Arrays class in Java

Difficulty Level : Easy Last Updated : 25 Aug, 2021

The **Arrays** class in **java.util package** is a part of the **Java Collection Framework**. This class provides static methods to dynamically create and access **Java arrays**. It consists of only static methods and the methods of Object class. The methods of this class can be used by the class name itself.

The class hierarchy is as follows:

```
java.lang.Object
 ↳ java.util.Arrays
```

Geek, now you must be wondering why do we need java Arrays class when we are able to declare, initialize and compute operations over arrays. The answer to this though lies within the methods of this class which we are going to discuss further as practically these functions help programmers expanding horizons with arrays for instance there are often times when **loops** are used to do some tasks on an array like:

- Fill an array with a particular value.
- Sort an Arrays.
- Search in an Arrays.
- And many more.

Here Arrays class provides several static methods that can be used to perform these tasks di rectly without the use of loops, hence forth making our code super short and optimized.

Syntax: Class declaration

```
public class Arrays
    extends Object
```

Syntax: In order to use Arrays

```
Arrays.<function name>;
```

Methods in Java Array Class

The Arrays class of the java.util package contains several static methods that can be used to fill, sort, search, etc in arrays. Now let us discuss the methods of this class which are shown below in a tabular format as follows:

Methods	Action Performed
asList()	Returns a fixed-size list backed by the specified Arrays
binarySearch()	Searches for the specified element in the array with the help of the Binary Search Algorithm
binarySearch(array, fromIndex, toIndex, key, Comparator)	Searches a range of the specified array for the specified object using the Binary Search Algorithm
compare(array 1, array 2)	Compares two arrays passed as parameters lexicographically.
copyOf(originalArray, newLength)	Copies the specified array, truncating or padding with the default value (if necessary) so the copy has the specified length.
copyOfRange(originalArray, fromIndex, endIndex)	Copies the specified range of the specified array into a new Arrays.
deepEquals(Object[] a1, Object[] a2)	Returns true if the two specified arrays are deeply equal to one another.
deepHashCode(Object[] a)	Returns a hash code based on the "deep contents" of the specified Arrays.
deepToString(Object[] a)	Returns a string representation of the "deep contents" of the specified Arrays.
equals(array1, array2)	Checks if both the arrays are equal or not.
fill(originalArray, fillValue)	Assigns this fill value to each index of this arrays.
hashCode(originalArray)	Returns an integer hashCode of this array instance.
mismatch(array1, array2)	Finds and returns the index of the first unmatched element between the two specified arrays.

Methods	Action Performed
parallelPrefix(originalArray, fromIndex, endIndex, functionalOperator)	Performs parallelPrefix for the given range of the array with the specified functional operator.
parallelPrefix(originalArray, operator)	Performs parallelPrefix for complete array with the specified functional operator.
parallelSetAll(originalArray, functionalGenerator)	Sets all the elements of this array in parallel, using the provided generator function.
parallelSort(originalArray)	Sorts the specified array using parallel sort.
setAll(originalArray, functionalGenerator)	Sets all the elements of the specified array using the generator function provided.
sort(originalArray)	Sorts the complete array in ascending order.
sort(originalArray, fromIndex, endIndex)	Sorts the specified range of array in ascending order.
sort(T[] a, int fromIndex, int toIndex, Comparator< super T> c)	Sorts the specified range of the specified array of objects according to the order induced by the specified comparator.
sort(T[] a, Comparator< super T> c)	Sorts the specified array of objects according to the order induced by the specified comparator.
spliterator(originalArray)	Returns a Spliterator covering all of the specified Arrays.
spliterator(originalArray, fromIndex, endIndex)	Returns a Spliterator of the type of the array covering the specified range of the specified arrays.
stream(originalArray)	Returns a sequential stream with the specified array as its source.
toString(originalArray)	It returns a string representation of the contents of this array. The string representation consists of a list of the array's elements, enclosed in square brackets ("[]"). Adjacent elements are separated by the characters a comma followed by a space. Elements are converted to strings as by String.valueOf() function.

