Assignment 1 , 2 –

import java.util.Arrays;

import java.util.Random;

public class QuickAndMerge{

public static void main(String[] args) {

int[] arr = getRandomArray(500);

System.out.println(Arrays.toString(arr));

long s = System.nanoTime();

sort(arr, 0, arr.length-1);

long e = System.nanoTime();

System.out.println("It took " + (e-s) + " nanoseconds to quicksort the array");

System.out.println(Arrays.toString(arr));

System.out.println();

arr = getRandomArray(500);

System.out.println(Arrays.toString(arr));

s = System.nanoTime();

mergeSort(arr, arr.length-1);

e = System.nanoTime();

System.out.println("It took " + (e-s) + " nanoseconds to mergesort the array");

System.out.println(Arrays.toString(arr));

}

private static int[] getRandomArray(int N) {

Random random = new Random();

return random.ints(N, 0, N\*100).toArray();

}

private static void sort(int[] array, int low, int high) {

int p; //index of partition

if (low < high) {

p = partition(array, low, high);

// System.out.println(p);

sort(array, low, p-1);

sort(array, p+1, high);

}

}

private static int partition(int[] array, int low, int high) {

int i = low;

int j = high+1;

int valueOfPivot = array[low];

while(true) {

while(array[++i] < valueOfPivot) {

if (i == high)

break;

}

while(valueOfPivot < array[--j]) {}

if (i>=j) break;

exch(array, i, j);

}

exch(array, low, j);

return j;

}

private static void exch(int[] a, int i, int j) {

int swap = a[i];

a[i] = a[j];

a[j] = swap;

}

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

if (n < 2) {

return;

}

int mid = n / 2;

int[] l = new int[mid];

int[] r = new int[n - mid];

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

l[i] = a[i];

}

for (int i = mid; i < n; i++) {

r[i - mid] = a[i];

}

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++];

}

}

}