Collection:

1) What is the difference between ArrayList and Vector?

|  |  |  |
| --- | --- | --- |
| **No.** | **ArrayList** | **Vector** |
| 1) | ArrayList is not synchronized. | Vector is synchronized. |
| 2) | ArrayList is not a legacy class. | Vector is a legacy class. |
| 3) | ArrayList increases its size by 50% of the array size. | Vector increases its size by doubling the array size. |

2) What is the difference between ArrayList and LinkedList?

|  |  |  |
| --- | --- | --- |
| **No.** | **ArrayList** | **LinkedList** |
| 1) | ArrayList uses a dynamic array. | LinkedList uses doubly linked list. |
| 2) | ArrayList is not efficient for manipulation because a lot of shifting is required. | LinkedList is efficient for manipulation. |
| 3) | ArrayList is better to store and fetch data. | LinkedList is better to manipulate data. |

3) What is the difference between Iterator and ListIterator?

Iterator traverses the elements in forward direction only whereas ListIterator traverses the elements in forward and backward direction.

|  |  |  |
| --- | --- | --- |
| **No.** | **Iterator** | **ListIterator** |
| 1) | Iterator traverses the elements in forward direction only. | ListIterator traverses the elements in backward and forward directions both. |
| 2) | Iterator can be used in List, Set and Queue. | ListIterator can be used in List only. |

4) What is the difference between Iterator and Enumeration?

|  |  |  |
| --- | --- | --- |
| **No.** | **Iterator** | **Enumeration** |
| 1) | Iterator can traverse legacy and non-legacy elements. | Enumeration can traverse only legacy elements. |
| 2) | Iterator is fail-fast. | Enumeration is not fail-fast. |
| 3) | Iterator is slower than Enumeration. | Enumeration is faster than Iterator. |

5) What is the difference between List and Set?

List can contain duplicate elements whereas Set contains only unique elements.

6) What is the difference between HashSet and TreeSet?

HashSet maintains **no order** whereas TreeSet maintains **ascending order**.

7) What is the difference between Set and Map?

Set contains values only whereas Map contains key and values both.

8) What is the difference between HashSet and HashMap?

HashSet contains only values whereas HashMap contains entry(key,value). HashSet can be iterated but HashMap need to convert into Set to be iterated.

9) What is the difference between HashMap and TreeMap?

HashMap maintains **no order** but TreeMap maintains **ascending order**.

10) What is the difference between HashMap and Hashtable?

|  |  |  |
| --- | --- | --- |
| **No.** | **HashMap** | **Hashtable** |
| 1) | HashMap is not synchronized. | Hashtable is synchronized. |
| 2) | HashMap can contain one null key and multiple null values. | Hashtable cannot contain any null key or null value. |

11) What is the difference between Collection and Collections?

Collection is an interface whereas Collections is a class. Collection interface provides normal functionality of data structure to List, Set and Queue. But, Collections class is to sort and synchronize collection elements.

12) What is the difference between Comparable and Comparator?

|  |  |  |
| --- | --- | --- |
| **No.** | **Comparable** | **Comparator** |
| 1) | Comparable provides only one sort of sequence. | Comparator provides multiple sort of sequences. |
| 2) | It provides one method named compareTo(). | It provides one method named compare(). |
| 3) | It is found in java.lang package. | it is found in java.util package. |
| 4) | If we implement Comparable interface, actual class is modified. | Actual class is not modified. |

13) What is the advantage of Properties file?

If you change the value in properties file, you don't need to recompile the java class. So, it makes the application easy to manage.

14) What does the hashCode() method?

The hashCode() method returns a hash code value (an integer number).

The hashCode() method returns the same integer number, if two keys (by calling equals() method) are same.

But, it is possible that two hash code numbers can have different or same keys.

15) Why we override equals() method?

The equals method is used to check whether two objects are same or not. It needs to be overridden if we want to check the objects based on property.

For example, Employee is a class that has 3 data members: id, name and salary. But, we want to check the equality of employee object on the basis of salary. Then, we need to override the equals() method.

16) How to synchronize List, Set and Map elements?

Yes, Collections class provides methods to make List, Set or Map elements as synchronized:

|  |
| --- |
| public static List synchronizedList(List l){} |
| public static Set synchronizedSet(Set s){} |
| public static SortedSet synchronizedSortedSet(SortedSet s){} |
| public static Map synchronizedMap(Map m){} |
| public static SortedMap synchronizedSortedMap(SortedMap m){} |

17) What is the advantage of generic collection?

