**CORE JAVA ASSIGNMENT 6 – COLLECTIONS**

**Assignment on Collection Framework**

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:

1. Contacts should be stored in descending order of phone number
2. Contact Class:
   1. PhoneNumer: <long>
   2. Name: <String>
   3. Email: <String>
   4. Gender: <Enum>

**Solution:**

package com.collectionsassignments;

import java.util.TreeMap;

enum Gender{

male,female}

class Contact{

long phoneNumber;

String name;

String email;

Gender gender;

Contact(long no,String name,String email,Gender gender)

{

this.email=email;

this.name = name;

this.phoneNumber = no;

this.gender = gender; }s

@Override

public String toString() {

return "\n\nContact " +

" phoneNumber=" + phoneNumber +

",\n name='" + name + '\'' +

",\n email='" + email + '\'' +

",\n gender=" + gender +

'}';

}

}

public class Main {

public static void main(String[] args) {

TreeMap<Long,Contact> data = new TreeMap<>();

data.put(5000000000L,new Contact(5000000000L,"Naman","namanchauhan@gmail.com",Gender.male));

data.put(4000000000L,new Contact(4000000000L,"Rahul","rahul@gmail.com",Gender.male));

data.put(3000000000L,new Contact(3000000000L,"Rohit","rohit@gmail.com",Gender.male));

data.put(2000000000L,new Contact(2000000000L,"neha","neha@gmail.com",Gender.female));

System.out.println(data.keySet());

System.out.println(data.values());

System.out.println(data);

}

}

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 Hashset or TreeSet

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

**Solution:**

package com.collectionsassignments;

import java.util.HashSet;

class Product{

private String productName;

Product(String productName)

{

this.productName = productName;

}

public String getProductName() {

return productName;

}

}

public class AssignmentQues2 {

public static void main(String[] args) {

HashSet<Product> productsSet = new HashSet<>(10);

Product A = new Product("A");

productsSet.add(A);

Product B = new Product("B");

productsSet.add(B);

Product C = new Product("C");

productsSet.add(C);

Product D = new Product("D");

productsSet.add(D);

Product E = new Product("E");

productsSet.add(E);

Product F = new Product("F");

productsSet.add(F);

Product G = new Product("G");

productsSet.add(G);

Product H = new Product("H");

productsSet.add(H);

Product I = new Product("I");

productsSet.add(I);

Product J = new Product("J");

productsSet.add(J);

productsSet.add(J);

for(Product str : productsSet)

{

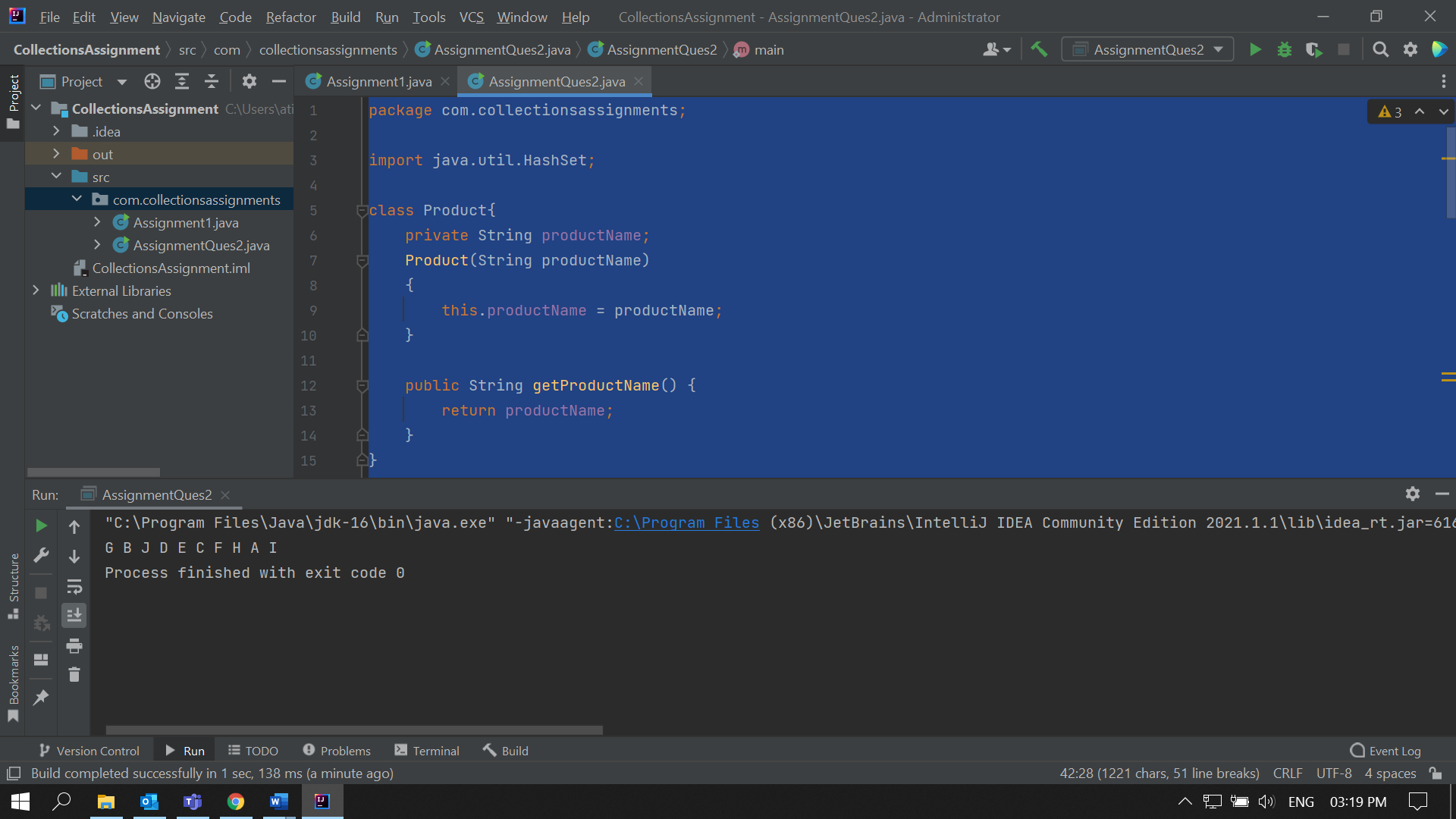
System.out.print(str.getProductName()+ " ");

