Repaso de Los Métodos en JAVA.

Un método es una estructura del lenguaje Java que nos sirve para encapsular cierta funcionalidad, la cual podamos llamar desde diferentes sitios y así no tener que repetir el código.

Los métodos definen las acciones/comportamientos de los objetos de una clase dada.
Para la creación de un método en Java debemos conocer la estructura del mismo:

```
tipo_acceso tipo_dinamico_o_no tipo_dato nombre_metodo(tipo_parametro parametro)
```

Un método generalmente usa toda esa estructura solo exceptuando la declaración de si es dinámico u estático. La primera parte de creación de un método se refiere a el tipo de acceso que puede ser:

- **protected**, acceso protegido de datos
- **private**, acceso solo de modo interno de la clase
- **public**, acceso desde una instancia externa de la clase

La segunda parte se refiere a el uso del método Java, si es estático lo cual significa que el método sería accesible desde fuera de la clase sin necesidad de instanciar la clase.

- **static**, el acceso al método es estático.

El tipo de dato es dependiente de lo que se desea como resultado del método como puede ser por ejemplo **void** si nuestro método no tiene salida alguna, o un tipo de dato específico como puede ser **double** o **int** si es una salida de tipo numérico.

El nombre de método de preferencia debe ser escrito en *notación camelCase* por ejm: (la notación camel case detalla que se debe usar en los métodos con nombres compuestos siempre la primera letra de la segunda palabra en mayúscula) Siempre comienza en minúscula.

Para la creación del método no en todos los casos es necesario argumentos pero si deseamos usar algún argumento, cada argumento deberá tener su tipo de dato y nombre de argumento.

```
1. public void miMetodo(int argumento1){
2.  //funcionamiento debe ser escrito aqui....
3. //Este es el cuerpo del metodo!!!!
4.
5. }
```

El cuerpo del método va delimitado por {}

Bueno ahora solo nos queda ver un ejemplo de cómo crear el método con Java. Para ello vamos a definir un método que nos sume dos números con Java. De esta forma, cada vez que queramos sumar dos números nos bastará con llamar a este método.

```
1. public static int sumarNumeros (int numero1, int numero2) {
2.  return numero1 + numero2;
3. }
```

En este método podemos ver que *el tipo de acceso es público*, cabe detallar que este tipo de método también es de acceso estático por tanto no necesitamos instanciar un objeto de la clase a la cual pertenece este método. También tomando en cuenta el tipo de dato a devolver del método se puede decir que trabaja con entradas de tipo entero tanto como salidas de tipo entero tal como detalla su signatura.

Para poder ver como este ejemplo funcionaria en código lo probamos:

```
1. System.out.println("Programa de Suma de números iniciando");
2. //iniciamos sumando
3. int sumando1=4234;
4. System.out.println("Sumando 1: "+sumando1);
5. //iniciamos sumando 2
6. int sumando2=64782;
7. System.out.println("Sumando 2: "+sumando1);
8. // obtenemos el resultado de la suma de los dos sumandos
9. int resultado= sumarNumeros(sumando1, sumando2);
10.System.out.println("Resultado: "+resultado);
11.//fin de ejecucion
12.System.out.println("Programa de Suma de números finalizando");
13.
```

Sobrecarga de Métodos:

Lenguajes como Java permiten que existan distintos métodos con el mismo nombre siempre y cuando su signatura no sea idéntica (algo que se conoce con el nombre de *sobrecarga*)

Ejemplo

```
System.out.prinln(…);
    - System.out.println()
    - System.out.println(boolean)
    - System.out.println(char)
    - System.out.println(char[])
    - System.out.println(double)
    - System.out.println(float)
    - System.out.println(int)
    - System.out.println(long)
    - System.out.println(Object)
    - System.out.println(String)
```

No es válido definir dos métodos con el mismo nombre que difieran únicamente por el tipo del valor que devuelven.

Como se usan los métodos?

Ejemplo de ejecución paso a paso

Los Parámetros:

return

Constructores: