```java
public class MergeSortAlgorithm {
    public static void main(String[] args) {
        AlgorithmService alg = new AlgorithmService();
        int size = 10;
        long sum = 0;

        int k = size / 4;
        int k1 = size / 2;
        int k11 = 3 * size / 4;

        int arr[] = new int[size];
        alg.fillArray(arr);

        for (int i = 0; i < 20; i++) {
            long start = System.nanoTime();

            mergeSort(arr, 0, arr.length - 1);
            alg.kthSmallest(arr, arr[0], arr.length, k);

            long end = System.nanoTime();
            long total = end - start;
            sum += total;

            System.out.println(total);
        }
        System.out.println("\nThe average time is: " + sum / 15 + " nanoseconds");
    }

    // -----------------------------MERGE SORT-----------------------------
    /**
     * Given a list of n numbers, the Selection Problem is to find the xth smallest
     * element in the list.
     * This algorithm sorts merges two subarrays of arr[]
     *
     * @param x
     * @param y
     * @param z
     * @param arr
     */
    public static void mergeSortSplit(int left, int middle, int right, int arr[]) {
        // size of the subarray two be merged
        int size1 = (middle - left) + 1;
        int size2 = (right - middle);

        // temporary array
        int tempArray1[] = new int[size1];
        int tempArray2[] = new int[size2];

        // Copy the arr from the parameter into the tempporary arrays
        for (int i = 0; i < size1; ++i)
            tempArray1[i] = arr[left + 1];
        for (int j = 0; j < size2; ++j)
            tempArray2[j] = arr[middle + 1 + j];

        // intitial index
        int a = 0;
        int b = 0;

        // initial index of merged subarray
        while (a < size1 && b < size2) {
            if (tempArray1[a] <= tempArray2[b]) {
                arr[left] = tempArray1[a];
                a++;
            } else {
                arr[left] = tempArray2[b];
                b++;
            }
            left++;
        }

        // Copy any remaining elements onto the temparrays
        while (a < size1) {
            arr[left] = tempArray1[a];
            a++;
            left++;
        }
        while (b < size2) {
            arr[left] = tempArray2[b];
            b++;
            left++;
        }
    }

    /**
     * The main merge algorithm that sorts the array
     *
     * @param arr
     * @param left
     * @param right
     */
    public static void mergeSort(int arr[], int x, int k) {
        if (x < k) {
            int middle = x + (k - x) / 2;

            mergeSort(arr, x, middle);
            mergeSort(arr, middle + 1, k);

            mergeSortSplit(x, middle, k, arr);
        }
    }
    // -----------------------------END MERGE SORT-----------------------------
}
```