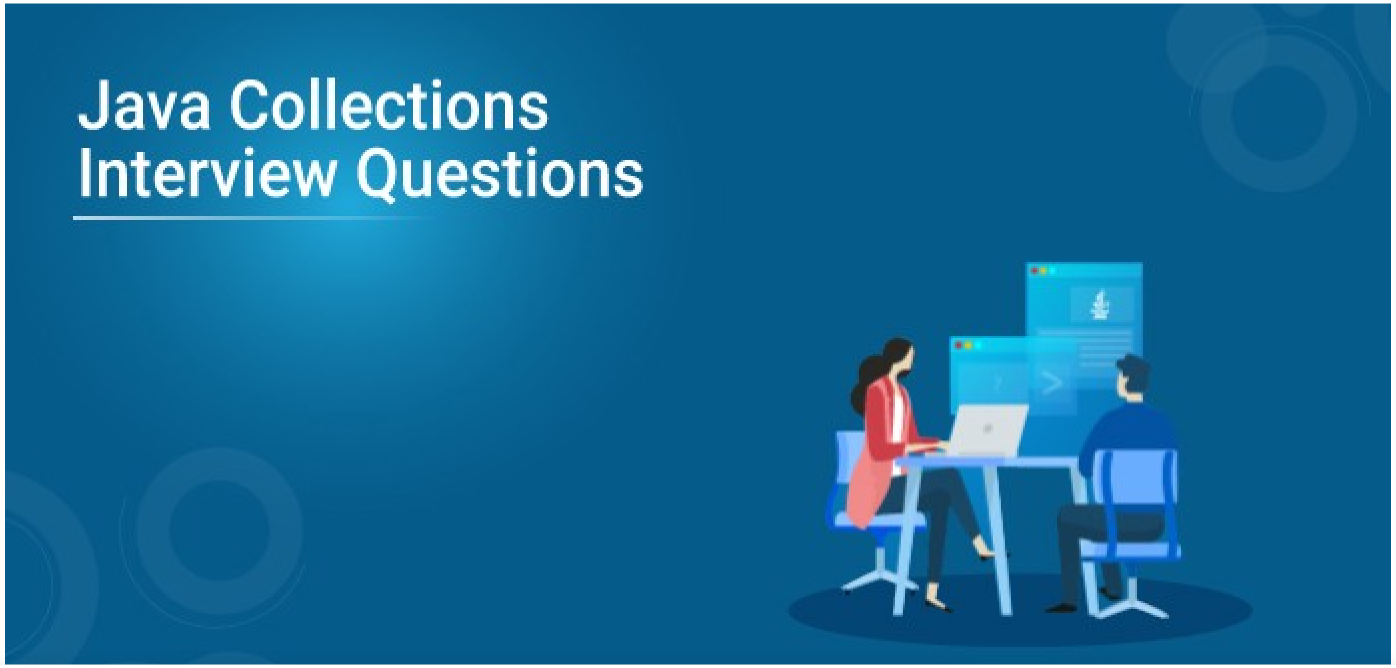
Published in Edureka

Swatee Chand Follow
Jul 14, 2020 · 14 min read · Listen

Top 50 Java Collections Interview Questions You Need to Know



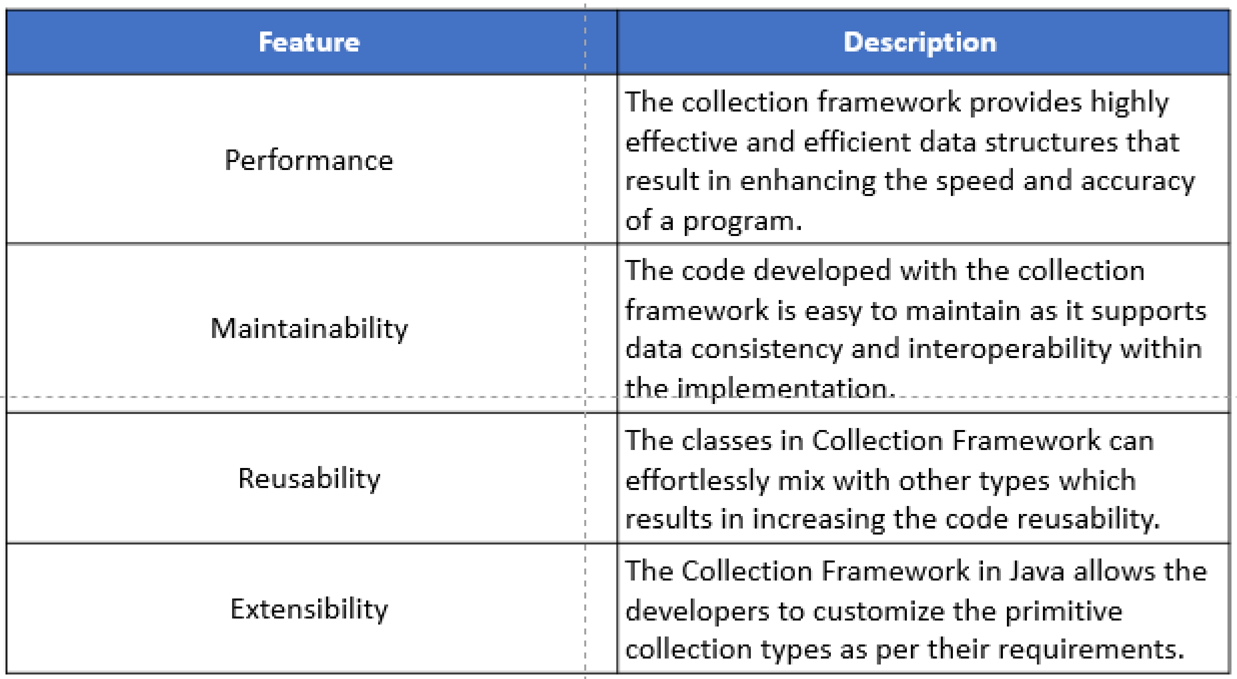
Collection Framework is one of the most important pillars that support the fundamental concepts of the Java programming language. If you are an aspiring Java Developer, it is very important for you to have a strong knowledge of these core concepts before you appear for an interview. Through the medium of this article, I will share the *Top 50 Java Collections Interview Questions and Answers* that will definitely help you in clearing your interview with flying colors.

The questions in this article have been divided into the following sections:

- Generic
- List
- Queue
- Set
- Map
- Differences

Generic — Java Collections Interview Questions

Below table contains the major advantages of the Java Collection Framework:



Feature	Description
