**COLLECTIONS ASSIGNMENT**

**RAGHAV GUPTA**

-------------------------------------------------

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** collectionsExercise;

*/\**

*AIM -*

*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.sql.SQLOutput;

**import** java.util.ArrayList;

**import** java.util.Iterator;

**import** java.util.List;

**public class** Question1{

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

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

list.add(1.112f);

list.add(2.33f);

list.add(4.55f);

list.add(6.7778f);

list.add(9.7576f);

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

Float sum = 0f;

**while**(it.hasNext()){

Float val = it.next();

*// System.out.println(val);*

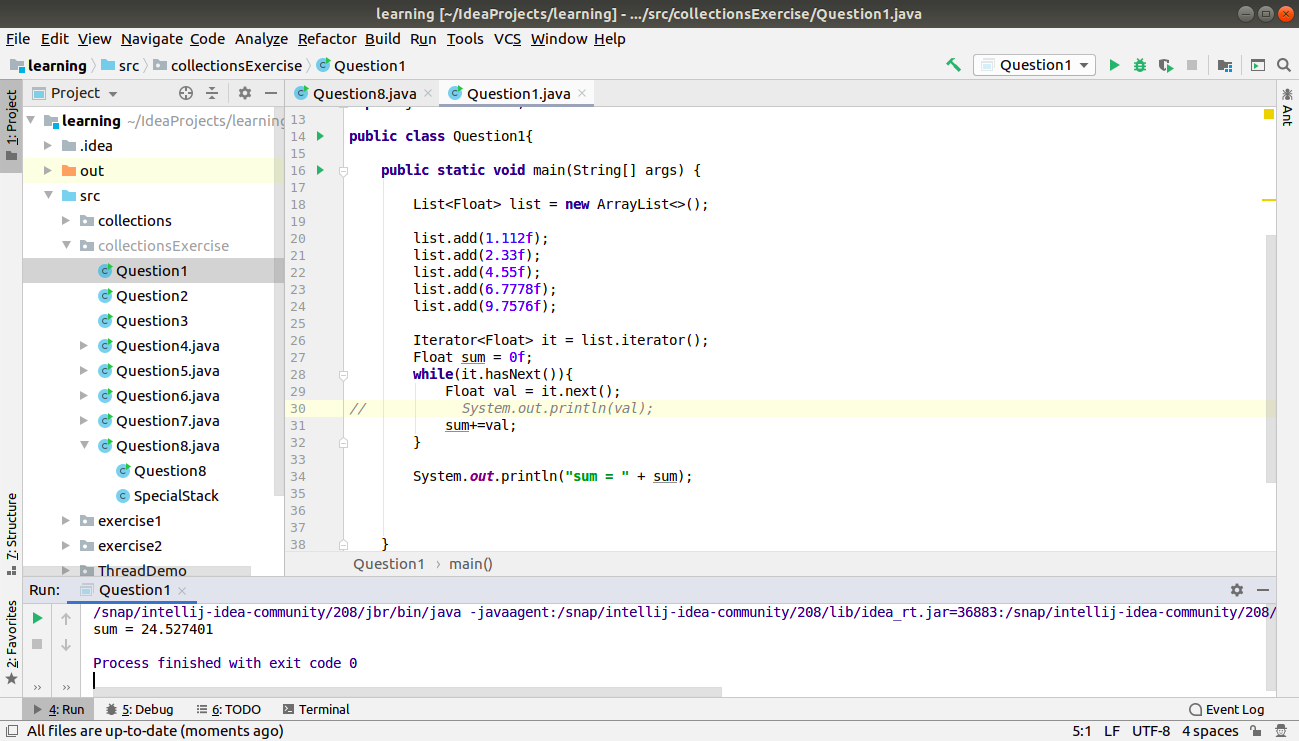
sum+=val;

}

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

}

}



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

**package** collectionsExercise;

*/\**

*Aim:*

*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** Question2 {

**public static int** getUniqueChars(String s){

*// count the unique characters from a string and return it.*

*// because we want unique characers, so use set.*

Set<Character> charSet = **new** HashSet<>();

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

charSet.add(s.charAt(i));

