**Q1. 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.**

**Solution**

**import** java.util.Iterator;

**import** java.util.LinkedList;

**import** java.util.List;

*/\*Question 1\*/*

**public class** SumList {

**public static void** main(String[] args) {

LinkedList<Float> floatList=**new** LinkedList<>();

floatList.add(2.5f);

floatList.add(3.7f);

floatList.add(4.5f);

floatList.add(3.2f);

floatList.add(2.1f);

**float** sum=*calculateSum*(floatList);

System.***out***.println(**"Sum of numbers in list is:---->"**+sum);

}

**private static float** calculateSum(LinkedList<Float> floatList) {

**float** sum=0f;

Iterator<Float>iterator=floatList.iterator();

**while**(iterator.hasNext())

{

sum+=iterator.next();

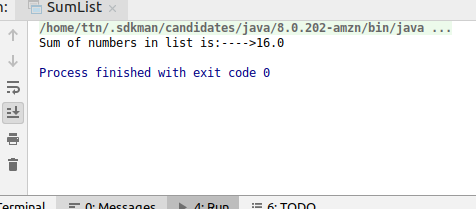
}

**return** sum;

}

}

**Output**



**Q2. Write a method that takes a string and returns the number of unique characters in the string.**

**Solution**

**1)In case here unique means that a character should be counted only once, even if it appears multiple times**

**import java.util.LinkedHashSet;**

**import java.util.Scanner;**

**import java.util.Set;**

***/\*Question-2\*/***

**public class CountUniqueChars {**

**public static int countUniqueChars(String inputString)**

**{**

**Set<Character>uniqueCharacters=new LinkedHashSet<Character>();**

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

**{**

**char currrentChar=inputString.charAt(i);**

**uniqueCharacters.add(currrentChar);**

**}**

**return uniqueCharacters.size();**

**}**

**public static void main(String[] args) {**

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

**System.*out*.println("Enter a string");**

**String inputString=in.nextLine();**

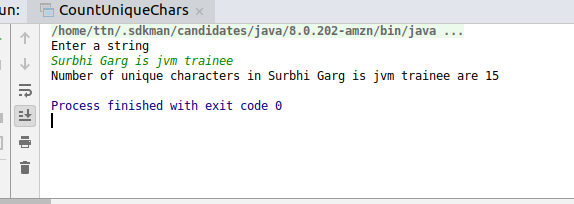
**int countUniqueCharacters=*countUniqueChars*(inputString);**

**System.*out*.println("Number of unique characters in "+inputString+" are "+countUniqueCharacters);**

**}**

**}**

**Output**



**2)In case here unique means characters that are occurring only once in string**

**import java.util.\*;**

**public class CountUniqueChars2 {**

**public static int countUniqueChars(String inputString) {**

**List<Character> uniqueCharList = new LinkedList<Character>();**

**inputString=inputString.toLowerCase();**

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

**char currrentChar = Character.*toLowerCase*(inputString.charAt(i));**

**int fistIndex=inputString.indexOf(currrentChar);**

**int lastIndex=inputString.lastIndexOf(currrentChar);**

**if(lastIndex-fistIndex==0)**

**uniqueCharList.add(currrentChar);**

**}**

**return uniqueCharList.size();**

**}**

**public static void main(String[] args) {**

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

**System.*out*.println("Enter a string");**

**String inputString = in.nextLine();**

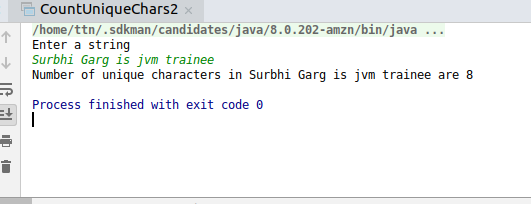
**int countUniqueCharacters = *countUniqueChars*(inputString);**