If we use generic class, we don't need typecasting. It is typesafe and checked at compile time.

18) What is hash-collision in Hashtable and how it is handled in Java?

Two different keys with the same hash value is known as hash-collision. Two different entries will be kept in a single hash bucket to avoid the collision.

19) What is the Dictionary class?

The Dictionary class provides the capability to store key-value pairs.

20) What is the default size of load factor in hashing based collection?

The default size of load factor is **0.75**. The default capacity is computed as initial capacity \* load factor. For example, 16 \* 0.75 = 12. So, 12 is the default capacity of Map.

General questions

[1) What is the Java Collections API? List down its advantages?](http://howtodoinjava.com/2013/07/09/useful-java-collection-interview-questions/#what_is_collection_in_java)

[2) Explain Collections hierarchy?](http://howtodoinjava.com/2013/07/09/useful-java-collection-interview-questions/#collections_hierarchy)

[3) Why Collection interface does not extend Cloneable and Serializable interface?](http://howtodoinjava.com/2013/07/09/useful-java-collection-interview-questions/#why_collection_not_extend_cloneable_serializable)

[4) Why Map interface does not extend Collection interface?](http://howtodoinjava.com/2013/07/09/useful-java-collection-interview-questions/#why_map_not_extend_collection)

List interface related

[5) Why we use List interface? What are main classes implementing List interface?](http://howtodoinjava.com/2013/07/09/useful-java-collection-interview-questions/#what_is_list_in_java)

[6) How to convert an array of String to ArrayList?](http://howtodoinjava.com/2013/07/09/useful-java-collection-interview-questions/#convert_array_of_String_to_ArrayList)

[7) How to reverse the list?](http://howtodoinjava.com/2013/07/09/useful-java-collection-interview-questions/#reverse_list)

Set interface related

[8) Why we use Set interface? What are main classes implementing Set interface?](http://howtodoinjava.com/2013/07/09/useful-java-collection-interview-questions/#what_is_set_in_java)

[9) How HashSet store elements?](http://howtodoinjava.com/2013/07/09/useful-java-collection-interview-questions/#how_set_store_elements)

[10) Can a null element added to a TreeSet or HashSet?](http://howtodoinjava.com/2013/07/09/useful-java-collection-interview-questions/#null_in_set)

Map interface related

[11) Why we use Map interface? What are main classes implementing Map interface?](http://howtodoinjava.com/2013/07/09/useful-java-collection-interview-questions/#what_is_map_in_java)

[12) What are IdentityHashMap and WeakHashMap?](http://howtodoinjava.com/2013/07/09/useful-java-collection-interview-questions/#identityHashMap_weakHashMap_differences)

[13) Explain ConcurrentHashMap? How it works?](http://howtodoinjava.com/2013/07/09/useful-java-collection-interview-questions/#how_concurrentHashMap_works)

[14) How hashmap works?](http://howtodoinjava.com/2013/07/09/useful-java-collection-interview-questions/#how_hashmap_works)

[15) How to design a good key for hashmap?](http://howtodoinjava.com/2013/07/09/useful-java-collection-interview-questions/#good_key_for_hashmap)

[16) What are different Collection views provided by Map interface?](http://howtodoinjava.com/2013/07/09/useful-java-collection-interview-questions/#different_collection_views)

[17) When to use HashMap or TreeMap?](http://howtodoinjava.com/2013/07/09/useful-java-collection-interview-questions/#hashmap_or_treemap)

Tell the difference questions

[18) Difference between Set and List?](http://howtodoinjava.com/2013/07/09/useful-java-collection-interview-questions/#difference_set_and_list)

[19) Difference between List and Map?](http://howtodoinjava.com/2013/07/09/useful-java-collection-interview-questions/#difference_map_and_list)

[20) Difference between HashMap and HashTable?](http://howtodoinjava.com/2013/07/09/useful-java-collection-interview-questions/#difference_hashmap_and_hashtable)

[21) Difference between Vector and ArrayList?](http://howtodoinjava.com/2013/07/09/useful-java-collection-interview-questions/#difference_vector_and_arraylist)