Performance	The collection framework provides highly effective and efficient data structures that result in enhancing the speed and accuracy of a program.
Maintainability	The code developed with the collection framework is easy to maintain as it supports data consistency and interoperability within the implementation.
Reusability	The classes in Collection Framework can effortlessly mix with other types which results in increasing the code reusability.
Extensibility	The Collection Framework in Java allows the developers to customize the primitive collection types as per their requirements.

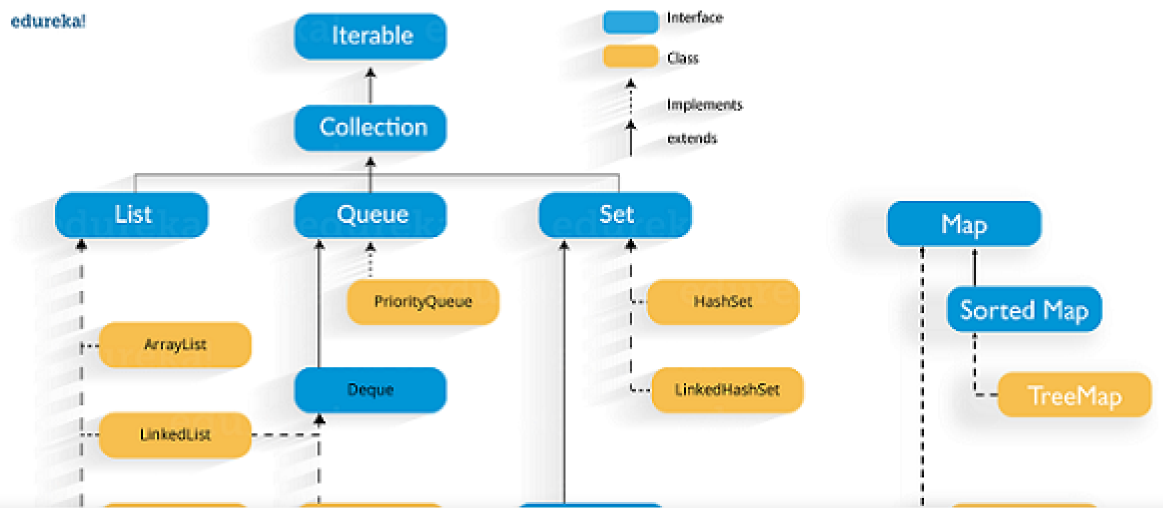
2. What do you understand by Collection Framework in Java?

The Java Collection framework provides an architecture to store and manage a group of objects. It permits the developers to access prepackaged data structures

- Interfaces
- Classes
- Algorithm

All these classes and interfaces support various operations such as Searching, Sorting, Insertion, Manipulation, and Deletion which makes the data manipulation really easy and quick.

