**Question 1**

**Contact class:**

**package** collectionAssignment;

**public** **class** Contact {

String name;

String email;

**public** Contact(String name, String email) {

**this**.name = name;

**this**.email = email;

}

}

**Main class:**

package collectionAssignment;

import java.util.Map;

import java.util.Map.Entry;

import java.util.TreeMap;

public class MainContact{

public static void main(String[] args) {

Map<Long,Contact> map=new TreeMap<Long,Contact>();

Contact c1=new Contact("Ram","ram@gmail.com");

Contact c2=new Contact("Raj","raj@gmail.com");

Contact c3=new Contact("Tony","tony@gmail.com");

map.put(9916376453L,c1);

map.put(9916376343L,c2);

map.put(9916376253L,c3);

System.out.println("All keys");

for(Entry<Long, Contact> entry:map.entrySet()){

long key=entry.getKey();

System.out.println(key);

}

System.out.println("All values");

for(Entry<Long, Contact> entry:map.entrySet()){

Contact c=entry.getValue();

System.out.println(c.name+","+c.email);

}

System.out.println("All keys and values");

for(Entry<Long, Contact> entry:map.entrySet()){

long key=entry.getKey();

Contact c=entry.getValue();

System.out.println(key+": "+c.name+","+c.email);

}

}

}

**Question 2**

**package** collectionAssignment;

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

**public** **class** uniqueObject {

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

Set<String> s=**new** HashSet<>();

s.add("Apple");

s.add("Mango");

s.add("Strawberry");

s.add("Grapes");

s.add("Guava");

s.add("PineApple");

s.add("lichee");

s.add("Orange");

s.add("Jackfriut");

s.add("Banana");

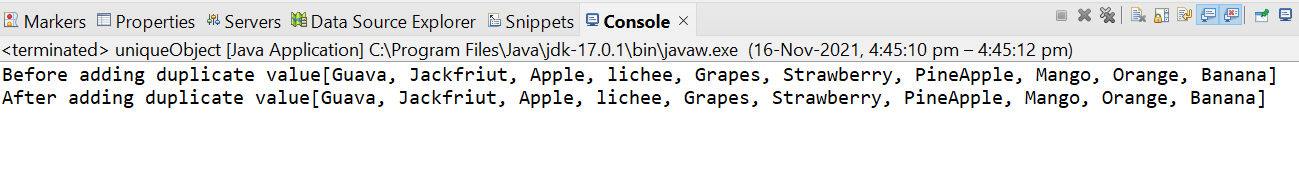
System.***out***.println("Before adding duplicate value"+s);

s.add("Apple");

System.***out***.println("After adding duplicate value"+s);

}

}



**Question 3**

**EmployeeOrder class:**

**package** collectionAssignment;

**public** **class** EmployeeOrder {

**int** id;

String name;

String dept;

**int** salary;

**public** EmployeeOrder(**int** id, String name, String dept, **int** salary) {

**this**.id = id;

**this**.name = name;

**this**.dept = dept;

**this**.salary = salary;

}

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

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

**this**.name = name;

}

**public** String getDept() {

**return** dept;

}

**public** **void** setDept(String dept) {

**this**.dept = dept;

}

**public** **int** getSalary() {

**return** salary;

}

**public** **void** setSalary(**int** salary) {

**this**.salary = salary;

}

@Override

**public** String toString() {

**return** "EmployeeOrder [id=" + id + ", name=" + name + ", dept=" + dept + ", salary=" + salary + "]";

}

}

**EmployeeMain class:**

package collectionAssignment;

import java.util.Comparator;

import java.util.Scanner;

import java.util.Set;

import java.util.TreeSet;

public class EmployeeMain {

public static void main(String[] args) {

Scanner sc=new Scanner(System.in);

System.out.println("Enter the a:To sort by ID, b: To sort by name, c:TO stort by Dept, d: to sorrt by Salary");

String x=sc.next();

if(x.equals("b")) {

TreeSet<EmployeeOrder> nameComp = new TreeSet<EmployeeOrder>(new MyNameComp());

nameComp.add(new EmployeeOrder(1,"Ram","HR",3000));

nameComp.add(new EmployeeOrder(2,"John","Manager",6000));

nameComp.add(new EmployeeOrder(3,"Crish","R&D",2000));

nameComp.add(new EmployeeOrder(4,"Tom","MA",2400));

for(EmployeeOrder e:nameComp){

System.out.println(e);

}

}

else if(x.equals("d")) {

TreeSet<EmployeeOrder> salComp = new TreeSet<EmployeeOrder>(new MySalaryComp());

salComp.add(new EmployeeOrder(1,"Ram","HR",3000));

salComp.add(new EmployeeOrder(2,"John","Manager",6000));

salComp.add(new EmployeeOrder(3,"Crish","R&D",2000));

salComp.add(new EmployeeOrder(4,"Tom","MA",2400));

for(EmployeeOrder e:salComp){

System.out.println(e);

}

}

else if(x.equals("a")) {

TreeSet<EmployeeOrder> idComp = new TreeSet<EmployeeOrder>(new MyIdComp());

idComp.add(new EmployeeOrder(3,"Crish","R&D",2000));

idComp.add(new EmployeeOrder(1,"Ram","HR",3000));

idComp.add(new EmployeeOrder(2,"John","Manager",6000));

idComp.add(new EmployeeOrder(4,"Tom","MA",2400));

for(EmployeeOrder e:idComp){

System.out.println(e);

}

}

else if(x.equals("c")) {

TreeSet<EmployeeOrder> deptComp = new TreeSet<EmployeeOrder>(new MyDeptComp());

deptComp.add(new EmployeeOrder(1,"Ram","HR",3000));

deptComp.add(new EmployeeOrder(2,"John","Manager",6000));

deptComp.add(new EmployeeOrder(3,"Crish","R&D",2000));

deptComp.add(new EmployeeOrder(4,"Tom","MA",2400));

for(EmployeeOrder e:deptComp){

System.out.println(e);

}

}

else {

System.out.println("You have to enter any value from a,b,c,d");

}

sc.close();

}

}

class MyDeptComp implements Comparator<EmployeeOrder>{

@Override

public int compare(EmployeeOrder e1, EmployeeOrder e2) {

return e1.getDept().compareTo(e2.getDept());

}

}

class MyNameComp implements Comparator<EmployeeOrder>{

@Override

public int compare(EmployeeOrder e1, EmployeeOrder e2) {

return e1.getName().compareTo(e2.getName());

}

}

class MySalaryComp implements Comparator<EmployeeOrder>{

@Override

public int compare(EmployeeOrder e1, EmployeeOrder e2) {

if(e1.getSalary() > e2.getSalary()){

return 1;

} else {

return -1;

}

}

}

class MyIdComp implements Comparator<EmployeeOrder>{

@Override

public int compare(EmployeeOrder e1, EmployeeOrder e2) {

if(e1.getId() > e2.getId()){

return 1;

} else {

return -1;

}

}

}