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.

Code:-

**import** java.util.ArrayList;

**import** java.util.Iterator;

**import** java.util.List;

**public class** Que1 {

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

{

**float** sum = 0;

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

list.add(2.5f);

list.add(2.6f);

list.add(2.7f);

list.add(2.8f);

list.add(2.9f);

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

**for** (**float** f: list) {

sum = sum + f;

}

System.***out***.println(**"sum of list of float = "** +sum);

}

}

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

Code:-

**public class** Que2 {

**static final int *chars***= 256;

**static void** printDistinct(String str)

{

**int**[] count = **new int**[***chars***];

**int** i;

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

**if**(str.charAt(i)!=**' '**)

count[(**int**)str.charAt(i)]++;

**int** n = i;

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

**if** (count[(**int**)str.charAt(i)] == 1)

System.***out***.print(str.charAt(i));

}

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

{

String str = **"ToTheNew"**;

*printDistinct*(str);

}

}

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

**import** java.io.\*;

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

**class** Que3 {

**static void** characterCount(String inputString)

{

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

**char**[] strArray = inputString.toCharArray();

**for** (**char** c : strArray) {

**if** (charCountMap.containsKey(c)) {

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

}

**else** {

charCountMap.put(c, 1);

}

}

**for** (Map.Entry entry : charCountMap.entrySet()) {

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

}

}

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

{

String str = **"AshishAyush"**;

*characterCount*(str);

}

}

4. Write a program to sort HashMap by value.

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

**import** java.lang.\*;

**public class** Que4 {

**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());

}

});

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

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

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

}

**return** temp;

}

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

{

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

hm.put(**"Ashish"**, 98);

hm.put(**"Dev"**, 85);

hm.put(**"Dhiraj"**, 91);

hm.put(**"Jiva"**, 95);

hm.put(**"nitin"**, 79);

hm.put(**"Nikhil"**, 80);

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

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

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

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

}

}

}

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

Code:-

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

**class Employee{**

**String name;**

**int id;**

**int salary;**

**public Employee(int salary,String name,int id){**

**this.id=id;**

**this.name=name;**

**this.salary=salary;**

**}**

**}**

**class main{**

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

**Employee e1 = new Employee(35000,"Tushar", 4156);**

**Employee e2 = new Employee(50000,"Yadav", 4186 );**

**Employee e3 = new Employee(40000,"ravi", 4158 );**

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

**employees.add(e2);**

**employees.add(e3);**

**employees.add(e1);**

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

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

**@Override**

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

**return o1.salary - o2.salary;**

**}**

**});**

**for (Employee i: employees)**

**System.*out*.println("id: "+i.id+" ,name: "+i.name+" ,salary:"+i.salary);**

**}**

**}**

6.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

Code:-

**import java.util.ArrayList;**

**import java.util.Comparator;**

**import java.util.List;**

**public class Que6 {**

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

**List<Student> list = new ArrayList<Student>();**

**list.add(new Student("Akash",50,21));**

**list.add(new Student("Rajesh",41,22));**

**list.add(new Student("Abhimanyu",41,18));**

**list.add(new Student("Ram",29,17));**

**list.sort(Comparator.*comparing*(Student::getScore).thenComparing(Student::getName));**

**System.*out*.println("Sorted list entries: ");**

**for(Student s:list){**

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

**}**

**}**

**}**

**class Student{**

**private String name;**

**private double score;**

**private double age;**

**public Student(String n, double s, double a){**

**this.name = n;**

**this.score = s;**

**this.age = a;**

**}**

**public String getName() {**

**return name;**

**}**

**public void setName(String name) {**

**this.name = name;**

**}**

**public double getScore() {**

**return score;**

**}**

**public void setScore(double score) {**

**this.score = score;**

**}**

**public void setAge(double age){**

**this.age=age;**

**}**

**public double getAge(){**

**return age;**

**}**

**public String toString(){**

**return String.*format*("%-15.30s %-15.30s %-15.30s","Name: "+this.name,"Score: "+this.score,"Age: "+this.age);**

**}**

**}**

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

**Code:-**

**import java.io.IOException;**

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

**class Que {**

**public static StringBuffer sortByFrequency(int arr1[], int l1) {**

**Map<Integer, Integer> countMap = *getCountMap*(arr1, l1);**

**StringBuffer result = new StringBuffer();**

**countMap.entrySet().stream()**

**.sorted(Map.Entry.<Integer, Integer> *comparingByValue*().reversed())**

**.forEach(e -> {**

**int key = e.getKey();**

**int val = e.getValue();**

**for (int i = 0; i < val; i++) {**

**result.append(key + " ");**

**}**

**});**

**return result;**

**}**

**private static Map<Integer, Integer> getCountMap(int[] arr, int l1) {**

**Map<Integer, Integer> countMap = new LinkedHashMap<>();**

**for (int i = 0; i < l1; i++) {**

**if (countMap.containsKey(arr[i])) {**

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

**} else {**

**countMap.put(arr[i], 1);**

**}**

**}**

**return countMap;**

**}**

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

**int a[] = { 2, 5, 2, 6, -1, 9999999, 5, 8, 8, 8 };**

**System.*out*.println(*sortByFrequency*(a, a.length));**

**}**

**}**

**8.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))**

**Code:-**

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

**class MyStack {**

**Stack<Integer> s;**

**Integer minEle;**

**MyStack() {**

**s = new Stack<Integer>();**

**}**

**void getMin() {**

**if (s.isEmpty())**

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

**else**

**System.*out*.println("Minimum Element in the " + " stack is: " + minEle);**

**}**

**void peek() {**

**if (s.isEmpty()) {**

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

**return;**

**}**

**Integer t = s.peek(); *// Top element.***

**System.*out*.print("Top Most Element is: ");**

**if (t < minEle)**

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

**else**

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

**}**

**void pop() {**

**if (s.isEmpty()) {**

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

**return;**

**}**

**System.*out*.print("Top Most Element Removed: ");**

**Integer t = s.pop();**

**if (t < minEle) {**

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

**}**

**}**

**}**