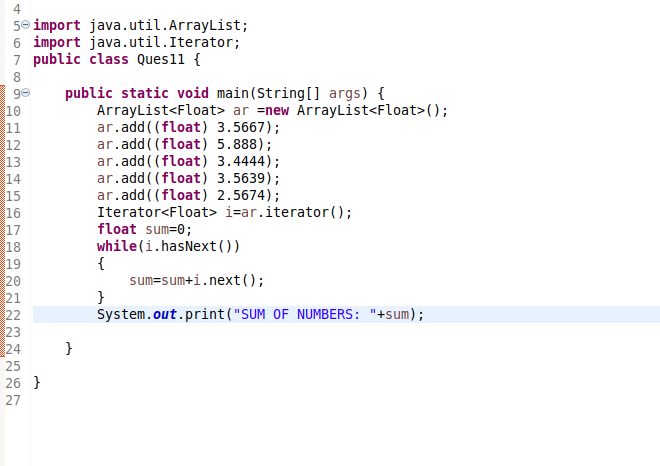
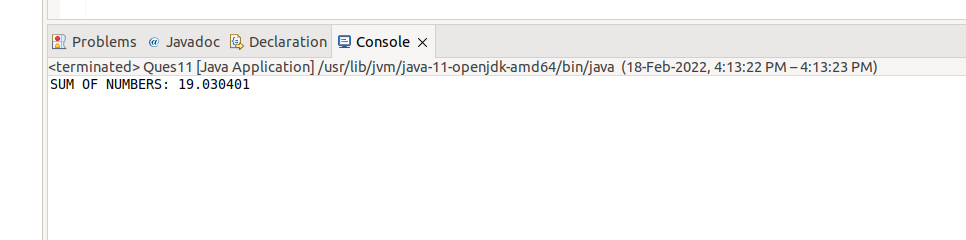
INTRODUCTION TO JAVA(Collections) HONEY ARORA

