**COLLECTIONS-ASSIGNMENT**

Ques 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.

Ans: **JAVA-CODE**

import java.util.ArrayList;

import java.util.Iterator;

import java.util.List;

import java.util.Scanner;

public class SumOfListElements {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

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

float sum= (float) 0.0;

System.out.println("Enter five float values to get sum");

for(int i=0; i<5; i++){

list.add(sc.nextFloat());

}

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

while(itr.hasNext()){

sum += itr.next();

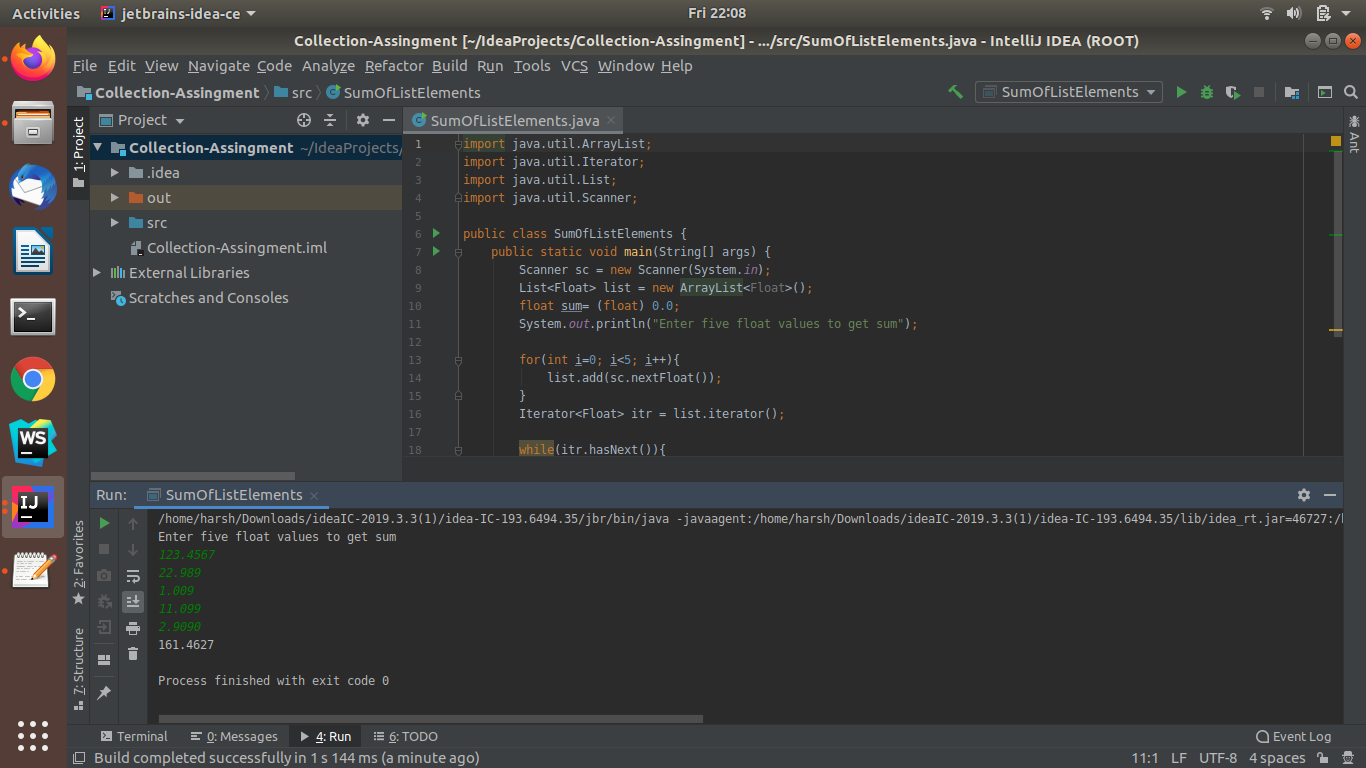
}

System.out.println(sum);

