

[JAVA TUTORIAL](#)[#INDEX POSTS](#)[#INTERVIEW QUESTIONS](#)[RESOURCES](#)[HIRE ME](#)[DOWNLOAD ANDROID APP](#)[CONTRIBUTE](#)

SIGN UP TO RECIEVE OUR UPDATES VIA EMAIL

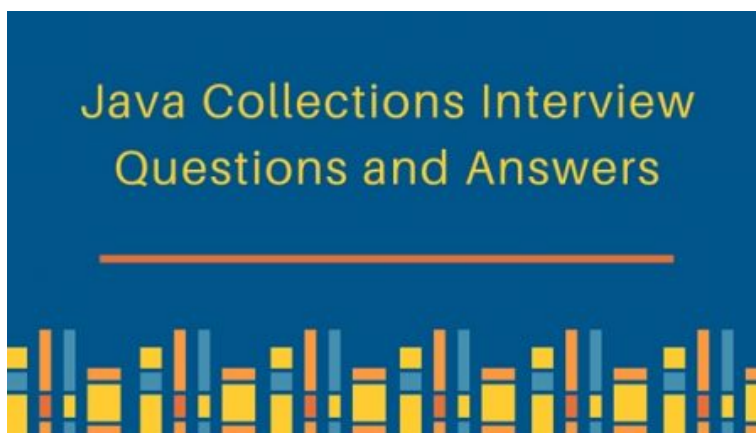
SUBSCRIBE

[HOME](#) » [INTERVIEW QUESTIONS](#) » 40 JAVA COLLECTIONS INTERVIEW QUESTIONS AND ANSWERS

# 40 Java Collections Interview Questions and Answers

APRIL 3, 2018 BY [PANKAJ](#) — [135 COMMENTS](#)

[Java Collections Framework](#) is the fundamental aspect of java programming language. It's one of the important topic for [java interview questions](#). Here I am listing some important java collections interview questions and answers for helping you in interview. This is directly coming from my 12+ year of experience in java programming.



## Java Collections Interview Questions

1. [What are Collection related features in Java 8?](#)

2. What is Java Collections Framework? List out some benefits of Collections framework?
3. What is the benefit of Generics in Collections Framework?
4. What are the basic interfaces of Java Collections Framework?
5. Why Collection doesn't extend Cloneable and Serializable interfaces?
6. Why Map interface doesn't extend Collection interface?
7. What is an Iterator?
8. What is difference between Enumeration and Iterator interface?
9. Why there is not method like Iterator.add() to add elements to the collection?
10. Why Iterator don't have a method to get next element directly without moving the cursor?
11. What is different between Iterator and ListIterator?
12. What are different ways to iterate over a list?
13. What do you understand by iterator fail-fast property?
14. What is difference between fail-fast and fail-safe?
15. How to avoid ConcurrentModificationException while iterating a collection?
16. Why there are no concrete implementations of Iterator interface?
17. What is UnsupportedOperationException?
18. How HashMap works in Java?
19. What is the importance of hashCode() and equals() methods?
20. Can we use any class as Map key?
21. What are different Collection views provided by Map interface?
22. What is difference between HashMap and Hashtable?
23. How to decide between HashMap and TreeMap?
24. What are similarities and difference between ArrayList and Vector?
25. What is difference between Array and ArrayList? When will you use Array over ArrayList?
26. What is difference between ArrayList and LinkedList?
27. Which collection classes provide random access of it's elements?
28. What is EnumSet?
29. Which collection classes are thread-safe?
30. What are concurrent Collection Classes?
31. What is BlockingQueue?
32. What is Queue and Stack, list their differences?
33. What is Collections Class?
34. What is Comparable and Comparator interface?
35. What is difference between Comparable and Comparator interface?
36. How can we sort a list of Objects?
37. While passing a Collection as argument to a function, how can we make sure the function will not be able to modify it?
38. How can we create a synchronized collection from given collection?
39. What are common algorithms implemented in Collections Framework?
40. What is Big-O notation? Give some examples?
41. What are best practices related to Java Collections Framework?
42. What is Java Priority Queue?
43. Why can't we write code as `List<Number> numbers = new ArrayList<Integer>();`?

44. Why can't we create generic array? or write code as `List<Integer>[] array = new ArrayList<Integer>[10];`

# Java Collections Interview Questions and Answers

## 1. What are Collection related features in Java 8?

Java 8 has brought major changes in the Collection API. Some of the changes are:

1. **Java Stream API** for collection classes for supporting sequential as well as parallel processing
2. **Iterable interface is extended with `forEach()`** default method that we can use to iterate over a collection. It is very helpful when used with **lambda expressions** because its argument **Consumer** is a **function interface**.
3. Miscellaneous Collection API improvements such as `forEachRemaining(Consumer action)` method in `Iterator` interface, `Map.replaceAll()`, `compute()`, `merge()` methods.

## 2. What is Java Collections Framework? List out some benefits of Collections framework?

Collections are used in every programming language and initial java release contained few classes for collections: **Vector**, **Stack**, **Hashtable**, **Array**. But looking at the larger scope and usage, Java 1.2 came up with Collections Framework that group all the collections interfaces, implementations and algorithms.

Java Collections have come through a long way with usage of Generics and Concurrent Collection classes for thread-safe operations. It also includes blocking interfaces and their implementations in java concurrent package.

Some of the benefits of collections framework are;

- Reduced development effort by using core collection classes rather than implementing our own collection classes.
- Code quality is enhanced with the use of well tested collections framework classes.
- Reduced effort for code maintenance by using collection classes shipped with JDK.
- Reusability and Interoperability

## 3. What is the benefit of Generics in Collections Framework?

Java 1.5 came with Generics and all collection interfaces and implementations use it heavily. Generics allow us to provide the type of Object that a collection can contain, so if you try to add any element of other type it throws compile time error.

This avoids `ClassCastException` at Runtime because you will get the error at compilation. Also Generics make code clean since we don't need to use casting and *instanceof* operator. I would highly recommend to go through **Java Generic Tutorial** to understand generics in a better way.

## 4. What are the basic interfaces of Java Collections Framework?

**Collection** is the root of the collection hierarchy. A collection represents a group of objects known as its elements. The Java platform doesn't provide any direct implementations of this interface.

**Set** is a collection that cannot contain duplicate elements. This interface models the mathematical set abstraction and is used to represent sets, such as the deck of cards.

**List** is an ordered collection and can contain duplicate elements. You can access any element from its index. List is more like array with dynamic length.

A **Map** is an object that maps keys to values. A map cannot contain duplicate keys: Each key can map to at most one value.

Some other interfaces are **Queue**, **Deque**, **Iterator**, **SortedSet**, **SortedMap** and **ListIterator**.

## 5. Why Collection doesn't extend Cloneable and Serializable interfaces?

Collection interface specifies group of Objects known as elements. How the elements are maintained is left up to the concrete implementations of Collection. For example, some Collection implementations like List allow duplicate elements whereas other implementations like Set don't. A lot of the Collection implementations have a public clone method. However, it doesn't really make sense to include it in all implementations of Collection. This is because Collection is an abstract representation. What matters is the implementation.

The semantics and the implications of either cloning or serializing come into play when dealing with the actual implementation; so concrete implementation should decide how it should be cloned or serialized, or even if it can be cloned or serialized.

So mandating cloning and serialization in all implementations is actually less flexible and more restrictive. The specific implementation should make the decision as to whether it can be cloned or serialized.

## 6. Why Map interface doesn't extend Collection interface?

Although Map interface and its implementations are part of Collections Framework, Map are not collections and collections are not Map. Hence it doesn't make sense for Map to extend Collection or vice versa.

If Map extends Collection interface, then where are the elements? Map contains key-value pairs and it provides methods to retrieve list of Keys or values as Collection but it doesn't fit into the "group of elements" paradigm.

