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.Iterator;

import java.util.LinkedList;

import java.util.List;

public class Ques1 {

public static void main(String[] args) {

List<Float> numbers = new LinkedList<>();

numbers.add(3.5f);

numbers.add(4.7f);

numbers.add(2.9f);

numbers.add(3.567f);

numbers.add(23.45f);

float sum = 0;

Iterator<Float> it = numbers.iterator();

while(it.hasNext()){

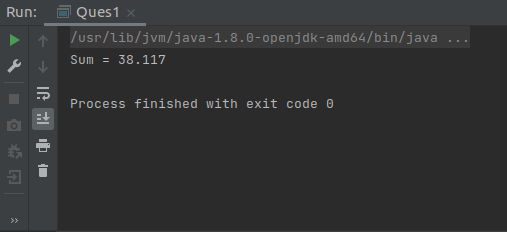
sum = sum + it.next();

}

System.out.println("Sum = " + sum);

}

}



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

import java.util.HashSet;

import java.util.Scanner;

import java.util.Set;

public class Ques2 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter a String");

String str = sc.nextLine();

System.out.println("Number of unique characters in string " + str + " is = "

+ countUniqueChar(str));

}

static int countUniqueChar(String str){

int count = 0;

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

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

if(!set.contains(str.charAt(i))){

set.add(str.charAt(i));

count++;

}

else

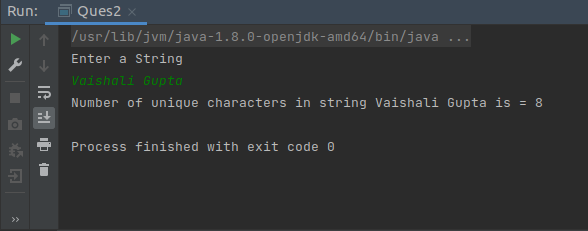
count--;

}

return count;

}

}



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 Ques3 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter A String");

String str = sc.nextLine();

Map<Character,Integer> mp;

mp = countFrequency(str);

mp.forEach((key,val) -> System.out.println("Frequency of Character " + key + " is = " + val));

}

static Map<Character,Integer> countFrequency(String str){

Map<Character,Integer> mp = new HashMap<>();

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

char ch = str.charAt(i);

if(!mp.containsKey(ch))

mp.put(ch,1);

else{

int val = mp.get(ch);

mp.put(ch,++val);

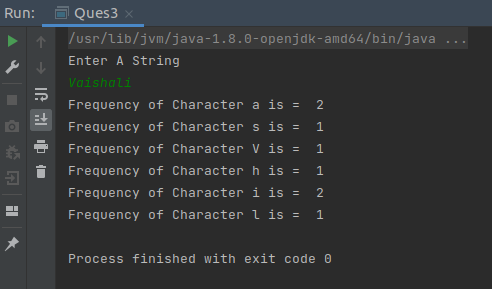
}

}

return mp;

}

}



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;

public class Ques4 {

public static void main(String[] args) {

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

employees.add(new Employee(23,34567.45,"Vaishali"));

employees.add(new Employee(28,25000.50,"Nidhi"));

employees.add(new Employee(19,15000.45,"Sonia"));

employees.add(new Employee(25,22567.45,"Sreyasi"));

Collections.sort(employees, new Comparator<Employee>() {

@Override

public int compare(Employee e1, Employee e2) {

if(e1.getSalary() > e2.getSalary() )

return 1;

else if(e1.getSalary() < e2.getSalary())

return -1;

return 0;

}

});

for(Employee emp:employees){

System.out.println(emp);

}

}

}

class Employee{

private double age;

private double salary;

private String name;

Employee(){}

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

this.age = age;

this.salary = salary;

this.name = name;

}

public double getSalary() {

return salary;

}

@Override

public String toString() {

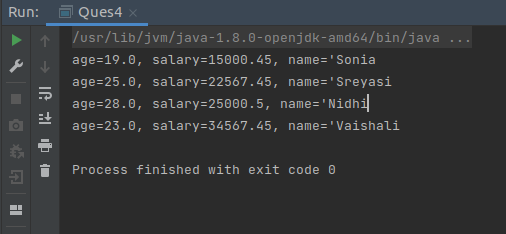
return "age=" + age +

", salary=" + salary +

", name='" + name ;

}

}



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.Collections;

import java.util.Comparator;

import java.util.LinkedList;

import java.util.List;

public class Ques5 {

public static void main(String[] args) {

List<Student> students = new LinkedList<>();

students.add(new Student("Vaishali",99.78,24));

students.add(new Student("Nidhi",90.78,27));

students.add(new Student("Sreyasi",88.78,23));

students.add(new Student("Ritika",99.78,22));

Collections.sort(students, (s1, s2) -> {

if(s1.getScore() > s2.getScore())

return 1;

else if(s1.getScore() < s2.getScore())

return -1;

else

return s1.getName().compareTo(s2.getName());

});

for(Student student:students){

System.out.println(student);

}

}

}

class Student{

private String name;

private double score;

private double age;

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

this.name = name;

this.score = score;

this.age = age;

}

public String getName() {

return name;

}

public double getScore() {

return score;

}

@Override

public String toString() {

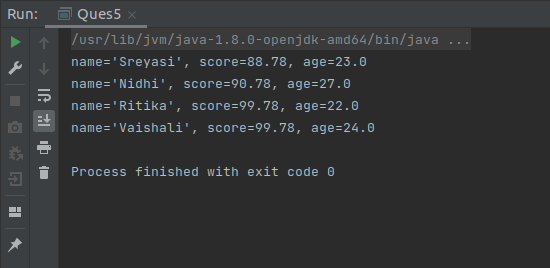
return "name='" + name + '\'' +

", score=" + score +

", age=" + age;

