import java.util.\*;

public class SortExample {

public static void main(String[] args) {

// This is an unsorted array

Integer[] array1 = {12, 13, 24, 10, 3, 6, 90, 70};

int[] array2 = {2, 6, 3, 5, 1};

// Sort using quick sort

quickSort(array1, 0, array1.length - 1);

System.out.println("QuickSort result: " + Arrays.toString(array1));

// Sort using merge sort

mergeSort(array2, array2.length);

System.out.println("MergeSort result: " + Arrays.toString(array2));

}

public static void quickSort(Integer[] arr, int low, int high) {

if (arr == null || arr.length == 0 || low >= high) {

return;

}

// Get the pivot element

int middle = low + (high - low) / 2;

int pivot = arr[middle];

int i = low, j = high;

while (i <= j) {

while (arr[i] < pivot) {

i++;

}

while (arr[j] > pivot) {

j--;

}

if (i <= j) {

swap(arr, i, j);

i++;

j--;

}

}

if (low < j) {

quickSort(arr, low, j);

}

if (high > i) {

quickSort(arr, i, high);

}

}

public static void swap(Integer[] array, int x, int y) {

int temp = array[x];

array[x] = array[y];

array[y] = temp;

}

public static void mergeSort(int[] a, int n) {

if (n < 2) {

return;

}

int mid = n / 2;

int[] l = Arrays.copyOfRange(a, 0, mid);

int[] r = Arrays.copyOfRange(a, mid, n);

mergeSort(l, mid);

mergeSort(r, n - mid);

merge(a, l, r, mid, n - mid);

}

public static void merge(int[] a, int[] l, int[] r, int left, int right) {

int i = 0, j = 0, k = 0;

while (i < left && j < right) {

if (l[i] <= r[j]) {

a[k++] = l[i++];

} else {

a[k++] = r[j++];

}

}

while (i < left) {

a[k++] = l[i++];

}

while (j < right) {

a[k++] = r[j++];

}

}

private static boolean isSorted(int[] x) {

for (int i = 0; i < x.length - 1; i++) {

if (x[i] > x[i + 1]) {

return false;

}

}

return true;

}

}