Java Arrays sort() – Examples

Java Arrays sort()

sort() method sorts the array inplace in ascending order.

sort() method uses the array element's compareTo() method to compare two elements at a time and check which is greater or lesser and arrange them accordingly.

java.util.Arrays class has two versions of sort() method for all the numeric, char, string and user defined types. The first version sorts all the elements, and the second one sorts elements only in the specified index range.

Example 1 – Arrays sort() – byte Array

A byte can store numbers of range [-128, 127]. In this example, we will take a byte array and sort the array using Arrays.sort().

Java Program

```
import java.util.Arrays;

public class Example {
    public static void main(String[] args) {
        byte arr[] = {2, 8, 1, 6, 9, 4, 3};
        Arrays.sort(arr);
        System.out.println(Arrays.toString(arr));
    }
}
```

Output

```
[1, 2, 3, 4, 6, 8, 9]
```

Example 2 – Arrays sort() – byte Array – Specific Range

We can also sort only a specific index range of the given array. We have to pass starting index, and the index to which sorting has to happen.

In the following example, we will take a byte array of size 7, and sort the array only starting with index 1 and up to index 4. So, the elements only with index 1, 2 and 3, will be sorted. Rest of the elements are unchanged.

Java Program

```java
import java.util.Arrays;

public class Example {
    public static void main(String[] args) {
        byte arr[] = {2, 8, 1, 6, 9, 4, 3};
        int fromIndex = 1;
        int toIndex = 4;
        Arrays.sort(arr, fromIndex, toIndex);
        System.out.println(Arrays.toString(arr));
    }
}
```

Output

```
[2, 1, 6, 8, 9, 4, 3]
```

Example 3 – Arrays sort() – char Array

char datatype can store single character/letter or ASCII values.

In the following example, we will initialize a char array and sort it using Arrays.sort() method.

Java Program

```java
import java.util.Arrays;

public class Example {
    public static void main(String[] args) {
        char arr[] = {'c', 'a', 'm', 'b', 'e', 'a', 'o'};
        Arrays.sort(arr);
        System.out.println(Arrays.toString(arr));
    }
}
```

Output

```
[a, a, b, c, e, m, o]
```

Example 4 – Arrays sort() – char Array – Specific Range

In the following example, we will sort the char array, only for the specified range of index.

Java Program

```java
import java.util.Arrays;

public class Example {
    public static void main(String[] args) {
        char arr[] = {'c', 'a', 'm', 'b', 'e', 'a', 'o'};
        int fromIndex = 1;
        int toIndex = 4;
        Arrays.sort(arr, fromIndex, toIndex);
        System.out.println(Arrays.toString(arr));
    }
}
```

Output

```
[2, 1, 6, 8, 9, 4, 3]
```

Example 5 – Arrays sort() – double Array

In the following example, we will take a double array and sort it using Arrays.sort() method.

Java Program

```java
import java.util.Arrays;

public class Example {
    public static void main(String[] args) {
        double arr[] = {2.365, 1.06, 5.236, 88, 865.58, 8.01, 0.21};
        Arrays.sort(arr);
        System.out.println(Arrays.toString(arr));
    }
}
```

Output

```
[0.21, 1.06, 2.365, 5.236, 8.01, 88.0, 865.58]
```

Example 6 – Arrays sort() – double Array – Specific Range

In the following program, we will sort only a specific index range of the given double array.

Java Program

```java
import java.util.Arrays;

public class Example {
    public static void main(String[] args) {
        double arr[] = {2.365, 1.06, 5.236, 88, 865.58, 8.01, 0.21};
        int fromIndex = 1;
        int fromIndex = 1;
        int toIndex = 4;
        Arrays.sort(arr, fromIndex, toIndex);
        System.out.println(Arrays.toString(arr));
    }
}
```

Output

```
[2.365, 1.06, 5.236, 88.0, 865.58, 8.01, 0.21]
```

Example 7 – Arrays sort() – float Array

In the following example, we will sort a float array using Arrays.sort().

Java Program

```
import java.util.Arrays;

public class Example {
    public static void main(String[] args) {
        float arr[] = {2.365f, 1.06f, 5.236f, 88f, 865.58f, 8.01f, 0.21f};
        Arrays.sort(arr);
        System.out.println(Arrays.toString(arr));
    }
}
```

Output

```
[0.21, 1.06, 2.365, 5.236, 8.01, 88.0, 865.58]
```

Example 8 – Arrays sort() – float Array – Specific Range

In this example, we will sort a float array only in the specified index range of fromIndex upto toIndex using Arrays.sort() method.

Java Program

```
import java.util.Arrays;

public class Example {
    public static void main(String[] args) {
        float arr[] = {2.365f, 1.06f, 5.236f, 88f, 865.58f, 8.01f, 0.21f};
        int fromIndex = 1;
        int toIndex = 4;
        Arrays.sort(arr, fromIndex, toIndex);
        System.out.println(Arrays.toString(arr));
    }
}
```

Output

```
[2.365, 1.06, 5.236, 88.0, 865.58, 8.01, 0.21]
```

Example 9 – Arrays sort() – int Array

In the following program, we will sort an integer array using Arrays.sort() method.

