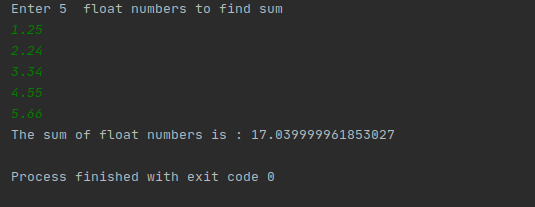
1. Write Java code to define List . Insert 5 floating point numbers in List, and using an iterator, find the sum of the numbers in List.



import java.util.Scanner;

import java.util.\*;

import java.util.Enumeration;

import java.util.Iterator;

import static java.text.NumberFormat.\*;

class q1List

{

public static void main(String[] args) {

double sum=0d;

List<Float> list = new ArrayList<Float>();

Scanner scan=new Scanner(System.*in*);

System.*out*.println("Enter 5 float numbers to find sum");

for (int i = 0; i < 5; i++) {

float num=scan.nextFloat();

list.add(num);

}

Iterator<Float> itr=list.iterator();

while(itr.hasNext())

{

sum=sum+(Float)itr.next();

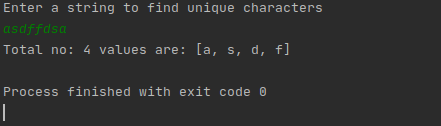
}

System.*out*.println("The sum of float numbers is : "+sum);

}

}

1. Write a method that takes a string and returns the number of unique characters in the string.



// Java code for adding elements in Set

import java.sql.SQLOutput;

import java.util.\*;

public class q2unique\_char {

public static void main(String[] args) {

Set<String> hash\_Set = new HashSet<String>();

//scanner

Scanner scanner = new Scanner(System.*in*);

System.*out*.println("Enter a string to find unique characters");

String string = scanner.nextLine();

String[] chars = string.split("");

for (String s : chars) {

hash\_Set.add(s);

}

System.*out*.println("Total no: " + hash\_Set.size()+ " values are: "+hash\_Set);

}

}

1. Write a method that takes a string and print the number of occurrence of each character in the string.

import java.util.HashMap;

import java.util.Map;

import java.util.Scanner;

public class q3\_no\_of\_occurrennce {

public static void countoccurence(String str)

{

HashMap<Character, Integer> hasmap= new HashMap<>();

char[] chararray= str.toCharArray();

for (char c: chararray)

{

if (hasmap.containsKey(c))

{

hasmap.put(c,hasmap.get(c)+1);

}

else {

hasmap.put(c,1);

}

}

for (Map.Entry entry: hasmap.entrySet())

{

System.*out*.println(entry.getKey()+" "+ entry.getValue());

}

}

public static void main(String[] args) {

Scanner sc= new Scanner(System.*in*);

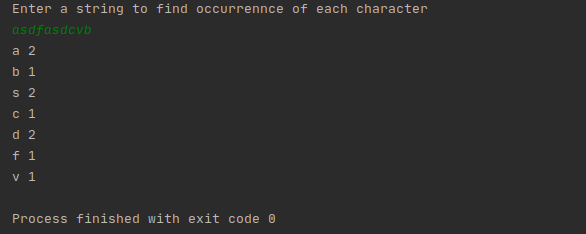
System.*out*.println("Enter a string to find occurrennce of each character");

String str= sc.nextLine();

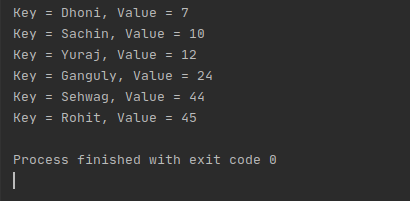
*countoccurence*(str);

}

}



1. Write a program to sort HashMap by value.



// Java program to sort hashMap by values

import java.util.\*;

import java.lang.\*;

public class q4\_sort\_hashmap {

public static HashMap<String, Integer> sortByValue(HashMap<String, Integer> hm)

{

List<Map.Entry<String, Integer> > list =

new LinkedList<Map.Entry<String, Integer> >(hm.entrySet());