**System.*out*.println("Number of unique characters in " + inputString + " are " + countUniqueCharacters);**

**}**

**}**

**Output**



**Q3. Write a method that takes a string and print the number of occurrence of each character characters in the string**

**Solution**

**import java.util.HashMap;**

**import java.util.Scanner;**

**public class CountCharOccurrence {**

**public static HashMap<Character, Integer> countCharacterOccurrenceInString(String inputString) {**

**HashMap<Character, Integer> characterOccurrenceMap = new HashMap<Character, Integer>();**

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

**char currentChar = Character.*toLowerCase*(inputString.charAt(i));**

**int currentValuePresentInMap = characterOccurrenceMap.getOrDefault(currentChar, 0);**

**characterOccurrenceMap.put(currentChar, currentValuePresentInMap + 1);**

**}**

**return characterOccurrenceMap;**

**}**

**public static void main(String[] args) {**

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

**System.*out*.println("Enter a string");**

**String inputString=in.nextLine();**

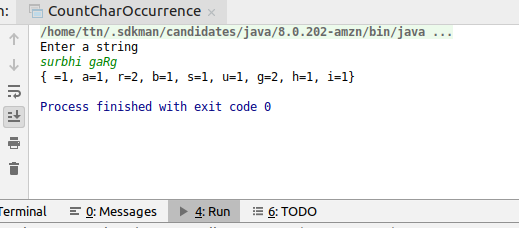
**HashMap<Character,Integer>characterIntegerHashMap=*countCharacterOccurrenceInString*(inputString);**

**System.*out*.println(characterIntegerHashMap);**

**}**

**}**

**Output**



**Q4. Write a program to sort Employee objects based on highest salary using Comparator. Employee class{ Double Age; Double Salary; String Name**

**Solution**

**import java.util.\*;**

**class Employee**

**{**

**String name;**

**double age;**

**double salary;**

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

**this.name = name;**

**this.age = age;**

**this.salary = salary;**

**}**

**@Override**

**public String toString() {**

**return "Employee{" +**

**"name='" + name + '\'' +**

**", age=" + age +**

**", salary=" + salary +**

**'}';**

**}**

**}**

**public class EmployeeSort {**

**public static void main(String[] args) {**

**List<Employee> employeeList= Arrays.*asList*(**

**new Employee("Surbhi",23,20000),**

**new Employee("Vagish",23,15000),**

**new Employee("Rishabh",23,18000),**

**new Employee("Yukti",23,14000));**

**System.*out*.println("------------Employee list before sorting-----");**

**System.*out*.println(employeeList);**

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

**@Override**

**public int compare(Employee o1, Employee o2) {**

**return (int) (o2.salary-o1.salary);**

**}**

**});**

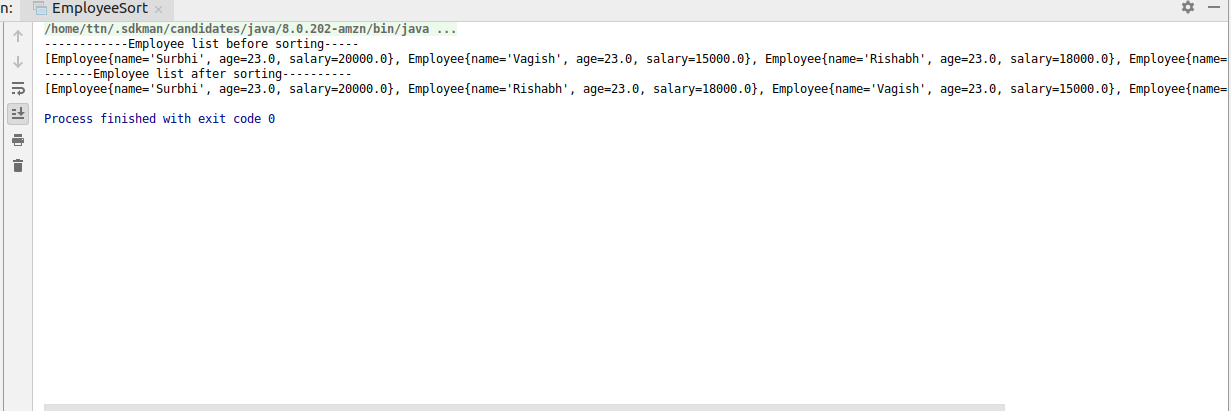
**System.*out*.println("-------Employee list after sorting----------");**

