Collection Interview Questions

How can we make Hashmap synchronized?
HashMap can be synchronized by Map m = Collections.synchronizedMap(hashMap);

Question 13 : What are IdentityHashMap and WeakHashMap?

IdentityHashMap :
IdentityHashMap is similar to HashMap except that it uses reference equality when comparing elements.
IdentityHashMap class is not a widely used Map implementation.
While this class implements the Map interface, it intentionally violates Map's general contract, which mandates the use of the equals() method when comparing objects.
IdentityHashMap is designed for use only in the rare cases wherein reference-equality semantics are required.

WeakHashMap :
WeakHashMap is an implementation of the Map interface that stores only weak references to its keys.
Storing only weak references allows a key-value pair to be garbage collected when its key is no longer referenced outside of the WeakHashMap.
This class is intended primarily for use with key objects whose equals methods test for object identity using the == operator.
Once such a key is discarded it can never be recreated, so it is impossible to do a look-up of that key in a WeakHashMap at some later time and be surprised that its entry has been removed.

Question 14 : How to make a collection read only?
Answer : Use following methods:
Collections.unmodifiableList(list);
Collections.unmodifiableSet(set);
Collections.unmodifiableMap(map);
These methods takes collection parameter and return a new read-only collection with same elements as in original collection.

Question 15 : What is NavigableMap in Java ? What are its benefits over Map?
Answer :
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.

Question 16 : Which collection classes provide random access of it's elements?
Answer : ArrayList, HashMap, TreeMap, Hashtable classes provide random access to it's elements.

Question 17 : What is BlockingQueue?

Answer :
A Queue that additionally supports operations that wait for the queue to become non-empty when retrieving an element, and wait for space to become available in the queue when storing an element.
BlockingQueue methods come in four forms: one throws an exception, the second returns a special value (either null or false, depending on the operation), the third blocks the current thread indefinitely until the operation can succeed, and the fourth blocks for only a given maximum time limit before giving up.

Question 18 : When do you use ConcurrentHashMap in Java?
Answer :
This is another advanced level collection interview questions in Java which normally asked to check whether 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 write operation.
If you have equal number of reader and writer than ConcurrentHashMap will perform in line of Hashtable or synchronized HashMap.

Question 19 : Which implementation of the List interface provides for the fastest insertion of a new element into the middle of the list?
Answer :
We have three implementation of List : Vector, ArrayList, LinkedList.
ArrayList and Vector both use an array to store the elements of the list.
When an element is inserted into the middle of the list the elements that follow the insertion point must be shifted to make room for the new element.
The LinkedList is implemented using a doubly linked list; an insertion requires only the updating of the links at the point of insertion.
Therefore, the LinkedList allows for fast insertions and deletions.

Question 20 : What is the Difference between the Iterator and ListIterator?
Answer :
Iterator : Iterator Can Only get Data From forward Direction .
ListIterator : An iterator for lists that allows one to traverse the list in either direction.modify the list during iteration, and obtain the iterator's current position in the list.
A ListIterator has no current element. its cursor position always lies between the element that would be returned by a call to previous() and the element that would be returned by a call to next(). In a list of length n, there are n+1 valid index values, from 0 to n, inclusive.

Question 21 : What is CopyOnWriteArrayList, how it is different than ArrayList and Vector?
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 its different than Vector in terms of Concurrency.
CopyOnWriteArrayList provides better Concurrency by reducing contention among readers and writers.

Question 22 : Difference between Vector and ArrayList?
Answer :
Vector & ArrayList both classes are implemented using dynamically resizable arrays, providing fast random

access and fast traversal. ArrayList and Vector class both implement the List interface.
1) Synchronization - ArrayList is not thread-safe whereas Vector is thread-safe. In Vector class each method like add(), get(int i) is surrounded with a synchronized block and thus making Vector class thread-safe.
2) Data growth - Internally, both the ArrayList and Vector hold onto their contents using an Array. When an element is inserted into an ArrayList or a Vector, the object will need to expand its internal array if it runs out of room. A Vector defaults to doubling the size of its array, while the ArrayList increases its array size by 50 percent.

