**Java Comparator interface** is used to order the objects of a user-defined class.

This interface is found in java.util package and contains 2 methods compare(Object obj1,Object obj2) and equals(Object element).

It provides multiple sorting sequences, i.e., you can sort the elements on the basis of any data member, for example, rollno, name, age or anything else.

**Syntax of compare method**

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

**Collections** class provides static methods for sorting the elements of collection. If collection elements are of Set type, we can use TreeSet. But We cannot sort the elements of List. Collections class provides methods for sorting the elements of List type elements.

**Method of Collections class for sorting List elements**

**public void sort(List list,Comparator c):** is used to sort the elements of List by the given comparator.

Example of sorting the elements of List that contains user-defined class objects on the basis of age and name

In this example, we have created 4 java classes:

Student.java

AgeComparator.java

NameComparator.java

Simple.java

**Student.java**

This class contains three fields rollno, name and age and a parameterized constructor.

**class** Student{

**int** rollno;

String name;

**int** age;

Student(**int** rollno,String name,**int** age){

**this**.rollno=rollno;

**this**.name=name;

**this**.age=age;

}

}

**AgeComparator.java**

This class defines comparison logic based on the age. If age of first object is greater than the second, we are returning positive value, it can be any one such as 1, 2 , 10 etc. If age of first object is less than the second object, we are returning negative value, it can be any negative value and if age of both objects are equal, we are returning 0.

**import** java.util.\*;

**class** AgeComparator **implements** Comparator{

**public** **int** Compare(Object o1,Object o2){

Student s1=(Student)o1;

Student s2=(Student)o2;

**if**(s1.age==s2.age)

**return** 0;

**else** **if**(s1.age>s2.age)

**return** 1;

**else**

**return** -1;

}

}

**NameComparator.java**

This class provides comparison logic based on the name. In such case, we are using the compareTo() method of String class, which internally provides the comparison logic.

**import** java.util.\*;

**class** NameComparator **implements** Comparator{

**public** **int** Compare(Object o1,Object o2){

Student s1=(Student)o1;

Student s2=(Student)o2;

**return** s1.name.compareTo(s2.name);

}

}

**Simple.java**

In this class, we are printing the objects values by sorting on the basis of name and age.

**import** java.util.\*;

**import** java.io.\*;

**class** Simple{

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

ArrayList al=**new** ArrayList();

al.add(**new** Student(101,"Vijay",23));

al.add(**new** Student(106,"Ajay",27));

al.add(**new** Student(105,"Jai",21));

System.out.println("Sorting by Name...");

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

Iterator itr=al.iterator();

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

Student st=(Student)itr.next();

System.out.println(st.rollno+" "+st.name+" "+st.age);

}

System.out.println("sorting by age...");

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

Iterator itr2=al.iterator();

**while**(itr2.hasNext()){

Student st=(Student)itr2.next();

System.out.println(st.rollno+" "+st.name+" "+st.age);

}

}

}

Output:Sorting by Name...

106 Ajay 27

105 Jai 21

101 Vijay 23

Sorting by age...

105 Jai 21

101 Vijay 23

106 Ajay 27