

# Tema 4 – Estructuras básicas de datos

1º Ciclo Formativo Grado Superior - Desarrollo de Aplicaciones Web

Módulo: Programación

Prof. José de la Torre López (adaptación Antonio Hernández)

# ÍNDICE

- Arrays
  - Unidimensionales
  - Multidimensionales
- La clase String
- Arrays de Strings
- Parámetros de la línea de comandos

# Introducción - Arrays

¿Y si tenemos que guardar muchas variables del mismo tipo?

```
int nota1;  
int nota2;  
int nota3;  
int nota4;  
int nota5;  
int nota6;
```

...

```
int notas[10];
```

# Arrays

- Los arrays son una colección de datos del mismo tipo al que se le pone un nombre (por ejemplo **nota**). Para acceder a un dato individual de la colección hay que utilizar su posición. La posición es un número entero, normalmente se le llama **índice** (por ejemplo **nota[4]** es el nombre que recibe el cuarto elemento de la sucesión de notas).
- Otros nombres habituales además de **arrays** son: listas, matrices, arreglos,...



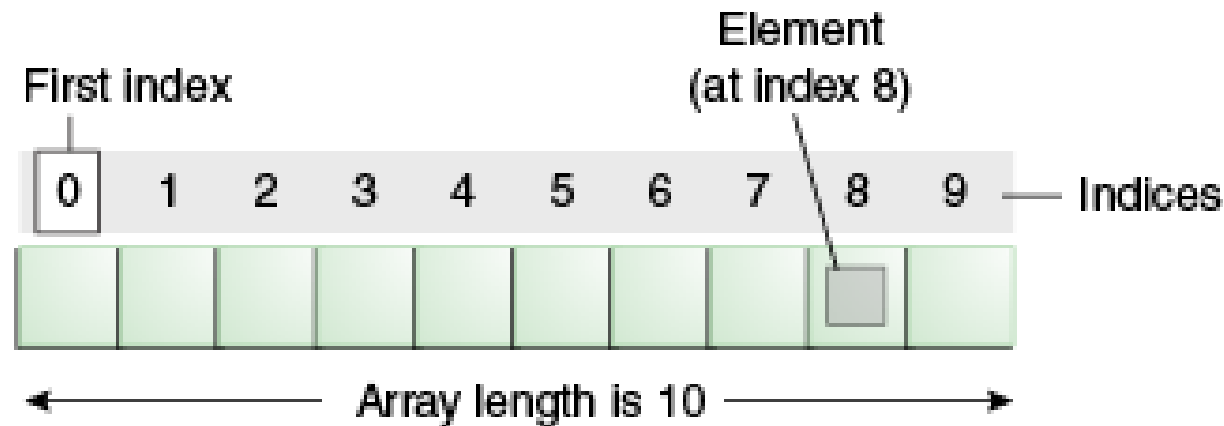
# Arrays – Ejemplo gráfico

- La mejor manera de entender los arrays es imaginarse una estantería:



# Arrays - Unidimensionales

- Nuestra estantería tiene una única balda:



Declaración:

tipo nombre[];

ó

tipo[] nombre;

# Arrays – Unidimensionales. Declaración.

```
double cuentas[]; //Declara un array que almacenará valores  
// doubles
```

Con esto todavía no tiene valores asignados, ni siquiera se sabe su tamaño

```
int notas[]; //sería válido también int[] notas;  
notas = new int[3]; //indica que el array constará de tres  
//valores de tipo int
```

```
int notas[]=new int[3];
```

Con esto todavía no tiene valores asignados, pero ahora sí se sabe su tamaño, se crean 3 elementos y se le asigna por defecto el valor 0 a cada uno.

```
notas[2]=8;
```

Con esto las posiciones [0] y [1] todavía no tienen valores asignados, la [2] sí (=8)

# Arrays – Unidimensionales. Declaración.

```
int notas[] = {8, 7, 9};  
int notas2[] = new int[] {8,7,9}; //Equivalente a la anterior
```

Esto **declara e inicializa** un array de tres elementos.  
En el ejemplo lo que significa es que:

***notas[0]*** vale **8**

***notas[1]*** vale **7**

***notas[2]*** vale **9**.



# Arrays – Unidimensionales. Declaración.

Recuerda que por buenas prácticas de programación en la declaración elegiremos **poner los corchetes al lado del tipo de dato**.  
De esta manera queda más claro que es una variable un tipo

*int [] notas;*

MEJOR QUE:

~~*int notas[];*~~

# Arrays

- En Java (como en otros lenguajes) el primer elemento de un array es el cero. El primer elemento del array notas, es ***notas[0]***. Se pueden declarar arrays a cualquier tipo de datos (enteros, booleanos, doubles, ... e incluso objetos como se verá más adelante).
- La ventaja de usar arrays (volviendo al caso de las notas) es que gracias a un simple bucle **for** se puede recorrer fácilmente todos los elementos de un array :

```
//Calcular la media 18 notas  
suma=0;  
for (int i=0;i<=17;i++){  
    suma+=nota[i];  
}  
media=suma/18;
```

