**Basic Difference:**

**1. Comparable interface:**

Class whose objects to be sorted **must** implement this interface. In this, need to implement compareTo(Object) method.

For example:

public class Country **implements Comparable**{

       @Override

    public int compareTo(Country country) {

        return (this.countryId < country.countryId ) ? -1: (this.countryId > country.countryId ) ? 1:0 ;

}}

If any class implements comparable inteface then collection of object can be sorted automatically using Collection.sort() or Arrays.sort().Object will be sort on the basis of compareTo method in that class.

#### 2. Comparator interface:

Class whose objects to be sorted **do not need** to implement this interface. Some third class can implement this interface to sort.e.g.**CountrySortByIdComparator** class can implement Comparator interface to sort collection of **Country** object by id.

For example:

public class CountrySortByIdComparator **implements Comparator**{

    @Override

**public int compare(Country country1, Country country2)** {

        return (country1.getCountryId() < country2.getCountryId() ) ? -1: (country1.getCountryId() > country2.getCountryId() ) ? 1:0 ;

    }

}

Also You can use anonymous comparator to compare at particular line of code. For example:(In this case we don’t need another class). Important

Collections.sort(listOfCountries,**new Comparator() {//Anonymous bz no name for Comparater**

                @Override

                public int compare(Country o1, Country o2) {

                    return o1.getCountryName().compareTo(o2.getCountryName());

                }

            });

### **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 |

For More Detail Example:

Please see: <https://java2blog.com/difference-between-comparator-and-comparable-in-java/>