Differences between TreeMap, HashMap and LinkedHashMap in Java
Differences between HashMap and HashTable in Java
Polymorphism in Java
Main/varargs absolutes difference in an array
What's the connection between Java and Blockchain?
Difference between static and non static method in Java
Spring Dependency injection with Example
Count occurrences of a given
Difference between static and non static variables in Java
Message Passing in Java
File Handling in Java with OWIO operations
What is the Role of Java in the IT industry?
Singleton multiple() Method in Java with Examples
Difference between concat() and + operator in Java
Difference between String and Character array in Java
Recursion in Java
Different Approaches to Concurrent Programming in Java
Java program to check whether a string is a Palindrome
How to create a Java Multiplayer or user defined class type?
Differences between interface and Class in Java
How to read a Matrix from user in Java?
Singleton add() Method in Java with Examples
Creating an Asynchronous Multithreaded chat Application in Java
Implementing next_permutation() in Java with Examples
Difference between while and do-while loop in C, C++, Java
Aspect Oriented Programming and AOP in Spring Framework
ClassLoader in Java
Different ways for String to Integer Conversions in Java
Count the pairs of vowels in the given string
How to Clone a List in Java?

Classroom program • Online
Live classes • 10 November 2019
Machine Learning
C++ | Python | Java
7TH,8PM
Beginner Batch
Expert Mentors
Flexible Timings
Live Interactive Classes
Doubts Clearing Sessions
Hands-on Projects
Industry Oriented Curriculum
Placement Assistance
Certification
Lifetime Access
Enrollment Ends 10th November 2019
2

Comparable vs Comparator in Java

Java provides two interfaces to sort objects using data members of the class:

1. Comparable
2. Comparator

Using Comparable Interface

A comparable object is capable of comparing itself with another object. The class itself must implements the **Java.lang.Comparable** interface to compare its instances.

Consider a Movie class that has members rating, name, year. Suppose we wish to sort a list of Movies based on year of release. We can implement the Comparable interface with the Movie class, and we override the method compareTo() of Comparable interface.

Ad
amcat
Get Hired
5 lacs+ opportunities created in a year
A good AMCAT score great opportunities for you
amcat AMCAT

```
// A Java program to demonstrate use of Comparable
import java.lo.*;
import java.util.*;

// A class 'Movie' that implements Comparable
class Movie implements Comparable<Movie>
{
    private double rating;
    private String name;
    private int year;

    // Used to sort movies by year
    public int compareTo(Movie m)
    {
        return this.year - m.year;
    }

    // Constructor
    public Movie(String nm, double rt, int yr)
    {
        this.name = nm;
        this.rating = rt;
        this.year = yr;
    }

    // Getter methods for accessing private data
    public double getRating() { return rating; }
    public String getName() { return name; }
    public int getYear()    { return year; }
}

// Driver class
class Main
{
    public static void main(String[] args)
    {
        ArrayList<Movie> list = new ArrayList<Movie>();
        list.add(new Movie("Force Awakens", 8.3, 2015));
        list.add(new Movie("Star Wars", 8.7, 1977));
        list.add(new Movie("Empire Strikes Back", 8.8, 1980));
        list.add(new Movie("Return of the Jedi", 8.4, 1983));

        Collections.sort(list);

        System.out.println("Movies after sorting :");
        System.out.println("Name Rating Year");
        for (Movie movie: list)
            System.out.println(movie.getName() + " " +
                               movie.getRating() + " " +
                               movie.getYear());
    }
}
```

Output:

```
Movies after sorting :
Star Wars 8.7 1977
Empire Strikes Back 8.8 1980
Return of the Jedi 8.4 1983
Force Awakens 8.3 2015
```

Note: suppose we want sort movies by their rating and names also. When we make a collection element comparable(by having it implement Comparable), we get only one change to implement the compareTo() method. The solution is using Comparator.

Using Comparator

Unlike Comparable, Comparator is external to the element type we are comparing. It's a separate class. We can create multiple separate classes (that implement Comparator) to compare by different members. Collections class has a second sort() method and it takes Comparator. The sort() method invokes the compare() to sort objects.

