import java.util.Random;

public class Mergesort {

public static void merge(int arr[], int left, int mid, int right) {

int n1 = mid - left + 1;

int n2 = right - mid;

int leftArr[] = new int[n1];

int rightArr[] = new int[n2];

for (int i = 0; i < n1; i++) {

leftArr[i] = arr[left + i];

}

for (int j = 0; j < n2; j++) {

rightArr[j] = arr[mid + 1 + j];

}

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

while (i < n1 && j < n2) {

if (leftArr[i] <= rightArr[j]) {

arr[k] = leftArr[i];

i++;

} else {

arr[k] = rightArr[j];

j++;

}

k++;

}

while (i < n1) {

arr[k] = leftArr[i];

i++;

k++;

}

while (j < n2) {

arr[k] = rightArr[j];

j++;

k++;

}

}

public static void mergeSort(int arr[], int left, int right) {

if (left < right) {

int mid = (left + right) / 2;

mergeSort(arr, left, mid);

mergeSort(arr, mid + 1, right);

merge(arr, left, mid, right);

}

}

public static void threadedMergeSort(int arr[], int left, int right, int depth) {

if (left < right) {

if (depth < 2) {

int mid = (left + right) / 2;

Thread t1 = new Thread(() -> threadedMergeSort(arr, left, mid, depth + 1));

Thread t2 = new Thread(() -> threadedMergeSort(arr, mid + 1, right, depth + 1));

t1.start();

t2.start();

try {

t1.join();

t2.join();

} catch (InterruptedException e) {

e.printStackTrace();

}

} else {

mergeSort(arr, left, right);

}

}

}

public static void main(String[] args) {

int arrSize = 1000;

int[] inputArray = new int[arrSize];

Random random = new Random();

System.out.println("Input Array (1000 elements):");

for (int i = 0; i < arrSize; i++) {

inputArray[i] = random.nextInt(1000); // Generates random integers from 0 to 999.

System.out.print(inputArray[i] + " ");

}

int[] simpleMergeSortArray = inputArray.clone();

int[] threadedMergeSortArray = inputArray.clone();

long startTime, endTime;

// Simple Merge Sort

startTime = System.nanoTime();

mergeSort(simpleMergeSortArray, 0, arrSize - 1);

endTime = System.nanoTime();

System.out.println("\nSimple Merge Sort Time: " + (endTime - startTime) + " ns");

// Threaded Merge Sort

startTime = System.nanoTime();

threadedMergeSort(threadedMergeSortArray, 0, arrSize - 1, 0);

endTime = System.nanoTime();

System.out.println("Threaded Merge Sort Time: " + (endTime - startTime) + " ns");

}

}