Java Sorting Guide: Comparable vs Comparator

1. Comparable

- **Purpose:** Defines natural ordering of objects.
- **Implemented by:** The class whose objects are being sorted.
- **Package:** java.lang
- **Method:** `int compareTo(T o)`

Example:

```
public class Student implements Comparable<Student> {
    private String name;
    private int age;

    public Student(String name, int age) {
        this.name = name;
        this.age = age;
    }

    @Override
    public int compareTo(Student other) {
        return this.age - other.age;
    }
}

List<Student> students = new ArrayList<>();
students.add(new Student("Alice", 23));
students.add(new Student("Bob", 20));
Collections.sort(students); // uses compareTo
```

Key Points: - Only one natural ordering per class. - Implemented inside the class.

2. Comparator

- **Purpose:** Defines custom ordering, multiple ways possible.
- **Implemented by:** Separate class or lambda expression.
- **Package:** java.util
- **Method:** `int compare(T o1, T o2)`

Example:

```
import java.util.Comparator;
```

```java
public class StudentNameComparator implements Comparator<Student> {
    @Override
    public int compare(Student s1, Student s2) {
        return s1.getName().compareTo(s2.getName());
    }
}

Collections.sort(students, new StudentNameComparator());
```

Key Points: - Multiple ways to sort. - Does not modify the class itself.

Comparison Table

Feature	Comparable	Comparator
Package	java.lang	java.util
Implemented by	Class itself	Separate class/lambda
Methods	compareTo(T o)	compare(T o1, T o2)
Number of ways	1 (natural)	Multiple
Modifies class?	Yes	No