**Arrays (Vectores) en Java**

**¿Qué es un Array?**

Un **array** (o vector) es una **estructura de datos** que almacena una **colección ordenada de elementos del mismo tipo** bajo un **único nombre**, accesibles mediante un **índice numéri**co

**2. Características de los Arrays**

✅ **Tamaño fijo**: Una vez definido, no puede crecer ni reducirse.  
✅ **Almacena elementos del mismo tipo**: Todos los elementos deben ser del tipo declarado (int, String, etc.).  
✅ **Acceso rápido por índice**: O(1) para lectura/escritura.  
✅ **Ordenados**: Mantienen el orden de inserción.  
❌ **No dinámicos**: No pueden cambiar de tamaño después de crearse.

**3. Declaración e Inicialización**

**Declaración**

tipo[] nombreArray; // Sintaxis recomendada

tipo nombreArray[]; // Sintaxis alternativa (menos usada)

**Inicialización**

nombreArray = new tipo[tamaño];

**Inicialización directa (con valores)**

tipo[] nombreArray = {valor1, valor2, valor3};

String[] colores = {"Rojo", "Verde", "Azul"};

**4. Acceso a Elementos**

* Se accede con nombreArray[indice] (índices comienzan en **0**).
* Si se intenta acceder a un índice fuera de los límites (< 0 o >= array.length), lanza ArrayIndexOutOfBoundsException.

**Ejemplo:**

int[] edades = {25, 30, 18};

System.out.println(edades[0]); // 25 (primer elemento)

edades[1] = 35; // Modifica el segundo elemento

System.out.println(edades[1]); // 35

**5. Atributo**length

* Devuelve el **tamaño del array** (número de elementos).
* Es un **atributo**, no un método → No lleva paréntesis ().

**Ejemplo:**

int[] numeros = {10, 20, 30, 40};

System.out.println(numeros.length); // 4

**6. Recorrido de un Vector**

**Con**for**tradicional (usando índices)**

for (int i = 0; i < numeros.length; i++) {

System.out.println(numeros[i]);

}

**Con**for-each**(Java 5+)**

for (int num : numeros) {

System.out.println(num);

}

**Ventaja:** Más legible, no requiere manejo de índices.  
**Desventaja:** No permite modificar el array (solo lectura).

**7. Algoritmos de Ordenación**

**1. Burbuja (Bubble Sort)**

Compara pares de elementos y los intercambia si están en desorden.

for (int i = 0; i < numeros.length - 1; i++) {

for (int j = 0; j < numeros.length - i - 1; j++) {

if (numeros[j] > numeros[j + 1]) {

int temp = numeros[j];

numeros[j] = numeros[j + 1];

numeros[j + 1] = temp;

}

}

}

**2. Selección (Selection Sort)**

Busca el mínimo y lo coloca en su posición correcta.

for (int i = 0; i < numeros.length - 1; i++) {

int minIndex = i;

for (int j = i + 1; j < numeros.length; j++) {

if (numeros[j] < numeros[minIndex]) {

minIndex = j;

}

}

int temp = numeros[minIndex];

numeros[minIndex] = numeros[i];

numeros[i] = temp;

}

**3. Inserción (Insertion Sort)**

Toma un elemento y lo inserta en su posición correcta.

for (int i = 1; i < numeros.length; i++) {

int key = numeros[i];

int j = i - 1;

while (j >= 0 && numeros[j] > key) {

numeros[j + 1] = numeros[j];

j--;

}

numeros[j + 1] = key;

}

**4.**Arrays.sort()**(Método de Java)**

Ordena arrays de forma optimizada (usa **QuickSort** o **MergeSort**).

import java.util.Arrays;

Arrays.sort(numeros);

**8. Algoritmos de Búsqueda**

**1. Búsqueda Lineal (Secuencial)**

Recorre el array hasta encontrar el elemento.

public static int busquedaLineal(int[] arr, int valor) {

for (int i = 0; i < arr.length; i++) {

if (arr[i] == valor) {

return i; // Devuelve el índice

}

}

return -1; // No encontrado

}

**2. Búsqueda Binaria (Requiere array ordenado)**

Divide el array en mitades para reducir la búsqueda.

import java.util.Arrays;

public static int busquedaBinaria(int[] arr, int valor) {

Arrays.sort(arr); // Asegura que esté ordenado

int left = 0, right = arr.length - 1;

while (left <= right) {

int mid = left + (right - left) / 2;

if (arr[mid] == valor) {

return mid;

} else if (arr[mid] < valor) {

left = mid + 1;

} else {

right = mid - 1;

}

}

return -1; // No encontrado

}

**Método de Java (**Arrays.binarySearch()**)**

int pos = Arrays.binarySearch(numeros, valor);

// Si `pos >= 0`, el elemento existe en ese índice.

// Si `pos < 0`, el elemento no está en el array.

**Resumen Final**

| **Concepto** | **Ejemplo** |
| --- | --- |
| Declaración | int[] numeros; |
| Inicialización | numeros = new int[5]; |
| Acceso | numeros[0] = 10; |
| Longitud | numeros.length |
| Recorrido (for) | for (int i=0; i<numeros.length; i++) |
| Recorrido (for-each) | for (int num : numeros) |
| Ordenar | Arrays.sort(numeros); |
| Búsqueda binaria | Arrays.binarySearch(numeros, valor) |