Material descargable

egg

Teoría JAVA I

Clase Arrays

La clase "**Arrays**" en Java es una herramienta fundamental que **ofrece una variedad de métodos estáticos** para facilitar la manipulación, copiado, ordenamiento, búsqueda y comparación de arreglos. Para acceder a esta clase y utilizar sus funcionalidades, es necesario importarla desde el paquete java.util.

Veamos un ejemplo:

```
import java.util.Arrays;

public class ArraysEnJava {
    public static void main(String[] args) {
        int[] original = {1,2,3};
        System.out.println(original .length); //Imprime 3
        original = Arrays.copyOf(original , 10);
        System.out.println(original .length); //Imprime 10
    }
}
```

En este código, hemos importado la *clase "Arrays"* para poder utilizarla en nuestro programa. El *método "copyOf()"* nos permite crear un nuevo arreglo con los valores del arreglo original pero con un nuevo tamaño. En este caso, el nuevo arreglo creado se asigna nuevamente a la *variable "original"*.

Algunos de los métodos más comunes de la *clase "Arrays"* son:

- **Arrays.sort():** se emplea para ordenar un arreglo en orden ascendente. Esta función es aplicable tanto a tipos primitivos como a objetos que implementen la interfaz Comparable.

```
int[] arr = {1, 5, 2, 6, 3, 7};
Arrays.sort(arr);
//El arreglo arr ahora está ordenado de forma ascendente
```

❗ *Más adelante hablaremos de lo que es una interfaz. Por ahora, solo tengamos en cuenta que tanto "String" como "Wrappers" utilizan esta interfaz*

.**Arrays.binarySearch():** Realiza una búsqueda binaria de un valor específico en un arreglo ordenado. Devuelve el índice del valor buscado en el arreglo si está presente; de lo contrario, devuelve un valor negativo.

```
int[] arr = {1, 2, 3, 4, 5};
int index = Arrays.binarySearch(arr, 3);
//Devuelve 2, que es el índice de 3 en el arreglo
```

💡**La búsqueda binaria es un algoritmo de búsqueda eficiente que se utiliza para buscar elementos específicos en colecciones ordenadas. Para utilizarla, es necesario que los objetos implementen la interfaz Comparable. Esto permite comparar y ordenar los elementos de manera adecuada, asegurando un funcionamiento correcto del algoritmo.**

- **Arrays.equals():** Compara dos arreglos para determinar si son iguales, es decir, si tienen la misma longitud y los mismos elementos en la misma posición.

```
int[] arr1 = {1, 2, 3};
int[] arr2 = {1, 2, 3};
boolean isEqual = Arrays.equals(arr1, arr2);
//Devuelve true si los arreglos son iguales
```

- **Arrays.fill():** Se utiliza para llenar todos los elementos de un arreglo con un valor específico.

```
int[] arr = new int[5];
Arrays.fill(arr, 1);
//Todos los elementos del arreglo son ahora 1
```

- **Arrays.copyOf() y Arrays.copyOfRange():** Estos métodos se utilizan para copiar un arreglo o una parte de él en un nuevo arreglo.

```
int[] original = {1, 2, 3, 4, 5};
int[] copia = Arrays.copyOf(original, original.length);
//Crea una copia del arreglo original
int[] parteDeUnaCopia= Arrays.copyOfRange(original, 1, 3);
//Crea una copia de una parte del arreglo original (índices 1 a 2 - El tercer parámetro no es inclusivo)
```

- **Arrays.toString():** Convierte un arreglo en una cadena legible, lo cual es útil para la depuración.

```
int[] arr = {1, 2, 3};
System.out.println(Arrays.toString(arr));
//Imprime "[1, 2, 3]"
```

- **Arrays.asList()** convierte un arreglo en una lista, lo que facilita su manipulación utilizando métodos proporcionados por la interfaz List.

Estos métodos son solo algunos ejemplos de las funcionalidades que ofrece la clase "Arrays" en Java para trabajar con arreglos.

💡Siempre es recomendable utilizar la **documentación oficial** para obtener más detalles sobre el material proporcionado, su uso y su implementación. Puedes acceder a la información sobre la clase Arrays desde **aquí**