Java Journal

Just sharing some knowledge...

When to use Comparable and When to use Comparator in java?

Now from part 1, 2, 3 and 4 you have learned *how to use Comparable and Comparator* in java and now you also know that main purpose of both interfaces is to compare things *so that we can sort them*.

Often some questions are asked in interviews with respect to Comparable and Comparator like:

1) When to use Comparable and When to use Comparator in java. OR

2) What is the difference between Comparable and Comparator in java? OR

3) Which is preferred Comparable or Comparator? OR

4) Why do you need Comparable if Comparator can also sort things?

5) Or interviewer may give you some class and some criteria and ask you to write code to sort instances of that class based on that criteria.You have to decide whether to use Comparable or to use Comparator. or even, not to use any of them.

This article outline some of the facts which help you decide in which situations one should use Comparable and in which Comparator.

1) Code Availabilty

The first thing to note is that while using Comparable you have to implement it in your class i.e *you need to change your class*. Example

```
public class Book implements Comparable{
   ...
```

```
}
```

For this, code of that class should be availaible to you . If you dont have access to the code of that class (say class belongs to third party), then there is no choice but to use Comparator because *Comparator does not need to change the original class.*

2) Single Versus Multiple Sorting Criteria

If you have only single sorting criteria to sort your elements then you can use Comparable but if you have more than one sorting criterias then you have to go for Comparator *also*.

3) Arays.sort() and Collection.sort()

Using Comparable has a advantage over Comparator. If your class implements Comparable then Arrays.sort() and Collections.sort() can sort its instances automatically. You do not need to write *separate comparators* and pass them to *overloaded* sort() as shown here.

4) As keys in SortedMap and SortedSet

This is another advantage of Comparable over Comparator. Objects which implement Comparable interface can be used as keys in a SortedMap(like TreeMap) or as elements in a SortedSet (like TreeSet). Otherwise you have to write separate Comparator and pass it in the constructor of TreeMap.

5) More Number of classes Versus flexibilty

Use of Comparable does not require creation of extra classes while use of Comparator requires writing of *separate comparators* i.e *more number of classes*.

But this has a advantage also.You can add as many sorting criteria later as you want or modify the existing ones without changing the class whose instances you are sorting.

Thus comparators provides flexibilty while Comparable avoids extra classes.

Note that you can also write Comparators as anonymous classes. In that case you can avoid separate comparators also.

6) Interclass comparisions

If you are going to compare *instances of same class* then you *should* use Comparable. Though we can also compare objects of different types while using Comparable as shown here but we should avoid it.

If you are going to compare instances of different classes then you should use Comparator. But this was valid upto pre

Sorting based on multiple attributes of a Class.

When to use Comparable and When to use Comparator ...

Comparable and Comparator -Part 3

Comparable and Comparator -Part 4

Comparable and Comparator -Part 2

Comparable and Comparator -Part 1

► 2010 (24)

Java 5, before the introduction of generics.

With the introduction of generics syntax of Comparator has been changed from :

`public interface Comparator` to `public interface Comparator <T>`

and of compare() from:

`public int compare(Object o1, Object o2)` to `public int compare(T o1,T o2)`

As you are seeing in new syntax both o1 and o2 are of type T. If their types would be different as:

```
compare(Integer o1, String o2)
```

then this will give compile time error.

Therefore if generic form of compare() is used then it compares objects of only same types.

With non generic form of compare() you can still compare objects of different types.

7) Natural Order

If you are going to sort elements according to their natural order then you should use Comparable and for any other order different from natural order Comparator should be used.

Thus to answer above given questions:-

1) When to use Comparable and When to use Comparator?

Ans:- When you are going to sort according to natural order, have single sort criteria and have access to the class you would use Comparable.

Otherwise,

If you cannot change the class and have multiple sorting criteria use Comparator.

2) Interviewer may give you some class and some criteria and ask you to write code to sort instances of that class based on that criteria.You have to decide there whether to use Comparable or to use Comparator. or even, not use any of them?

