**Core Java Assignment 6**

**1**

Contact.java

**package** Collection\_assignment;

**public** **class** Contact {

**long** pNumber;

String name;

String email;

String gender;

**public** Contact(**long** pNumber, String name, String email, String gender) {

**super**();

**this**.pNumber = pNumber;

**this**.name = name;

**this**.email = email;

**this**.gender = gender;

}

@Override

**public** String toString() {

**return** "[Number=" + pNumber + ", Name=" + name + ", Email=" + email + ", Gender=" + gender + "]" + "\n";

}

}

TreeMap.java

**package** Collection\_assignment;

**import** java.util.Collections;

**import** java.util.Map;

**import** java.util.TreeMap;

**public** **class** TreeMap {

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

Contact obj1 = **new** Contact( 9553454764L, "Vinod" , "vv@gmail.com" , "Male");

Contact obj2 = **new** Contact( 9121971719L, "Shiva" , "vs@gmail.com" , "Male");

Contact obj3 = **new** Contact( 9502376475L, "Teja" , "nt@gmail.com" , "Female");

Map<Long, Contact> data = **new** TreeMap<Long, Contact>(Collections.*reverseOrder*());

//the treemap by default keeps the key sorted in ascending

//so to sort in descending order we have to pass Collections.reverseOrder() to the constructor

data.put(1234L, obj1);

data.put(123456780L, obj2);

data.put(1238244L, obj3);

System.***out***.println("Keys : "+data.keySet() + "\n");

System.***out***.println("Values : " + data.values() + "\n");

System.***out***.println(data + "\n");

}

}

**2**

**package** Collection\_assignment;

**import** java.util.Iterator;

**import** java.util.Set;

**import** java.util.TreeSet;

**public** **class** UniqueProducts {

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

Products product1 = **new** Products("A");

Products product2 = **new** Products("B");

Products product3 = **new** Products("C");

Set<Products> unique = **new** TreeSet<>();

unique.add(product1);

unique.add(product2);

unique.add(product3);

unique.add(product1); //Try to add duplicate object it get replaced with the Similar one

Iterator<Products> i= unique.iterator();

**while**(i.hasNext())

{

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

}

}

}

**class** Products **implements** Comparable<Products>{

String name;

**public** Products(String name) {

**super**();

**this**.name = name;

}

@Override

**public** **int** compareTo(Products o) {

**if** (name.compareTo(o.name)>0) {

**return** 1;

}

**else** **if** (name.compareTo(o.name)<0) {

**return** -1;

}

**else** {

**return** 0;

}

}

}

**3**

**package** Collection\_assignment;

**import** java.util.Comparator;

**import** java.util.Iterator;

**import** java.util.Scanner;

**import** java.util.Set;

**import** java.util.TreeSet;

**public** **class** PrintSorted {

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

Employee emp1 = **new** Employee(1, "Vinod", "A", 20000L);

Employee emp2 = **new** Employee(3, "Teju", "D", 50000L);

Employee emp3 = **new** Employee(2, "Shiva", "C", 30000L);

Employee emp4 = **new** Employee(4, "Meghana", "B", 70000L);

System.***out***.println("Enter a to sort according to id "+"\n"+"Enter b to sort according to Name "

+"\n"+"Enter c to sort according to department "+"\n"+"Enter d to sort according to Salary ");

Scanner sc = **new** Scanner(System.***in***);

String ch = sc.nextLine();

Set<Employee> set = **new** TreeSet<Employee>(**new** CustomSort(ch));

set.add(emp1);

set.add(emp2);

set.add(emp3);

set.add(emp4);

Iterator<Employee> i= set.iterator();

**while**(i.hasNext())

{

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

}

sc.close();

}

}

**class** Employee {

**int** id;

String name;

String dept;

**long** salary;

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

**super**();

**this**.id = id;

**this**.name = name;

**this**.dept = dept;

**this**.salary = salary;

}

@Override

**public** String toString() {

**return** "Employee [id=" + id + ", name=" + name + ", dept=" + dept + ", salary=" + salary + "]";

}

}

**class** CustomSort **implements** Comparator<Employee>{

String a;

**public** CustomSort(String a) {

**super**();

**this**.a = a;

}

@Override

**public** **int** compare(Employee o1, Employee o2) {

**if**(a.equalsIgnoreCase("a")) {

**return** o1.id-o2.id;

}**else** **if**(a.equalsIgnoreCase("b")) {

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

}**else** **if**(a.equalsIgnoreCase("c")) {

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

}**else** **if**(a.equalsIgnoreCase("d")) {

**if** (o1.salary>o2.salary) {

**return** 1;

}

**else** **if** (o1.salary<o2.salary) {

**return** -1;

}

**else** {

**return** 0;

}

}

**return** 0;

}

}

Output

Enter a to sort according to id

Enter b to sort according to Name

Enter c to sort according to department

Enter d to sort according to Salary

c

Employee [id=1, name=Vinod, dept=A, salary=20000]

Employee [id=4, name=Meghana, dept=B, salary=70000]

Employee [id=2, name=Teju, dept=C, salary=30000]

Employee [id=3, name=Shiva, dept=D, salary=50000]

**4**

**package** Collection\_assignment;

**import** java.time.LocalDateTime;

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

**import** java.time.format.DateTimeFormatterBuilder;

**import** java.util.LinkedList;

**import** java.util.List;

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

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

DateDemo date = **new** DateDemo("11/11/1999");

# DateDemo date1 = **new** DateDemo("09/12/2000");

DateDemo date2 = **new** DateDemo("01/10/2001");

DateDemo date3 = **new** DateDemo("10/10/2002");

DateDemo date4 = **new** DateDemo("10/10/2003");

DateDemo date5= **new** DateDemo("10/10/2004");

DateDemo date6 = **new** DateDemo("10/10/2005");

List<DateDemo> dobList = **new** LinkedList<>();

dobList.add(date);

dobList.add(date1);

dobList.add(date2);

dobList.add(date3);

dobList.add(date4);

dobList.add(date5);

dobList.add(date6);

Object df = **new** DateTimeFormatterBuilder();

**for** (**int** i = 0; i < dobList.size(); i++) {

LocalDateTime ld= LocalDateTime.*parse*(dobList.get(i).date);

String sd=(ld).format((DateTimeFormatter) df);

**if**(ld.getYear() % 4 == 0) {

System.***out***.println(sd + " is an leap year");

}**else** {

System.***out***.println(sd + " is not an leap year");

}

}

}

}

**class** DateDemo {

String date;

**public** DateDemo(String date) {

**super**();

**this**.date = date;

}

@Override

**public** String toString() {

**return** "[date=" + date + "]";

}

**public** String getDate() {

**return** date;

}

}