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

Write solution to

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

Note:

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

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

Ans🡪

**package** org.collection.java;

**import** java.util.Collections;

**import** java.util.Map;

**import** java.util.TreeMap;

**import** org.collection.java.Contact.gender;

**public** **class** Tree {

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

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

Contact c1 = **new** Contact((**long**)45678425,"Ansh","Ansh@.com",gender.***M***);

Contact c2 = **new** Contact((**long**)75688425,"Ayank","Ayank@.com",gender.***M***);

Contact c3 = **new** Contact((**long**)12568785,"Aavani","Aavani@.com",gender.***F***);

Contact c4 = **new** Contact((**long**)75869312,"Krish","Krish@.com",gender.***M***);

map.put((**long**)45678425, c1);

map.put((**long**)75688425, c2);

map.put((**long**)12568785, c3);

map.put((**long**)75869312, c4);

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

Long key=entry.getKey();

Contact c= entry.getValue();

System.***out***.println(key+" Details: ");

System.***out***.println(c.phoneno+" "+c.name+" "+c.email+" "+c.g);

}

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

System.***out***.println("After Sorted: ");

Map<Long, Contact> sortedMapDesc = **new** TreeMap<>(Collections.*reverseOrder*());

sortedMapDesc.putAll(map);

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

Long key = entry1.getKey();

Contact c8= entry1.getValue();

System.***out***.println(key+" Details:");

System.***out***.println(c8.phoneno+" "+c8.name+" "+c8.email+" "+c8.g );

}

}

}

**package** org.collection.java;

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

**long** phoneno;

String name, email;

**public** **enum** gender {***F***,***M***};

gender g;

**public** Contact(**long** phoneno, String name, String email, gender g) {

**super**();

**this**.phoneno = phoneno;

**this**.name = name;

**this**.email = email;

**this**.g = g;

}

**public** **long** getPhoneno() {

**return** phoneno;

}

**public** **void** setPhoneno(**long** phoneno) {

**this**.phoneno = phoneno;

}

**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** gender getG() {

**return** g;

}

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

**this**.g = g;

}

}

Output:

12568785 Details:

12568785 Aavani Aavani@.com F

45678425 Details:

45678425 Ansh Ansh@.com M

75688425 Details:

75688425 Ayank Ayank@.com M

75869312 Details:

75869312 Krish Krish@.com M

-----------------

After Sorted:

75869312 Details:

75869312 Krish Krish@.com M

75688425 Details:

75688425 Ayank Ayank@.com M

45678425 Details:

45678425 Ansh Ansh@.com M

12568785 Details:

12568785 Aavani Aavani@.com F

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. Use HasSet or TreeSet.

Ans🡪

**package** org.collection.java;

**import** java.util.TreeSet;

**public** **class** Example2 {

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

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

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

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

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

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

//adding a duplicate product Id

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

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

**for**(Function f : func) {

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

}

}

}

**package** org.collection.java;

**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;

}

}

Output:

Scale - 4

Sharpner - 3

Pen - 2

Pencil - 1

1. 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

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

Your choice: b

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

Ans🡪

**import** java.util.TreeSet;

**public** **class** Example3 {

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

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

Employee emp1= **new** Employee(1,"Chaitu","Software",30000);

Employee emp2= **new** Employee(2,"Ayank","developer",25000);

Employee emp3= **new** Employee(3,"Aadi","Analyst",20000);

Employee emp4= **new** Employee(4,"Aadesh","software",35000);

Employee emp5= **new** Employee(5,"Ansh","Analyst",25000);

Employee emp6= **new** Employee(6,"Jyoti","programmer",45000);

Employee emp7= **new** Employee(7,"neha","developer",40000);

Employee emp8= **new** Employee(8,"hema","Software",48000);

Employee emp9= **new** Employee(9,"Anju","Analyst",53000);

Employee emp10= **new** Employee(10,"Krish","programmer",10000);

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);

}

}

}

**package** org.collection.java;

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

**int** id;

String Name;

String Department;

**int** Salary;

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

**super**();

**this**.id = id;

Name = name;

Department = department;

Salary = salary;

}

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

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

**return** 1;

}

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

**return** -1;

}

**else** {

**return** 0;

}

}

}

Output:

Chaitu

Ayank

Aadi

Aadesh

Ansh

Jyoti

neha

hema

Anju

Krish

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: You 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

1. For the date 23-12-2001

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

Ans🡪

**package** org.collection.java;

**import** java.time.LocalDate;

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

**import** java.util.LinkedList;

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

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

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

LocalDate date2 = LocalDate.*of*(1998, 01, 03);

LocalDate date3 = LocalDate.*of*(2001, 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 03-01-1998 and it was not a leap year

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