**1. How does HashMap work in Java? (**[**answer**](http://java67.blogspot.com/2013/06/how-get-method-of-hashmap-or-hashtable-works-internally.html)**)**

This is *Classical Java Collection interview questions* which I have also discussed in my earlier article [how does HashMap works in Java](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html). This collection interview questions is mostly asked during AVP Role interviews on Investment-Banks and has a lot of follow-up questions based on the response of interviewee e.g. Why HashMap keys need to be [immutable](http://javarevisited.blogspot.com/2010/10/why-string-is-immutable-in-java.html), what is race conditions on HashMap and how HashMap resize in Java. For explanation and answers of these questions Please see earlier link.

**2. What is the difference between poll() and remove() method of Queue interface? (answer)**  
Though both poll() and remove() method from Queue is used to remove the object and returns the head of the queue, there is a subtle difference between them. If Queue is empty() then a call to remove() method will throw Exception, while a call to poll() method returns null. By the way, exactly which element is removed from the queue depends upon queue's ordering policy and varies between different implementation, for example, PriorityQueue keeps the lowest element as per Comparator or Comparable at head position. 

**3. What is the difference between fail-fast and fail-safe Iterators? (**[**answer**](http://java67.blogspot.com/2015/06/what-is-fail-safe-and-fail-fast-iterator-in-java.html)**)**

This is relatively *new collection interview questions* and can become trick if you hear the term fail-fast and fail-safe first time. Fail-fast Iterators throws ConcurrentModificationException when one [Thread](http://javarevisited.blogspot.com/2011/02/how-to-implement-thread-in-java.html) is iterating over collection object and other thread structurally modify Collection either by adding, removing or modifying objects on underlying collection. They are called fail-fast because they try to immediately throw Exception when they encounter failure. On the other hand [fail-safe Iterators](http://javarevisited.blogspot.com/2011/10/java-iterator-tutorial-example-list.html) works on copy of collection instead of original collection

**4. How do you remove an entry from a Collection? and subsequently what is the difference between the remove() method of Collection and remove() method of Iterator, which one you will use while removing elements during iteration?**  
  
Collection interface defines remove(Object obj) method to remove objects from Collection. List interface adds another method remove(int index), which is used to remove object at specific index. You can use any of these method to remove an entry from Collection, while not iterating. Things change, when you iterate. Suppose you are traversing a List and removing only certain elements based on logic, then you need to use Iterator's remove() method. This method removes current element from Iterator's perspective. If you use Collection's or List's remove() method during iteration then your code will throw ConcurrentModificationException. That's why it's advised to use Iterator remove() method to remove objects from Collection.

**5. What is the difference between Synchronized Collection and Concurrent Collection? (**[**answer**](http://javarevisited.blogspot.com/2011/04/difference-between-concurrenthashmap.html)**)**

Java 5 has added several new Concurrent Collection classes e.g. ConcurrentHashMap, CopyOnWriteArrayList, BlockingQueue etc, which has made Interview questions on Java Collection even trickier. Java Also provided a way to get Synchronized copy of collection e.g. ArrayList, HashMap by using Collections.synchronizedMap() Utility function.One Significant difference is that Concurrent Collections has better performance than synchronized Collection because they lock only a portion of Map to achieve concurrency and Synchronization. See the difference between Synchronized Collection and Concurrent Collection in Java for more details.

**6. What is the difference between Iterator and Enumeration? (**[**answer**](http://javarevisited.blogspot.com/2010/10/what-is-difference-between-enumeration.html)**)**

This is a beginner level collection interview questions and mostly asked during interviews of Junior Java developer up to experience of 2 to 3 years Iterator duplicate functionality of Enumeration with one addition of remove() method and both provide navigation functionally on objects of Collection.Another difference is that Iterator is more safe than Enumeration and doesn't allow another thread to modify collection object during iteration except remove() method and throws ConcurrentModificaitonException. See Iterator vs Enumeration in Java for more differences.

