



Java Métodos



Octavio Robleto



octavio.robleto@gmail.com

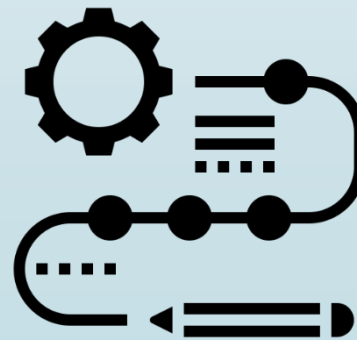


<https://octaviorobleto.com>



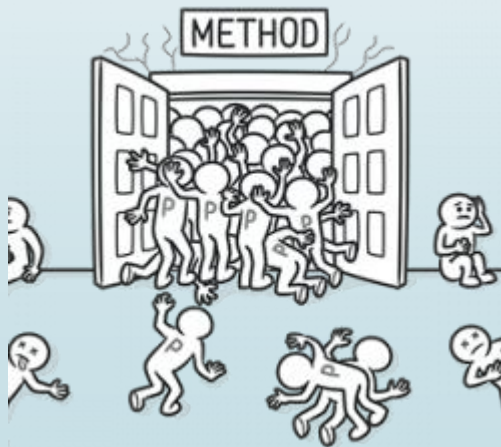
Métodos

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



Métodos - Parámetros y Argumentos

- Los parámetros o argumentos son una forma de intercambiar información con el método.
- Los parámetros son la declaración de uno o mas objetos o variables.
- Los argumentos son valores que se envían a los métodos.



Métodos - Parámetros y Argumentos

```
// metodo 1  
void encender() {  
    encendido = true;  
}  
  
// métodos 2  
void apagar() {  
    encendido = false;  
}
```



Métodos con Parámetros

Método con
parámetros

```
void encendido(boolean encender) {  
    encendido = encender;  
}
```



Métodos - Parámetros y Argumentos

```
auto1.encendido(true);  
boolean encendidoAuto2 = false;  
auto2.encendido(encendidoAuto2);
```

Llamado al método
con argumentos



Constructores

- Los constructores son métodos (como el **main**) especiales invocados al instanciar una clase

```
class Auto {  
  
    // atributos o características  
    String color;  
    String marca;  
    String patente;  
    boolean encendido;  
  
    // constructor  
    Auto() {  
        // TODO Auto-generated constructor stub  
    }  
}
```

Constructor simple o por defecto.

- 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.



Sobrecarga de Nombres

- Esto se da cuando tenemos una variable local de un método o constructor, o un parámetro formal de un método o constructor, con un nombre idéntico al de una Variable de Instancia o Clase.
- Cuando ocurre esto debemos recurrir a llamar a la Variable de Instancia o Clase anteponiéndoles la palabra reservada **this**.

```
class Auto {  
  
    // atributos o características  
    String color;  
    String marca;  
    String patente;  
    boolean encendido;  
  
    void encendido(boolean encendido) {  
        this.encendido = encendido;  
    }  
  
}
```



Tipos de Métodos

- **Tipo función:** son métodos que pueden realizar ciertas operaciones y nos devuelven algo. Se identifican por que comienzan con un tipo de dato u objeto después de su modificador de acceso. 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 si comienza con la palabra reservada **void** después de su modificador de acceso.

```
public static double calculateArea( double radius ) {  
    return Math.PI * radius * radius;  
}  
  
public static void main(String[] args) {  
    double diameter = 10.0;  
    double area = calculateArea( diameter / 2 );  
    System.out.println(area);  
}
```



Sobrecarga de Métodos y Nombres

- Permite 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.

```
// atributos o características
public String color;
public String marca;
public String patente;
public boolean encendido;
```

```
/**
 * constructor simple por defecto
 */
public Auto() {
}
```

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

Constructor simple o por defecto.

Constructor con parámetros.



