```
/**
 * Take leftArray and rightArray, which we assume are individually sorted,
 * and combine them, sorted, into outputArray.
 */
private static void merge(int[] outputArray, int[] leftArray, int[] rightArray) {
    int i = 0;     // Index into the left input array.
    int j = 0;     // Index into the right input array.
    int k = 0;     // Index into the output array.

    // While there is data in both input arrays:
    while (i < leftArray.length && j < rightArray.length) {

        // Find the smaller item and insert it into the output array.
        if (leftArray[i] < rightArray[j]) {
            outputArray[k] = leftArray[i];
            k++;
            i++;
        } else {
            outputArray[k] = rightArray[j];
            k++;
            j++;
        }
    }

    // We know that one of the arrays has more items to copy, and the other is empty.
    // Copy remaining input from left array into the output.
    while (i < leftArray.length) {
        outputArray[k] = leftArray[i];
        k++;
        i++;
    }
    // Copy remaining input from right array into output.
    while (j < rightArray.length) {
        outputArray[k] = rightArray[j];
        k++;
        j++;
    }
}

public static void mergesort(int[] array) {
    // A array with one element is sorted already.
    if (array.length > 1) {
        // Split array into halves.
        int halfSize = array.length / 2;
        int[] leftArray = new int[halfSize];
        int[] rightArray = new int[array.length - halfSize];

        // Copy left half of array into leftArray:
        System.arraycopy(array, 0, leftArray, 0, halfSize);

        // Copy right half of array into rightArray:
        System.arraycopy(array, halfSize, rightArray, 0, array.length - halfSize);

        // Sort the halves.
        mergesort(leftArray);
        mergesort(rightArray);

        // Merge the halves.
        merge(array, leftArray, rightArray);
    }
}
```