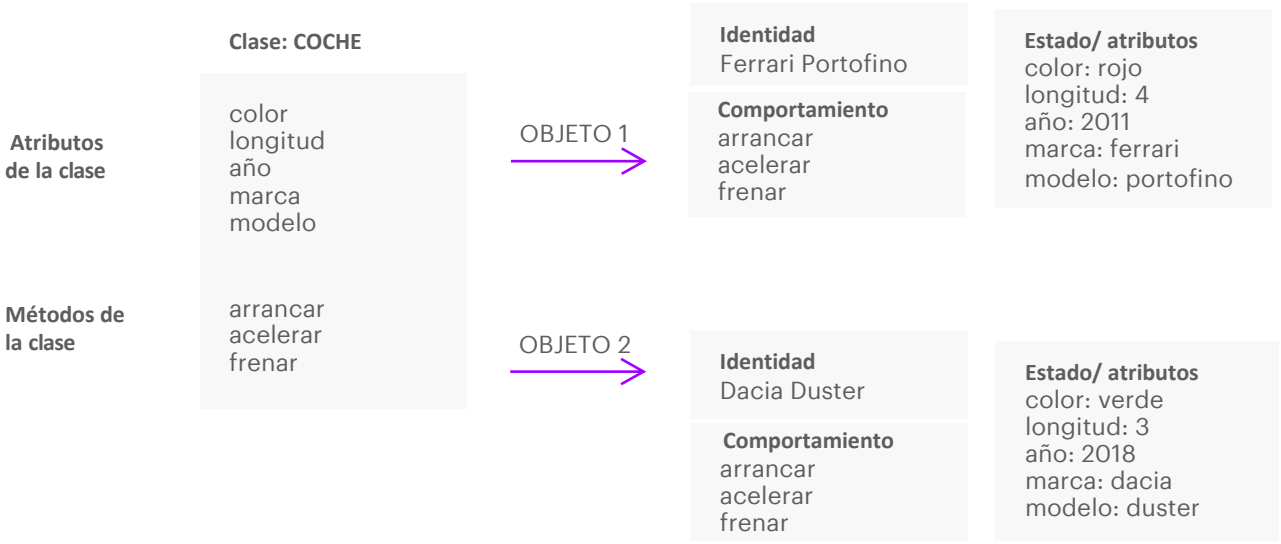
Además, el lenguaje Java permite que:

→ Los objetos se puedan agrupar por **categorías** (coches descapotables, coches de campo, coches minis) según características comunes entre ellos.

→ Los programas se organicen como **colecciones** de objetos que cooperan o interactúan entre sí.

Siguiendo nuestro **ejemplo** de los coches: hemos creado la Clase Coche añadiendo algún atributo más (marca y modelo) y de esta vamos a crear dos objetos diferentes. El primero que identificaremos será un Ferrari Portofino (asignamos la variable c1) y el segundo será un Dacia Duster (asignamos la variable c2). Cada uno de ellos tendrá diferente estado/atributos (marca, modelo, color, año de fabricación…) y comportamiento (arrancar, girar, frenar).

Es decir, que de la clase Coche, podremos crear distintos objetos con distintas características.



CREACIÓN DE UN OBJETO

Ya sabemos qué es un objeto, ahora vamos a crearlo. Cada objeto que se crea de una **clase** es una instancia. Para crear una instancia o un objeto de una clase, debemos usar la palabra clave **new**. En nuestro caso tenemos dos objetos así que crearemos los dos para la misma clase (Coche).

Su sintaxis es: `Nombreclase variableReferencia = new NombreClase(atrib1,atrib2...);`

Creación del objeto 1: `Coche c1 = new Coche ("rojo", 4, 2011, "Ferrari", "Portofino")`

Creación del objeto 2: `Coche c2 = new Coche("verde", 3, 2018, "Dacia", "Duster");`

MODIFICAR O ACCEDER A UN OBJETO

Una vez creado el objeto, podemos acceder a sus **atributos** o **métodos** usando el punto (.)
Por ejemplo, si queremos modificar uno de sus atributos, primero ponemos el **nombre del objeto**, luego el punto **(.)**, un igual **(=)** y finalmente el valor que queremos darle al atributo. Imagina que pintamos el Ferrari en color negro y queremos cambiar el color en el programa...

```
C1.color="negro";
```

PALABRA RESERVADA NULL

La palabra reservada *null* indica que una variable que referencia a un objeto se encuentra "sin objeto", es decir, la variable ha sido declarada pero no apunta a ningún objeto. Esto puede deberse a que no se haya inicializado asignándole un objeto mediante la instrucción *new*, o a que hayamos borrado la referencia que contenía la variable.

Siguiendo nuestro ejemplo, imagina que procedemos a modificar el color del Ferrari pero no hemos creado previamente el objeto. Nos dará null.

c1 será nuestra variable de referencia que almacena la dirección de memoria donde se guarda realmente el objeto y toda su información.

NULL indica que el **objeto no existe**, o que **no está apuntando a ningún sitio en la memoria**.