Ejemplo de Instancia de los objetos con diferentes constructores

Usando Constructor simple

```
// creamos o instanciamos los objetos
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);

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

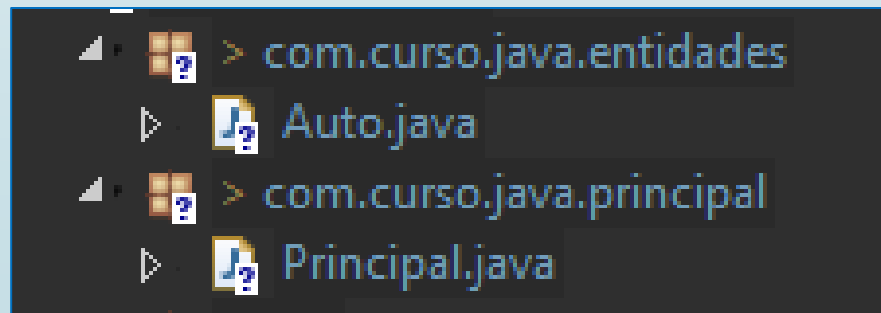
//comportamiento de mostrar datos
auto1.mostrarDatos();
auto2.mostrarDatos();
```

Usando Constructor con parámetros.



Paquetes

- Los paquetes son una forma de organizar grupos de clases. Un paquete contiene un conjunto de clases relacionadas bien por finalidad, por ámbito o por herencia.
- Los paquetes resuelven el problema del conflicto entre los nombres de las clases. Al crecer el número de clases crece la probabilidad de designar con el mismo nombre a dos clases diferentes.
- Las clases tienen ciertos privilegios de acceso a los miembros de datos y a las funciones miembro de otras clases dentro de un mismo paquete.



Modificadores de Acceso

- Los modificadores de acceso permiten dar un nivel de seguridad mayor a nuestras aplicaciones restringiendo el acceso a diferentes atributos, métodos, constructores asegurándonos que el usuario deba seguir una "ruta" especificada por nosotros para acceder a la información.
- Es muy posible que nuestras aplicaciones vayan a ser usadas por otros programadores o usuarios con cierto nivel de experiencia; haciendo uso de los modificadores de acceso podremos asegurarnos de que un valor no será modificado incorrectamente por parte de otro programador o usuario



Modificadores de Acceso Por Defecto

- Java nos da la opción de no usar un modificador de acceso y al no hacerlo, el elemento tendrá un acceso conocido como default o acceso por defecto que permite que tanto la propia clase como las clases del mismo paquete accedan a dichos componentes (de aquí la importancia de declarar siempre un paquete a nuestras clases)

```
class Auto {  
    // atributos o características  
    String color;  
    String marca;  
    String patente;  
    boolean encendido;  
  
    // constructor  
    Auto() {  
  
    }  
  
    // metodo 1  
    void encender() {  
        encendido = true;  
    }  
  
    // métodos 2  
    void apagar() {  
        encendido = false;  
    }  
}
```



Modificadores de Acceso

Privado

- Es el más restrictivo de todos, básicamente cualquier elemento de una clase que sea privado puede ser accedido únicamente por la misma clase por nada más

```
class Auto {  
    // atributos o características  
    private String color;  
    private String marca;  
    private String patente;  
    private boolean encendido;  
  
    // constructor  
    private Auto() {  
  
    }  
  
    // metodo 1  
    private void encender() {  
        encendido = true;  
    }  
  
    // métodos 2  
    private void apagar() {  
        encendido = false;  
    }  
}
```



Modificadores de Acceso

Protected

- Nos permite acceso a los componentes con dicho modificador desde la misma clase, clases del mismo paquete y clases que hereden de ella (incluso en diferentes paquetes)

```
class Auto {  
    // atributos o características  
    protected String color;  
    protected String marca;  
    protected String patente;  
    protected boolean encendido;  
  
    // constructor  
    protected Auto() {  
  
    }  
  
    // metodo 1  
    protected void encender() {  
        encendido = true;  
    }  
  
    // métodos 2  
    protected void apagar() {  
        encendido = false;  
    }  
}
```



Modificadores de Acceso

Public

- Es el más permisivo de todos, básicamente public es lo contrario a private en todos los aspectos (lógicamente), esto quiere decir que si un componente de una clase es public, tendremos acceso a él desde cualquier clase o instancia sin importar el paquete o procedencia de ésta

```
public class Auto {  
    // atributos o características  
    public String color;  
    public String marca;  
    public String patente;  
    public boolean encendido;  
  
    // constructor  
    public Auto() {  
  
    }  
  
    // metodo 1  
    public void encender() {  
        encendido = true;  
    }  
  
    // métodos 2  
    public void apagar() {  
        encendido = false;  
    }  
}
```



Modificadores de Acceso

Modificador	La misma clase	Mismo paquete	Subclase	Otro paquete
private	✓	x	x	x
default	✓	✓	x	x
protected	✓	✓	✓	x
public	✓	✓	✓	✓

