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

Write solutions to

* Fetch all the keys and print them,
* Fetch all the values and print them
* Print all key-value pairs

Note:

* Contacts should be stored in descending order of phone number
* Contact Class:

PhoneNumer: Name: Email: Gender:

**package** comparision;

**import** java.util.EnumSet;

//import Contact.gen;

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

**long** phn;

String name, email;

**public** **enum** gen {***F***,***M***}

gen g;

**public** Contact(**long** phn, String name, String email, gen g) {

**super**();

**this**.phn = phn;

**this**.name = name;

**this**.email = email;

**this**.g = g;

}

**public** **long** getPhn() {

**return** phn;

}

**public** **void** setPhn(**long** phn) {

**this**.phn = phn;

}

**public** String getName() {

**return** name;

}

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

**this**.name = name;

}

**public** String getEmail() {

**return** email;

}

**public** **void** setEmail(String email) {

**this**.email = email;

}

**public** gen getG() {

**return** g;

}

**public** **void** setG(gen g) {

**this**.g = g;

}

}

**package** comparision;

**package** comparision;

**import** java.util.Set;

**import** java.util.Collections;

**import** java.util.Map;

**import** java.util.TreeMap;

**import** comparision.Contact.gen;

**public** **class** PhnNum {

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

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

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

Contact c1 = **new** Contact((**long**)978009594, "Ram", "ram@gmail.com",gen.***M***);

Contact c2 = **new** Contact((**long**)770502245, "Sam", "sam@gmail.com",gen.***M***);

Contact c3 = **new** Contact((**long**)991928877, "Jay", "jay@gmail.com",gen.***F***);

Contact c4 = **new** Contact((**long**)983848150, "Vijay", "vijay@gmail.com",gen.***M***);

Contact c5 = **new** Contact((**long**)635478941, "Sk", "sk@gmail.com",gen.***M***);

map.put((**long**)978009594, c1);

map.put((**long**)770502245, c2);

map.put((**long**)991928877, c3);

map.put((**long**)983848150, c4);

map.put((**long**)635478944, c5);

Map<Long,Contact> sortedMapDesc = **new** TreeMap<>(

Collections.*reverseOrder*());

sortedMapDesc.putAll(map);

**for**(Map.Entry<Long, Contact> entry1: sortedMapDesc.entrySet())

{

Long key = entry1.getKey();

Contact c = entry1.getValue();

System.***out***.println(key + " -->Phone Number in descending order");

System.***out***.println(c.name+" "+c.email+" "+c.g + " -->Other Details");

System.***out***.println(c.phn + " "+ c.name+ " "+ c.email+ " "+ c.g + " -->Full Details");

}

}

}

OUTPUT:

991928877 -->Phone Number in descending order

Jay jay@gmail.com F -->Other Details

991928877 Jay jay@gmail.com F -->Full Details

983848150 -->Phone Number in descending order

Vijay vijay@gmail.com M -->Other Details

983848150 Vijay vijay@gmail.com M -->Full Details

978009594 -->Phone Number in descending order

Ram ram@gmail.com M -->Other Details

978009594 Ram ram@gmail.com M -->Full Details

770502245 -->Phone Number in descending order

Sam sam@gmail.com M -->Other Details

770502245 Sam sam@gmail.com M -->Full Details

635478944 -->Phone Number in descending order

Sk sk@gmail.com M -->Other Details

635478941 Sk sk@gmail.com M -->Full Details

1. 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.

##### class name - Item

**package** collecton\_java;

**public** **class** Item **implements** Comparable<Item>

{

**private** String product\_name;

**private** **int** product\_id;

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

{

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

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

}

**private** String getName()

{

**return** product\_name;

}

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

{

**return** product\_id;

}

**public** **int** compareTo(Item f)

{

**if**(product\_id == f.getId())

{

**return** 0;

}

**else** **if**(product\_name.compareTo(f.getName()) < 0)

{

**return** -1;

}

**else**

{

**return** -1;

}

}

**public** String toString()

{

**return** product\_name + " - " + product\_id;

}

}

##### class name -Main

**package** collecton\_java;

**import** java.util.TreeSet;

**public** **class** Main {

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

TreeSet<Item> test = **new** TreeSet<>();

test.add(**new** Item("toyota",1));

test.add(**new** Item("suzki",2));

test.add(**new** Item("BMW",3));

//adding a duplicate product name

test.add(**new** Item("suzki",4));

//adding a duplicate product ID

test.add(**new** Item("Hyundai",2));

test.add(**new** Item("Mahindra",5));

**for**(Item f : test)

{

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

}

}

}

OUTPUT:

Mahindra - 5

suzki - 4

BMW - 3

suzki - 2

toyota - 1

1. Store atleast 10 Employee objects in a TreeSet. When your application runs the user should be asked to selct one of the options upon which you will print the employee details in a sorted manner.