Collections.*sort*(list, new Comparator<Map.Entry<String, Integer> >() {

public int compare(Map.Entry<String, Integer> o1,

Map.Entry<String, Integer> o2)

{

return (o1.getValue()).compareTo(o2.getValue());

}

});

// put data from sorted list to hashmap

HashMap<String, Integer> temp = new LinkedHashMap<String, Integer>();

for (Map.Entry<String, Integer> aa : list) {

temp.put(aa.getKey(), aa.getValue());

}

return temp;

}

//main

public static void main(String[] args)

{

HashMap<String, Integer> hm = new HashMap<String, Integer>();

// enter data into hashmap

hm.put("Dhoni", 7);

hm.put("Rohit", 45);

hm.put("Yuraj", 12);

hm.put("Sehwag", 44);

hm.put("Ganguly", 24);

hm.put("Sachin", 10);

Map<String, Integer> hm1 = *sortByValue*(hm);

// print the sorted hashmap

for (Map.Entry<String, Integer> en : hm1.entrySet()) {

System.*out*.println("Key = " + en.getKey() +

", Value = " + en.getValue());

}

}

}

1. Write a program to sort Employee objects based on highest salary using Comparator. Employee class{ Double Age; Double Salary; String Name}

import java.util.ArrayList;

import java.util.Collections;

import java.util.Comparator;

import java.util.List;

class Employee

{

double age;

double salary;

String name;

public Employee(double age, double salary, String name) {

this.age = age;

this.salary = salary;

this.name = name;

}

public double getAge() {

return age;

}

public double getSalary() {

return salary;

}

public String getName() {

return name;

}

public void setAge(double age) {

this.age = age;

}

public void setSalary(double salary) {

this.salary = salary;

}

public void setName(String name) {

this.name = name;

}

}

class SortEmployee implements Comparator<Employee>

{

@Override

public int compare(Employee employee, Employee employee1) {

if(employee.getSalary()< employee1.getSalary())

return 1;

else if (employee.getSalary()> employee1.getSalary())

return -1;

else

return 0;

}

}

public class q5\_comparator {

public static void main(String[] args) {

List<Employee> emp= new ArrayList<Employee>();

Employee e1= new Employee(25,2300,"Shivam");

Employee e2=new Employee(23,2009,"Manish");

Employee e3=new Employee(38,1400,"Vishal");

emp.add(e1);

emp.add(e2);

emp.add(e3);

Collections.*sort*(emp,new SortEmployee());

for(Employee emp1:emp)

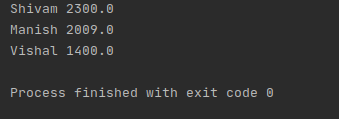
{

System.*out*.println(emp1.getName()+" "+ emp1.getSalary());

}

}

}



1. Write a program to sort the Student objects based on Score , if the score are same then sort on First Name . Class Student{ String Name; Double Score; Double Age

import java.util.ArrayList;

import java.util.Collections;

import java.util.Comparator;

import java.util.List;

class Students

{

double age;

double score;

String name;

public Students(double age, double score, String name) {

this.age = age;

this.score = score;

this.name = name;

}

public double getAge() {

return age;

}

public double getScore() {

return score;

}

public String getName() {

return name;

}

public void setAge(double age) {

this.age = age;

}

public void setScore(double score) {

this.score = score;

}

public void setName(String name) {

this.name = name;

}

}

class SortStudents implements Comparator<Students>

{

@Override

public int compare(Students student, Students student1) {

if(student.getScore()> student1.getScore())

return 1;

else if (student.getScore()<student1.getScore())

return -1;

else

return student.getName().compareTo(student1.getName());

}

}

public class q6\_sort\_again {

public static void main(String[] args) {

List<Students> stud= new ArrayList<Students>();

Students s1= new Students(21,2300,"Shivam");

Students s2=new Students(23,2009,"Vishal");

Students s3=new Students(30,1000,"Sandeep");

Students s4=new Students(24,1000,"Ankit");

Students s5=new Students(25,1000,"Abhinav");

stud.add(s1);

stud.add(s2);

stud.add(s3);

stud.add(s4);

stud.add(s5);