}

**return** charSet.size();

}

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

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

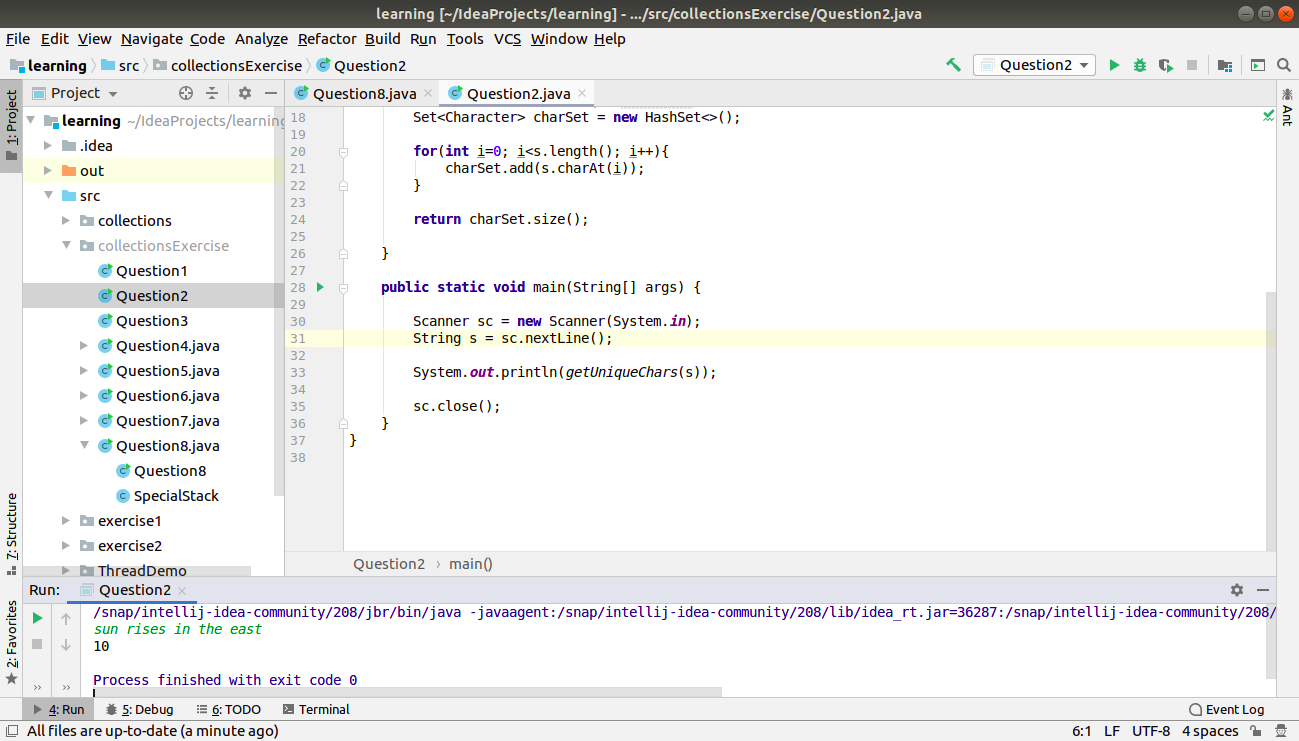
String s = sc.nextLine();

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

sc.close();

}

}



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

**package** collectionsExercise;

*/\**

*Aim: Write a method that takes a string and print the number of occurrence of each character*

*characters in the string.*

*\*/*

**import** java.util.HashMap;

**import** java.util.Map;

**import** java.util.Scanner;

**public class** Question3 {

**public static void** uniqueCharCount(String s){

*// unique characters as the key, count as value.*

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

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

Character ch = s.charAt(i);

*// if char exists, increment the count*

**if**(charMap.containsKey(ch)){

Integer value = charMap.get(ch);

value++;

charMap.replace(ch, value);

}

*// else insert into the map*

**else**

charMap.put(s.charAt(i), 1);

}

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

}

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

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

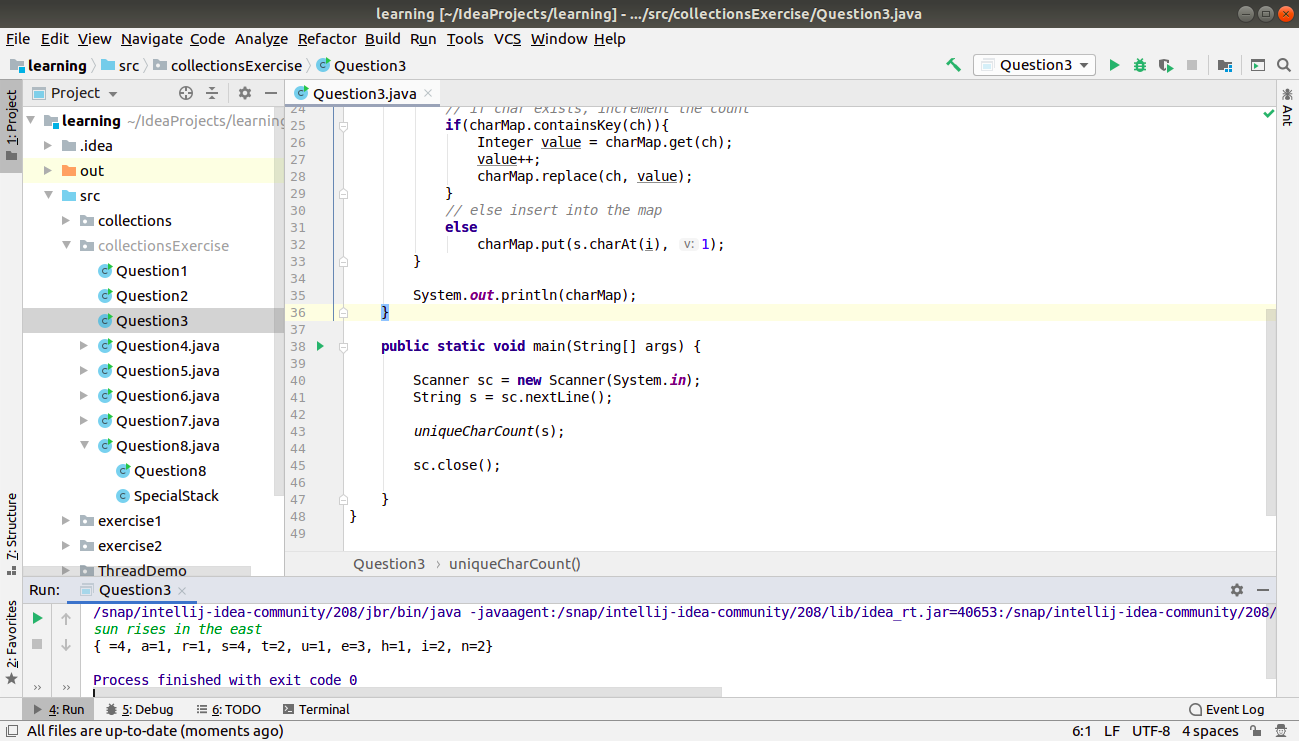
String s = sc.nextLine();

