13-05-2020

Today we will continue from previous class TreeSet by another example :-

```
import java.util.TreeSet;
class TreeSetDemo1 {
    public static void main(String[] args) {
        TreeSet t = new TreeSet();
        t.add(new StringBuffer("Shershah"));
        t.add(new StringBuffer("College"));
        System.out.println(t);//ClassCastException
    }
}
```

The ClassCastException occurs :- Why ?

→ If we are depending on default natural sorting order then objects should be homogeneous and comparable. Otherwise we will get runtime exception — ClassCastException

→ An object is said to be comparable if and only if the corresponding class implements Comparable interface present in java.lang package.

→ String class and all wrapper classes are already implementing Comparable interface. But StringBuffer class does not implement Comparable interface.
You can see them by typing the following

Page 1 of 1

Signature of Examiner Date :
Signature of Head Examiner Date :
Signature of Examiner Date :
Signature of Head Examiner Date :

Scanned with CamScanner

commands from command prompt :—

javap java.lang.String ↵
javap java.lang.StringBuffer ↵

(Lets do & see the output self.)

You will see the difference of class definition that String implements java.io.Serializable and java.lang.Comparable

and StringBuffer implements java.io.Serializable but not java.lang.Comparable

Let us Know about Comparable interface :-

Comparable (interface) is present in java.lang package.

Function of Comparable(I) :—

```
public int compareTo (Object obj);
```

Ex: obj1.compareTo(obj2)

→ returns -ve if and only if obj1 has to come before obj2

→ returns +ve if and only if obj1 has to come after obj2

→ returns 0 if and only if obj1 and obj2 are equal.

eg:-
```
System.out.println("B".compareTo("D")); // -ve
System.out.println("Z".compareTo("B")); // +ve
System.out.println("A".compareTo("A")); // 0
System.out.println("A".compareTo(null)); //NullPointerException
```

Scanned with CamScanner

Let us see the following set of instructions :—

```
TreeSet t = new TreeSet();
t.add("B");
t.add("Z"); // JVM calls "Z".compareTo("B"); (got +ve)
t.add("P"); // "P".compareTo("B"); (got +ve)
System.out.println(t); // [B, P, Z]
```

If we depending on default natural sorting order, internally JVM will call compareTo() method for inserting objects to the TreeSet. Hence the Objects should be comparable.

→ Sometimes, if we need our own defined sorting order then Comparator is used to define our own customized sorting order.

→ Comparable ment for Default natural sorting order where as Comparator ment for customized sorting order.

Comparator (I)

→ This interface is ment for Customized sorting order.

→ Comparator interface is present in java.util package.

→ Comparator methods are — compare(), and equals().

Scanned with CamScanner

Complete Prototypes of methods :-

① public int compare(Object obj1, Object obj2);

returns +ve : if and only if obj1 has to come after obj2.
returns -ve : if and only if obj1 has to come before obj2.
returns 0 : if and only if obj1 and obj2 are equal.

Comparator Interface

★ We can use Comparator to define our own sorting.
★ Comparator interface is defined in java.util package.
★ It defines two methods — compare() and equals()

① public int compare(Object obj1, Object obj2)
② public boolean equals();

★ When ever we are implementing Comparator interface, compulsory we should provide implementation for compare() method.
★ And implementing equals() method is optional, because it is already available in every java class from Object class through inheritance.

Scanned with CamScanner

Let us do-Example :- To insert integers into the TreeSet where the sorting order is descending order.

```
import java.util.*;

class TreeSetDemo3
{
    public static void main(String [] args)
    {
        TreeSet t = new TreeSet(new MyComparator());
        t.add(200);
        t.add(150);
        t.add(300);
        t.add(199);
        t.add(300);
        System.out.println(t);
    }
}

class MyComparator implements Comparator
{
    public int compare(Object obj1, Object obj2)
    {
        Integer ob1 = (Integer) obj1;
        Integer ob2 = (Integer) obj2;
        if(ob1 < ob2) return (+1);
        else if(ob1 > ob2) return (-1);
        else return 0;
    }
}
```

Scanned with CamScanner

Output :- [300, 200, 199, 150].

In the above code if we write 1st line of code as :-

```
TreeSet t = new TreeSet();
```

then output is according to natural sorting order — [150, 199, 200, 300].

If the 1st line is as follows :-

```
TreeSet t = new TreeSet(new MyComparator());
```

then the output is customized according to our class MyComparator :-

[300, 200, 199, 150].

☆ At 1st line when we are not passing Comparator object then internally JVM will call compareTo() method which meant for default natural sorting order (ascending) with output :-

[150, 199, 200, 300].

☆ If we are passing comparator object at line 1 then internally JVM will call compare() method which is defined for descending sorting order. In this case output is :- [300, 200, 199, 150]

Lets take another implementation of Comparator :-

```
class MyComparator implements Comparable
{
    public int compare(Object obj1, Object obj2)
```

Scanned with CamScanner

```
Integer i1 = (Integer) obj1;
Integer i2 = (Integer) obj2;
//return i1.compareTo(i2); //[150, 199, 200, 300] Asc
//return -i1.compareTo(i2); // [300, 200, 199, 150] Dsc.
//return i2.compareTo(i1); // [300, 200, 199, 150] Dsc
//return -i2.compareTo(i1); // [150, 199, 200, 300] Asc
return i2.compareTo(i1);
// return +1; // [200, 150, 300, 199, 300] Inserted order
// return -1; // [300, 199, 300, 150, 200] Reverse of inserted order
return 0; // [200] only first element. all other elements are considered as duplicates.
}
}
```

Above are the all possibilities which we can run and see the outputs from above program.

[I have also included these all possibilities with program with output - with this lecture Notes.]

Let us again make a customized Comparator class for String object Sorting for reverse of alphabetical order.

Scanned with CamScanner

```
import java.util.*;
class MyComparator implements Comparator
{
    public int compare(Object obj1, Object obj2)
    {
    String s1 = (String)obj1; //String s1 = obj1.toString();
    String s2 = (String)obj2; //String s2 = obj2.toString();

   //return s2.compareTo(s1); // Reverse of alphabetic
     return -s1.compareTo(s2); //Reverse of alphabetic
  } }

class myTreeSet2{
  public static void main(String []args)
   { TreeSet t = new TreeSet(new MyComparator());
     t.add("Ram");
     t.add("Mohan");
     t.add("Suraj");
     t.add("Ajeet");
     t.add("sanjay");
   System.out.println(t);
   }
}
```

Output will be [Suraj, Sanjay, Ram, Mohan, Ajeet]

Reverse of sorting (alphabetic order)

—END—

Scanned with CamScanner