



Java

Tema 3

Arrays

1

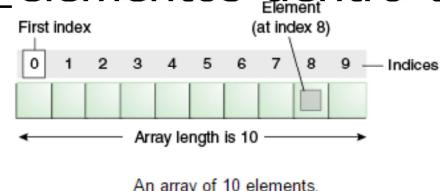


Contenido

- ❖ Concepto de **array** (vector)
- ❖ Array de **una dimensión**
- ❖ Array de **dos dimensiones**
- ❖ **Acceso** a los elementos de un array
- ❖ Longitud/tamaño de un array. **length**
- ❖ **Bucle** for-each
- ❖ API de la **clase Arrays**
- ❖ **Copia** de arrays

Concepto de Array (vector)

- Un **array** (vector) es un grupo de variables del mismo tipo de datos a las que se hace referencia por medio de un nombre común.
- Los arrays en Java se implementan como objetos de la **clase Array**.
- Un array puede contener tipos primitivos u objetos.
- Los arrays pueden tener una o más dimensiones o índices.
- Los índices indican la posición de los elementos dentro del array.



3

Arrays de una dimensión

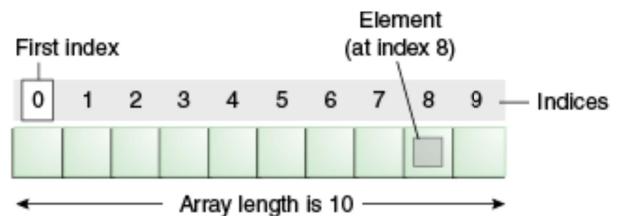
- Un **array** de una dimensión es un grupo de variables relacionadas todas del mismo tipo de datos, donde se usa un solo índice.
- Declaración y creación de un array de una dimensión

```
tipoDatos[] nombre_array = new tipoDatos[num_elementos];
```

- *TipoDatos* declara el tipo de datos de cada elemento del array.
- *Num_elementos* define el tamaño, la cantidad total de elementos del array

Ejemplo:

```
int[] vector = new int[10];
```



4

- También se permite esta declaración pero no es recomendable usarla:

```
tipoDatos nombre[];
```

Arrays de una dimensión

□ Inicialización:

```
tipo[] nombre_array = { valor1, valor2, . . . , valorN };
```

- Los valores son asignados en orden, de izquierda a derecha.
- En estos casos, no es necesario hacer uso del operador `new`. El array se construye automáticamente a partir del número de valores.

Ejemplos:

```
int[] numeros = {10, 20, 30, 40, 50}      10 | 20 | 30 | 40 | 50
```

```
String[] nombres = {"Pedro", "Luis", "Juan"};
```

Pedro	Luis	Juan
-------	------	------

5

Acceso a los elementos de un array

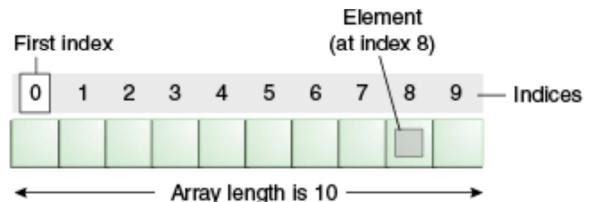
□ Para acceder a un elemento de un array se utiliza el nombre del array seguido de corchetes `[]` para cada dimensión, con el índice del elemento que queremos acceder.

`nombre_array[posicion]`

□ La primera posición de un array para cada dimensión tiene índice 0. La última posición tiene como índice numElementos-1

Ejemplo 1:

```
numeros[2] = 10;
alumnosGrupo[5] = "Pedro";
int a = numeros[0];
if (numeros[4] == 5) {
    ...
}
```



Ejemplo 2:

```
String [] nombres= { "Juan", "Pedro", "Victoria" };
System.out.println(nombres[0]);
System.out.println(nombres[1]);
System.out.println(nombres[2]);
```

6



Acceso a los elementos de un array

```
int[] numeros = new int [100];
```

- **Rellenar** un array de una dimensión

```
for (int i=0; i < 100 ; i++) {  
    numeros[i] = i;  
}
```

- **Mostrar** un array de una dimensión

```
for (int i=0; i < 100 ; i++) {  
    System.out.println(numeros[i]);  
}
```

7



Longitud/tamaño de un array. length

- Para conocer la longitud/tamaño, es decir numero de elementos, de un array, utilizamos el atributo length de la clase Array.

```
int[] numeros = { 3, 5, 7, 9, 11, 13, 15, -1, -3 };  
System.out.println("Tamaño del vector" + numeros.length);
```