*uniqueCharCount*(s);

sc.close();

}

}



1. **Write a program to sort HashMap by value.**

**package** collectionsExercise;

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

**class** myComparator **implements** Comparator<Map.Entry<String, Integer>> {

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

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

*// if(o1.getValue() > o2.getValue())*

*// return 1;*

*// else if(o1.getValue() < o2.getValue())*

*// return -1;*

*// return 0;*

}

}

**public class** Question4 {

**public static** Map<String, Integer> sortByValue(Map<String, Integer> hmap){

*// convert hashmap to a linked list*

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

*// also need to convert hmap to a set of Map.Entry*

ll.addAll(hmap.entrySet());

Collections.*sort*(ll, **new** myComparator());

*// convert linked list back to a LinkedMap because it preserves the order of insertion*

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

**for**(Map.Entry<String,Integer> entry : ll){

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

}

**return** sortedMap;

}

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

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

mp.put(**"raghav"**, 100);

mp.put(**"gunjan"**, 22);

mp.put(**"mohit"**, 99);

mp.put(**"raju"**, 101);

mp.put(**"manish"**, 78);

Map<String, Integer> sorted = *sortByValue*(mp);

*// print the sorted map*

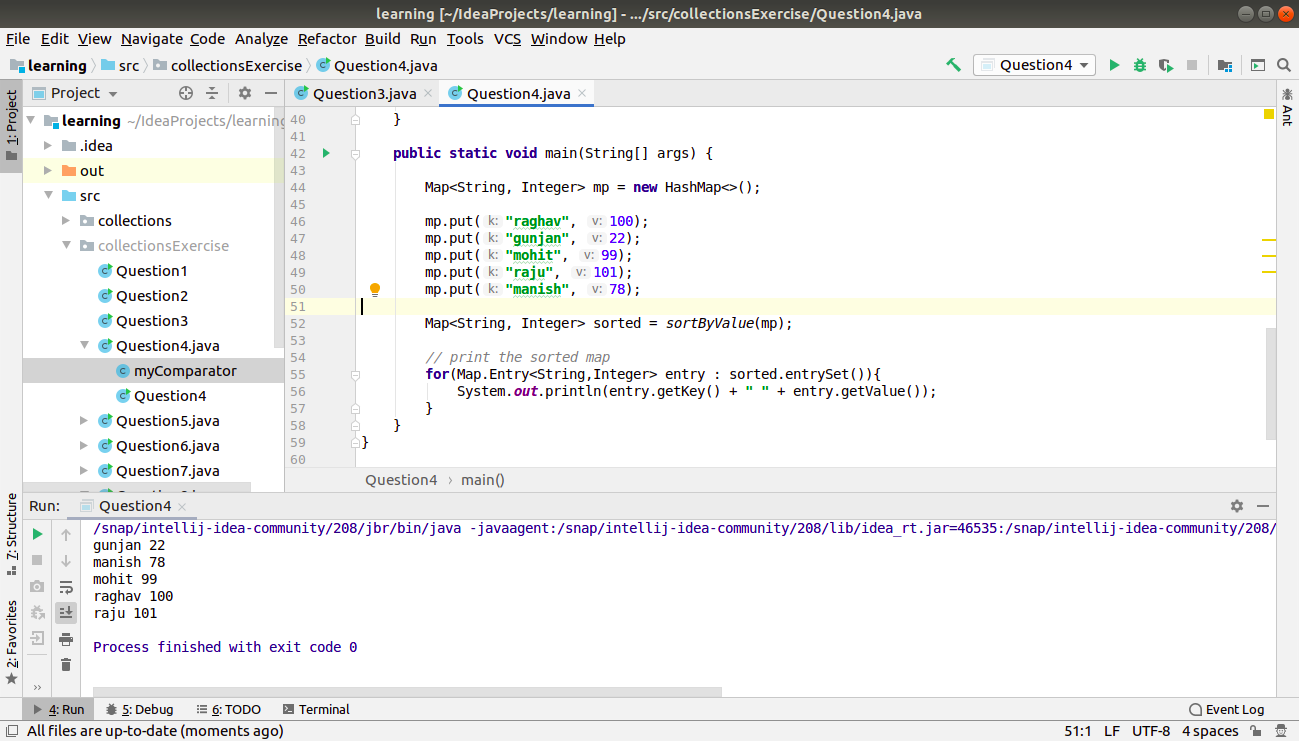
**for**(Map.Entry<String,Integer> entry : sorted.entrySet()){