Implementation:

Example 1: asList() Method

```
// Java Program to Demonstrate Arrays Class
// Via asList() method

// Importing Arrays utility class
// from java.util package
import java.util.Arrays;

// Main class
class GFG {

    // Main driver method
    public static void main(String[] args)
    {
        // Get the Array
        int intArr[] = { 10, 20, 15, 22, 35 };

        // To convert the elements as List
        System.out.println("Integer Array as List: "
                           + Arrays.asList(intArr));
    }
}
```

Output

```
Integer Array as List: [[I@2f4d3709]
```

Example 2: binarySearch() Method

This methods search for the specified element in the array with the help of the binary search algorithm.

```
// Java Program to Demonstrate Arrays Class
// Via binarySearch() method

// Importing Arrays utility class
// from java.util package
```

```
import java.util.Arrays;

// Main class
public class GFG {

    // Main driver method
    public static void main(String[] args)
    {

        // Get the Array
        int intArr[] = { 10, 20, 15, 22, 35 };

        Arrays.sort(intArr);

        int intKey = 22;

        // Print the key and corresponding index
        System.out.println(
            intKey + " found at index = "
            + Arrays.binarySearch(intArr, intKey));
    }
}
```

Output

```
22 found at index = 3
```

Example 3: binarySearch(array, fromIndex, toIndex, key, Comparator) Method

This method searches a range of the specified array for the specified object using the binary search algorithm.

```
// Java program to demonstrate
// Arrays.binarySearch() method

import java.util.Arrays;

public class Main {
    public static void main(String[] args)
    {

        // Get the Array
        int intArr[] = { 10, 20, 15, 22, 35 };
        Arrays.sort(intArr);

        int intKey = 22;

        System.out.println(
            intKey
            + " found at index = "
            + Arrays
                  .binarySearch(intArr, 1, 3, intKey));
    }
}
```

Output

```
22 found at index = -4
```

Example 4: compare(array 1, array 2) Method

```
// Java program to demonstrate
// Arrays.compare() method

import java.util.Arrays;

public class Main {
    public static void main(String[] args)
    {

        // Get the Array
        int intArr[] = { 10, 20, 15, 22, 35 };

        // Get the second Array
        int intArr1[] = { 10, 15, 22 };

        // To compare both arrays
        System.out.println("Integer Arrays on comparison: "
                           + Arrays.compare(intArr, intArr1));
    }
}
```

Output

```
Integer Arrays on comparison: 1
```

Example 5: compareUnsigned(array 1, array 2) Method

```java
// Java program to demonstrate
// Arrays.compareUnsigned() method

import java.util.Arrays;

public class Main {
    public static void main(String[] args)
    {

        // Get the Arrays
        int intArr[] = { 10, 20, 15, 22, 35 };

        // Get the second Arrays
        int intArr1[] = { 10, 15, 22 };

        // To compare both arrays
        System.out.println("Integer Arrays on comparison: "
                           + Arrays.compareUnsigned(intArr, intArr1));
    }
}
```

Output

```
Integer Arrays on comparison: 1
```

Example 6: copyOf(originalArray, newLength) Method

```java
// Java program to demonstrate
// Arrays.copyOf() method

import java.util.Arrays;
```

```
public class Main {
    public static void main(String[] args)
    {

        // Get the Array
        int intArr[] = { 10, 20, 15, 22, 35 };

        // To print the elements in one line
        System.out.println("Integer Array: "
                           + Arrays.toString(intArr));

        System.out.println("\nNew Arrays by copyOf:\n");

        System.out.println("Integer Array: "
                           + Arrays.toString(
                                 Arrays.copyOf(intArr, 10)));
    }
}
```

