**ASSIGNMENT ON COLLECTIONS**

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-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>

public class first

{

public static void main(String args[])

{

TreeMap<Integer,String> t = new TreeMap();

t.put(3,"DINESH");

t.put(2,"PRAVEEN");

t.put(1,"KARTHICK");

System.out.println("Keys");

Set keys = tm.keySet();

Iterator i = keys.iterator();

while (i.hasNext())

{

System.out.println(i.next());

}

System.out.println("Values: ");

Collection getValues = t.values();

i = getValues.iterator();

while (i.hasNext())

{

System.out.println(i.next());

}

System.out.println("Map = "+t);

}

}

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)

Public class Duplicate implements Comparable<Duplicate>

|  |
| --- |
|  |
|  | private String prod\_name; |
|  | private int prod\_id; |
|  |  |
|  | Duplicate(String prod\_name, int prod\_id) |
|  | { |
|  | this.prod\_id = prod\_id; |
|  | this.prod\_name = prod\_name; |
|  | } |
|  |  |
|  | private String getName() |
|  | { |
|  | return prod\_name; |
|  | } |
|  |  |
|  | public int getId() |
|  | { |
|  | return prod\_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;

}

}

|  |
| --- |
|  |

|  |  |
| --- | --- |
|  |  |
|  | import java.util.TreeSet; |
|  |  |
|  | public class Duplicat { |
|  |  |
|  | public static void main(String[] args) { |
|  |  |
|  | TreeSet<Duplicat> d = new TreeSet<>(); |
|  |  |
|  | d.add(new Duplicate("Redmi",1)); |
|  | d.add(new Duplicate("Poco",2)); |
|  | d.add(new Duplicate("LG",3)); |
|  |  |
|  |  |
|  |  |
|  |  |
|  | for(Duplicate f:d)  {  System.out.println(f);  }  }}}  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.  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>  class Employee implements Comparable<Employee>{  int id;  String Name;  String Department;  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 compareTo(Employee e) {  if(id>e.id) {  return 1;  }  else if(id<e.id)  {  return -1;  }  else {  return 0;  }  }  }  public class second {  public static void main(String[] args) {  Set <Employee>set=new TreeSet<Employee>();  Employee emp1=new Employee(1,"DINESH","Developer",35000);  Employee emp2=new Employee(2,"PRAVEEN","Enginneer",35000);  Employee emp3=new Employee(3,"KARTHIKA","Programmer",27000);  Employee emp4=new Employee(4,"DINA","Analyst",45000);  Employee emp5=new Employee(5,"UDAY","Software",15000);  Employee emp6=new Employee(6,"RAHUL","Assistant Engineer",45000);  Employee emp7=new Employee(7,"DRAVID”,"Project Engineer",25000);  Employee emp8=new Employee(8,"SACHIN","Senior Analyst",5000);  Employee emp9=new Employee(9,"SAM","Programer Analyst",15000);  Employee emp10=new Employee(10,"PRIYANKA","Test Engineer",25000);    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);  }  }  }   |  | | --- | |  | |  | |  | |  |  | |  | |  |  | |

4. 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

b)For the date 23-12-2001

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

|  |
| --- |
|  |
|  | 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) { |
|  |  |
|  | LocalDate l1 = LocalDate.of(1999, 12, 05); |
|  | LocalDate l2 = LocalDate.of(2018, 12, 11); |
|  | LinkedList<LocalDate> callist = new LinkedList<LocalDate>(); |
|  | callist.add(l1); |
|  | callist.add(l2); |
|  | 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"); |
|  | } |
|  |
| }}} |
|  |
|  |