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

}

}

}



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

**package** collectionsExercise;

*/\**

*Aim : Write a program to sort Employee objects based on highest salary using Comparator.*

*Employee class{ Double Age; Double Salary; String Name*

*\*/*

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

**class** Employee{

*// data members*

**double age**;

**double salary**;

String **name**;

*// parameterised cTor*

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

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

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

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

}

**double** getSalary(){

**return this**.**salary**;

}

String print(){

**return name** + **" "** + **age** + **" "** + **salary**;

}

}

*// custom comparator*

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

**public int** compare(Employee a, Employee b){

**if**(a.getSalary() > b.getSalary())

**return** -1;

**else if**(a.getSalary() < b.getSalary())

**return** 1;

**else**

**return** 0;

}

}

**public class** Question5 {

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

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

*// add objects in list*

emp.add(**new** Employee(20.4, 20000, **"aman"**));

emp.add(**new** Employee(21.4, 200000, **"sujata"**));

emp.add(**new** Employee(22.4, 3100, **"sunanda"**));

emp.add(**new** Employee(23.4, 989000, **"faizan"**));

emp.add(**new** Employee(24.4, 200, **"kamal"**));

emp.add(**new** Employee(25.4, 77870, **"meena"**));

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

*// print the sorted list now*

Iterator<Employee> it = emp.iterator();

