

# Bootcamp Java Developer

**Fase 1 - Java Analyst** Módulo 7



# **Arreglos**



## Introducción

En ocasiones, será necesario utilizar muchas variables que pueden tener el mismo sentido. Para esos casos, Java proporciona un tipo de objeto llamado *arreglo*.

Se almacena un **conjunto de variables**, que pertenecen al **mismo tipo**, bajo un **mismo identificador**.

Al tener todos los valores bajo un mismo nombre se accede a cada elemento a través de un índice que comienza con el número cero, por lo tanto el último elemento se encuentra en el índice (longitudArreglo - 1)

#### Números

índice	0	1	2
Valor	23.3	24.6	16.0

#### **Nombres**

índice	0	1	2	3
Valor	Octavio	Mariana	Analia	Ariel



# **Arreglos Unidimensionales**

Se pueden crear arreglos unidimensionales y también de dos dimensiones (Matrices). En este curso solo veremos los de una sola dimensión, pero ambos son muy parecidos.

#### Hay dos formas de declarar un arreglo en Java:

- Las dos son correctas pero se recomienda la primera.
- Se utiliza con más frecuencia, en diferentes lenguajes.

```
tipo[] identificador;
tipo identificador[];
```



#### También hay dos maneras de inicializarlos:

- 1. La primera la utilizamos cuando tenemos los valores iniciales definidos.
- 2. Se utiliza una nueva sentencia "new" seguido del tipo de dato y, entre corchetes {}, se especifica la longitud del arreglo. Esto le indica a Java que construya un nuevo arreglo y después se le asignan los valores correspondientes en el índice indicado.

```
identificador = { valor1 , valor2 , valorn };
identificador2 = new tipo[n];
identificador2[0] = valor1;
identificador2[1] = valor2;
identificador2[n] = valorn;
```



### **Acceder a los datos**

Para acceder a cada uno de los elementos que posee el arreglo debemos indicar su posición a través del **índice**.

```
System.out.println("Valor: " + (identificadorArreglo[0]));
```

Es decir, se accede al arreglo: **elemento por elemento**.

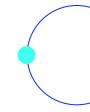
Existe una estructura mucho más cómoda para hacer eso: los **bucles**. Específicamente, uno donde se conozca la cantidad de iteraciones a realizar.



Aunque es posible colocar el número en el ciclo directamente, no es recomendable. Para evitarlo se accede a un atributo que poseen los arreglos que indica su longitud. Se utiliza el identificador del arreglo seguido de ".length".

```
System.out.println("Longitud del Arreglo: " + (identificadorArreglo.length));
```

```
for (int i = 0; i < identificadorArreglo.length; i++) {
    System.out.println("Valor del Arreglo " + identificadorArreglo[i] );
}</pre>
```





# **Utilidades para los arreglos**

Existen métodos que ayudan a trabajar con arreglos en Java. Para utilizarlos, primero se debe importar la biblioteca "java.util.Arrays".

Nombre	Descripción	
copyOf	Copia un arreglo y lo devuelve en un nuevo arreglo.	
copyOfRange	Copia un arreglo dado un rango y lo devuelve en un nuevo arreglo.	
equals	Indica si dos arreglos son iguales.	
sort	Ordena el arreglo de forma natural.	
toString	Muestra el contenido del arreglo.	

```
// paquete al que pertenece la clase
package identificadorPaquete;

// bibliotecas importadas
import java.util.Arrays;

// declaracion de la clase
public class IdentificadorClase {
```



# **Utilidades para los arreglos**

```
int numerosOriginal[] = { 8, 9, 5, 12, 14, 50, 60, 16};
// Copia el arreglo numerosOriginal hasta la quinta posicion(este ultimo no incluido) devuelve un arreglo
int numeros2[] = arreglos.copvOf(numerosOriginal, 4);
// Copia el arreglo numerosOriginal de la tercera hasta la octava posicion, devuelve un arreglo
int numeros3[] = arreglos.copyOfRange(numerosOriginal, 2, 6);
// Compara si los arreglos numeros2 y numeros3 son iguales
System.out.println("Metodo equals: " + arreglos.equals(numeros2, numeros3));
arreglos.sort(numerosOriginal);
// Muestra los valores que contiene el arreglo numerosOriginal
System.out.println(arreglos.toString(numerosOriginal));
```





## Valores por defecto

¿Qué ocurre si se inicializa el arreglo con un tamaño n y luego se decide acceder a los elementos, antes de asignarle valores?

```
identificador2 = new tipo[n];
```

Para **evitar errores**, Java inicializa los arreglos de la siguiente manera:

- 1. Para **números** el valor cero "0" o "0.0".
- 2. Para **caracteres** el espacio en blanco ' '.
- 3. Para booleanos el valor false.
- 4. Para los **objetos "string"** la ausencia de valor, y se mostrará la palabra **"null"**, que es distinto de vacío.



# **Bonus Consola**



### Introducción

Existe una utilidad que ayuda a capturar los datos de la consola. Para ello, se debe importar la biblioteca "java.util.Scanner".

```
// paquete al que pertenece la clase
package identificadorPaquete;

// bibliotecas importadas
import java.util.Scanner;

// declaracion de la clase
public class IdentificadorClase {
```

Para poder obtener los **datos ingresados por el usuario**, se necesita crear un elemento con la siguiente estructura:

```
// la clase scanner para capturar valores del teclado
Scanner identificador = new Scanner(System.in);
```





## Métodos

Cada vez que se use uno de los **métodos de la clase Scanner**, la consola se quedará esperando que el usuario ingrese una información.
Es recomendable mostrar por pantalla un mensaje que indique qué dato debe ingresar:

```
// ejemplo de como capturar un numero entero
System.out.println("Ingrese el numero entero");
int identificadorEntero = identificador.nextInt();
```

Tipo de Dato	Método	
boolean	nextBoolean()	
byte	nextByte()	
double	nextDouble()	
float	nextFloat()	
int	nextInt()	
long	nextLong()	
short	nextShort()	
String	next()	
String	nextLine()	



¡Sigamos trabajando!