## 7. What is an Iterator?

Iterator interface provides methods to iterate over any Collection. We can get iterator instance from a Collection using *iterator()* method. Iterator takes the place of Enumeration in the **Java Collections Framework**. Iterators allow the caller to remove elements from the underlying collection during the

iteration. Java Collection iterator provides a generic way for traversal through the elements of a collection and implements **Iterator Design Pattern**.

## 8. What is difference between Enumeration and Iterator interface?

Enumeration is twice as fast as Iterator and uses very less memory. Enumeration is very basic and fits to basic needs. But Iterator is much safer as compared to Enumeration because it always denies other threads to modify the collection object which is being iterated by it.

Iterator takes the place of Enumeration in the Java Collections Framework. Iterators allow the caller to remove elements from the underlying collection that is not possible with Enumeration. Iterator method names have been improved to make it's functionality clear.

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

The semantics are unclear, given that the contract for Iterator makes no guarantees about the order of iteration. Note, however, that ListIterator does provide an add operation, as it does guarantee the order of the iteration.

## 10. Why Iterator don't have a method to get next element directly without moving the cursor?

It can be implemented on top of current Iterator interface but since it's use will be rare, it doesn't make sense to include it in the interface that everyone has to implement.

## 11. What is different between Iterator and ListIterator?

- We can use Iterator to traverse Set and List collections whereas ListIterator can be used with Lists only.
- Iterator can traverse in forward direction only whereas ListIterator can be used to traverse in both the directions.
- ListIterator inherits from Iterator interface and comes with extra functionalities like adding an element, replacing an element, getting index position for previous and next elements.

## 12. What are different ways to iterate over a list?

We can iterate over a list in two different ways – using iterator and using for-each loop.

```
List<String> strList = new ArrayList<>();

//using for-each loop
for(String obj : strList){
    System.out.println(obj);
}
```

```
//using iterator
Iterator<String> it = strList.iterator();
while(it.hasNext()){
    String obj = it.next();
    System.out.println(obj);
}
```

Using iterator is more thread-safe because it makes sure that if underlying list elements are modified, it will throw `ConcurrentModificationException`.

### 13. What do you understand by iterator fail-fast property?

Iterator fail-fast property checks for any modification in the structure of the underlying collection everytime we try to get the next element. If there are any modifications found, it throws `ConcurrentModificationException`. All the implementations of Iterator in Collection classes are fail-fast by design except the concurrent collection classes like `ConcurrentHashMap` and `CopyOnWriteArrayList`.

### 14. What is difference between fail-fast and fail-safe?

Iterator fail-safe property work with the clone of underlying collection, hence it's not affected by any modification in the collection. By design, all the collection classes in `java.util` package are fail-fast whereas collection classes in `java.util.concurrent` are fail-safe.

Fail-fast iterators throw `ConcurrentModificationException` whereas fail-safe iterator never throws `ConcurrentModificationException`.

Check this post for [CopyOnWriteArrayList Example](#).

### 15. How to avoid ConcurrentModificationException while iterating a collection?

We can use concurrent collection classes to avoid `ConcurrentModificationException` while iterating over a collection, for example `CopyOnWriteArrayList` instead of `ArrayList`.

Check this post for [ConcurrentHashMap Example](#).

### 16. Why there are no concrete implementations of Iterator interface?

Iterator interface declare methods for iterating a collection but it's implementation is responsibility of the Collection implementation classes. Every collection class that returns an iterator for traversing has it's own Iterator implementation nested class.

This allows collection classes to chose whether iterator is fail-fast or fail-safe. For example `ArrayList` iterator is fail-fast whereas `CopyOnWriteArrayList` iterator is fail-safe.

### 17. What is UnsupportedOperationException?

`UnsupportedOperationException` is the exception used to indicate that the operation is not supported. It's used extensively in **JDK** classes, in collections framework

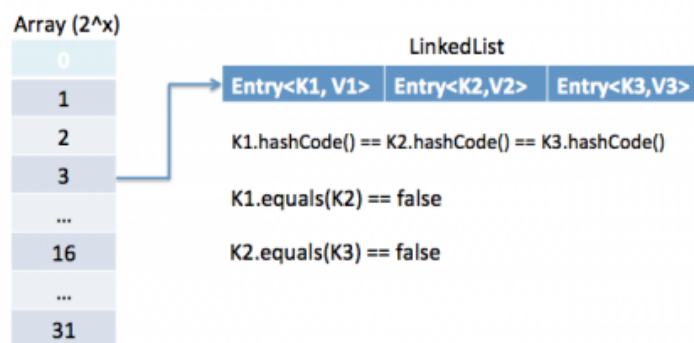
`java.util.Collections.UnmodifiableCollection` throws this exception for all add and remove operations.

## 18. How HashMap works in Java?

HashMap stores key-value pair in `Map.Entry` static nested class implementation. HashMap works on hashing algorithm and uses `hashCode()` and `equals()` method in `put` and `get` methods.

When we call `put` method by passing key-value pair, HashMap uses Key `hashCode()` with hashing to find out the index to store the key-value pair. The Entry is stored in the `LinkedList`, so if there are already existing entry, it uses `equals()` method to check if the passed key already exists, if yes it overwrites the value else it creates a new entry and store this key-value Entry.

When we call `get` method by passing Key, again it uses the `hashCode()` to find the index in the array and then use `equals()` method to find the correct Entry and return its value. Below image will explain these detail clearly.



The other important things to know about HashMap are capacity, load factor, threshold resizing. HashMap initial default capacity is **16** and load factor is 0.75. Threshold is capacity multiplied by load factor and whenever we try to add an entry, if map size is greater than threshold, HashMap rehashes the contents of map into a new array with a larger capacity. The capacity is always power of 2, so if you know that you need to store a large number of key-value pairs, for example in caching data from database, it's good idea to initialize the HashMap with correct capacity and load factor.

## 19. What is the importance of `hashCode()` and `equals()` methods?

HashMap uses Key object `hashCode()` and `equals()` method to determine the index to put the key-value pair. These methods are also used when we try to get value from HashMap. If these methods are not implemented correctly, two different Key's might produce same `hashCode()` and `equals()` output and in that case rather than storing it at different location, HashMap will consider them same and overwrite them.

Similarly all the collection classes that doesn't store duplicate data use `hashCode()` and `equals()` to find duplicates, so it's very important to implement them correctly. The implementation of `equals()` and `hashCode()` should follow these rules.

- If `o1.equals(o2)`, then `o1.hashCode() == o2.hashCode()` should always be true.
- If `o1.hashCode() == o2.hashCode` is true, it doesn't mean that `o1.equals(o2)` will be true.

## 20. Can we use any class as Map key?

We can use any class as Map Key, however following points should be considered before using them.

- If the class overrides `equals()` method, it should also override `hashCode()` method.
- The class should follow the rules associated with `equals()` and `hashCode()` for all instances. Please refer earlier question for these rules.
- If a class field is not used in `equals()`, you should not use it in `hashCode()` method.
- Best practice for user defined key class is to make it immutable, so that `hashCode()` value can be cached for fast performance. Also immutable classes make sure that `hashCode()` and `equals()` will not change in future that will solve any issue with mutability.

For example, let's say I have a class `MyKey` that I am using for `HashMap` key.

```
//MyKey name argument passed is used for equals() and hashCode()
MyKey key = new MyKey("Pankaj"); //assume hashCode=1234
myHashMap.put(key, "Value");

// Below code will change the key hashCode() and equals()
// but it's location is not changed.
key.setName("Amit"); //assume new hashCode=7890

//below will return null, because HashMap will try to look for key
//in the same index as it was stored but since key is mutated,
//there will be no match and it will return null.
myHashMap.get(new MyKey("Pankaj"));
```

This is the reason why `String` and `Integer` are mostly used as `HashMap` keys.