- **Recorrido** de un array utilizando length

```
int i;  
int[] alumnos = new int [10];  
  
for (i=0; i < alumnos.length; i++) {  
    System.out.print( alumnos[i] + " " );  
}
```

Obtiene el número total de elementos

8



Bucle for - each

- La sintaxis siguiente facilita el recorrido de todos los elementos de un array.

```
for (tipo_datos variable: nombre_array) {  
    sentencias;  
}
```

- *Tipo_datos* especifica el tipo de datos de los elementos del vector
- *variable* es el nombre de la variable que recibirá cada uno de los elementos del array. Debe ser del mismo tipo que los elementos del vector.

Ejemplo:

```
String[] ciudades = {"Madrid", "Barcelona", "Valencia"};  
  
for (String vciudad: ciudades)  
    System.out.println(vciudad);
```

9



Algunos métodos de la clase Arrays

- La clase *Arrays* ofrece algunos métodos que nos ayudan a gestionar vectores y sus elementos.
- Estos métodos son estáticos, por tanto accedemos a ellos a través del nombre de la clase, sin necesidad de crear un objeto.

Arrays.sort(miVector);



Algunos métodos de la clase Arrays

Métodos estáticos	Descripción y uso
boolean equals (int[] a, int[] a2)	Devuelve true si los vectores a y a2 son del mismo tamaño y tienen los mismos elementos en la misma posición. iguales = Arrays.equals(v1, v2);
void fill (int[] a, int val)	Rellena el vector a con el valor val Arrays.fill(v1, 5);
void sort (int[] a)	Ordena los elementos del vector a Arrays.sort(v1);
int binarySearch (int[] a, int key)	Busca el elemento key en el vector a y devuelve la posición que ocupa, si no esta devuelve valor negativo. El vector debe estar ordenado con sort() pos = Arrays.binarySearch(v1, 99);
String toString (int[] a)	Devuelve el contenido del array a como String, entre [] valores separados por , ...println(Arrays.toString(v1));



Copia de arrays (I)

```
// ejemplo de inicialización de vector con valores
int[] vector = {5,6,7};

// copia de arrays, en vector2 queremos una copia de vector
int[] vector2 = vector;

vector2[1]=30; //se modifica un valor de vector2

System.out.println("El valor 1 de vector: " + vector[1]);
System.out.println("El valor 1 de vector2: " + vector2[1]);
```

¿El resultado es el esperado?

¿Qué ha ocurrido con vector2=vector?



Copia de arrays (II)

Forma correcta de copiar un array a otro

```
// ejemplo de inicialización de vector con valores
int[] vector = {5, 6, 7};
int[] vector2 = new int[3];

// copia de arrays
System.arraycopy(vectorOrigen, posicionOrigen,
vectorDestino, posicionDestino,
numElementosACopiar);

System.arraycopy(vector, 0, vector2, 0, vector.length);

vector2[1]=30;
System.out.println("El valor 1 de vector es: " + vector[1]);
System.out.println("El valor 1 de vector2 es: " +
vector2[1]);
```

13



ARRAYS MULTIDIMENSIONALES

Arrays de más de una dimensión

14



Arrays de dos dimensiones (matriz)

- Un **array** de dos dimensión es una matriz de variables relacionadas del mismo tipos de datos. La posición de cada elemento viene determinada por dos índices (*fila* y *columna*)
- Declaración de un array de dos dimensiones

tipo[][] nombre_array = new tipo[num_filas][num_columnas]

