**What is Comparable in Java?**

As the name itself suggests, **Comparable** is an [interface](https://www.edureka.co/blog/java-interface/) which defines a way to compare an object with other objects of the same type. It helps to sort the objects that have self-tendency to sort themselves, i.e., the objects must know how to order themselves. **Eg:** Roll number, age, salary. This interface is found in *java.lang package* and it contains only one method, i.e., *compareTo().*Comparable is not capable of sorting the [objects](https://www.edureka.co/blog/java-tutorial/#obj) on its own, but the interface defines a method *int compareTo()*which is responsible for sorting.

## ****What is the compareTo method and how it is used?****

This method is used to compare the given object with the current object. The **compareTo()** method returns an int value. The value can be either positive, negative, or zero. So now we are well acquainted with the theoretical knowledge of Comparable [interface in Java](https://www.edureka.co/blog/java-interface/) and **compareTo** method.

|  |
| --- |
| public class Student implements Comparable {  private String name;  private int age;  public Student(String name, int age) {  this.name = name;  this.age = age;  }  public int getAge() {  return this.age;  }  public String getName() {  return this.name;  }  @Override  public String toString() {  return "";  }  @Override  public int compareTo(Student per) {  if(this.age == per.age)  return 0;  else  return this.age > per.age ? 1 : -1;  }    public static void main(String[] args) {  Person e1 = new Person("Adam", 45);  Person e2 = new Person("Steve", 60);  int retval = e1.compareTo(e2);    switch(retval) {  case -1: {  System.out.println("The " + e2.getName() + " is older!");  break;  }    case 1: {  System.out.println("The " + e1.getName() + " is older!");  break;  }    default:  System.out.println("The two persons are of the same age!");    }  }  } |

In the above example, I have created a class Student with two fields, name and age. Class Student is implementing the Comparable interface and overrides the compareTo method. This method sorts the [instances](https://www.edureka.co/blog/instance-variable-in-java/) of the Student class, based on their age.

Now that I have covered Comparable in Java, moving on I will talk about another interface i.e. Comparator in [Java](https://www.edureka.co/blog/what-is-java/). Let’s move to understanding Comparator in Java!

## ****What is Comparator in Java?****

A Comparator interface is used to order the objects of a specific class. This interface is found in java.util package. It contains two methods;

* compare(Object obj1,Object obj2)
* equals(Object element).

The first method, compare(Object obj1,Object obj2)  compares its two input arguments and showcase the output. It returns a negative integer, zero, or a positive integer to state whether the first argument is less than, equal to, or greater than the second.

The second method, equals(Object element), requires an Object as a parameter and shows if the input object is equal to the comparator. The method will return true, only if the mentioned object is also a Comparator. The order remains the same as that of the Comparator.

|  |
| --- |
| import java.util.Comparator;    public class School {  private int num\_of\_students;  private String name;  public Company(String name, int num\_of\_students) {  this.name = name;  this.num\_of\_students = num\_of\_students;  }  public int getNumOfStudents() {  return this.num\_of\_students;  }  public String getName() {  return this.name;  }  }  public class SortSchools implements Comparator {  @Override  public int compare(School sch1, School sch2) {  if(sch1.getNumOfStudents()== sch2.getNumOfStudents())  return 0;  else  return sch1.getNumOfStudents() > sch2.getNumOfStudents() ? 1 : -1;  }  public static void main(String[] args) {  School sch1 = new School("sch1", 20);  School sch2 = new School("sch2", 15);  SortSchools sortSch = new SortSchools();  int retval = sortSch.compare(sch1, sch2);  switch(retval) {  case -1: {  System.out.println("The " + sch2.getName() + " is bigger!");  break;  }  case 1: {  System.out.println("The " + sch1.getName() + " is bigger!");  break;  }  default:  System.out.println("The two schools are of the same size!");  }  }  }    Output:  The sch1 is bigger! |

Well, no need to panic here. The above-written code is really easy to understand. Let’s go!

First, I created a class School that consists of the name and age of the students. After that, I created another class, SortSchools, in order to implement the Comparator interface which accomplishes the goal of imposing an order between instances of the first class named, School, according to the number of students.

After understanding about Comparator as well as Comparable in Java, let’s move towards our next topic.

