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

Write solutions to

1. Fetch all the keys and print them,
2. Fetch all the values and print them
3. Print all key-values pairs

Note:

1. Contacts should be stored in descending order of phone number
2. Contact class:

* PhoneNumber:<long>
* Name:<String>
* Email:<String>
* Gender:<Enum>

**package** Tree;

**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** Tree;

**import** java.util.Set;

**import** java.util.Collections;

**import** java.util.Map;

**import** java.util.TreeMap;

**import** Tree.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**)326452139, "Rob", "rob@gmail.com",gen.***M***);

Contact c2 = **new** Contact((**long**)986312475, "Sam", "sam@gmail.com",gen.***M***);

Contact c3 = **new** Contact((**long**)694213857, "Emily", "emily@gmail.com",gen.***F***);

Contact c4 = **new** Contact((**long**)123698547, "Emma", "emma@gmail.com",gen.***F***);

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

map.put((**long**)326452139, c1);

map.put((**long**)986312475, c2);

map.put((**long**)694213857, c3);

map.put((**long**)123698547, c4);

map.put((**long**)635478941, 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:**  986312475 -->Phone Number in descending order

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

986312475 Sam sam@gmail.com M -->Full Details

694213857 -->Phone Number in descending order

Emily emily@gmail.com F -->Other Details

694213857 Emily emily@gmail.com F -->Full Details

635478941 -->Phone Number in descending order

Tom tom@gmail.com M -->Other Details

635478941 Tom tom@gmail.com M -->Full Details

326452139 -->Phone Number in descending order

Rob rob@gmail.com M -->Other Details

326452139 Rob rob@gmail.com M -->Full Details

123698547 -->Phone Number in descending order

Emma emma@gmail.com F -->Other Details

123698547 Emma emma@gmail.com F -->Full Details

1. Write an application to store unique product objects. In case there is an attempt to add duplicate product, it should be silently rejected. Hint: Use HashSet or TreeSet

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

**public** **class** Duplicate1 **implements** Comparable<Duplicate1>{

**private** String product\_name;

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

Duplicate1(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(Duplicate1 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;

}

}

**import** java.util.TreeSet;

**public** **class** Duplicate {

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

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

TreeSet<Duplicate1> dup = **new** TreeSet<>();

dup.add(**new** Duplicate1("iPhone",1));

dup.add(**new** Duplicate1("Samsung",2));

dup.add(**new** Duplicate1("Motorola",3));

//adding a duplicate product name

dup.add(**new** Duplicate1("iPhone",4));

//adding a duplicate product ID

dup.add(**new** Duplicate1("OnePlus",2));

dup.add(**new** Duplicate1("Redmi",5));

**for**(Duplicate1 f : dup)

{

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

}

}

}

**Output:** Redmi - 5

iPhone - 4

Motorola - 3

Samsung - 2

iPhone - 1

1. Store atleast 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 all print the employee details in a sorted manner.

For E.g.,

Run Application:

1. ID
2. Name
3. Department
4. Salary

Your choice: b

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

**public** **class** Emp1 {

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

**private** String name;

**private** String dept;

**private** **double** sal;

**public** Emp1(**int** id, String name, String dept, **double** sal) {

**super**();

**this**.id = id;

**this**.name = name;

**this**.dept = dept;

**this**.sal = sal;

}

**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** **double** getSal() {

**return** sal;

}

**public** **void** setSal(**double** sal) {

**this**.sal = sal;

}

}

**import** java.util.Comparator;

**public** **class** IdCompare **implements** Comparator<Emp1>{

**public** **int** compare(Emp1 o1, Emp1 o2)

{

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

}

}

**import** java.util.Comparator;

**public** **class** NCompare **implements** Comparator<Emp1>{

**public** **int** compare(Emp1 o1, Emp1 o2)

{

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

}

}

**import** java.util.Comparator;

**public** **class** DCompare **implements** Comparator<Emp1>{

**public** **int** compare(Emp1 o1, Emp1 o2)

{

**return** o1.getDept().compareTo(o2.getDept());

}

}

**import** java.util.Comparator;

**public** **class** SCompare **implements** Comparator<Emp1>{

**public** **int** compare(Emp1 o1, Emp1 o2)

{

**return** o1.getDept().compareTo(o2.getDept());

}

}

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

**import** java.util.TreeSet;

**public** **class** Compare {

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

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

Scanner sc = **new** Scanner(System.***in***);