Ans: For example,
If you are given a class named Employee and asked to sort it's instances by employees *id*, then you should go Comparable as it seems its natural order.

But if you have to sort them by its salary or Date of Joining () etc.(which does not seem to be natural order of employee) then you can go for either Comparable or Comparator. Both are legal. If you would use Comparator then you have to write extra classes as comparator.

And if you have to sort employees on id and *also* on salary and Date of Joining, then use Comparable for *id* ad create comparators for salary and Date of joining.

Suppose you are given list of objects of type *Integer*. Then which interface you will use? None, as Integer already implements Comparable. Just pass its array (or List) to Arrays.sort() (or Collections.sort()).

3) What is the difference between Comparable and Comparator?

Check here for this: Difference between Comparable and Comparator Interfaces.

4) Which is preffered Comparable or Comparator?

Comparable has advantage that it avoids the creation of more number of classes and also elements implementing Comparable can be used directly in utility functions like Arrays.sort() and Collections.sort().

While Comparator is more flexible.It has advantage that it avoids the changing the class you are going to sort and more sorting criterias can be added later.

So in my view one should go for Comparable first for that sorting criteria *which is not going to change in future* and for additional criterias we can use Comparator in addition also.

5) Why do you need Comparable if Comparator can also sort things?

Because Comparable :-

1) makes it easy to use the elements implementing it in some utility functions and classes like TreeSet and TreeMap.

2) Avoids creation of new classes.

I would like to know your comments and if you liked the article then please share it on social networking buttons.

Posted by Rajinder Singh 

18 comments:

Anonymous 4 August 2013 at 05:37

Really Nice Article :).Keep up the Good Work.

Reply

Anonymous 11 October 2013 at 10:02

Thanks for giving clear view of comparable vs comparator

Reply

Anonymous 17 December 2013 at 20:09

very very nice and clear article.....

Reply

Anonymous 4 September 2014 at 08:04

Excellent:)

Reply

Anonymous 24 January 2015 at 18:16

Very helpful

Reply

Anonymous 5 June 2015 at 08:45

superrrrrrrrrr

Reply

amar singh 25 September 2015 at 20:49

Really appreciable job done by you....

Reply

Replies

Rajinder 27 September 2015 at 06:29

Thanks Amar.

Reply

VILAS DHORMARE 24 December 2015 at 07:11

Very good explanation... Thank you very much

Reply

Vishal Aggarwal 21 April 2016 at 20:09

Very informative post, thanks

Reply

Suraj Hirade 15 July 2016 at 00:45

Very Very Nice Explanation.. It's clearing all doubt.Thanks!!!

Reply

HVGowda 19 October 2016 at 09:35

Perfect.. Very help full.. Thank you very much

Reply

tushar pandit 9 January 2017 at 20:59

Excellent explanation. It would be great if you provide an example for "And if you have to sort employees on id and also on salary and Date of Joining, then use Comparable for id ad create comparators for salary and Date of joining."

Thanks,
Tushar

Reply

tushar pandit 9 January 2017 at 21:00

Very nice article. It would be great if you could provide an example for "And if you have to sort employees on id and also on salary and Date of Joining, then use Comparable for id ad create comparators for salary and Date of joining."

Reply

Anonymous 18 May 2017 at 22:38

very nice and brief explanation. thank you so much.

Reply

anurag singh 19 June 2017 at 05:16

this is really very nice post about comparable and comparator in java
thank you so much for this nice and useful post.

Reply

Anonymous 20 September 2017 at 18:19

The given answer beats other blog sites..

"For this, code of that class should be available to you. If you don’t have access to the code of that class (say class belongs to third party), then there is no choice but to use Comparator because Comparator does not need to change the original class."

Wonderful answer. This is what I was exactly looking for.. It took around 4 days to find exactly what I want.. Many Thanks..

Reply

Piyush Kumar 6 January 2018 at 11:11

Very nice article.. thank you so much!!

Reply

Enter your comment...

Comment as: Amitabh Mandal Sign out

Publish Preview Notify me

Links to this post

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)