To compare movies by Rating, we need to do 3 things :

1. Create a class that implements Comparator (and thus the compare() method that does the work previously done by compareTo()).
2. Make an instance of the Comparator class.
3. Call the overloaded sort() method, giving it both the list and the instance of the class that implements Comparator.

```
// A Java program to demonstrate Comparator interface
import java.io.*;
import java.util.*;

// A class 'Movie' that implements Comparable
class Movie implements Comparable<Movie>
{
    private double rating;
    private String name;
    private int year;

    // Used to sort movies by year
    public int compareTo(Movie m)
    {
        return this.year - m.year;
    }

    // Constructor
    public Movie(String nm, double rt, int yr)
    {
        this.name = nm;
        this.rating = rt;
        this.year = yr;
    }

    // Getter methods for accessing private data
    public double getRating() { return rating; }
    public String getName() { return name; }
    public int getYear()    { return year; }
}

// Class to compare Movies by ratings
class RatingComparer implements Comparator<Movie>
{
    public int compare(Movie m1, Movie m2)
    {
        if (m1.getRating() > m2.getRating()) return -1;
        if (m1.getRating() < m2.getRating()) return 1;
        else return 0;
    }
}

// Class to compare Movies by name
class NameComparer implements Comparator<Movie>
{
    public int compare(Movie m1, Movie m2)
    {
        return m1.getName().compareTo(m2.getName());
    }
}

// Driver class
class Main
{
    public static void main(String[] args)
    {
        ArrayList<Movie> list = new ArrayList<Movie>();
        list.add(new Movie("Force Awakens", 8.3, 2015));
        list.add(new Movie("Star Wars", 8.7, 1977));
        list.add(new Movie("Empire Strikes Back", 8.8, 1980));
        list.add(new Movie("Return of the Jedi", 8.4, 1983));

        // Sort by rating : (1) Create an object of ratingComparer
        //                  (2) Call Collections.sort
        //                  (3) Print Sorted List
        System.out.println("Sorted by rating");
        RatingComparer ratingComparer = new RatingComparer();
        Collections.sort(list, ratingComparer);
        for (Movie movie: list)
            System.out.println(movie.getName() + " " +
                               movie.getRating() + " " +
                               movie.getYear());

        // Call overloaded sort method with RatingComparer
        // (same three steps as above)
        System.out.println("Sorted by name");
        NameComparer nameComparer = new NameComparer();
        Collections.sort(list, nameComparer);
        for (Movie movie: list)
            System.out.println(movie.getName() + " " +
                               movie.getRating() + " " +
                               movie.getYear());

        // Then Comparable to sort by year
        System.out.println("Sorted by year");
        Collections.sort(list);
        for (Movie movie: list)
            System.out.println(movie.getName() + " " +
                               movie.getRating() + " " +
                               movie.getYear());
    }
}
```

Output :

```
Sorted by rating
8.3 Force Awakens 2015
8.4 Return of the Jedi 1983
8.7 Star Wars 1977
8.8 Empire Strikes Back 1980

Sorted by name
Empire Strikes Back 8.8 1980
Force Awakens 8.3 2015
Return of the Jedi 8.4 1983
Star Wars 8.7 1977

Sorted by year
1977 8.7 Star Wars
1980 8.8 Empire Strikes Back
1983 8.4 Return of the Jedi
2015 8.3 Force Awakens
```

- Comparable is meant for objects with natural ordering which means the object itself must know how it is to be ordered. For example lists of numbers of students. Whereas, Comparator interface sorting is done through a separate class.
- Logically, Comparable interface compares "this" reference with the object specified and Comparator in Java compares two different class objects provided.
- If any class implements Comparable interface in Java then collection of that object either List or Array can be sorted automatically by using Collections.sort() or Arrays.sort(). Method objects are sorted on the sorted based on there natural order defined by CompareTo method.

To summarize, if sorting of objects needs to be based on natural order then use Comparable whereas if you sorting needs to be done on attributes of different objects, then use Comparator in Java.

This article is contributed by **Sourabeep Banua**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

Ad
Coding interview books
Learn & discuss programming problem Algo and data structure
ofile.google.com
OPEN

Recommended Posts:

Java.util.BitSet class methods in Java with Examples | Set 2
Shadowing of static functions in Java
How does default virtual behavior differ in C++ and Java ?
How are Java objects stored in memory?
How are parameters passed in Java?
Are static local variables allowed in Java?
final variables in Java
Default constructor in Java
Assigning values to static final variables in Java
Comparison of Exception Handling in C++ and Java
Does Java support goto?
Arrays in Java
Inheritance and constructors in Java
More restrictive access to a derived class method in Java
Comparison of static keyword in C++ and Java

Ad
shutterstock

Article Tags : Sort
Practice Tags : Sort

To do Done

Based on 34 vote(s)

2.7

Feedback/ Suggest Improvement Add Notes Improve Article
Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Previous
Networking in Java | Set 1 (Java.net.InetAddress class)
Next
Myth about the file name and class name in Java

