Day 7

Arrays Class

Java Arrays Class:

The Java **Arrays** class is a pre-defined class in java available in **java.util** package. It has methods that allows us to manipulate arrays.

Methods in Arrays Class:

- toString()
- sort()
- parallelSort()
- binarySearch()
- equals()
- deepEquals()
- copyOf()
- compare()
- fill()
- mismatch()

1. toString() Method

The toString() method of the Arrays class converts an array into a string representation. It is often used to print the contents of an array.

Example:

```
import java.util.Arrays;

public class ToStringExample {

   public static void main(String[] args) {

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

      System.out.println("Array: " + Arrays.toString(numbers));

   }

}
```

Output:

```
Array: [1, 2, 3, 4, 5]
```

2. sort() Method

The sort() method sorts the elements of an array in ascending order.

Example:

```
import java.util.Arrays;

public class SortExample {
    public static void main(String[] args) {
        int[] numbers = {5, 3, 1, 4, 2};
        Arrays.sort(numbers);
        System.out.println("Sorted Array: " + Arrays.toString(numbers));
    }
}
```

Output:

```
Sorted Array: [1, 2, 3, 4, 5]
```

3. parallelSort() Method

The parallelSort() method is similar to sort(), but it uses multiple threads to sort large arrays more efficiently.

Example:

```
import java.util.Arrays;
public class ParallelSortExample {
    public static void main(String[] args) {
        int[] numbers = {5, 3, 1, 4, 2};
        Arrays.parallelSort(numbers);
        System.out.println("Parallel Sorted Array: " + Arrays.toString(numbers));
    }
}
```

Output:

```
Parallel Sorted Array: [1, 2, 3, 4, 5]
```

4. binarySearch() Method

The binarySearch() method searches for a specific element in a sorted array and returns its index. If the element is not found, it returns a negative value.

Example:

```
import java.util.Arrays;
public class BinarySearchExample {
    public static void main(String[] args) {
        int[] numbers = {1, 2, 3, 4, 5};
        int index = Arrays.binarySearch(numbers, 4);
        System.out.println("Element found at index: " + index);
    }
}
```

Output:

```
Element found at index: 3
```

5. equals() Method

The equals() method checks if two arrays are equal. It returns true if the arrays have the same elements in the same order.

Example:

```
import java.util.Arrays;
public class EqualsExample {
    public static void main(String[] args) {
        int[] array1 = {1, 2, 3};
        int[] array2 = {1, 2, 3};
        int[] array3 = {3, 2, 1};
        System.out.println("Array1 equals Array2: " + Arrays.equals(array1, array2));
        System.out.println("Array1 equals Array3: " + Arrays.equals(array1, array3));
    }
}
```

Output:

```
Array1 equals Array2: true
Array1 equals Array3: false
```

6. deepEquals() Method

The deepEquals() method is used to compare two arrays, including nested arrays or 2D arrays. It returns true if both arrays are deeply equal.

Example:

```
import java.util.Arrays;
public class DeepEqualsExample {
    public static void main(String[] args) {
        int[][] array1 = {{1, 2, 3}, {4, 5, 6}};
        int[][] array2 = {{1, 2, 3}, {4, 5, 6}};
        int[][] array3 = {{3, 2, 1}, {6, 5, 4}};
        System.out.println("Array1 deep equals Array2: " + Arrays.deepEquals(array1, array2));
        System.out.println("Array1 deep equals Array3: " + Arrays.deepEquals(array1, array3));
    }
}
```

Output:

```
Array1 deep equals Array2: true
Array1 deep equals Array3: false
```

7. copyOf() Method

The copyOf() method creates a copy of the specified array, truncating or padding with default values if necessary.

Example:

```
import java.util.Arrays;
public class CopyOfExample {
    public static void main(String[] args) {
        int[] original = {1, 2, 3, 4, 5};
        int[] copy = Arrays.copyOf(original, 3);
        System.out.println("Original Array: " + Arrays.toString(original));
        System.out.println("Copied Array: " + Arrays.toString(copy));
    }
}
```

Output:

```
Original Array: [1, 2, 3, 4, 5]
Copied Array: [1, 2, 3]
```

8. compare() Method

The compare() method compares two arrays lexicographically. It returns a negative value if the first array is lexicographically less than the second array, a positive value if it's greater, and 0 if they are equal.

Example:

```
import java.util.Arrays;
public class CompareExample {
    public static void main(String[] args) {
        int[] array1 = {1, 2, 3};
        int[] array2 = {1, 2, 3};
        int[] array3 = {2, 2, 3};
       System.out.println("Array1 compared to Array2: " + Arrays.compare(array1, array2));
        System.out.println("Array1 compared to Array3: " + Arrays.compare(array1, array3));
    }
}
```

Output:

```
Array1 compared to Array2: 0
Array1 compared to Array3: -1
```

9. fill() Method

The fill() method assigns the specified value to each element of the array.

Example:

```
import java.util.Arrays;
public class FillExample {
    public static void main(String[] args) {
        int[] numbers = new int[5];
        Arrays.fill(numbers, 7);
        System.out.println("Filled Array: " + Arrays.toString(numbers));
    }
}
```

Output:

```
Filled Array: [7, 7, 7, 7, 7]
```

10. mismatch() Method

The mismatch() method returns the index of the first mismatch between two arrays. If there is no mismatch, it returns -1.

Example:

```
import java.util.Arrays;
public class MismatchExample {
    public static void main(String[] args) {
        int[] array1 = {1, 2, 3};
        int[] array2 = {1, 2, 4};

        int mismatchIndex = Arrays.mismatch(array1, array2);

        System.out.println("Mismatch index: " + mismatchIndex);

    }

}
```

Output:

```
Mismatch index: 2
```