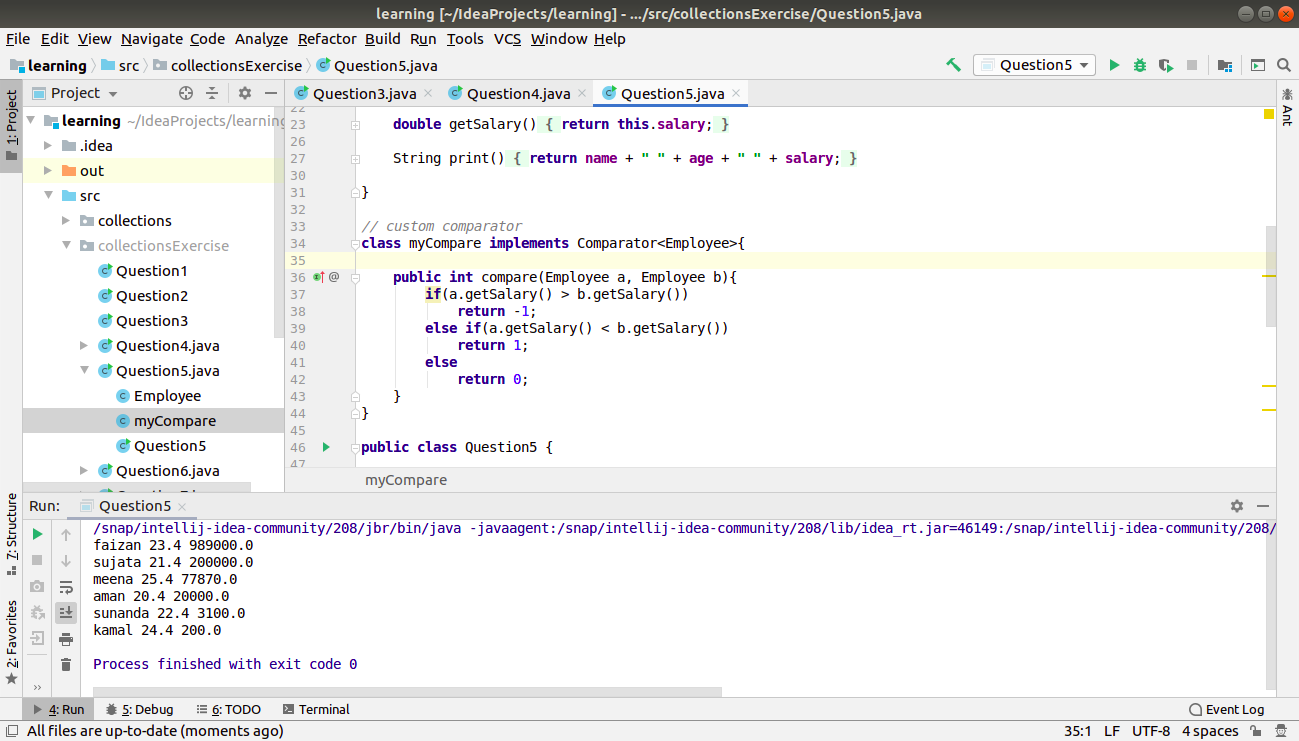
**while**(it.hasNext()){

System.***out***.println(it.next().print());

}

}

}



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**

**package** collectionsExercise;

*/\**

*Aim : 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** Student{

String **name**;

**double score**;

**double age**;

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

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

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

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

}

**public double** getScore(){

**return score**;

}

**public** String getName(){

**return name**;

}

@Override

**public** String toString(){

**return " name - "** + **name** + **", age - "** + **age** + **", score - "** + **score**;

}

}

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

**public int** compare(Student a, Student b){

**if**(a.getScore() > b.getScore())

**return** 1;

**else if**(a.getScore() < b.getScore())

**return** -1;

**else**{

*// if 2 students have same score.*

**return** a.getName().compareTo(b.getName());

}

}

}

**public class** Question6 {

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

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

*// add elements to array*

list.add(**new** Student(**"kapil"**, 90.50, 21));

list.add(**new** Student(**"sonam"**, 44.90, 21));

list.add(**new** Student(**"chaman"**, 90.50, 22));

list.add(**new** Student(**"kamal"**, 67.78, 25));

list.add(**new** Student(**"kandy"**, 90.50, 22));

System.***out***.println(**"before sorting"**);