3. Describe the Collection hierarchy in Java.



4. List down the primary interfaces provided by Java Collections Framework?

Below are the major interfaces provided by the Collection Framework:

- ***Collection Interface*:** java.util.The collection is the root of the Java Collection framework and most of the collections in Java are inherited from this interface.

```
public interface Collection<E>extends Iterable
```

- ***List Interface*:** java.util.List is an extended form of an array that contains ordered elements and may include duplicates. It supports the index-based search, but elements can be easily inserted irrespective of the position. The List interface is implemented by various classes such as ArrayList, LinkedList, Vector, etc.

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

index-based search is not supported. It is majorly used as a mathematical set abstraction model. The Set interface is implemented by various classes such as HashSet, TreeSetand LinkedHashSet.

```
public interface Set<E> extends Collection<E>
```

- ***Queue Interface***: java.util.Queue in Java follows a FIFO approach i.e. it orders the elements in First In First Out manner. Elements in Queue will be
- ***Map Interface***: java.util.Map is a two-dimensional data structure in Java that is used to store the data in the form of a Key-Value pair. The key here is the unique hashcode and value represent the element. Map in Java is another form of the Java Set but can't contain duplicate elements.

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

The Collection interface in Java specifies a group of objects called elements. The maintainability and ordering of elements are completely dependent on the concrete implementations provided by each of the Collection. Thus, there is no use of extending the Cloneable and Serializable interfaces.

Below are the main advantages of using the generic collection in Java:

- Provides stronger type checks at the time of compilation
- Eliminates the need for typecasting
- Enables the implementation of generic algorithms which makes the code customizable, type-safe and easier to read

7. What is the main benefit of using the Properties file?

The main advantage of using the properties file in Java is that in case the values in the properties file are changed it will be automatically reflected without having to recompile the java class. Thus it is mainly used to store information that is liable to change such as username and passwords. This makes the management of the application easy and efficient. Below is an example of the same:

```
import java.util.*;
import java.io.*;
public class PropertiesDemo{
public static void main(String[] args)throws Exception{
    }
    }
```

8. What do you understand by the Iterator in the Java Collection Framework?

Iterator in Java is an interface of the Collection framework present in java.util package. It is a Cursor in Java which is used to iterate a collection of objects. Below are a few other major functionalities provided by the Iterator interface:

- Traverse a collection object elements one by one
- Known as Universal Java Cursor as it is applicable for all the classes of the Collection framework
- Supports READ and REMOVE Operations.
- Iterator method names are easy to implement

9. What is the need for overriding equals() method in Java?

The initial implementation of the equals method helps in checking whether two objects are the same or not. But in case you want to compare the objects based on

Sorting in Java Collections is implemented via Comparable and Comparator interfaces. When Collections.sort() method is used the elements get sorted based on the natural order that is specified in the compareTo() method. On the other hand when Collections.sort(Comparator) method is used it sorts the objects based on compare() method of the Comparator interface.

