<codoa codo/>

Agencia de Aprendizaje a lo largo de la vida

Codo a Codo inicial

BA

Les damos la bienvenida

Vamos a comenzar a grabar la clase

Formulario de presentismo

Link:

Clase 20

Estructuras de almacenamiento

- Creación del array.
- Incorporación de elementos.
- Borrado de elementos.
- Modificación de elementos.
- Recorrido un array.

Clase 21

Funciones Void sin parámetros y con parámetros.

- Conceptos.
- Declaración e invocación de funciones.
- Funciones del tipo void.
- Desafíos de clases.

Clase 22

Funciones Return

- Funciones de tipo void recibiendo parámetros.
- Análisis de problemas.
- Desafíos de clases.

Write Once, Run Anywhere

(Escríbelo una vez, ejecútalo en cualquier lugar)

Funciones, métodos, procedimientos

Métodos en Java, funciones y procedimientos.

- Los **métodos, las funciones y los procedimientos,** en Java son una herramienta **indispensable** para programar.
- Las funciones **permiten automatizar** tareas que requerimos **con frecuencia** y que además se pueden generalizar por medio de parámetros.
- **Java nos permite crear** o hacer nuestros **propios métodos** y usarlos sencillamente como si fueran **nuestra propia librería**..
- **Aprender a crear** métodos en Java y usarlos correctamente es de **gran importancia**, de esta manera separaremos nuestro código **en módulos** y según las tareas que requerimos.

Métodos en Java, funciones y procedimientos.

En Java es mucho más común hablar de métodos que de funciones y procedimientos y esto se debe a que en realidad un **método, una función y un procedimiento NO son lo mismo**, veamos la diferencia:

Funciones

- Las funciones son un **conjunto de líneas de código** (instrucciones), encapsulados en un bloque
- Usualmente, **reciben parámetros**, cuyos valores utilizan para **efectuar operaciones** y adicionalmente **retornan un valor**.
- En otras palabras una función **puede recibir parámetros** (algunas no reciben nada), y hacer uso de dichos valores para **devolver algo**.
- **Retorna un valor** mediante la instrucción **return**, si no retorna algo, entonces no es una función.
- En java las funciones usan el modificador ***static***

Funciones - Alerta spoiler

```
public class MiPrimerFuncion {
    //Zona de las funciones, procedimientos y métodos

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

    public static void main(String[] args) {

        saludo("Codo a Codo");

    }

}
```

Métodos

- Los **métodos y las funciones** en Java, son **funcionalmente idénticos**, pero su diferencia radica en el **contexto** en el que se las utiliza.
- Un método también puede recibir parámetros, efectuar operaciones con estos y retornar valores; sin embargo, **un método está asociado a un objeto**, **SIEMPRE**,
- Básicamente, un **método** es una función que **pertenece a un objeto o clase**, mientras que una **función existe por sí sola**, sin necesidad de un objeto para ser usada.
- Para crear **un método como tal** en Java, estamos obligados a **crear un objeto** si no, no se podrá usar dicho método.
- Es por ello que una función debe tener el atributo ***static***, para que **no requiera de un objeto para ser llamada**.

Métodos

```java
public class Funcion {

// Complete la funcion suma que devuelve un tipo double
        static double sumar (double num1, double num2){
            double result;
            result = num1+num2;;
            return result;
        }
```

En el archivo Funcion

```java
public class Main2 {
    public static void main(String[] args) {

        //Llamado a los métodos dentro de una variable
        double miSuma = Funcion.sumar(3, 2);

        //Instrucción de salida
        System.out.print("Mi suma vale: "+ miSuma);

    }
}
```

Llamada desde otro archivo

Procedimientos

- Los procedimientos son básicamente un **conjunto de instrucciones** que se ejecutan **sin retornar ningún valor**.
- Un procedimiento **puede o no recibir valores**.
- En el contexto de Java **un procedimiento** es básicamente un método cuyo tipo de **retorno es void** que **no nos obliga a utilizar una sentencia return**.

