**Comparator vs Comparable**

<http://stackoverflow.com/questions/19682818/collections-sort-using-comparator>  
<http://www.dreamincode.net/forums/topic/169079-how-collectionssort-is-doing-its-stuff-here/>

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Comparable** | **Comparator** |
| Sorting logic | Sorting logic must be in same class whose objects are being sorted. Hence this is called natural ordering of objects | Sorting logic is in separate class. Hence we can write different sorting based on different attributes of objects to be sorted. E.g. Sorting using id,name etc. |
| Implementation | Class whose objects to be sorted must implement this interface.e.g Country class needs to implement comparable to collection of country object by id | Class whose objects to be sorted do not need to implement this interface.Some other class can implement this interface. E.g.-CountrySortByIdComparator class can implement Comparator interface to sort collection of country object by id |
| Sorting method | int compareTo(Object o1) This method compares this object with o1 object and returns a integer.Its value has following meaning 1. positive – this object is greater than o1 2. zero – this object equals to o1 3. negative – this object is less than o1 | int compare(Object o1,Object o2) This method compares o1 and o2 objects. and returns a integer.Its value has following meaning. 1. positive – o1 is greater than o2 2. zero – o1 equals to o2 3. negative – o1 is less than o1 |
| Calling method | Collections.sort(List) Here objects will be sorted on the basis of CompareTo method | Collections.sort(List, Comparator) Here objects will be sorted on the basis of Compare method in Comparator |
| Package | Java.lang.Comparable | Java.util.Comparator |

**Java Programming questions**

<http://www.java2novice.com/java-interview-programs/>

**Other imp:**

* The Single Responsibility Principle
* Can you overload/override run() or start() method of thread
* Why does Map not extend Collection interface?
* Unit testing
* Design Patterns: <http://www.javacamp.org/designPattern/>
* Features of Java <http://www.javatpoint.com/features-of-java>
* Static binding vs dynamic binding (type of object is determined at compile time and run time respectively) <http://www.javatpoint.com/static-binding-and-dynamic-binding>
* Exception handling:

<http://www.javatpoint.com/exception-handling-in-java>

* **Autoboxing unboxing**

<http://www.javatpoint.com/autoboxing-and-unboxing>

* **Static vs Non-Static nested class /Inner class**

<http://java67.blogspot.in/2012/10/nested-class-java-static-vs-non-static-inner.html>

* **Deadlock and preventing it.**

<http://javarevisited.blogspot.in/2010/10/what-is-deadlock-in-java-how-to-fix-it.html>

* **Use of finally block:**

<http://tutorials.jenkov.com/java-exception-handling/basic-try-catch-finally.html>

The main usage of finally block is to do clean up job. Keeping cleanup code in a finally block is always a good practice, even when no exceptions are occured. - See more at: <http://www.java2novice.com/java_exception_handling_examples/finally_block/#sthash.EKGEAC8B.dpuf>

# [Is there an advantage to use a Synchronized Method instead of a Synchronized Block?](http://stackoverflow.com/questions/574240/is-there-an-advantage-to-use-a-synchronized-method-instead-of-a-synchronized-blo)

<http://stackoverflow.com/questions/574240/is-there-an-advantage-to-use-a-synchronized-method-instead-of-a-synchronized-blo>

**Collection questions:**

**Diff between arraylist and linkedlist -** <http://beginnersbook.com/2013/12/difference-between-arraylist-and-linkedlist-in-java/>

<http://www.javabeat.net/difference-arraylist-vector-linkedlist-java/#sthash.v0B51CnO.dpuf>

<http://www.javafaq.nu/java-article1111.html>

**Difference between array and arraylist:**

<http://java67.blogspot.in/2012/12/difference-between-array-vs-arraylist-java.html>

**How hashmap works in java?**

<http://javarevisited.blogspot.in/2011/02/how-hashmap-works-in-java.html>

**Difference between hashmap and hashtable.**

<http://javarevisited.blogspot.in/2010/10/difference-between-hashmap-and.html>

# [What's the difference between ConcurrentHashMap and Collections.synchronizedMap(Map)?](http://stackoverflow.com/questions/510632/whats-the-difference-between-concurrenthashmap-and-collections-synchronizedmap)

<http://stackoverflow.com/questions/510632/whats-the-difference-between-concurrenthashmap-and-collections-synchronizedmap>

**Fail fast vs fail safe iterator.**

<http://javahungry.blogspot.com/2014/04/fail-fast-iterator-vs-fail-safe-iterator-difference-with-example-in-java.html>

**Difference between arraylist and vector**

<http://beginnersbook.com/2013/12/difference-between-arraylist-and-vector-in-java/>

# [Difference between Iterator and Listiterator?](http://stackoverflow.com/questions/10977992/difference-between-iterator-and-listiterator)

# <http://stackoverflow.com/questions/10977992/difference-between-iterator-and-listiterator>

# <http://beginnersbook.com/2014/06/difference-between-iterator-and-listiterator-in-java/>

# [Why do we need to use iterator on ArrayList in Java?](http://stackoverflow.com/questions/16000282/why-do-we-need-to-use-iterator-on-arraylist-in-java)

# (or) For vs Iterator.

<http://stackoverflow.com/questions/16000282/why-do-we-need-to-use-iterator-on-arraylist-in-java>

As you have stated iterator is used when you want to remove stuff whilst you iterate over the array contents. If you don't use an iterator but simply have a for loop and inside it use the remove method you will get exceptions because the contents of the array changes while you iterate through. e.g: you might think array size is 10 at the start of the for loop but it wont be the case once you remove stuff.. so when u reach the last loops probably there will be **IndexOutofBoundsException** etc.