[22) Difference between Iterator and Enumeration?](http://howtodoinjava.com/2013/07/09/useful-java-collection-interview-questions/#difference_iterator_and_enumerator)

[23) Difference between HashMap and HashSet?](http://howtodoinjava.com/2013/07/09/useful-java-collection-interview-questions/#difference_hashmap_and_hashset)

[24) Difference between Iterator and ListIterator?](http://howtodoinjava.com/2013/07/09/useful-java-collection-interview-questions/#difference_iterator_and_listiterator)

[25) Difference between TreeSet and SortedSet?](http://howtodoinjava.com/2013/07/09/useful-java-collection-interview-questions/#difference_treeset_and_sortedset)

[26) Difference between ArrayList and LinkedList?](http://howtodoinjava.com/2013/07/09/useful-java-collection-interview-questions/#difference_arraylist_and_linkedlist)

More questions

[27) How to make a collection read only?](http://howtodoinjava.com/2013/07/09/useful-java-collection-interview-questions/#read_only_collection)

[28) How to make a collection thread safe?](http://howtodoinjava.com/2013/07/09/useful-java-collection-interview-questions/#thread_safe_collection)

[29) Why there is not method like Iterator.add() to add elements to the collection?](http://howtodoinjava.com/2013/07/09/useful-java-collection-interview-questions/#why_no_iterator_add)

[30) What are different ways to iterate over a list?](http://howtodoinjava.com/2013/07/09/useful-java-collection-interview-questions/#different_ways_to_iterate_list)

[31) What do you understand by iterator fail-fast property?](http://howtodoinjava.com/2013/07/09/useful-java-collection-interview-questions/#fail_fast_iterator)

[32) What is difference between fail-fast and fail-safe?](http://howtodoinjava.com/2013/07/09/useful-java-collection-interview-questions/#fail_safe_vs_fail_fast)

[33) How to avoid ConcurrentModificationException while iterating a collection?](http://howtodoinjava.com/2013/07/09/useful-java-collection-interview-questions/#avoid_ConcurrentModificationException)

[34) What is UnsupportedOperationException?](http://howtodoinjava.com/2013/07/09/useful-java-collection-interview-questions/#UnsupportedOperationException)

[35) Which collection classes provide random access of it’s elements?](http://howtodoinjava.com/2013/07/09/useful-java-collection-interview-questions/#random_access_collections)

[36) What is BlockingQueue?](http://howtodoinjava.com/2013/07/09/useful-java-collection-interview-questions/#BlockingQueue)

[37) What is Queue and Stack, list their differences?](http://howtodoinjava.com/2013/07/09/useful-java-collection-interview-questions/#queue_and_stack_difference)

[38) What is Comparable and Comparator interface?](http://howtodoinjava.com/2013/07/09/useful-java-collection-interview-questions/#Comparable_and_Comparator)

[39) What are Collections and Arrays class?](http://howtodoinjava.com/2013/07/09/useful-java-collection-interview-questions/#Collections_and_arrays)

[40) Recommended resources](http://howtodoinjava.com/2013/07/09/useful-java-collection-interview-questions/#resources)

Without wasting time, let dig into concepts.

### ****General question****

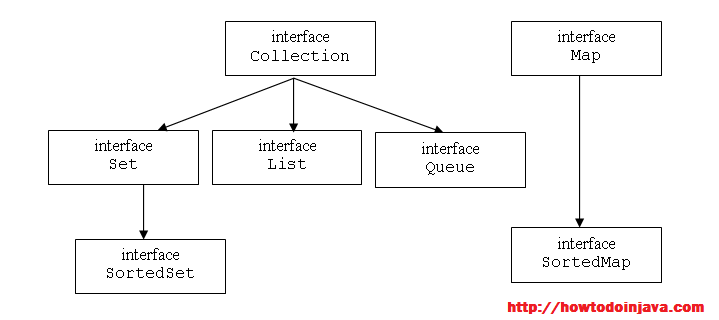
##### **1) What is the Java Collection framework? List down its advantages?**

By definition, a collection is **an object that represents a group of objects**. Like in set theory, a set is group of elements. Easy enough !!  
Prior to JDK 1.2, JDK has some utility classes such as Vector and HashTable, but there was no concept of Collection framework. Later from JDK 1.2 onwards, JDK felt the need of having a consistent support for reusable data structures. Finally, the collections framework was designed and developed primarily by Joshua Bloch, and was **introduced in JDK 1.2**.

