**COLLECTIONS FRAMEWORK**

1.CONTACT

import java.util.TreeMap;

class Contact{

long PhoneNumber;

String Name;

String Email;

String Gender;

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

super();

PhoneNumber = phoneNumber;

Name = name;

Email = email;

Gender = gender;

}

@Override

public String toString() {

return "[Number=" + PhoneNumber + ", Name=" + Name + ", Email=" + Email + ", Gender=" + Gender + "]" + "\n";

}

}

public class collOne {

public static void main(String[] args) {

Contact obj1 = new Contact( 8108764545L, "Glenn" , "glenn@gmail.com" , "Male");

Contact obj2 = new Contact( 877991234L, "Sam" , "sam@gmail.com" , "Male");

Contact obj3 = new Contact( 8655454545L, "Kim" , "kim@gmail.com" , "Male");

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

tr.put(8108764545L, obj1);

tr.put(877991234L, obj2);

tr.put(8655454545L, obj3);

System.out.println("Fetching all the keys");

for(Long intk : tr.keySet())

{

System.out.println(intk);

}

System.out.println("Fetching all the Values");

for (Contact strV : tr.values())

{

System.out.println(strV);

}

System.out.println("Printing all the Key-Values pairs:"+ tr);

}  
}

1 OUTPUT

C:\Users\GLMACHAD\Documents>javac collOne.java

C:\Users\GLMACHAD\Documents>java collOne

Fetching all the keys

877991234

8108764545

8655454545

Fetching all the Values

[Number=877991234, Name=Sam, Email=sam@gmail.com, Gender=Male]

[Number=8108764545, Name=Glenn, Email=glenn@gmail.com, Gender=Male]

[Number=8655454545, Name=Kim, Email=kim@gmail.com, Gender=Male]

Printing all the Key-Values pairs:{877991234=[Number=877991234, Name=Sam, Email=sam@gmail.com, Gender=Male]

, 8108764545=[Number=8108764545, Name=Glenn, Email=glenn@gmail.com, Gender=Male]

, 8655454545=[Number=8655454545, Name=Kim, Email=kim@gmail.com, Gender=Male]

}

2.HASHSET

import java.util.HashSet;

public class collTwo {

public static void main(String[] args)

{

// HashSet initialization

HashSet<Integer> myhashset = new HashSet<>();

myhashset.add(11);

myhashset.add(21);

myhashset.add(3);

myhashset.add(4);

myhashset.add(50);

myhashset.add(6);

myhashset.add(7);

myhashset.add(87);

myhashset.add(9);

myhashset.add(10);

myhashset.add(10);//silently ignoring the duplicate data

myhashset.add(11);

System.out.println(myhashset);

}

}

2 OUTPUT

C:\Users\GLMACHAD\Documents>javac collTwo.java

C:\Users\GLMACHAD\Documents>java collTwo

[50, 3, 4, 21, 6, 7, 87, 9, 10, 11]

3.TREESET

import java.util.\*;

import java.util.TreeSet;

public class collThree {

public static void main(String[] args) {

// TODO Auto-generated method stub

Employee emp\_1 = new Employee(1, "GLENN", "A", 10000L);

Employee emp\_2 = new Employee(3, "GLENN1", "B", 20000L);

Employee emp\_3 = new Employee(2, "GLENN2", "C", 30000L);

Employee emp\_4 = new Employee(4, "GLENN3", "D", 50000L);

Employee emp\_5 = new Employee(5, "GLENN4", "E", 60000L);

Employee emp\_6 = new Employee(6, "GLENN5", "F", 68600L);

Employee emp\_7 = new Employee(7, "GLENN6", "G", 37000L);

Employee emp\_8 = new Employee(8, "GLENN7", "H", 75000L);

Employee emp\_9 = new Employee(9, "GLENN8", "I", 24000L);

Employee emp\_10 =new Employee(10, "GLENN9", "J", 33000L);

System.out.println("1.Enter a to sort according to id: ");

System.out.println( "2.Enter b to sort according to Name: ");

System.out.println("3.Enter c to sort according to department :");

System.out.println("4.Enter d to sort according to Salary:\n ");

System.out.println("Please Enter the options according to your choice");

Scanner sc = new Scanner(System.in);

String ch = sc.nextLine();

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

set.add(emp\_1);

set.add(emp\_2);

set.add(emp\_3);

set.add(emp\_4);

set.add(emp\_5);

set.add(emp\_6);

set.add(emp\_7);

set.add(emp\_8);

set.add(emp\_9);

set.add(emp\_10)

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;

}

}

3rd PROGRAM OUTPUT

C:\Users\GLMACHAD\Documents>javac collThree.java

C:\Users\GLMACHAD\Documents>java collThree

1.Enter a to sort according to id:

2.Enter b to sort according to Name:

3.Enter c to sort according to department :