# Arrays

- A un array se le puede inicializar las veces que haga falta:
- Pero hay que tener en cuenta que el segundo **new** hace que se pierda el contenido anterior. De hecho elimina ese contenido.

```
int notas[]=new notas[16];  
...  
notas=new notas[25];
```

# Arrays - Ejemplo

## instrucción

```
int notas[];
```

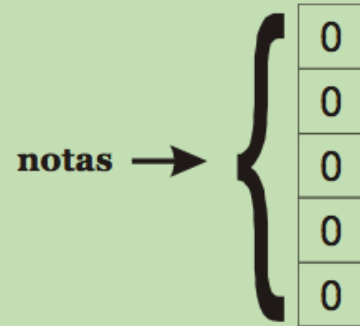
```
notas=new[5];
```

```
for(int i=0;i<5;i++){  
    nota[i]=i*i;  
}
```

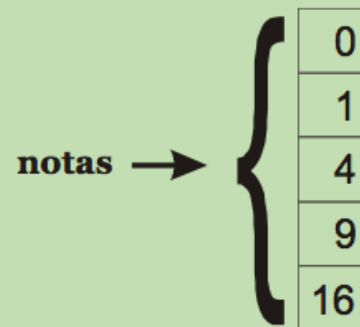
## efecto

se crea la referencia **notas**

se crea un array de cinco números enteros y **notas** hace referencia a ellos:



se asignan valores en el array

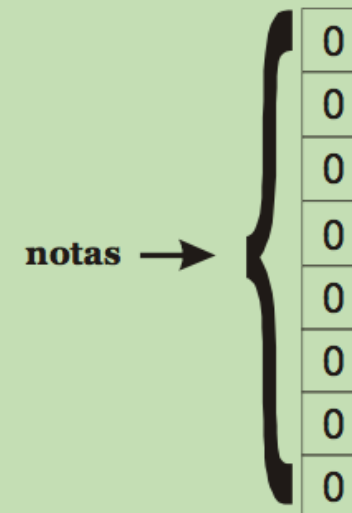
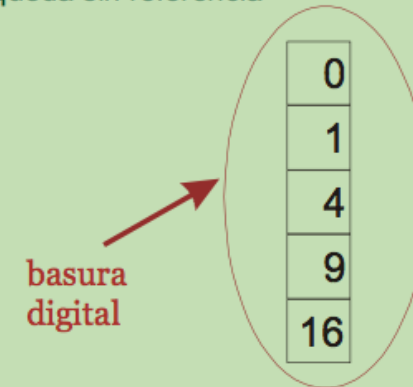


## instrucción

```
notas=new[8];
```

## efecto

se crea un nuevo array de ocho enteros **notas** le hace referencia, el anterior se queda sin referencia



# ¿Qué ocurre con la basura?

## Garbage Collector

Se puede forzar a que actúe llamando a\*:

`System.gc();`

\*Aunque dicha instrucción no garantiza la recolección de basuras. Sólo se suele utilizar cuando nuestro código genera basura más rápidamente de lo que el recolector puede eliminar.

