47.Find max repeated character count in a string:

public class Characters

{

public static void main(String[] args) {

String str = "grass is greener on the other side";

int[] freq = new int[str.length()];

char minChar = str.charAt(0), maxChar = str.charAt(0);

int i, j, min, max;

char string[] = str.toCharArray();

for(i = 0; i < string.length; i++) {

freq[i] = 1;

for(j = i+1; j < string.length; j++) {

if(string[i] == string[j] && string[i] != ' ' && string[i] != '0') {

freq[i]++;

string[j] = '0';

min = max = freq[0];

for(i = 0; i <freq.length; i++) {

if(min > freq[i] && freq[i] != '0') {

min = freq[i];

minChar = string[i];

}

if(max < freq[i]) {

max = freq[i];

maxChar = string[i];

}

}

System.out.println("Minimum occurring character: " + minChar);

System.out.println("Maximum occurring character: " + maxChar);

}

}

Output: Minimum occurring character: a

Maximum occurring character: e

Java Program to Implement Quick Sort Algorithm:.

Quicksort algorithm is based on the divide and conquer approach where an array is divided into subarrays by selecting a pivot element.While dividing the array, the pivot element should be positioned in such a way that elements less than pivot are kept on the left side and elements greater than pivot are on the right side.

The same process is continued for both left and right subarrays. Finally, sorted elements are combined to form a sorted array.

class Quicksort(){

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

int pivot = array[high];

int i = (low - 1);

for (int j = low; j < high; j++) {

if (array[j] <= pivot) {

i++;

int temp = array[i];

array[i] = array[j];

array[j] = temp;

int temp = array[i + 1];

array[i + 1] = array[high];

array[high] = temp

return (i + 1);

} static void quickSort(int array[], int low, int high) {

if (low < high) {

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

quickSort(array, low, pivot);

quickSort(array, pi + 1, high);

}

}

class Main {

public static void main(String args[]) {

int[] data = { 8, 7, 2, 1, 0, 9, 6 };

System.out.println("Unsorted Array");

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

int size = data.length;

Quicksort.quickSort(data, 0, size – 1)

System.out.println("Sorted Array in Ascending Order ");

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

}

}

Output:

Unsorted Array

[8, 7, 2, 1, 0, 9, 6]

Sorted Array in Ascending Order

[0, 1, 2, 6, 7, 8, 9]

48.pgm to iterate hashmap using while loop:

import java.util.HashMap;

import java.util.Iterator;

import java.util.Map;

import java.util.Map.Entry;

public class GFG{

public static void main(String[] arguments) {

Map<Integer, String> intType = new HashMap<Integer, String>()

intType.put(1, "First");

intType.put(2, "Second");

intType.put(3, "Third");

intType.put(4, "Fourth");

Iterator<Entry<Integer, String> > new\_Iterator= intType.entrySet().iterator()

while (new\_Iterator.hasNext()) {

Map.Entry<Integer, String> new\_Map= (Map.Entry<Integer, String>)

new\_Iterator.next();

System.out.println(new\_Map.getKey() + " = + new\_Map.getValue());

}

}

output

1= First

2 = Second

3 = Third

4 = Fourth

Implement merge sort:

class MergeSort {

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

int n1 = m - l + 1;

int n2 = r - m;

int L[] = new int[n1];

int R[] = new int[n2];

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];

int i = 0, j = 0;

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++;

while (i < n1) {

arr[k] = L[i];

i++;

while (j < n2) {

arr[k] = R[j];

j++;

k++;}

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

if (l < r) {

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

sort(arr, l, m);

sort(arr, m + 1, r);

merge(arr, l, m, r);

} }

static void printArray(int arr[]) {

int n = arr.length;

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

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

System.out.println();}

public static void main(String args[]){

int arr[] = { 12, 11, 13, 5, 6, 7 };

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

printArray(arr);

MergeSort ob = new MergeSort();