4.Enter d to sort according to Salary:

Please Enter the options according to your choice

a

Employee [id=1, name=GLENN, dept=A, salary=10000]

Employee [id=2, name=GLENN2, dept=C, salary=30000]

Employee [id=3, name=GLENN1, dept=B, salary=20000]

Employee [id=4, name=GLENN3, dept=D, salary=50000]

Employee [id=5, name=GLENN4, dept=E, salary=60000]

Employee [id=6, name=GLENN5, dept=F, salary=68600]

Employee [id=7, name=GLENN6, dept=G, salary=37000]

Employee [id=8, name=GLENN7, dept=H, salary=75000]

Employee [id=9, name=GLENN8, dept=I, salary=24000]

Employee [id=10, name=GLENN9, dept=J, salary=33000]

C:\Users\GLMACHAD\Documents>java collThree

1.Enter a to sort according to id:

2.Enter b to sort according to Name:

3.Enter c to sort according to department :

4.Enter d to sort according to Salary:

Please Enter the options according to your choice

b

Employee [id=1, name=GLENN, dept=A, salary=10000]

Employee [id=3, name=GLENN1, dept=B, salary=20000]

Employee [id=2, name=GLENN2, dept=C, salary=30000]

Employee [id=4, name=GLENN3, dept=D, salary=50000]

Employee [id=5, name=GLENN4, dept=E, salary=60000]

Employee [id=6, name=GLENN5, dept=F, salary=68600]

Employee [id=7, name=GLENN6, dept=G, salary=37000]

Employee [id=8, name=GLENN7, dept=H, salary=75000]

Employee [id=9, name=GLENN8, dept=I, salary=24000]

Employee [id=10, name=GLENN9, dept=J, salary=33000]

C:\Users\GLMACHAD\Documents>java collThree

1.Enter a to sort according to id:

2.Enter b to sort according to Name:

3.Enter c to sort according to department :

4.Enter d to sort according to Salary:

Please Enter the options according to your choice

c

Employee [id=1, name=GLENN, dept=A, salary=10000]

Employee [id=3, name=GLENN1, dept=B, salary=20000]

Employee [id=2, name=GLENN2, dept=C, salary=30000]

Employee [id=4, name=GLENN3, dept=D, salary=50000]

Employee [id=5, name=GLENN4, dept=E, salary=60000]

Employee [id=6, name=GLENN5, dept=F, salary=68600]

Employee [id=7, name=GLENN6, dept=G, salary=37000]

Employee [id=8, name=GLENN7, dept=H, salary=75000]

Employee [id=9, name=GLENN8, dept=I, salary=24000]

Employee [id=10, name=GLENN9, dept=J, salary=33000]

C:\Users\GLMACHAD\Documents>java collThree

1.Enter a to sort according to id:

2.Enter b to sort according to Name:

3.Enter c to sort according to department :

4.Enter d to sort according to Salary:

Please Enter the options according to your choice

d

Employee [id=1, name=GLENN, dept=A, salary=10000]

Employee [id=3, name=GLENN1, dept=B, salary=20000]

Employee [id=9, name=GLENN8, dept=I, salary=24000]

Employee [id=2, name=GLENN2, dept=C, salary=30000]

Employee [id=10, name=GLENN9, dept=J, salary=33000]

Employee [id=7, name=GLENN6, dept=G, salary=37000]

Employee [id=4, name=GLENN3, dept=D, salary=50000]

Employee [id=5, name=GLENN4, dept=E, salary=60000]

Employee [id=6, name=GLENN5, dept=F, salary=68600]

Employee [id=8, name=GLENN7, dept=H, salary=75000]

4.LEAP YEAR

import java.time.LocalDate;

import java.time.format.DateTimeFormatter;

import java.util.LinkedList;

import java.util.List;

public class collFour {

public static void main(String[] args) {

Date date = new Date("01/01/1999");

Date date1 = new Date("12/02/2010");

Date date2 = new Date("13/03/2011");

Date date3 = new Date("10/10/2012");

Date date4 = new Date("15/10/2013");

Date date5= new Date("16/10/2004");

Date date6 = new Date("10/10/2005");

List<Date> dobList = new LinkedList<>();

dobList.add(date);

dobList.add(date1);

dobList.add(date2);

dobList.add(date3);

dobList.add(date4);

dobList.add(date5);

dobList.add(date6);

DateTimeFormatter df = DateTimeFormatter.ofPattern("dd/MM/yyyy");

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

LocalDate ld =LocalDate.parse(dobList.get(i).date,df);

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

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

System.out.println(sd + "This is a leap year");

}

else{

System.out.println(sd+"This is not a leap year");

}

}

}

}

class Date {

String date;

public Date(String date) {

super();

this.date = date;

}

@Override

public String toString() {

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

}

public String getDate() {

return date;

}  
}