The Arrays class has a `sort()` method that is used to sort the arrays of objects and primitives. If we are sorting a primitive array, then **quicksort** is used. And if we are sorting an object array, then **merge sort** is used.

Although quicksort is faster in both cases, it is not a stable algorithm. Merge sort is a stable algorithm, so it is used in the case of sorting an object array. In the case of the primitive array, we don't care about stability, so quicksort is used.

> Stable sorting algorithms are algorithms that maintain the relative order of equal elements. For example, we have an array [1,4,6,8,6], which we need to sort. Now after sorting this array, the result is [1,4,6,6,8]. Although there are two sixes in the array, we don't care which six came first in the sorted array. But in the case of an object array, the relative order of elements also matters. If two objects are the same in an object array, then their relative order should be the same in the sorted array.

The sort method has two variants:

1. `sort(array)` – sorts the full array into ascending order
2. `sort(array, fromIndex, toIndex)` – sorts only the elements from *fromIndex* to *toIndex*.

```
import java.util.Arrays;

public class ArraysDemo {

    public static void main(String args[]) {

        Integer[] numbers = { 10, 2, 32, 12, 15, 76, 17, 48, 79, 9 };
        Arrays.sort(numbers);

        for (int i : numbers) {
            System.out.print(i + " ");
        }
    }
}

```

Run Save Reset

Sorting an array in parallel#

In Java 8, a new method `parallelSort()` was introduced to sort the arrays parallelly. Unlike `sort()`, which sorts data sequentially using a single thread, `parallelSort()` uses a parallel sort-merge sorting algorithm. It breaks the array into sub-arrays that are themselves sorted and then merged.

This method uses the ForkJoin pool for executing parallel tasks. The array is sorted parallelly only when certain conditions are met. If the array size is less than or equal to 8192 or the processor has only one core, then the sequential dual-pivot Quicksort algorithm is used. Otherwise, it uses a parallel sort.

```
import java.util.Arrays;

public class ArraysDemo {

    public static void main(String args[]) {

        Integer[] numbers = { 10, 2, 32, 12, 15, 76, 17, 48, 79, 9 };
        Arrays.parallelSort(numbers);

        for (int i : numbers) {
            System.out.print(i + " ");
        }
    }
}

```

Run Save Reset