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.
7 Comments GeeksforGeeks Login
Recommend 3 Tweet Share Sort by Newest
Join the discussion…
LOG IN WITH OR SIGN UP WITH DISQUS ?
Name
Frogbeness · 2 years ago · edited
Missing from this article is the fact that there can be only one comparison implemented using comparable. On the other hand, you can write as many custom comparators as you want for a given type, all using different interpretations of what sorting means.
Reply · Share ›
Manoj Sahoo · 3 years ago
what is natural order in below summary?
"To summarize, if sorting of objects needs to be based on natural order then use Comparable whereas if you sorting needs to be done on attributes of different objects, then use Comparator in Java."
Reply · Share ›
Frogbeness ➡ Manoj Sahoo · a year ago
Natural order for b,c,a would be a,b,c.
Natural order for 3,1,4,2 would be 1,2,3,4.
In these cases, the element to be sorted has intrinsic properties that can be used to provide a sort order without any additional inputs. So when implementing the above as a comparable you do not need any additional input, you can easily use comparison operators to deduce the right order.
Contrast this with another scenario where b,c,a could be sorted as c,a,b based on some external input like their frequency in a string. Here you need additional input of frequency that is not available to just looking at the character c,a,b (unless you have used a custom type to represent the element and also its frequency).
That being said, I do feel the word 'natural' opens it up for a lot of interpretations. My explanation only serves to highlight the probable view of the author of what 'natural' means.
I think a more helpful description would be:
- Use comparable if there is exactly one way to compare elements of this type and that makes the most sense for most use cases of this type.
- Use comparator when there is more than one way of comparing 2 objects of a type and you want to give the users of this type a flexibility in implementing an ordering which might be custom written for their unique scenarios (and might be considered unnatural).
Refer: https://stackoverflow.com/q/...
Reply · Share ›
Manoj Sahoo ➡ Frogbeness · 5 months ago
Thanks!!
Reply · Share ›
Ravikumar · 3 years ago
public class ComparatorDemo {
public static void main(String[] args) {
ArrayList<Movie> listMovies = new ArrayList<Movie>();
list.add(new Movie("1", "name", 2016));
list.add(new Movie("2", "Numa", 2011));
list.add(new Movie("3", "nada", 20));
list.add(new Movie("78", "kambum", 2013));
list.add(new Movie("3", "raju", 2012));
System.out.println(list);
ComparatorNewDemo cmpr=new ComparatorNewDemo();
Collections.sort(list,cmpr);
Reply · Share ›
Paul2001 · 3 years ago
Though its obvious or can be inferred from the above article, but still to make it more explicit :
Comparable is used when the class whose objects are to be compared is a 3rd party classes (classes that you can't modify either because they are shipped to you by other vendor or developed by other team who provide you only the jars) classes same is not true for Comparator interface.
Reply · Share ›
Ravikumar ➡ Paul2001 · 3 years ago
Check the above code.
Reply · Share ›
Subscribe Add Disqus to your site Disqus' Privacy Policy
DISQUS

simplilearn
Boost your career with Simplilearn's
Artificial Intelligence Engineer
Master's Program and Job Assist.
₹44,999
GET STARTED
15 IT Drivers
DSA
SELF-PACED COURSE
STUDENTS PLACED IN TOP PRODUCT BASED COMPANIES AMAZON MICROSOFT ADOBE
WHAT ARE YOU WAITING FOR ?
₹2,499
Register Now
2000+
ENROLLMENTS TO DATE

Most popular in Java
First characters and their frequencies in order of occurrence using a LinkedHashMap in Java
How to overload and override main method in Java
Difference between the Constructors and Methods
Lambda Expressions | Concurrent Programming Approach-4
Count occurrences of a given character in a string using Stream API in Java

Ad
Sencos Gold and Diamonds Nearby
OPEN
Sencos Gold and Diamonds
Cashless Dettiyar Jewellery Only At Sencos Gold and Diamonds
Jack Code
WEBSITE DIRECTIONS

More related articles in Java
Number(int val) getQuiet(Integer instance) method in Java with Examples
Introduction to Spring Boot
Difference between an Iterator and ListIterator in Java
How to Compile and Run C/C++/Java Programs in Linux
What are the main differences between the Java platform and other platforms?

Starting from 1st September 2019
SDE
TEST SERIES
₹999
₹199
Register Now
ONE TIME OFFER
Test Series
Features
Job

Learn and code with the best industry experts