Its most **noticeable advantages** can be listed as:

* Reduced programming effort due to ready to use code
* Increased performance because of high-performance implementations of data structures and algorithms
* Provides interoperability between unrelated APIs by establishing a common language to pass collections back and forth
* Easy to learn APIs by learning only some top level interfaces and supported operations

##### **2) Explain Collection’s hierarchy?**



*Java Collection Hierarchy*

As shown in above image, collection framework has one interface at top i.e. **Collection**. It is **extended by Set, List and Queue interfaces**. Then there are loads of other classes in these 3 branches which we will learn in following questions.

Remember the signature of Collection interface. It will help you in many question.

public interface Collection extends Iterable {

//method definitions

}

Framework also consist of Map interface, which is part of collection framework. but it does not extend Collection interface. We will see the reason in 4th question in this question bank.

##### **3) Why Collection interface does not extend Cloneable and Serializable interface?**

Well, simplest answer is “**there is no need to do it**“. Extending an interface simply means that you are creating a subtype of interface, in other words a more specialized behavior and Collection interface is not expected to do what Cloneable and Serializable interfaces do.

Another reason is that not everybody will have a reason to have Cloneable collection because if it has very large data, then every **unnecessary clone operation will consume a big memory**. Beginners might use it without knowing the consequences.

Another reason is that **Cloneable and Serializable are very specialized behavior** and so should be implemented only when required. For example, many concrete classes in collection implement these interfaces. So if you want this feature. use these collection classes otherwise use their alternative classes.

##### **4) Why Map interface does not extend Collection interface?**

A good answer to this interview question is “**because they are incompatible**“. Collection has a method add(Object o). Map can not have such method because it need key-value pair. There are other reasons also such as Map supports keySet, valueSet etc. Collection classes does not have such views.

Due to such big differences, Collection interface was not used in Map interface, and it was build in separate hierarchy.

### ****List interface related****

##### **5) Why we use List interface? What are main classes implementing List interface?**

A java list is a **“ordered” collection of elements**. This ordering is a **zero based index**. It does not care about duplicates. Apart from methods defined in Collection interface, it does **have its own methods** also which are largely to manipulate the collection **based on index location of element**. These methods can be grouped as search, get, iteration and range view. All above operations support index locations.

The main classes implementing List interface are: **Stack, Vector, ArrayList and LinkedList**. Read more about them in java documentation.

##### **6) How to convert an array of String to arraylist?**

This is more of a programmatic question which is seen at beginner level. The intent is to check the knowledge of applicant in Collection utility classes. For now, lets learn that there are two utility classes in Collection framework which are mostly seen in interviews i.e. **Collections and Arrays**.

Collections class provides some static functions to perform specific operations on collection types. And Arrays provide utility functions to be performed on array types.

//String array

String[] words = {&quot;ace&quot;, &quot;boom&quot;, &quot;crew&quot;, &quot;dog&quot;, &quot;eon&quot;};

//Use Arrays utility class

List wordList = Arrays.asList(words);

//Now you can iterate over the list

Please not that this function is not specific to String class, it will return List of element of any type, of which the array is. e.g.

//String array

Integer[] nums = {1,2,3,4};

//Use Arrays utility class

List numsList = Arrays.asList(nums);

##### **7) How to reverse the list?**

This question is just like above to test your knowledge of **Collections** utility class. Use it **reverse()** method to reverse the list.

Collections.reverse(list);

### ****Set interface related****

##### **8) Why we use Set interface? What are main classes implementing Set interface?**

It **models the mathematical set in set theory**. Set interface is like List interface but with some differences. First, it is **not ordered collection**. So no ordering is preserved while adding or removing elements. The main feature it does provide is “**uniqueness of elements**“. It does not support duplicate elements.

Set also adds a stronger contract on the behavior of the equals and hashCode operations, allowing Set instances to be compared meaningfully even if their implementation types differ. Two Set instances are equal if they contain the same elements.

Based on above reasons, it **does not have operations based on indexes of elements like List**. It only has methods which are inherited by Collection interface.

Main classes implementing Set interface are :**EnumSet, HashSet, LinkedHashSet, TreeSet**. Read more on related java documentation.