## 21. What are different Collection views provided by Map interface?

Map interface provides three collection views:

1. **Set<K> keySet():** Returns a Set view of the keys contained in this map. The set is backed by the map, so changes to the map are reflected in the set, and vice-versa. If the map is modified while an iteration over the set is in progress (except through the iterator's own remove operation), the results of the iteration are undefined. The set supports element removal, which removes the



corresponding mapping from the map, via the `Iterator.remove`, `Set.remove`, `removeAll`, `retainAll`, and `clear` operations. It does not support the `add` or `addAll` operations.

2. **`Collection<V> values()`**: Returns a Collection view of the values contained in this map. The collection is backed by the map, so changes to the map are reflected in the collection, and vice-versa. If the map is modified while an iteration over the collection is in progress (except through the iterator's own `remove` operation), the results of the iteration are undefined. The collection supports element removal, which removes the corresponding mapping from the map, via the `Iterator.remove`, `Collection.remove`, `removeAll`, `retainAll` and `clear` operations. It does not support the `add` or `addAll` operations.
3. **`Set<Map.Entry<K, V>> entrySet()`**: Returns a Set view of the mappings contained in this map. The set is backed by the map, so changes to the map are reflected in the set, and vice-versa. If the map is modified while an iteration over the set is in progress (except through the iterator's own `remove` operation, or through the `setValue` operation on a map entry returned by the iterator) the results of the iteration are undefined. The set supports element removal, which removes the corresponding mapping from the map, via the `Iterator.remove`, `Set.remove`, `removeAll`, `retainAll` and `clear` operations. It does not support the `add` or `addAll` operations.

## 22. What is difference between HashMap and Hashtable?

HashMap and Hashtable both implements Map interface and looks similar, however there are following difference between HashMap and Hashtable.

1. HashMap allows null key and values whereas Hashtable doesn't allow null key and values.
2. Hashtable is synchronized but HashMap is not synchronized. So HashMap is better for single threaded environment, Hashtable is suitable for multi-threaded environment.
3. `LinkedHashMap` was introduced in Java 1.4 as a subclass of HashMap, so incase you want iteration order, you can easily switch from HashMap to `LinkedHashMap` but that is not the case with Hashtable whose iteration order is unpredictable.
4. HashMap provides Set of keys to iterate and hence it's fail-fast but Hashtable provides Enumeration of keys that doesn't support this feature.
5. Hashtable is considered to be legacy class and if you are looking for modifications of Map while iterating, you should use `ConcurrentHashMap`.

## 23. How to decide between HashMap and TreeMap?

For inserting, deleting, and locating elements in a Map, the HashMap offers the best alternative. If, however, you need to traverse the keys in a sorted order, then `TreeMap` is your better alternative. Depending upon the size of your collection, it may be faster to add elements to a HashMap, then convert the map to a `TreeMap` for sorted key traversal.

## 24. What are similarities and difference between ArrayList and Vector?

`ArrayList` and `Vector` are similar classes in many ways.

1. Both are index based and backed up by an array internally.
2. Both maintains the order of insertion and we can get the elements in the order of insertion.
3. The iterator implementations of ArrayList and Vector both are fail-fast by design.
4. ArrayList and Vector both allows null values and random access to element using index number.

These are the differences between ArrayList and Vector.

1. Vector is synchronized whereas ArrayList is not synchronized. However if you are looking for modification of list while iterating, you should use CopyOnWriteArrayList.
2. ArrayList is faster than Vector because it doesn't have any overhead because of synchronization.
3. ArrayList is more versatile because we can get synchronized list or read-only list from it easily using Collections utility class.

## 25. **What is difference between Array and ArrayList? When will you use Array over ArrayList?**

Arrays can contain primitive or Objects whereas ArrayList can contain only Objects.

Arrays are fixed size whereas ArrayList size is dynamic.

Arrays doesn't provide a lot of features like ArrayList, such as addAll, removeAll, iterator etc.

Although ArrayList is the obvious choice when we work on list, there are few times when array are good to use.

- If the size of list is fixed and mostly used to store and traverse them.
- For list of primitive data types, although Collections use autoboxing to reduce the coding effort but still it makes them slow when working on fixed size primitive data types.
- If you are working on fixed multi-dimensional situation, using `[][]` is far more easier than `List<List<>>`

## 26. **What is difference between ArrayList and LinkedList?**

ArrayList and LinkedList both implement List interface but there are some differences between them.

1. ArrayList is an index based data structure backed by Array, so it provides random access to it's elements with performance as  $O(1)$  but LinkedList stores data as list of nodes where every node is linked to it's previous and next node. So even though there is a method to get the element using index, internally it traverse from start to reach at the index node and then return the element, so performance is  $O(n)$  that is slower than ArrayList.
2. Insertion, addition or removal of an element is faster in LinkedList compared to ArrayList because there is no concept of resizing array or updating index when element is added in middle.
3. LinkedList consumes more memory than ArrayList because every node in LinkedList stores reference of previous and next elements.

## 27. **Which collection classes provide random access of it's elements?**

ArrayList, HashMap, TreeMap, Hashtable classes provide random access to its elements. Download [java collections pdf](#) for more information.

## 28. What is EnumSet?

`java.util.EnumSet` is Set implementation to use with enum types. All of the elements in an enum set must come from a single enum type that is specified, explicitly or implicitly, when the set is created. EnumSet is not synchronized and null elements are not allowed. It also provides some useful methods like `copyOf(Collection c)`, `of(E first, E... rest)` and `complementOf(EnumSet s)`.

Check this post for [java enum tutorial](#).

## 29. Which collection classes are thread-safe?

Vector, Hashtable, Properties and Stack are synchronized classes, so they are thread-safe and can be used in multi-threaded environment. Java 1.5 Concurrent API included some collection classes that allows modification of collection while iteration because they work on the clone of the collection, so they are safe to use in multi-threaded environment.

## 30. What are concurrent Collection Classes?

Java 1.5 Concurrent package (`java.util.concurrent`) contains thread-safe collection classes that allow collections to be modified while iterating. By design Iterator implementation in `java.util` packages are fail-fast and throws `ConcurrentModificationException`. But Iterator implementation in `java.util.concurrent` packages are fail-safe and we can modify the collection while iterating. Some of these classes are `CopyOnWriteArrayList`, `ConcurrentHashMap`, `CopyOnWriteArraySet`.

Read these posts to learn about them in more detail.

- [Avoid ConcurrentModificationException](#)
- [CopyOnWriteArrayList Example](#)
- [HashMap vs ConcurrentHashMap](#)

## 31. What is BlockingQueue?

`java.util.concurrent.BlockingQueue` is a Queue that supports operations that wait for the queue to become non-empty when retrieving and removing an element, and wait for space to become available in the queue when adding an element.

BlockingQueue interface is part of java collections framework and it's primarily used for implementing producer consumer problem. We don't need to worry about waiting for the space to be available for producer or object to be available for consumer in BlockingQueue as it's handled by implementation classes of BlockingQueue.

Java provides several BlockingQueue implementations such as ArrayBlockingQueue, LinkedBlockingQueue, PriorityBlockingQueue, SynchronousQueue etc.

Check this post for use of BlockingQueue for [producer-consumer problem](#).

### 32. What is Queue and Stack, list their differences?

Both Queue and Stack are used to store data before processing them. `java.util.Queue` is an interface whose implementation classes are present in java concurrent package. Queue allows retrieval of element in First-In-First-Out (FIFO) order but it's not always the case. There is also Deque interface that allows elements to be retrieved from both end of the queue.

Stack is similar to queue except that it allows elements to be retrieved in Last-In-First-Out (LIFO) order.

Stack is a class that extends Vector whereas Queue is an interface.

### 33. What is Collections Class?

`java.util.Collections` is a utility class consists exclusively of static methods that operate on or return collections. It contains polymorphic algorithms that operate on collections, "wrappers", which return a new collection backed by a specified collection, and a few other odds and ends.

