Java Comparable Interface

Java Comparable interface is a member of collection framework which is used to compare objects and sort them according to the natural order.

The natural ordering refers to the behavior of `compareTo()` method which is defined into Comparable interface. Its sorting technique depends on the type of object used by the interface. If object type is string then it sorts it Lexicographically.

If object type is wrapper class object like: integer or list then it sorts according to their values.

If object type is custom object like: user defined object then sorts according to the defined `compareTo()` method.

Classes that implements this interface can be sorted automatically by calling `Collections.sort()` method. Objects that implement this interface can be used as keys in a sorted map or as elements in a sorted set, without the need to specify a comaprator. Declaration of this interface is given below.

Declaration

```
public interface Comparable<T>
```

Copy

Comparable Method

It contains single method `compareTo()` that is given below.

int compareTo(T o) : It compares object with the specified object for order.

The `compareTo()` function compares the current object with the provided object. This function is already implemented for default wrapper classes and primitive data types but, this function also needs to be implemented for user-defined classes.

It **returns** positive integer, if the current object is greater than the provided object.

If the current object is less than the provided object then it returns negative integer.

If the current object is equal to the provided object then it returns zero.

Exceptions

This method returns NullPointerException, if the specified object is null and `ClassCastException` if the specified object's type prevents it from being compared to this object.

Example : Sorting list

Lets take an example to sort an ArrayList that stores integer values. We are using `sort()` method of Collections class that sort those object which implements Compares interface. Since integer wrapper class implements Comparable so we are able to get sorted objects. See the below example.

```


import java.util.*;

public class Demo {
    public static void main(String a[]){

      // Creating List

        ArrayList<Integer> list = new ArrayList<>();

        // Adding elements

        list.add(100);

        list.add(2);

        list.add(66);

        list.add(22);

        list.add(10);

        // Displaying list

        System.out.println(list);

        // Sorting list

        Collections.sort(list);

        // Displaying sorted list

        System.out.println("Sorted List : "+list);

    }

}
```

Copy

```
[100, 2, 66, 22, 10]

Sorted List : [2, 10, 22, 66, 100]
```

Example: Sorting String objects

While sorting string objects, the comparable sorts it based on lexicographically. It means a dictionary like sorting order. See the below example.

```
import java.util.*;

public class Demo {

    public static void main(String a[]){

      // Creating List

        ArrayList<String> list = new ArrayList<>();

        // Adding elements
        list.add("D");

        list.add("L");

        list.add("A");

        list.add("Z");

        list.add("C");

        // Displaying list

        System.out.println(list);

        // Sorting list

        Collections.sort(list);

        // Displaying sorted list

        System.out.println("Sorted List : "+list);

    }
```

Copy

```
[D, L, A, Z, C]

Sorted List : [A, C, D, L, Z]
```

Example: Sorting User Defined Object

If we have custom objects then we have to implement the Comparable interface and override its compareTo() method. Now it will compare based on the logic we defined in our `compareTo()` method. See the below example.

```
import java.util.*;


class Employee implements Comparable <Employee>

{

    int empId;

    String name;

    public Employee (int empId, String name)

    {

        this.empId=empId;

        this.name=name;
    }

    public String toString()

    {

        return this.empId + " " + this.name;

    }


     // Sorting by empId

    public int compareTo(Employee std){


        return this.empId - std.empId;

    }

}


public class Demo {

    public static void main(String a[]){

      ArrayList <Employee> list = new ArrayList <Employee> ( );

        list.add(new Employee(2, "Boman"));

        list.add(new Employee(1, "Abram"));

        list.add(new Employee(3, "Dinesh"));


        // Displaying

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

            System.out.println(list.get(i));


        // Sorting

        Collections.sort(list);


        // Displaying after sorting

        System.out.println("\nAfter Sorting :\n");

        for (int i=0; i<list.size(); i++)
```

```
            System.out.println(list.get(i));

    }

}
```

Copy

```
OUTPUT

2 Boman

1 Abram

3 Dinesh


After Sorting :


1 Abram

2 Boman

3 Dinesh
```