##### **9) How HashSet store elements?**

You must know that HashMap store key-value pairs, with one condition i.e. keys will be unique. HashSet uses Map’s this feature to ensure uniqueness of elements. In HashSet class, a map declaration is as below:

private transient HashMap<E,Object> map;

//This is added as value for each key

private static final Object PRESENT = new Object();

So **when you store a element in HashSet, it stores the element as key in map and “PRESENT” object as value**. (See declaration above).

public boolean add(E e) {

return map.put(e, PRESENT)==null;

}

I will highly suggest you to read this post: [**How HashMap works in java?**](http://howtodoinjava.com/2012/10/09/how-hashmap-works-in-java/) This post will help you in answering all the HashMap related questions very easily.

##### **10) Can a null element added to a TreeSet or HashSet?**

As you see, There is no null check in add() method in previous question. And HashMap also allows one null key, so **one “null” is allowed in HashSet**.

TreeSet uses the same concept as HashSet for internal logic, but uses NavigableMap for storing the elements.

private transient NavigableMap<E,Object> m;

// Dummy value to associate with an Object in the backing Map

private static final Object PRESENT = new Object();

NavigableMap is subtype of SortedMap which does not allow null keys. So essentially,**TreeSet also does not support null keys**. It will throw NullPointerException if you try to add null element in TreeSet.

### ****Map interface related****

##### **11) Why we use Map interface? What are main classes implementing Map interface?**

Map interface is a special type of collection which is **used to store key-value pairs**. It does not extend Collection interface for this reason. This interface provides methods to add, remove, search or iterate over various views of Map.

Main classes implementing Map interface are:**HashMap, Hashtable, EnumMap, IdentityHashMap, LinkedHashMap and Properties.**

##### **12) What are IdentityHashMap and WeakHashMap?**

**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** 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.

##### **13) Explain ConcurrentHashMap? How it works?**

Taking from java docs:

**A hash table supporting full concurrency of retrievals and adjustable expected concurrency for updates**. This class obeys the same functional specification as Hashtable, and includes versions of methods corresponding to each method of Hashtable. However, even though all operations are thread-safe, retrieval operations do not entail locking, and there is not any support for locking the entire table in a way that prevents all access. This class is fully interoperable with Hashtable in programs that rely on its thread safety but not on its synchronization details.

Read more about how [**concurrent hashmap works and related interview questions**](http://howtodoinjava.com/2013/06/14/popular-hashmap-and-concurrenthashmap-interview-questions/).

##### **14) How hashmap works?**

The **most important question** which is most likely to be seen in every level of job interviews. You must be very clear on this topic., not only because it is most asked question but also it will open up your mind in further questions related to collection APIs.

Answer to this question is very large and you should read it my post: [**How HashMap works?**](http://howtodoinjava.com/2012/10/09/how-hashmap-works-in-java/) For now, lets remember that HashMap works **on principle of Hashing**. A map by definition is : “An object that maps keys to values”. To store such structure, **it uses an inner class Entry**:

static class Entry implements Map.Entry

{

final K key;

V value;

Entry next;

final int hash;

...//More code goes here

}

Here key and value variables are used to store key-value pairs. Whole entry object is stored in an array.

/\*\*

\* The table, re-sized as necessary. Length MUST Always be a power of two.

\*/

transient Entry[] table;

The index of array is calculated on basis on hashcode of Key object. Read more of linked topic.

##### **15) How to design a good key for hashmap?**

Another good question usually followed up after answering how hashmap works. Well, the most important constraint is **you must be able to fetch the value object back in future**. Otherwise, there is no use of having such a data structure. If you understand the working of hashmap, you will find it largely depends on hashCode() and equals() method of Key objects.

So a good key object**must provide same hashCode() again and again**, no matter how many times it is fetched. Similarly, same keys**must return true when compare with equals() method and different keys must return false**.

For this reason,**immutable classes are considered best candidate for HashMap keys**.

Read more : [**How to design a good key for HashMap?**](http://howtodoinjava.com/2013/05/02/how-to-design-a-good-key-for-hashmap/)

##### **16) What are different Collection views provided by Map interface?**

Map interface provides 3 views of key-values pairs stored in it:

* key set view
* value set view
* entry set view

All the views can be navigated using iterators.

##### **17) When to use HashMap or TreeMap?**

HashMap is well known class and all of us know that. So, I will leave this part by saying that it is used to store key-value pairs and allows to perform many operations on such collection of pairs.

