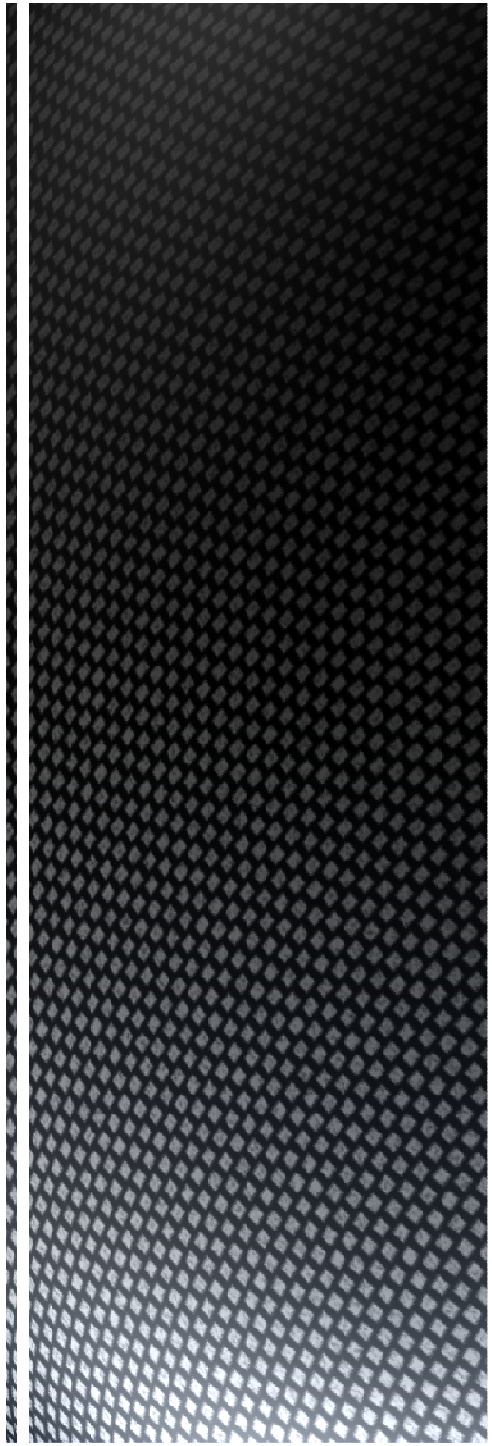
Arrays

Unidad 6



Contenidos

- Definición
- Arrays unidimensionales
- Arrays de varias dimensiones
- Ordenación y búsqueda en arrays
- Otras funciones útiles de arrays

Arrays

Definición

Los arrays son estructuras de datos que permiten almacenar un conjunto de valores del mismo tipo.

Pueden ser:

- Arrays unidimensionales (vectores)

valor	**5**	**9**	**6**	**-4**	**0**	**7**
índice	0	1	2	3	4	5

- Arrays multidimensionales (matrices)

		columnas			
		0	1	2	3
filas	0	**8**	**4**	**2**	**-6**
	1	**-2**	**6**	**5**	**12**
	2	**3**	**1**	**0**	**7**

Arrays unidimensionales (vectores)

Definición

Un **array** (o **vector**) es un conjunto finito de valores homogéneos (del mismo tipo) y ordenado, es decir, cada valor está identificado por una posición concreta del vector.

- El **índice** va de 0 a n-1 (**n:** cantidad total de elementos del array)
- **Declaración en java:**

```
int[] vector1;
```

- **[]:** identifican que el array es de una dimensión.

Arrays unidimensionales (vectores)

Inicialización

Inicializar el array: indicar cuantos elementos va a tener (se reserva el espacio necesario en memoria para albergar los elementos del vector).

```
int[] vector1;
vector1 = new int[3];

int[] vector2 = {7,42,10};
```

Arrays unidimensionales (vectores)

Asignación de un nuevo valor en una posición

Para asignar un nuevo valor a una posición del array, por ejemplo al índice 2 sería:

```
vector1[2] = 12;
System.out.println("Vector1[2] = " + vector1[2]);
```

Salida por pantalla:

```
Vector1[2] = 12
```

Arrays unidimensionales (vectores)

Recorrer un array

```java
// Declaramos el primer array
int[] vector1;
vector1 = new int[3];

// Declaramos el segundo array
int[] vector2 = {7,42,10};

// Damos valores al primer array
vector1[0] = 4;
vector1[1] = 8;
vector1[2] = 12;

// Imprimimos el primer array
System.out.println("vector1:");
for(int i=0; i<3; i++){
    System.out.println(vector1[i]);
}

// Imprimimos el segundo array
System.out.println("\nvector2:");
for(int i=0; i<vector2.length; i++){
    System.out.println(vector2[i]);
}
```

```
vector1:
4
8
12

vector2:
7
42
10
```

Arrays de varias dimensiones (matrices)

Definición

Se trata de arrays que a su vez contienen otros arrays.

- El **índice** de cada dimensión va de 0 a n-1.
- **Declaración en java:**

```
int[][] matriz;
```

- **[][]:** identifican que el array es de dos dimensiones.

