Comparable and Comparator are two interfaces used for sorting objects, but they serve different purposes.

**1. Comparable Interface**

* Used when you want to define a **natural ordering** for the objects of a class.
* It is implemented by the class whose objects you want to sort.
* The class needs to implement the compareTo(T o) method, which compares the current object (this) with another object (o).

A black background with red letters

AI-generated content may be incorrect.

A black background with white text

AI-generated content may be incorrect.

A computer screen shot of a program code

AI-generated content may be incorrect.

**2. Comparator Interface**

* Used when you want to **define an external comparator** to sort objects, **without modifying the class itself**.
* You can create multiple comparator classes for different sorting strategies (e.g., sorting by name, age, etc.).
* The Comparator interface has two commonly used methods:
  + compare(T o1, T o2) — Compares two objects for order.
  + equals(Object obj) — (Inherited from Object), checks if two comparators are equal. It's usually not used.

A black background with white text

AI-generated content may be incorrect.

A black and white text box

AI-generated content may be incorrect.

A computer screen shot of code

AI-generated content may be incorrect.

import java.util.\*;

class Student implements Comparable<Student> {

int rollNumber;

String name;

public Student(int rollNumber, String name) {

this.rollNumber = rollNumber;

this.name = name;

}

@Override

public int compareTo(Student other) {

return Integer.compare(this.rollNumber, other.rollNumber); // Default sorting by rollNumber

}

}

class NameComparator implements Comparator<Student> {

@Override

public int compare(Student s1, Student s2) {

return s1.name.compareTo(s2.name); // Sorting by name

}

}

public class Main {

public static void main(String[] args) {

List<Student> students = new ArrayList<>();

students.add(new Student(1, "Alice"));

students.add(new Student(3, "Bob"));

students.add(new Student(2, "Charlie"));

// Sorting by roll number (using Comparable)

Collections.sort(students);

for (Student student : students) {

System.out.println(student.rollNumber + " - " + student.name);

}

System.out.println("-----");

// Sorting by name (using Comparator)

Collections.sort(students, new NameComparator());

for (Student student : students) {

System.out.println(student.rollNumber + " - " + student.name);

}

}

}