Atributos: almacenan las características de un objeto (la información que tenemos del objeto)

Clase: define la forma que tendrán los objetos creados a partir de ella. En la clase se incluyen los atributos y métodos que tendrán los objetos.

Métodos: representan el comportamiento de un objeto (lo que hace).

3.2. MÉTODOS DE LOS OBJETOS

MÉTODO EQUALS

Sirve para **comparar dos objetos** de una clase que han sido previamente creados.

Con nuestro **ejemplo** de los coches, vamos a comparar los dos objetos que hemos creado. En primer lugar, tenemos que saber **¿qué se quiere comparar?**

→ Si queremos comparar **las referencias**, es decir, las direcciones de memoria donde se ubican dos objetos, se puede usar directamente el operador **= =** que vimos en temas anteriores. Esto ocurre al comparar objetos, ya que en el caso de comparar variables, este operador compara el valor. Vamos a entenderlo mejor con estos ejemplos:

Ejemplo con variables:

```
int a = 7;
int b = 7;
```

`¿a==b?` **Verdadero**, ya que compara el valor de a y b, es decir, compara si 7 es igual a 7.

Ejemplo con objetos 1:

```
Coche c1 = new Coche("verde", 3, 2018, "Dacia", "Duster");
Coche c2 = new Coche("verde", 3, 2018, "Dacia", "Duster");
```

`¿c1==c2?` **Falso**, ya que, aunque todos los valores son iguales, hemos creado dos instancias diferentes que ocupan diferentes direcciones en la memoria, c1 y c2 apuntan a dos lugares diferentes en la memoria.

Ejemplo con objetos 2:

```
Coche c3 = new Coche("rojo", 4, 2011, "Ferrari", "Portofino");
Coche c4 = c3;
```

`¿c3==c4?` **Verdadero**, ya que hemos creado una instancia y guardado su dirección de memoria en c3 y, posteriormente, hemos asignado a c4 la dirección de memoria a la que apunta c3, así que c3 y c4 apuntan a la misma dirección de memoria.

→ Si queremos comparar **el contenido o atributos** de dos objetos creados (es decir, comparar uno, varios o todos sus atributos) se ha de usar el método equals. Para ello debemos sobrescribir el método `equals()` para definir cómo debe actuar a la hora de comparar los dos objetos.

Para este **ejemplo** vamos a suponer que se ha sobrescrito para comparar los atributos marca y modelo, es decir, consideraremos dos coches iguales cuando sean de la misma marca y modelo, aunque tengan el color u otros atributos diferentes. Vamos a comparar c1 con un nuevo objeto, el c5.

```
Coche c1 = new Coche("verde", 3, 2018, "Dacia", "Duster");
Coche c5 = new Coche("azul", 3, 2019, "Dacia", "Duster");
```

`¿c1.equals(c5)?` **Verdadero**, los coches tienen la misma marca y modelo, por tanto son iguales, aunque el resto de los atributos puedan ser distintos, ya que habíamos definido equals para que solo considerase la marca y el modelo.

¿Sabías que...? Eclipse (la aplicación para programar en Java) permite generar este método automáticamente sólo con un botón. Además, te da la opción de elegir que atributo/s de tu clase quieres comparar.

MÉTODO TOSTRING

El método `toString(),` se usa para **mostrar un texto con la información de un objeto**.

La mayoría de las clases de la API de Java, tienen sobrescrito este método para que muestre la información de acuerdo con el objeto que se maneja. En cambio, si es un **objeto de creación propia** y se invoca al método toString(), mostrará la información de la variable de referencia, es decir, la dirección de memoria (dónde está el objeto) y no la información del objeto (atributos, comportamiento…).

Conclusión: para mostrar la información de nuestros objetos, antes de invocar al método toString, tenemos que **sobrescribirlo** para especificar qué componentes (atributos o métodos) de la clase queremos mostrar y en qué formato.

```
Coche c1 = new Coche("rojo", 4, 2011, "Ferrari", "Portofino");
System.out.printLn(c1.toString());
```

Ten en cuenta que…

Cuando se usan los métodos println() o print() y se pasa un objeto, este llamará automáticamente al método toString() de ese objeto, sin necesidad de invocarlo de manera explícita, por ello podríamos haberlo escrito también de la siguiente manera:

```
Coche c1 = new Coche("rojo", 4, 2011, "Ferrari", "Portofino");
System.out.printLn(c1);
```

3.3. PREGUNTAS FRECUENTES

- **¿Al crear un objeto puedo escribir sus atributos de tipo String (texto) como quiera (mayúscula o minúscula)?**

  Sí, puedes darle el nombre que quieras, pero siempre escribiéndolo entre doble comilla “color“ o “Color”. Y para los de tipo char (códigos de números y letras) entre comilla simple `123ab´ o ´123AB´.

- **¿Dos objetos con los mismos valores en sus atributos pueden ser iguales?**

  Depende:

  - No, porque ocupan distinto espacio en la memoria, aunque tengan iguales atributos.
  - Sí, cuando comparamos sus atributos sobreescribiendo el método equals, los daremos por iguales porque todos sus atributos lo son.

FUNDACIÓN ACCENTURE

accenture