Sorting

import org.w3c.dom.css.CSSStyleDeclaration;  
  
import java.lang.reflect.Array;  
import java.util.\*;  
  
import static java.util.Objects.*hash*;  
  
class Sorting {  
 int[] bubble(int[] arr) {  
 int n = arr.length;  
 for (int i = 0; i < n - 1; i++) {//-n  
 for (int j = i + 1; j < n; j++) {//--n\*n  
 if (arr[i] > arr[j]) {  
 swap(arr, i, j);  
 }  
 }  
 }  
  
 return arr;  
 }  
  
 int[] selection(int[] arr) {  
 int n = arr.length;  
 for (int i = 0; i < n; i++) {  
 int last = n - i - 1;  
 int maxindex = MAXOF(arr, 0, last);  
 swap(arr, maxindex, last);  
 }  
 return arr;  
 }  
  
 int[] insertion(int[] arr) {  
 int n = arr.length;  
 for (int i = 0; i < n - 1; i++) {  
 int j = i + 1;  
 while (j > 0) {  
 if (arr[j] >= arr[j - 1]) {  
 break;  
 } else {  
 swap(arr, j, j - 1);  
 j--;  
 }  
 }  
 }  
 return arr;  
 }  
  
 void Cyclic(int[] arr) {  
 int n = arr.length;  
 int i = 0;  
 while (i < arr.length) {  
 int curr = arr[i]  
 - 1;  
 if (arr[i] == arr[curr]) {  
 i++;  
 } else {  
 swap(arr, i, arr[i] - 1);  
 }  
 }  
  
  
 }  
 public void mergesort(int[] arr, int s,int e){  
 if(s>=e){  
 return;  
 }  
 int mid=s+(e-s)/2;  
 mergesort(arr,s,mid);  
 mergesort(arr,mid+1,e);  
 mereger(arr,s,mid,e);  
 }  
  
 private void mereger(int[] arr, int s, int mid, int e) {  
 int idx=s;  
 int idx2=mid+1;  
 int k=0;  
 int[] mix=new int[e-s+1];  
 while(idx<=mid && idx2<=e){  
 if(arr[idx]<=arr[idx2]){  
 mix[k++]=arr[idx++];  
  
 }  
 else{  
 mix[k++]=arr[idx2++];  
 }  
 }  
 while(idx<=mid){  
 mix[k++]=arr[idx++];  
 }  
 while(idx2<=e){  
 mix[k++]=arr[idx2++];  
 }  
  
 for(int i=0;i<mix.length;i++){  
 arr[s+i]=mix[i];  
 }  
 }  
  
 void Quicksort(int[] arr,int low,int hi){  
 if(low==hi){  
 return;  
 }  
 int s=low;  
 int e=hi;  
 int mid=s+(e-s)/2;  
 int pivot=arr[mid];  
 while(s<=e){  
 while(arr[s]<pivot){  
 s++;  
 }  
 while(arr[e]>pivot){  
 e--;  
 }  
  
 if(s<=e){  
 swap(arr,s,e);  
 }  
 }  
  
 Quicksort(arr,low,e);  
 Quicksort(arr,s,hi);  
 }  
  
 int[] Countsort(int[] arr){  
 HashMap<Integer,Integer> map=new HashMap<>();  
 for(int i=0;i<arr.length;i++){  
 map.put(arr[i], map.getOrDefault(arr[i],0)+1);  
 }  
// int[] ans=new int[arr.length];  
 int k=0;  
 for(int i=0;i<=arr.length;i++){  
 if(map.containsKey(i)){  
 while(map.get(i)>0){  
 arr[k++]=i;  
 map.put(i,map.get(i)-1);  
 }  
 }  
 }  
  
 return arr;  
 }  
  
 class Node{  
 int val;  
 Node left;  
 Node right;  
 Node(int val){  
 this.val=val;  
 }  
 }  
 Node root;  
 void BST(int val){  
 root=BSTREE(root,val);  
 }  
 void insert(int[] arr){  
  
 for(int i=0;i<arr.length;i++){  
 BST(arr[i]);  
 }  
 }  
  
 private Node BSTREE(Node node, int val) {  
 if(node==null){  
 Node n=new Node(val);  
 return n;  
 }  
  
 if(val<node.val){  
 node.left=BSTREE(node.left,val);  
 }  
 if(val>node.val){  
 node.right=BSTREE(node.right,val);  
 }  
  
 return node;  
 }  
 public void treesort(ArrayList<Integer> list){  
 inorder(root,list);  
 }  
 public void inorder(Node node,ArrayList<Integer> list){  
 if(node==null){  
 return;  
 }  
  
 inorder(node.left,list);  
 list.add(node.val);  
 inorder(node.right,list);  
 }  
  
  
 private void swap(int[] arr, int first, int second) {  
 int temp = arr[first];  
 arr[first] = arr[second];  
 arr[second] = temp;  
 }  
  
 private int MAXOF(int[] arr, int start, int end) {  
 int max = start;  
 for (int i = 0; i <= end; i++) {  
 if (arr[i] > arr[max]  
 ) {  
 max = i;  
 }  
 }  
 return max;  
  
 }  
  
 void bucketsort(int[] arr,int noofbuckets){  
  
 int globalmax= Arrays.*stream*(arr).max().getAsInt();  
 int globalmin= Arrays.*stream*(arr).min().getAsInt();  
  
 int range=globalmax-globalmin;  
  
 int bucketrange=(int)Math.*ceil*((double)range/noofbuckets);  
 List<Integer>[] Bucket=new LinkedList[noofbuckets];  
 for(int i=0;i<noofbuckets;i++){  
 Bucket[i]=new LinkedList<>();  
 }  
  
  
 for(int num:arr){  
 Bucket[*hash*(num,bucketrange,noofbuckets)].add(num);  
 }  
 for(List<Integer> key: Bucket){  
 Collections.*sort*(key);  
 }  
 int index=0;  
 for(List<Integer> key: Bucket){  
 for(int i:key){  
 arr[index++]=i;  
 }  
 }  
  
 }  
 private static int hash(int num, int hashValue, int numberOfBuckets) {  
 int bucketNumber = num / hashValue;  
 if (bucketNumber == numberOfBuckets)  
 bucketNumber--;  
 return bucketNumber;  
 }  
  
  
  
  
}  
  
  
  
public class Main {  
  
 public static void main(String[] args) throws java.lang.Exception{  
 Sorting b=new Sorting();  
 int[] arr={11,9,21,8,17,19,13,1,2};  
// b.insert(arr);  
// ArrayList<Integer> list=new ArrayList<>();  
// b.treesort(list);  
//  
//// b.treesort(arr,0);  
// System.out.println(list);  
int noofbuckets=(int)Math.*sqrt*(arr.length);  
 b.bucketsort(arr,noofbuckets);  
 System.*out*.println(Arrays.*toString*(arr));  
  
  
  
 }  
  
}