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

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

}

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

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

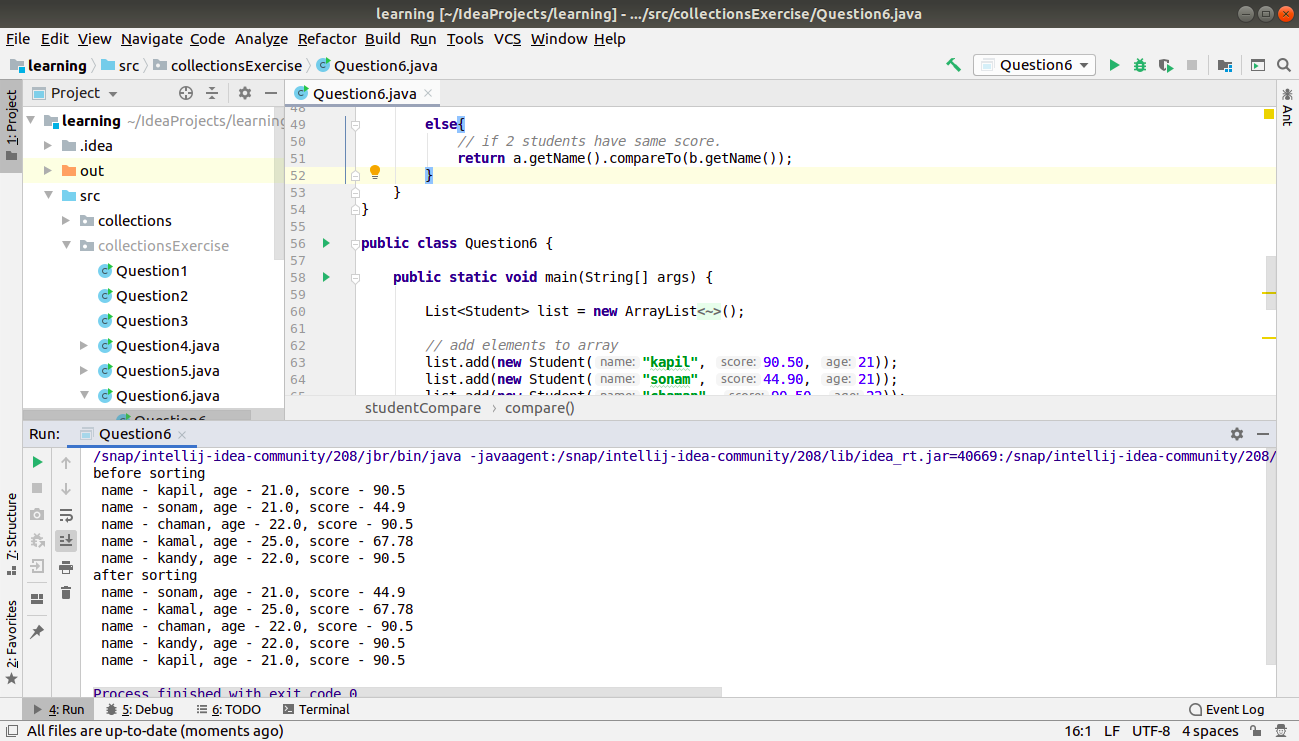
**for**(Student s : list){

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

}

}

}



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

**package** collectionsExercise;

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

**class** SortByValue **implements** Comparator<Integer>{

**private final** Map<Integer, Integer> **indexMap**;

**private final** Map<Integer, Integer> **freqMap**;

SortByValue(Map<Integer, Integer> freqMap, Map<Integer, Integer> indexMap){

**this**.**indexMap** = indexMap;

**this**.**freqMap** = freqMap;

}

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

**if**(**freqMap**.get(o1) > **freqMap**.get(o2))

**return** -1;

**else if**(**freqMap**.get(o1) < **freqMap**.get(o2))

**return** 1;

**else**{

**return indexMap**.get(o1) - **indexMap**.get(o2);

}

}

}

**public class** Question7 {

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

Integer arr[] = {1,1,1,1,1,2,2,2,2,3,3,3,4,4,5,5,5,7,8,8,9,9,9};

*// make the frequency map for the array - value->frequency*

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

*// stores value -> minIndex*

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

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

**int** element = arr[i];

**if**(freqMap.containsKey(element)){

**int** freq = freqMap.get(element);

freq++;

freqMap.put(element, freq);

}

**else**{

freqMap.put(element, 1);

}

**if**(!indexMap.containsKey(element)){

indexMap.put(element, i);

}

}

Arrays.*sort*(arr, **new** SortByValue(freqMap, indexMap));

**for**(Integer val : arr){

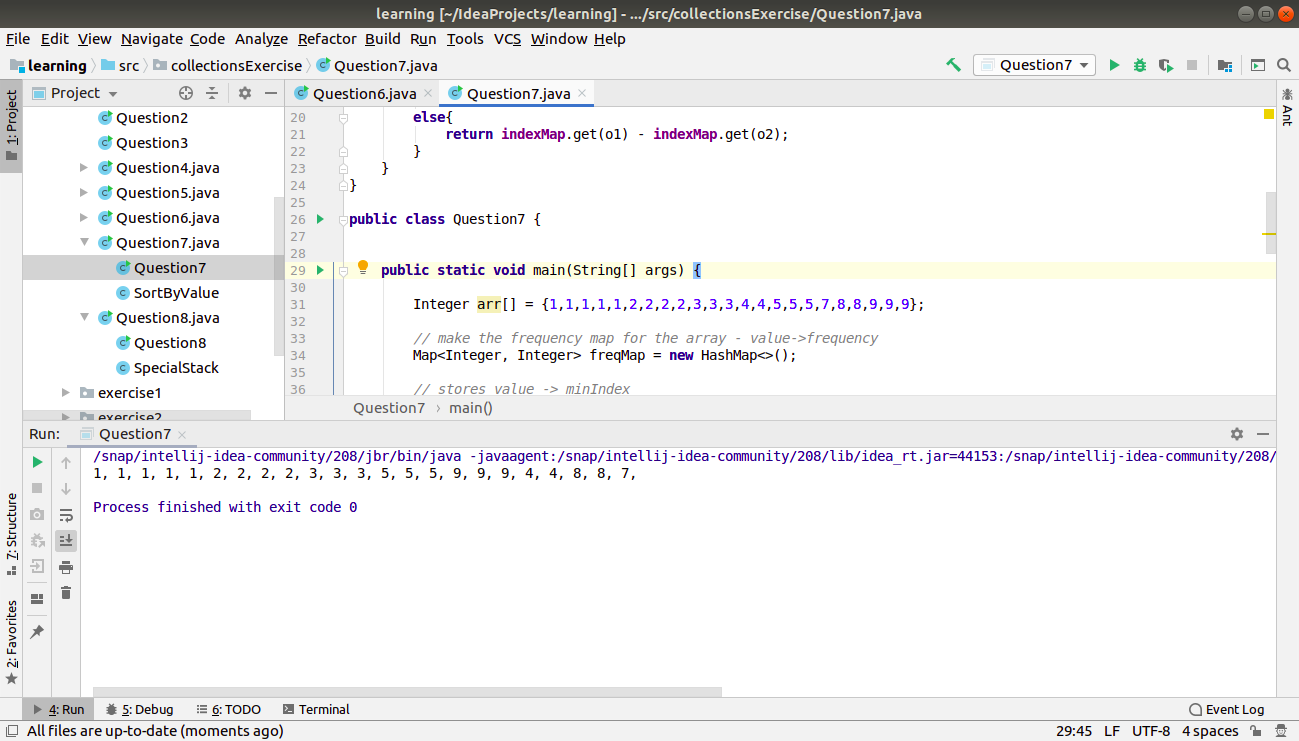
System.***out***.print(val + **", "**);