- El número total de elementos del array es *num_filas x num_columnas*

Ejemplo: **int[][] pares = new int[3][4]; //3x4=12 elementos**



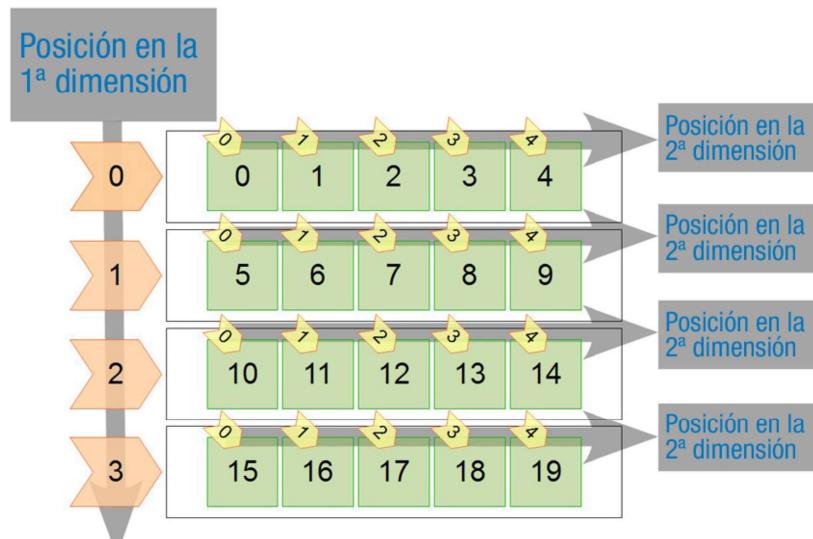
15



Arrays de dos dimensiones (matriz)

- También se puede definir como un **array de arrays**.
- En el ejemplo:
 - primera dimensión sería nº de filas (nº de arrays) y
 - segunda dimensión sería nº de elementos de cada array

int[][] matriz = new int[4][5];



16



Arrays de dos dimensiones

□ Inicialización:

```
tipo_datos[][] nombre = {
    filas   ↗ { val1, val2, ..., valN } ,
    ↗ { val1, val2, ..., valN } ,
    ...
};
```

Cada bloque de valores entre {} inicializa una fila

□ Ejemplo de inicialización:

```
int[][] pares = {{2,4,6,8},{10,12,14,16},{18,20,22,24}}
```

	0	1	2	3
0	2	4	6	8
1	10	12	14	16
2	18	20	22	24

índice fila → 1 índice columna → 2
 pares[1][2]

17



Acceso a los elementos de un array

□ Recorrer y llenar una matriz

```
int fila, columna, valor;
int [][] pares = new int [3] [4];

for(fila=0, valor=2; fila < 3 ; fila++) { //filas
    for(columna=0;columna<4;columna++, valor+=2) { //columna
        pares[fila] [columna] = valor;
    }
}
```

	0	1	2	3
0	2	4	6	8
1	10	12	14	16
2	18	20	22	24

índice fila 1 índice columna 2
 pares[1][2]

18



Acceso a los elementos de un array

□ Recorrer y mostrar una matriz

```
int fila, columna;  
int [ ] [ ] pares = {{2,4,6,8},{10,12,14,16},{18,20,22,24}};
```

```
System.out.println("Los 12 primeros números pares son:" );  
for (fila=0; fila < 3 ; fila++) {  
    for (columna=0; columna < 4; columna++) {  
        System.out.printf("%3d ", pares[fila][columna]);  
    }  
    System.out.println();  
}
```

	0	1	2	3
0	2	4	6	8
1	10	12	14	16
2	18	20	22	24

índice fila

índice columna

pares[1][2]

19



Longitud de un array. length

□ Recorrido de una matriz utilizando length

```
int fila, columna;  
int[][] matriz = new int[10][20];  
  
for (fila=0; fila < matriz.length ; fila++) {  
    for (columna =0; columna < matriz[fila].length; columna++) {  
        System.out.printf("%3d", matriz[fila][columna]);  
    }  
    System.out.println();  
}
```

Obtiene el número total de filas

Obtiene el número de elementos del array de la posición fila

20

Bucle for – each en matriz

- Recorrido de una matriz utilizando bucle for-each

```
int[][] numeros = {{1,2}, {6,7}};
```

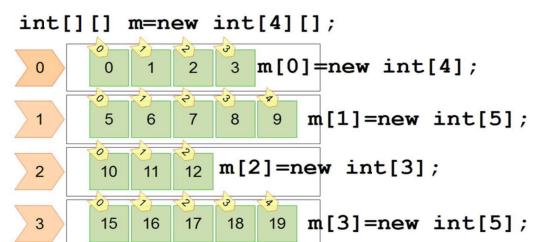
Cada elemento de la matriz *numeros*
es un vector de int

```
for (int[] vfila: numeros)
    for(int elemento: vfila)
        System.out.println(elemento);
```

21

Arrays multidimensionales irregulares

- Java permite crear un array de dos dimensiones, donde cada fila tiene un número diferente de elementos



1. **Declaramos y creamos el array pero sin especificar la segunda dimensión.** Lo que estamos haciendo en realidad es crear simplemente un array que contendrá arrays, sin decir como son de grandes los arrays de la siguiente dimensión:

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

2. **Después creamos cada uno de los arrays unidimensionales** (del tamaño que queramos) y lo asignamos a la posición correspondiente del array anterior:

```
irregular[0] = new int[7];
irregular[1] = new int[15];
irregular[2] = new int[9];
```

22