Output

```
Integer Array: [10, 20, 15, 22, 35]


New Arrays by copyOf:


Integer Array: [10, 20, 15, 22, 35, 0, 0, 0, 0, 0]
```

Example 7: copyOfRange(originalArray, fromIndex, endIndex) Method

```
// Java program to demonstrate
// Arrays.copyOfRange() method

import java.util.Arrays;

public class Main {
    public static void main(String[] args)
    {

        // Get the Array
        int intArr[] = { 10, 20, 15, 22, 35 };

        // To print the elements in one line
        System.out.println("Integer Array: "
                              + Arrays.toString(intArr));

        System.out.println("\nNew Arrays by copyOfRange:\n");

        // To copy the array into an array of new length
        System.out.println("Integer Array: "
                              + Arrays.toString(
                                    Arrays.copyOfRange(intArr, 1, 3)));
    }
}
```

Output

```
Integer Array: [10, 20, 15, 22, 35]


New Arrays by copyOfRange:


Integer Array: [20, 15]
```

Example 8: deepEquals(Object[] a1, Object[] a2) Method

```
// Java program to demonstrate
// Arrays.deepEquals() method

import java.util.Arrays;

public class Main {
    public static void main(String[] args)
    {

        // Get the Arrays
        int intArr[][] = { { 10, 20, 15, 22, 35 } };

        // Get the second Arrays
        int intArr1[][] = { { 10, 15, 22 } };

        // To compare both arrays
        System.out.println("Integer Arrays on comparison: "
                              + Arrays.deepEquals(intArr, intArr1));
    }
}
```

Output

```
Integer Arrays on comparison: false
```

Example 9: deepHashCode(Object[] a) Method

```java
// Java program to demonstrate
// Arrays.deepHashCode() method

import java.util.Arrays;

public class Main {
    public static void main(String[] args)
    {

        // Get the Array
        int intArr[][] = { { 10, 20, 15, 22, 35 } };

        // To get the dep hashCode of the arrays
        System.out.println("Integer Array: "
                           + Arrays.deepHashCode(intArr));
    }
}
```

Output

```
Integer Array: 38475344
```

Example 10: deepToString(Object[] a) Method

```java
// Java program to demonstrate
// Arrays.deepToString() method

import java.util.Arrays;
```

```
public class Main {
    public static void main(String[] args)
    {

        // Get the Array
        int intArr[][] = { { 10, 20, 15, 22, 35 } };

        // To get the deep String of the arrays
        System.out.println("Integer Array: "
                           + Arrays.deepToString(intArr));
    }
}
```

Output

```
Integer Array: [[10, 20, 15, 22, 35]]
```

Example 11: equals(array1, array2) Method

```
// Java program to demonstrate
// Arrays.equals() method

import java.util.Arrays;

public class Main {
    public static void main(String[] args)
    {

        // Get the Arrays
        int intArr[] = { 10, 20, 15, 22, 35 };

        // Get the second Arrays
        int intArr1[] = { 10, 15, 22 };

        // To compare both arrays
        System.out.println("Integer Arrays on comparison: "
                           + Arrays.equals(intArr, intArr1));
    }
}
```

Output

```
Integer Arrays on comparison: false
```

Example 12: fill(originalArray, fillValue) Method

```java
// Java program to demonstrate
// Arrays.fill() method

import java.util.Arrays;

public class Main {
    public static void main(String[] args)
    {

        // Get the Arrays
        int intArr[] = { 10, 20, 15, 22, 35 };

        int intKey = 22;

        Arrays.fill(intArr, intKey);

        // To fill the arrays
        System.out.println("Integer Array on filling: "
                           + Arrays.toString(intArr));
    }
}
```

