

#### **UNIDAD DE TRABAJO 4. ESTRUCTURAS ESTÁTICAS CON LENGUAJE JAVA**

1. Tablas o arrays. Características.
2. Tipos de Arrays.
3. Declaración de Arrays.
4. Creación de Arrays.
5. Inicialización
6. Operaciones con Arrays.
  1. Acceso a elementos
  2. Recorrido de un array.
  3. Búsqueda de un elemento
  4. Ordenación de un array.

#### **TABLAS O ARRAYS. CARACTERÍSTICAS**

**TABLA o ARRAY** .- estructura de datos constituida por un número fijo de elementos, todos ellos del mismo tipo y ubicados en direcciones de memoria físicamente contiguas.

**ELEMENTOS DE UN ARRAY** .- también denominado componente, es cada uno de los datos que forman parte integrante del array.

**NOMBRE DE UN ARRAY** .- identificador utilizado para referenciar el array y los elementos que lo forman.

**TIPO DE DATO DE UN ARRAY** .- marca el tipo de dato básico que es común a todos y cada uno de los elementos o componentes que forman dicha estructura (numérico, carácter, etc.).

**ÍNDICE** .- valor numérico entero y positivo a través del cual podemos acceder directa e individualmente a los distintos elementos de un array, pues marca la situación relativa de cada elemento o componente dentro de la misma.

**TAMAÑO DE UN ARRAY** .- el tamaño o longitud de una tabla viene determinado por el número máximo de elementos que la forman, siendo el tamaño mínimo un elemento, y el tamaño máximo n elementos. Antes de utilizar un array, se ha de indicar su tamaño. A partir de ese momento, el tamaño no puede cambiar.

**ACCESO A LOS ELEMENTOS O COMPONENTES DE UN ARRAY (ÍNDICE)** .- los elementos o componentes de un array tratados individualmente son auténticos datos básicos que reciben el mismo trato que cualquier otra variable, con un tipo de dato que coincide con el tipo del array y una denominación propia que les distingue del resto de elementos o componentes que constituyen dicha estructura. Para acceder o referenciar un elemento en particular es suficiente con indicar el nombre del array seguido del índice (posición relativa que ocupar dicho elemento dentro del array) correspondiente a dicho elemento entre corchetes.

**DIMENSIÓN DEL ARRAY** .- viene determinado por el número de índices que necesitamos para acceder a cualquiera de los elementos que la forman.

### **TIPOS DE ARRAYS.**

Existen arrays de una dimensión (**unidimensionales**) o vectores. Para acceder a un dato del array se ha de indicar una única posición.

Por ejemplo, Un array que guarda las notas que han obtenido 20 alumnos en el módulo de programación en la primera evaluación.

Y tablas de más de una dimensión, hasta n dimensiones o **multidimensionales**.

En este caso particular de 2 dimensiones se le denomina **matriz o array bidimensional**. Para acceder a un dato del array se han de indicar dos posiciones.

Se utilizan cuando los datos se van a representar por filas y columnas.

Por ejemplo, las notas que han obtenido 20 alumnos en 3 evaluaciones. La fila representa a alumno y la columna a una evaluación, y el valor es una nota.

Los array multidimensionales son aquellos donde para poder acceder a un determinado dato hacen falta tres o más posiciones.

### **DECLARACIÓN DE ARRAYS.**

#### **Una dimensión**

Tipo\_base nombre\_array [];  
o bien  
Tipo\_base [] nombre\_array;

Ejemplo de declaración de un array llamado tabla cuyos elementos serán de tipo int.  
`int tabla[];`

#### **Varias dimensiones**

Tipo\_base nombre\_tabla [][]..[];  
o bien  
Tipo\_base [][]..[] nombre\_tabla;

Ejemplo de declaración de un array bidimensional y tipo base float.

```
float notas[][];  
o bien  
float [][]notas;
```

### **CREACIÓN DE ARRAYS**

Los arrays se crean con el operador new.

1. Para arrays unidimensionales o vectores:  
 nombre\_array = new Tipo\_base[Tamaño];  
 donde Tamaño es el número de elementos.