}

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

}

}



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))**

**package** collectionsExercise;

*/\**

*Aim :*

*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{*

*//*

*// Stack<Integer> stk = new Stack<Integer>();*

*// TreeMap<Integer, Integer> mp = new TreeMap<>();*

*//*

*// public void push(int val){*

*// stk.push(val);*

*// mp.put(val, 0);*

*// }*

*//*

*// public void pop(){*

*// try{*

*// stk.pop();*

*// }catch (EmptyStackException e){*

*// System.out.println("stack empty");*

*// }*

*// }*

*//*

*// public Integer top(){*

*// return stk.peek();*

*// }*

*//*

*// public boolean isEmpty(){*

*// return stk.empty();*

*// }*

*//*

*// public boolean isFull(){*

*//*

*// }*

*//*

*// public Integer getMin(){*

*// return mp.firstKey();*

*// }*

*//*

*//}*

**import** com.sun.source.tree.Tree;

**import** java.util.Scanner;

**import** java.util.TreeMap;

**class** SpecialStack{

**int arr**[];

**int top**;

**int size**;

TreeMap<Integer, Integer> **mp** = **new** TreeMap<>();

SpecialStack(){

**top**=-1;

**size** = 20;

**arr** = **new int**[20];

}

SpecialStack(**int** size){

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

**top** = -1;

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

}

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

**if**(isFull())

System.***out***.println(**"cannot push into stack - already full"**);

**else**{

**arr**[++**top**] = val;

**if**(**mp**.containsKey(val)){

**int** count = **mp**.get(val);

**mp**.put(val, ++count);

}

**else**{

**mp**.put(val, 1);

}

}

}

**public int** pop() {

**if** (isEmpty()) {

System.***out***.println(**"stack empty"**);

**return** -99999999;

}

**else** {

**top**--;

**size**--;

**int** val = **arr**[**top** + 1];

**if**(**mp**.get(val)==1){

**mp**.remove(val);

}

**else**{

**int** count = **mp**.get(val);

**mp**.put(val, --count);

}

**return** val;

}

}

**public boolean** isEmpty(){

**return top**==-1;

}

**public boolean** isFull(){

**return top**==**size**-1;

}

**public int** getMin(){

**if**(isEmpty()){

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

**return** -99999999;

}

**return mp**.firstKey();

}

}

**public class** Question8 {

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

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

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

**int** val;

**int** choice=0;

**do**{

System.***out***.println(**"------ MENU ------"**);

System.***out***.println(**"1. PUSH"**);

System.***out***.println(**"2. POP"**);

System.***out***.println(**"3. GET MINIMUM ELEMENT"**);

System.***out***.println(**"0. EXIT"**);

choice = sc.nextInt();

**switch** (choice){

**case** 0: **break**;

**case** 1:

System.***out***.println(**"enter a value - "**);

val = sc.nextInt();

s.push(val);

**break**;

**case** 2:

val = s.pop();

System.***out***.println(**"value popped is - "** + val);

**break**;

**case** 3:

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

**break**;

**default**:

System.***out***.println(**"invalid input"**);

