**[Comparable vs Comparator](http://www.instanceofjava.com/2015/05/comparable-vs-comparator-in-java-example.html)**

* One of the common interview question what are the differences between comparable and comparator and how to sort collections using these interfaces?
* What are the differences between comparable and comparator and how to sort employee object by empId or empName using these interfaces.
* Before discussing about differences let’s see brief description about these two interfaces

### Comparable Interface:

* Comparable Interface is actually from java.lang package.
* It will have a method compareTo(Object obj)to sort objects
* public int compareTo(Object obj){ }
* Comparable interface used for natural sorting these is the reason all wrapper classes and string class implementing this comparator and overriding compareTo(Object obj) method.
* So in String and all wrapper classes compareTo(Object  obj) method is implemented in such way that it will sort all objects.

### String class:

* If we observe String class it is implementing comparable interface.
* If compareTo(String str) methods returns 0 : both strings are equal
* If compareTo(String str) method returns 1: string is lexicographically greater than the string argument
* If compareTo(String str) method returns -1: string is lexicographically less than the string argument

*package java.lang;*

*public final class String extends Object implements Serializable,* ***Comparable<String>****, CharSequence*

*{*

*public int compareTo(String anotherString)*

*{*

*//logic*

*}*

*}*

### Comparing String Objects:

* Lets see an example java program that will explains how two string objects are compared using compareTo(String str) method in String class.

**package com.instanceofjava;**

**public class StringCompareDemo**

**{**

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

**{**

**String str1 = "comparable";**

**String str2 = "comparator";**

**int value = str1.compareTo(str2);**

**if (value == 0)**

**{**

**System.*out*.println("Strings are equal");**

**} else**

**{**

**System.*out*.println("Strings are not equal");**

**}**

**}**

**}**

**Output:**

1. Strings are not equal

### Wrapper classes:

* Wrapper classes are used to convert primitive data values into java objects. for 8 primitive data types java has 8 corresponding wrapper classes. All these classes implementing comparable interface.
* Let’s see an example on Integer wrapper class

#### Integer:

**package java.lang;**

**public final class Integer extends Number implements Comparable<Integer>**

**{**

**public int compareTo(Integer i)**

**{**

**//**

**}**

**}**

* Lets see an example program of comparing two integer objects

**package instanceofjava;**

**public class IntegerComparableDemo**

**{**

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

**{**

**// compares two Integer objects numerically**

**Integer obj1 = new Integer("37");**

**Integer obj2 = new Integer("37");**

**int retval = obj1.compareTo(obj2);**

**if (retval > 0)**

**{**

**System.*out*.println("obj1 is greater than obj2");**

**}**

**else if (retval < 0)**

**{**

**System.*out*.println("obj1 is less than obj2");**

**} else**

**{**

**System.*out*.println("obj1 is equal to obj2");**

**}**

**}**

**}**

**Output:**

1. obj1 is equal to obj2;

### Sorting Collections using Comparator:

* By using Collections.sort(list); method we can sort objects in natural object sorting order
* An example program on Collections.sort(list);
* Sorting Employee objects by id.

**package** instanceofjava;

**import** java.util.ArrayList;

**import** java.util.Collections;

**import** java.util.Iterator;

**public** **class** Employee **implements** Comparable

{

String name;

**int** id;

**public** String getName()

{

**return** name;

}

**public** **void** setName(String name)

{

**this**.name = name;

}

**public** **int** getId()

{

**return** id;

}

**public** **void** setId(**int** id)

{

**this**.id = id;

}

**public** Employee(String name, **int** id)

{

**this**.name = name;

**this**.id = id;

}

@Override

**public** **int** compareTo(Object in)

{

**return** **new** Integer(**this**.id).compareTo(((Employee) in).id);

}

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

{

Employee e1 = **new** Employee("xyz", 37);

Employee e2 = **new** Employee("abc", 46);

Employee e3 = **new** Employee("sai", 38);

ArrayList al = **new** ArrayList();

al.add(e1);

al.add(e2);

al.add(e3);

Collections.*sort*(al);

Iterator itr = al.iterator();

**while** (itr.hasNext())

{

Employee em = (Employee) itr.next();

System.***out***.println(em.getId() + " " + em.getName());

}

}

}

**Output:**

1. 37 xyz
2. 38 sai
3. 46 abc

### Comparator:

* Comparator Interface is actually from java.util package.
* It will have a method compare(Object obj1, Object obj2)to sort objects
* public int  compare(Object obj1, Object obj2){ }
* Comparator interface used for customized sorting.
* Will place sorting logic in other class so that its easy to change.
* An example program which will sort employee objects by name

#### 1.Employee.java

**package instanceofjava;**

**import java.util.ArrayList;**

**import java.util.Collections;**

**import java.util.Iterator;**

**public class Employee**

**{**

**String name;**

**int id;**

**public String getName()**

**{**

**return name;**

**}**

**public void setName(String name)**

**{**

**this.name = name;**

**}**

**public int getId()**

**{**

**return id;**

**}**

**public void setId(int id)**

**{**

**this.id = id;**

**}**

**public Employee(String name, int id)**

**{**

**this.name = name;**

**this.id = id;**

**}**

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

**{**

**Employee e1 = new Employee("xyz", 37);**

**Employee e2 = new Employee("abc", 46);**

**Employee e3 = new Employee("sai", 38);**

**ArrayList al = new ArrayList();**

**al.add(e1);**

**al.add(e2);**

**al.add(e3);**

**Collections.*sort*(al, new EmpSortByName());**

**Iterator itr = al.iterator();**

**while (itr.hasNext())**

**{**

**Employee em = (Employee) itr.next();**

**System.*out*.println(em.getId() + " " + em.getName());**

**}**

**}**

**}**

#### EmpSortByName.java:

**package instanceofjava;**

**import java.util.Comparator;**

**public class EmoSortByName implements Comparator<Employee>**

**{**

**@Override**

**public int compare(Employee arg0, Employee arg1)**

**{**

**return arg0.getName().compareTo(arg1.getName());**

**}**

**}**

**Output:**

1. 46 abc
2. 38 sai
3. 37 xyz

### Differences Between Comparable and Comparator:

| **Parameter** | **Comparable** | **Comparator** |
| --- | --- | --- |
| Package | java.lang.Comparable | java.util.Comparator |
| Sorting logic | Must be in Same class | Separate class |
| Method definition | public int compareTo(Object obj) | public int compare(Object obj1, Object obj2) |
| Method call | Collections.sort(list) | Collections.sort(list, new OtherSortClass()) |
| Purpose | Natural Sorting | Custom Sorting |