códigofacilito

Java Profesional

Bloque 4 : Métodos y estructuras básicas

Tatiana Borda
https://tatianaborda.com/
https://www.linkedin.com/in/tatiana-borda/
https://www.youtube.com/@AlienExplorer

códigofacilito

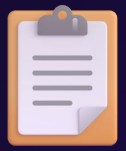

Tabla de contenido

códigofacilito

/01 Qué es un método?

En Java, un método (también conocido como función) es un bloque de código que realiza una tarea específica y que puede ser reutilizado en diferentes partes de un programa. Es como una subrutina que encapsula un conjunto de instrucciones para realizar una acción. Son una herramienta fundamental para organizar, modularizar y reutilizar el código, lo que facilita la creación de programas más legibles, mantenibles y eficientes.

códigofacilito

Cómo funciona un método?

Dijimos que es un bloque de código que se puede ejecutar cuando lo necesitás.
Esta es su estructura básica:

```
modificador tipoDeRetorno nombreDelMétodo(parámetros) {
    // Cuerpo del método
    return valor;  // Solo si el tipo de retorno no es void
}
```

Tiene:
*modificador: public, private, protected, etc. determinan su visibilidad y comportamiento
*tipoDeRetorno: int, String, void (si no retorna nada)
*parámetros: Datos que recibe el método (opcional)

códigofacilito

/02 Parámetros y retorno de valores

Los métodos pueden recibir datos (parámetros) y devolver resultados (valor de retorno) o no, cuando son void
Veamoslo en profundidad!

códigofacilito

Métodos con parámetros

Los parámetros en un método son variables opcionales que reciben datos externos para ser utilizados dentro de la lógica del método. Permiten que un método sea flexible y reutilizable, ya que puede operar con diferentes valores cada vez que se llama.
Actúan como variables locales dentro del método y permiten personalizar el comportamiento del método sin modificar su código interno.

// Definición (parámetros)

```
public static void saludarPersona(String nombre) {
    System.out.println("Hola " + nombre);
}
```

// Llamada (argumentos)

```
saludarPersona("Tatiana");
```

códigofacilito

Métodos con Retorno

Es un bloque de código que devuelve un valor específico después de ejecutarse. A diferencia de los métodos void (que no retornan nada), estos métodos deben incluir la palabra clave return seguida de un valor compatible con su tipo de retorno declarado.

```java
public static int sumar(int a, int b) {
    return a + b;
}
```

```java
int resultado = sumar(5, 3); // resultado = 8
```

códigofacilito

Métodos sin Retorno (void)

Cuando un método se declara con void, significa que no devuelve ningún valor después de ejecutarse. En otras palabras:
No usa return (o usa return solo para salir del método sin retornar datos).
Se ejecuta por sus efectos secundarios (como imprimir un mensaje, modificar un objeto, o guardar datos).

```
public static void saludar() {
    System.out.println("Hola desde un método");
}
```

```
saludar();
```

códigofacilito

Puede un método void usar return?

Sí, pero solo para terminar su ejecución, no para devolver un valor.

```java
public void validarEdad(int edad) {
    if (edad < 0) {
        System.out.println("Edad inválida");
        return;  // Sale del método sin hacer lo demás
    }
    System.out.println("Edad válida: " + edad);
}
```

códigofacilito

/03

Scope de una variable y visibilidad

En programación, el término "scope" o "ámbito" se refiere a la porción de código dentro de la cual una variable, función o constante es visible y accesible. En esencia, define dónde y cómo se puede utilizar una determinada entidad dentro de un programa
Cada variable en Java vive dentro de un bloque de código: eso se llama alcance o “scope”.

códigofacilito

Variables locales

Son las que se declaran dentro de un método y solo se pueden usar dentro de ese método.

```
public static void saludar() {
    String nombre = "Tatiana";
    System.out.println("Hola " + nombre);
}
```

Si intentás usar nombre en otro método, te da error. Solo “vive” adentro de saludar()

códigofacilito

Variables de bloque

También tienen un alcance limitado solo al bloque donde se declaran

```
for (int i = 0; i < 3; i++) {
    System.out.println(i);
}
System.out.println(i); // ❌ Error: i ya no existe
```

Por bloque entendemos if, for, etc.

códigofacilito

Variables globales

Eso lo vas a ver más adelante, cuando trabajemos con clases y objetos 😉

códigofacilito

/04 Overloading

Overloading o sobrecarga de métodos significa que podés definir varios métodos con el mismo nombre, siempre y cuando tengan diferentes parámetros.
Cómo funciona la sobrecarga?
*Los métodos sobrecargados comparten el mismo nombre.
*La diferencia radica en los tipos, el número o el orden de los parámetros.
*El compilador determina qué método se debe llamar según los argumentos proporcionados en la llamada.

códigofacilito

Ejemplo de overloading

En este ejemplo:

sumar(int, int)

sumar(double, double)

sumar(int, int, int)

```java
public class Calculadora {

    public static int sumar(int a, int b) {
        return a + b;
    }

    public static double sumar(double a, double b) {
        return a + b;
    }

    public static int sumar(int a, int b, int c) {
        return a + b + c;
    }
}
```

Los tres se llaman sumar, pero Java sabe cuál usar según los parámetros que le pases

códigofacilito

Ejemplo de overloading erroneo

En este ejemplo:

restar(int, int)

restar(int, int)

```
public static int restar(int a, int b) { ... }

public static double restar(int a, int b) { ... } // ✕ Error
```

Aunque cambien el tipo de retorno, los parámetros son iguales y eso NO está permitido.

códigofacilito

/06 HANDS ON

A codear, vamos al editor de código!

REPOSITORIO CON EL CÓDIGO:
https://github.com/tatianaborda/java-course

Objetivo:
Separar las acciones del menú en métodos reutilizables

Aplicar lo aprendido sobre parámetros, retorno y scope

Temas aplicados del bloque:

Tema	Aplicación
Métodos simples	mostrarMenu(), agregarTarea()
Parámetros y retorno	leerOpcion(Scanner) devuelve int
Scope de variables	variables locales en cada método