Java Program

```java
import java.util.Arrays;

public class Example {
    public static void main(String[] args) {
        int arr[] = {2, 14, 5, 8, 7, 9, 0};
        Arrays.sort(arr);
        System.out.println(Arrays.toString(arr));
    }
}
```

Output

```
[0, 2, 5, 7, 8, 9, 14]
```

Example 10 – Arrays sort() – int Array – Specific Range

In the following Java program, we will sort the elements that are present in the given range of index. The elements outside of the range are not considered for sorting.

Java Program

```java
import java.util.Arrays;

public class Example {
    public static void main(String[] args) {
        int arr[] = {2, 14, 5, 8, 7, 9, 0};
        int fromIndex = 1;
        int toIndex = 4;
        Arrays.sort(arr, fromIndex, toIndex);
        System.out.println(Arrays.toString(arr));
    }
}
```

Output

```
[2, 5, 8, 14, 7, 9, 0]
```

Example 11 – Arrays sort() – long Array

In this example, we will take an array of long numbers, and sort the array using Arrays.sort().

Java Program

```java
import java.util.Arrays;

public class Example {
    public static void main(String[] args) {
        long arr[] = {2, 14, 5, 8, 7, 9, 0};
        Arrays.sort(arr);
        System.out.println(Arrays.toString(arr));
    }
}
```

Output

```
[0, 2, 5, 7, 8, 9, 14]
```

Example 12 – Arrays sort() – long Array – Specific Range

In the following program, we will sort the long array, only in the specified array range.

Java Program

```java
import java.util.Arrays;

public class Example {
    public static void main(String[] args) {
        long arr[] = {2, 14, 5, 8, 7, 9, 0};
        int fromIndex = 1;
        int toIndex = 4;
        Arrays.sort(arr, fromIndex, toIndex);
        System.out.println(Arrays.toString(arr));
    }
}
```

Output

```
[2, 5, 8, 14, 7, 9, 0]
```

Example 13 – Arrays sort() – Object Array

When sorting items of type Object, Java uses natural ordering[https://docs.oracle.com/javase/7/docs/api/java/lang/Comparable.html] of its elements. The natural ordering decides which object is less than or greater than or equal to another object. And based on these decisions, an Object array is sorted.

In the following example, we have taken an object array. Of course, we have given integers as elements to not get confused with sorting of objects. But only with natural ordering, these elements are sorted.

Java Program

```
import java.util.Arrays;

public class Example {
    public static void main(String[] args) {
        Object arr[] = {2, 14, 5, 8, 7, 9, 0};
        Arrays.sort(arr);
        System.out.println(Arrays.toString(arr));
    }
}
```

Output

```
[0, 2, 5, 7, 8, 9, 14]
```

Example 14 – Arrays sort() – Object Array – Specific Range

Java Program

```
import java.util.Arrays;

public class Example {
    public static void main(String[] args) {
        Object arr[] = {2, 14, 5, 8, 7, 9, 0};
        int fromIndex = 1;
        int toIndex = 4;
        Arrays.sort(arr, fromIndex, toIndex);
        System.out.println(Arrays.toString(arr));
    }
}
```

Output

```
[2, 5, 8, 14, 7, 9, 0]
```

Example 15 – Arrays sort() – short Array

Java Program

```
import java.util.Arrays;
```

```
public class Example {
    public static void main(String[] args) {
        short arr[] = {2, 14, 5, 8, 7, 9, 0};
        Arrays.sort(arr);
        System.out.println(Arrays.toString(arr));
    }
}
```

Output

```
[0, 2, 5, 7, 8, 9, 14]
```

Example 16 – Arrays sort() – short Array – Specific Range

Java Program

```
import java.util.Arrays;

public class Example {
    public static void main(String[] args) {
        short arr[] = {2, 14, 5, 8, 7, 9, 0};
        int fromIndex = 1;
        int toIndex = 4;
        Arrays.sort(arr, fromIndex, toIndex);
        System.out.println(Arrays.toString(arr));
    }
}
```

Output

```
[2, 5, 8, 14, 7, 9, 0]
```

Conclusion

In this Java Tutorial, we learned how to use Arrays.sort() method of java.util package, to sort an array in ascending order, with the help of example programs.

✦ Java Tutorial

✦ Java java.util.Arrays

✦ Java Arrays.asList()

✦ Java Arrays.binarySearch()

✦ Java Arrays.copyOf()

✦ Java Arrays.copyOfRange()

✦ Java Arrays.deepEquals()

✦ Java Arrays.deepToString()

✦ Java Arrays.equals()

✦ Java Arrays.fill()

✦ Java Arrays.hashCode()

➩ **Java Arrays.sort()**

✦ Java Arrays.toString()