Collection Assignment-6

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

Write solutions to

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

Note:

1. Contact should be sorted in descending order of phone number
2. Contact Class:

PhoneNumber : <long>

Name : <String>

Email : <String>

Gender : <Enum>

Class1 : Contact.java

**package** org.collection.app;

**import** java.util.EnumSet;

**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;

}

}

Class : TreeMapDemo

**package** org.collection.app;

**import** java.util.Set;

**import** java.util.Collections;

**import** java.util.Map;

**import** java.util.TreeMap;

**import** org.collection.app.Contact.gen;

**public** **class** TreeMapDemo

{

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

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

Contact c1 = **new** Contact((**long**)889702956, "Jack", "Jack@gmail.com", gen.***M***);

Contact c2 = **new** Contact((**long**)986689189, "Chinnu", "Chinnu@gmail.com",gen.***F***);

Contact c3 = **new** Contact((**long**)694212857, "Manu", "manu@gmail.com",gen.***F***);

Contact c4 = **new** Contact((**long**)143698547, "Rita", "Rita@gmail.com",gen.***F***);

Contact c5 = **new** Contact((**long**)632478941, "Raju", "raju@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

Chinnu Chinnu@gmail.com F -->Other Details

986689189 Chinnu Chinnu@gmail.com F -->Full Details

694213857 -->Phone Number in descending order

Manu manu@gmail.com F -->Other Details

694212857 Manu manu@gmail.com F -->Full Details

635478941 -->Phone Number in descending order

Raju raju@gmail.com M -->Other Details

632478941 Raju raju@gmail.com M -->Full Details

326452139 -->Phone Number in descending order

Jack Jack@gmail.com M -->Other Details

889702956 Jack Jack@gmail.com M -->Full Details

123698547 -->Phone Number in descending order

Rita Rita@gmail.com F -->Other Details

143698547 Rita Rita@gmail.com F -->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. Hint: Use Hashset or TreeSet.

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

**package** org.collection.app;

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

**public** **class** Duplicate

{

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

{

**int** arr[] = {5,4,0,5,2,2,1,4};

ArrayList<Integer> a = **new** ArrayList<>();

HashSet<Integer> hs = **new** HashSet<>();

**for**(**int** i : arr)

{

**if**(!hs.contains(i))

{

a.add(i);

hs.add(i);

}

}

**for** (**int** i : a)

{

System.***out***.print(i + " ");

}

}

}

OUTPUT:

5 4 0 2 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.

a. ID

b. Name

c. Department

d.Salary

Your choice : b

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

Class1 : Emp1

**package** org.collection.app;

**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;

}

}

Class 2 : Dcompare

**package** org.collection.app;

**import** java.util.Comparator;

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

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

{

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

}

}

Class3 : Idcompare

**package** org.collection.app;

**import** java.util.Comparator;

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

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

{

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

}

}

Class4 : Ncompare

**package** org.collection.app;

**import** java.util.Comparator;

**public** **class** Ncompare **implements** Comparator<Emp1>

{

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

{

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

}

}

Class5 : SCompare

**package** org.collection.app;

**import** java.util.Comparator;

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

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

{

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

}

}

Class6 : Compare

**package** org.collection.app;

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

**import** java.util.TreeSet;

**public** **class** Compare

{

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

{

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

1. Given a LinkedList of Objects representing date of birth’s(use any inbulid java class to represents 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.

1. For the date 23-12-2000

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

**package** org.collection.app;

**import** java.time.LocalDate;

**import** java.time.LocalDateTime;

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

**import** java.time.format.FormatStyle;

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

**public** **class** Leap

{

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

{

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

List<LocalDate> calendarList = **new** LinkedList<>();

calendarList.add(cal);

**for** (LocalDate c : calendarList)

{

String formattedDate = c.format(DateTimeFormatter.*ofPattern*("dd-MM-yyyy"));

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

{

System.***out***.println("Your Date is "+formattedDate + " is a leap year");

} **else**

{

System.***out***.println(("Your Date is "+formattedDate + " is not a leap year");

}

}

}

}

OUTPUT:

Your Date is 23-12-2000 is 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

**package** org.collection.app;

**import** java.time.LocalDate;

**import** java.time.LocalDateTime;

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

**import** java.time.format.FormatStyle;

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

**public** **class** Leap

{

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

{

LocalDate cal = LocalDate.*of*(2001, 12, 23);

List<LocalDate> calendarList = **new** LinkedList<>();

calendarList.add(cal);

**for** (LocalDate c : calendarList)

{

String formattedDate = c.format(DateTimeFormatter.*ofPattern*("dd-MM-yyyy"));

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

{

System.***out***.println(("Your Date is "+formattedDate + " is a leap year");

} **else**

{

System.***out***.println(("Your Date is "+formattedDate + " is not a leap year");

}

}

}

}

OUTPUT :

Your Date is 23-12-2001 is not a leap year