Arrays Class

-Scribbles by Miss Geetanjali

Introduction to Arrays Class

The `Arrays` class in Java is a utility class from the `java.util` package that provides various methods for working with arrays. It simplifies common array operations like sorting, searching, copying, and comparing.

Practical Applications of Arrays Class

- Sorting student scores or names in ascending order.
- Searching for a specific product ID in a sorted product list.
- Filling an array with default values in game development.
- Comparing two datasets for equality in data analysis.
- Efficiently copying arrays in memory management.

Key Methods in Arrays Class

Sorting an Array (`sort()`) →The `sort()` method arranges elements in ascending order.

```
import java.util.Arrays;

public class SortTheArray {
    public static void main(String[] args) {
        int[] numbers = {5, 2, 8, 1, 3};
        Arrays.sort(numbers);
        System.out.println(Arrays.toString(numbers)); // Output: [1, 2, 3, 5, 8]
    }

}
```

Binary Search (`binarySearch()`)

This method searches for an element in a **sorted array** and returns its index.

```
package ArraysClass;

import java.util.Arrays;

public class BinarySearchArray {
    public static void main(String[] args) {
        int[] numbers = { 5, 2, 8, 1, 3 };
        // first sort the array
        Arrays.sort(numbers);
        System.out.println(Arrays.binarySearch(numbers, 1)); // 0
    }
}
```

Filling an Array (`fill()`)

The `fill()` method assigns a specified value to every element of the array.

```
public class FillMethod {
    public static void main(String[] args) {
        int[] arr = new int[5];
        Arrays.fill(arr, 7); // Output: [7, 7, 7, 7, 7]
        for(int a:arr) {
            System.out.println(a);
        }
    }

}
```

Comparing Arrays (`equals()`)

Checks if two arrays are equal element-wise.

```
package ArraysClass;

import java.util.Arrays;

public class CompareArray {
    public static void main(String[] args) {
        int[] arr1 = {1, 2, 3};
        int[] arr2 = {1, 2, 3};
        boolean isEqual = Arrays.equals(arr1, arr2); // Returns true
        System.out.println(isEqual);
    }

}
```

Converting an Array to String (`toString()`)

Prints the elements of an array as a readable string.

```
int[] numbers = {1, 2, 3};
System.out.println(Arrays.toString(numbers)); // Output: [1, 2, 3]
```

Copying Arrays (`copyOf()` and `copyOfRange()`)

- `copyOf()` copies an array up to a specified length.

- `copyOfRange()` copies a specific range of elements.

```
import java.util.Arrays;

public class CopyingArray {
    public static void main(String[] args) {
        int[] original = { 1, 2, 3, 4, 5 };
        int[] copiedArray = Arrays.copyOf(original, 3);
        int[] rangeArray = Arrays.copyOfRange(original, 1, 4);

        System.out.println(Arrays.toString(copiedArray));//[1, 2, 3]
        System.out.println(Arrays.toString(rangeArray));//[2, 3, 4]
    }
}
```

Parallel Sorting (`parallelSort()`)

A faster sorting method that uses multiple threads for sorting large arrays.

```
import java.util.Arrays;

public class ParallelSort {
    public static void main(String[] args) {
        int[] numbers = { 5, 2, 8, 1, 3 };
        Arrays.parallelSort(numbers);
        System.out.println(Arrays.toString(numbers));
    }
}
```