Java Arrays   Java Strings   Java OOPs   Java Collection   Java 8 Tutorial   Java Multithreading   Java Exceptio

Java Collections Interview Questions and Answers

Last Updated : 17 Jan, 2025

Java Collection Framework was introduced in *JDK 1.2* which contains all the collection classes and interfaces. **Java Collection** is a framework that provides a mechanism to store and manipulate the collection of objects. It allows developers to access prepackaged data structures and algorithms for manipulating data.

Now, Here in this Interview questions on Java Collection, we've covered the **50+ Java Collections Framework Questions** along with their answers tailored for both **Fresher** and **experienced professionals**, which cover everything from basic to advanced Java collection concepts such as navigation collection, WeakHashMap, streams Lambdas, etc.

Whether you are a **fresher** or an **experienced Java developer**, these **Java Collections Interview Questions** give you all the confidence you need to ace your next Java interview.

 Table of Content

We have divided the 50 questions into two parts: Experienced and Freshers. Let's begin with the questions for Freshers.

Java Collection Interview Questions For Freshers

1. What is Collection in Java?

The term **collection** refers to a group of objects represented as one unit. Classes in the Java collection class hierarchy are divided into two "**root**" interfaces: Collection **(java.util.Collection)** and Map **(java.util.Map)**. Terms that you will encounter while learning about the collection in Java:

- **Collection Framework:** Java's Collection Framework defines classes and interfaces for representing groups of objects as a single entity. C++ developers can compare the Collection framework with **STL (Standard Template Library)** and Container Framework with the Collection Framework if they come from a C++ background.

- **Collection Interface:** A class's interface specifies what it should do, not how. In other words, it is the blueprint for the class. This interface provides the most common methods for all collection objects that are part of the Collection Framework. Alternatively, it represents the individual object as a whole.

- **Collections Class:** A member of Collection Framework, it is part of java.util package. The collection object is provided with many utility methods in this class.

2. What is a Framework in Java?

Frameworks are sets of classes and interfaces that provide a ready-made architecture. It is not necessary to define a framework in order to implement new features or classes. As a result, an optimal object-oriented design includes a framework containing a collection of classes that all perform similar tasks. The framework can be used in a variety of ways, such as by calling its methods, extending it, and supplying "callbacks", listeners, and other implementations. Some of the popular frameworks in java are:

- Spring
- Hibernate
- Struts
- Google Web Toolkit (GWT)
- JavaServer Faces (JSF)

3. What is the difference between Array and Collection in Java?

Arrays are a collection of similar-typed variables with a common name in Java. There are some differences between arrays in Java and C/C++. On the other hand, Collections are groups of individual objects that form a single entity known as the collection of objects.

Arrays	Collection
Arrays are fixed in size that is once we create an array we can not increase or decrease based on our requirements.	The collection is growable in nature and is based on our requirements. We can increase or decrease of size.
With respect to memory, Arrays are not recommended for use.	With respect to memory, collections are recommended for use.
With respect to performance, Arrays are recommended for use.	With respect to performance, collections are not recommended for use.
Arrays can hold only homogeneous data types elements.	Collection can hold both homogeneous and heterogeneous elements.

For more information, refer to the article – [Difference Between Arrays and Collections in Java](#)

4. What are the various interfaces used in Java Collections Framework?

The collection is known as the root of the collection hierarchy. Collections represent groups of objects known as elements. The java platform does not provide any direct implementation of this interface but the Collection interface is being implemented by List and Set classes.

- **Collection interface**
- **List interface**
- **Set interface**
- **Queue interface**
- **Dequeue interface**
- **Map interface**

5. Explain the hierarchy of the Collection framework in Java.

All classes and interfaces required by the collection framework are contained in the utility package (java. util). Collection frameworks have an interface called an iterable interface, which allows the iterator to iterate over all collections. In addition to this interface, the main collection interface acts as a root for the collection framework. All the collections extend this collection interface thereby extending the properties of the iterator and the methods of this interface. The following figure illustrates the hierarchy of the collection framework.
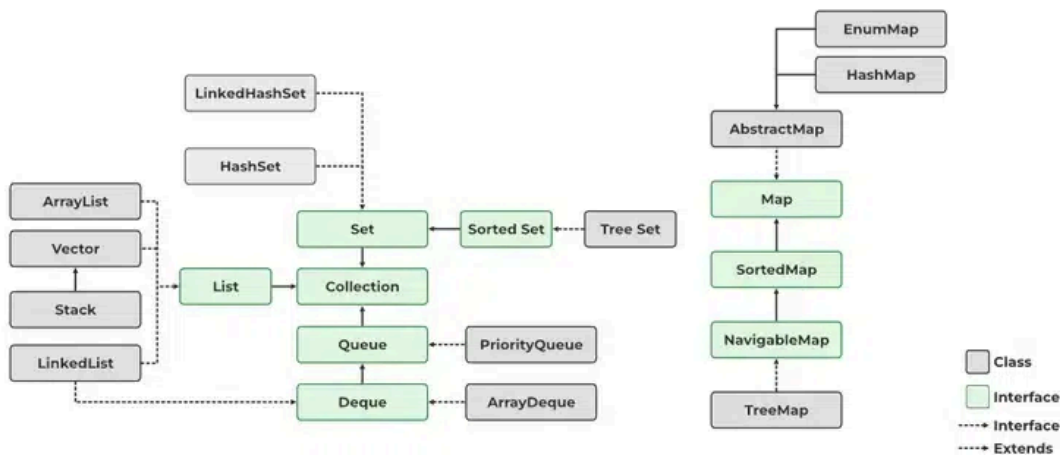
Java Collection Hierarchy

6. What are the advantages of the collection Framework?

Advantages of the Collection Framework: Since the lack of a collection framework gave rise to the above set of disadvantages, the following are the advantages of the collection framework.

- **Consistent API:** The API has a basic set of interfaces like *Collection*, *Set*, *List*, or *Map*, all the classes (ArrayList, LinkedList, Vector, etc) that implement these interfaces have *some* common set of methods.

- **Reduces programming effort:** A programmer doesn't have to worry about the design of the Collection but rather he can focus on its best use in his program. Therefore, the basic concept of Object-oriented programming (i.e.) abstraction has been successfully implemented.

- **Increases program speed and quality:** Increases performance by providing high-performance implementations of useful data structures and algorithms because in this case, the programmer need not think of the best implementation of a specific data structure. He can simply use the best implementation to drastically boost the performance of his algorithm/program.

7. What is ArrayList in Java?

ArrayList is a part of the Java collection framework and it is a class of java.util package. It provides us with dynamic arrays in Java. The main advantages of ArrayList are, if we declare an array then it's needed to mention the size but in ArrayList, it is not needed to mention the size of ArrayList if you want to mention the size then you can do it.