**7. How does HashSet is implemented in Java, How does it use Hashing? (**[**answer**](http://java67.blogspot.com/2014/01/how-hashset-is-implemented-or-works-internally-java.html)**)**  
This is a tricky question in Java because for hashing you need both key and value and there is no key for the store it in a bucket, then how exactly HashSet store element internally. Well, HashSet is built on top of HashMap. If you look at source code of java.util.HashSet class, you will find that that it uses a HashMap with same values for all keys, as shown below:  
  
private transient HashMap map;  
  
// Dummy value to associate with an Object in the backing Map  
private static final Object PRESENT = new Object();  
  
When you call add() method of HashSet, it put entry in HashMap :  
  
public boolean add(E e) {  
  return map.put(e, PRESENT)==null;  
}  
  
Since keys are unique in a HashMap, it provides uniqueness guarantee of Set interface.

**8. What do you need to do to use a custom object as a key in Collection classes like Map or Set? (**[**answer**](http://javarevisited.blogspot.com/2015/01/why-override-equals-hashcode-or-tostring-java.html)**)**  
The answer is: If you are using any custom object in Map as key, you need to override equals() and hashCode() method, and make sure they follow their contract. On the other hand if you are storing a custom object in Sorted Collection e.g. SortedSet or SortedMap, you also need to make sure that your equals() method is consistent to compareTo() method, otherwise that collection will not follow there contacts e.g. Set may allow duplicates.

**9. The difference between HashMap and Hashtable? (**[**answer**](http://javarevisited.blogspot.com/2015/08/difference-between-HashMap-vs-TreeMap-vs-LinkedHashMap-Java.html)**)**

This is another Classical Java Collection interview asked on beginner’s level and most of Java developer has a predefined answer for this interview questions e.g. HashMap is not synchronized while Hashtable is not or hashmap is faster than hash table etc. What could go wrong is that if he placed another follow-up question like how hashMap works in Java or can you replace Hashtable with ConcurrentHashMap etc. See [Hashtable vs HashMap in Java](http://javarevisited.blogspot.com/2010/10/difference-between-hashmap-and.html) for detailed answer of this interview question.

**10. When do you use ConcurrentHashMap in Java? (**[**answer**](http://javarevisited.blogspot.com/2011/04/difference-between-concurrenthashmap.html)**)**

This is another advanced level collection interview questions in Java which normally asked to check whether the interviewer is familiar with optimization done on ConcurrentHashMap or not. ConcurrentHashMap is better suited for situation where you have multiple readers and one

Writer or fewer writers since Map gets locked only during the write operation. If you have an equal number of reader and writer than [ConcurrentHashMap](http://javarevisited.blogspot.com/2011/04/difference-between-concurrenthashmap.html) will perform in the line of Hashtable or synchronized HashMap.

**11. What is the difference between Set and List in Java? (**[**answer**](http://java67.blogspot.com/2012/08/difference-between-list-and-set-in-java.html)**)**

Another classical Java Collection interviews popular on telephonic round or the first round of interview. Most of Java programmer knows that Set doesn't allowed duplicate while List does and List maintains insertion order while Set doesn't. What is key here is to show the interviewer that you can decide which collection is more suited based on requirements.

**12. How do you Sort objects on the collection? (**[**solution**](http://java67.blogspot.com/2012/07/sort-list-ascending-descending-order-set-arraylist.html)**)**

This Collection interview question serves two purpose it not only test an important programming concept Sorting but also utility class like Collections which provide several methods for creating synchronized collection and sorting. Sorting is implemented using Comparable and Comparator in Java and when you call Collections.sort() it gets sorted based on the natural order specified in compareTo() method while Collections.sort(Comparator) will sort objects based on compare() method of Comparator.

**13. What is the difference between Vector and ArrayList? (**[**answer**](http://java67.blogspot.com/2012/09/arraylist-vs-vector-in-java-interview.html)**)**

One more beginner level collection interview questions, this is still very popular and mostly asked in the telephonic round. [ArrayList in Java](http://javarevisited.blogspot.com/2011/05/example-of-arraylist-in-java-tutorial.html) is one of the most used Collection class and the most interviewers asked questions on ArrayList. See Difference between Vector and ArrayList for the answer to this interview question.