**System.*out*.println(employeeList);**

**}**

**}**

**Output**



**Q5. 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**

**Solution**

**Using Comparable**

**import java.util.Arrays;**

**import java.util.Collections;**

**import java.util.List;**

**class Student implements Comparable**

**{**

**String name;**

**double age;**

**double score;**

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

**this.name = name;**

**this.age = age;**

**this.score = score;**

**}**

**@Override**

**public int compareTo(Object o) {**

**Student student=null;**

**if(o instanceof Student) {**

**student = (Student) o;**

**}**

**if(student!=null)**

**if (this.score == student.score)**

**return name.compareTo(student.name);**

**return (int) (this.score-student.score);**

**}**

**@Override**

**public String toString() {**

**return "Employee{" +**

**"name='" + name + '\'' +**

**", age=" + age +**

**", score=" + score +**

**'}';**

**}**

**}**

**public class StudentSort {**

**public static void main(String[] args) {**

**List<Student> studentList= Arrays.*asList*(**

**new Student("Vagish",23,20),**

**new Student("Surbhi",23,20),**

**new Student("Rishabh",23,18),**

**new Student("Yukti",23,14));**

**System.*out*.println("------------Student list before sorting-----");**

**for(Student student:studentList)**

**System.*out*.println(student);**

**Collections.*sort*(studentList);**

**System.*out*.println("-------Student list after sorting----------");**

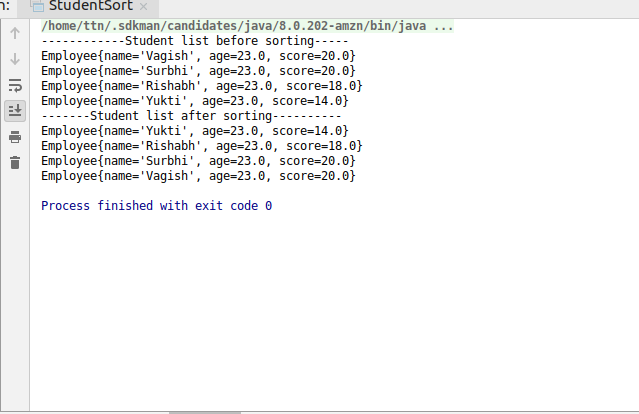
**for(Student student:studentList)**

**System.*out*.println(student);**

**}**

**}**

**Output**



**Using Comparator**

**import java.util.Arrays;**

**import** java.util.Collections;

**import** java.util.Comparator;

**import** java.util.List;

**class** Student1

{

String **name**;

**double age**;

**double score**;

**public** Student1(String name, **double** age, **double** score) {

**this**.**name** = name;

**this**.**age** = age;

**this**.**score** = score;

}

@Override

**public** String toString() {

**return "Employee{"** +

**"name='"** + **name** + **'\''** +

**", age="** + **age** +

**", score="** + **score** +

**'}'**;

}

}

**public class** StudentSortComparator {

**public static void** main(String[] args) {

List<Student> studentList= Arrays.*asList*(

**new** Student(**"Vagish"**,23,20),

**new** Student(**"Surbhi"**,23,20),

**new** Student(**"Rishabh"**,23,18),

**new** Student(**"Yukti"**,23,14));

System.***out***.println(**"------------Student list before sorting-----"**);

**for**(Student student:studentList)

System.***out***.println(student);

Collections.*sort*(studentList, **new** Comparator<Student>() {

@Override

**public int** compare(Student o1, Student o2) {

**if** (o1.**score** == o2.**score**)

**return** o1.**name**.compareTo(o2.**name**);

**return** (**int**) (o1.**score**-o2.**score**);

}

});

System.***out***.println(**"-------Student list after sorting----------"**);

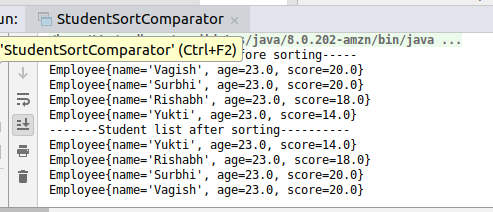
**for**(Student student:studentList)

System.***out***.println(student);

}

}

**Output**



**Q6. Print the elements of an array in the decreasing frequency if 2 numbers have same frequency then print the one which came first**

