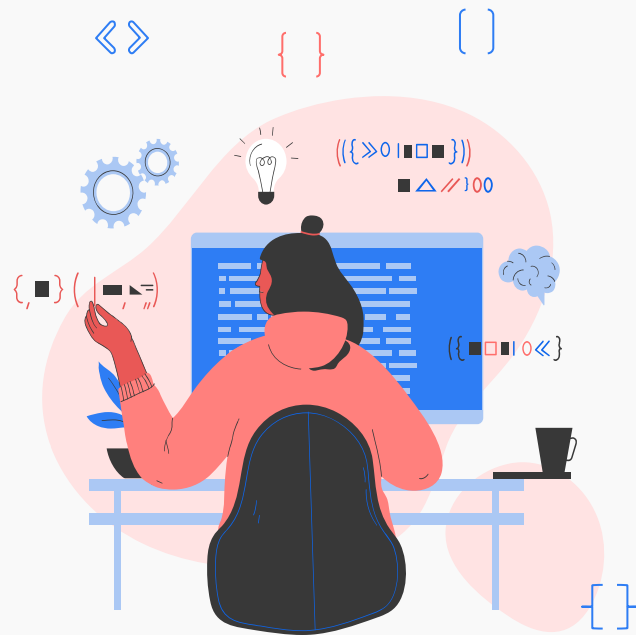


Programación

Tema 3 - Estructuras complejas de datos

Marina Hurtado Rosales
marina.hurtado@escuelaartegranada.com





Índice



1. Arrays
2. Arrays y bucles
3. Matrices
4. Listas



Arrays

Arrays

En Java existen unas estructuras concretas que nos permiten almacenar múltiples valores en una sola variable.

Almacenan valores del mismo tipo de dato.

Tienen un tamaño fijo, no pueden ni crecer ni reducirse.

Se almacenan en posiciones consecutivas de memoria.

	0	1	2	3	4	5
Str	G	e	e	k	s	\0
Address	0x23452	0x23453	0x23454	0x23455	0x23456	0x23457

Arrays

Los arrays se declaran indicando el tipo de dato que van a almacenar junto con corchetes [].

En cuanto al nombre, se sigue la misma convención que con las variables

```
String[] cadenaStrings;
```

Inicialización de arrays

Para inicializar un array hay que tener en cuenta que estos tienen un tamaño fijo, por lo que una vez inicializados no se pueden cambiar de tamaño.

Una manera de evitar quedarnos cortos en el tamaño del array es **sobredimensionarlo**.

[]

Existen dos formas distintas de inicializar un array:

- **Inicialización directa.** Se usa una lista con los elementos **separados por coma**, y acotada por llaves {}

```
int[] edades = {18, 20, 22, 19};
```

- **Inicialización con tamaño**

```
String[] nombres = new String[3];
```

[]

Acceso a elementos de un array

Para acceder a valores concretos dentro de un array se utiliza su posición dentro de la lista, es decir, **su índice**.

En Java, los índices empiezan **siempre desde 0**.

```
String[] arrayCadenas = {"Hello", "world", "!"};  
int[] numeros = {1, 45, 32, 5, -15};  
System.out.println(arrayCadenas[0] + " " + numeros[2]);  
// Imprime Hello 32
```

Acceso a elementos de un array

De igual forma, podemos utilizar esta manera de acceder a las posiciones concretas para **modificar o añadir valores** a la lista.

```
int[] numeros = {1, 45, 32, 5, -15};  
numeros[2] = -52;  
System.out.println(numeros[2]);  
//Imprime -52 en lugar de 32
```

Si hemos inicializado un array con un tamaño, pero no le hemos asignado valores, podemos usar esto para añadirle elementos.

```
String[] nombres = new String[3];  
nombres[0] = "Ana";  
nombres[1] = "Luis";  
nombres[2] = "Sofia";
```

Arrays y bucles

Recorrer un array

Podemos **recorrer** los elementos de un array mediante el uso de **los bucles for**.

Se especifica cuántas veces se hace el bucle usando el atributo **length** de los array

```
int[] numeros = {1, 45, 32, 5, -15};  
for (int i = 0; i < numeros.length; i++){  
    System.out.println(numeros[i]);  
    // Imprime el elemento i del array  
}
```

Ejemplo

Podemos calcular la media de un array de enteros usando un bucle for

```
int[] notas = {5, 7, 9, 6};
int suma = 0;

for (int n : notas) {
    suma += n;
}

double media = (double) suma / notas.length;
System.out.println("Media: " + media);
```

Matrices

Matrices

Las matrices también se llaman **arrays multidimensionales**. Se definen como arrays de arrays.

Este tipo de estructuras son útiles cuando se quieren guardar datos en forma de tabla. Por ejemplo, si quisiéramos guardar el registro de temperaturas a lo largo de una semana.

Al igual que con los arrays, existen dos formas de inicializar las matrices:

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

```
int [][] tablero = {  
    {1,0,1},  
    {0,1,0},  
    {1,0,1}  
};
```

Acceso a elementos de matrices

Para acceder a los datos de la matriz, se necesitan **2 índices**. Uno para seleccionar el sub-array y el otro para seleccionar el dato en sí.

```
int [][] tablero = {{1,2,3}, {4,5,6}};  
System.out.println(tablero[1][0]);  
// Saca por pantalla el valor 4  
  
tablero[0][2] = 7;  
// Sustituye el 3 por 7
```

Recorrer una matriz

Para recorrer una matriz, necesitamos usar **2 bucles for anidados** ya que tenemos que llevar la cuenta de 2 índices a la vez.

Nota: `matriz.length` devuelve cuántos sub-arrays hay incluidos en la matriz, mientras que `matriz[i].length` devuelve los espacios del sub-array i.

```
int [][] tablero = {{1,2,3}, {4,5,6}};
for(int i = 0; i < tablero.length; i++){
    for(int j = 0; j < tablero[i].length; j++){
        System.out.println(tablero[i][j]);
    }
}
```