**Use of comparable and comparator interface**

A comparable object is capable of comparing itself with another object. The class itself must implements the java.lang.Comparable interface in order to be able to compare its instances.

A comparator object is capable of comparing two different objects. The class is not comparing its instances, but some other class’s instances. This comparator class must implement the java.util.Comparator interface.

<http://www.java-questions.com/use_of_comparable_comparator.html>

In java the element in collections can be sorted by using TreeSet or TreeMap. To sort the data elements a class needs to implement Comparator or Comparable interface. That’s why all Wrapper classes like Integer, Double and String class implements Comparable interface.

A class implementing **Comparable** interface need to override **compareTo**(Object obj) method and put the logic for sorting.

[?](http://www.java-questions.com/use_of_comparable_comparator.html)

|  |  |
| --- | --- |
| 1  2  3  4 | The method returns an int value :-1,0,1   It will return -1 : If this object is lesser than the passed object   It will return  0 : If this object is same the passed object   It will return 1 : If this object is greater than the passed object |

Consider a class Person.

[?](http://www.java-questions.com/use_of_comparable_comparator.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | class Person {    public String name;    public String lastName;      public Person(String name, String lastName){       this.name=name;       this.lastName=lastName;    }    public String getName(){      return name;    }    public String getLastName(){      return lastName;    }      public static void main(String arg[]){          List<person> myList = new ArrayList<person>();        myList.add(new Person("Robert","USA"));      myList.add(new Person("Andy","UK"));      myList.add(new Person("Harish","India"));      for(Person person : myList){       System.out.println("My name is "+person.getName());      }    }   }    Output is :    My name is Robert    My name is Andy    My name is Harish  </person></person> |

But now i want that the objects to be sorted on name basis should be retrieved in sorted order.   
Consider a class Person.

[?](http://www.java-questions.com/use_of_comparable_comparator.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29 | class Person implements Comparable{    public String name;    public String lastName;      public Person(String name, String lastName){       this.name=name;       this.lastName=lastName;    }    public String getName(){      return name;    }    public String getLastName(){      return lastName;    }      public int compareTo(Object obj){      Person p = (Person)obj;      return this.name.compareTo(p.getName);    }    public static void main(String arg[]){          List<person> myList = new ArrayList<person>();        myList.add(new Person("Robert","USA"));      myList.add(new Person("Andy","UK"));      myList.add(new Person("Harish","India"));      Collections.sort(myList);      for(Person person : myList){       System.out.println("My name is "+person.getName());      }    }   }    Output is :    My name is Andy    My name is Harish    My name is Robert  </person></person> |

Couple of things which needs to be taken in consideration:  
1) Collections.sort() will sort only the collection having objects which implements either one of the comparing interface.  
2) Collections.sort() will sort the same list.

**Comparator interface** is used when an extra logic is required to sort the objects. One need to override **compare(Object obj1, Object obj2)** method. For example you want the list of Person object to be sorted on the basis of complete name i.e "name lastName" but also on the other hand doesn’t want to change the Person class default sorting implementation or Person class is a jar so so no code modification in it can be done. First create a Custom Comparator.

|  |  |
| --- | --- |
| 5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | public class MyCustomComparator implements Comparator{    public int compare(Object obj1, Object obj2){     Person p1 =(Person) obj1;     Person p2 =(Person) ob2;     String p1Name = p1.getName()+ " " +p1.getLastName();     String p2Name = p2.getName()+ " " +p2.getLastName();     return p1Name.compareTo(p2Name);    }   }  // Changes made in main method of Person class.    public static void main(String arg[]){          List<person> myList = new ArrayList<person>();        myList.add(new Person("Robert","USA"));      myList.add(new Person("Robert","UK"));      myList.add(new Person("Robert","India"));      Collections.sort(myList new MyCustomComparator());      for(Person person : myList){       System.out.println("My name is "+person.getName() + " " + person.getLastName());      }    }      OutPut:    My name is Robert India    My name is Robert UK    My name is Robert USA  </person></person> |

Couple of things which needs to be taken in consideration:  
1) For Comparator interface you need to override method compare(obj)  
2) In collections.sort() the instance of Comparator need to be passed. In this example the list is sorted according to the custom Comparator created