

U6. Estructuras de almacenamiento.

Objetivos:

- Crear y gestionar arrays unidimensionales y multidimensionales: Declarar, inicializar, recorrer, buscar y ordenar.
- Profundizar en un array específico de cadena de caracteres: String.

1. Arrays

1.1. Estructuras.

Una estructura de datos es una organización de estos en una única entidad contenedora. Se define así con la idea de agrupar toda la información en una sola entidad.

¿Qué tipos de datos puede agrupar?

- Elementales: int, char, float...
- Estructuras: Agrupar agrupaciones de datos.
- Objetos.

Nombre del arreglo (c) →	c[0]	-45
	c[1]	6
	c[2]	0
	c[3]	72
	c[4]	1543
	c[5]	-89
	c[6]	0
	c[7]	62
	c[8]	-3
	c[9]	1
Índice (o subíndice) del elemento en el arreglo c	c[10]	6453
	c[11]	78

Un arreglo con 12 elementos.

1.2 Arrays unidimensionales y multidimensionales.

- Array unidimensional. Una única dimensión. Vector.
- Array multidimensional. Dos o más dimensiones. Matriz.

1.3. Declaración.

Se nombra el tipo de los elementos del array que se está formando, incluyendo unos corchetes

1 DIMENSIÓN (N)

int[] vector;

int vector[];

2 DIMENSIONES (N x N)

int tabla[][];

int [][]tabla;

3 DIMENSIONES (N x N x N)

int tabla[][][];

int [][][]tabla;

1.4. Instanciación o creación del array unidireccional y multidimensional.

1.4.1. Creación de un array unidimensional (vector) de 10 elementos:

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

```
int vector[];
```

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



```
tipo_dato nombre_array[];  
nombre_array = new tipo_dato[tamano];
```

```
char arrayCaracteres[];  
arrayCaracteres = new char[10];
```

1.4.2. Creación de un array bidimensional (matriz)

```
int vector[][];
```

```
vector = new int[3][4];
```

```
int vector[] = new int [3][4];
```

	Columna 0	Columna 1	Columna 2	Columna 3
Fila 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Fila 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Fila 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

Subíndice de columna
Subíndice de fila
Nombre del arreglo

1.5. Inicialización.

Se puede indicar los valores que tendrá el array dentro de la propia declaración. De este modo el array asumirá el tamaño que le corresponde.

```
int[] vector = { 3, 5, 6 };
```

3	5	6
---	---	---

```
int[][] vector = { { 4, 7 }, { 2, 6 }, { 3, 4 } };
```

4	7
2	6
3	4

OJO: Las dimensiones no han de tener el mismo tamaño.

```
int[][] vector = { { 4, 7 }, { 3, 5, 6 }, { 2 } };
```

4	7	
3	5	6
2		

1.5. Acceso a los datos.

Indicando una posición determinada del array entre corchetes accedemos a un elemento concreto dentro de la agrupación correspondiente.

Obtener el valor y mostrarlo por pantalla (Sysout) :

```
System.out.println(vector[2]);
```

Modificar un elemento:

```
vector[2] = 7;
```

EJ) Alternativa a la inicialización del vector. Lo inicializamos e indicamos uno a uno los elementos que lo compone.

```
int vector[];
vector = new int[4];
vector[0] = 2;
vector[1] = 5;
vector[2] = 7;
vector[3] = 1;
```

Posición	0	1	2	3
Valor	2	5	7	1

1.6. Método Length.

Permite obtener el tamaño de un vector.

```
int vector[] = {1,2,3};
System.out.println(vector.length);
```

Obteniendo 3, el número de elementos del vector.

OJO1: Sabiendo que el vector tiene elementos la última posición es 2, porque las posiciones dentro del array van de 0 a length-1

OJO2: En el caso de disponer de un array de arrays debemos proceder del siguiente modo:

```
int vector[3][];  
vector[0] = new int[5];  
vector[1] = new int[7];  
vector[2] = new int[3];
```

```
System.out.print("Numero de filas: "+numeros.length);  
for (int i = 0; i < numeros.length; i++) {  
    System.out.print("Fila " + i + " contiene : " + numeros[i].length + " columnas");  
}
```

1.7. Recorrido: búsqueda y ordenación.

Como los arrays tienen una disposición conocida previamente, encaja muy bien una estructura de repetición acotada como **for** para recorrer dicho array.

```
int[] vector = { 3, 5, 6 };  
for (int i = 0; i < vector.length; i++)  
    System.out.println("elemento en " + i + ": " + vector[i]);
```

¿Búsqueda? Crea una función que dado un array de entrada y determinado valor a buscar muestre por pantalla en qué posiciones se muestra.