}

}

OUTPUT:



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

Ans:**JAVA-CODE**

import java.util.HashSet;

import java.util.Scanner;

import java.util.Set;

public class UniqueCharacterOfString {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter String To Find Unique Characters");

String str = sc.nextLine();

int len = str.length();

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

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

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

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

}

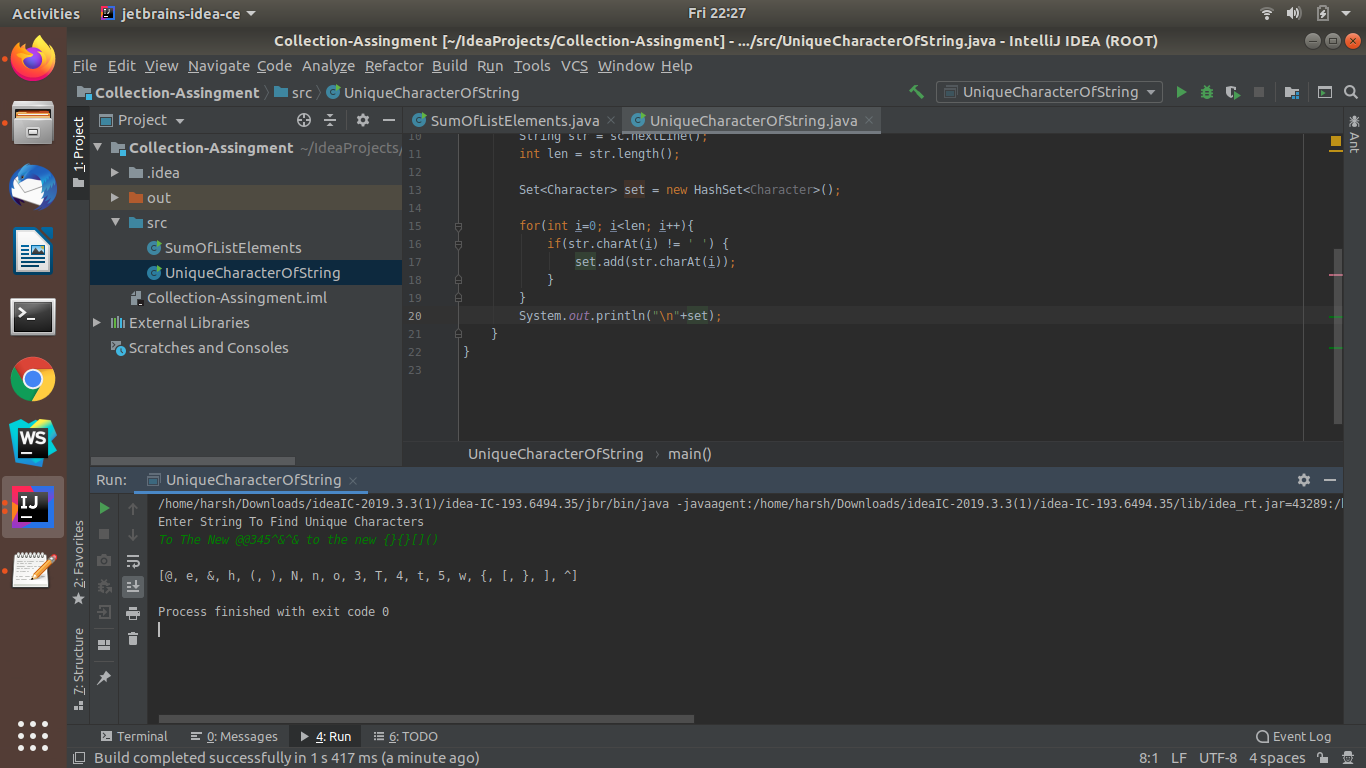
}

System.out.println("\n"+set);

}

}

OUTPUT:



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

Ans :**JAVA-CODE**

import java.util.HashMap;

import java.util.Map;

import java.util.Scanner;

public class NumberOfOccerenceOfCharacterInString {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter String To Find Unique Characters");

String str = sc.nextLine();

int len = str.length();

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

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

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

if(map.containsKey(str.charAt(i))){

int value = map.get(str.charAt(i));

value++;

map.put(str.charAt(i),value);

}

else{

map.put(str.charAt(i),1);

}

}

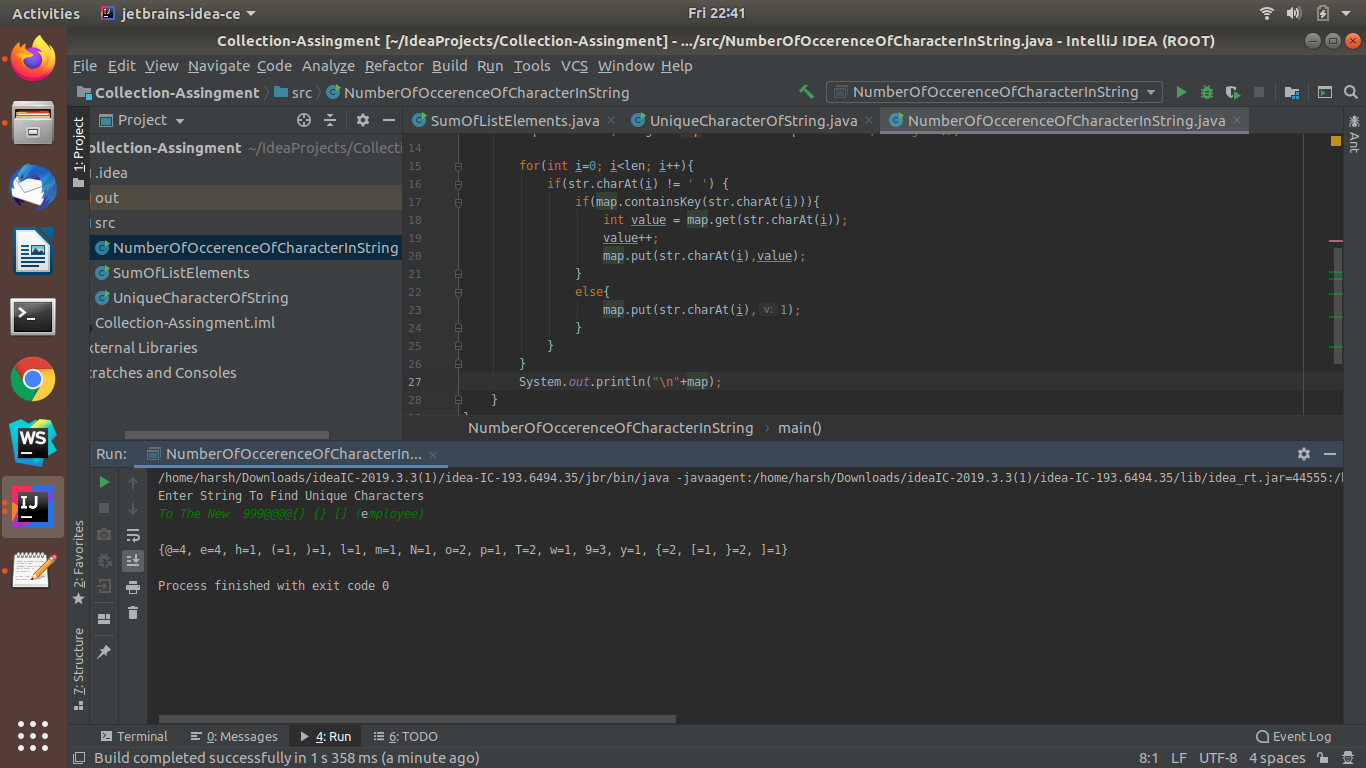
}

System.out.println("\n"+map);