Image of Array List

> For more information, refer to the article – *ArrayList in Java*

8. What is the difference between Collection and Collections?

Collection	Collections
It is an interface.	It is a utility class.

Collection	Collections
It is used to represent a group of individual objects as a single unit.	It defines several utility methods that are used to operate on collection.
The Collection is an interface that contains a static method since java8. The Interface can also contain abstract and default methods.	It contains only static methods.

For more information, refer to the article – [Collection vs Collections in Java with Example](#)

9. Difference between ArrayList and LinkedList in the java collection framework?

ArrayList and LinkedList

ArrayList	LinkedList
This class uses a dynamic array to store the elements in it. With the introduction of generics, this class supports the storage of all types of objects.	This class uses a doubly linked list to store the elements in it. Similar to the ArrayList, this class also supports the storage of all types of objects.

ArrayList	LinkedList
Manipulating ArrayList takes more time due to the internal implementation. Whenever we remove an element, internally, the array is traversed and the memory bits are shifted.	Manipulating LinkedList takes less time compared to ArrayList because, in a doubly-linked list, there is no concept of shifting the memory bits. The list is traversed and the reference link is changed.
This class implements a List interface. Therefore, this acts as a list.	This class implements both the List interface and the Deque interface. Therefore, it can act as a list and a deque.
This class works better when the application demands storing the data and accessing it.	This class works better when the application demands manipulation of the stored data.

For more information, refer to the article – *ArrayList vs LinkedList in Java*

10. What is an iterator?

Java's Collection framework uses iterators to retrieve elements one by one. This iterator is universal since it can be used with any type of Collection object. Using Iterator, we can perform both reading and removing operations. This is an improved version of Enumeration with the addition of removing elements.

When enumerating elements in all Collection framework implemented interfaces, such as **Set**, **List**, **Queue**, **Deque**, and all implemented classes of Map, an Iterator must be used. The only cursor available for the entire collection framework is the iterator. Using the iterator() method in the Collection interface, you can create an iterator object.

Syntax:

```
Iterator itr = c.iterator();
```

Note: *Here "c" is any Collection object. itr is of type Iterator interface and refers to "c".*

For more information, refer to the article – *Iterators in Java*

11. What is the difference between an Iterator and an Enumeration?

A major difference between iterator and enumeration is that iterators have a remove() method while enumerations do not. Thus, using Iterator we can manipulate objects by adding and removing them from collections. Since enumeration can only traverse objects and fetch them, it behaves like a read-only interface.

For more information, refer to the article – *Difference between Iterator and Enumeration*

12. What is the difference between List and Set in Java

A major difference between a List and a Set is that a List can contain duplicate elements while a set contains only unique elements. The list is Ordered and maintains the order of the object to which they are added. The set is unordered.

List	Set
The List is an indexed sequence.	The Set is a non-indexed sequence.
The list allows duplicate elements	The set doesn't allow duplicate elements.
Elements by their position can be accessed.	Position access to elements is not allowed.
Multiple null elements can be stored.	Null elements can store only once.

List	Set
List implementations are ArrayList, LinkedList, Vector, Stack	Set implementations are HashSet, LinkedHashSet.

For more information, refer to the article – [*Difference Between List and Set in Java*](#)

13. What are the best practices for Java Collections Framework?

Following are some of the best practices while using Java Collections:

- Programs should be written as interfaces, not implementations, so we can modify the implementation later.
- Whenever possible, use Generics to ensure type safety and avoid ClassCastExceptions.
- Choosing the appropriate type of collection based on the need. For example, if the size is fixed, we might want to use an Array over an ArrayList. When iterating over the Map, we should use LinkedHashMap. Set is the best way to avoid duplicates.
- Use immutable classes provided by JDK as keys in Map to avoid implementation of hashCode() and equals().
- In order to increase the readability of the code, we should use isEmpty() instead of finding the size of the collection and comparing it to zero.
- Rather than writing your own implementation, use the Collections utility class to get read-only, Synchronized, or empty collections instead. It enhances code reuse while resulting in greater stability.

14. What is a priority queue in Java?

PriorityQueues are used to process objects according to their priority. Queues follow the First-In-First-Out algorithm, but sometimes the elements of the queue need to be processed according to their priority, which is where PriorityQueue comes into play. Priority queues are based on priority heaps.

The elements of the priority queue are ordered according to the natural ordering, or by a Comparator provided at queue construction time, depending on which constructor is used.

Priority Queue

Initial Queue = { }

Operation	Return Value	Queue Content
insert (C)		C
insert (O)		C O D
insert (D)		C O D
remove max	O	C D
insert (I)		C D I
insert (N)		C D I N
remove max	N	C D I
insert (G)		C D I G

Priority Queues in Java

Declaration:

```
public class PriorityQueue<E> extends AbstractQueue<E> implements
Serializable
where E is the type of elements held in this queue
```

The class implements Serializable, Iterable<E>, Collection<E>, and Queue<E> interfaces.

15. What is the difference between List, set, and map in java?

List	Set	Map
The list interface allows duplicate elements	The set does not allow duplicate elements.	The map does not allow duplicate elements
The list maintains insertion order.	The set does not maintain any insertion order.	The map also does not maintain any insertion order.

List	Set	Map
We can add any number of null values.	But in the set almost only one null value.	The map allows a single null key at most and any number of null values.
The list implementation classes are Array List and LinkedList.	Set implementation classes are HashSet, LinkedHashSet, and TreeSet.	Map implementation classes are HashMap, HashTable, TreeMap, ConcurrentHashMap, and LinkedHashMap.

For more information, refer to the article – [*Difference between List, Set, and Map in Java*](#)

16. What is the difference between Queue and Stack?

Stack	Queue
Stacks works on the LIFO principle, which means that the element inserted at the last will be the first element that will be taken out.	Queues work on the FIFO principle, which means that the element inserted first will be the first element that will be taken out.
In stacks, insertion, and deletions take place only from the top.	In queues, insertion occurs at the rear of the list and deletion takes place from the front of the list.
Insert operation is called push operation.	Insert operation is called enqueue operation.
Delete operation is called pop operation.	Delete operation is called dequeue operation.
The top of a stack always points to the last element in the list, which is	Two pointers are maintained for accessing queues. The front pointer points to the first

| Stack	Queue
the only pointer used to access the list.	inserted element, and the rear pointer points to the last inserted element.

17. What is BlockingQueue in Java?

The **BlockingQueue interface in Java** is added in Java 1.5 along with various other concurrent Utility classes like ConcurrentHashMap, Counting Semaphore, CopyOnWriteArrrayList, etc. The BlockingQueue interface supports flow control (in addition to queue) by introducing blocking if either BlockingQueue is full or empty.