Question 23 : How Can I Synchronized ArrayList Object?
Answer : Yes, we can Synchronized Array list Object with the help of Collections.synchronizedList(<list object>);
List list = Collections.synchronizedList(new ArrayList());

Question 24 : Which two method you need to implement for key Object in HashMap ?
Answer : In order to use any object as Key in HashMap, it must implements equals and hashcode method in Java. Read How HashMap works in Java for detailed explanation on how equals and hashcode method is used to put and get object from HashMap.

Question 25 : What will be the problem if you don't override hashcode() method ?
Answer : You will not be able to recover your object from hash Map if that is used as key in HashMap.

Question 26 : What will happen if we put a key object in a HashMap which is already there ?
Answer : This tricky Java questions is part of How HashMap works in Java, which is also a popular topic to create confusing and tricky question in Java. well if you put the same key again than it will replace the old mapping because HashMap doesn't allow duplicate keys.

Question 27 : What is the difference between a HashMap and a Hashtable in Java?
Answer :
Hashtable is synchronized, whereas HashMap is not.
This makes HashMap better for non-threaded applications, as unsynchronized Objects typically perform better than synchronized ones.
Hashtable does not allow null keys or values.
HashMap allows one null key and any number of null values.
One of HashMap's subclasses is LinkedHashMap, so in the event that you'd want predictable iteration order (which is insertion order by default), you could easily swap out the HashMap for a LinkedHashMap. This wouldn't be as easy if you were using Hashtable.

Question 28 : What is difference between Iterator and Enumeration?
Answer :
Enumeration :
Enumeration doesn't have a remove() method.
Enumeration acts as Read-only interface, because it has the methods only to traverse and fetch the objects
So Enumeration is used whenever we want to make Collection objects as Read-only.
Iterator :
Iterator has a remove() method.
Can be abstract, final, native, static, or synchronized.
The main difference between the two is that Iterator is fail-safe. i.e, If you are using an iterator to go through a collection you can be sure of no concurrent modifications in the underlying collection which may happen in multi-threaded environments.

Question 29 : Why insertion and deletion in ArrayList is slow compared to LinkedList ?
Answer :
ArrayList internally uses and array to store the elements, when that array gets filled by inserting elements a new array of roughly 1.5 times the size of the original array is created and all the data of old array is copied to new array.
During deletion, all elements present in the array after the deleted elements have to be moved one step back to fill the space created by deletion. In linked list data is stored in nodes that have reference to the previous node and the next node so adding element is simple as creating the node an updating the next pointer on the last node and the previous pointer on the new node.
Deletion in linked list is fast because it involves only updating the next pointer in the node before the deleted node and updating the previous pointer in the node after the deleted node.

Question 30 : Why are Iterators returned by ArrayList called Fail Fast ?
Answer :
Because, if list is structurally modified at any time after the iterator is created, in any way except through the iterator's own remove or add methods, the iterator will throw a ConcurrentModificationException. Thus, in the face of concurrent modification, the iterator fails quickly and cleanly, rather than risking arbitrary, non-deterministic behavior at an undetermined time in the future.

Question 31 : How do you decide when to use ArrayList and When to use LinkedList?
Answer :
If you need to support random access, without inserting or removing elements from any place other than the end, then ArrayList offers the optimal collection.
If, however, you need to frequently add and remove elements from the middle of the list and only access the list elements sequentially, then LinkedList offers the better implementation.

Question 32 : What is the Properties class?
Answer : The properties class is a subclass of Hashtable that can be read from or written to a stream. It also provides the capability to specify a set of default values to be used.

Question 33 : How to convert a string array to arraylist?
Answer : new ArrayList(Arrays.asList(myArray));

Question 34 : How can you suppress unchecked warning in Java ?
Answer :
javac compiler for Java 5 generates unchecked warnings if you use combine raw types and generics types. You can be suppress those warnings by using @SuppressWarnings("unchecked") annotation.

Question 35 : What is difference between List<Object> and raw type List in Java?
Answer :
Main difference between raw type and parametrized type List<Object> is that :
Compiler will not check type-safety of raw type at compile time but it will do that for parametrized type and by using Object as Type it inform compiler that it can hold any Type of Object.
And second difference between them is that you can pass any parametrized type to raw type List but you can not pass List<String> to any method which accept List<Object> it will result in compilation error.