Por ejemplo,  
 float notas[];  
 notas=new float[10];

```
int [] temperaturas;  
temperaturas=new int[7];
```

2. Para array bidimensionales o matrices:  
 nombre\_array = new Tipo\_base[Tamaño1][Tamaño2];

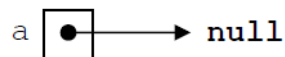
Por ejemplo,  
 int [][]temperaturas;  
 temperaturas=new int[12][2]; //las filas representan los meses, y la columnas la máxima y la mínima.

Por ejemplo,

```
int a[];  
int b[] = new int[3];  
a = new int[3];  
a[0] = 1;  
b[1] = a[0];  
a = b;  
b[0] = a[1];
```

### **Solución:**

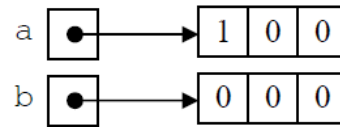
```
int a[];
```



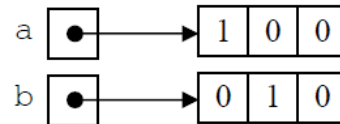
```
int b[] = new int[3];
```



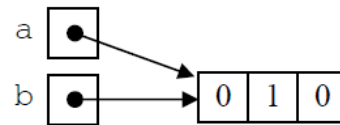
`a[0] = 1;`



`b[1] = a[0];`



`a = b;`



Java permite que cada fila tenga un número diferente de columnas.

Por ejemplo,

```

int [][] matriz;
matriz = new int[3][];
matriz[0]=new int[3];
matriz[1]=new int[4];
matriz [2]=new int[2];
  
```

Ejercicios:

1. Declara y crea un tipo de variable que sea capaz de contener el número de alumnos matriculados en 4 institutos en 5 niveles de estudios.: ESO, BACHILLERATO, CGM, CGS, PCPI.
2. Declara y crea un tipo de variable que sea capaz de contener la nota que han obtenido 4 alumnos en los diferentes módulos en los que están matriculados. Sabemos que el primer alumno está matriculado en 4 módulos; el segundo, en 7 módulos; el tercero, en 3 módulos, y el cuarto, en 2 módulos.

Si el tipo de la tabla no es un tipo de básico de Java (int, double...), sino que es una clase, estaríamos hablando de arrays de objetos.

Clase tabla[];

tabla=new Clase[num\_objetos]; /\* Lo que hace aquí es crear un número de referencias (punteros) que apuntan al espacio donde se van a guardar los datos del objeto posteriormente, habría que reservar memoria para cada objeto, con el fin de poder guardarsus datos \*/

Por ello, hay que hacer lo siguiente:

```
for(int i=0;i<tabla.length;i++) tabla[i]=new Clase();
```

Y a partir de aquí ya se podrían guardar los datos de los objetos.

### INICIALIZACIÓN

Para dar valores al array en el mismo momento de la declaración, sería:

En un vector:

```
int vector[]={1,2,3,4,5,7};  
String nombres[]={“María”, “David”};
```

En una matriz:

```
int matriz[][]={{1,2,3},{4,5,6}};
```

Java determina el tamaño del array en función de los valores asignados y hace la reserva de memoria sin tener que hacer el new.

Ejercicios:

1. Queremos guardar en la memoria el nombre de alumnos matriculados (por módulos) en el ciclo de “Desarrollo de Aplicaciones Multiplataforma”.

Sabemos que los alumnos matriculados por cada módulo son:

21 alumnos matriculados en Sistemas Informáticos.

17 alumnos matriculados en Bases de Datos.

30 alumnos matriculados en Programación.

14 alumnos matriculados en Lenguaje de Marcas.

25 alumnos matriculados en Entornos de Desarrollo.

Según estos datos, haz la declaración e inicialización que creas conveniente para poder guardar toda esta información.

### OPERACIONES CON ARRAYS

#### Acceso a los elementos:

```
nombre_vector[pos]
```

**En Java, el índice del primer componente de un array es siempre 0.**

**El tamaño del vector se puede conocer utilizando la propiedad length.**

**Por tanto, el índice del último componente es nombre\_vector.length -1, y su valor es nombre\_vector[nombre\_vector.length - 1].**

Ejemplo 1:

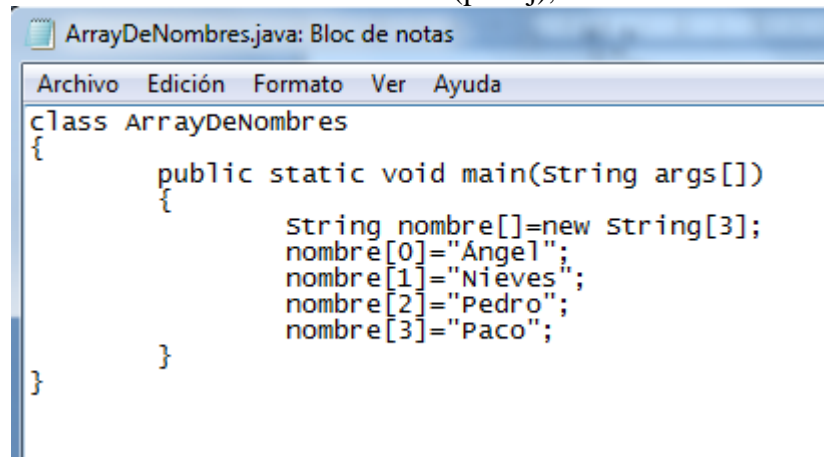
```
float notas[]=new float[3];  
Para acceder a todas las posiciones del vector, sería:  
notas[0], notas[1] y notas[2].
```

Ejemplo 2:

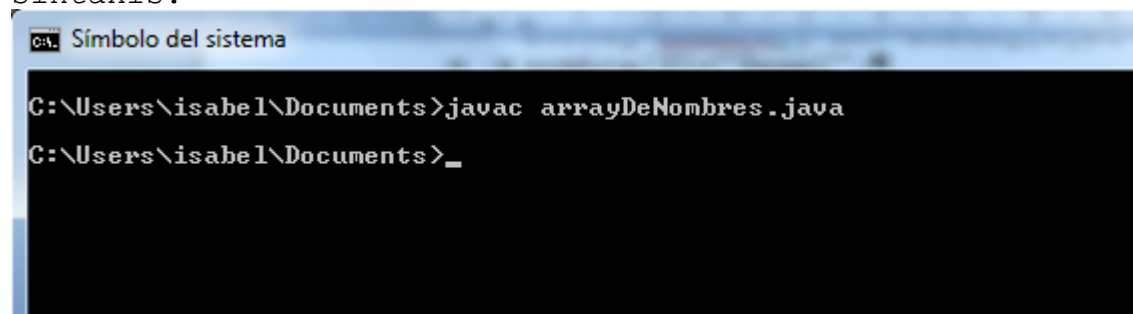
Supongamos que tenemos este código:

```
class ArrayDeNombres
{
    public static void main(String args[])
    {
        String nombre[]=new String[3];
        nombre[0]="Ángel";
        nombre[1]="Nieves";
        nombre[2]="Pedro";
        nombre[3]="Paco";
    }
}
```

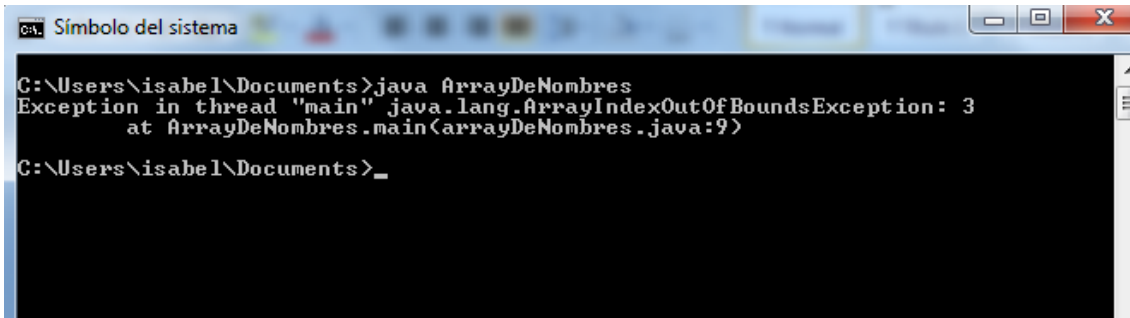
Lo escribimos en el Bloc de notas (por ej),



Si compilamos este programa, no dará ningún error de sintaxis:



Sin embargo, a la hora de ejecutarlo, nos daría un error como este: "ArrayIndexOutOfBoundsException". Este mensaje de error significa "desbordamiento de array", es decir, que tratamos de acceder a una posición del array que no existe.



```

C:\Users\isabel\Documents>java ArrayDeNombres
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 3
    at ArrayDeNombres.main(ArrayDeNombres.java:9)
C:\Users\isabel\Documents>_

```

Para acceder a los elementos de una matriz hay que indicar dos posiciones; por eso, se llaman tablas bidimensionales. Hay que indicar en qué fila y columna, en ese orden, está el dato al que queremos acceder:

`matriz[posFila][posColumna]`

Java empieza a enumerar las filas por el 0, al igual que las columnas; por ello, `matriz[0][0]` es el primer elemento de la matriz.