A thread trying to enqueue an element in a full queue is blocked until some other thread makes space in the queue, either by dequeuing one or more elements or clearing the queue completely. Similarly, it blocks a thread trying to delete from an empty queue until some other threads insert an item. BlockingQueue does not accept a null value. If we try to enqueue the null item, then it throws NullPointerException.

Usage of BlockingQueue
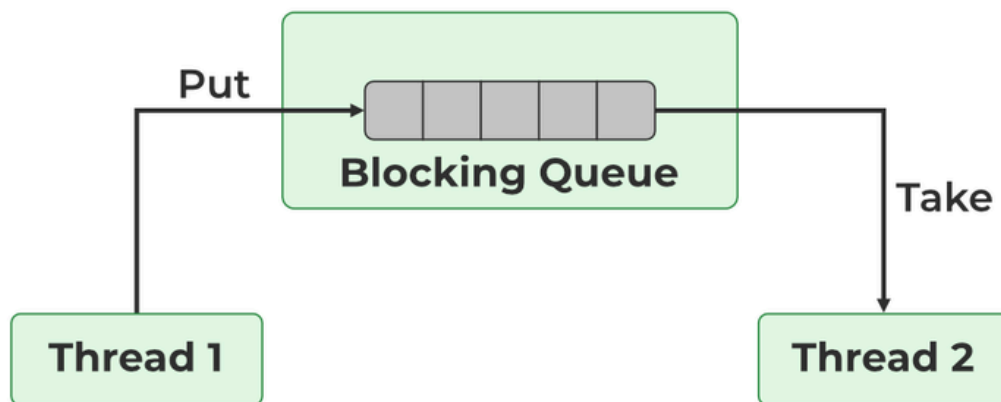
Blocking Queue in Java

The Hierarchy of BlockingQueue
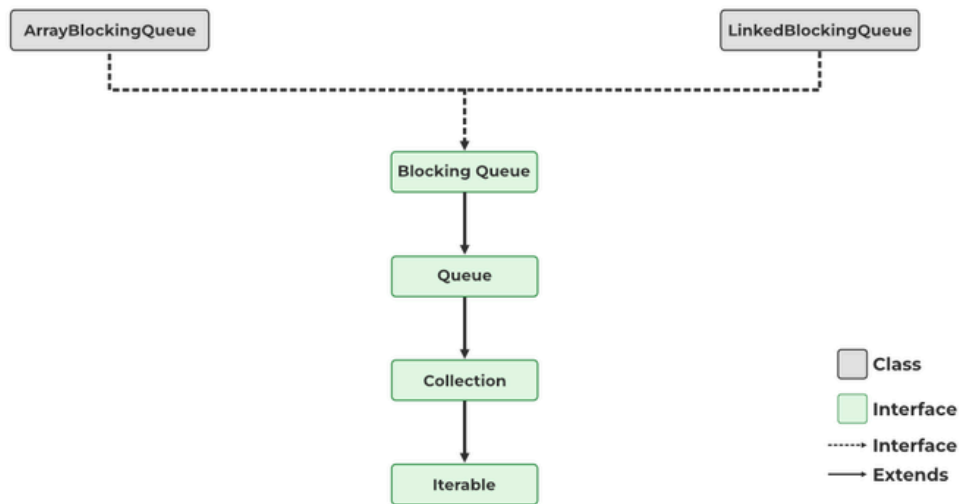
Hierarchy of Blocking Queue in Java

Declaration:

```
public interface BlockingQueue<E> extends Queue<E>
```

Here, **E** is the type of elements stored in the Collection.

> *For more information, refer to the article –* [*BlockingQueue Interface in Java*](#)
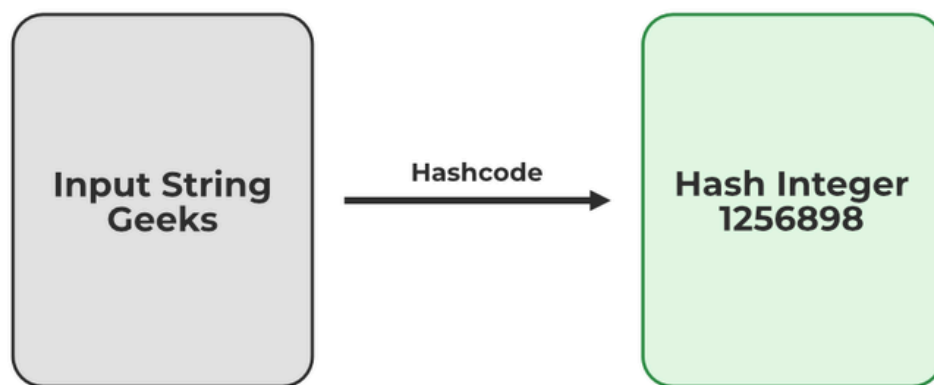
18. What is the hashCode()?

Image to demonstrate Java Hash Code

hashCode() method returns the hashcode value as an Integer. It is defined in the Java Object class which computes the hash values of given input objects. Hashcode value is mostly used in hashing-based collections like HashMap,

HashSet, HashTable....etc. This method must be overridden in every class which overrides the equals() method.

Syntax :

```
public int hashCode()
// This method returns the hash code value
// for the object on which this method is invoked.
```

For more information, refer to the article – equals() and hashCode() methods in Java

19. Distinguish between ArrayList and Vector in the Java Collection Framework.

In collection interviews, this question is frequently asked; however, Vector is synchronized whereas ArrayList is not. ArrayList is faster than Vector. ArrayList's Array size is increased by 50% when needed, while Vector's capacity is doubled whenever it is needed.

Array List vs Vector in java

ArrayList	Vector
ArrayList is not Synchronized	The vector is synchronized.

ArrayList	Vector
The size of ArrayList is incremented up to 50% of the current array size if the number of elements exceeds its capacity.	The size of Vector is incremented up to 100% of the current array size if the number of elements exceeds its capacity.
ArrayList is fast because it is non-Synchronized.	Vector is slower because it's synchronized.
The iterator interface is used to traverse the elements	An iterator interface or Enumeration can be used to traverse the vector.

For more information, refer to the article – *Vector vs ArrayList in Java*

20. Differentiate between Iterator and ListIterator.

Iterator	ListIterator
Can traverse elements present in Collection only in the forward direction.	Can traverse elements present in the Collection both in the forward and backward directions.
Helps to traverse Map, List, and Set.	Can only traverse List and not the other two.
Indexes cannot be obtained by using Iterator.	It has methods like nextIndex() and previousIndex() to obtain indexes of elements at any time while traversing the List.
Cannot modify or replace elements present in the Collection	We can modify or replace elements with the help of set(E e)

For more information, refer to the article – [*Difference Between an Iterator and ListIterator*](#)

21. What is the difference between an Iterator and an Enumeration?

Iterator: It is a universal iterator as we can apply it to any Collection object. By using an Iterator, we can perform both read and remove operations.