}

}

OUTPUT:



Ques 4: Write a program to sort HashMap by value

Ans:**JAVA-CODE**

import java.util.\*;

public class SortHashMap {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter number of map entries");

int num = sc.nextInt();

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

System.out.println("Enter key and values of map");

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

String key=sc.next();

int value = sc.nextInt();

map.put(key,value);

}

SortHashMap obj = new SortHashMap();

System.out.println(obj.sortbyvalues(map));

}

public HashMap<String,Integer> sortbyvalues(HashMap<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> stringIntegerEntry, Map.Entry<String, Integer> t1) {

return stringIntegerEntry.getValue().compareTo(t1.getValue());

}

});

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

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

ans.put(itr.getKey(),itr.getValue());

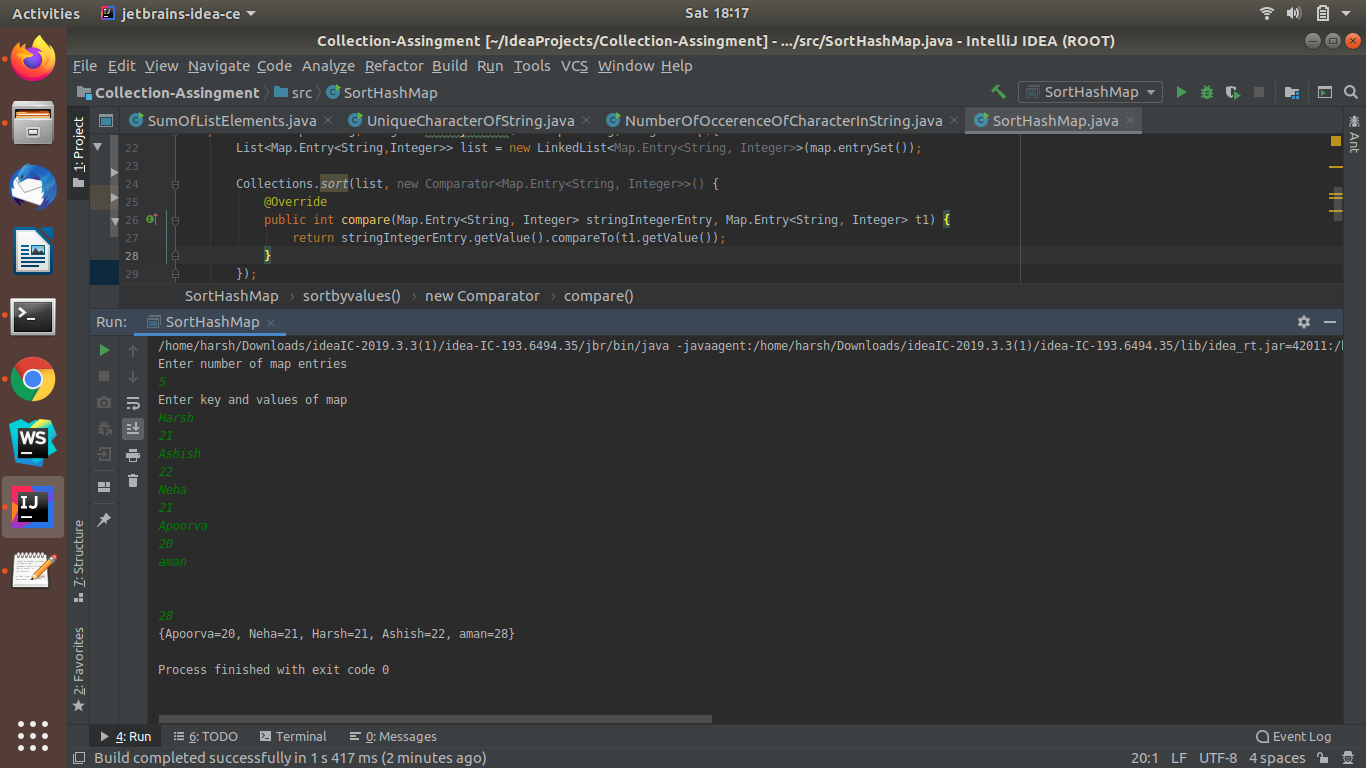
}

return ans;

