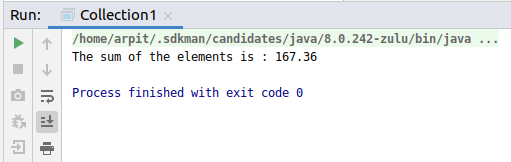
Collections

Exercise

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:

**package** Collection;

**import** java.util.ArrayList;

**import** java.util.ListIterator;

**public class** Collection1

{

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

**float** result = (**float**) 0.0;

ArrayList<Float> list1 = **new** ArrayList<>();

list1.add((**float**) 13.43);

list1.add((**float**) 23.34);

list1.add((**float**) 10.93);

list1.add((**float**) 29.33);

list1.add((**float**) 90.33);

ListIterator<Float> iterator = list1.listIterator();

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

result = result + iterator.next();

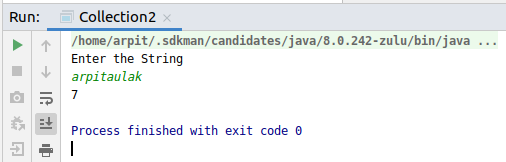
}

System.***out***.println(**"The sum of the elements is : "**+result);

}

}

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



Code:

**package** Collection;

**import** java.util.HashMap;

**import** java.util.Map;

**import** java.util.Scanner;

**public class** Collection2 {

**static void** countOccurences(String inputString)

{

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

**int** count = 0;

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

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

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

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

}

**else** {

occurenses.put(c, 1);

}

}

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

**if**(entry.getValue().equals(1)){

count++;

}

}

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

}

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

{

System.***out***.println(**"Enter the String"**);

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

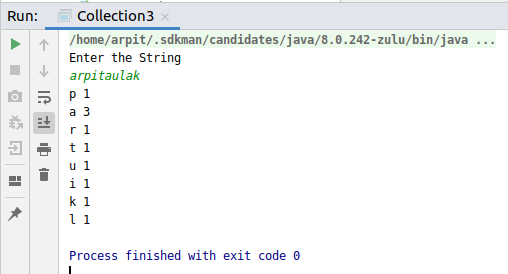
String str = sc.nextLine();

*countOccurences*(str);

}

}

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



Code:

**package** Collection;

**import** java.util.HashMap;

**import** java.util.Map;

**import** java.util.Scanner;

**public class** Collection3 {

**static void** countOccurences(String inputString)

{

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

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

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

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

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

}

**else** {

occurenses.put(c, 1);

}

}

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

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

}

}

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

{

System.***out***.println(**"Enter the String"**);

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

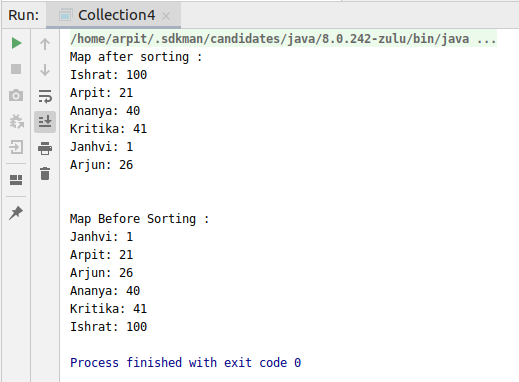
String str = sc.nextLine();

*countOccurences*(str);

}

}

1. Write a program to sort HashMap by value.



Code:

**package** Collection;

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

**public class** Collection4 {

**public static**

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

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

*map*.put(**"Arpit"**, 21);

*map*.put(**"Ishrat"**, 100);

*map*.put(**"Ananya"**, 40);

*map*.put(**"Janhvi"**, 1);

*map*.put(**"Kritika"**, 41);

*map*.put(**"Arjun"**, 26);

System.***out***.println(**"Map after sorting :"**);

*printMap*(*map*);

System.***out***.println(**"\n"**);

*sortMap*(*map*);

System.***out***.println(**"Map Before Sorting :"**);

*printMap*(*sortMap*(*map*));

}

**private static** HashMap<String, Integer> sortMap(Map<String, Integer> map) {

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

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

@Override

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

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

}

});

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

**for** (Map.Entry<String, Integer> map1 : list){

sortedMap.put(map1.getKey(), map1.getValue());

}

**return** sortedMap;

}

**private static void** printMap(Map map) {

Iterator<String> keyIterator = map.keySet().iterator();

Iterator<Integer> valueIterator = map.values().iterator();

**while** (keyIterator.hasNext()&&valueIterator.hasNext()){

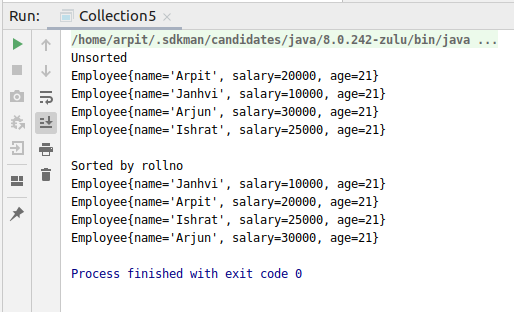
System.***out***.println(keyIterator.next()+**": "**+valueIterator.next());

}

}

}

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



Code:

**package** Collection;

**import** java.util.ArrayList;

**import** java.util.Collections;

**import** java.util.Comparator;

**class** Employee{

String **name**;