}

}



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.\*;

import java.util.Map.Entry;

public class Ques6 {

public static void sortByFrequencyAndIndex(int[] arr)

{

if (arr == null || arr.length < 2) {

return;

}

// insert frequency of each array element into the map

// and index of its first occurrence in the array

Map<Integer, Data> hm = new HashMap<>();

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

{

if(!hm.containsKey(arr[i])){

hm.put(arr[i],new Data(arr[i],1,i));

}

else{

int index = hm.get(arr[i]).getIndex();

int count = hm.get(arr[i]).getCount();

hm.put(arr[i],new Data(arr[i],++count,index));

}

}

Set<Entry<Integer,Data>> set = hm.entrySet();

List<Entry<Integer,Data>> values = new ArrayList<>(set);

Collections.sort(values, new Comparator<Entry<Integer, Data>>() {

@Override

public int compare(Entry<Integer,Data> m1, Entry<Integer, Data> m2) {

// If two elements have different frequencies, then

// the one which has more frequency should come first

if (m1.getValue().getCount() != m2.getValue().getCount()) {

return m2.getValue().getCount() - m1.getValue().getCount();

}

// If two elements have the same frequencies, then the

// one which has less index should come first

return m1.getValue().getIndex() - m2.getValue().getIndex();

}

});

int k = 0;

for (Entry<Integer,Data> data: values)

{

for (int j = 0; j < data.getValue().getCount(); j++) {

arr[k++] = data.getValue().getElement();

}

}

}

public static void main(String[] args)

{

Scanner sc = new Scanner(System.in);

System.out.println("Enter the size of an array");

int n = sc.nextInt();

int[] arr = new int[n];

System.out.println("Enter the elements of an array");

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

arr[i] = sc.nextInt();

}

System.out.println("Array Before Sorting");

for (Integer i : arr) {

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

}

System.out.println();

sortByFrequencyAndIndex(arr);

System.out.println("Array After Sorting");

for (Integer i : arr) {

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

}

}

}

class Data {

private final int element;

private final int count;

private final int index;

public int getElement() {

return element;

}

public int getCount() {

return count;

}

public int getIndex() {

return index;

}

public Data(int element, int count, int index)

{

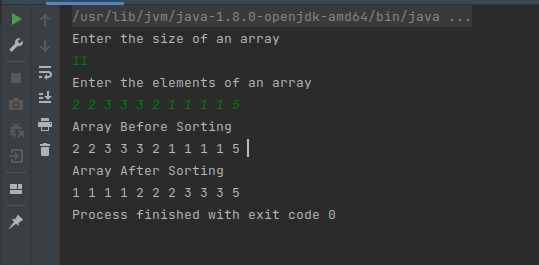
this.element = element;

this.count = count;

this.index = index;

}

}



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 a minimum element from the SpecialStack. (Expected complexity ­ O(1)).

package Ques7;

import java.util.Stack;

class SpecialStack extends Stack {

Stack<Integer> minStack = new Stack<>();

void push(int x){

if(isEmpty()){

super.push(x);

minStack.push(x);

}

else{

super.push(x);

if(x < minStack.peek()){

minStack.push(x);

}

else{

minStack.push(minStack.peek());

}

}

}

public Integer pop(){

int x = (int)super.pop();

minStack.pop();

return x;

}

public int getMin(){

int x = minStack.pop();

return x;

}

public boolean empty(){

return super.empty();

}

}

public class Ques7 {

public static void main(String[] args) {

SpecialStack s = new SpecialStack();

s.push(23);

s.push(19);

s.push(18);

s.push(32);

s.push(20);

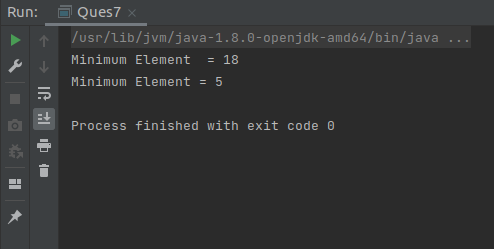
System.out.println("Minimum Element = " + s.getMin());

s.push(5);

System.out.println("Minimum Element = " + s.getMin());

}

}



1. Write a program to format date as example "21-March-2016".

import java.text.SimpleDateFormat;

import java.util.Date;

public class Ques8 {

public static void main(String[] args) {

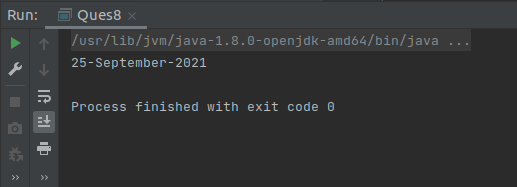
Date today = new Date();

SimpleDateFormat formatter = new SimpleDateFormat("dd-MMMM-YYYY");

System.out.println(formatter.format(today));

}

}



1. Write a program to display times in different country format.

import java.text.DateFormat;

import java.util.Date;

import java.util.Locale;

public class Ques9 {

public static void main(String[] args) {

Date d = new Date();

Locale list[] = DateFormat.getAvailableLocales();

for (Locale aLocale : list) {

DateFormat df = DateFormat.getDateInstance(DateFormat.FULL, aLocale);

DateFormat df1 = DateFormat.getTimeInstance(DateFormat.FULL, aLocale);

System.out.print(df.format(d) + " ");

System.out.println(df1.format(d));

}

}

}