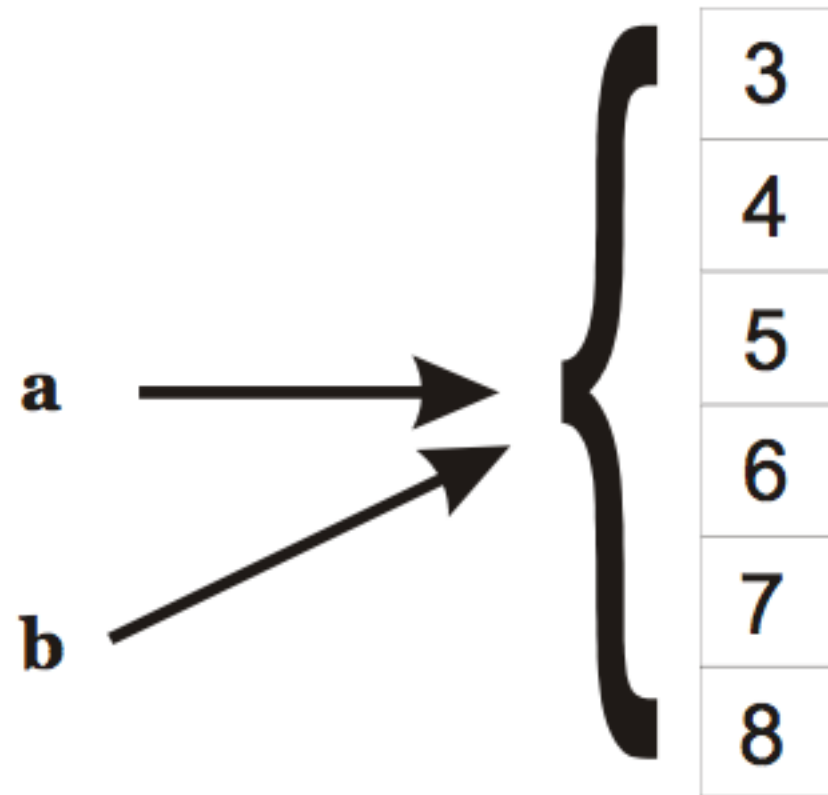
# Garbage Collector

- Java se ha molestado mucho en perfeccionarlo → Mejor no llamarlo
- Es muy eficiente y no hay que preocuparse de él en JAVA.

# Arrays – Asignación

- Un array se puede asignar a otro array (si son del mismo tipo):

```
int a[];  
int b[]=new int[]{3,4,5,6,7,8};  
a=b;
```



# Arrays – Asignación

- Esta asignación provoca que cualquier cambio en ***a*** también cambie el array ***b*** (ya que, de hecho, es el mismo array). Ejemplo:

```
int a[]={3,3,3};  
int b[];  
b= a;  
b[0]=8;  
System.out.println(a[0]);//Escribirá el número 8
```





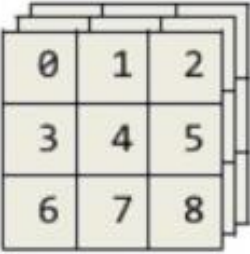
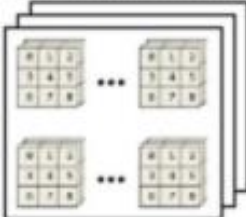
# Arrays – Comparación

- Finalmente, como detalle final a este tema, el operador de igualdad (==) se puede utilizar con arrays, pero nuevamente no compara el contenido sino si las referencias señalan al mismo array.

```
int a[]={3,3,3};  
int b[]={3,3,3};  
System.out.println(a==b); //escribe false, aunque ambos arrays  
                           //tienen el mismo contenido  
  
int c[]=b;  
System.out.println(b==c); //escribe true
```

# Arrays multidimensionales

- Pasamos de una balda a varias baldas (estantería)... ¡o muchas más dependiendo de sus dimensiones!

Dimensions	Example	Terminology
1		Vector
2		Matrix
3		3D Array (3 <sup>rd</sup> order Tensor)
N		ND Array

*Ahora es cuando  
hacéis la  
croqueta...*

# Arrays multidimensionales

- **nota** es un array que contiene arrays de enteros. La primera dimensión podría significar el número de un aula y el segundo el número del alumno. De modo que, por ejemplo, **nota[2][7]** significaría la nota del alumno número ocho del tercer aula.

```
int nota[][];
```

# Arrays multidimensionales

```
notas = new int[3][12];    //notas está compuesto por 3 arrays
                           //de 12 enteros cada uno
notas[0][0]=9;             //el primer valor es un 9
```

```
int notas[][]=new int[5][]; //Hay 5 arrays de enteros (aún por definir)
notas[0]=new int[100];      //El primer array es de 100 enteros
notas[1]=new int[230];      //El segundo de 230
notas[2]=new int[400];      //...
notas[3]=new int[100];
notas[4]=new int[200];
```

Hay que tener en cuenta que en el ejemplo anterior, **notas[0]** es un array de **100** enteros. Mientras que **notas**, es un array de **5** arrays de enteros.

# Arrays multidimensionales

```
int nota[][][]=new int[5][7][90];
```

Esta instrucción crea cinco arrays de siete arrays de noventa números enteros.

Es un array de arrays de arrays (un tanto complicado de describir) o también un array tridimensional.

Es raro necesitar más de tres dimensiones, pero sería posible hacerlo.

# Cosas interesantes sobre los arrays

```
int a[]=new int [17];  
int b[][]=new int [30][7];  
  
System.out.println(a.length);    //Escribe: 17  
System.out.println(b.length);    // Escribe: 30  
System.out.println(b[7].length); // Escribe: 7
```

```
for (int i = 0; i < x.length; i++) {  
    System.out.print(x[i]+" ");  
}  
System.out.println();
```

# Clase Arrays

En el paquete **java.util** se encuentra una **clase estática** llamada **Arrays**.

```
Arrays.método(argumentos);
```

# Clase Arrays - fill

```
int a[]=new int[23];  
Arrays.fill(a,-1);    //Todo el array vale -1
```



# Clase Arrays - equals

```
int a[] = {2,3,4,5,6};  
int b[] = {2,3,4,5,6};  
int c[] = a;  
System.out.println(a==b);           //false  
System.out.println(Arrays.equals(a,b)); //true  
System.out.println(a==c);           //true  
System.out.println(Arrays.equals(a,c)); //true
```

# Clase Arrays - sort

```
int x[]={4,5,2,3,7,8,2,3,9,5};  
Arrays.sort(x,2,5);    //El array queda {4 5 2 3 7 8 2 3 9 5}  
Arrays.sort(x);        //Estará completamente ordenado
```

# Clase Arrays – binarySearch

Permite buscar un elemento de forma ultrarrápida en un array ordenado (en un array desordenado sus resultados son impredecibles).

