**PROGRAMACIÓN ORIENTADA A OBJETOS**

**Los tipos primitivos**, llamamos así a los tipos de datos que solo nos permiten almacenar un valor. Por ejemplo, **int, float, double y char**. Cuando definimos una variable con estos tipos primitivos, solo podemos almacenar valores.

Algo a tener en cuenta cuando usamos estas clases es que no podemos usar operadores como “==”, para efectuar una comparación por igual usamos. equals**()**, por ejemplo: nombre.**equals**(“Juan”) esto nos devuelve true en el caso que en nombre se guarde la cadena “Juan” y falso en caso contrario. El equals se utiliza para comparar por igual, siempre que estemos trabajando con clases.

Si queremos comparar si un valor es mayor o menor que otro debemos usar **.compareTo()** Otra cosa a destacar es que una String a la cual no le asignamos nada tiene el **valor null**. Esto sucede con todas las clases, si definimos un elemento (objeto) de una clase inicialmente tendrá el valor **null.**

n las clases de Java se encuentran agrupadas en en paquetes, o como su nombre en inglés: **package.**

Una clase muy importante es System, en ella encontramos **System.in** y **System.ou**t, que nos permitirán interactuar con las entradas y salidas del programa.

Ya vimos que **Sistem.out.println** nos permite mostrar un dato o mensaje.

Para ingresar valores vamos a utilizar **System.in**

**Scanner:**

Clase propia de Java, que nos permite ingresar valores. Tiene métodos. Funciones ya programadas, que nos permite ingresar diferentes datos.

**PASOS:**

1. **CRECIÓN:**

Para poderlo utilizar debemos agregar la clase correspondiente, que se encuentra en **java.util**

1. **DEFINICIÓN:**

Cuando aceptamos la sugerencia de la clase Scanner, nos agrega el **import** y finalizamos dándole un **nombre** como con cualquier variable.

1. **CREACIÓN DELOBJETO SCANNER:**

Luego de la definición es necesario crear el objeto e instanciarlo

**Ej:**

Scanner lector;

Lector=new Scanner(System.in);

Se crea asociado a System.in, es decir todo ingreso de datos será interceptado por el Scanner.

**MÉTODOS DE SCANNER:**

* **nextByte()** para leer un dato de tipo byte.
* **nextShort()**para leer un dato de tipo short.
* **nextInt()**para leer un dato de tipo int.
* **nextLong()**para leer un dato de tipo long.
* **nextFloat()** para leer un dato de tipo float.
* **nextDouble()** para leer un dato de tipo double.
* **nextBoolean()**para leer un dato de tipo boolean.
* **nextLine()** para leer un string hasta encontrar un salto de línea.
* **next()**para leer un string hasta el primer delimitador, generalmente hasta un espacio en blanco o hasta un salto de línea.

**FUNCIONES**

Para definirla vamos a considerar 3 cosas:

- Qué devuelve la función

- Qué nombre tiene

- Los parámetros que necesitamos

(Cuando decimos qué devuelve nos referimos al tipo de dato que devuelve la función.)

1. **NOMBRES:**

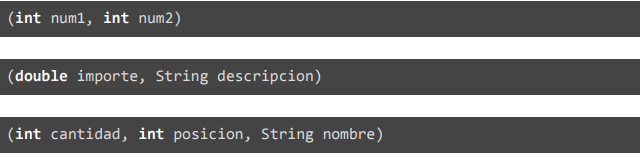
nombres descriptivos

EJ: asignarTurno, sumar, calcularTotal

1. **PARÁMETROS:**

solo que es necesario indicar el tipo de cada uno

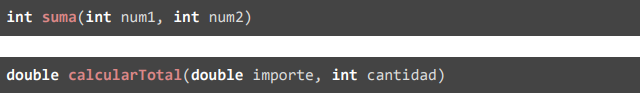
EJ:



1. **TIPO DEVUELTO:**

Las funciones pueden devolver un valor de retorno de algún tipo determinado, por ejemplo int, double, Integer, String, etc. En realidad pueden devolver cualquier cosa no solo valores, también estructuras enteras, lo veremos más adelante, pero hay que indicar que tipo tiene lo que devolvemos.

EJ:



Pero hay otro tipo de funciones, las que no devuelven nada en ese caso en donde indicamos el tipo devuelto colocaremos la palabra reservada **void.**

**(**Usamos **las funciones de tipo void**, cuando queremos que nuestra función sólo realice una serie de pasos o acciones y no nos devuelva nada**)**

**EJ:**



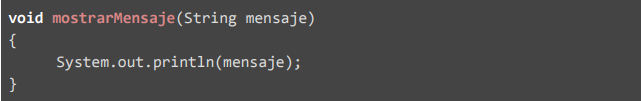
1. **DESARROLLO DE LA FUNCIÓN:**

Ahora veamos que varía en la implementación, vamos a tener dos situaciones. Que la función tenga valor de retorno o que no devuelve nada.

* En el primer caso, debemos incluir un return con el valor devuelto, el tipo de este valor tiene que coincidir con el tipo de dato indicado como tipo devuelto.



¡El valor retornado tiene que ser del tipo indicado! En caso de tener una función no tenga tipo de retorno, nos quedaría así,



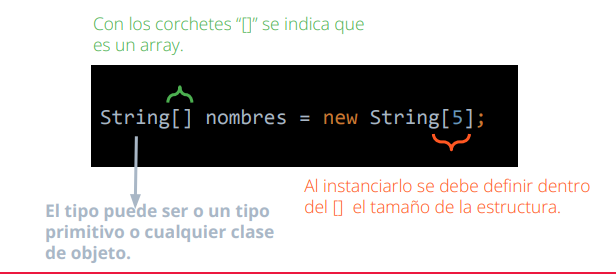
ARRAY

Los arrays son estructuras estáticas para almacenar objetos.

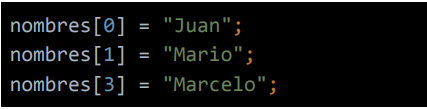
Permiten el acceso a sus elementos de forma aleatoria a través de un índice que comienza desde 0 (cero)

**(**La colección **ArrayList** tiene este mismo comportamiento y, por ello, su nombre**.)**

En Java, un array es un **objeto** y, como tal, debe usarse el operador **new** para crear una instancia, pero a diferencia de las colecciones**, los array son de longitud fija**, la cual debe definirse en la creación, siendo inmutable.

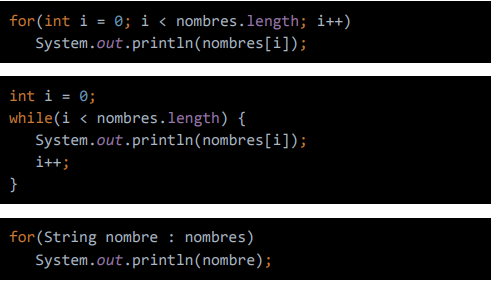


Establecemos valores a un array a través de su **índice**. Dado que es una estructura fija, no se pueden eliminar elementos.



**(Intentar acceder a un índice fuera de rango como por ejemplo nombres[10] provoca una excepción)**

Podemos recorrer un array a través de un **ciclo for, while o for each** y también utilizar la propiedad l**ength** que nos indica el tamaño del array.



**(En programación wraper = significa envolver)**

**(Atajo: ctrl + alt+ m = para repetición de funcionalidad)**

**Alt j = multiplico cursor**