Procedimientos

```
    // Este procedimiento solo realizará impresiones
    static void imprimir (String mensaje){
    System.out.println(mensaje);;
}
```

NO utiliza return

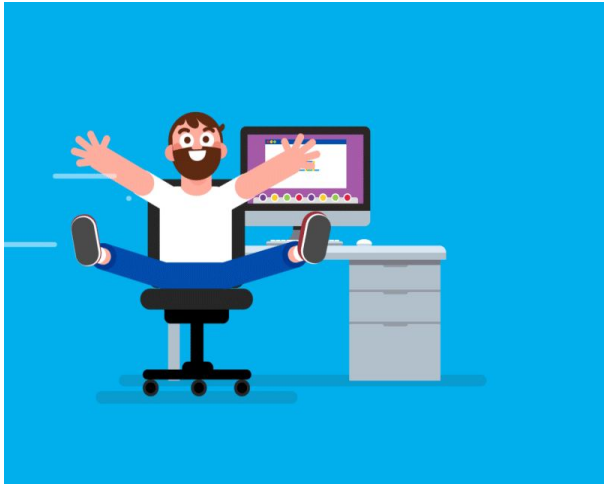
En resumen... métodos en Java

- **Un método** es un bloque de código que funciona cuando es llamado.
- En un método se **pueden pasar datos** conocidos como parámetros.
- Estos parámetros son utilizados para realizar ciertas **acciones.**
- Si el método es una **función,** entonces debe tener el atributo ***static*, y devolver un tipo específico.**
- Si el método es un **procedimiento,** entonces debe tener un atributo **static y ser del tipo void.**

¿Por qué usamos métodos?

Para reutilizar el código:
entonces definimos el código una vez y lo utilizamos muchas veces.

Así optimizamos tiempo, trabajo y líneas de código.

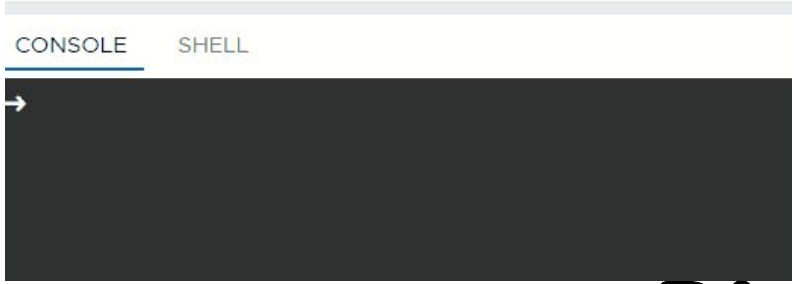


¿Cómo crear un *método void static*?

1. Un ***método static*** debe declararse **dentro** de una clase y **antes** del método main.
2. Se define con **atributo, tipo y nombre** del método, seguido de paréntesis () luego entre llaves el bloque de código a ser ejecutado.

Java proporciona algunos métodos predefinidos, como .add(), .set(), .get que los vimos en ArrayList.

```
1  public class Main {
2
3
4      public static void main(String[] args) {
5
6      }
7
8  }
9
```



Sintaxis de creación de un método void (procedimiento)

```
static void [nombre](tipo parametro1, tipo parametro2...])
{
    /*
        * Bloque de instrucciones
    */
}
```

Ejemplo de creación de un método void (Procedimiento)

```
public class FuncionesEnJava{
  static void miMetodo() {
    // código a ser ejecutado
  }
//Luego viene el método main
  }
```

- `miMetodo()` es el nombre del método
- `static` es un modificador que indica que se trata de un método de la clase, ocupando un único lugar en memoria.
- `void` tipo ***void*** significa que este método no tiene un valor de retorno, es decir cumple solamente con un procedimiento.

Ejemplo de clase

Creamos una función **Hola Mundo!**

1. Crear un Hola mundo de manera tradicional
2. Crear un método static sin recepcion de valores
3. Crear un método static con recepcion de valores
4. Crear un método static para despedir

Repo de clase

https://app.codingrooms.com/w/ccqQ9szQa8rK

Desafío de clase

Desafío de clase

Leer y entender la teoría

Repasar los prácticos dados en clases

Herramientas que utilizamos en clases

IDE IntelliJ o VSCode+plugins

No te olvides de dar el presente

Recordá:

- Revisar la Cartelera de Novedades.
- Hacer tus consultas en el Foro.
- Realizá los ejercicios obligatorios.

Todo en el Aula Virtual.