}

}

}

Output:



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>

**Solution:**

package com.collectionsassignments;

import java.util.Comparator;

import java.util.Scanner;

import java.util.TreeSet;

class 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 getId() {

return id;

}

public String getName() {

return name;

}

public String getDepartment() {

return department;

}

public int getSalary() {

return salary;

}

}

public class Assignment3 {

public static void main(String[] args) {

System.out.println("1. ID");

System.out.println("2. Name");

System.out.println("3. Department");

System.out.println("4. Salary");

Scanner input = new Scanner(System.in);

System.out.print("Select option: ");

int choice = input.nextInt();

TreeSet<Employee> employeeSet = null;

switch (choice)

{

case 1: employeeSet = new TreeSet(new IDComparator());

break;

case 2: employeeSet = new TreeSet(new NameComparator());

break;

case 3: employeeSet = new TreeSet(new DepartmentComparator());

break;

case 4: employeeSet = new TreeSet(new SalaryComparator());

break;

}

employeeSet.add(new Employee(1, "ABC" , "Dept - B",10000));

employeeSet.add(new Employee(4, "LMN" , "Dept - C",12000));

employeeSet.add(new Employee(5, "XYZ" , "Dept - X",10400));

employeeSet.add(new Employee(2, "EFG" , "Dept - F",140000));

employeeSet.add(new Employee(6, "IJK" , "Dept - G",25000));

employeeSet.add(new Employee(7, "BCD" , "Dept - K",5000));

employeeSet.add(new Employee(8, "UVW" , "Dept - L",65000));

employeeSet.add(new Employee(3, "OPQ" , "Dept - U",76500));

employeeSet.add(new Employee(9, "RST" , "Dept - T",43000));

employeeSet.add(new Employee(10, "NAME" , "Dept - S",28000));

for(Employee ele : employeeSet)

{

System.out.print(ele.getId() + " " + ele.getName() + " " +ele.getDepartment()+ " "+ ele.getSalary());

System.out.println();

}

}

}

class IDComparator implements Comparator< Employee > {

@Override

public int compare(Employee o1, Employee o2) {

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

}

}

class NameComparator implements Comparator< Employee > {

@Override

public int compare(Employee o1, Employee o2) {

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

}

}

class DepartmentComparator implements Comparator< Employee > {

@Override

public int compare(Employee o1, Employee o2) {

return o1.getDepartment().compareTo(o2.getDepartment());

}

}

class SalaryComparator implements Comparator< Employee > {

@Override

public int compare(Employee o1, Employee o2) {

return o1.getSalary()-o2.getSalary();

}

}

4) Given a Linked List 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

**Solution:**  
  
import java.util.LinkedList;

class Date

{

String birthDate;

int year;

Date(String date)

{

this.birthDate = date;

String[] strings = date.split("-");

year = Integer.parseInt(strings[2]);

}

@Override

public String toString() {

if (((year % 4 == 0) && (year % 100!= 0)) || (year%400 == 0))

return "Your date of birth is "+birthDate+" and it was a leap year\n";

else

return "Your date of birth is "+birthDate+" and it was not a leap year\n";

}

}

public class BirthDates {

public static void main(String[] args) {

LinkedList<Date> list = new LinkedList<>();

list.add(new Date("11-10-1999"));

list.add(new Date("14-11-2001"));

list.add(new Date("11-10-2020"));

System.out.println(list);

}

}