}

}

OUTPUT:



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

Ans:**JAVA-CODE**

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;

}

public double getAge(){

return this.age;

}

public double getSalary(){

return this.salary;

}

public String getName(){

return this.name;

}

public String toString(){

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

}

}

public class SortEmployeeDetails {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

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

System.out.println("Enter the number of Employees");

int num = sc.nextInt();

System.out.println("Enter Employee details as: name,age,salary");

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

String name = sc.next();

double age = sc.nextDouble();

double salary = sc.nextDouble();

employeelist.add(new Employee(name,age,salary));

}

// System.out.println(list);

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

@Override

public int compare(Employee employee, Employee t1) {

if(employee.getSalary() > t1.getSalary()){

return 1;

}

else{

return -1;

}

}

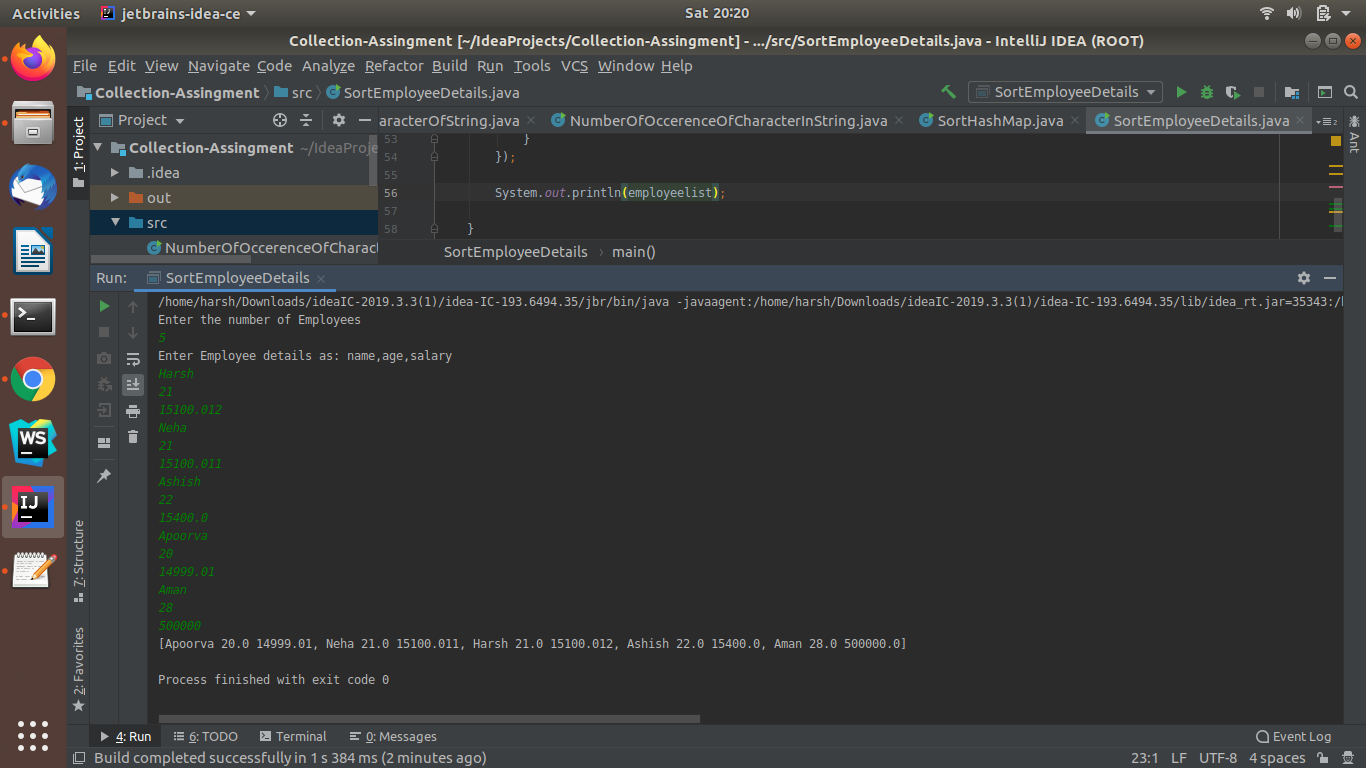
});