TreeMap is special form of HashMap. **It maintains the ordering of keys** which is missing in HashMap class. This ordering is**by default “natural ordering”**. The default ordering can be override by providing an instance of Comparator class, whose compare method will be used to maintain ordering of keys.

Please note that **all keys inserted into the map must implement the Comparable interface** (this is necessary to decide the ordering). Furthermore, all such keys must be mutually comparable: k1.compareTo(k2) must not throw a ClassCastException for any keys k1 and k2 in the map. If the user attempts to put a key into the map that violates this constraint (for example, the user attempts to put a string key into a map whose keys are integers), the put(Object key, Object value) call will throw a ClassCastException.

### ****Tell the difference questions****

##### **18) Difference between Set and List?**

The most noticeable differences are :

* Set is unordered collection where List is ordered collection based on zero based index.
* List allow duplicate elements but Set does not allow duplicates.
* List does not prevent inserting null elements (as many you like), but Set will allow only one null element.

##### **19) Difference between List and Map?**

Perhaps most easy question. **List is collection of elements where as map is collection of key-value pairs**. There is actually lots of differences which originate from first statement. They have**separate top level interface, separate set of generic methods, different supported methods and different views of collection**.

I will take much time hear as answer to this question is enough as first difference only.

##### **20) Difference between HashMap and HashTable?**

There are several differences between HashMap and Hashtable in Java:

* Hashtable is synchronized, whereas HashMap is not.
* Hashtable does not allow null keys or values. HashMap allows one null key and any number of null values.
* The third significant difference between HashMap vs Hashtable is that Iterator in the HashMap is a fail-fast iterator while the enumerator for the Hashtable is not.

##### **21) Difference between Vector and ArrayList?**

Lets note down the differences:

* All the methods of Vector is synchronized. But, the methods of ArrayList is not synchronized.
* Vector is a Legacy class added in first release of JDK. ArrayList was part of JDK 1.2, when collection framework was introduced in java.
* By default, Vector doubles the size of its array when it is re-sized internally. But, ArrayList increases by half of its size when it is re-sized.

##### **22) Difference between Iterator and Enumeration?**

Iterators differ from enumerations in three ways:

* Iterators allow the caller to remove elements from the underlying collection during the iteration with its remove() method. You can not add/remove elements from a collection when using enumerator.
* Enumeration is available in legacy classes i.e Vector/Stack etc. whereas Iterator is available in all modern collection classes.
* Another minor difference is that Iterator has improved method names e.g. Enumeration.hasMoreElement() has become Iterator.hasNext(), Enumeration.nextElement() has become Iterator.next() etc.

##### **23) Difference between HashMap and HashSet?**

HashMap is collection of key-value pairs whereas HashSet is un-ordered collection of unique elements. That’s it. No need to describe further.

##### **24) Difference between Iterator and ListIterator?**

There are three Differences are there:

* We can use Iterator to traverse Set and List and also Map type of Objects. But List Iterator can be used to traverse for List type Objects, but not for Set type of Objects.
* By using Iterator we can retrieve the elements from Collection Object in forward direction only whereas List Iterator, which allows you to traverse in either directions using hasPrevious() and previous() methods.
* ListIterator allows you modify the list using add() remove() methods. Using Iterator you can not add, only remove the elements.

##### **25) Difference between TreeSet and SortedSet?**

SortedSet is an interface which TreeSet implements. That’ it !!

##### **26) Difference between ArrayList and LinkedList?**

* LinkedList store elements within a doubly-linked list data structure. ArrayList store elements within a dynamically resizing array.
* LinkedList allows for constant-time insertions or removals, but only sequential access of elements. In other words, you can walk the list forwards or backwards, but grabbing an element in the middle takes time proportional to the size of the list. ArrayLists, on the other hand, allow random access, so you can grab any element in constant time. But adding or removing from anywhere but the end requires shifting all the latter elements over, either to make an opening or fill the gap.
* LinkedList has more memory overhead than ArrayList because in ArrayList each index only holds actual object (data) but in case of LinkedList each node holds both data and address of next and previous node.

### ****More questions****

##### **27) How to make a collection read only?**

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.

##### **28) How to make a collection thread safe?**

Use below methods:

* Collections.synchronizedList(list);
* Collections.synchronizedSet(set);
* Collections.synchronizedMap(map);

Above methods take collection as parameter and return same type of collection which are synchronized and thread safe.

