Comparable and Comparator both are interfaces and can be used to sort collection elements.

**Java** **Comparable**

Java provides **Comparable** interface which should be implemented by any custom class if we want to use Arrays or Collections sorting methods. Comparable interface has **compareTo(T obj)** method which is used by sorting methods, you can check any Wrapper, String or Date class to confirm this. We should override this method in such a way that it returns a negative integer, zero, or a positive integer if “this” object is less than, equal to, or greater than the object passed as argument.

int compareTo(T o) :Compares this object with the specified object for order.

Use the compareTo method of the [Comparable](https://msdn.microsoft.com/en-us/library/aa986890(v=vs.80).aspx) interface to compare two objects of the same type. The rules for the return values for compareTo are summarized as follows:

|  |  |
| --- | --- |
| Condition | Return Value |
| The object implementing compareTo is less than the parameter passed to compareTo. | -1 |
| The object implementing compareTo is equal to the parameter passed to compareTo. | 0 |
| The object implementing compareTo is greater than the parameter passed to compareTo. | 1 |

**Java Comparator**

Comparator interface compare(Object o1, Object o2) method need to be implemented that takes two Object argument, it should be implemented in such a way that it returns negative int if first argument is less than the second one and returns zero if they are equal and positive int if first argument is greater than second one.

compare() method

**public int compare(Object obj1,Object obj2):** compares the first object with second object.

Differences between Comparable and Comparator interfaces that are given below.

**Comparator vs Comparable**

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Comparable** | **Comparator** |
| Sorting logic | Sorting logic must be in same class whose objects are being sorted. Hence this is called natural ordering of objects | Sorting logic is in separate class. Hence we can write different sorting based on different attributes of objects to be sorted. E.g. Sorting using id,name etc. |
| Implementation | Class whose objects to be sorted must implement this interface.e.g Country class needs to implement comparable to collection of country object by id | Class whose objects to be sorted do not need to implement this interface.Some other class can implement this interface. E.g.-CountrySortByIdComparator class can implement Comparator interface to sort collection of country object by id |
| Sorting method | int compareTo(Object o1) This method compares this object with o1 object and returns  a integer.Its value has following meaning 1. positive – this object is greater than o1 2. zero – this object equals to o1 3. negative – this object is less than o1 | int compare(Object o1,Object o2) This method compares o1 and o2 objects. and returns  a integer.Its value has following meaning. 1. positive – o1 is greater than o2 2. zero – o1 equals to o2 3. negative – o1 is less than o1 |
| Calling method | Collections.sort(List) Here objects will be sorted on the basis of CompareTo method | Collections.sort(List, Comparator) Here objects will be sorted on the basis of Compare method in Comparator |
| Package | Java.lang.Comparable | Java.util.Comparator |

1. Comparable interface can be used to provide single way of sorting whereas Comparator interface is used to provide different ways of sorting.
2. For using Comparable, Class needs to implement it whereas for using Comparator we don’t need to make any change in the class.
3. Comparable interface is in java.lang package whereas Comparator interface is present in java.util package.
4. We don’t need to make any code changes at client side for using Comparable, Arrays.sort() or Collection.sort() methods automatically uses the compareTo() method of the class. For Comparator, client needs to provide the Comparator class to use in compare() method.

|  |  |
| --- | --- |
| **Comparable** | **Comparator** |
| 1) Comparable provides **single sorting sequence**. In other words, we can sort the collection on the basis of single element such as id or name or price etc. | Comparator provides **multiple sorting sequence**. In other words, we can sort the collection on the basis of multiple elements such as id, name and price etc. |
| 2) Comparable **affects the original class** i.e. actual class is modified. | Comparator **doesn't affect the original class** i.e. actual class is not modified. |
| 3) Comparable provides **compareTo() method** to sort elements. | Comparator provides **compare() method** to sort elements. |
| 4) Comparable is found in **java.lang** package. | Comparator is found in **java.util** package. |
| 5) We can sort the list elements of Comparable type by**Collections.sort(List)** method. | We can sort the list elements of Comparator type by**Collections.sort(List,Comparator)** method. |

|  |  |
| --- | --- |
| **Comparable** | **Comparator** |
| 1) Comparable provides **single sorting sequence**. In other words, we can sort the collection on the basis of single element such as id or name or price etc. | Comparator provides **multiple sorting sequence**. In other words, we can sort the collection on the basis of multiple elements such as id, name and price etc. |
| 2) Comparable **affects the original class** i.e. actual class is modified. | Comparator **doesn't affect the original class** i.e. actual class is not modified. |
| 3) Comparable provides **compareTo() method** to sort elements. | Comparator provides **compare() method** to sort elements. |
| 4) Comparable is found in **java.lang** package. | Comparator is found in **java.util** package. |
| 5) We can sort the list elements of Comparable type by**Collections.sort(List)** method. | We can sort the list elements of Comparator type by**Collections.sort(List,Comparator)** method. |