Collections.*sort*(stud,new SortStudents());

for(Students stud1:stud)

{

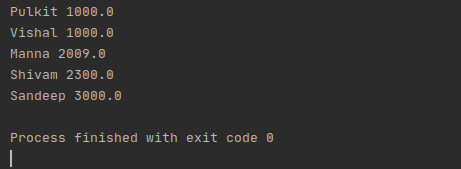
System.*out*.println(stud1.getName()+" "+ stud1.getScore());

}

}

}





1. Print the elements of an array in the decreasing frequency if 2 numbers have same frequency then print the one which came first.

import java.util.\*;

public class q7\_sequence {

public static void main(String[] args) {

int[] arr = new int[]{2,2,2,5,7,4,4,5,6,6,5};

LinkedHashMap<Integer, Integer> hashMap = new LinkedHashMap<>();

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

if (hashMap.containsKey(arr[i])){

hashMap.put(arr[i],hashMap.get(arr[i])+1);

} else {

hashMap.put(arr[i],1);

}

}

List<Map.Entry<Integer,Integer>> list = new LinkedList<Map.Entry<Integer, Integer>>(hashMap.entrySet());

Collections.*sort*(list, new Comparator<Map.Entry<Integer, Integer>>() {

@Override

public int compare(Map.Entry<Integer, Integer> obj, Map.Entry<Integer, Integer> obj1) {

if (obj.getValue() > obj1.getValue()) {

return -1;

}

if (obj1.getValue() > obj.getValue()) {

return 1;

}

return 0;

}

});

LinkedHashMap<Integer, Integer> map = new LinkedHashMap<>();

for(Map.Entry<Integer,Integer> entry:list) {

map.put(entry.getKey(),entry.getValue());

}

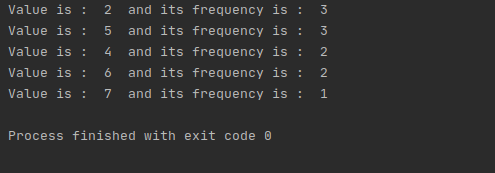
for(Map.Entry<Integer,Integer> entry: map.entrySet()) {

System.*out*.println("Value is : "+entry.getKey() + " and its frequency is : " + entry.getValue());

}

}

}



1. Design a Data Structure SpecialStack that supports all the stack operations like push(), pop(), isEmpty(), isFull() and an additional operation getMin() which should return minimum element from the SpecialStack. (Expected complexity ­ O(1))

class SpecialStack {

private int array[];

private static int *top*;

private int capacity;

int minelement;

public SpecialStack(int size) {

array = new int[size];

capacity = size;

*top* =-1;

}

static boolean isEmpty() {

if (*top* == -1) {

System.*out*.println("SpecialStack is empty");

return true;

}

return false;

}

void isFull() {

if (*top* == capacity - 1) {

System.*out*.println("SpecialStack is full");

}

}

void getMin() {

if (*isEmpty*()) {

System.*out*.println("SpecialStack is empty");

} else {

System.*out*.println("Current Minimum element is : " + minelement);

}

}

void pop() {

if (*isEmpty*()) {

System.*out*.println("SpecialStack is empty\n");

return;

}

System.*out*.print("Top element popped \t");

int top1 = array[*top*--];

if (top1 < minelement) {

System.*out*.println(minelement);

minelement = minelement \*2 -top1;

}

else {

System.*out*.print(top1+"\n");

}

}

void push(int element) {

if (*isEmpty*()) {

minelement = element;

array[++*top*] = element;

System.*out*.println(element + " is inserted");

return;

}

if (element < minelement) {

array[++*top*] = 2\*element - minelement;

minelement = element;

}

else {

array[++*top*] = element;

}

System.*out*.println(element + " is inserted");

}

}

public class q8\_special\_stack{

public static void main(String[] args) {

SpecialStack specialStack = new SpecialStack(10);

specialStack.push(85);

specialStack.push(20);

specialStack.push(57);

specialStack.push(69);

specialStack.pop();

specialStack.pop();

specialStack.getMin();

}

}

