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

   Write solutions to

   a. Fetch all the keys and print them,
   b. Fetch all the values and print them
   c. Print all key-value pairs

   Note:

   a) Contacts should be stored in descending order of phone number
   b) Contact Class:
      - PhoneNumer: <long>
      - Name: <String>
      - Email: <String>
      - Gender: <Enum>

```java
package Assignment6;
import java.util.Collections;
import java.util.Map;
import java.util.TreeMap;
class Contact{
    private long phNumber;
    private String name;
    private String email;
    private gender g;
    enum gender{
        Male,
        Female;
    }
    public Contact(long phNumber, String name, String email, Contact.gender g) {
        super();
        this.phNumber = phNumber;
        this.name=name;
        this.email=email;
        this.g=g;
    }
    public long getPhNumber() {
        return phNumber;
    }
    public String getName() {
        return name;
    }
    public String getEmail() {
        return email;
    }
    public gender getG() {
        return g;
    }
}
public class Collection1 {
     public static void main(String[] args) {
            Contact c1 = new Contact(1234567890, "abc", "abc@gmail.com", Contact.gender.Female);
```

```
            Contact c2 = new Contact(1234567891, "bcd", "bcd@gmail.com", Contact.gender.Male);
            Contact c3 = new Contact(1234567892, "cde", "cde@gmail.com", Contact.gender.Female);

            Map<Long,Contact> c = new TreeMap<>(Collections.reverseOrder());
            c.put(c1.getPhNumber(),c1);
            c.put(c2.getPhNumber(),c2);
            c.put(c3.getPhNumber(),c3);

            for(long l: c.keySet()) {
                System.out.println(l);
            }
            for(Contact l: c.values()) {
                System.out.println(l.getName());
                System.out.println(l.getEmail());
                System.out.println(l.getG());
            }
            System.out.println(c);
            for(Map.Entry<Long, Contact> entry : c.entrySet()) {
                System.out.println("Key: "+entry.getKey());
                System.out.println("Name "+entry.getValue().getName());
                System.out.println("Email "+entry.getValue().getEmail());
                System.out.println("Gender "+entry.getValue().getG());
                System.out.println();
            }

    }
}
```

```
<terminated> Collection1 [Java Application] C:\Users\valaanus\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.5.v20221102-0933\jre\bin\javaw.exe  (15-Dec-202
1234567892
1234567891
1234567890
cde
cde@gmail.com
Female
bcd
bcd@gmail.com
Male
abc
abc@gmail.com
Female
{1234567892=Assignment6.Contact@41a4555e, 1234567891=Assignment6.Contact@3830f1c0, 1234567890=Assignment6.Contact@39ed3c8d}
Key: 1234567892
Name cde
Email cde@gmail.com
Gender Female

Key: 1234567891
Name bcd
Email bcd@gmail.com
Gender Male

Key: 1234567890
Name abc
Email abc@gmail.com
Gender Female
```

2) 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** HasSet or TreeSet

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

```
package Assignment6;
import java.util.*;
public class Frame2 {
        public static void main(String[] args) {
            HashSet<String> set= new HashSet<String>();
            set.add("Apple");
            set.add("Mango");
            set.add("banana");
            set.add("Apple");
            Iterator<String> itr=set.iterator();
            while(itr.hasNext())
            {
            System.out.println(itr.next());
            }
        }
}
```

```
<terminated> Frame2 [Java Application] C:\Users\valaanus\.p2\
banana
Apple
Mango
```

3) 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.4

For E.g.

Run Application:

a) ID
b) Name
c) Department
d) Salary

Your choice: b

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

```java
package Assignment6;
import java.util.Comparator;
import java.util.Scanner;
import java.util.Set;
import java.util.TreeSet;
class Employee {
    private int id;
    private String name;
    private String Department;
    private int Salary;
    public Employee(int id, String name, String Department, int Salary) {

        this.id=id;
        this.name=name;
        this.Department=Department;
        this.Salary=Salary;
    }
    public int getId() {
        return id;
    }
    public String getname() {
        return name;
    }
    public String getDepartment() {
        return Department;
    }
    public int getSalary() {
        return Salary;
    }
}
class idComparator implements Comparator<Employee>{
    public int compare(Employee e1,Employee e2) {
        return e1.getId()-e2.getId();
    }
}
class nameComparator implements Comparator<Employee>{
```

```
class nameComparator implements Comparator<Employee>{
    public int compare(Employee e1,Employee e2) {
        return e1.getname().compareTo(e2.getname());
    }
}
class depComparator implements Comparator<Employee>{
    public int compare(Employee e1,Employee e2) {
        return e1.getDepartment().compareTo(e2.getDepartment());
    }
}
class salComparator implements Comparator<Employee>{
    public int compare(Employee e1,Employee e2) {
        return e1.getSalary()-e2.getSalary();
    }
}
public class Collection3 {
     public static void main(String[] args) {
            Employee e1=(new Employee(18401,"Gracely","IT",35000));
            Employee e2=(new Employee(18489,"Akhila","MT",32000));
            Employee e3=(new Employee(18490,"Sravani","KT",29000));
            Employee e4=(new Employee(18407,"Snehitha","IT",28000));
            Employee e5=(new Employee(18402,"Mahi","HT",40000));
            Employee e6=(new Employee(18403,"Niha","IT",31000));
            Employee e7=(new Employee(18404,"Prathima","ST",20000));
            Employee e8=(new Employee(18405,"Divya","MT",27000));
            Employee e9=(new Employee(18406,"Usha","DT",26000));
            Employee e10=(new Employee(18407,"Kavya","CT",45000));
            Set<Employee> ts1;
            Scanner sc=new Scanner(System.in);
            System.out.println("Enter 1. Id 2. Name, 3.Department, 4.Salary ");
            int n=sc.nextInt();
            sc.close();
            if(n==1) {
                ts1=new TreeSet<>(new idComparator());
            }
            else if(n==2) {
                ts1=new TreeSet<>(new nameComparator());
            }
            else if(n==3) {
                ts1=new TreeSet<>(new depComparator());
            }
            else {
                ts1=new TreeSet<>(new salComparator());
            }
            ts1.add(e1);
            ts1.add(e2);
            ts1.add(e3);
            ts1.add(e4);
            ts1.add(e5);
            ts1.add(e6);
            ts1.add(e7);
            ts1.add(e8);
            ts1.add(e9);
            ts1.add(e10);
            for(Employee e:ts1) {
                System.out.println(e.getId()+" "+e.getname()+" "+e.getDepartment()+" "+e.getSalary());
            }
     }
}
```

```
<terminated> Collection3 [Java Application] C:\Users\valaanus\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspc
Enter 1. Id 2. Name, 3.Department, 4.Salary
1
18401 Gracely IT 35000
18402 Mahi HT 40000
18403 Niha IT 31000
18404 Prathima ST 20000
18405 Divya MT 27000
18406 Usha DT 26000
18407 Snehitha IT 28000
18489 Akhila MT 32000
18490 Sravani KT 29000
```

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 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

c) 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

```java
package Assignment6;
import java.util.Calendar;
import java.util.GregorianCalendar;
import java.util.LinkedList;
class cal{
    int day;
    int month;
    int year;
    Calendar c;
    public cal(Calendar c) {
        day=c.get(Calendar.DAY_OF_MONTH);
        month=c.get(Calendar.MONTH);
        year=c.get(Calendar.YEAR);
        this.c=c;
    }
    public int getDay() {
        return day;
    }
    public int getMonth() {
        return month;
    }
    public int getYear() {
        return year;
    }
    public String getCal() {
        String str=Integer.toString(day)+"-"+Integer.toString(month)+"-"+Integer.toString(year);
        return str;
    }
}
public class Collection4 {
    public static void main(String[] args) {
        cal c=new cal(new GregorianCalendar(2000,5,2));
        cal c1=new cal(new GregorianCalendar(1999,9,23));
        cal c2=new cal(new GregorianCalendar(1998,7,18));
        cal c3=new cal(new GregorianCalendar(1997,11,18));
        cal c4=new cal(new GregorianCalendar(1998,12,18));

        LinkedList<cal> ll=new LinkedList<>();
        ll.add(c);
        ll.add(c1);
        ll.add(c2);
        ll.add(c3);
        ll.add(c4);
        for(cal dob:ll) {
            int year=dob.getYear();
            if((year%400==0)||((year%4==0)&&(year%100!=0))) {
                System.out.println("Your dob is "+dob.getCal()+" and it was a leap year");
            }
            else
            {
                 System.out.println("Your dob is "+dob.getCal()+" and it was not a leap year");
            }
        }
    }
            }
```

```
Your dob is 2-5-2000 and it was a leap year
Your dob is 23-9-1999 and it was not a leap year
Your dob is 18-7-1998 and it was not a leap year
Your dob is 18-11-1997 and it was not a leap year
Your dob is 18-0-1999 and it was not a leap year
```