Java ArrayList of Object Sort Example (Comparable And Comparator)

BY CHAITANYA SINGH | FILED UNDER: JAVA COLLECTIONS

In this tutorial we will see how to **sort an ArrayList of Objects by property** using comparable and comparator interface. If you are looking for sorting a simple `ArrayList` of String or Integer then you can refer the following tutorials –

- Sorting of ArrayList<String> and ArrayList<Integer>
- Sorting of ArrayList in descending order

We generally use `Collections.sort()` method to sort a simple array list. However if the ArrayList is of **custom object** type then in such case you have two options for sorting- **comparable and comparator** interfaces. Before going through the example of them, let's see what's the output when we try to sort arraylist of Objects without implementing any of these interfaces.

What's the need of comparable and comparator?

Consider the below example – I have a Student class which has properties like Student name, roll no and student age.

```
public class Student  {
    private String studentname;
    private int rollno;
    private int studentage;

    public Student(int rollno, String studentname, int studentage) {
         this.rollno = rollno;
         this.studentname = studentname;
         this.studentage = studentage;
    }

    public String getStudentname() {
         return studentname;
    }
    public void setStudentname(String studentname) {
        this.studentname = studentname;
    }
    public int getRollno() {
        return rollno;
    }
    public void setRollno(int rollno) {
        this.rollno = rollno;
    }
    public int getStudentage() {
        return studentage;
    }
    public void setStudentage(int studentage) {
        this.studentage = studentage;
    }
}
```

And I want to have an ArrayList of Student Object. We do it like this –

```
import java.util.*;
public class ArrayListSorting  {

     public static void main(String args[]){
            ArrayList<Student> arraylist = new ArrayList<Student>();
            arraylist.add(new Student(223, "Chaitanya", 26));
            arraylist.add(new Student(245, "Rahul", 24));
            arraylist.add(new Student(209, "Ajeet", 32));

            Collections.sort(arraylist);

            for(Student str: arraylist){
                           System.out.println(str);
            }
     }
}
```

I tried to call the `Collections.sort()` on the List of Objects and boom! I got the the error message like this –
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
Bound mismatch: The generic method sort(List) of type Collections is not applicable for the arguments (ArrayList). The inferred type Student is not a valid substitute for the bounded parameter > at beginnersbook.com.Details.main(Details.java:11)

Reason: I Just called the sort method on an ArrayList of Objects which actually doesn't work until unless we use interfaces like `Comparable` and `Comparator`.

Now you must have understood the importance of these interfaces. Let's see how to use them to get the sorting done in our way.

Sorting of ArrayList<Object> with Comparable

Let's say we need to sort the ArrayList<Student> based on the student Age property. This is how it can be done – First implement `Comparable` interface and then Override the `compareTo` method.

```
package beginnersbook.com;

public class Student implements Comparable {
    private String studentname;
    private int rollno;
    private int studentage;

    public Student(int rollno, String studentname, int studentage) {
        this.rollno = rollno;
        this.studentname = studentname;
        this.studentage = studentage;
    }
    ...
    //getter and setter methods same as the above example
    ...
    @Override
    public int compareTo(Student comparestu) {
        int compareage=((Student)comparestu).getStudentage();
        /* For Ascending order*/
        return this.studentage-compareage;

        /* For Descending order do like this */
        //return compareage-this.studentage;
    }

    @Override
    public String toString() {
        return "[ rollno=" + rollno + ", name=" + studentname + ", age=" +
studentage + "]";
    }

}
```

Now we can very well call Collections.sort on ArrayList

```
import java.util.*;
public class ArrayListSorting  {

        public static void main(String args[]){
           ArrayList<Student> arraylist = new ArrayList<Student>();
           arraylist.add(new Student(223, "Chaitanya", 26));
           arraylist.add(new Student(245, "Rahul", 24));
           arraylist.add(new Student(209, "Ajeet", 32));

           Collections.sort(arraylist);

           for(Student str: arraylist){
                        System.out.println(str);
           }
        }
}
```

Output:

```
[ rollno=245, name=Rahul, age=24]
[ rollno=223, name=Chaitanya, age=26]
[ rollno=209, name=Ajeet, age=32]
```

Comparable did our job why do we need Comparator anymore?

Since Comparable is implemented by the same class whose objects are sorted so it binds you with that sorting logic which is ok in most of the cases but in case you want to have more than way of sorting your class objects you should use comparators. Read more about them here:

1. Comparable in Java
2. Comparator in Java

Sorting ArrayList<Object> multiple properties with Comparator

We are overriding `compare` method of `Comparator` for sorting.

```
package beginnersbook.com;
import java.util.Comparator;
public class Student  {
    private String studentname;
    private int rollno;
    private int studentage;

    public Student(int rollno, String studentname, int studentage) {
        this.rollno = rollno;
        this.studentname = studentname;
        this.studentage = studentage;
    }
    ...
    //Getter and setter methods same as the above examples
    ...
    /*Comparator for sorting the list by Student Name*/
    public static Comparator<Student> StuNameComparator = new
Comparator<Student>() {

        public int compare(Student s1, Student s2) {
           String StudentName1 = s1.getStudentname().toUpperCase();
           String StudentName2 = s2.getStudentname().toUpperCase();

           //ascending order
           return StudentName1.compareTo(StudentName2);

           //descending order
           //return StudentName2.compareTo(StudentName1);
    }};

    /*Comparator for sorting the list by roll no*/
    public static Comparator<Student> StuRollno = new Comparator<Student>() {

        public int compare(Student s1, Student s2) {

           int rollno1 = s1.getRollno();
           int rollno2 = s2.getRollno();

           /*For ascending order*/
           return rollno1-rollno2;

           /*For descending order*/
           //rollno2-rollno1;
   }};
    @Override
    public String toString() {
        return "[ rollno=" + rollno + ", name=" + studentname + ", age=" +
studentage + "]";
    }

}
```

ArrayList class:

```
package beginnersbook.com;
import java.util.*;
public class Details  {

        public static void main(String args[]){
           ArrayList<Student> arraylist = new ArrayList<Student>();
           arraylist.add(new Student(101, "Zues", 26));
           arraylist.add(new Student(505, "Abey", 24));
           arraylist.add(new Student(809, "Vignesh", 32));

           /*Sorting based on Student Name*/
           System.out.println("Student Name Sorting:");
           Collections.sort(arraylist, Student.StuNameComparator);

           for(Student str: arraylist){
                        System.out.println(str);
           }

           /* Sorting on Rollno property*/
           System.out.println("RollNum Sorting:");
           Collections.sort(arraylist, Student.StuRollno);
           for(Student str: arraylist){
                        System.out.println(str);
           }
        }
}
```

Output:

```
Student Name Sorting:
[ rollno=505, name=Abey, age=24]
[ rollno=809, name=Vignesh, age=32]
[ rollno=101, name=Zues, age=26]
RollNum Sorting:
[ rollno=101, name=Zues, age=26]
[ rollno=505, name=Abey, age=24]
[ rollno=809, name=Vignesh, age=32]
```