Syntax:

```
// Here "c" is any Collection object. itr is of
// type Iterator interface and refers to "c"
Iterator itr = c.iterator();
```

Enumeration: Enumeration (or enum) is a user-defined data type. It is mainly used to assign names to integral constants, the names make a program easy to read and maintain.

Syntax:

```
// A simple enum example where enum is declared
// outside any class (Note enum keyword instead of
// class keyword)
enum Color
{
    RED, GREEN, BLUE;
}
```

| Iterator	Enumeration
The iterator is a universal cursor as it is applicable to all the collection classes.	Enumeration is not a universal cursor as it applies only to legacy classes.
The iterator has the remove() method.	Enumeration does not have the remove() method.

Iterator	Enumeration
The iterator can do modifications (e.g using the remove() method which removes the element from the Collection during traversal).	The enumeration interface acts as a read-only interface, one can not do any modifications to the Collection while traversing the elements of the Collection.
Iterator is not a legacy interface. Iterator can be used for the traversal of HashMap, LinkedList, ArrayList, HashSet, TreeMap, and TreeSet.	Enumeration is a legacy interface that is used for traversing Vector, and Hashtable.

For more information, refer to the article – Difference between Iterator and Enumeration

22. What are the features of Java Hashmap?

HashMap is similar to HashTable, but it is unsynchronized. It allows us to store the null keys as well, but there should be only one null key object and there can be any number of null values. This class makes no guarantees as to the order of the map. To use this class and its methods, you need to import **java.util. HashMap** package or its superclass.
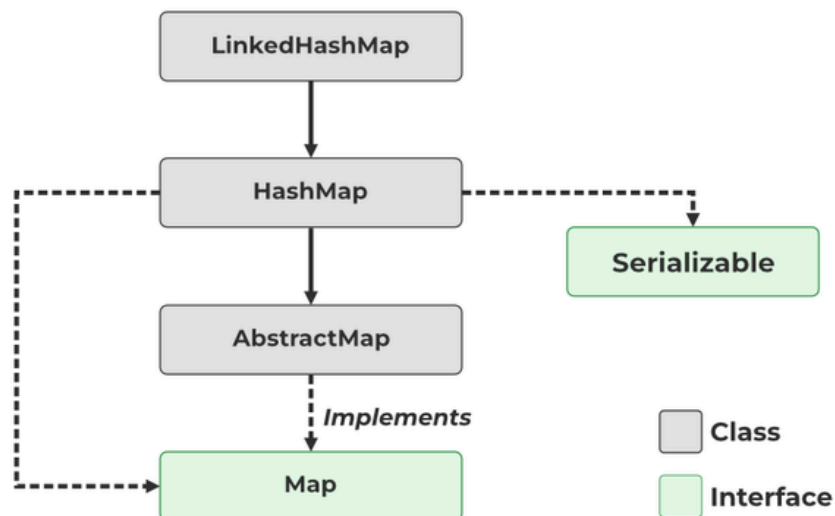
HashMap in Java

Syntax:

```
public class HashMap<K,V> extends AbstractMap<K,V> implements
Map<K,V>, Cloneable, Serializable
```

Parameters: It takes two parameters namely as follows:

- The type of keys maintained by this map (K)
- The type of mapped values (V)

> *For more information, refer to the article – [HashMap in Java with Examples](#)*
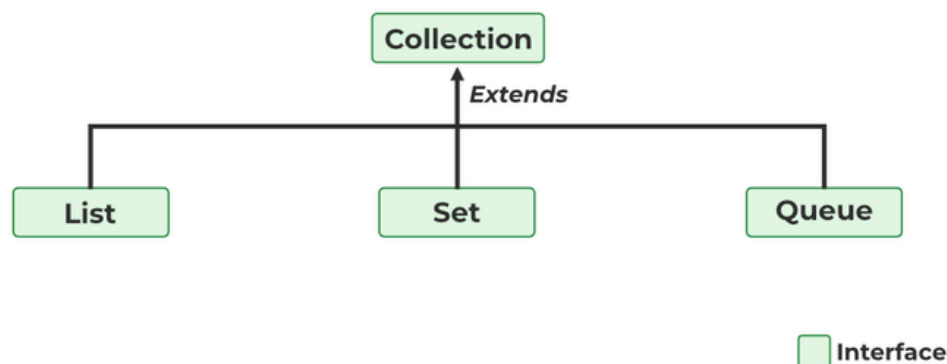
23. What are Collection Interfaces?

The **Collection** interface is a member of the Java Collections Framework. It is a part of **java.util** package. It is one of the root interfaces of the Collection Hierarchy. The Collection interface is not directly implemented by any class. However, it is implemented indirectly via its subtypes or subinterfaces like List, Queue, and Set.

For Example, the HashSet class implements the Set interface which is a subinterface of the Collection interface. If a collection implementation doesn't implement a particular operation, it should define the corresponding method to throw UnsupportedOperationException.

The Hierarchy of Collection:

Collection Interface in Java

24. Explain the list interface.

In Java, the List interface allows the user to store an ordered collection of objects. The list is the child interface of Collection. In Collection, a list is an ordered collection of objects which can have duplicate values. Since List preserves the insertion order, it allows positional access and insertion, which also allows duplicate values.
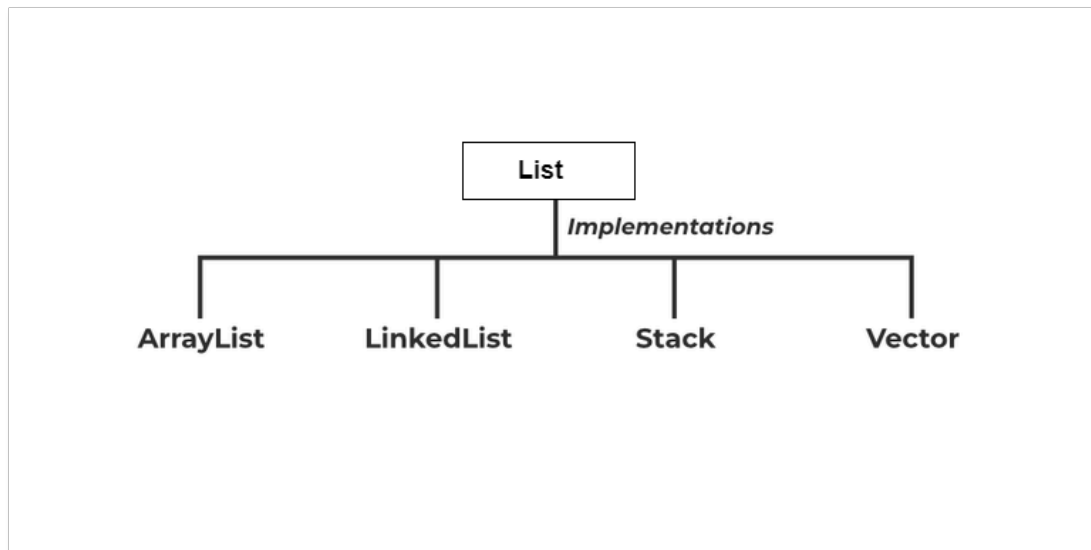
Syntax:

```
public interface List<E> extends Collection<E> ;
```

This list interface is implemented by various classes like ArrayList, Vector, Stack, etc. Since all the subclasses implement the list, we can instantiate a list object with any of these classes.