This class contains methods for collection framework algorithms, such as binary search, sorting, shuffling, reverse etc.

### 34. What is Comparable and Comparator interface?

Java provides Comparable interface which should be implemented by any custom class if we want to use Arrays or Collections sorting methods. Comparable interface has `compareTo(T obj)` method which is used by sorting methods. We should override this method in such a way that it returns a negative integer, zero, or a positive integer if "this" object is less than, equal to, or greater than the object passed as argument.

But, in most real life scenarios, we want sorting based on different parameters. For example, as a CEO, I would like to sort the employees based on Salary, an HR would like to sort them based on the age.

This is the situation where we need to use Comparator interface because

`Comparable.compareTo(Object o)` method implementation can sort based on one field only and we can't chose the field on which we want to sort the Object.

Comparator interface `compare(Object o1, Object o2)` method need to be implemented that takes two Object argument, it should be implemented in such a way that it returns negative int if first argument is less than the second one and returns zero if they are equal and positive int if first argument is greater than second one.

Check this post for use of Comparable and Comparator interface to [sort objects](#).

### 35. What is difference between Comparable and Comparator interface?

Comparable and Comparator interfaces are used to sort collection or array of objects.

Comparable interface is used to provide the natural sorting of objects and we can use it to provide sorting based on single logic.

Comparator interface is used to provide different algorithms for sorting and we can choose the comparator we want to use to sort the given collection of objects.

### 36. **How can we sort a list of Objects?**

If we need to sort an array of Objects, we can use `Arrays.sort()`. If we need to sort a list of objects, we can use `Collections.sort()`. Both these classes have overloaded `sort()` methods for natural sorting (using Comparable) or sorting based on criteria (using Comparator).

Collections internally uses Arrays sorting method, so both of them have same performance except that Collections take sometime to convert list to array.

### 37. **While passing a Collection as argument to a function, how can we make sure the function will not be able to modify it?**

We can create a read-only collection using `Collections.unmodifiableCollection(Collection c)` method before passing it as argument, this will make sure that any operation to change the collection will throw `UnsupportedOperationException`.

### 38. **How can we create a synchronized collection from given collection?**

We can use `Collections.synchronizedCollection(Collection c)` to get a synchronized (thread-safe) collection backed by the specified collection.

### 39. **What are common algorithms implemented in Collections Framework?**

Java Collections Framework provides algorithm implementations that are commonly used such as sorting and searching. Collections class contain these method implementations. Most of these algorithms work on List but some of them are applicable for all kinds of collections.

Some of them are sorting, searching, shuffling, min-max values.

### 40. **What is Big-O notation? Give some examples?**

The Big-O notation describes the performance of an algorithm in terms of number of elements in a data structure. Since Collection classes are actually data structures, we usually tend to use Big-O notation to choose the collection implementation to use based on time, memory and performance.

Example 1: `ArrayList.get(index i)` is a constant-time operation and doesn't depend on the number of elements in the list. So its performance in Big-O notation is  $O(1)$ .

Example 2: A linear search on array or list performance is  $O(n)$  because we need to search through entire list of elements to find the element.

#### 41. What are best practices related to Java Collections Framework?

- Choosing the right type of collection based on the need, for example if size is fixed, we might want to use Array over ArrayList. If we have to iterate over the Map in order of insertion, we need to use TreeMap. If we don't want duplicates, we should use Set.
- Some collection classes allow to specify the initial capacity, so if we have an estimate of number of elements we will store, we can use it to avoid rehashing or resizing.
- Write program in terms of interfaces not implementations, it allows us to change the implementation easily at later point of time.
- Always use Generics for type-safety and avoid ClassCastException at runtime.
- Use immutable classes provided by JDK as key in Map to avoid implementation of hashCode() and equals() for our custom class.
- Use Collections utility class as much as possible for algorithms or to get read-only, synchronized or empty collections rather than writing own implementation. It will enhance code-reuse with greater stability and low maintainability.

#### 42. What is Java Priority Queue?

PriorityQueue is an unbounded queue based on a priority heap and the elements are ordered in their natural order or we can provide **Comparator** for ordering at the time of creation. PriorityQueue doesn't allow null values and we can't add any object that doesn't provide natural ordering or we don't have any comparator for them for ordering. Java PriorityQueue is not **thread-safe** and provided  $O(\log(n))$  time for enqueueing and dequeuing operations. Check this post for [java priority queue example](#).

#### 43. Why can't we write code as `List<Number> numbers = new ArrayList<Integer>();`?

Generics doesn't support sub-typing because it will cause issues in achieving type safety. That's why `List<T>` is not considered as a subtype of `List<S>` where `S` is the super-type of `T`. To understand why it's not allowed, let's see what could have happened if it has been supported.

```
List<Long> listLong = new ArrayList<Long>();  
listLong.add(Long.valueOf(10));  
List<Number> listNumbers = listLong; // compiler error  
listNumbers.add(Double.valueOf(1.23));
```

As you can see from above code that IF generics would have been supporting sub-typing, we could have easily add a Double to the list of Long that would have caused `ClassCastException` at runtime while traversing the list of Long.

#### 44. **Why can't we create generic array? or write code as** **`List<Integer>[] array = new ArrayList<Integer>[10];`**

We are not allowed to create generic arrays because array carry type information of it's elements at runtime. This information is used at runtime to throw `ArrayStoreException` if elements type doesn't match to the defined type. Since generics type information gets erased at compile time by Type Erasure, the array store check would have been passed where it should have failed. Let's understand this with a simple example code.

```
List<Integer>[] intList = new List<Integer>[5]; // compile error
Object[] objArray = intList;
List<Double> doubleList = new ArrayList<Double>();
doubleList.add(Double.valueOf(1.23));
objArray[0] = doubleList; // this should fail but it would pass because at runtime
intList and doubleList both are just List
```

Arrays are covariant by nature i.e `S[]` is a subtype of `T[]` whenever `S` is a subtype of `T` but generics doesn't support covariance or sub-typing as we saw in last question. So if we would have been allowed to create generic arrays, because of type erasure we would not get array store exception even though both types are not related.

To know more about Generics, read [Java Generics Tutorial](#).

I will keep on adding more questions on java collections framework as and when I found them, if you found it useful please share it with others too, it motivates me in writing more like these. □

Please let me know if I have missed any important question, I will include that to list.

**« PREVIOUS**[Composition in Java Example](#)**NEXT »**[How to create JAR from Java Classes](#)**About Pankaj**

If you have come this far, it means that you liked what you are reading. Why not reach little more and connect with me directly on [Google Plus](#), [Facebook](#) or [Twitter](#). I would love to hear your thoughts and opinions on my articles directly.

Recently I started creating video tutorials too, so do check out my videos on [Youtube](#).

FILED UNDER: [INTERVIEW QUESTIONS](#), [JAVA](#)

**Comments****Java says**

JULY 1, 2018 AT 2:22 AM



Please make correct question 12. For-each loop under the hood use iterator to go through elements of collection and It throw ConcurrentModificationException if something has changed. The explanation to this question sounds like "Iterator is better than Iterator".

[Reply](#)

**Tsetso says**

JUNE 8, 2018 AT 7:03 AM

In the "What are the basic interfaces of Java Collections Framework?" is missed Queue Collection Interface which is one of the four based Java Collection interfaces

[Reply](#)

**Venus says**

APRIL 20, 2018 AT 10:22 AM

Can someone pls tell difference between for loop and iterator .. even in Iterator we can not modify collection as it throws Concurrent ModificationException.

[Reply](#)

**Tsetso says**

JUNE 8, 2018 AT 7:10 AM

If I remember correctly iterator is released in java 1.2

Advantage of iterator is that you can remove element during the iteration. Also you can you iterate backward.