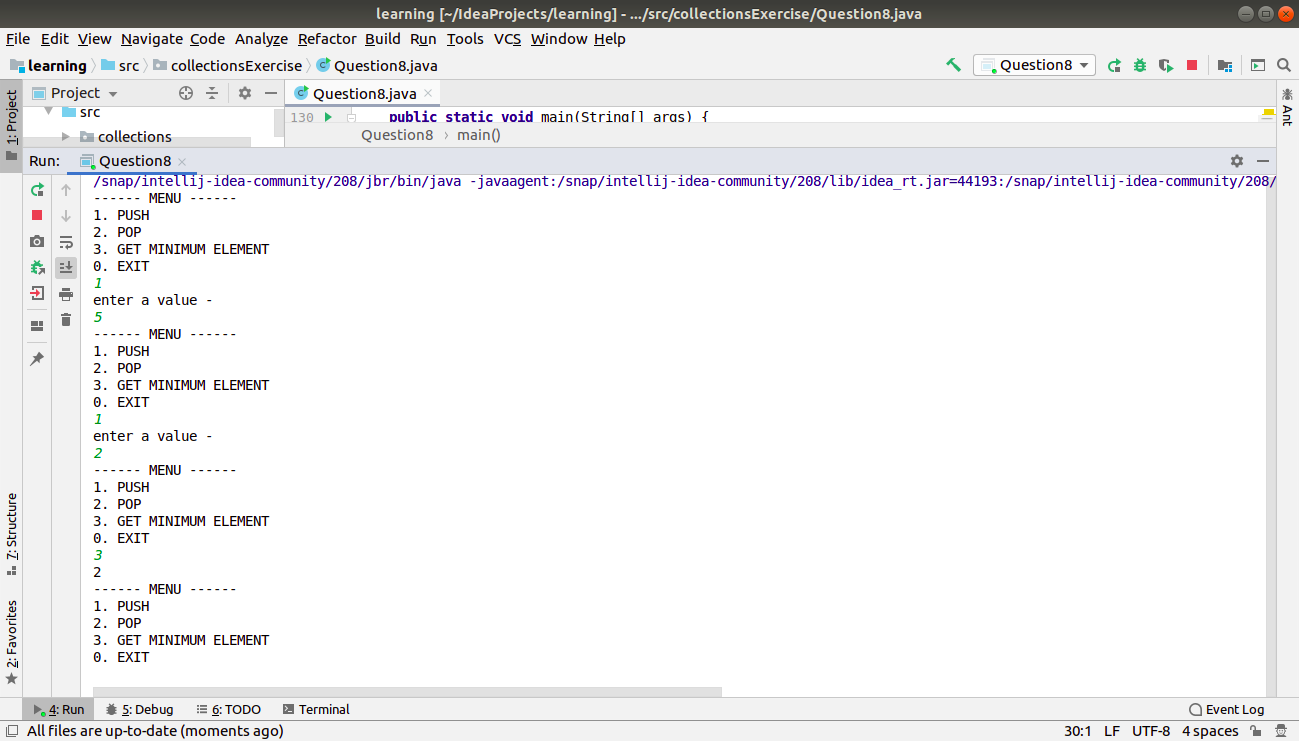
}

}**while**(choice!=0);

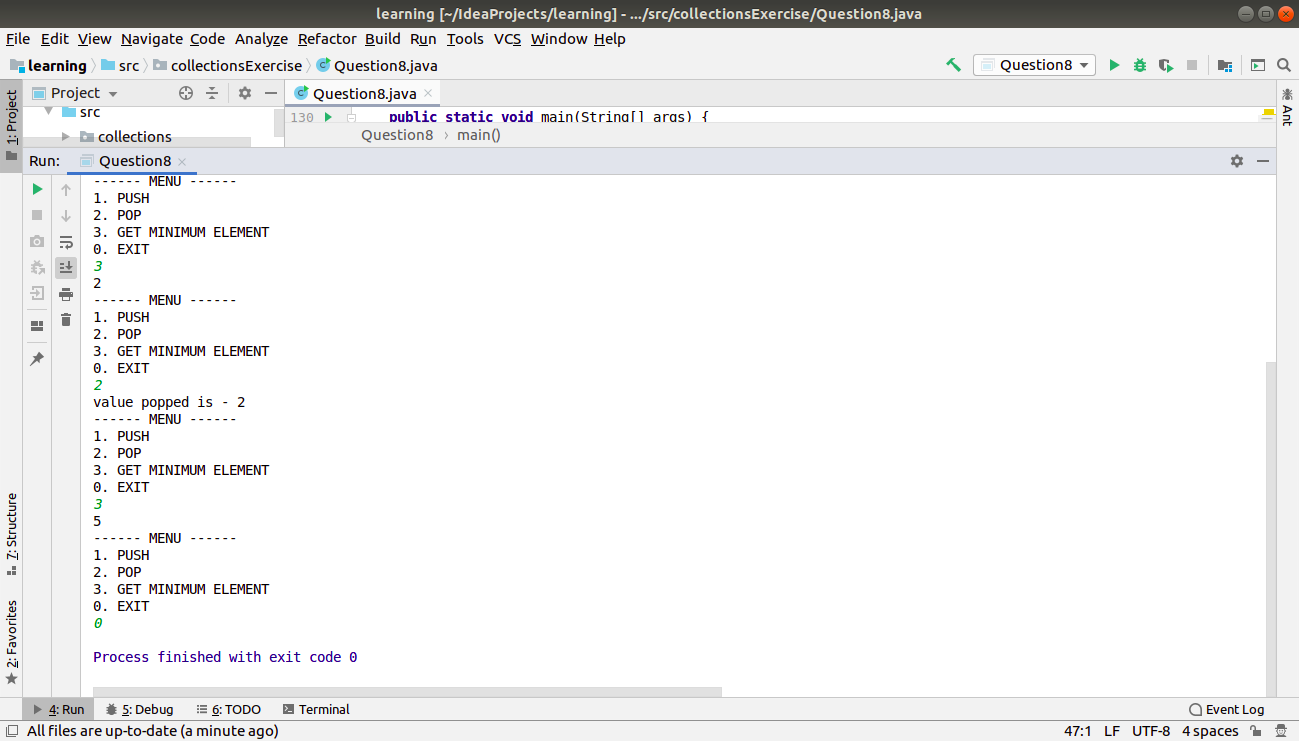
sc.close();

}

}



Please run this program for more clarity.



------------------------------------------------------------------------------------------------------------------

------------------------------------------------------------------------------------------------------------------

------------------------------------------------------------------------------------------------------------------