TRAINEE

**QUES1. 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.**

****

**OUTPUT:**

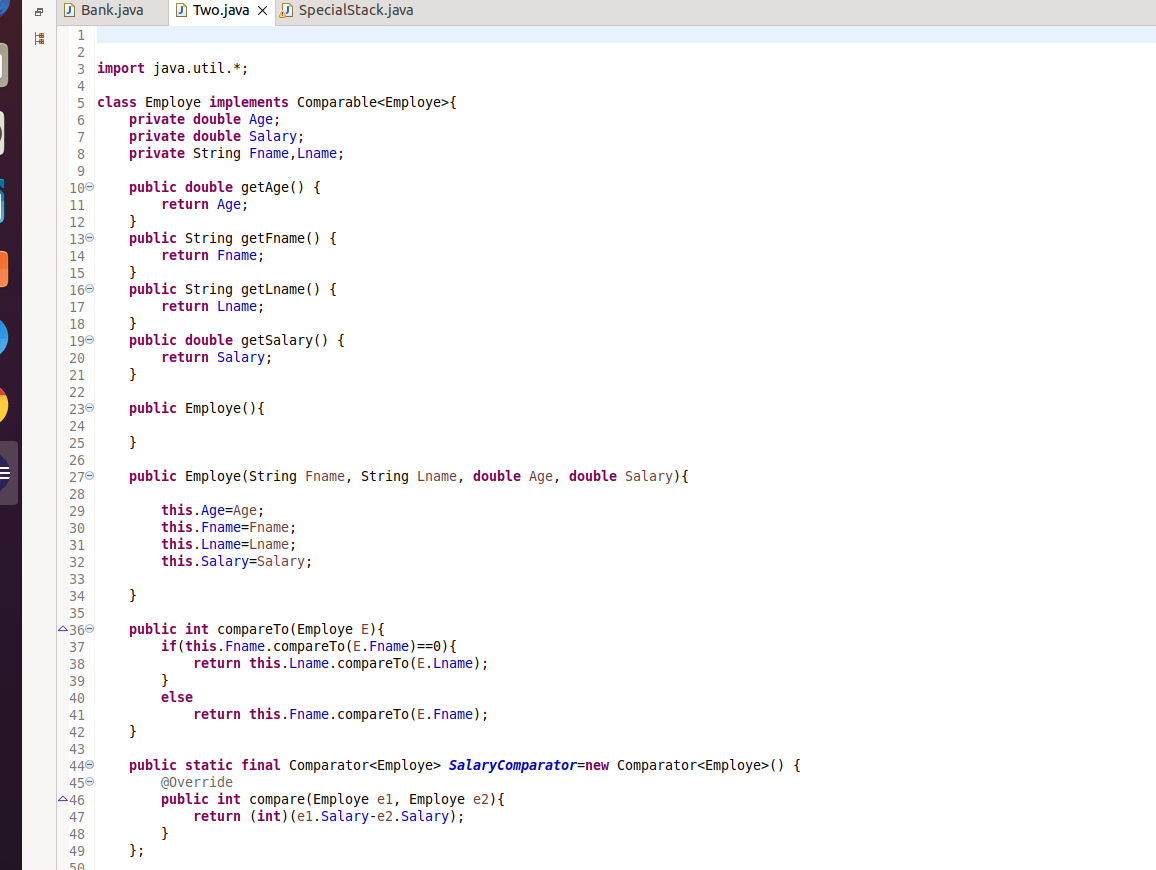
****

**QUES2. Given the following class**

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

**Design the class in such a way that the default sorting should work on firstname and lastname.**

**Also, Write a program to sort Employee objects based on salary using Comparator.**

****

****

**(Below pasted code too)**

import java.util.\*;

class Employe implements Comparable<Employe>{

private double Age;

private double Salary;

private String Fname,Lname;

public double getAge() {

return Age;

}

public String getFname() {

return Fname;

}

public String getLname() {

return Lname;

}

public double getSalary() {

return Salary;

}

public Employe(){

}

public Employe(String Fname, String Lname, double Age, double Salary){

this.Age=Age;

this.Fname=Fname;

this.Lname=Lname;

this.Salary=Salary;

}

public int compareTo(Employe E){

if(this.Fname.compareTo(E.Fname)==0){

return this.Lname.compareTo(E.Lname);

}

else

return this.Fname.compareTo(E.Fname);

}

public static final Comparator<Employe> SalaryComparator=new Comparator<Employe>() {

@Override

public int compare(Employe e1, Employe e2){

return (int)(e1.Salary-e2.Salary);

}

};

public String toString() {

StringBuffer sb=new StringBuffer();

sb.append("Name of employee is :").append(Fname).append(" ").append(Lname).append("\n");

sb.append("Age of employee is :").append(Age).append(".\n");

sb.append("Salary of employee is :").append(Salary).append(".\n");

return sb.toString();

}

}

class Two{

public static void main(String[] Args){

Scanner sc=new Scanner(System.in);

System.out.println("Enter number of employee : ");

int n=sc.nextInt();

//Employee E[]=new Employee[n];

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

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

System.out.println("Enter Fname, Lname, age and salary");

sc.nextLine();

String Fname=sc.nextLine();

String Lname=sc.nextLine();

double Age=sc.nextDouble();

double Salary=sc.nextDouble();

list.add(new Employe(Fname, Lname, Age, Salary));

}

Collections.sort(list,Employe.SalaryComparator);

for(Employe e:list){

System.out.println(e);