Hope this help you

[Reply](#)

**Raghib says**

JULY 2, 2018 AT 9:07 AM

Iterator doesn't move backward it is a One Directional Cursor Whereas ListIterator is a Bidirectional cursor.

In Enumeration:- Read Only

In Iterator:- Read/Remove

In ListIterator:- Read/Remove/Add/replace.

[Reply](#)

**Nand says**

MARCH 23, 2018 AT 9:39 AM

Hi Pankaj.

Please explain once again below line .

Hashtable is synchronized but HashMap is not synchronized. So HashMap is better for single threaded environment, Hashtable is suitable for multi-threaded environment.

[Reply](#)**Sanjay Roy says**

JANUARY 28, 2018 AT 10:44 AM

Hi,

I do follow your posts for various topics in Java. As I am not very old as a programmer , I was going through the fail-fast property of Iterator by writing small programs to just get accustomed with the exception it throws when any modification is done to the underlying Collection being iterated by it, For me the concept goes well with List but when I was implementing the fail-fast property of Iterator with Map, I was not getting any Exception even after modification to the Collection. I am attaching a snippet of the code below. Please do check and let me know why I wasn't getting any Exception even after I add Entries in the Map.

```
Map test = new HashMap();
test.put(1, "abc");
test.put(2, "cde");
test.put(3, "efg");
test.put(4, "ghi");
test.put(5, "ijk");
test.put(6, "klm");
Iterator iterator = test.keySet().iterator();
while(iterator.hasNext()) {
    if(it.next().equals(6)) {
        test.put(6, "yui");
        test.put(7, "lll");
        test.put(8, "kkk");
        test.put(9, "sss");
    }
}
for(Map.Entry m: test.entrySet()) {
    System.out.println(m.getKey() + " " + m.getValue());
}
```

Do bare me with all the reference names been used in the program. It was only for test purpose

[Reply](#)**Pankaj says**

JANUARY 28, 2018 AT 8:33 PM

The problem is with your if condition, you are doing the modification on the last item. Change it to `if(it.next().equals(4))` and you will get the exception.

Reply

**Sanjay Roy says**

JANUARY 28, 2018 AT 9:15 PM

Thanks for the response. It does work but can you tell me if possible why the exception is not getting thrown if the condition is made on the last entry?. I am modifying something there also. Thanks in advance

Reply

**arvind says**

FEBRUARY 15, 2018 AT 9:00 AM

Because it is the next() method that is throwing the ConcurrentModificationException and if it is not called then no exception.

And this is what happening in your code.

Reply

**Rajesh says**

NOVEMBER 24, 2017 AT 5:24 PM

Very good collection...

Reply

**Sanju says**

SEPTEMBER 19, 2017 AT 10:10 PM

Question#44 : "Since generics type information gets erased at runtime "  
but according to <https://docs.oracle.com/javase/tutorial/java/generics/erasure.html> information  
erased at compile time. Is there any misunderstand from my side?

Reply

**Pankaj says**

SEPTEMBER 19, 2017 AT 10:21 PM

You are absolutely right, it was a typo error. I have fixed the post. Thanks for the comment, I appreciate it.

[Reply](#)

**Anurag Singh says**

JULY 24, 2017 AT 3:08 AM

this post is awesome for every one who wanted to crack the java interview.  
thank you so much.

[Reply](#)

**Veera Kanisetty says**

JULY 27, 2017 AT 12:26 AM

Really superb , looking for these kind of stuff in all Java frameworks as well. Thanks

[Reply](#)

**livermor says**

JANUARY 19, 2017 AT 6:02 PM

4 point is wrong.  
Map is not under the Collections interface.

[Reply](#)

**Vineet Kapoor says**

MAY 29, 2017 AT 1:48 AM

Yes True.

[Reply](#)

**Vikas Palakurthi says**

MAY 29, 2017 AT 11:22 PM

The question is "What are the basic interfaces of Java Collections Framework". It specifically says "Collections Framework".

Map is an interface in Collections framework, but is not a sub-interface of Collection interface.

[Reply](#)

**Avisha Jindal says**

NOVEMBER 15, 2016 AT 11:30 PM

I was once asked in an interview. What is the meaning of making an array list 'final'? Does it mean that we won't be able to add elements to it? Till date whomever i've asked this question to , has not been able to answer it correctly. Please add more such type of tricky interview questions.

[Reply](#)**Tarun says**

DECEMBER 14, 2016 AT 3:33 AM

@avisha if you use final with any attribute or any other thing you can't instantiate again eg . final int i =6;

you not change value of i again same as

final List list = new ArrayList (); you cant instantiate it again like list = new LinkedList(); or something but you add the vlaue inside the list .

[Reply](#)**Vineet Kapoor says**

MAY 29, 2017 AT 1:51 AM

This means that you cannot rebind the variable to point to a different collection instance:

```
final List list = new ArrayList();
```

```
list = new ArrayList(); // Since 'list' is final, this won't compile
```

[Reply](#)**zurreyab says**

JULY 17, 2017 AT 2:50 AM

compile

[Reply](#)**Rupa says**

APRIL 12, 2018 AT 7:40 AM

It simply means that you cant change or modify its reference.

```
final List al = new ArrayList();
```

The al reference is pointing to the object Arraylist() cannot be modified

[Reply](#)

**Dabbiru Gautam says**

NOVEMBER 15, 2016 AT 9:48 PM

HI,

Great work done for collating all questions at one place.

For 36. The object needs to Implement Comparable or Comparator first and should have the implementation for compareTo() or compare(). Then we can use Collections.sort().

Regards,

Gautam

[Reply](#)**Kannan says**

MAY 5, 2017 AT 2:40 AM

When you use collection to hold any wrapper class objects, then there is no need of implementing Comparable or Comparator interface.

Ex: List list = new ArrayList();

Collections.sort(list); //will sort based on lexicographical.

When you use any custom objects (says Person, Employee), then you have implement any of interface and tell based which fields collections has to be sorted.

Say ArrayList holds list of Employee object, and you want to sort by employee salary or name then you have to go with any above interface.

Ex. List empList = new ArrayList();

Collections.sort(empList); //compile time error if you don't implement Comparable or Comparator interface

[Reply](#)**Vineet Kapoor says**

MAY 29, 2017 AT 1:55 AM

Yes true

[Reply](#)**vedant maheshwari says**

AUGUST 27, 2016 AT 8:31 AM

hi Pankaj ,

Can you give more explanation on Q44.

As it says that Arrays doesn't allow to be generic. so what happen in case of Q44.  
Will it give error or simply store the list of elements of integer and double?

[Reply](#)

**Pankaj says**

AUGUST 27, 2016 AT 11:00 AM

It will give compile error, the code is just to show what will happen if we assume that code will not give compile error.

[Reply](#)

**GOPINATH M B says**

JULY 8, 2016 AT 9:30 PM

Good collection bro...thank you:)

[Reply](#)

**Dhananjay says**

JUNE 21, 2016 AT 10:29 PM

Good work, very clear explanation.

[Reply](#)

**Shreya says**

MARCH 21, 2016 AT 3:38 AM

Thank you so much for providing the explanations.

[Reply](#)

**ramana reddy says**

FEBRUARY 10, 2016 AT 12:18 PM

Amazing work, very detailed. Thank you so much.

[Reply](#)

**Raju says**

JANUARY 26, 2016 AT 7:42 AM

@Pankaj how are writing this much long tutorial. You didn't get married ?? ☐

But any how i love your work and effort its very useful. thanks

[Reply](#)

**Pankaj says**

JANUARY 26, 2016 AT 8:11 AM

This is funny and nice. Actually I am married and my wife provide a lot of support in this and help me motivated.

[Reply](#)

**Inder says**

AUGUST 13, 2016 AT 9:38 AM

Hi Pankaj ,

Can you please revisit the answer to the question 22. I guess in HashMap only null keys are supported and not null values..