##### **29) Why there is not method like Iterator.add() to add elements to the collection?**

The sole purpose of an Iterator is to enumerate through a collection. All collections contain the add() method to serve your purpose. There would be no point in adding to an Iterator because the **collection may or may not be ordered**. And **add() method can not have same implementation for ordered and unordered collections**.

##### **30) What are different ways to iterate over a list?**

You can iterate over a list using following ways:

* Iterator loop
* For loop
* For loop (Advance)
* While loop

Read more : <http://www.mkyong.com/java/how-do-loop-iterate-a-list-in-java/>

##### **31) What do you understand by iterator fail-fast property?**

**Fail-fast Iterators fail as soon as they realized that structure of Collection has been changed since iteration has begun**. Structural changes means adding, removing or updating any element from collection while one thread is Iterating over that collection.

Fail-fast behavior is implemented by keeping a modification count and if iteration thread realizes the change in modification count it throws ConcurrentModificationException.

##### **32) What is difference between fail-fast and fail-safe?**

You have understood fail-fast in previous question. **Fail-safe iterators** are just opposite to fail-fast. **They never fail if you modify the underlying collection on which they are iterating**, because they work on clone of Collection instead of original collection and that’s why they are called as fail-safe iterator.

Iterator of CopyOnWriteArrayList is an example of fail-safe Iterator also iterator written by ConcurrentHashMap keySet is also fail-safe iterator and never throw ConcurrentModificationException.

##### **33) How to avoid ConcurrentModificationException while iterating a collection?**

You should first try to **find another alternative iterator which are fail-safe**. For example if you are using List and you can use ListIterator. If it is legacy collection, you can use enumeration.

If above options are not possible then you can use one of three changes:

* If you are using JDK1.5 or higher then you can use ConcurrentHashMap and CopyOnWriteArrayList classes. It is the recommended approach.
* You can convert the list to an array and then iterate on the array.
* You can lock the list while iterating by putting it in a synchronized block.

Please note that last two approaches will cause a performance hit.

##### **34) What is UnsupportedOperationException?**

This exception is thrown **on invoked methods which are not supported by actual collection type**.

For example, if you make a read-only list list using “Collections.unmodifiableList(list)” and then call add() or remove() method, what should happen. It should clearly throw UnsupportedOperationException.

##### **35) Which collection classes provide random access of it’s elements?**

ArrayList, HashMap, TreeMap, Hashtable classes provide random access to it’s elements.

##### **36) What is BlockingQueue?**

**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.

Read the example usage of blocking queue in post : [**How to use blocking queue?**](http://howtodoinjava.com/2012/10/20/how-to-use-blockingqueue-and-threadpoolexecutor-in-java/)

##### **37) What is Queue and Stack, list down their differences?**

**A collection designed for holding elements prior to processing.** Besides basic Collection operations, queues provide additional insertion, extraction, and inspection operations.  
**Queues typically, but do not necessarily, order elements in a FIFO (first-in-first-out) manner.**

**Stack is also a form of Queue but one difference, it is LIFO (last-in-first-out).**

Whatever the ordering used, the head of the queue is that element which would be removed by a call to remove() or poll(). Also note that Stack and Vector are both synchronized.

**Usage:** Use a queue if you want to process a stream of incoming items in the order that they are received.Good for work lists and handling requests.  
Use a stack if you want to push and pop from the top of the stack only. Good for recursive algorithms.

##### **38) What is Comparable and Comparator interface?**

In java. all collection which have feature of automatic sorting, uses compare methods to ensure the correct sorting of elements. For example classes which use sorting are TreeSet, TreeMap etc.

**To sort the data elements a class needs to implement Comparator or Comparable interface**. That’s why all Wrapper classes like Integer,Double and String class implements Comparable interface.

**Comparable helps in preserving default natural sorting, whereas Comparator helps in sorting the elements in some special required sorting pattern.** The instance of comparator if passed usually as collection’s constructor argument in supporting collections.

##### **39) What are Collections and Arrays classes?**

**Collections and Arrays classes are special utility classes to support collection framework core classes.** They provide utility functions to get read-only/ synchronized collections, sort the collection on various ways etc.

Arrays also helps array of objects to convert in collection objects. Arrays also have some functions which helps in copying or working in part of array objects.