Example:

```
List <T> al = new ArrayList<> ();
List <T> ll = new LinkedList<> ();
List <T> v = new Vector<> ();

Where T is the type of the object
```

Array List in Java

The classes which implement the List interface are as follows:

- ArrayList
- LinkedList
- Vector
- Stack

25. Write a program to convert a given array into a collection with the asList() method.

To convert array-based data into Collection based we can use **java.util.Arrays** class. This class provides a static method asList(T... a) that converts the array into a Collection.

```java
// Convert an Array into Collection in Java
// import java util library
import java.util.*;

// class for writing logic of the problem
public class ArrayToCollection {
    public static void main(String args[])
    {
        // array input
        String students[] = { "Kamlesh", "Abhay",
                              "Abhishek", "Shivansh" };

        // printing input elements for comparison
        System.out.println("Array input: "
```

```
                    + Arrays.toString(students));

        // converting array into Collection
        // with asList() function
        List studentList = Arrays.asList(students);

        // print converted elements
        System.out.println("Converted elements: "
                            + studentList);
    }
}
```

Output

```
Array input: [Kamlesh, Abhay, Abhishek, Shivansh]
Converted elements: [Kamlesh, Abhay, Abhishek, Shivansh]
```

26. Differentiate between HashSet and HashMap

HashSet	HashMap
HashSet implements the Set interface	HashMap implements the Map interface
No Duplicates are allowed	Yes duplicates values are allowed but no duplicate key is allowed
Dummy values are allowed in HashSet.	No Dummy values are allowed in HashMap.
A single Object is required during an add operation	2 Objects are required during an add operation
Speed is comparatively slower than HashMap	Speed is comparatively faster than HashSet because of hashing technique has been used here.

HashSet	HashMap
Have a single null value	Single null key and any number of null values
Add() method is used for the insertion	The put () method is used for insertion.

For more information, refer to the article – [Difference between HashMap and HashSet](#)

27. Differentiate between HashSet and HashTable.

HashSet	HashTable
HashSet allows NULL Elements	HashTable does not allow NULL Elements.
Objects that you insert in HashSet are not guaranteed to be inserted in the same order. Objects are inserted based on their hash code. LinkedHashSet can be used to maintain order.	HashTable does not maintain insertion order.
HashSet is not Synchronized but it can be synchronized externally.	HashTable is Synchronized.
add() method is used to insert into HashSet	put() method is used to insert into HashTable

28. What is the default size of the load factor in the hashing-based collection?

As the Load Factor increases, the capacity increases so that the operational complexity of the HashMap remains O(1) if the ratio of the current element to the initial capacity crosses the threshold. The meaning of operational complexity of O(1) means the retrieval and insertion operations take constant time. The default load factor size is **0.75**. The default capacity is calculated by multiplying the initial capacity by the load factor.

> For more information, refer to the article – _Load Factor in HashMap in Java with Examples_

Java Collection Interview Questions For Experienced

29. What is the difference between Comparable and Comparator in Java?

Java provides two interfaces to sort objects using data members of the class:

- Comparable
- Comparator

Comparable	Comparator
The Comparable interface provides a single sorting sequence.	The Comparator interface provides multiple sorting sequences.

Comparable	Comparator
The actual class is modified by a comparable interface	The actual class is not modified by the Comparator interface.
compareTo() method is used to sort elements.	compare() method is used to sort elements.
Comparable is present in the package java.lang	Comparator is present in the package java.util

For more information, refer to the article – *Comparable vs Comparator in Java*

30. What is the difference between fail-fast and failsafe?

Iterators in Java are used to iterate over the Collection objects. Fail-Fast iterators immediately throw *ConcurrentModificationException* if there is a **structural modification** of the collection. Structural modification means adding, or removing any element from a collection while a thread is iterating over that collection. Iterator on ArrayList and HashMap classes are some examples of fail-fast Iterator.

Fail-Fast	Fail-Safe
ConcurrentModificationException is thrown while modifying the object during the iteration process.	No Exception is thrown
Fail-Fast needs less memory during the process.	Fail-Safe iterator requires more memory during the process.

Fail-Fast	Fail-Safe
A clone Object is not created during the iteration process.	A clone Object or a Copy is created during the iteration process.
Fail-Fast does not allow modification during the process of iteration.	Fail-Safe allows modification during the process of iteration.
Fail-Fast is fast,	Fail-Safe is slightly slower than fail fast.
Examples: ArrayList, Vector, HashMap, HashSet, etc.	**Examples:** ConcurrentHashMap, CopyOnWriteArrayList, etc.

For more information, refer to the article – *Fail Fast and Fail Safe Iterators in Java*

31. Write a program to iterate the list using the lambda expression.

Iteration can be done using a lambda expression.

Syntax:

```
list_name.forEach(variable->{//block of code})
```

```java
// Java Program to iterate over a List
// using forEach()

// Importing all classes of
// java.util method
import java.util.*;

// Class
class GFG {

    // Main driver method
```

```java
    public static void main(String args[])
    {
        // Creating an ArrayList
        List<String> l = new ArrayList<String>();

        // Adding elements to the List
        // Custom inputs
        l.add("Geeks");
        l.add("for");
        l.add("Geeks");

        // Lambda expression printing all elements in a List
        l.forEach((temp) -> { System.out.println(temp); });
    }
}
```

Output

```
Geeks
for
Geeks
```

For more information, refer to the article – [Iterate through List in Java](#)

32. What is IdentityHashMap?

The IdentityHashMap implements the Map interface using Hashtable, comparing keys (and values) using reference equality instead of object equality. This class implements the Map interface, but it intentionally breaks Map's general contract, which demands that objects are compared using the equals() method. This class is used when the user allows objects to be compared using references. It belongs to java.util package.

