

# Arreglos

Definición e implementación

Verónica E. Arriola-Rios

Facultad de Ciencias, UNAM

19 de noviembre de 2025



# Definición

1 Definición

2 Arreglos en Java

3 Ejercicios interesantes

4 Bibliografía

# Temas

## 1 Definición

- Definiciones formales

# Tipo abstracto de datos: arreglo

## Conjunto de datos:

Un *arreglo* es una estructura de datos que contiene un conjunto de **elementos del mismo tipo**, un conjunto de **índices** y un conjunto de operaciones que se utilizan para definir, manipular y abstraer estos elementos de datos.

Sengupta y Korobkin 2014

50	25	75	10	39	0	100	1	0	29	42	0	0	0	120
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

## Operaciones:

- Creación: dado su tamaño devuelve un arreglo.
- Asignación: dados los índices, guarda el valor en la posición correspondiente.
- Lectura: dados los índices, devuelve el valor almacenado en la posición indicada.

# Arreglo multidimensional

## Definición (Arreglo)

Un arreglo de

- ① dimensión  $n$ ,
- ② de elementos de tipo  $X$  y
- ③ de tamaño  $T = t_1 \times t_2 \times \dots \times t_n$ , donde  $t_j$  es el tamaño del arreglo en la  $j$ -ésima dimensión,

es un conjunto de  $T$  elementos de tipo  $X$ , en el que cada uno de ellos es identificado **únivamente** por un **vector coordenado de  $n$  índices**  $(i_1, i_2, \dots, i_n)$ , con  $0 \leq i_j < t_j$ .

	0	1	2	3	4
0	50	25	75	10	39
1	0	100	1	0	29
2	42	0	0	0	120

# Arreglos en Java

1 Definición

2 Arreglos en Java

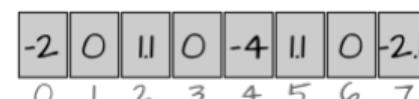
3 Ejercicios interesantes

4 Bibliografía

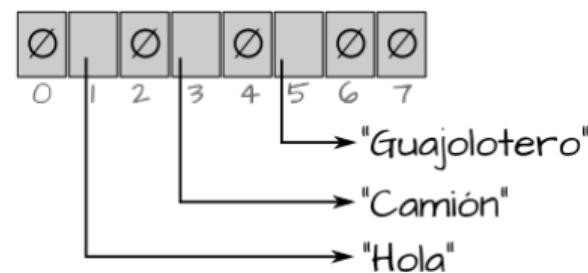
# Arreglos en Java

En Java se distinguen dos tipos de arreglos:

- Arreglos de tipos primitivos: contienen al dato en su interior.



- Arreglos de objetos: contienen las direcciones de los objetos en el heap.



# Temas

## 2 Arreglos en Java

- Arreglos 1D
- Arreglos nD

# Arreglos de primitivos

Se declaran e instancian con:

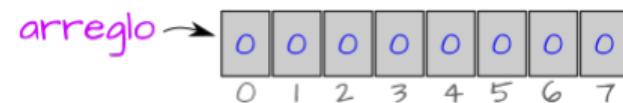
```
1 int [] arreglo;           // Vale null
2 int arreglo [];
3 arreglo = new int [8];   // Tamaño entre []
4
```

Ahora hay que asignar los valores:

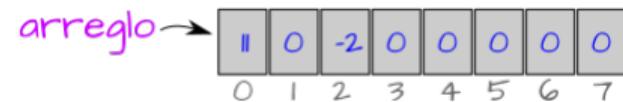
```
1 arreglo[0] = 11;
2 arreglo[2] = -2;
3
```

Creación rápida:

```
1 int arreglo [] = {11, -2};
2
```



Ojo: Los espacios se llenan con el valor por defecto.



# Arreglos de Objetos

Se declaran e instancian con:

```
1 String arreglo[];           // Vale null
2 arreglo = new String[8];   // Tamaño entre []
3
```

Ahora hay que crear los objetos y asignarlos:

