INVESTIGACION DE ARREGLOS EN JAVA

Como declarar un arreglo en java

El arreglo en java es una estructura de datos que nos permite almacenar varios valores del mismo tipo en una sola variable. Siendo útil cuando necesitamos guardar múltiples datos de que se relacionan entre si, como una lista de números, nombres o calificaciones.

Ejemplos

1. Asignación de valores manualmente

```
numeros[0] = 10;
numeros[1] = 20;
numeros[2] = 30;
numeros[3] = 40;
numeros[4] = 50;
```

2. inicialización directa

```
int[] valores = {1, 2, 3, 4, 5};
```

Métodos y utilidades principales para arreglos

- **Arrays.sort**

Ordena los elementos del arreglo de forma ascendente .

- **Arrays.binarySearch()**

Busca un elemento dentro del arreglo y devuelve su índice. El arreglo debe estar ordenado antes de usarlo.

- **Arrays.copyOf()**

Copia un arreglo y permite definir el nuevo tamaño.

- **Arrays.fill()**

Llena todas las posiciones del arreglo con un mismo valor.

- **Arrays.equals()**

Compara dos arreglos y devuelve true si son iguales, o false si son diferentes.

Ejemplos en el proyecto

¿Cómo se recorren los arreglos en Java?

For tradicional (con índices)
Se puede utilizar una variable contador para recorrer las posiciones del arreglo.

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


for (int i = 0; i < valores.length; i++) {
   System.out.println(valores[i]);
}
```

Que nos permite usar el índice para modificar posiciones específicas.

For-each (para cada elemento)
Este permite recorre directamente los valores sin usar índices.

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

for (int numero : valores) {
   System.out.println(numero);
}
```

La ventaja de este es que es mas simple de y fácil de leer. Su desventaja es que no puedes acceder al índice directamente.

Streams (Arrays.stream().forEach())
Esta es la forma mas moderna la cual esta desde el java 8 en adelante

```
import java.util.Arrays;

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

Arrays.stream(valores).forEach(n -> {
   System.out.println(n);
});
```

Este permite usar programación funcional de manera filtrada, transformar y muchas mas.

Diferencias entre arreglos y ArrayList en Java

(Arrays)
Tamaño fijo vs tamaño dinámico:
Tiene un tamaño fijo
Cuando declaras un arreglo debes indicar cuántas posiciones tendrá.
Después de creado, no se puede cambiar su tamaño.

int[] numeros = new int[3];

Tipos primitivos vs clases envolventes
Pueden almacenar tipos primitivos directamente como int, double, char, etc.
También pueden almacenar objetos.

int[] edades = {15, 16, 17};

Métodos disponibles
No tienen muchos métodos propios.
Para ordenarlos o buscarlos se usa la clase Arrays.
Se accede a los elementos con índice

numeros.length
Arrays.sort(numeros);}

Rendimiento
Son más rápidos porque trabajan directamente con memoria.
Consumen menos recursos.
Son ideales cuando el tamaño es fijo y se necesita eficiencia.

¿Cuándo usar cada uno?
Usa Arreglos cuando:
Sabes que el tamaño no cambiará.
Necesitas mayor rendimiento.
Trabajas con muchos datos numéricos.
El programa es más simple.

ArrayList

Tamaño fijo vs tamaño dinámico:

Tiene tamaño dinámico.

No necesitas definir el tamaño al inicio.

Puede crecer o disminuir automáticamente según los elementos que agregues o elimines.

```
import java.util.ArrayList;

ArrayList<Integer> numeros = new ArrayList<>();
numeros.add(10);
numeros.add(20);
```

Tipos primitivos vs clases envolventes

Solo almacenan objetos, no tipos primitivos.

Por eso se usan las clases envolventes (Wrapper)

Primitivo	**Clase Envolvente**
int	Integer
double	Double
char	Character

```
ArrayList<Integer> edades = new ArrayList<>();
```

Métodos disponibles

Incluye muchos métodos que facilitan el trabajo:

```
numeros.add(30);     // Agrega elemento
numeros.remove(0);   // Elimina elemento
numeros.size();      // Devuelve tamaño
numeros.get(1);      // Obtiene elemento
numeros.set(1, 50);  // Modifica elemento
```

Rendimiento

Son un poco más lentos porque manejan objetos.

Consumen más memoria.

Sin embargo, ofrecen mayor flexibilidad y facilidad de uso.

¿Cuándo usar cada uno?

Usa ArrayList cuando:

No sabes cuántos elementos tendrás.

Necesitas agregar o eliminar elementos constantemente.

Quieres usar métodos ya integrados.

Buscas mayor flexibilidad.