Devuelve el índice en el que está colocado el elemento.

```
int x[]={1,2,3,4,5,6,7,8,9,10,11,12};  
Arrays.sort(x);  
System.out.println(Arrays.binarySearch(x,8));    //Escribe: 7
```

# Clase Arrays – copyOf

```
int a[] = {1,2,3,4,5,6,7,8,9};  
int b[]=Arrays.copyOf(a, a.length);//b es {1,2,3,4,5,6,7,8,9}  
int c[]=Arrays.copyOf(a, 12);      //c es {1,2,3,4,5,6,7,8,9,0,0,0}  
  
int d[]=Arrays.copyOf(a, 3);      //d es {1,2,3}
```

# Clase String

- Declaración:

```
String texto1 = "¡Prueba de texto!";
```

- Concatenación

```
String texto2 = "Este es un texto que ocupa " +  
                "varias líneas, no obstante se puede " +  
                "perfectamente encadenar";
```

# Clase String

## Char[] vs String

```
char[] palabra = {'P','a','l','a','b','r','a'}; //palabra es un array de  
caracteres  
  
//no es lo mismo que un  
String
```

a partir de ese array de caracteres podemos crear un String:

```
String cadena = new String(palabra); //mediante el operador new  
//podemos convertir el array de  
caracteres en  
//un String
```

# Formas de comparar cadenas en Java

- **String.Equals(String s)**
- **String.EqualsIgnoreCase(String s)**
- Operador == → No...
- **String.CompareTo(String)**

# Clase String – Documentación oficial

<https://docs.oracle.com/javase/7/docs/api/java/lang/String.html>

Google: Java string



# Ejercicio de clase – Investigación String

- Buscar los métodos de String que sirven para:
  - Obtener una subcadena dentro de una cadena.
  - Indique el índice donde aparece una subcadena.
  - Indique si la cadena termina con una subcadena.
  - Indique si la cadena empieza con una subcadena.
  - Reemplace un carácter dentro de la cadena
  - Devuelva la cadena en mayúsculas
  - Devuelva la cadena en minúsculas
  - Pase la cadena a un array de caracteres.

# Ejercicio en clase

- Dada la cadena de texto “*Ratata es un pokemon de tipo Normal que evoluciona a Raticate.*” Se pide utilizar **TODOS** los métodos anteriores en un programa Java y ver cómo funcionan.

# Arrays de Strings

- Los arrays se aplican a cualquier tipo de datos y eso incluye a los Strings. Es decir, se pueden crear arrays de textos.

```
String[] texto=new String[4];
texto[0]="Hola";
texto[1]=new String();
texto[2]="Adiós";
texto[3]=texto[0];
for (int i = 0; i < texto.length; i++) {
    System.out.println(texto[i]);
}
/*se escribirá:
    Hola

    Adiós
    Hola
*/
```

# ¿Nunca os habéis preguntado para qué está el args del main?

- Uno de los problemas de trabajar con entornos de desarrollo es que nos abstrae quizá en exceso de la realidad. En esa realidad un programa java se ejecuta gracias a la orden **java programa**; orden que hay que ejecutar en la línea de comandos del sistema.

**java** Veces Hola 10

- Eso podría significar que invocamos al programa llamado **Veces** y le pasamos dos parámetros: el primero es el texto a escribir () y el segundo el número de veces.
- El método **main** es el encargado de recoger los parámetros a través de la variable **args**. Esta variable es un array de Strings que recoge cada uno de los parámetros. De tal modo que **args[0]** es un String que recoge el primer parámetro, **args[1]** el segundo, etc. Si no hay parámetros, **args.length** devuelve cero.

# ¿Nunca os habéis preguntado para qué está el args del main?

```
public class Veces {  
    public static void main(String[] args) {  
        if(args.length>=2) {  
            int tam=Integer.parseInt(args[1]);  
            for (int i = 1; i <=tam; i++) {  
                System.out.println(args[0]);  
            }  
        }  
        else {  
            System.out.println("Faltan parámetros");  
        }  
    }  
}
```

# Ok... pero cómo se hace esto con Eclipse

MENÚ:

Run → Run configurations...