Para saber cuál es el tamaño de una tabla bidimensional se usa la propiedad `length`:

`matriz.length` nos da el número de filas

`matriz[posFila].length` nos da el número de columnas que tiene la fila `posFila`.

Ejercicio:

*¿En qué posición estará el último elemento de la matriz?*

El uso de un array de objetos es igual que si fuera un solo objeto, pero siempre indicando la posición que ocupa dentro de la tabla:

`tabla[i].nombre_método_clase();`

### **Carga de datos.**

Consiste en asignar un valor a cada uno de los elementos del array.

```

...
int tabla [6],i;
Scanner teclado=new Scanner(System.in);
for(i=0;i<tabla.length;i++) tabla[i]=teclado.nextInt();
...
0
tabla[0]=1;
tabla[1]=2;
tabla[2]=3;
tabla[3]=4;
tabla[4]=5;
tabla[5]=6;

```

### **Recorrido secuencial.**

Consiste en procesar todos los elementos de el array según el orden de su índice.

```

for(i=0;i<tabla.length;i++) System.out.print(tabla[i]);

```

o

```
float notas[][]={{ 12,13,23},{45,34,67,89,123,12,34,56}};
for(int i=0;i<notas.length;i++)
{
    System.out.println("Los valores que hay en la fila "+i+" son: ");
    for(int j=0;j<notas[i].length;j++)
        System.out.print(notas[i][j]+" ");
}
```

En este ejemplo estamos recorriendo la tabla por filas.

### **Búsqueda de un elemento**

Consiste en localizar un valor dentro del array. Esta operación se puede realizar de dos formas dependiendo de que el array esté ordenado o no.

#### **Secuencial,**

Cuando el array ordenado o no está ordenado.

```
for(int i=0;tabla[i]!=valor && i<tabla.length; i++);
if (i==tabla.length)
    System.out.print(valor + " no encontrado");
else
    System.out.print(valor + " encontrado");
o
boolean encontrado=false;
for(int i=0;i<tabla.length && !encontrado;i++)
{
    if (tabla[i]==valor) encontrado=true;
}

if (encontrado)
    System.out.print(valor + " encontrado");
else
    System.out.print(valor + " no encontrado");

O
boolean encontrado=false;
for(int i=0;i<tabla.length;i++)
{
    if (tabla[i]==valor) {encontrado=true; break;}
}

if (encontrado)
    System.out.print(valor + " encontrado");
else
    System.out.print(valor + " no encontrado");
```

**Binaria o Dicotómica, sólo si el array está ordenado.**

```
int izqda, drcha, centro;
izqda=0;
```



```
drcha=tabla.length - 1;
centro=(drcha + izqda) /2;
while (izqda <drcha && tabla[centro]!=valor)
{
    if (tabla[centro]>valor) drcha=centro -1;
    else izqda=centro + 1;
    centro=(izqda + drcha)/2;
}
if (tabla[centro]==valor)
    System.out.print(valor + " encontrado");
else
    System.out.print(valor + " no encontrado");
```

O

```
int izqda, drcha, centro;
izqda=0;
drcha=tabla.length - 1;
centro=(drcha + izqda) /2;
encontrado=false;
while(izqda <drcha)
{
    if (tabla[centro]==valor){encontrado=true; break;}
    else if(tabla[centro]>valor) drcha=centro -1;
    else izqda=centro + 1;
    centro=(izqda + drcha)/2;
}
if (encontrado)
    System.out.print(valor + " encontrado");
else
    System.out.print(valor + " no encontrado");
```

**Ordenación.**

Consiste en intercambiar el contenido de los elementos que la componen de manera que un recorrido secuencial del array muestre los valores de forma ordenada ascendente o descendente.

Existen muchos algoritmos de ordenación de tablas, unos son más eficientes que otros dependiendo del estado inicial del array y/o del número de elementos de que consta.