##### **40) Recommended resources**

Well it is not interview question.. :-). This is only for fun. But you should really read my blog for more posts on collection framework knowledge.

I hope these java collection interview questions will help in in your next interview. Further, I will suggest you to read more on above questions apart from this post. A more knowledge will only help you.

Real Time Interview question:

IGT (Inter Globe Technologies)

Integer in = **null**;

**int** i = in;

System.*out*.println(i);

Output:

Exception in thread "main" java.lang.NullPointerException

at com.test.Test13.main(Test13.java:7)

1. How to make a class immutable?

Answer:

1. Don't provide "setter" methods — methods that modify fields or objects referred to by fields.
2. Make all fields final and private.
3. Don't allow subclasses to override methods. The simplest way to do this is to declare the class as final. A more sophisticated approach is to make the constructor private and construct instances in factory methods.
4. If the instance fields include references to mutable objects, don't allow those objects to be changed:
   * Don't provide methods that modify the mutable objects.
   * Don't share references to the mutable objects. Never store references to external, mutable objects passed to the constructor; if necessary, create copies, and store references to the copies. Similarly, create copies of your internal mutable objects when necessary to avoid returning the originals in your methods.
5. Write down the singleton class?
6. In a given sentence how do you find out the words which is duplicate and also find the count of duplicity.
7. Example of Spring AOP.
8. Difference between get and load.
9. Version of technology
10. Difference between Collection and Collections.
11. Two arrays are there how to put in third array all the data in the sorted order.
12. How do you do the installation of the servers?
13. Which Aop you used in you program Spring AOP or Aspect AOP?
14. How did you use multithreading in your current project?
15. How did you call the stored procedure in the hibernate?
16. What are the benefit of HQL over sql native language?
17. What is the criteria in the hibernate?
18. Major components of the wsdl file?
19. Components of web Services?
20. What is graph data type? It’s not the part of Java technology

Accenture:

1. Spring MVC, how to use controller into it?
2. In Spring MVC, how to call two controller method from the client?
3. In spring MVC, how to configure the controller?
4. In Spring MVC, how to call DAO method from controller?
5. What is Spring IOC?
6. Can try and finally exists?
7. If there is a return in try block then what happen with catch and finally?
8. If there is a “exit()” in try block then what happen with catch and finally?
9. What is the difference between forward and sendRedirect in jsp?

Sapient:

1. Which framework have you worked?
2. Which module in Spring frame you worked?
3. How did you use AOP in you project?
4. How did you use IOC concept without using Spring framework?
5. What is the significant of Factory design pattern?
6. How Factory design pattern works?
7. If a singleton class is serialized and persist into database and then again read from database then there will be two objects of the singleton class, how could you prevent this?
8. How did you handle the exception after any validation fails?
9. How did you handle the exception in the web application?
10. Where implicit objects are defined in the JSP page or how did you declare implicit page in the JSP page?
11. How did you add new tags in the JSP page, the exact syntax ?

Round 2:

1. Tell me about yourself.
2. Tell about your roles in the project.
3. About SRS and SDS
4. About class diagram?
5. Can try exist without catch?
6. **public** **static** **void** main(String[] args) **throws** Exception {

List l = **new** ArrayList();

**try** {

System.*out*.println(l);

**throw** **new** Exception();

} **catch** (Exception e) {

// throw new Exception();

}

}

Will it compile or not?

1. What is difference between NoClassDefFoundError and ClassNotFoundException?
2. What is enumeration?
3. What is difference between enumeration and iteration?
4. What is ConcurrentModificatoinException? How can prevent it?
5. How HashMap works internally?
6. What is hibernate?
7. Is Session thread safe or not?
8. What the difference is between save and persist?
9. What the difference is between load and get?
10. What is qualifier in spring?
11. When should we use String and StringBuilder?
12. How many types of injections are there in the spring?
13. What is method injection?
14. How to inject null value in Spring?
15. How to divide the task to team members as a team lead?

Software AG:

1. How to make a class immutable?
2. If field will not be final in the immutable class it’ll break in reflection.
3. What is the best way to write the Singleton class?
4. How to achieve singleton in case of serialization?
5. How to execute two threads one after another using wait and notify
6. How to sort an array which contains only 0 and 1 numbers?
7. How to override remove(int [] indexes) method to remove all the data with given indexes?
8. Reverse the data of LinkedList?