1. Given a TreeMap<Long,Contact> which has phone number 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 descending order of phone number
2. Contact Class:

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

Ans)

**public class** Contact{

String name,email;

**long** phonenumber; **enum** Gender{

***Female***,***Male***,***Other***;

}

Gender gender;

**public** Contact(String name,String email,Gender gender) {

**this**.name=name;

**this**.email=email;

**this**.gender=gender;

**this**.phonenumber=phonenumber;

}

**public** String toString() {

**return** **this**.name+' '+**this**.email+' '+**this**.gender;

}

**public** **class** User {

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

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

c.put(9980986778L,**new** Contact("Ram","ram@gmail.com",Contact.Gender.***Male***));

c.put(9546778886L,**new** Contact("sam","sam@gmail.com",Contact.Gender.***Male***));

c.put(9980986790L,**new** Contact("John","John@gmail.com",Contact.Gender.***Male***));

c.put(9980986723L,**new** Contact("Anu","anu@gmail.com",Contact.Gender.***Female***));

List<Long> keys=**new** ArrayList<Long>();

Collections.*sort*(keys);

Collections.*reverse*(keys);

Set<Map.Entry<Long, Contact>> values=c.entrySet();

List<Map.Entry<Long,Contact>> entryList=**new** ArrayList<>(values);

**for**(Map.Entry<Long, Contact> e : values) {

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

}

}

}

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

Ans) **public** **class** Demo {

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

TreeSet<Product> ts=**new** TreeSet<Product>();

ts.add(**new** Product("ABC",1234));

ts.add(**new** Product("ABC",2452));

ts.add(**new** Product("xyz",5689));

ts.add(**new** Product("pqr",1234));

ts.add(**new** Product("nsh",2878));

ts.add(**new** Product("csz",5667));

ts.add(**new** Product("ABC",1234));

ts.add(**new** Product("ABC",1234));

ts.add(**new** Product("ABC",1234));

ts.add(**new** Product("ABC",1234));

**for**(Product p:ts) {

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

}

}

}

**class** Product{

String productname;

**int** price;

**public** Product(String productname,**int** price) {

**this**.productname=productname;

**this**.price=price;

}

**public** String getProductname() {

**return** productname;

}

**public** **void** setProductname(String productname) {

**this**.productname = productname;

}

**public** **int** getPrice() {

**return** price;

}

**public** **void** setPrice(**int** price) {

**this**.price = price;

}

**public** String toString() {

**return** "Productname: "+**this**.productname+" Price: "+**this**.price;

}

**public** **int** compareTo() {

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

**return** 0;

}

}

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 employees details sorted by name>

Ans)

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

**private** String name,Department;

**private** **int** Id,salary;

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

**super**();

**this**.Id=id;

**this**.name=name;

**this**.Department=Department;

**this**.salary=salary;

}

**public** String getName() {

**return** name;

}

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

**this**.name = name;

}

**public** String getDepartment() {

**return** Department;

}

**public** **void** setDepartment(String department) {

Department = department;

}

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

**return** Id;

}

**public** **void** setId(**int** id) {

Id = id;

}

**public** **int** getSalary() {

**return** salary;

}

**public** **void** setSalary(**int** salary) {

**this**.salary = salary;

}

**public** **void** display() {

System.***out***.println("Id: "+getId()+" Empname "+getName()+" Department "+getDepartment()+" salary "+getSalary());

}

@Override

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

**return** emp.getName().compareTo(**this**.getName());

}

}

**public** **class** TreeSetExample {

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

Set<Employee> ts= **new** TreeSet<Employee>().descendingSet();

Employee e1=**new** Employee(12,"John","IT",25000);

Employee e2=**new** Employee(24,"Sam","CSE",35000);

Employee e3=**new** Employee(11,"Ann","Testing",57000);

Employee e4=**new** Employee(45,"David","IT",25000);

Employee e5=**new** Employee(89,"Anamika","Analyst",45000);

Employee e6=**new** Employee(15,"Arjun","Developer",35000);

Employee e7=**new** Employee(16,"Sanjna","Programmer",35000);

Employee e8=**new** Employee(22,"Akash","Developer",55000);

Employee e9=**new** Employee(23,"Abhay","Mechanic",75000);

Employee e0=**new** Employee(33,"Hari","Analyst",35000);

ts.add(e1);

ts.add(e2);

ts.add(e3);

ts.add(e4);

ts.add(e5);

ts.add(e6);

ts.add(e7);

ts.add(e8);

ts.add(e9);

ts.add(e0);

**for**(Employee emp:ts) {

emp.display();

}

}

}

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 leapyear.

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

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

Ans)

**import** java.time.LocalDate;

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

**import** java.util.LinkedList;

**import** java.util.List;

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

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

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

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

list.add(date);

**for** (LocalDate c : list) {

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

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

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

}

**else** {

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

}

}

}

}