[Reply](#)

**Pankaj says**

AUGUST 13, 2016 AT 10:35 PM

For these, best way to confirm is write a simple program. See below program compiles and runs fine.

```
import java.util.HashMap;

public class Test {

    public static void main(String args[]){

        HashMap<String,String> m = new HashMap<String,String>();

        m.put(null,null);
        m.put("1","1");

        System.out.println(m);
    }

}
```

[Reply](#)



**Kannan says**

MAY 5, 2017 AT 2:42 AM

HashMap will allow only one null but n number of null values.

[Reply](#)**Bharath says**

JANUARY 12, 2016 AT 4:01 AM

Very Nice collection Interview questions.

Thanks Pankaj.

[Reply](#)**Pratap Shinde says**

JANUARY 11, 2016 AT 10:05 AM

Great work, Pankaj

[Reply](#)**vivek says**

DECEMBER 21, 2015 AT 6:11 AM

Please do a program by adding scanning without using add method.

[Reply](#)**Bhaumik says**

DECEMBER 3, 2015 AT 4:57 AM

Hello sir i cant see questions answers after the 21 no questions

[Reply](#)**Viswa says**

OCTOBER 8, 2015 AT 11:22 PM

Excellent material..A big thanks for your efforts !!

[Reply](#)

**Ashish Mane says**

AUGUST 3, 2015 AT 12:17 AM

Grate efforts....keep it up.

[Reply](#)

**Arun says**

JULY 12, 2015 AT 12:17 AM

Dear Sir;

Greetings!!!

I have some doubt in Collection Frame Work-

1-Why we use interface why we did not use directly class.

like list interface accept duplicacy and arraylist iimplement it

then why we did not implement all the list feature in array list directly

and what is advantage of list iterface

[Reply](#)

**Quang says**

JANUARY 27, 2016 AT 10:03 PM

The interface useful as you want to change the implementation classes.

Maybe this time, you're using ArrayList and in five years I assume that your business will be upgraded so you have to replace ArrayList by LinkedList. If you use the implementation classes in function arguments , you have to replace all of them but if you use the interface, you dont need to update.

Let see example below:

void getInfo(List b) -> you can pass any implementation classes

void getInfo(ArrayList b) -> you can only pass ArrayList class.

[Reply](#)

**RAHUL SAXENA says**

JULY 11, 2015 AT 8:02 AM

Can you please explain in concurrent collection why iterator does not throw concurrent modification exception.You have written that it works on the clone of that object.

So please explain it deeply like where this clone is maintained and what operations are performed in the memory??

[Reply](#)**Ankita says**

AUGUST 18, 2017 AT 1:14 AM

In concurrent collection like CopyOnWriteArrayList and CopyOnWriteArrayset, for every update operation a cloned copy of original object is created and the update operation is performed on that cloned copy rather than original copy.

Later on this cloned copy and original copy object are synched by JVM internally to perform the update. Since the update is performed on cloned copy rather than original copy so we don't get concurrentModificationException.

Also if we have many write operations it is not recommended to use this concurrent class becoz for every update one cloned copy is created and performance degrades.

[Reply](#)**Vishwadeep Aggarwal says**

APRIL 2, 2015 AT 2:57 AM

Could you please explain Que 29. and if Collection Classes are thread safe then please explain how??

[Reply](#)**Paresh Bhavsar says**

MARCH 19, 2015 AT 11:46 PM

very good Collections of questions and answers ... thanx.

[Reply](#)**Vishy says**

MARCH 16, 2015 AT 12:16 PM

Fine list of questions & answers..

[Reply](#)**Eswaran says**

MARCH 16, 2015 AT 10:21 AM

please tell about equals and hashCode method deeply

[Reply](#)

**Vinod Karathiya says**

MARCH 13, 2015 AT 11:50 AM

Really Nice work, As experienced person I don't have these much knowledge. Again Thanks a lot.

[Reply](#)**Sharmi says**

JANUARY 21, 2015 AT 10:13 PM

Please use the language short and precise. I read a question "Why Collection doesn't extend Cloneable and Serializable interfaces?" it took me 5 minutes to understand what exactly are you trying to say.

[Reply](#)**dakshina gandikota says**

JANUARY 10, 2015 AT 5:31 PM

In Question#20

Line 12: myHashMap.get(new MyKey("Pankaj"));

If Line 12: were to be

myHashMap.get(key);

would the "theory" that mutable class as key is a bad idea hold?

By the way, the definition of mutable is up in the air. Is it a final class with final methods and fields?

[Reply](#)**dakshina gandikota says**

JANUARY 7, 2015 AT 2:49 PM

Questions 14 and 30 contradict each other wrt ConcurrentHashMap.

[Reply](#)**Pankaj says**

JANUARY 9, 2015 AT 1:51 AM

No it was not contradicting, however answer in Q30 was confusing. I have modified it and it should be clear now.

[Reply](#)**dakshina gandikota says**

JANUARY 9, 2015 AT 10:10 AM

Thanks. Why not Vector for random access of its contents, along side ArrayList, HashMap, TreeMap and Hashtable?

[Reply](#)

**Anoop says**

DECEMBER 23, 2014 AT 2:13 AM

Correction: "If we have to iterate over the Map in order of insertion, we need to use TreeMap". Its LinkedHashMap.

[Reply](#)

**Vu Le says**

DECEMBER 30, 2014 AT 10:57 PM

If we have to iterate over the Map in order of insertion, we need to use LinkedHashMap. With TreeMap if you need to iterate by sorted order

[Reply](#)

**kiran says**

DECEMBER 20, 2014 AT 1:37 AM

Great Job sir..... Thank u

[Reply](#)

**Pabel Lopez says**

NOVEMBER 23, 2014 AT 11:03 PM

It will be very helpful to have an image of the main collections (collection, set, list, Queue and map) sometimes is easier to watch the image as a tree than reading the description.

Thank you.

[Reply](#)

**sampath kumar madala says**

NOVEMBER 22, 2014 AT 9:26 AM

Q: 30- " By design iterator is fail fast and throws ConcurrentModificationException" Is this statement conflicts with explanation in Q:14 !!!

[Reply](#)

**Pankaj says**

NOVEMBER 22, 2014 AT 10:11 AM

Don't get confused, both are right explanation. Iterator is just an interface and their corresponding implementation are either fail-fast or fail-safe.

All the iterator implementation in java.util package collection classes are fail-fast whereas collection classes in java.util.concurrent implement fail-safe iterator.

[Reply](#)

**Prakash says**

NOVEMBER 19, 2014 AT 2:57 AM

What is ThreadLocal? Please explain with some realtime example?

[Reply](#)

**Jagadeesh says**

OCTOBER 16, 2014 AT 2:16 AM

Hi Pankaj,

I'm great to see this site, It is very useful for Interviews, But here I found only 21 questions and answers, Can u post all the answers?

Thanks a lot

[Reply](#)

**Dharmendra says**

SEPTEMBER 21, 2014 AT 1:43 AM

Very good collection of Collection questions ☐

[Reply](#)

**Danny says**

SEPTEMBER 18, 2014 AT 11:04 PM

each and every question is very nicely explained, good job!

[Reply](#)

**Aditya Peshave says**

SEPTEMBER 14, 2014 AT 8:05 PM

Hi,

really nice tutorial.

Just a correction:

the default capacity of HashMap is 16.

reference:

<http://greppcode.com/file/repository.greppcode.com/java/root/jdk/openjdk/6-b14/java/util/HashMap.java#HashMap.%3Cinit%3E%28%29>

[Reply](#)

**Karl says**

AUGUST 26, 2014 AT 1:09 AM

Very good collection of Collections interview questions.

[Reply](#)

**Karl says**

AUGUST 25, 2014 AT 7:57 PM

Very good Collections interview questions!

[Reply](#)

**Amit says**

AUGUST 22, 2014 AT 1:28 AM

wow! Really very nice articles

[Reply](#)

**vinita says**

JULY 22, 2014 AT 3:58 AM

good collection of question....thanx

[Reply](#)

**pk says**

JULY 9, 2014 AT 4:44 AM

gud collection of questions

[Reply](#)**sudheer says**

JULY 4, 2014 AT 3:25 AM

awesome posts by you pankaj...thanks a lot.

[Reply](#)**Arti says**

JUNE 28, 2014 AT 8:30 AM

I am not able to view all the questions though i am following you on twitter. I was able to view all the 40 questions for some days but not now. Please check.

[Reply](#)**chandan says**

JUNE 6, 2014 AT 1:27 AM

Thanks Pankaj...Excellent post. Please also add answers for questions from 22 onwards.

[Reply](#)**Abhay says**

JUNE 5, 2014 AT 8:14 AM

Hi Pankaj,

You have not added answers to all questions listed above ? Please provide answers also if possible.

Your answers are easy to understand and very clear. Thanks in advance.

[Reply](#)**Abhay says**

JUNE 5, 2014 AT 8:17 AM

Please ignore:)

[Reply](#)



**Maruthi says**

MAY 4, 2014 AT 10:40 PM

Good collection of collections interview questions with nice explanation

[Reply](#)

**jkbrk says**

MARCH 19, 2014 AT 7:36 PM

Great site! One question about no. 40:

"If we have to iterate over the Map in order of insertion, we need to use TreeMap."

Shouldn't it be:

"If we have to iterate over the Map in order of insertion, we need to use **LinkedHashMap**." or

"If we have to iterate over the Map in **natural order**, we need to use TreeMap."

Thanks.

[Reply](#)

**Sateesh Kumar U says**

MARCH 19, 2014 AT 5:58 PM

Great work. Thanks alot Pankaj..!!

[Reply](#)

**Sushil says**

MARCH 12, 2014 AT 5:12 AM

Thank you very much.

I deeply appreciate your efforts.

[Reply](#)

**nikhil sethiya says**

MARCH 7, 2014 AT 9:58 AM

great work sir.....!!

[Reply](#)

**nikhil sethiya says**

MARCH 7, 2014 AT 9:57 AM

Excellent Work.....!!

[Reply](#)**jameer says**

MARCH 6, 2014 AT 7:54 AM

Hello, Pankaj.

Awesome Explanation.

[Reply](#)**vidhya says**

MARCH 1, 2014 AT 9:27 AM

Amazing work, thanks a lot.

[Reply](#)**Narayanarao Menda says**

FEBRUARY 17, 2014 AT 7:46 PM

It's amazing tutorial...thank u very much..

[Reply](#)**Shelly says**

FEBRUARY 11, 2014 AT 9:40 AM

Hi Pankaj,

Its a great article just one thing that i question 40 you mention If we have to iterate over the Map in order of insertion, we need to use TreeMap it shouldn't be linkedhashmap instead of treemap.

Thanks

[Reply](#)**Pankaj says**

FEBRUARY 12, 2014 AT 12:05 AM

We can use both TreeMap or LinkedHashMap.

[Reply](#)

**Brijeshwar says**

SEPTEMBER 27, 2014 AT 9:39 AM

Hi Pankaj,

Insertion order is maintained only in LinkedHashMap whereas TreeMap maintains key based SortedOrder, Please correct me if I am wrong.

[Reply](#)**Vu Le says**

DECEMBER 30, 2014 AT 11:00 PM

It should be LinkedHashMap not TreeMap, TreeMap with natural order

[Reply](#)**Divya says**

FEBRUARY 2, 2014 AT 10:46 AM

Hello Pankaj,

Can you please explain the the question no 43 "Why can't we create generic array" in more detail and in easy way. It's very difficult to understand it.

Please help me out.

Thanks,

Divya ☐

[Reply](#)**Pankaj says**

FEBRUARY 2, 2014 AT 9:34 PM

Let's get through each line in the sample code and try to understand what is happening.

1. `List<Integer>[] intList = new List<Integer>[5];`

This line will throw compile time exception now because we can't have generic array. Just for the sake of argument, let's assume that it's allowed and compiler don't throw any error.

Since generics type information gets erased at runtime by Type Erasure, it will be like an array of List at runtime like `List[] intList = new List[5];`

2. `Object[] objArray = intList;`

Perfectly valid statement, creating an object array.

3. `List<Double> doubleList = new ArrayList<Double>();`

At runtime `List doubleList = new ArrayList();`

4. `doubleList.add(Double.valueOf(1.23));`

Valid statement, adding Double to the list.

5. objArray[0] = doubleList;

You see the problem here, since the objArray is referring to intList, we are basically setting doubleList to one of the elements of the array that is supposed to contain only Integer List. And it will work fine because for Array both intList and doubleList are just list, since there is no type information available.

That's why Generic Arrays are not supported in Java.

[Reply](#)

#### Harsha says

JANUARY 26, 2014 AT 4:06 PM

Good one !!! Keep it up , your explanation is good and practical .

[Reply](#)

#### Rajan says

JANUARY 21, 2014 AT 9:44 AM

Hi Pankaj,

Thanks for such a nice detail on collections.

[Reply](#)

#### Akhil says

JANUARY 18, 2014 AT 3:01 PM

Hi Pankaj,

In question no 19, below line will always return null.

```
myHashMap.get(new MyKey('Pankaj'));
```

since its creating new key everytime and there is no value available for this key. Please update it,if i'm correct and one more please replace all the strings enclosed with double quotes.

Thanks,

Akhil

[Reply](#)

#### Pankaj says

JANUARY 18, 2014 AT 6:51 PM

Not that is not true. As stated HashMap uses equals() and hashCode() functions to find the value in the map. So if MyKey override these methods and provide correct implementation, then the hashCode will be same as any other value stored with same hashCode and we will get it. The code given in the question is to prove that immutability is required to avoid any data integrity issues in the

Map, because if we change the data hashCode is changed and we will never be able to get the associated value.

And thanks for pointing out the quotation marks, I will change it.

[Reply](#)

**sam says**

JANUARY 16, 2014 AT 12:33 PM

Hi Pankaj ,

I was looking for some good interview question and found this . Its really very helpful and descriptive. You deserve praise for it . Excellent work Sir and keep it up !

[Reply](#)

**Malli says**

DECEMBER 20, 2013 AT 7:11 AM

Hi Pankaj,

Thanks for given a good kind of materiel. very useful to us. I request you, please provide JSF, EJB and JPA.

[Reply](#)

**krishna says**

DECEMBER 5, 2013 AT 5:06 AM

Thanks a TON ☐

[Reply](#)

**Snehal Masne says**

DECEMBER 1, 2013 AT 5:22 PM

Thanks a lot for this blog. I bookmarked it ☐

Thanks a lot buddy!!

[Reply](#)

**test says**

JANUARY 1, 2014 AT 11:25 AM

wow gr8 that u bukmrkd it....

[Reply](#)

**Ganesh says**

DECEMBER 1, 2013 AT 10:25 AM

Good Knowledge sharing and valuable informations not only from interview aspective but also to grow technically strong.

You are doing a fantastic job, Pls continue.

[Reply](#)

**Arun says**

NOVEMBER 15, 2013 AT 5:06 PM

good work Pankaj. its a wonderful collection..

[Reply](#)

**siri says**

NOVEMBER 13, 2013 AT 5:44 AM

Collections is a class it will be helpful to collection framework by providing methods.

[Reply](#)

**Rameshwar Singh says**

NOVEMBER 11, 2013 AT 6:48 AM

Hi,

Kindly help to remove extra characters from the code like "".

[Reply](#)

**siri says**

NOVEMBER 13, 2013 AT 5:41 AM

It is collection framework not collections framework.....