**int salary**;

**int age**;

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

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

**this**.**salary** = salary;

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

}

@Override

**public** String toString() {

**return "Employee{"** +

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

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

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

**'}'**;

}

}

**class** SortBySalary **implements** Comparator<Employee>{

@Override

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

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

}

}

**public class** Collection5 {

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

ArrayList<Employee> list = **new** ArrayList<Employee>();

list.add(**new** Employee(**"Arpit"**, 20000, 21));

list.add(**new** Employee(**"Janhvi"**, 10000, 21));

list.add(**new** Employee(**"Arjun"**, 30000, 21));

list.add(**new** Employee(**"Ishrat"**, 25000, 21));

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

**for** (**int** i=0; i<list.size(); i++)

System.***out***.println(list.get(i));

Collections.*sort*(list, **new** SortBySalary());

System.***out***.println(**"\nSorted by rollno"**);

**for** (**int** i=0; i<list.size(); i++)

System.***out***.println(list.get(i));

}

}

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

Code:

**package** Collection;

**import** java.util.ArrayList;

**import** java.util.Collections;

**import** java.util.Comparator;

**class** Student{

String **Name**;

**int Score**;

**int age**;

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

**Name** = name;

**Score** = score;

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

}

@Override

**public** String toString() {

**return "Student{"** +

**"Name='"** + **Name** + **'\''** +

**", Score="** + **Score** +

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

**'}'**;

}

}

**class** Sort **implements** Comparator<Student>{

@Override

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

**if** (o1.**Score** == o2.**Score**){

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

}**else**

**return** o1.**Score** - o2.**Score**;

}

}

**public class** Collection6 {

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

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

list.add(**new** Student(**"Arpit"**, 95, 21));

list.add(**new** Student(**"Janhvi"**, 80, 21));

list.add(**new** Student(**"Arjun"**, 90, 21));

list.add(**new** Student(**"Ishrat"**, 90, 21));

list.add(**new** Student(**"Chirag"**, 90, 21));

System.***out***.println(**"Student List: "**);

**for** (**int** i = 0; i < list.size(); i++) {

System.***out***.println(list.get(i));

}

System.***out***.println(**"\n"**);

Collections.*sort*(list, **new** Sort());

System.***out***.println(**"Student List after sorting: "**);

**for** (**int** i = 0; i < list.size(); i++) {

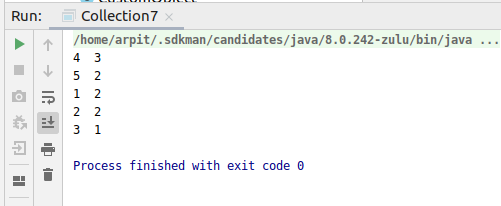
System.***out***.println(list.get(i));

}

}

}

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



Code:

**package** Collection;

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

**public class** Collection7 {

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

**int** a[]={5,5,1,1,2,2,3,4,4,4};

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

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

{

**if**(map.containsKey(a[i]))

{

map.put(a[i],map.get(a[i])+1);

}

**else**

{

map.put(a[i],1);

}

}

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

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

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

Map.Entry<Integer, Integer> o2)

{

**return** (o1.getValue()).compareTo(o2.getValue())\*-1;

}

});

HashMap<Integer,Integer> map2 = **new** LinkedHashMap<Integer, Integer>();

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

{

map2.put(aa.getKey(),aa.getValue());

}

**for**(Map.Entry<Integer, Integer> traverse : map2.entrySet())

{

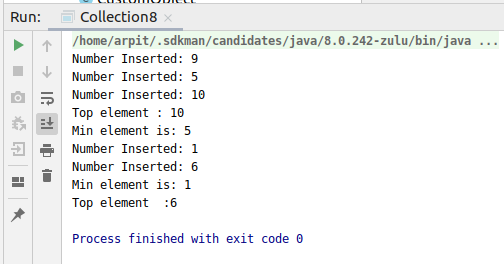
System.***out***.println(traverse.getKey()+ **" "**+traverse.getValue());

}

}

}

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))



Code:

**package** Collection;

**import** java.util.EmptyStackException;

**class** SpecialStack {

**private int arr**[];

**private int size**;

**private int index** = 0;

**private int minValue**;

SpecialStack() {

}

SpecialStack(**int** size) {

**this**.**size** = size;

**arr** = **new int**[size];

}

**public void** push(**int** element) {

**if** (isFull()) {

**throw new** StackOverflowError(**"SpecialStack is full"**);

}

**if** (isEmpty())

{

**minValue** = element;

**arr**[**index**] = element;

**index**++;

System.***out***.println(**"Number Inserted: "** + element);

**return**;

}

**if** (element < **minValue**)

{

**minValue** = element;

}

**arr**[**index**] = element;

**index**++;

System.***out***.println(**"Number Inserted: "** + element);

}

**public int** pop() {

**if** (isEmpty()) {

**throw new** EmptyStackException();

}

**return arr**[--**index**];

}

**public boolean** isEmpty() {

**if** (**index** == 0) {

**return true**;

}

**return false**;

}

**public boolean** isFull() {

**if** (**index** == **size**) {

**return true**;

}

**return false**;

}

**public int** getMin() {

**return minValue**;

}

}

**public class** Collection8 {

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

SpecialStack specialStack = **new** SpecialStack(5);

specialStack.push(9);

specialStack.push(5);

specialStack.push(10);

System.***out***.println(**"Top element : "**+specialStack.pop());

System.***out***.println(**"Min element is: "**+specialStack.getMin());

specialStack.push(1);

specialStack.push(6);

System.***out***.println(**"Min element is: "**+specialStack.getMin());

System.***out***.println(**"Top element :"**+specialStack.pop());

}

}