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

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

printArray(arr); }

Output

Given array is

12 11 13 5 6 7

Sorted array is

5 6 7 11 12 13

52.Java Program to find common elements in the two array:

import java.io.\*;

import java.util.\*;

class GFG {

private static void FindCommonElemet(String[] arr1,String[] arr2) {

Set<String> set = new HashSet<>();

for (int i = 0; i < arr1.length; i++) {

for (int j = 0; j < arr2.length; j++) {

if (arr1[i] == arr2[j]) {

set.add(arr1[i]);

break;} }

} for (String i : set) {

System.out.print(i + " ");}

public static void main(String[] args) {

String[] arr1 = { "Article", "in", "Geeks", "for", "Geeks" };

String[] arr2 = { "Geeks", "for", "Geeks" };

System.out.println("Array 1: "+ Arrays.toString(arr1));

System.out.println("Array 2: " + Arrays.toString(arr2));

System.out.print("Common Elements: ");

FindCommonElemet(arr1, arr2); }

}

Output

Array 1: [Article, in, Geeks, for, Geeks]

Array 2: [Geeks, for, Geeks]

Common Elements: Geeks for Geeks

53.pgm to find string is palindrome or not:

class Main {

public static void main(String[] args) {

String str = "Radar", reverseStr = "";

int strLength = str.length();

for (int i = (strLength - 1); i >=0; --i) {

reverseStr = reverseStr + str.charAt(i);

} if (str.toLowerCase().equals(reverseStr.toLowerCase())) {

System.out.println(str + " is a Palindrome String.");

}

else {

System.out.println(str + " is not a Palindrome String.");

}

}

}

Output:

Radar is a Palindrome String.

54.Implement more than one interface in a class:

public interface InterfaceX

{ public int geek();

}

public interface InterfaceY

{ public String geek();

public class Testing implements InterfaceX, InterfaceY {

public String geek()

{ return "hello";

}

55.Iterate linked hash map values:

import java.util.LinkedHashMap;

import java.util.Set;

public class GFG {

public static void main(String a[]) {

LinkedHashMap<String, String> linkedHashMap = new LinkedHashMap<String, String>(); linkedHashMap.put("One", "First element");

linkedHashMap.put("Two", "Second element");

linkedHashMap.put("Three", "Third element");

Set<String> keys = linkedHashMap.keySet();

for (String key : keys) {

System.out.println(key + “ + linkedHashMap.get(key));

}

Output:

One -- First element

Two -- Second element

Three -- Third element

56.Implement pgm for encapsulation:

class EncapsulationDemo{

private int ssn;

private String empName;

private int empAge;

public int getEmpSSN(){

return ssn;

} public String getEmpName(){

return empName;

}public int getEmpAge(){

return empAge;

} public void setEmpAge(int newValue){

empAge = newValue;

}public void setEmpName(String newValue){

empName = newValue;

}public void setEmpSSN(int newValue){

ssn = newValue;

}

}

public class EncapsTest{

public static void main(String args[]){

EncapsulationDemo obj = new EncapsulationDemo();

obj.setEmpName("Mario");

obj.setEmpAge(32);

obj.setEmpSSN(112233);

System.out.println("Employee Name: " + obj.getEmpName());

System.out.println("Employee SSN: " + obj.getEmpSSN());

System.out.println("Employee Age: " + obj.getEmpAge());

}

}

Output:

Employee Name: Mario

Employee SSN: 112233

Employee Age: 32

57 Convert string to array:

public class StringToCharExample2{

public static void main(String args[]){

String s="hello";

for(int i=0; i<s.length();i++){

char c = s.charAt(i);

System.out.println("char at "+i+" index is: "+c);

}

}

Output;

char at 0 index is: h

char at 1 index is: e

char at 2 index is: l

char at 3 index is: l

char at 4 index is: o

59 implement pgm for Abstraction

abstract class MotorBike {

abstract void brake();

}

class SportsBike extends MotorBike {

public void brake() {

System.out.println("SportsBike Brake");

}

}class MountainBike extends MotorBike {

public void brake() {

System.out.println("MountainBike Brake");

}

}class Main {

public static void main(String[] args) {

MountainBike m1 = new MountainBike();

m1.brake();

SportsBike s1 = new SportsBike();

s1.brake();

}

}

output: MountainBike Brake

SportsBike Brake