System.out.println(employeelist);

}

}

OUTPUT:



Ques 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

Ans: **JAVA-CODE**

import java.util.\*;

class Student{

String name;

double score;

double age;

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

this.name=name;

this.age=age;

this.score=score;

}

public double getAge(){

return this.age;

}

public double getScore(){

return this.score;

}

public String getName(){

return this.name;

}

public String toString(){

return name+" "+age+" "+score;

}

}

public class SortStudentDetails {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

List<Student> studentlist = new ArrayList<Student>();

System.out.println("Enter the number of Students");

int num = sc.nextInt();

System.out.println("Enter Employee details as: name,age,score");

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

String name = sc.next();

double age = sc.nextDouble();

double score = sc.nextDouble();

studentlist.add(new Student(name,age,score));

}

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

@Override

public int compare(Student student, Student t1) {

if(student.getScore() == t1.getScore()){

return student.getName().compareTo(t1.getName());

}

else{

if(student.getScore() >t1.getScore()){

return 1;

}

else{

return -1;

}

}

}

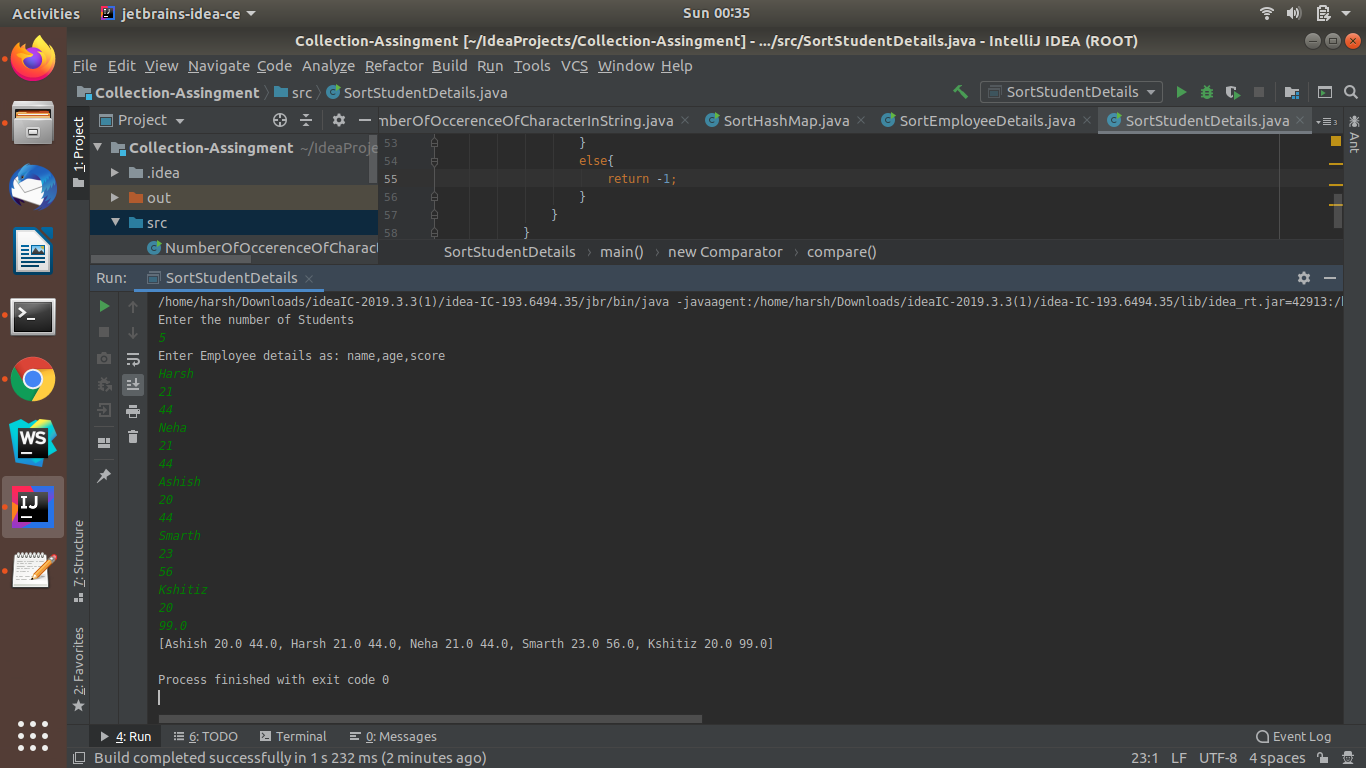
});