######## class name - Options

public class Options {

private int id;

private int salary;

private String name;

private String department;

public Options(int id, int salary, String name, String department)

{

this.id = id;

this.salary = salary;

this.name = name;

this.department = department;

}

public String getName()

{

return name;

}

public void setName(String name)

{

this.name = name;

}

public String getDep()

{

return department;

}

public void setDep(String department)

{

this.department = department;

}

public int getId()

{

return id;

}

public int getSalary()

{

return salary;

}

}

##### class name - IdCompare

import java.util.Comparator;

public class IdCompare implements Comparator<Options>

{

public int compare(Options o1, Options o2)

{

return o1.getId() - o2.getId();

}

}

//class name - NameCompare

import java.util.Comparator;

public class NameCompare implements Comparator<Options>

{

public int compare(Options o1, Options o2)

{

return o1.getName().compareTo(o2.getName());

}

}

#### class name - DepComapre

import java.util.Comparator;

public class DepComapre implements Comparator<Options>

{

public int compare(Options o1, Options o2)

{

return o1.getDep().compareTo(o2.getDep());

}

}

##### class name - SalaryCompare

import java.util.Comparator;

public class SalaryCompare implements Comparator<Options>{

public int compare(Options o1, Options o2)

{

return o1.getSalary() - o2.getSalary();

}

}

###### class name - TestCompare

import java.util.\*;

public class TestCompare {

public static void main(String[] args) {

Scanner scan = new Scanner(System.in);

System.out.println("You want to sort in order of \n\n1.ID\n2.Name\n3.Department\n4.Salary\n\nEnter your option: ");

int option = scan.nextInt();

switch(option)

{

case 1:

TreeSet<Options> set = new TreeSet<Options>(new IdCompare());

setS.add(new Options(534000,"susheel","A"));

setS.add(new Options(322000,"Ramesh","C"));

setS.add(new Options(384000,"Ajay","B"));

System.out.println(" Increasing Order with the Id : ");

for(Options o : set)

{

System.out.print(o.getId()+","+o.getName()+","+o.getDep()+","+o.getSalary());

System.out.println();

}

break;

case 2:

TreeSet<Options> setN = new TreeSet<Options>(new NameCompare());

setS.add(new Options(534000,"susheel","A"));

setS.add(new Options(322000,"Ramesh","C"));

setS.add(new Options(384000,"Ajay","B"));

System.out.println(" Increasing Order with the Name : ");

for(Options o : setN)

{

System.out.print(o.getId()+","+o.getName()+","+o.getDep()+","+o.getSalary());

System.out.println();

}

break;

case 3:

TreeSet<Options> setD = new TreeSet<Options>(new DepComapre());

setS.add(new Options(534000,"susheel","A"));

setS.add(new Options(322000,"Ramesh","C"));

setS.add(new Options(384000,"Ajay","B"));

System.out.println(" Increasing Order with the Department : ");

for(Options o : setD)

{

System.out.print(o.getId()+","+o.getName()+","+o.getDep()+","+o.getSalary());

System.out.println();

}

break;

case 4:

TreeSet<Options> setS = new TreeSet<Options>(new SalaryCompare());

setS.add(new Options(534000,"susheel","A"));

setS.add(new Options(322000,"Ramesh","C"));

setS.add(new Options(384000,"Ajay","B"));

System.out.println(" Increasing Order with the Salary : ");

for(Options o : setS)

{

System.out.print(o.getId()+","+o.getName()+","+o.getDep()+","+o.getSalary());

System.out.println();

}

break;

}

}

}

OUTPUT: You want to sort in order of

1.ID

2.Name

3.Department

4.Salary

Enter your option:

4

Increasing Order with the Salary :

3,Ramesh,c,322000

1,Susheel,A,534000

2,Ajay,B,384000

1. Given a Linked List 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. For the date 23-12-2000

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

####### class name - LeapYr

**package** collecton\_java;

**import** java.time.LocalDate;

**import** java.time.format.DateTimeFormatter;

**import** java.util.LinkedList;

**public** **class** LeapYr {

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

LocalDate date1 = LocalDate.*of*(2008, 11, 16);

LocalDate date2 = LocalDate.*of*(2024, 8, 10);

LocalDate date3 = LocalDate.*of*(1987, 4, 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 16-11-2008 and it was a leap year

Your Date of Birth is 10-08-2024 and it was a leap year

Your Date of Birth is 23-04-1987 and it was not a leap year