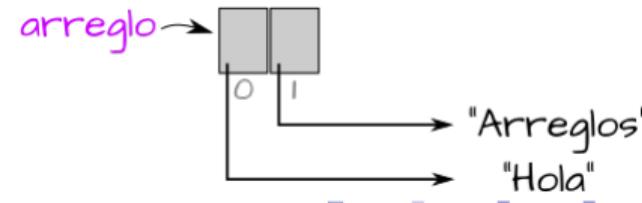
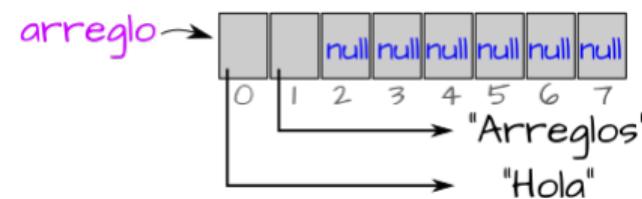
```
1 arreglo[0] = "Hola";
2 arreglo[1] = new String("arreglos");
3
```

Creación rápida:

```
1 String arreglo[] = {"Hola", "arreglos"};
2
```



**Ojo:** Sólo se reservan espacios para referencias a cadenas.



# Ejemplo de uso

```
1 package demos;  
2  
3 public class PrintCadenas{  
4     public static void main(String args[]){  
5  
6         String cadenas [] = {"Hola", "Arreglos"};  
7  
8         for(int i = 0; i < cadenas.length; i++){  
9             System.out.println(cadenas[i]);  
10        }  
11    }  
12 }
```

```
1 $ java demos.PrintCadenas  
2 Hola  
3 Arreglos
```

# Temas

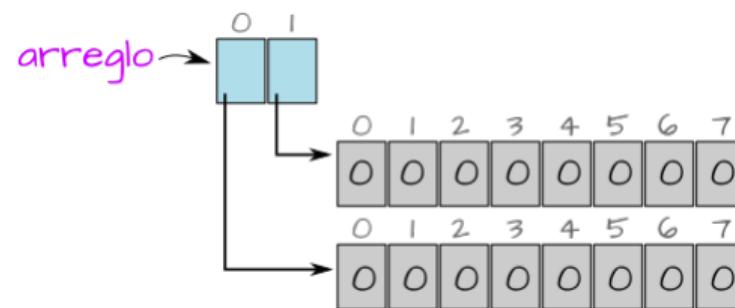
## 2 Arreglos en Java

- Arreglos 1D
- Arreglos nD

# Arreglos 2D

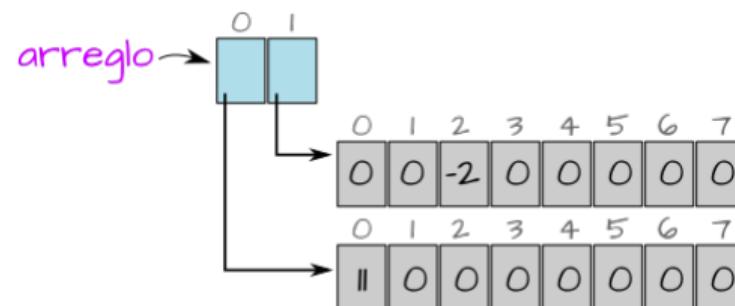
Se declaran e instancian con:

```
1 int [][] arreglo0;
2 int arreglo [] [];
3 arreglo = new int [2] [8];
4
```



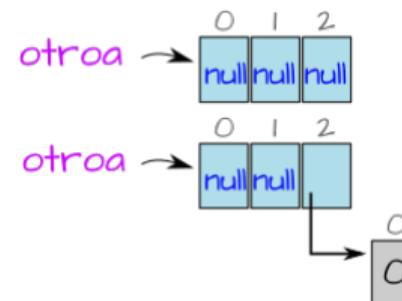
Ahora hay que asignar los valores:

```
1 arreglo [0] [0] = 11;
2 arreglo [1] [2] = -2;
3
```



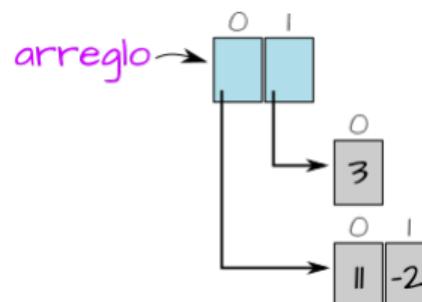
También se pueden crear renglón por renglón:

```
1 int [][] otroa = new int [3] [];
2 otroa[2] = new int [1];
3
```



Creación rápida:

```
1 int arreglo [][] = {{11, -2}, {3}};
2
```



# Excepciones

- Si se intenta asignar un valor a una posición que no existe, la ejecución del código se interrumpirá con una `ArrayIndexOutOfBoundsException`.