Output

```
Integer Array on filling: [22, 22, 22, 22, 22]
```

Example 13: hashCode(originalArray) Method

```java
// Java program to demonstrate
// Arrays.hashCode() method

import java.util.Arrays;
```

```
public class Main {
    public static void main(String[] args)
    {

        // Get the Array
        int intArr[] = { 10, 20, 15, 22, 35 };

        // To get the hashCode of the arrays
        System.out.println("Integer Array: "
                           + Arrays.hashCode(intArr));
    }
}
```

Output

```
Integer Array: 38475313
```

Example 14: mismatch(array1, array2) Method

```
// Java program to demonstrate
// Arrays.mismatch() method

import java.util.Arrays;

public class Main {
    public static void main(String[] args)
    {

        // Get the Arrays
        int intArr[] = { 10, 20, 15, 22, 35 };

        // Get the second Arrays
        int intArr1[] = { 10, 15, 22 };

        // To compare both arrays
        System.out.println("The element mismatched at index: "
                           + Arrays.mismatch(intArr, intArr1));
    }
}
```

Output

```
The element mismatched at index: 1
```

Example 15: parallelSort(originalArray) Method

```java
// Java program to demonstrate
// Arrays.parallelSort() method

// Importing Arrays class from
// java.util package
import java.util.Arrays;

// Main class
public class Main {
    public static void main(String[] args)
    {

        // Get the Array
        int intArr[] = { 10, 20, 15, 22, 35 };

        // To sort the array using parallelSort
        Arrays.parallelSort(intArr);

        System.out.println("Integer Array: "
                           + Arrays.toString(intArr));
    }
}
```

Output

```
Integer Array: [10, 15, 20, 22, 35]
```

Example 16: sort(originalArray) Method

```java
// Java program to demonstrate
// Arrays.sort() method
```

```
import java.util.Arrays;

public class Main {
    public static void main(String[] args)
    {

        // Get the Array
        int intArr[] = { 10, 20, 15, 22, 35 };

        // To sort the array using normal sort-
        Arrays.sort(intArr);

        System.out.println("Integer Array: "
                           + Arrays.toString(intArr));
    }
}
```

Output

```
Integer Array: [10, 15, 20, 22, 35]
```

Example 17: sort(originalArray, fromIndex, endIndex) Method

```
// Java program to demonstrate
// Arrays.sort() method

import java.util.Arrays;

public class Main {
    public static void main(String[] args)
    {

        // Get the Array
        int intArr[] = { 10, 20, 15, 22, 35 };

        // To sort the array using normal sort
        Arrays.sort(intArr, 1, 3);

        System.out.println("Integer Array: "
                           + Arrays.toString(intArr));
    }
}
```

Output

```
Integer Array: [10, 15, 20, 22, 35]
```

Example 18: sort(T[] a, int fromIndex, int toIndex, Comparator< super T> c) Method

```java
// Java program to demonstrate working of Comparator
// interface
import java.util.*;
import java.lang.*;
import java.io.*;

// A class to represent a student.
class Student {
    int rollno;
    String name, address;

    // Constructor
    public Student(int rollno, String name,
                   String address)
    {
        this.rollno = rollno;
        this.name = name;
        this.address = address;
    }

    // Used to print student details in main()
    public String toString()
    {
        return this.rollno + " "
            + this.name + " "
            + this.address;
    }
}

class Sortbyroll implements Comparator<Student> {
    // Used for sorting in ascending order of
    // roll number
    public int compare(Student a, Student b)
    {
        return a.rollno - b.rollno;
    }
}

// Driver class
class Main {
    public static void main(String[] args)
    {
        Student[] arr = { new Student(111, "bbbb", "london"),
                          new Student(131, "aaaa", "nyc"),
                          new Student(121, "cccc", "jaipur") };

        System.out.println("Unsorted");
        for (int i = 0; i < arr.length; i++)
            System.out.println(arr[i]);

        Arrays.sort(arr, 1, 2, new Sortbyroll());

        System.out.println("\nSorted by rollno");
        for (int i = 0; i < arr.length; i++)
            System.out.println(arr[i]);
    }
}
```