Arrays de varias dimensiones (matrices)

Inicialización

La **instanciación** y **acceso** es similar al caso de los arrays unidimensionales.

```
int[][] matriz1;
matriz1 = new int[2][3];

matriz1[1][2] = 34;

int[][] matriz2 = {{3,2,0},{12,7,2}};
```

Arrays de varias dimensiones (matrices)

Recorrer un array bidimensional

```
public class Matriz {
    public static void main(String args[]) {
        // Declaramos la matriz
        int[][] matriz = {{3,2,0},{12,7,2}};

        // Imprimimos la matriz
        System.out.println("Matriz:");
        for(int i=0; i<2; i++) // filas
            for(int j=0; j<3; j++) // columnas
                System.out.println(matriz[i][j]);
    }
}
```

```
Matriz:
3
2
0
12
7
2
```

Ordenación y búsqueda en arrays

Definición

- La **ordenación** es la operación de organizar un conjunto de datos en algún orden dado (normalmente en datos numéricos se usa una ordenación creciente o decreciente y para cadenas en orden alfabético).
  - Método de la **burbuja**
  - Método de **selección**
  - Método de **inserción**.
- La **búsqueda** es la recuperación de información para extraer cierta información.
  - Búsqueda **lineal**
  - Búsqueda **binaria**.

Ordenación y búsqueda en arrays

Ordenación. Método de la burbuja

- El **método de la burbuja** se basa en comparar pares de elementos adyacentes e intercambiarlos entre sí hasta que estén todos ordenados.

```
(N es el tamaño del vector elementos)
desde i ← 0 hasta N-2 entonces
        desde j ← 0 hasta N-i-3 entonces
                si elementos[j+1] < elementos[j] entonces
                        aux ← elementos[j+1]
                        elementos[j+1] ← elementos[j]
                        elmentos[j] ← aux
                fin_si
        fin_desde
fin_desde
```

Ordenación y búsqueda en arrays

Ordenación. Método de inserción

- El **método de inserción** consiste en insertar un elemento en el vector, en una parte ya ordenada de este vector, y comenzar de nuevo con los elementos restantes, por tanto, son necesarios una serie de comparaciones y desplazamientos sucesivos.

```
(N es el tamaño del vector elementos)
desde i ← 0 hasta N-1 entonces
        aux ← elementos[i]
        j ← i-1
        mientras j >= 0 y elementos[j] > aux hacer
                elementos[j+1] ← elementos[j]
                elmentos[j] ← aux
                j--
        fin_mientras
fin_desde
```

Ordenación y búsqueda en arrays

Ordenación. Método de selección

- El **método de selección** se basa en la búsqueda del elemento menor del vector y colocarlo en primera posición. Posteriormente se buscará el segundo elemento más pequeño y se colocará en la posición segunda y así sucesivamente.

Ordenación y búsqueda en arrays

Ordenación. Método de selección

```
(N es el tamaño del vector elementos)
desde i ← 0 hasta N-2 entonces
        menor ← elementos[i]
        desde j ← i+1 hasta N-1 entonces
                si elementos[j] < menor
                        menor ← elementos[j]
                        posicion ← j
                fin_si
        fin_desde
        si posicion <> i entonces
                aux ← elementos[i]
                elementos[i] ← elementos[posicion]
                elementos[posicion] ← aux
        fin_si
fin_desde
```

Ordenación y búsqueda en arrays

Búsqueda lineal

- La **búsqueda lineal** consiste en recorrer un vector comparando el contenido de cada celda con el dato del que queremos conocer la información.

Ordenación y búsqueda en arrays

Búsqueda lineal

(N es el tamaño del vector elementos y
elemento_buscado el elemento que queremos saber si está)

```
pos ← 0
encontrado ← falso
mientras pos < N y encontrado == falso entonces
        si elemento_buscado == elementos[i] entonces
                encontrado ← verdadero
        sino
                pos  ← pos + 1
        fin_si
fin_mientras
si encontrado == verdadero entonces
        escribir('Elemento encontrado en la posición pos')
sino
        escribir('Elemento no encontrado')
fin_si
```

Ordenación y búsqueda en arrays

Búsqueda binaria

- La **búsqueda binaria** utiliza un método ***divide y vencerás*** para dar con el valor deseado. En primer lugar se examina el elemento central del vector. Si es el elemento buscado entonces el método finaliza. En caso contrario se determina si el elemento buscado está en la primera o en la segunda parte del array y se vuelve a repetir el proceso. Para el uso de este método de búsqueda **es imprescindible que el vector se encuentre ordenado**.

Ordenación y búsqueda en arrays

Búsqueda binaria

```
(N es el tamaño del vector elementos y
 elemento_buscado el elemento que queremos saber si está)
inicio ← 0
fin ← N-1
encontrado ← falso
mientras inicio <= fin entonces
        pos ← (inicio + fin)/2;
        si elemento_buscado == elementos[pos] entonces
                devolver pos
        si elemento_buscado > elementos[pos]
                inicio ← pos + 1
        sino
                fin ← pos - 1
        fin_si
fin_mientras
devolver -1
```

Funciones útiles de arrays

NOMBRE	DESCRIPCIÓN	PARÁMETROS	DATO DEVUELTO
copyOf	Copia un array y lo devuelve en un nuevo array.	***Un array y la longitud.*** Si se pasa del tamaño del array original, rellena con ceros las posiciones sobrantes. Estos pueden ser un byte, char, double, float, int, long, short u objecto.	array del mismo tipo que se introduce
copyOfRange	Copia un array y lo devuelve en un nuevo array. Le indicamos la posición de origen y de destino.	***Un array, posición origen y destino.*** Estos pueden ser un byte, char, double, float, int, long, short u objecto.	array del mismo tipo que se introduce
equals	Indica si dos arrays son iguales.	***Dos arrays del mismo tipo.***	true o false
fill	Rellena un array con un valor que le indiquemos como parámetro.	***Un array y el valor a rellenar.*** Estos pueden ser un byte, char, double, float, int, long, short u objecto.	No devuelve nada