**Solution**

**import java.util.\*;**

**class MyComparator implements Comparator<Map.Entry<Integer,Integer>>**

**{**

**@Override**

**public int compare(Map.Entry<Integer, Integer> o1, Map.Entry<Integer, Integer> o2) {**

**if(o1.getValue()==o2.getValue())**

**{**

***//o1 must have come in list before,therefore,o2 must come in list after o1 when there frequencies are same***

***//compareTo positive value-->second is greater than first.***

**return 1;**

**}**

**return o2.getValue()-o1.getValue();**

**}**

**}**

**public class ArrayDecreasingFrequency {**

**static Scanner *in*;**

**public static void main(String[] args) {**

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

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

**int eleCount=*in*.nextInt();**

**int[] inputArray=new int[eleCount];**

***readArray*(inputArray,eleCount);**

**LinkedHashMap<Integer,Integer>elementOccurrenceMap=*countOccurrenceOfEachElement*(inputArray);**

**System.*out*.println("Occurrence of each element in array is as follows:");**

**System.*out*.println("format (element=occurrence)");**

**System.*out*.println(elementOccurrenceMap);**

**List<Map.Entry<Integer,Integer>> elementOcurrenceList=new ArrayList(elementOccurrenceMap.entrySet());**

**Collections.*sort*(elementOcurrenceList,new MyComparator());**

**System.*out*.println("Elements sorted in descending order of frequency...and in case frequency is same,sorted in insertion order");**

