**Collections Assignment**

1. import java.util.\*;

class Contact {

long phoneNumber;

String name,email,gender;

public Contact(long phoneNumber, String name, String email, String gender) {

this.phoneNumber = phoneNumber;

this.name = name;

this.email = email;

this.gender = gender;

}

}

public class MapExample

{

public static void main(String[] args)

{

Map<Long,Contact> map=new TreeMap<Long,Contact>();

Contact c1=new Contact(7569473323L,"SriRam","sriram.com","M");

Contact c2=new Contact(9701141237L,"Manikanta","manikanta.com","M");

Contact c3=new Contact(8978505669L,"Harshi","harshi.com","F");

map.put(2L,c2);

map.put(1L,c1);

map.put(3L,c3);

//Traversing map

for(Map.Entry<Long, Contact> entry:map.entrySet()){

long key=entry.getKey();

Contact sri=entry.getValue();

System.out.println(key+" Details:");

System.out.println(sri.phoneNumber+" "+sri.name+" "+sri.email+" "+sri.gender);

}

}

}

2. import java.util.\*;

class TreeSet1 {

public static void main(String args[]){

//Creating and adding elements

TreeSet<Integer> uniqueVal=new TreeSet<Integer>();

uniqueVal.add(1);

uniqueVal.add(2);

uniqueVal.add(3);

uniqueVal.add(4);

uniqueVal.add(5);

uniqueVal.add(6);

uniqueVal.add(7);

uniqueVal.add(8);

uniqueVal.add(9);

uniqueVal.add(10);

//adding duplicate value to set but it will not store null values ,it simply rejects duplicate at run time

uniqueVal.add(10);

System.out.println(uniqueVal);

System.out.println("Reverse Set: "+ uniqueVal.descendingSet());

}

}

3.

import java.util.Objects;

class Employee implements Comparable<Employee> {

private int id;

private String name;

private String dept;

private int salary;

public static String field;

public Employee() {

}

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

this.id = id;

this.name = name;

this.dept = dept;

this.salary = salary;

}

public int getId() {

return id;

}

public String getName() {

return name;

}

public String getDept() {

return dept;

}

public int getSalary() {

return salary;

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (o == null || getClass() != o.getClass()) return false;

Employee employee = (Employee) o;

return id == employee.id && Objects.equals(dept, employee.dept);

}

@Override

public int hashCode() {

return Objects.hash(id, dept);

}

@Override

public int compareTo(Employee e) {

if (field == "salary") {

return this.getSalary() > e.getSalary() ? 1 : -1;

} else if (field == "id") {

return this.getId() > e.getId() ? 1 : -1;

} else if (field == "name") {

return this.getName().compareTo(e.getName());

} else if (field == "dept") {

return this.getDept().compareTo(e.getDept());

}

return 0;

}

}

4.