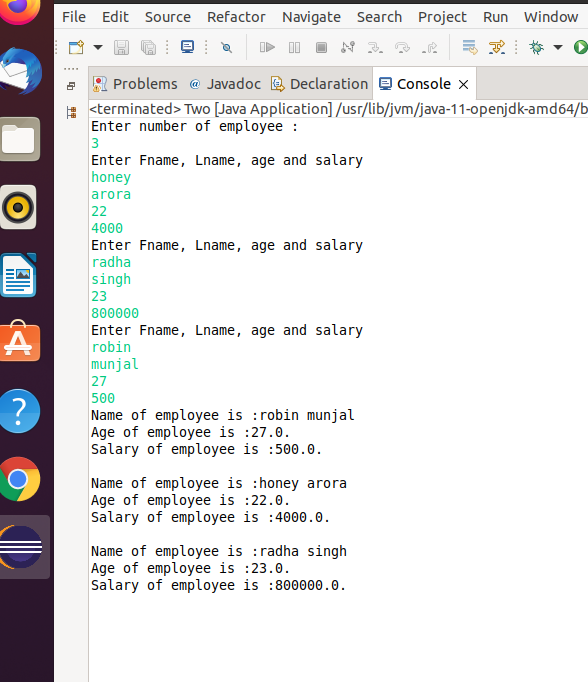
}

sc.close();

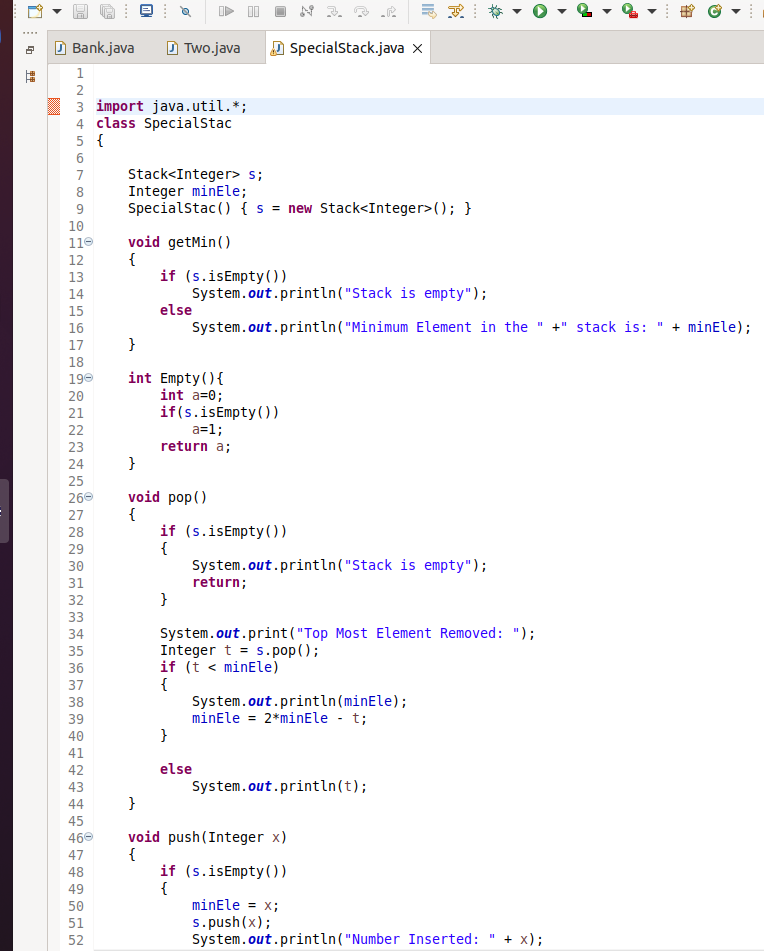
}

}

**OUTPUT:**

****

**Ques3. Design a Data Structure SpecialStack that supports all the stack operations like push(), pop(), isEmpty(), isFull() and an additional operation getMin() which should return a minimum element from the SpecialStack. (Expected complexity ­ O(1))**

****

****

**(Below pasted code too)**

import java.util.\*;

class SpecialStac

{

Stack<Integer> s;

Integer minEle;

SpecialStac() { 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);

}

int Empty(){

int a=0;

if(s.isEmpty())

a=1;

return a;

}

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);

minEle = 2\*minEle - t;

}

else

System.out.println(t);

}

void push(Integer x)