**14. What is the difference between HashMap and HashSet? (**[**answer**](http://java67.blogspot.com/2012/08/difference-between-hashset-and-hashmap.html)**)**

This collection interview questions is asked in conjunction with HashMap vs Hashtable. HashSet implements java.util.Set interface and that's why only contains unique elements, while HashMap allows duplicate values.  In fact, HashSet is actually implemented on top of java.util.HashMap. If you look internal implementation of java.util.HashSet, you will find that it adds element as key on internal map with same values. For a more detailed answer, see [HashMap vs HashSet](http://javarevisited.blogspot.com/2011/09/difference-hashmap-vs-hashset-java.html).

|  |  |
| --- | --- |
| **HashMap** | **Hash Set** |
| HashMap  is an implementation of Map interface | HashSet is an implementation of Set Interface |
| HashMap Stores data in form of  key-value pair | HashSet Store only objects |
| Put method is used to add element in map | Add method is used to add element is Set |
| In hash map hashcode value is calculated using key object | Here member object is used for calculating hashcode value which can be same for two objects so equal () method is used to check for equality if it returns false that means two objects are different. |
| HashMap is faster than HashSet because unique key is used to access object | HashSet is slower than Hashmap |

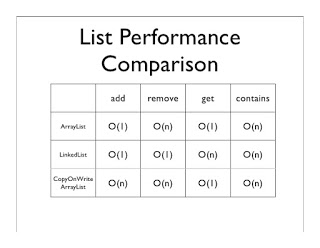
**15) What is NavigableMap in Java? What is a benefit over Map? (**[**answer**](http://javarevisited.blogspot.com/2013/01/what-is-navigablemap-in-java-6-example-submap-head-tail.html)**)**  
NavigableMap Map was added in Java 1.6, it adds navigation capability to Map data structure. It provides methods like lowerKey() to get keys which is less than specified key, floorKey() to return keys which is less than or equal to specified key, ceilingKey() to get keys which is greater than or equal to specified key and higherKey() to return keys which is greater specified key from a Map. It also provide similar methods to get entries e.g. lowerEntry(), floorEntry(), ceilingEntry() and higherEntry(). Apart from navigation methods, it also provides utilities to create sub-Map e.g. creating a Map from entries of an exsiting Map like tailMap, headMap and subMap. headMap() method returns a NavigableMap whose keys are less than specified, tailMap() returns a NavigableMap whose keys are greater than the specified and subMap() gives a NavigableMap between a range, specified by toKey to fromKey.    
  
  
  
**16) Which one you will prefer between Array and ArrayList for Storing object and why? (**[**answer**](http://java67.blogspot.com/2012/12/difference-between-array-vs-arraylist-java.html)**)**  
Though ArrayList is also backed up by array, it offers some usability advantage over array in Java. Array is fixed length data structure, once created you can not change it's length. On the other hand, ArrayList is dynamic, it automatically allocate a new array and copies content of old array, when it resize. Another reason of using ArrayList over Array is support of Generics. Array doesn't support Generics, and if you store an Integer object on a String array, you will only going to know about it at runtime, when it throws ArrayStoreException. On the other hand, if you use ArrayList, compiler and IDE will catch those error on the spot. So if you know size in advance and you don't need re-sizing than use array, otherwise use ArrayList.

**17) Can we replace Hashtable with ConcurrentHashMap? (**[**answer**](http://java67.blogspot.com/2014/07/21-frequently-asked-java-interview-questions-answers.html)**)**

Answer 3: Yes we can replace Hashtable with ConcurrentHashMap and that's what suggested in Java documentation of ConcurrentHashMap. but you need to be careful with code which relies on locking behavior of Hashtable. Since Hashtable locks whole Map instead of a portion of Map, compound operations like if(Hashtable.get(key) == null) put(key, value) works in Hashtable but not in concurrentHashMap. instead of this use putIfAbsent() method of ConcurrentHashMap

**18) What is CopyOnWriteArrayList, how it is different than ArrayList and Vector? (**[**answer**](http://java67.blogspot.com/2015/06/difference-between-synchronized-arraylist-and-copyOnWriteArrayList-java.html)**)**

Answer: CopyOnWriteArrayList is new List implementation introduced in Java 1.5 which provides better concurrent access than Synchronized List. better concurrency is achieved by Copying ArrayList over each write and replace with original instead of locking. Also CopyOnWriteArrayList doesn't throw any ConcurrentModification Exception. Its different than ArrayList because its thread-safe and ArrayList is not thread-safe and it's different than Vector in terms of Concurrency. CopyOnWriteArrayList provides better Concurrency by reducing contention among readers and writers. Here is a nice table which compares performance of three of popular List implementation ArrayList, LinkedList and CopyOnWriteArrayList in Java:

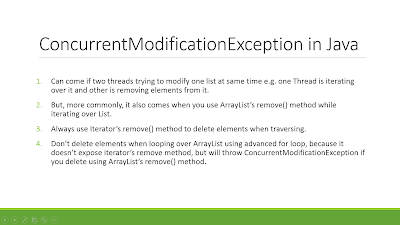
[](https://3.bp.blogspot.com/-C3omsD-5Dmk/VmQRsg6xYjI/AAAAAAAAEQk/Q2h35mRULLc/s1600/ArrayList%2BLinkedList%2Band%2BCopyOnWriteArrayList.jpg)

**19) Why ListIterator has added() method but Iterator doesn't or Why to add() method is declared in ListIterator and not on Iterator. (**[**answer**](http://java67.blogspot.com/2013/02/java-iterator-example-and-tutorial.html)**)**

Answer: ListIterator has added() method because of its ability to traverse or iterate in both direction of the collection. it maintains two pointers in terms of previous and next call and in a position to add a new element without affecting current iteration.

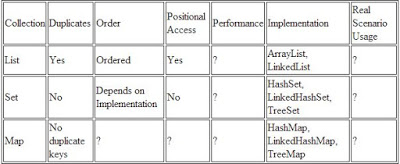
**20) When does ConcurrentModificationException occur on iteration? (**[**answer**](http://java67.blogspot.com/2015/10/how-to-solve-concurrentmodificationexception-in-java-arraylist.html)**)**

When you remove object using Collection's or List's remove method e.g. remove(Object element) or remove(int index), instead of Iterator's remove() method than ConcurrentModificationException occurs. As per Iterator's contract, if it detect any structural change in Collection e.g. adding or removing of the element, once Iterator begins, it can throw ConcurrentModificationException.  Here are some tips to avoid ConcurrentModification in Java.

[](https://4.bp.blogspot.com/-XAU2_20M9Yg/VmQRaG8-IBI/AAAAAAAAEQc/G9dmlrMXdU0/s1600/ConcurrentModificationException%2Bwhile%2BIterating%2Bover%2BArrayList.png)

**21) Difference between Set, List and Map Collection classes? (**[**answer**](http://java67.blogspot.com/2013/01/difference-between-set-list-and-map-in-java.html)**)**

java.util.Set, java.util.List and java.util.Map defines three of most popular data structure support in Java. Set provides uniqueness guarantee i.e.g you can not store duplicate elements on it, but it's not ordered. On the other hand List is an ordered Collection and also allowes duplicates. Map is based on hashing and stores key and value in an Object called entry. It provides O(1) performance to get object, if you know keys, if there is no collision. Popular impelmentation of Set is HashSet, of List is ArrayList and LinkedList, and of Map are HashMap, Hashtable and ConcurrentHashMap. Another key difference between Set, List and Map are that Map doesn't implement Collection interface, while other two does. For a more detailed answer, see Set vs List vs Map in Java

[](https://3.bp.blogspot.com/-6H0yWpx3nEQ/VmQRFIW7GxI/AAAAAAAAEQM/cYFACp53rj0/s1600/List%2Bvs%2BMap%2Bvs%2BSet%2Bin%2BJava.jpg)

**22) What is BlockingQueue, how it is different than other collection classes? (**[**answer**](http://javarevisited.blogspot.com/2012/12/blocking-queue-in-java-example-ArrayBlockingQueue-LinkedBlockingQueue.html)**)**

BlockingQueue is a Queue implementation available in java.util.concurrent package. It's one of the concurrent Collection class added on Java 1.5, main difference between BlockingQueue and other collection classes is that apart from storage, it also provides flow control. It can be used in inter-thread communication and also provides built-in thread-safety by using happens-before guarantee. You can use BlockingQueue to solve Producer Consumer problem, which is what is needed in most of concurrent applications.

Read more: <https://javarevisited.blogspot.com/2011/11/collection-interview-questions-answers.html#ixzz6QeiHtSBi>