Output

```
Unsorted
111 bbbb london
131 aaaa nyc
121 cccc jaipur

Sorted by rollno
111 bbbb london
131 aaaa nyc
121 cccc jaipur
```

Example 19: sort(T[] a, Comparator< super T> c) Method

```
// Java program to demonstrate working of Comparator
// interface
import java.util.*;
import java.lang.*;
import java.io.*;

// A class to represent a student.
class Student {
    int rollno;
    String name, address;

    // Constructor
    public Student(int rollno, String name,
                   String address)
    {
        this.rollno = rollno;
        this.name = name;
        this.address = address;
    }

    // Used to print student details in main()
    public String toString()
    {
        return this.rollno + " "
            + this.name + " "
            + this.address;
    }
}

class Sortbyroll implements Comparator<Student> {

    // Used for sorting in ascending order of
    // roll number
    public int compare(Student a, Student b)
    {
        return a.rollno - b.rollno;
    }
}

// Driver class
class Main {
    public static void main(String[] args)
    {
        Student[] arr = { new Student(111, "bbbb", "london"),
                          new Student(131, "aaaa", "nyc"),
                          new Student(121, "cccc", "jaipur") };

        System.out.println("Unsorted");
        for (int i = 0; i < arr.length; i++)
            System.out.println(arr[i]);

        Arrays.sort(arr, new Sortbyroll());

        System.out.println("\nSorted by rollno");
        for (int i = 0; i < arr.length; i++)
            System.out.println(arr[i]);
    }
}
```

Output

```
Unsorted
111 bbbb london
131 aaaa nyc
121 cccc jaipur

Sorted by rollno
111 bbbb london
121 cccc jaipur
131 aaaa nyc
```

Example 20: spliterator(originalArray) Method

```java
// Java program to demonstrate
// Arrays.spliterator() method

import java.util.Arrays;

public class Main {
    public static void main(String[] args)
    {

        // Get the Array
        int intArr[] = { 10, 20, 15, 22, 35 };

        // To sort the array using normal sort
        System.out.println("Integer Array: "
                           + Arrays.spliterator(intArr));
    }
}
```

Output

```
Integer Array: java.util.Spliterators$IntArraySpliterator@4e50df2e
```

Example 21: spliterator(originalArray, fromIndex, endIndex) Method

```java
// Java program to demonstrate
// Arrays.spliterator() method

import java.util.Arrays;

public class Main {
    public static void main(String[] args)
    {

        // Get the Array
        int intArr[] = { 10, 20, 15, 22, 35 };

        // To sort the array using normal sort
        System.out.println("Integer Array: "
                           + Arrays.spliterator(intArr, 1, 3));
    }
}
```

Output

```
Integer Array: java.util.Spliterators$IntArraySpliterator@4e50df2e
```

Example 22: stream(originalArray) Method

```java
// Java program to demonstrate
// Arrays.stream() method

import java.util.Arrays;

public class Main {
    public static void main(String[] args)
    {

        // Get the Array
        int intArr[] = { 10, 20, 15, 22, 35 };

        // To get the Stream from the array
        System.out.println("Integer Array: "
                           + Arrays.stream(intArr));
    }
}
```

Output

```
Integer Array: java.util.stream.IntPipeline$Head@7291c18f
```

Example 23: toString(originalArray) Method

```
// Java program to demonstrate
// Arrays.toString() method

import java.util.Arrays;

public class Main {
    public static void main(String[] args)
    {

        // Get the Array
        int intArr[] = { 10, 20, 15, 22, 35 };

        // To print the elements in one line
        System.out.println("Integer Array: "
                           + Arrays.toString(intArr));
    }
}
```

Output

```
Integer Array: [10, 20, 15, 22, 35]
```

This article is contributed by **Rishabh Mahrsee**. If you like GeeksforGeeks and would like to contribute, you can also write an article and mail your article to review-team@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.
Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above