For more information, refer to the article – [IdentityHashMap class in Java](#)

33. Write a program in Java to display the contents of a HashTable using enumeration.

The hashtable class implements a hash table, which maps keys to values. Any non-null object can be used as a key or as a value. To successfully store and retrieve objects from a hashtable, the objects used as keys must implement the hashCode method and the equals method. Below is the program to display the contents of a HashTable using enumeration:

```java
// Java Program to Demonstrate Getting Values
// as an Enumeration of Hashtable class

import java.io.*;
import java.util.Enumeration;
import java.util.Hashtable;

// Main class
// EnumerationOnKeys
public class GFG {

    // Main driver method
    public static void main(String[] args)
    {
        // Creating an empty hashtable
        Hashtable<Integer, String> hash
            = new Hashtable<Integer, String>();

        // Inserting key-value pairs into hash table
        // using put() method
        hash.put(1, "Geeks");
        hash.put(2, "for");
        hash.put(3, "Geeks");

        // Now creating an Enumeration object
        //  to read elements
        Enumeration e = hash.elements();

        // Condition holds true till there is
        // single key remaining

        // Printing elements of hashtable
        // using enumeration
        while (e.hasMoreElements()) {

            // Printing the current element
```

```
                System.out.println(e.nextElement());
            }
        }
    }
```

Output

```
Geeks
for
Geeks
```

34. Write a program in java to get the collection view of the values present in a HashMap.

Java's HashMap class has the java.util.HashMap.values() method for creating collections out of HashMap values. It basically returns a Collection view of HashMap values.

```java
// Java code to illustrate the values() method
import java.util.*;

public class Hash_Map_Demo {
    public static void main(String[] args)
    {

        // Creating an empty HashMap
        HashMap<Integer, String> hash_map
            = new HashMap<Integer, String>();

        // Mapping string values to int keys
        hash_map.put(0, "Welcome");
        hash_map.put(1, "to");
        hash_map.put(2, "Geeks");
        hash_map.put(3, "4");
        hash_map.put(4, "Geeks");

        // Displaying the HashMap
        System.out.println("Initial Mappings are: "
                            + hash_map);
```

```
        // Using values() to get the set view of values
        System.out.println("The collection is: "
                                + hash_map.values());
    }
}
```

Output

```
Initial Mappings are: {0=Welcome, 1=to, 2=Geeks, 3=4, 4=Geeks}
The collection is: [Welcome, to, Geeks, 4, Geeks]
```

For more information, refer to the article – [HashMap values() Method in Java](#)

35. Write a program to join two ArrayList into one single ArrayList.

Given two ArrayLists in Java, our task is to join these ArrayLists.

```java
// Java program to demonstrate
// How to join ArrayList

import java.util.*;

public class GFG {
    public static void main(String args[])
    {

        ArrayList<String> list_1 = new ArrayList<String>();

        list_1.add("Geeks");
        list_1.add("For");
        list_1.add("ForGeeks");

        // Print the ArrayList 1
        System.out.println("ArrayList 1: " + list_1);

        ArrayList<String> list_2 = new ArrayList<String>();

        list_2.add("GeeksForGeeks");
        list_2.add("A computer portal");

        // Displaying the ArrayList 2
        System.out.println("ArrayList 2: " + list_2);
```

```java
        // using Collection.addAll() method to join two
        // arraylist
        list_1.addAll(list_2);

        // Print the joined ArrayList
        System.out.println("Joined ArrayLists: " + list_1);
    }
}
```

Output

```
ArrayList 1: [Geeks, For, ForGeeks]
ArrayList 2: [GeeksForGeeks, A computer portal]
Joined ArrayLists: [Geeks, For, ForGeeks, GeeksForGeeks, A
computer portal]
```

For more information, refer to the article – [Join two ArrayLists in Java](#)

36. How can you synchronize an ArrayList in Java?

Using the Collections.synchronizedList() method, we can synchronize our collections in Java. SynchronizedList() returns a synchronized (thread-safe) list backed by a selection.

```java
// Java program to show synchronization of ArrayList
import java.io.*;
import java.util.*;

class GFG {
    public static void main(String[] args)
    {
        // Non Synchronized ArrayList
        List<String> list = new ArrayList<String>();

        list.add("Eat");
        list.add("Coffee");
        list.add("Code");
        list.add("Sleep");
        list.add("Repeat");
```

```java
        // Synchronizing ArrayList in Java
        list = Collections.synchronizedList(list);

        // we must use synchronize block to avoid
        // non-deterministic behavior
        synchronized (list)
        {
            Iterator<String> it = list.iterator();
            while (it.hasNext()) {
                System.out.println(it.next());
            }
        }
    }
}
```

Output

```
Eat
Coffee
Code
Sleep
Repeat
```

37. What is a Properties Class in Java?

The properties class is a subclass of Hashtable. The properties class stores a list of values whose key is a string and whose value is also a string. Properties can define other properties class lists, but the default is properties.

Features of Properties class:

- Property is a subclass of Hashtable.
- The Properties file is used to store and retrieve string data type for a list of values where the key is a string and the value is also a string.
- If the original properties list does not contain a certain key property, the default properties list will be searched instead.
- Objects can be shared by multiple threads without external synchronization.
- The properties class can be used to retrieve the properties of the system.

For more information, refer to the article – [*Properties Class in Java*](#)

38. What will happen if you use HashMap in a multithreaded Java application?

In a multi-threaded environment, if multiple threads alter the map structurally, such as adding, removing, or modifying mappings, the internal data structure of HashMap may become corrupted and there may be some missing links, incorrect entries, and the map itself may become completely useless. Thus, you should not use HashMap in a concurrent application; instead, use ConcurrentHashMap or Hashtable which is thread-safe. The ConcurrentHashMap includes all the Hashtable's methods as well as full concurrency of retrievals and updates.

How did ThreadSafeConcurrentHashMap become thread-safe?

- java.util.Concurrent.ConcurrentHashMap class provides thread safety by dividing the map into segments, which allows the lock to be taken only once per segment, i.e, once for each thread.
- The read operation in ConcurrentHashMap does not require a lock.

For more information, refer to the article – [*How Does ConcurrentHashMap Achieve Thread-Safety in Java?*](#)

39. What will happen if two different keys of HashMap return the same hashcode()?

When two different keys of HashMap return the same hash code, they will end up in the same bucket; therefore, collisions will occur. n case of collision, i.e. index of two or more nodes is the same, nodes are joined by a link list i.e. the second node is referenced by the first node and the third by the second, and so on.

For more information, refer to the article – [*Internal Working of HashMap in Java*](#)

40. What is WeakHashMap?

WeakHashMap implements the Map interface. Unlike HashMap, WeakHashMap allows garbage collection even if the object specified as the key doesn't contain any references despite being associated with WeakHashMap. In other words, Garbage Collector is better than WeakHashMap.