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

**int** option = sc.nextInt();

**switch**(option)

{

**case** 1:

TreeSet<Emp1> tset = **new** TreeSet<Emp1>(**new** IdCompare());

tset.add(**new** Emp1(1,"Trainee","Sam",18000.0));

tset.add(**new** Emp1(2,"Manager","Rob",32000.0));

tset.add(**new** Emp1(3,"Analyst","Tom",21000.0));

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

**for**(Emp1 o : tset)

{

System.***out***.print(o.getId()+","+o.getDept()+","+o.getName()+","+o.getSal());

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

}

**break**;

**case** 2:

TreeSet<Emp1> tset1 = **new** TreeSet<Emp1>(**new** NCompare());

tset1.add(**new** Emp1(1,"Trainee","Sam",18000.0));

tset1.add(**new** Emp1(2,"Manager","Rob",32000.0));

tset1.add(**new** Emp1(3,"Analyst","Tom",21000.0));

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

**for**(Emp1 o : tset1)

{

System.***out***.print(o.getId()+","+o.getDept()+","+o.getName()+","+o.getSal());

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

}

**break**;

**case** 3:

TreeSet<Emp1> tset2 = **new** TreeSet<Emp1>(**new** DCompare());

tset2.add(**new** Emp1(1,"Trainee","Sam",18000.0));

tset2.add(**new** Emp1(2,"Manager","Rob",32000.0));

tset2.add(**new** Emp1(3,"Analyst","Tom",21000.0));

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

**for**(Emp1 o : tset2)

{

System.***out***.print(o.getId()+","+o.getDept()+","+o.getName()+","+o.getSal());

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

}

**break**;

**case** 4:

TreeSet<Emp1> tset3 = **new** TreeSet<Emp1>(**new** SCompare());

tset3.add(**new** Emp1(1,"Trainee","Sam",18000.0));

tset3.add(**new** Emp1(2,"Manager","Rob",32000.0));

tset3.add(**new** Emp1(3,"Analyst","Tom",21000.0));

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

**for**(Emp1 o : tset3)

{

System.***out***.print(o.getId()+","+o.getDept()+","+o.getName()+","+o.getSal());

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

}

**break**;

}

}

}

**Output:** You want to sort in order of

1.ID

2.Department

3.Name

4.Salary

Enter your option:

2

Increasing Order with the Name :

3,Tom,Analyst,21000.0

2,Rob,Manager,32000.0

1,Sam,Trainee,18000.0

You want to sort in order of

1.ID

2.Department

3.Name

4.Salary

Enter your option:

3

Increasing Order with the Department :

2,Rob,Manager,32000.0

1,Sam,Trainee,18000.0

3,Tom,Analyst,21000.0

1. 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 and it was a leap year.

b) For the date 23-12-2001

Your date of birth 23-12-2001 and it was not a leap year

**Note:** You need to access the Dates in the reverse order, i.e. start from the last object and move towards the first object.

**import** java.time.LocalDate;

**import** java.util.ArrayList;

**import** java.util.LinkedList;

**import** java.util.List;

**import** java.util.Calendar;

**import** java.time.LocalDateTime;

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

**public** **class** LeapYear {

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

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

LocalDate cal1 = LocalDate.*of*(2000, 07, 26);

LocalDate cal2 = LocalDate.*of*(2017, 11, 17);

LinkedList<LocalDate> callist = **new** LinkedList<LocalDate>();

callist.add(cal1);

callist.add(cal2);

**for**(LocalDate c: callist)

{

String fDate = c.format(DateTimeFormatter.*ofPattern*("dd-MM-YYYY"));

**if**(c.isLeapYear())

{

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

}

**else**

{

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

}

}

}

}

**Ouput:** Your Date of Birth 26-07-2000 and it was a leap year

Your Date of Birth 17-11-2017 and it was not a leap year