# [For-each vs Iterator. Which will be the better option](http://stackoverflow.com/questions/18508786/for-each-vs-iterator-which-will-be-the-better-option)

# for-each is an advanced looping construct. Internally it creates an Iterator and iterates over the the Collection. Only possible advantage of using actual Iterator object over for-each construct is that you can modify your collection using Iterator's methods like .remove(). Modifying the collection without using Iterator's methods while iterating will lead to ConcurrentModificationException[.](http://docs.oracle.com/javase/7/docs/api/java/util/ConcurrentModificationException.html)

# <http://stackoverflow.com/questions/18508786/for-each-vs-iterator-which-will-be-the-better-option>

# [How Iterator's remove method actually remove an object](http://stackoverflow.com/questions/15993356/how-iterators-remove-method-actually-remove-an-object)

# <http://stackoverflow.com/questions/15993356/how-iterators-remove-method-actually-remove-an-object>

# [Why does Map not extend Collection interface](http://stackoverflow.com/questions/5700135/why-does-map-not-extend-collection-interface)

# <http://stackoverflow.com/questions/5700135/why-does-map-not-extend-collection-interface>

# [Difference between Java Enumeration and Iterator](http://stackoverflow.com/questions/948194/difference-between-java-enumeration-and-iterator)

# <http://javarevisited.blogspot.sg/2010/10/what-is-difference-between-enumeration.html>

# Iterators are fail-fast . i.e. when one thread changes the collection by add / remove operations , while another thread is traversing it through an Iterator using hasNext() or next() method, the iterator fails quickly by throwing ConcurrentModificationException . The fail-fast behavior of iterators can be used only to detect bugs. The Enumerations returned by the methods of classes like Hashtable, Vector are not fail-fast that is achieved by synchronizing the block of code inside the nextElement() method that locks the current Vector object which costs lots of time.

### What is CopyOnWriteArrayList in Java

<http://java67.blogspot.in/2012/09/what-is-copyonwritearraylist-in-java-example-vs-arraylist.html>

**Deprecated :**

A program element annotated @Deprecated is one that programmers are discouraged from using, typically because it is dangerous, or because a better alternative exists.

**String literal vs String new object**

**Equals vs ==**

**Thread sleep vs suspend vs wait**

<http://coddicted.com/difference-between-sleep-suspend-and-wait/>

<http://www.jguru.com/faq/view.jsp?EID=47127>

**How to kill a thread in java?**

<http://stackoverflow.com/questions/10961714/how-to-properly-stop-the-thread-in-java>

### What is the difference between Serializable and Externalizable interface in Java?

This is most frequently asked question in Java serialization interview. Here is my version Externalizable provides us writeExternal() and readExternal() method which gives us flexibility to control java serialization mechanism instead of relying on Java's default serialization. Correct implementation of Externalizable interface can [improve performance of application](http://javarevisited.blogspot.sg/2012/01/improve-performance-java-database.html) drastically.  
  
Read more: <http://javarevisited.blogspot.com/2011/04/top-10-java-serialization-interview.html#ixzz3QmeJLvXe>

### While serializing you want some of the members not to serialize? How do you achieve it?

Another frequently asked Serialization interview question. This is sometime also asked as what is the use of [transient variable](http://javarevisited.blogspot.sg/2011/09/transient-keyword-variable-in-java.html), does transient and [static variable](http://javarevisited.blogspot.sg/2011/11/static-keyword-method-variable-java.html) gets serialized or not etc. so if you don't want any field to be part of object's state then declare it either static or transient based on your need and it will not be included during Java serialization process.  
  
  
Read more: <http://javarevisited.blogspot.com/2011/04/top-10-java-serialization-interview.html#ixzz3QmgU2e00>

**Static vs transient variable / Serialization example with code**

<http://javabeginnerstutorial.com/core-java-tutorial/transient-vs-static-variable-java/>

--read comments in the above page.

Static transient var – (static governs the behavior not transient)

Final transient var – (final governs the behavior not transient)

**Java’s volatile keyword:**

<http://tutorials.jenkov.com/java-concurrency/volatile.html>

* Types of class loaders in java.

<http://javarevisited.blogspot.in/2012/12/how-classloader-works-in-java.html>

* Classpath vs Path

The PATH variable contains directories where binary files are located [ C:\Program Files\Java\jdk1.6.0\_07\bin; ]

<http://java67.blogspot.in/2012/08/what-is-path-and-classpath-in-java-difference.html>

Correct. And it is used by your OS and command shells to find executable programs. So that, for example, when you type java on the command line, each element of the PATH will be searched for an executable named java (or, on Windows, also java.exe or java.bat).

Garbage collection:

<http://www.javatpoint.com/Garbage-Collection>