System.out.println(studentlist);

}

}

OUTPUT:



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

Ans:**JAVA-CODE**

import java.util.\*;

public class ArrayElementsByDecreasingFrequency {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter NUmber Of Elements");

int limit = sc.nextInt();

int[] arr = new int[limit+1];

System.out.println("Enter The Array Elements");

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

arr[i]=sc.nextInt();

}

System.out.println("Elements Sorted ByFrequency: ");

System.out.println(sort(arr,limit));

}

public static ArrayList<Integer> sort(int[] arr, int size){

LinkedHashMap<Integer,Integer> map = new LinkedHashMap<Integer, Integer>();//HashMap<Integer, Integer>();

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

if (map.containsKey(arr[i])){

int num = map.get(arr[i]);

num++;

map.put(arr[i],num);

}

else {

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

}

}

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

System.out.println(list);

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

@Override

public int compare(Map.Entry<Integer, Integer> integerIntegerEntry, Map.Entry<Integer, Integer> t1) {

if(integerIntegerEntry.getValue() > t1.getValue()){

return -1;

}

else {

return 1;

}

}

});

ArrayList<Integer> result = new ArrayList<Integer>();

for (Map.Entry<Integer,Integer> itr : list){

result.add(itr.getKey());

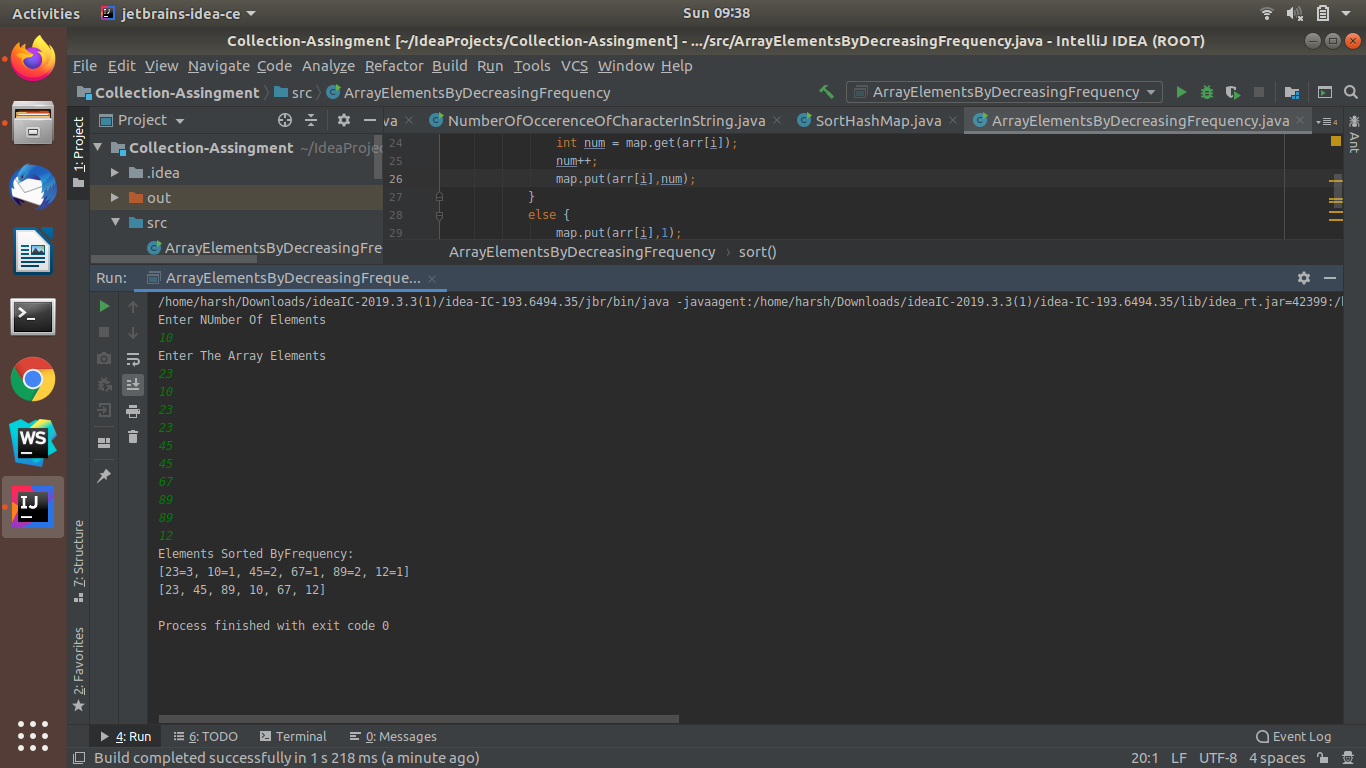
}

return result;

}

}

OUTPUT:



Ques 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))

Ans: **JAVA-CODE**