> *For more information, refer to the article –* [*Hashmap vs WeakHashMap in Java*](#)

41. What is UnsupportedOperationException?

In the context of APIs or list implementations, the UnsupportedOperationException is a common exception. The exception is thrown when the requested operation cannot be performed. This class is a member of the Java Collections Framework.

Syntax:

```
public class UnsupportedOperationException
extends RuntimeException
```

> *For more information, refer to the article –*
> [*UnsupportedOperationException*](#)

42. How to make a Collection Read-Only in Java?

Creating a Read-Only Collection involves restricting the object to only fetching the data and not adding or removing data. Java has different methods for different Collection types like unmodifiableCollection(), unmodifiableMap(), ununmodifiableSet(), etc. java.util.The collections class defines all methods. The unmodifiableCollection() method creates a Read-Only collection. It requires a reference to the Collection class. If we have an object of Set Interface, we can use **ununmodifiableSet()** to make Read-Only.

> *For more information, refer to the article –* [*How to Make a Collection Read-Only in Java?*](#)

43. Difference between PriorityQueue and TreeSet in Java?

PriorityQueue	TreeSet
PriorityQueue comes in JDK 1.5.	TreeSet comes in JDK 1.4.
The data structure used by PriorityQueue is Queue	The data structure used by TreeSet is Set.
Duplicate elements are allowed.	Duplicate elements are not allowed.
Except for the root element, the rest of the elements do not follow any particular order in PriorityQueue.	In TreeSet all the elements remain in the sorted order.
Using PriorityQueue, we can retrieve the largest or smallest element in O(1) time.	TreeSet doesn't provide a way to retrieve the largest or smallest element in O(1) time, but since they are in sorted order it gets the first or last element in O(1) time.

For more information, refer to the article – [Difference Between PriorityQueue and TreeSet](#)

44. What is the diamond operator in Java?

Diamond operators are used for simplifying the use of generics when creating objects while avoiding unchecked warnings in a program. When the Diamond operator was introduced in Java 7, we can create the object without mentioning the generic type on the right side of the expression as shown below.

Syntax:

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

For more information, refer to the article – [Diamond operator](#)

45. How TreeMap works in Java?

TreeMap stores the key-value pairs, but TreeMap sorts the keys ascending rather than descending like HashMap. Depending on which constructor is used, TreeMap will be sorted either based on its keys, or by a Comparator. In TreeMap, the elements are sorted based on a Red-Black tree. A red-black tree is a self-balancing binary search tree where each node has an extra bit, and that bit is often interpreted as the color (red or black). These colors are used to ensure that the tree remains balanced during insertions and deletions.

Structure of a Node in Java

> For more information, refer to the article – Internal Working of TreeMap in Java

## 46. List down ways to iterate over Map in java?

The HashMap class provides Java's Map interface by storing data in (Key, Value) pairs and accessing them by an index of another type. To use this class it is necessary to import *java.util.HashMap* package or its superclass.

There are numerous ways to iterate over HashMap of which 5 are listed below:

1. Iterate through a HashMap EntrySet using Iterators.
2. Iterate through HashMap KeySet using Iterator.
3. Iterate HashMap using for-each loop.
4. Iterating through a HashMap using Lambda Expressions.
5. Loop through a HashMap using Stream API.

For more information, refer to the article – How to Iterate HashMap in Java

47. What is CopyOnWriteArrayList in Java?
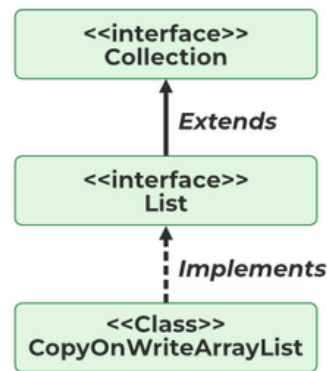
CopyOnWriteArrayList in Java

JDK 1.5 introduced an enhanced version of ArrayList called CopyOnWriteArrayList, where all modifications (add, set, remove, etc) are implemented by a new copy. It can be found in java.util.concurrent. It is a data structure created to be used in a concurrent environment. In a Thread-based environment, the CopyOnWriteArrayList is intended for frequent reading and infrequent updating. CopyOnWriteArrayList is a thread-safe version of ArrayList.

For more information, refer to the article – CopyOnWriteArrayList in Java

48. What is EnumMap in Java?

EnumMap is an implementation of the Map interface specific to enumeration types. EnumMap class is a member of the Java Collections Framework & is not synchronized. It extends AbstractMap and implements the Map interface in java. EnumMap belongs to java.util package.
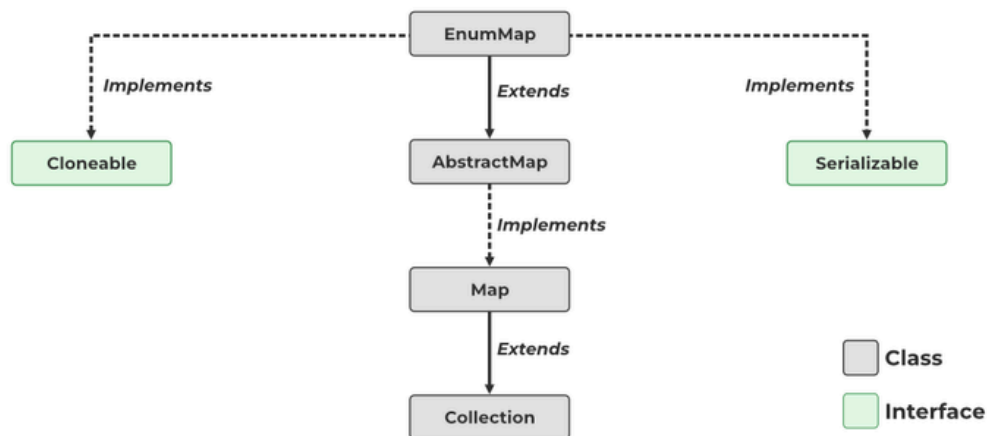
Syntax:

```
public class EnumMap<K extends Enum<K>,V> extends AbstractMap<K, V> implements Serializable, Cloneable
```

*// K must extend Enum, which enforces the requirement that the keys
must be of the specified enum type.*

Parameters:

- Key object type
- Value object type

EnumMap in Jav

For more information, refer to the article – EnumMap class in Java

49. How does Hashmap internally Works?

HashMap works on the principle of Hashing. HashMap contains an array of
Node and Node can represent a class having the following objects:

- int hash
- K key
- V value
- Node next

The internal workings of HashMap:

- Hashing
- Buckets
- Index Calculation in Hashmap

> For more information, refer to the article – *Internal Working of HashMap in Java*

50. Why iterator in hashmap is considered fail-fast?

fail-fast iterators immediately throw concurrent modification exceptions if any thread from outside attempts to modify the collection on which they are iterating. The fail-fast feature ensures that the iterator fails immediately if it detects that any modification of the collection will lead to anomalous behavior in the future.

