1. Given a Tree Map<Long, Contact> which has phone numbers for keys and contact objects for values. Write solutions to

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

**public** **class** first

{

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

{

TreeMap<Integer,String> tm = **new** TreeMap();

tm.put(5,"AB");

tm.put(4,"CD");

tm.put(3,"EF");

tm.put(2,"GH");

tm.put(1,"IJ");

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

Set keys = tm.keySet();

Iterator i = keys.iterator();

**while** (i.hasNext())

{

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

}

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

Collection getValues = tm.values();

i = getValues.iterator();

**while** (i.hasNext())

{

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

}

System.***out***.println("\nMap = "+tm);

}

}

2) Write an application to store 10 unique product objects. In case there is an attempt to add a duplicate product, it should be silently rejected. Hint: Use HasSet or TreeSet

Extra(optional): Use ArrayList in the above solution. (This is optional)

**import** java.util.TreeSet;

**public** **class** second {

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

TreeSet<Function> func = **new** TreeSet<>();

func.add(**new** Function("MILK",1));

func.add(**new** Function("BOTTLE",2));

func.add(**new** Function("MOBILE",3));

func.add(**new** Function("SHIRT",4));

//adding a duplicate product ID

func.add(**new** Function("PEN",2));

func.add(**new** Function("ERASER",4));

**for**(Function f : func)

{

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

}

}

}

---------------------Function class--------------------------

**public** **class** Function **implements** Comparable<Function>

{

**private** String name;

**private** **int** id;

Function(String product\_name, **int** product\_id)

{

**this**.id = product\_id;

**this**.name = product\_name;

}

**private** String getName()

{

**return** name;

}

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

{

**return** id;

}

**public** **int** compareTo(Function f)

{

**if**(id == f.getId())

{

**return** 0;

}

**else** **if**(name.compareTo(f.getName()) < 0)

{

**return** -1;

}

**else**

{

**return** -1;

}

}

**public** String toString()

{

**return** name + " - " + id;

}

}

3) Store at least 10 Employee Objects in an TreeSet<Employee>. When the application runs the user should be asked to select one of the options upon which you will print the employee details in a sorted manner.

For E.g. Run Application:

a) ID

b) Name

c) Department

d) Salary

Your choice: b

<Should print all the employee's details sorted by name>

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

**class** Employee **implements** Comparable<Employee>{

**int** id;

String Name;

String Department;

**int** Salary;

**public** Employee(**int** id,String name,String department,**int** salary) {

**this**.id=id;

**this**.Name=name;

**this**.Department=department;

**this**.Salary=salary;

}

**public** **int** compareTo(Employee e) {

**if**(id>e.id) {

**return** 1;

}

**else** **if**(id<e.id)

{

**return** -1;

}

**else** {

**return** 0;

}

}

}

**public** **class** third {

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

// **TODO** Auto-generated method stub

Set <Employee>set=**new** TreeSet<Employee>();

Employee emp1=**new** Employee(1,"AJAY","software",30000);

Employee emp2=**new** Employee(2,"VJAY","developer",35000);

Employee emp3=**new** Employee(3,"SANJAY","analyst",20000);

Employee emp4=**new** Employee(4,"PRANAY","software",30000);

Employee emp5=**new** Employee(5,"AMAY","programer",35000);

Employee emp6=**new** Employee(6,"MRUNAL","engineer",25000);

Employee emp7=**new** Employee(7,"SHUBHAM","manager",45000);

Employee emp8=**new** Employee(8,"NILESH","system analyst",35000);

Employee emp9=**new** Employee(9,"ROBIN","programer",35000);

Employee emp10=**new** Employee(10,"ROHIT","qa analyst",45000);

set.add(emp1);

set.add(emp2);

set.add(emp3);

set.add(emp4);

set.add(emp5);

set.add(emp6);

set.add(emp7);

set.add(emp8);

set.add(emp9);

set.add(emp10);

**for**(Employee e:set) {

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

}

}

}

4) Given a LinkedList of Objects representing date of birth's (use any inbuild java class to represent date), print the date's along with the message: Your date of Birth is DD-MM-YYYY, and it (was or was not) a leap year.

E.g.

a)For the date 23-12-2000

Your date of birth is 23-12-2000 and it was a leap year

b)For the date 23-12-2001

Your date of birth is 23-12-2000 and it was not a leap year

import java.time.LocalDate;

import java.time.format.DateTimeFormatter;

import java.util.LinkedList;

public class FOURTH {

public static void main(String[] args) {

LocalDate date1 = LocalDate.of(2000, 12, 23);

LocalDate date2 = LocalDate.of(1997, 8, 21);

LocalDate date3 = LocalDate.of(2000, 12, 23);

LinkedList<LocalDate> list = new LinkedList<LocalDate>();

list.add(date1);

list.add(date2);

list.add(date3);

for(LocalDate l : list)

{

String printDate = l.format(DateTimeFormatter.ofPattern("dd-MM-YYYY"));

if(l.isLeapYear())

{

System.out.println("Your Date of Birth is " + printDate + " and it was a leap year");

}

else

{

System.out.println("Your Date of Birth is " + printDate + " and it was not a leap year");

}

}

}

}

**OUTPUT:**

Your Date of Birth is 23-12-2000 and it was a leap year

Your Date of Birth is 21-08-1997 and it was not a leap year

Your Date of Birth is 23-12-2000 and it was a leap year