Merge Sort:

// package Codes.Sorting;

import java.util.\*;

import java.lang.Math;

public class mergeSort {

static void merge(int arr[], int l, int m, int r)

{

// Find sizes of two subarrays to be merged

int n1 = m - l + 1;

int n2 = r - m;

/\* Create temp arrays \*/

int L[] = new int[n1];

int R[] = new int[n2];

/\*Copy data to temp arrays\*/

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

L[i] = arr[l + i];

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

R[j] = arr[m + 1 + j];

/\* Merge the temp arrays \*/

// Initial indexes of first and second subarrays

int i = 0, j = 0;

// Initial index of merged subarray array

int k = l;

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

if (L[i] <= R[j]) {

arr[k] = L[i];

i++;

}

else {

arr[k] = R[j];

j++;

}

k++;

}

/\* Copy remaining elements of L[] if any \*/

while (i < n1) {

arr[k] = L[i];

i++;

k++;

}

/\* Copy remaining elements of R[] if any \*/

while (j < n2) {

arr[k] = R[j];

j++;

k++;

}

}

// Main function that sorts arr[l..r] using

// merge()

static void sort(int arr[], int l, int r)

{

if (l < r) {

// Find the middle point

int m =l+ (r-l)/2;

// Sort first and second halves

sort(arr, l, m);

sort(arr, m + 1, r);

// Merge the sorted halves

merge(arr, l, m, r);

}

}

/\* A utility function to print array of size n \*/

static void printArray(int arr[])

{

int n = arr.length;

for (int i = 0; i < n; ++i)

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

System.out.println();

}

// Driver code

public static void main(String args[])

{

Scanner sc = new Scanner(System.in);

System.out.println("Enter number of elements: ");

int N=sc.nextInt();

int arr[];

arr=new int[N];

// System.out.println("Given Array");

for(int i=0;i<N;i++)

{

double x=Math.random()\*300;

int num= (int)x;

arr[i]= num;

}

printArray(arr);

long start = System.nanoTime();

// mergeSort ob = new mergeSort();

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

System.out.println("\nSorted array");

printArray(arr);

long end = System.nanoTime();

long execution = end - start;

System.out.println("Execution time: " + execution + " nanoseconds");

}

}

Quick Sort:

// package Codes.Sorting;

import java.io.\*;

import java.time.\*;

import java.util.\*;

import java.lang.Math;

public class quickSort {

static void swap(int[] arr, int i, int j)

{

int temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

static int partition(int[] arr, int low, int high)

{

int pivot = arr[high];

int i = (low - 1);

for(int j = low; j <= high - 1; j++)

{

if (arr[j] < pivot)

{

i++;

swap(arr, i, j);

}

}

swap(arr, i + 1, high);

return (i + 1);

}

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

{

if (low < high)

{

int pi = partition(arr, low, high);

quickSort(arr, low, pi - 1);

quickSort(arr, pi + 1, high);

}

}

static void printArray(int[] arr, int size)

{

for(int i = 0; i < size; i++)

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

System.out.println();

}

// Driver Code

public static void main(String[] args)

{

Scanner sc = new Scanner(System.in);

System.out.println("Enter number of elements: ");

int N=sc.nextInt();

int arr[];

arr=new int[N];

// System.out.println("Given Array");

for(int i=0;i<N;i++)

{

double x=Math.random()\*300;

int num= (int)x;

arr[i]= num;

}

printArray(arr,N);

long start = System.nanoTime();

quickSort(arr, 0, N - 1);

System.out.println("Sorted array: ");

printArray(arr, N);

long end = System.nanoTime();

long execution = end - start;

System.out.println("Execution time: " + execution + " nanoseconds");

}

}