import java.util.Scanner;

class SpecialStack {

int limit;

class Node{

int data;

Node link;

}

Node top;

SpecialStack(int limit){

this.limit=limit;

top =null;

}

int min=Integer.MAX\_VALUE;

int size=0;

void push(int x){

Node temp = new Node();

if(size == limit){

System.out.println("\n Stack is Full");

return;

}

if (x < min){

min =x;

}

temp.data=x;

temp.link=top;

top=temp;

size++;

}

void pop(){

if (top == null){

System.out.println("Stack is Already Empty");

return;

}

top =(top).link;

}

int getMin(){

return min;

}

boolean isEmpty(){

return top==null;

}

boolean isFull(){

if(size == limit){

return true;

}

else {

return false;

}

}

void displayStack(){

if(top == null){

System.out.println("Stack is Empty Nothing to print");

return;

}

else {

Node temp = top;

while (temp != null){

System.out.print(" "+temp.data);

temp = temp.link;

}

}

}

}

public class SpecialStackEx{

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter The Size Of Stack");

int size = sc.nextInt();

SpecialStack stack = new SpecialStack(size);

if (stack.isEmpty()){

System.out.println("Stack is Empty Need To Insert ");

}

System.out.println("Enter Stack Elements: ");

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

int num = sc.nextInt();

stack.push(num);

}

stack.displayStack();

System.out.println("\nMinimum Element Of Stack is: "+stack.getMin());

if(stack.isFull()){

System.out.println("\nStack is Full");

}

stack.pop();

stack.pop();

System.out.println("\nStack Elements After Popping Out");

stack.displayStack();

}

}

OUTPUT:

