



# Curso de Java Standard



Ing. Octavio Robleto



octavio.robleto@gmail.com



<https://octaviorobleto.com>



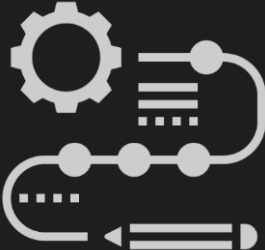
# Introducción

Recordemos que los métodos son un conjunto de instrucciones definidas dentro de una clase, que realizan una determinada tarea y a las que podemos invocar mediante un nombre.

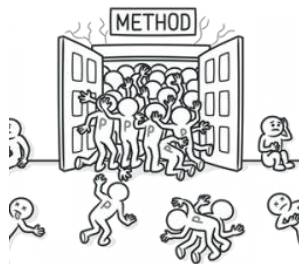
Cuando se llama a un método, la ejecución del programa pasa al método, ejecuta las instrucciones que se encuentren en él y la ejecución continúa a partir del punto donde se produjo el llamado, es decir, sigue su flujo natural.

La idea principal de los métodos es no escribir una misma instrucción varias veces, si la vamos a utilizar mas de una vez.

```
void encender() {  
    encendido = true;  
}  
  
void apagar() {  
    encendido = false;  
}
```



# Parámetros y Argumentos



Para que estos bloques de código sean mucho mas genéricos y los podamos aprovechar para resolver varios temas con el mismo propósito, los métodos nos proporcionan una forma de intercambiar información.

**Parámetros:** Son la declaración de uno o mas tipos de datos (Variables u Objetos)

**Argumentos:** Son los valores que se envían al llamar al método.

Con esto logramos reutilizar de una forma mas eficiente una porción del código.

Parámetro

```
void cambiarEstado(boolean estado){  
    encendido = estado;  
}
```

Argumentos

```
//Encender el Auto  
auto1.cambiarEstado(true);  
//Apagar el Auto  
boolean encendido = false;  
auto1.cambiarEstado(encendido);
```

# Tipos de Métodos

Existen diferentes tipos de métodos:

**Tipo función:** son métodos que pueden realizar operaciones y nos devuelven algo.

Se identifican por que comienzan con un tipo de dato u objeto. La devolución del resultado se expresa con la palabra reservada **return** seguida del dato u objeto a devolver.

**Tipo procedimiento:** son métodos que realizan operaciones sin devolver un valor u objeto concreto. Un método es tipo procedimiento en Java si comienza con la palabra reservada **void**.

***Nota:** Aunque es poco común los métodos de tipo procedimiento admiten la palabra **return**, aunque no tendría ninguna relevancia se comportará de la misma manera en ambos casos: saldrá de la función en ejecución.*

*La única diferencia consistirá en que junto con la instrucción **return** se devolverá (o no) un valor.*

```
// devuelve una cadena de caracteres con las características que posea el objeto
String mostrarDatos() {
    String mensaje = "El Auto es de color " + color + ", marca " + marca
        + ", patente " + patente + " y se encuentra "
        + ((encendido) ? "encendido" : "apagado");

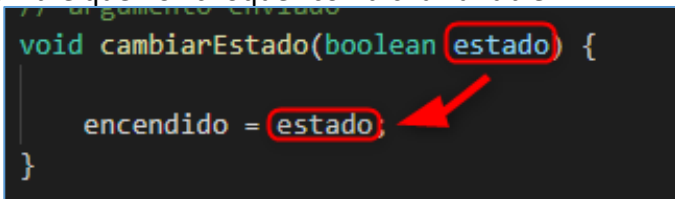
    // la palabra reservada return le indica al metodo que finalizo su ejecucion
    // y que devuelva el objeto mensaje
    return mensaje;
}

//cambia el estado del atributo encendido sin devolver ningun dato
void cambiarEstado(boolean encendido){
    this.encendido = encendido;
}
```

# Sobrecarga de Nombres

Muchas veces se nos presenta que debemos crear un parámetro para asignar el valor a una variable de instancia y nos surge la necesidad de pensar en un nombre que no “choque” con dicha variable.

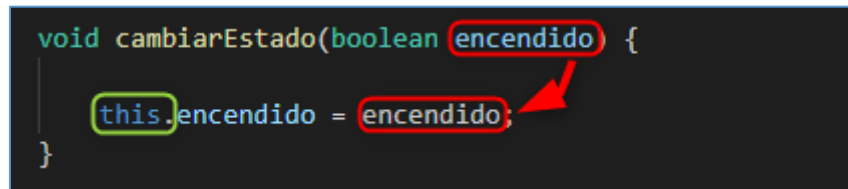
```
// argumento enviado  
void cambiarEstado(boolean estado) {  
    encendido = estado;  
}
```



Para este problema se nos proporciona la capacidad de sobrecargar o utilizar un nombre idéntico en el parámetro para asignárselo a una variable de instancia.

Cuando esto ocurre lo que debemos hacer es anteponer la palabra reservada **this**, antes de la variable de instancia y esto le indicará a Java que estás haciendo referencia al atributo de la clase y no al parámetro.

```
void cambiarEstado(boolean encendido) {  
    this.encendido = encendido;  
}
```



# Sobrecarga de Métodos (Overloading)

Algo parecido se nos proporciona al definir en una clase más de un método con el mismo nombre, con la condición de que no puede haber dos de ellos con el mismo número o tipo de parámetros.

```
// cambia el estado del atributo encendido sin devolver ningun dato por el
// argumento enviado
void cambiarEstado(boolean encendido) {
    this.encendido = encendido;
}

// cambia el estado del atributo encendido por el valor booleano contrario
void cambiarEstado() {
    encendido = !encendido;
}
```

Lo que hay que tomar en cuenta es el nombre del método y no lo que retorna.

# Constructores

Son un tipo de método especial que nos ayuda a construir un objeto.

La diferencia principal entre este método y los vistos anteriormente es que llevan el mismo nombre de la clase, además, no devuelven ningún tipo de dato (sin **return**) ni llevan la palabra reservada **void**.

Se puede decir que los constructores son de tipo procedimiento ya que solo ejecuta las instrucciones que se encuentre dentro del bloque.

```
// constructor simple por defecto
Auto() {

}

// constructor con parametros
Auto(String color, String marca, String patente, boolean encendido) {
    this.color = color;
    this.marca = marca;
    this.patente = patente;
    this.encendido = encendido;
}
```

# Constructores

Cuando la clase no tiene constructores o simplemente no los declaramos JAVA asume por defecto el constructor simple, por lo que podemos instanciar al objeto sin ningún problema.

```
// creamos o instanciamos
Auto auto1 = new Auto();

// le damos valores a los atributos del auto 1
auto1.color = "Rojo";
auto1.marca = "Ferrari";
auto1.patente = "ABC-188";

// Encendemos el Auto en true y apagamos el Auto en false
auto1.encendido(true);

// creamos o instanciamos y le damos valores a los atributos del auto 2
Auto auto2 = new Auto("Plateado", "Audi", "ZBG-999", true);
```

Cuando se definen constructores con parámetros en una clase, Java deja de ofrecer el constructor simple o por defecto de manera implícita por lo que si quieres instanciar un objeto con dicho constructor deberás especificarlo en la clase.