¿Ordenación? Crea una función que dado un array de entrada coloque todos los elementos agrupados de menor a mayor valor.

1.7.1 FOR MEJORADO.

Itera a través de los elementos de un arreglo o colección sin utilizar un contador

*for (parámetro : nombreArreglo)
instrucción*

```
int[] arr = {1,3,4};  
for (int element : arr) {  
    System.out.println(element);  
}
```

```
int[][] arr = {{1,2},{3,4}};  
for (int[] row : arr) {  
    for (int element : row) {  
        System.out.println(element);  
    }  
}
```

1.8. Copiar un array.

```
int[] arr_origen = {1,3,4};  
int[] arr_destino = new int[arr_origen.length];  
  
for (int i = 0; i < arr_origen.length; i++) {  
    arr_destino[i] = arr_origen[i];  
}
```

```
// Llamada a copia. 4 elementos desde el elemento 0 de origen  
// al elemento 0 en adelante del destino.  
System.arraycopy(b, 0, c, 0, 4);
```

1.9. Excepción.

```
int[] arr_origen = {1,3,4};  
System.out.println(arr_origen[6]);
```

Exception in thread "main" [java.lang.ArrayIndexOutOfBoundsException](#): Index 6 out of bounds for length 3

2. Cadena de caracteres (String).

Una cadena de caracteres se puede entender como una agrupación de variable del tipo char.

OJO: No confundir con un vector con elementos de tipo char. `vector char[]`

2.1. Declaración. `String str;`

2.2. Inicialización. `String str = new String ("Hola mundo");`
`String nombres[] = { "Juan","Luisa","Pedro","María" };`

2.3. Operaciones.

- **.length() : int**
Devuelve el tamaño de la cadena de caracteres.
`int x = str1.length(); // 6`
- **.charAt(int) : char**
Devuelve el carácter situado en la posición indicada.
`char c = str1.charAt(3); // t`
- **.equals(String) : boolean**
Compara dos cadenas de caracteres y son iguales devuelve "true".
`boolean b = str1.equals(str3); // false`
- **.indexOf(char) : int || .lastIndexOf(char) : int || .indexOf(String) : int**
Posición a partir del cual se encuentra el carácter o cadena o último carácter ("last")
`int n = str1.lastIndexOf('t'); // 3`
- **.substring(int, int) : String**
Devuelve la cadena comprendida en el rango de entrada
`String s = str1.substring(2,4); // "xt"`
- **.toLowerCase() : String && .toUpperCase() : String**
Devuelve el String convirtiendo todos sus elementos a mayúsculas o minúsculas.
`String s = str1.toUpperCase(); // "TEXT01"`
- **.replace(char objetivo, char nuevo) : String**
Devuelve la cadena donde devuelve todos los caracteres "objetivo" por el carácter "nuevo".
`String s = str1.replace('t','m'); // "mexmol"`
- **.valueOf(tipoBasico) : String**
Conversión a String de diferentes tipos de variables.
`String s = String.valueOf(15l); // 15`
- **.concat (String) : String || String + String || String + int**
Concatena cadenas de caracteres y números.
`String s = "Son las " + 18 + ":" + 30;`

EXTRA: Clase Arrays

La clase Arrays de java tiene múltiples métodos que nos ayudan a manejar arrays.

.toString()

Este método convierte el array en un String legible para un humano.

```
Boolean[] anArray = {Boolean.TRUE, Boolean.FALSE, Boolean.FALSE, Boolean.TRUE};  
System.out.println(Arrays.toString(anArray));
```

sort()

El método Arrays.sort() permite ordenar un array (default menor a mayor).

```
Double [] doubles = new Double[5];  
for (int i=0;i<doubles.length;i++){  
    doubles[i] = Math.random();  
}  
Arrays.sort(doubles);
```

copyOf() y copyOfRange()

Permite copiar el contenido de un array en otro.

```
double [] sourceArray = {1.1,2.2,3.3,4.4};  
double [] destinationArray = Arrays.copyOf(sourceArray, 5);  
System.out.println(Arrays.toString(destinationArray));
```

equals()

Arrays.equals() y Arrays.deepEquals() comparan si dos arrays son iguales.

```
String [] objectArray = {new String("uno"),new String("dos")};  
String [] anotherObjectArray = {new String("uno"),new String("dos")};  
System.out.println(Arrays.equals(objectArray, anotherObjectArray));
```

fill()

Este método es sencillo, permite rellenar un array con elementos iguales

```
Arrays.fill(array,"good bye");  
System.out.println(Arrays.toString(array));
```