## ****Comparable v/s Comparator in Java****

|  |  |
| --- | --- |
| Comparable in Java | Comparator in Java |
| Comparable interface is used to sort the objects with natural ordering. | Comparator in Java is used to sort attributes of different objects. |
| Comparable interface compares “this” reference with the object specified. | Comparator in Java compares two different class objects provided. |
| Comparable is present in java.lang package. | A Comparator is present in the java.util package. |
| Comparable affects the original class, i.e., the actual class is modified. | Comparator doesn’t affect the original class |
| Comparable provides compareTo() method to sort elements. | Comparator provides compare() method, equals() method to sort elements. |

I hope the above-mentioned differences brought some clarity regarding the two concepts.

## Com ****Comparable**** ****in Java****

in Java Comparable in Java

1. default nature of sorting order.
2. Present in java.lang
3. CompareTo() method only
4. Implemented in all wrapper and string class.

**Comparator in Java**

1. Custom sorting order.
2. Present in java.util
3. Compareto() and equal() method.
4. The only implemented class of comparator are collator and rulebasedcollator.

## ****What is Comparable in Java?****

## **Comparable** A comparable object is capable of comparing itself with another object.

As the name itself suggests, **Comparable** is an [interface](https://www.edureka.co/blog/java-interface/) which defines a way to compare an object with other objects of the same type. It helps to sort the objects that have self-tendency to sort themselves, i.e., the objects must know how to order themselves. **Eg:** Roll number, age, salary. This interface is found in java.lang package and it contains only one method, i.e., compareTo(). Comparable is not capable of sorting the [objects](https://www.edureka.co/blog/java-tutorial/#obj) on its own, but the interface defines a method int compareTo() which is responsible for sorting.

## ****What is the compareTo method and how it is used?****

This method is used to compare the given object with the current object. The **compareTo()** method returns an int value. The value can be either positive, negative, or zero. So now we are well acquainted with the theoretical knowledge of Comparable [interface in Java](https://www.edureka.co/blog/java-interface/) and **compareTo** method.

**What is Comparator in Java?**

**Comparator**  
A comparator object is capable of comparing two different objects. The class is not comparing its instances, but some other class’s instances.

A Comparator interface is used to order the objects of a specific class. This interface is found in java.util package. It contains two methods;

* compare(Object obj1,Object obj2)
* equals(Object element).

The first method, compare(Object obj1,Object obj2)  compares its two input arguments and showcase the output. It returns a negative integer, zero, or a positive integer to state whether the first argument is less than, equal to, or greater than the second.

The second method, equals(Object element), requires an Object as a parameter and shows if the input object is equal to the comparator. The method will return true, only if the mentioned object is also a Comparator. The order remains the same as that of the Comparator.

After attaining brief learning about Comparator in Java, it’s time to move a step ahead. Let me show you an example depicting Comparator in Java.

# [**What is the difference between compare() and compareTo()?**](https://stackoverflow.com/questions/420223/what-is-the-difference-between-compare-and-compareto)

* a.compareTo(b):  
  **Comparable interface :** Compares values and returns an int which tells if the values compare less than, equal, or greater than.  
  If your class objects have a **natural order**, implement the Comparable<T> interface and define this method. All Java classes that have a natural ordering implement Comparable<T> - Example: String, [wrapper classes](http://www.javatpoint.com/wrapper-class-in-java), BigInteger
* compare(a, b):  
  **Comparator interface :** Compares values of two objects. This is implemented as part of the Comparator<T> interface, and the **typical use is to define one or more small utility classes that implement this, to pass to methods such as sort() or for use by sorting data structures such as TreeMap and TreeSet**. You might want to create a Comparator object for the following:
  + **Multiple comparisons**. To provide several different ways to sort something. For example, you might want to sort a Person class by name, ID, age, height, ... You would define a Comparator for each of these to pass to the sort() method.
  + **System class** To provide comparison methods for classes that you have no control over. For example, you could define a Comparator for Strings that compared them by length.
  + **Strategy pattern** To implement a Strategy pattern, which is a situation where you want to represent an algorithm as an object that you can pass as a parameter, save in a data structure, etc.

If your class objects have one natural sorting order, you may not need compare().