Código: `ArrayIndexOutOfBoundsException`

```
1 int [] a = new int [3];
2 a[0] = 3;                                // Correcto
3 a[3] = 8;                                // Incorrecto
4 System.out.println(a[4]);      // Incorrecto
```

- Si se intenta llamar una posición en un arreglo de más de 1D, y ese subarreglo no ha sido creado, obtendremos `NullPointerException`.

Código: `NullPointerException`

```
1 int [][] a = new int [3] [];
2 a[0] = new int [2];
3 a[0][0] = 3;                                // Correcto
4 a[1][0] = 8;                                // Incorrecto
```

# Arreglos nD

- El mecanismo de *arreglos de arreglos* que utiliza Java conocido como [Vectores de Iliffe](#), permite crear fácilmente arreglos nD.
- En la sintaxis basta con agregar un par de [] por cada dimensión, tanto al declarar como al usar alguna posición.

Código: NullPointerException

---

```
1 int [][] [] [] a = new int [3] [8] [3] [2]; // Crear
2 a[2][7][0][1] = 7; // Asignar
3 int siete = a[2][7][0][1]; // Leer
```

---

# Ejercicios interesantes

- 1 Definición
- 2 Arreglos en Java
- 3 Ejercicios interesantes
- 4 Bibliografía

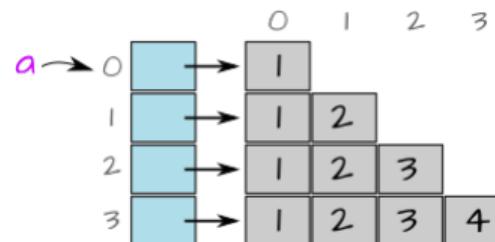
# Temas

## 3 Ejercicios interesantes

- Pirámides
- Ordenamientos
- Búsqueda binaria

# Base de la pirámide

```
1 public static int[][] base(int n) {  
2     int[][] a = new int[n][];  
3     for(int i = 0; i<n; i++) {  
4         a[i] = new int[i+1];  
5         for(int j = 0; j <= i; j++) {  
6             a[i][j] = j+1;  
7         }  
8     }  
9     return a;  
10 }
```

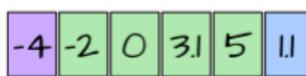
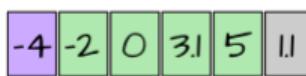
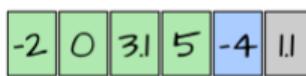
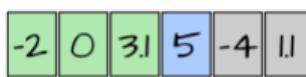
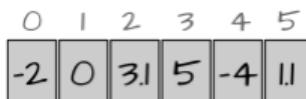


# Temas

## 3 Ejercicios interesantes

- Pirámides
- Ordenamientos
- Búsqueda binaria

# Ordenamiento por inserción



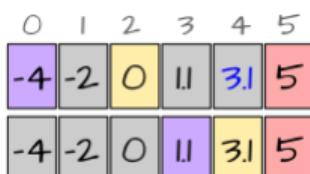
```
1 public static void ordenaPorInsercion(double[] a) {  
2     for(int i=1; i<a.length; i++) {  
3         for(int j=i; j>1; j--) {  
4             if(a[j] < a[j-1]) {  
5                 // Intercambia los valores  
6                 double temp = a[j];  
7                 a[j] = a[j-1];  
8                 a[j-1] = temp;  
9             } else break;  
10        }  
11    }  
12 }
```

# Temas

## 3 Ejercicios interesantes

- Pirámides
- Ordenamientos
- Búsqueda binaria

# Búsqueda binaria



```
1  /** Busca el elemento en un arreglo ordenado.  
2   * @returns el índice de la primer aparición o -1  
3   *         si no está. */  
4  public static int busca(double[] a, double x) {  
5      int izq = 0, der = arr.length - 1;  
6      while (izq <= der) {  
7          int m = (izq + der) / 2;  
8  
9          if (arr[m] == x) return m; // Aquí está  
10         if (arr[m] > x) der = m - 1; // Busca a la izquierda  
11         else izq = m + 1; // Busca a la derecha  
12     }  
13     return -1; // No estuvo  
14 }
```

# Bibliografía

1 Definición

2 Arreglos en Java

3 Ejercicios interesantes

4 Bibliografía

# Bibliografía I



Sengupta, Saumyendra y Carl Philip Korobkin (15 de feb. de 2014). *C++ Object-Oriented Data Structures*.

# Licencia

Creative Commons  
Atribución-No Comercial-Compartir Igual