{

if (s.isEmpty())

{

minEle = x;

s.push(x);

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

return;

}

if (x < minEle)

{

s.push(2\*x - minEle);

minEle = x;

}

else

s.push(x);

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

}

};

public class SpecialStack{

public static void main(String[] Args){

SpecialStac ss=new SpecialStac();

Scanner sc=new Scanner(System.in);

while(true){

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

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

System.out.println("3. CHECK WEATHER STACK IS EMPTY");

System.out.println("4. FIND MINIMUM ELEMENT");

System.out.println("5. EXIT");

int choice=sc.nextInt();

switch(choice){

case 1:

System.out.println("Enter value to push");

int val=sc.nextInt();

ss.push(val);

break;

case 2:

ss.pop();

break;

case 3:

int b=ss.Empty();

if(b==1)

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

else

System.out.println("Stack is not empty");

break;

case 4:

ss.getMin();

break;

case 5:

System.exit(0);

break;

default:

System.out.println("Wrong choice enter again");

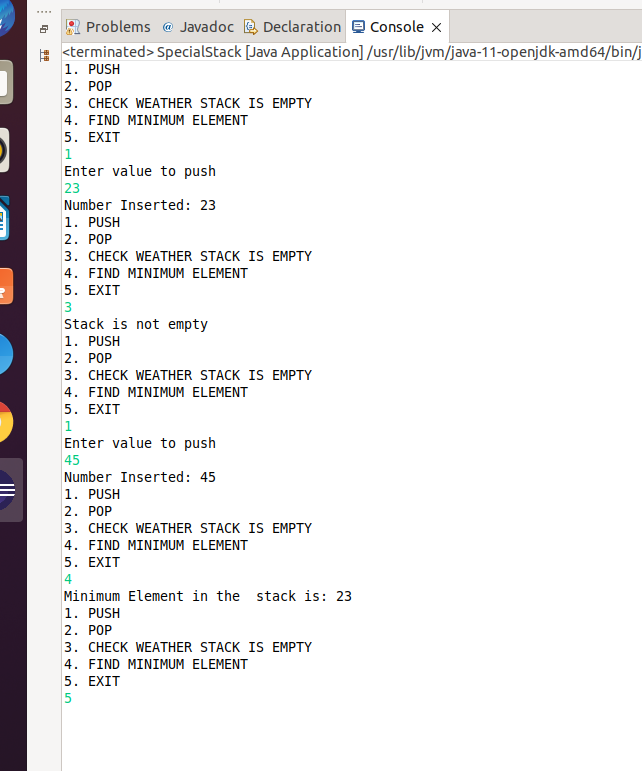
}

}

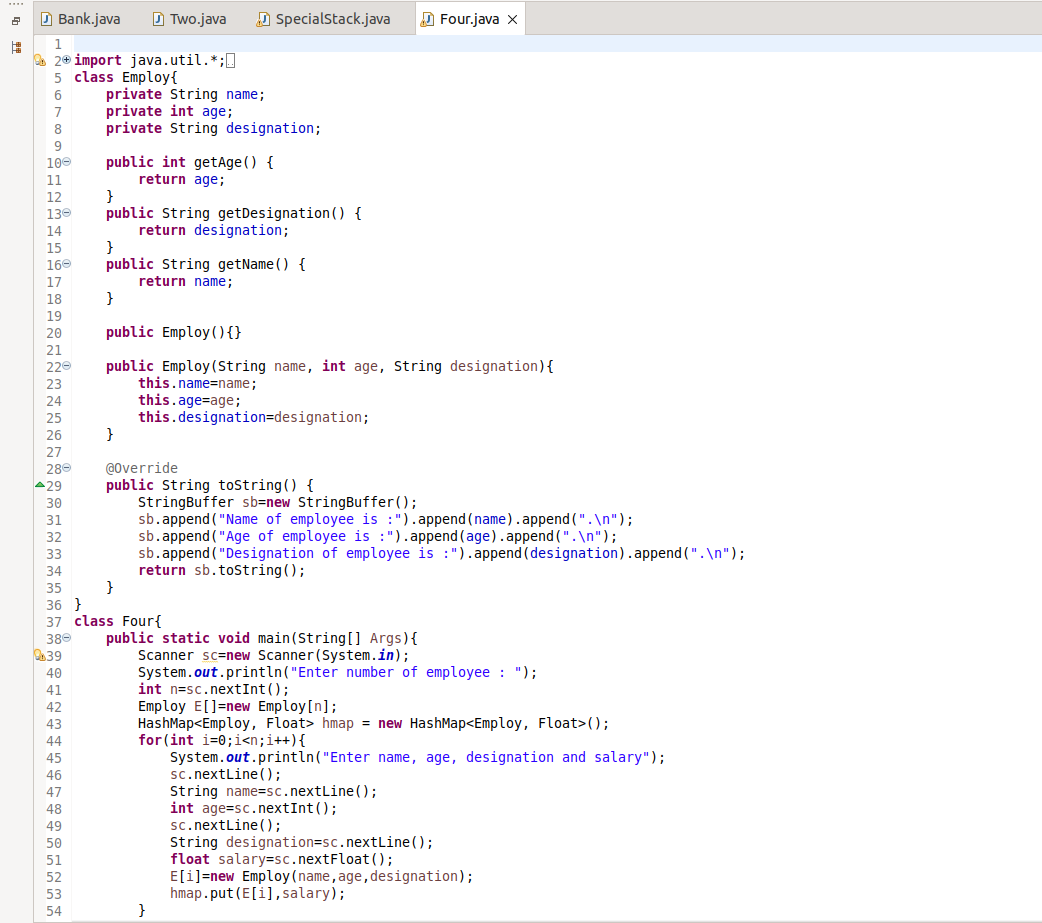
}

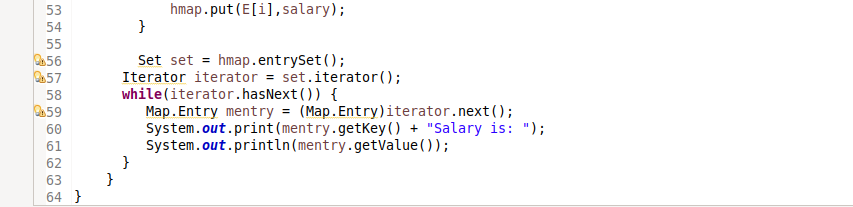
}

**OUTPUT:**

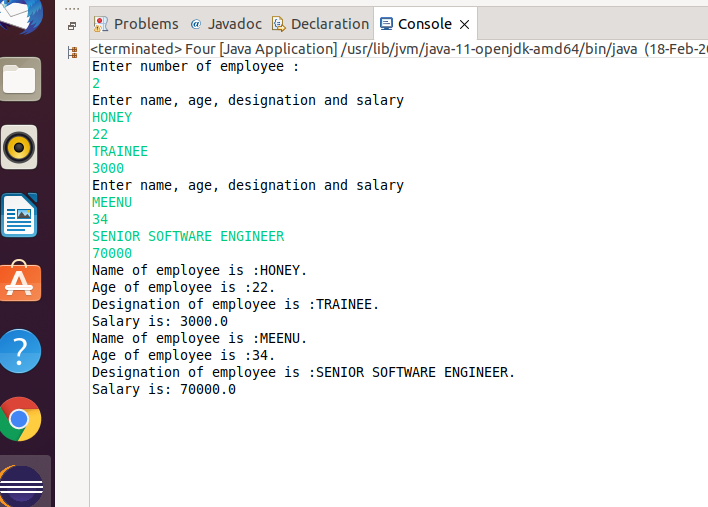
****

**QUES4 . Create class Employee with attributes name,age,designation and use instances of these class as keys in a Map and their salary as value**

****

****

**OUTPUT:**

****