[Reply](#)

**Abhishek Singh says**

NOVEMBER 8, 2013 AT 8:19 AM

Wow, beautyfull artical in simple language.....Thanks.

[Reply](#)**Naresh says**

NOVEMBER 5, 2013 AT 10:35 AM

Does the JDBC-ODBC Bridge support multiple concurrent open statements per connection?  
if No, How?

[Reply](#)**Sandeep says**

OCTOBER 29, 2013 AT 11:18 AM

Really helpful. keep it up ☐

[Reply](#)**Yogesh says**

JULY 18, 2013 AT 11:23 AM

Thanks.It was really nice. Keep such good work up!

[Reply](#)**G.Palanikumar says**

JULY 15, 2013 AT 6:13 AM

Respected Sir,

Really i proud of you. you were posted many questions always i am welcome. but company never discuss to candidate in first time they give some programs for interview time and write the output. i am expecting these kind of sample question with answer. thank you sir. if you find any mistake my words please forgive me sir. that is my suggestion thats all

[Reply](#)**Pankaj says**

JULY 15, 2013 AT 7:49 PM

It depends on the interview type, if its telephonic round then expect more questions on core java related technologies. For programming questions, until unless you have deep knowledge of core

technologies, you won't be able to answer them with confidence and answer the followup questions.

[Reply](#)

**hmm says**

JULY 11, 2013 AT 7:34 PM

what

[Reply](#)

**Manish Dass says**

JUNE 20, 2013 AT 12:42 PM

Good collection of questions from Collections.  
Fantastic and very helpful.

[Reply](#)

**Amit says**

JUNE 19, 2013 AT 3:53 PM

This is really one of the best !!. Thanks a lot. Please keep sharing more.

[Reply](#)

**vivek says**

JUNE 19, 2013 AT 8:31 AM

Hi,

This is regarding question 17, there you mention "if map size is greater thn threshold , hashmap rehashes the content s of hashmap into new 'ARRAY' with larger capacity"

Don't we use linkedlist for for storing hasmap, I am beginner.

Can you please explain ?

Regards

Vivek

[Reply](#)

**Pankaj says**

JUNE 19, 2013 AT 7:40 PM



Here I am referring to the array that keeps indexes to the linked list, check the image in question 17 for better understanding.

[Reply](#)

**Leo says**

JUNE 11, 2013 AT 4:45 PM

Excellent post.... Really learnt lot of things in collections with full clarity....  
thanks alot..

[Reply](#)

**Rahul Sharma says**

JUNE 9, 2013 AT 6:58 PM

Very well written and covering almost all the interviews related questions....great job indeed!!!

[Reply](#)

**Tarun says**

JUNE 8, 2013 AT 5:22 AM

nice collection of quesction thank for sharing.

[Reply](#)

**Brijs Modi\_1 says**

JUNE 7, 2013 AT 1:28 PM

Good one

[Reply](#)

**shashi says**

JUNE 6, 2013 AT 10:43 AM

good work very useful

[Reply](#)

**Rameshwar Singh says**

JUNE 4, 2013 AT 10:34 AM

Thanks a lot!!!

This covers everything what an individual expects ☐

Really helpful.. keep up the good work..

[Reply](#)**Brijs says**

JUNE 7, 2013 AT 1:26 PM

Good one

[Reply](#)**praveen says**

JUNE 4, 2013 AT 1:42 AM

great work pankaj.... this helped me a lot

[Reply](#)**Praveen Kumar says**

JUNE 2, 2013 AT 11:02 AM

Thanks a lot for this blog!! I bookmarked it..Thanks a lot buddy!!

[Reply](#)**abc says**

MAY 15, 2013 AT 10:21 AM

Very Nice ...

[Reply](#)**Anupam says**

MAY 15, 2013 AT 9:45 AM

itz realy so good

[Reply](#)

**Genia says**

MAY 14, 2013 AT 12:39 AM

After looking at a number of the blog posts on your blog, I seriously appreciate your way of blogging. I book marked it to my bookmark site list and will be checking back soon. Take a look at my web site as well and tell me your opinion.

[Reply](#)

**I M Futane says**

APRIL 20, 2013 AT 2:02 AM

Good Question Collect From Util Package

[Reply](#)

**Adinath Bhawani says**

MARCH 15, 2013 AT 5:38 PM

a good collection of Java

[Reply](#)

**Sushil says**

FEBRUARY 22, 2013 AT 9:35 AM

Its a very good collection of Collection IQ.

Keep the work going and Thank you very much for sharing

[Reply](#)

**Ambuj Tailang says**

MAY 3, 2013 AT 11:57 AM

Awesome collection

[Reply](#)

**GNS says**

FEBRUARY 12, 2013 AT 8:02 PM

Very good collection of Collections interview questions. ...

[Reply](#)

## Leave a Reply

Your email address will not be published. Required fields are marked \*

Comment

Name \*

Email \*

☐

Save my name, email, and website in this browser for the next time I comment.

POST COMMENT

---

DOWNLOAD ANDROID APP

---



---

## CORE JAVA TUTORIAL

---

Java 10 Tutorials   Java 9 Tutorials  
Java 8 Tutorials   Java 7 Tutorials   Core  
Java Basics   OOPS Concepts   Data  
Types and Operators   String Manipulation  
Java Arrays   Annotation and Enum  
Java Collections   Java IO Operations  
Java Exception Handling  
MultiThreading and Concurrency  
Regular Expressions   Advanced Java  
Concepts

---

---

## IMPORTANT INTERVIEW QUESTIONS

---

## Java Interview Questions

- > [Core Java Interview Questions](#)
- > [String Interview Questions](#)
- > [Multithreading Interview Questions](#)
- > [Collections Interview Questions](#)
- > [Exception Interview Questions](#)
- > [Java Programming Interview Questions](#)
- > [Java 8 Interview Questions Part 1](#)
- > [Java 8 Interview Questions Part 2](#)
- > [Servlet Interview Questions](#)
- > [JSP Interview Questions](#)
- > [Struts 2 Interview Questions](#)
- > [JDBC Interview Questions](#)
- > [Spring Interview Questions](#)
- > [Hibernate Interview Questions](#)
- > [JSF Interview Questions](#)
- > [Web Services Interview Questions](#)
- > [Scala Basic Interview Questions](#)
- > [Scala Intermediate Interview Questions](#)
- > [Scala Advanced Interview Questions](#)
- > [Scala Interview Questions Summary](#)
- > [Common Job Interview Questions](#)

Miscellaneous

- > [Java ClassLoader](#)
- > [String StringBuffer StringBuilder](#)
- > [Java is Pass By Value](#)
- > [Java Heap vs Stack Memory](#)

RECOMMENDED TUTORIALS

Java Tutorials

- > [Java IO](#)
- > [Java Regular Expressions](#)
- > [Multithreading in Java](#)
- > [Java Logging](#)
- > [Java Annotations](#)
- > [Java XML](#)
- > [Collections in Java](#)
- > [Java Generics](#)
- > [Exception Handling in Java](#)
- > [Java Reflection](#)
- > [Java Design Patterns](#)
- > [JDBC Tutorial](#)

Java EE Tutorials

- > [Servlet JSP Tutorial](#)
- > [Struts2 Tutorial](#)
- > [Spring Tutorial](#)
- > [Hibernate Tutorial](#)
- > [Primefaces Tutorial](#)
- > [Apache Axis 2](#)
- > [JAX-RS](#)
- > [Memcached Tutorial](#)

© 2018 · Privacy Policy · Don't copy, it's Bad Karma · Po



অরক্ষিত মিলনের  
৭২ ঘণ্টার মধ্যে  
১টি নরপিল™  
আপনাকে রাখবে  
সম্পূর্ণ দুর্শিতামুক্ত

Since 1958  
**SQUARE**  
PHARMACEUTICALS LTD.  
BANGLADESH