> *Fail fast* feature ensures that if iterator feels that modification of collection would result in anomalous behaviour at any point of time in future, it fails immediately.

Example:

```java
// Java code to demonstrate remove
// case in Fail-fast iterators

import java.io.*;
import java.util.ArrayList;
import java.util.Iterator;

public class GFG {
    public static void main(String[] args)
    {
        ArrayList<Integer> arr = new ArrayList<>();
        arr.add(1);
        arr.add(2);
        arr.add(3);
        arr.add(4);
        arr.add(5);

        Iterator<Integer> it = arr.iterator();
        while (it.hasNext()) {
            if (it.next() == 2) {
                // will not throw Exception
                it.remove();
            }
        }
    }
```

```java
        System.out.println(arr);

        it = arr.iterator();
        while (it.hasNext()) {
            if (it.next() == 3) {
                // will throw Exception on
                // next call of next() method
                arr.remove(3);
            }
        }
    }
}
```

Output:

```
[1, 3, 4, 5]
Exception in thread "main"
java.util.ConcurrentModificationException
    at
java.util.ArrayList$Itr.checkForComodification(ArrayList.java:901)
    at java.util.ArrayList$Itr.next(ArrayList.java:851)
    at FailFastExample.main(FailFastExample.java:28)
```

Conclusion

Java Collections is important to understand for Java developers or programmers because Java is widely used in various industries. It's important for developers to have a solid understanding of core concepts of Java Collections. Java is one of the most widely used languages in top companies such as Uber, Airbnb, Google, Netflix, Instagram, Spotify, Amazon, etc. To get into these companies or any other IT companies, you need to master these mostly asked Java Collections interview questions in order to crack their Java-based online assessment and technical interview.

> *If you're looking to practice coding questions, the JAVA Collection Programs may be a helpful resource.*

Comment | More info ⌄ | Advertise with us

Similar Reads

30 OOPs Interview Questions and Answers [2025 Updated]

Object-oriented programming, or OOPs, is a programming paradigm that implements the concept of objects in the program. It aims to provide an easier solution to real-world problems by implementing real-world entitie...

🕐 15 min read

C++ Interview Questions and Answers (2025)

C++ - the must-known and all-time favourite programming language of coders. It is still relevant as it was in the mid-80s. As a general-purpose and object-oriented programming language is extensively employed...

🕐 15+ min read

Top 100 C++ Coding Interview Questions and Answers [2025 Updated]

C++ is one of the most popular languages in the software industry for developing software ranging from operating systems, and DBMS to games. That is why it is also popular to be asked to write C++ Programs in...

🕐 15+ min read

Python Interview Questions and Answers

Python is the most used language in top companies such as Intel, IBM, NASA, Pixar, Netflix, Facebook, JP Morgan Chase, Spotify and many more because of its simplicity and powerful libraries. To crack their Online...

🕐 15+ min read

Java Interview Questions and Answers

Java is one of the most popular programming languages in the world, known for its versatility, portability, and wide range of applications. Java is the most used language in top companies such as Uber, Airbnb, Google,...

🕐 15+ min read

Java Collections Interview Questions and Answers

Java Collection Framework was introduced in JDK 1.2 which contains all the collection classes and interfaces. Java Collection is a framework that provides a mechanism to store and manipulate the collection of objects. It...

🕐 15+ min read

Java Multithreading Interview Questions and Answers

Java Multithreading lets developers run multiple tasks at the same time, making apps faster and more responsive. Java is used by over 10 million developers on 15 billion devices, from Big Data apps to everyday...

🕐 12 min read

Top 100 Data Structure and Algorithms DSA Interview Questions Topic-wise

DSA has been one of the most popular go-to topics for any interview, be it college placements, software developer roles, or any other technical roles for freshers and experienced to land a decent job. If you are...

🕐 3 min read

Top 50 Array Coding Problems for Interviews

Array is one of the most widely used data structure and is frequently asked in coding interviews to the problem solving skills. The following list of 50 array coding problems covers a range of difficulty levels, from...

🕐 2 min read

Most Asked Problems in Data Structures and Algorithms | Beginner DSA Sheet

In this Beginner DSA Sheet for Data Structures and Algorithms, we have curated a selective list of problems for you to solve as a beginner for DSA. After learning the fundamentals of programming, choosing a...

🕐 2 min read

GeeksforGeeks
Sanchhaya Education Private Limited

📍 **Corporate & Communications Address:**

A-143, 7th Floor, Sovereign Corporate Tower, Sector- 136, Noida, Uttar Pradesh (201305)

📍 **Registered Address:**

K 061, Tower K, Gulshan Vivante Apartment, Sector 137, Noida, Gautam Buddh Nagar, Uttar Pradesh, 201305

GET IT ON Google Play   Download on the App Store

Advertise with us

Company	Languages
About Us	Python
Legal	Java
Privacy Policy	C++
In Media	PHP
Contact Us	GoLang
Advertise with us	SQL

GFG Corporate Solution

Placement Training Program

R Language

Android Tutorial

Tutorials Archive

DSA

Data Structures

Algorithms

DSA for Beginners

Basic DSA Problems

DSA Roadmap

Top 100 DSA Interview Problems

DSA Roadmap by Sandeep Jain

All Cheat Sheets

Data Science & ML

Data Science With Python

Data Science For Beginner

Machine Learning

ML Maths

Data Visualisation

Pandas

NumPy

NLP

Deep Learning

Web Technologies

HTML

CSS

JavaScript

TypeScript

ReactJS

NextJS

Bootstrap

Web Design

Python Tutorial

Python Programming Examples

Python Projects

Python Tkinter

Python Web Scraping

OpenCV Tutorial

Python Interview Question

Django

Computer Science

Operating Systems

Computer Network

Database Management System

Software Engineering

Digital Logic Design

Engineering Maths

Software Development

Software Testing

DevOps

Git

Linux

AWS

Docker

Kubernetes

Azure

GCP

DevOps Roadmap

System Design

High Level Design

Low Level Design

UML Diagrams

Interview Guide

Design Patterns

OOAD

System Design Bootcamp

Interview Questions

Inteview Preparation

Competitive Programming

Top DS or Algo for CP

Company-Wise Recruitment Process

Company-Wise Preparation

Aptitude Preparation

Puzzles

School Subjects

Mathematics

Physics

Chemistry

GeeksforGeeks Videos

DSA

Python

Java

Biology

Social Science

English Grammar

Commerce

World GK

C++

Web Development

Data Science

CS Subjects