

ASSIGNMENT 12

ARRAY CLASS IN JAVA

Submitted by

Name : Yadhukrishna M K

Email: yadhu8824@gmail.com

Array Classes in java

The arrays class was introduced in JDK 1.2 version. This class contains various methods for manipulating arrays such as sorting, searching, copying, converting an array to a string and etc.

Java Arrays class also contains a static factory that allows arrays to be viewed as lists. It belongs to java.util package.

Package: `java.util.Arrays`

The `java.util.Arrays` class provides a set of utility methods for working with arrays in Java. These methods are useful for performing common operations on arrays such as sorting, searching, and comparing.

Methods() :

1. `Arrays.toString()`:

To convert single dimensional array to string.

In Java to convert a single-dimensional array to a string, we can use `Arrays.toString()` method.

The **Arrays.toString()** method returns a string representation of the contents of the specified array. The string representation consists of a list of the array's elements, enclosed in square brackets “[]” and the adjacent elements are separated by the characters “, ” (a comma followed by a space). It Returns “null” if the passed array is null.

Example:

```
import java.util.Arrays;

public class ArrayTest {

    public static void main(String[] args) {

        // int array

        int[] arr = { 10, 20, 30, 40, 50 };

        System.out.println("Array = " + Arrays.toString(arr));

    }

}
```

2. Arrays.sort():

The **Arrays.sort()** method sorts the specified array into ascending numerical order. While working with the concept of the array in Java, no need to write your own logic to implement any sorting algorithm. Just import the **Arrays** class and use the **sort()** method which gives the best performance in most cases compared to other sorting algorithms.

```
import java.util.Arrays;

public class SortArray {

    // main method

    public static void main(String[] args) {

        // declare and initialize arrays

        int arr[] = { 50, 25, 30, 55, 15 };

        // display array before sorting

        System.out.println("Before Sorting: " + Arrays.toString(arr));

        // sort array

        Arrays.sort(arr);

        // display array after sorting

        System.out.println("After Sorting: " + Arrays.toString(arr));

    }

}
```

3. Arrays.equals():

In Java, the `Arrays.equals()` method can be used to check whether the two given arrays are equal or not. It returns true if the given arrays are equal else it returns false. Two arrays are considered equal if both arrays contain the same number of elements, and all corresponding pairs of elements in the two arrays are equal. In other words, two arrays are equal if they contain the same elements in the same order.

Example:

```
// Java program to demonstrate working of Arrays.equals()
import java.util.Arrays;
public class ArrayEqualDemo
{
public static void main(String[] args)
{
int[] arr1 = new int [] {1, 2, 3, 4};
int[] arr2 = new int [] {1, 2, 3, 4};
int[] arr3 = new int [] {1, 2, 4, 3};
System.out.println("is arr1 equals to arr2 : " +
Arrays.equals(arr1, arr2));
System.out.println("is arr1 equals to arr3 : " + Arrays.equals(arr1, arr3)); } }
```

4. Arrays.Compare():

The compare method of the Java Arrays class compares two arrays lexicographically.

Example:

```
public static void main(String[] args)
{
//Initialized two integer array
int[] array1 = {6, 7, 8, 11, 18, 8, 2, 5};
int[] array2 = {3, 5, 9, 13, 28, 6, 8, 9};
//compare both integer array using compare method and finally print
result
```

```
System.out.println("Result is "+ Arrays.compare(array1,array2));
```

```
}
```

```
}
```

5. Arrays.asList():

The asList() method of the Java Arrays class returns a fixed-size list backed by the specified array. Changes made to the array will be visible in the returned list, and changes made to the list will be visible in the array.

Example:

```
import java.util.Arrays;
import java.util.List;
public class ArrayTest {
    public static void main(String[] args) {
        // Integer array
        Integer[] arr = { 11, 12, 13, 14, 15 };
        // convert Integer array to list
        List<Integer> list = Arrays.asList(arr);
        // display list
        System.out.println("List = " + list);
    }
}
```

6. Arrays.mismatch() :

The mismatch() is a method that is defined under the Arrays class of Java.util package and it is used with respect to the two arrays passed as an argument in the mismatch method. This method returns the index at which two arrays passed as a parameter to the mismatch() function have the first unequal element. It is quite useful to check whether two arrays contain the same corresponding elements or not. This responds when a mismatch occurs.

Example:

```
import java.util.Arrays;

public class Main {
    public static void main(String[] args)
    {
        int intArr[] = { 10, 20, 15, 22, 35 };
        int intArr1[] = { 10, 15, 22 };
        // To compare both arrays
        System.out.println("The element mismatched at index:
        "+Arrays.mismatch(intArr, intArr1));
    }
}
```

7.Arrays.deepToString() :

The Arrays.toString() method is capable of converting single dimension array to string, but it can't convert multidimensional Java array to string. For converting the multidimensional Java array to a string we can use the deepToString() method given in the Arrays class.

It returns a string representation of the “deep contents” of the specified array. If the array contains other arrays as elements, the string representation contains their contents, and so on.

This method is designed for converting multidimensional arrays to strings.

Example:

```
import java.util.Arrays;

public class TwoDArray {

    public static void main(String[] args) {

        int[][] arr = { { 50, 60 }, { 70, 80 }, { 90, 10 } };

        // display 2D array using Arrays.toString()

        System.out.println(Arrays.deepToString(arr))}}
```