**System.*out*.println("format (element=occurrence)");**

**System.*out*.println(elementOcurrenceList);**

**System.*out*.println("Printing array elements according to given scenario:");**

**for(Map.Entry<Integer,Integer>curEntry:elementOcurrenceList)**

**{**

**for(int i=0;i<curEntry.getValue();i++)**

**{**

**System.*out*.println(curEntry.getKey());**

**}**

**}**

**}**

**private static LinkedHashMap<Integer, Integer> countOccurrenceOfEachElement(int[] inputArray) {**

**LinkedHashMap<Integer,Integer>elementOccurrenceMap=new LinkedHashMap<Integer, Integer>();**

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

**{**

**int curArrayElement=inputArray[i];**

**int valuePresentInMap=elementOccurrenceMap.getOrDefault(curArrayElement,0);**

**elementOccurrenceMap.put(curArrayElement,valuePresentInMap+1);**

**}**

**return elementOccurrenceMap;**

**}**

**private static void readArray(int[] inputArray, int eleCount) {**

**for(int i=0;i<eleCount;i++)**

**{**

**System.*out*.println("Enter "+(i+1)+" element");**

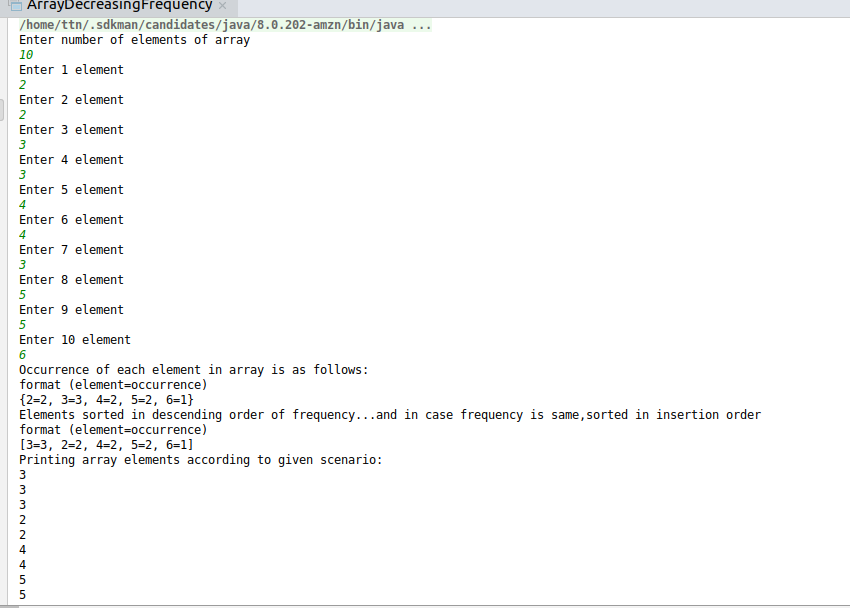
**inputArray[i]=*in*.nextInt();**

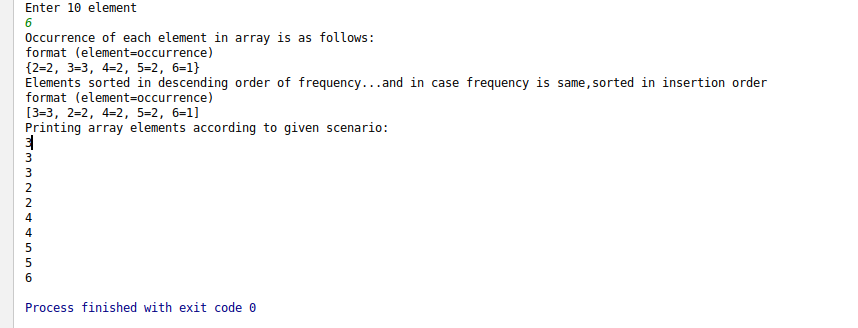
**}**

**}**

**}**

**Output**





**Q7. 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)**

**Solution**

**import java.util.Stack;**

**public class SpecialStack extends Stack<Integer> {**

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

**public void push(int element)**

**{**

***//push element in main stack***

***//if stack is empty push the element in minStack too***

**if(isEmpty())**

**{**

**super.push(element);**

**minStack.push(element);**

**}**

***/\****

***\* Check with current minimum element***

***\* 1. if currentMinium < new item to be inserted-->push new item to minimum stack***

***\* 2. if currentMinimum>new item to be inserted-->push current minimum again***

***\* (so that when we pop element,current minimum will still be maintained,unless and util***

***\* we actually pop the currentMinimum element. Any other element popping will just remove one occurrence***

***\* of current minimum value from the stack.)***

***\* \*/***

**else**

**{**

**super.push(element);**

**int minEle=minStack.peek();*//obtain the current minimum element present in stack***

**if(element<minEle)**

**minStack.push(element);**

**else**

**minStack.push(minEle);**

**}**

**}**

**public Integer pop()**

**{**

**Integer poppedElement=null;**

***//to avoid EmptyStackException***

**if(!isEmpty())**

**poppedElement=super.pop();**

**if(minStack.isEmpty()==false)**

**minStack.pop();**

**return poppedElement;**

**}**

**public Integer getMin()**

**{**

***//if original stack remains empty, there will be no minimum element too***

**if(isEmpty())**

**return null;**

**return minStack.peek();**

**}**

**}**

**public class SpecialStackDriver {**

**public static void main(String[] args) {**

**SpecialStack specialStack=new SpecialStack();**

**specialStack.push(15);**

**specialStack.push(11);**

**specialStack.push(18);**

**if(specialStack.getMin()!=null)**

**System.*out*.println("Minimum element of stack is: "+specialStack.getMin());**

**specialStack.pop();**

**specialStack.pop();**

**specialStack.push(7);**

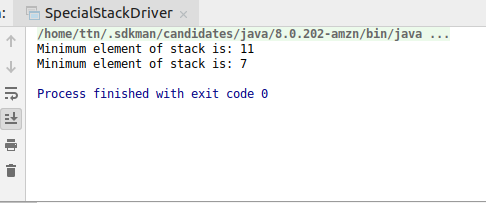
**specialStack.push(10);**

**System.*out*.println("Minimum element of stack is: "+specialStack.getMin());**

**}**

**}**

**Output**



**Q8. Write a program to format date as example "21-March-2016"**

**Solution**

**import java.text.DateFormat;**

**import java.text.SimpleDateFormat;**

**import java.util.Date;**

**public class DateFormatDemo {**

**public static void main(String[] args) {**

**DateFormat dateFormat=new SimpleDateFormat("dd-MMMMM-yyyy");**

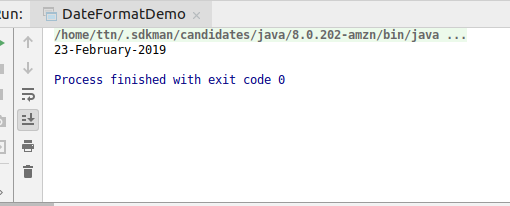
**Date date=new Date();**

**System.*out*.println(dateFormat.format(date));**

**}**

**}**

**Output**



**Q9. Write a program to display times in different country format.**

**Solution**

**import java.text.DateFormat;**

**import java.util.Date;**

**import java.util.Locale;**

**public class CountryWiseDateFormatter {**

**public static void main(String[] args) throws Exception {**

**Date d1 = new Date();**

**System.*out*.println("today is "+ d1.toString());**

**Locale localeItalian = new Locale("it","ch");**

**Locale localeCanada=new Locale(Locale.*CANADA\_FRENCH*.getLanguage(),Locale.*CANADA*.getCountry());**

**DateFormat dateFormat = DateFormat.*getDateInstance* (DateFormat.*FULL*, localeItalian);**

**System.*out*.println("today is in Italian Language in Switzerland Format : "+ dateFormat.format(d1));**

**DateFormat dateFormat2 = DateFormat.*getDateInstance* (DateFormat.*FULL*, localeCanada);**

**System.*out*.println("today is in French Language in Canada Format : "+ dateFormat2.format(d1));**

**}**

**}**

**Output**

