

# Bootcamp Java Developer

Fase 1 - Java Analyst  
Módulo 7

# Métodos

# 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 más de una vez.**

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

# Parámetros y argumentos

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



## Parámetros

Son la declaración de uno o más tipos de datos (Variables u Objetos).

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

Parámetros

## Argumentos

Son los valores que se envían al llamar al método.

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

Argumentos


# Tipos de métodos

Existen diferentes tipos de métodos:

## Tipo función

Son métodos que pueden realizar **operaciones y devuelven algo**.


Se identifican por comenzar 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**.



## Ejemplo

```
// 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;
}
```

### Notas

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.

## 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 esa variable.

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

¡Te mostramos la solución  
en la siguiente pantalla!





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

Cuando esto ocurre, se debe anteponer la palabra reservada **this**. **antes de la variable de instancia y esto le indicará a Java que se está haciendo referencia al atributo de la clase y no al parámetro.**

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



# Sobrecarga de métodos

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;
}
```

## Nota

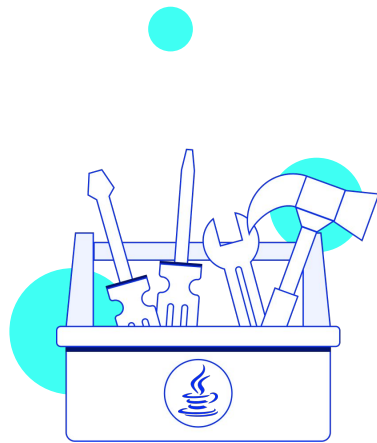
Lo que hay que tomar en cuenta es el nombre del método y no lo que retorna.

# Constructor

Es un tipo de método especial que 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 encuentren dentro del bloque.

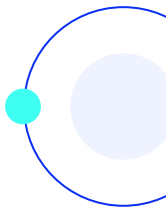


## Ejemplo

```
// 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;
}
```



Cuando la clase no tiene constructores o simplemente no fueron declarados, 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.



**¡Sigamos  
trabajando!**