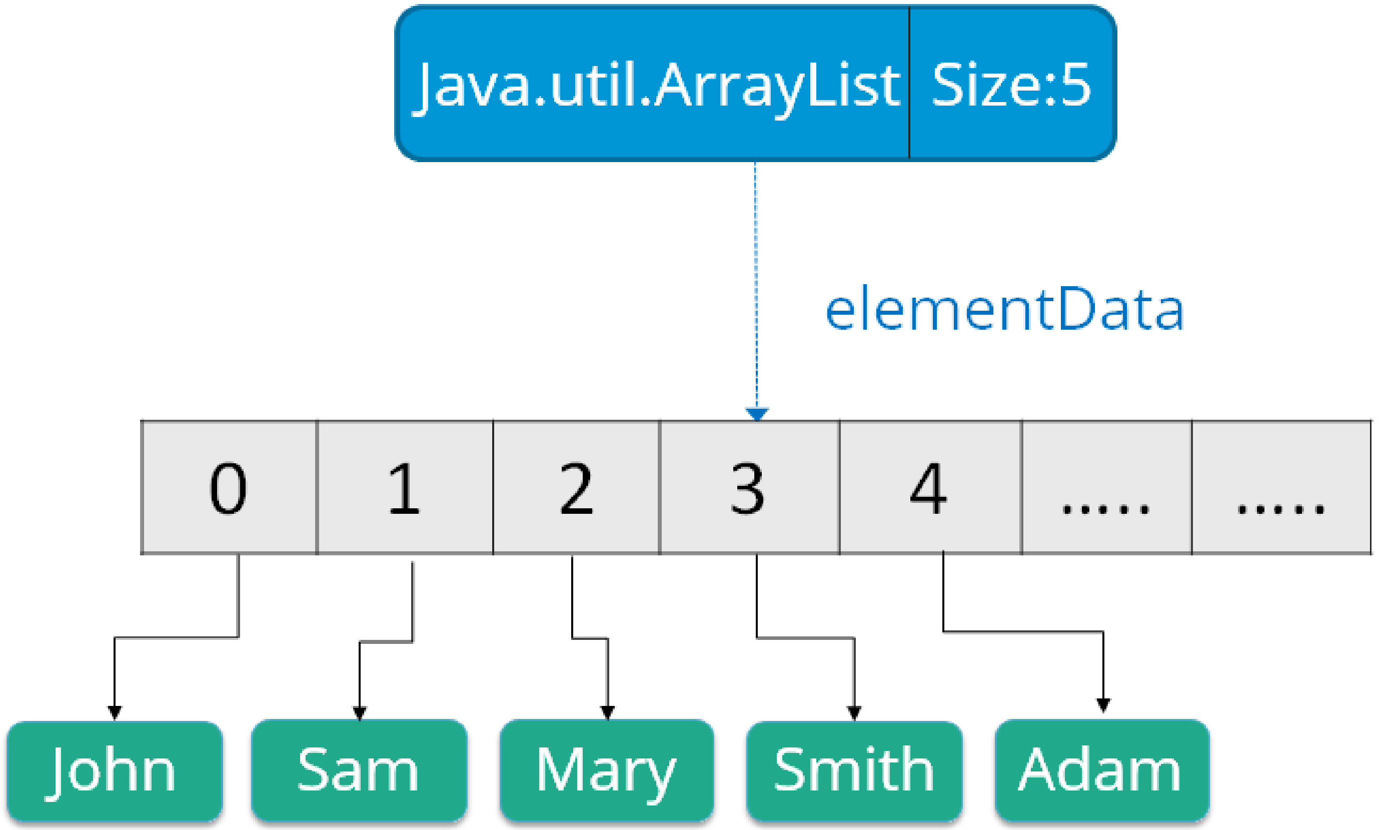
List — Java Collections Interview Questions

11. What is the use of the List interface?

The List interface in Java is an **ordered collection** of elements. It maintains the insertion order and allows duplicate values to be stored within. This interface contains various methods that enable smooth manipulation of elements based on the element index. The main classes implementing the List interface of the Collection framework are **ArrayList**, **LinkedList**, **Stack, and Vector**.

12. What is ArrayList in Java?

ArrayList is the implementation of List Interface where the elements can be collection that permits duplicate values. The size of an ArrayList can be increased dynamically if the number of elements is more than the initial size.



Syntax:

13. How would you convert an ArrayList to Array and an Array to ArrayList?

An Array can be converted into an ArrayList by making use of the asList() method provided by the Array class. It is a static method that accepts List objects as a parameter.

Syntax:

```
Arrays.asList(item)
```

Whereas an ArrayList can be converted into an array using the toArray() method of the ArrayList class.

Syntax:

```
List_object.toArray(new String[List_object.size()])
```

ArrayList can be reversed using the reverse() method of the Collections class.

Syntax:

```
public static void reverse(Collection c)
```

For Example:

```
public class ReversingArrayList {
public static void main(String[] args) {
List<String> myList = new ArrayList<String>();
myList.add("AWS");
myList.add("Java");
myList.add("Python");
myList .add("Blockchain");
System.out.println("Before Reversing");
System.out.println(myList.toString());
Collections.reverse(myList);
System.out.println("After Reversing");
System.out.println(myList);
}
}
```

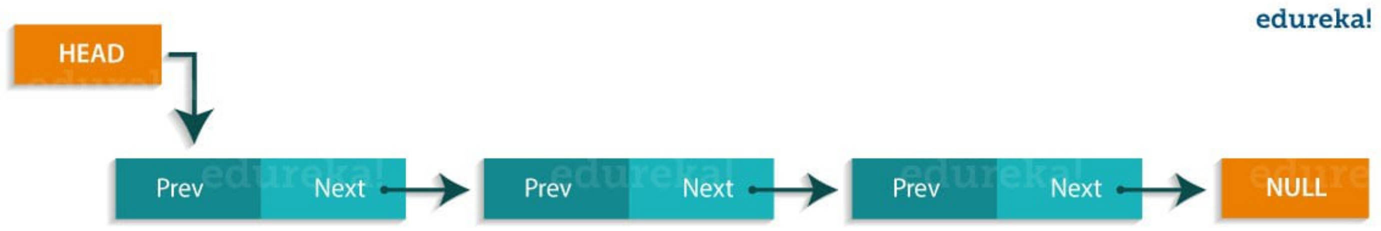
LinkedList in Java is a data structure that contains a sequence of links. Here each link contains a connection to the next link.

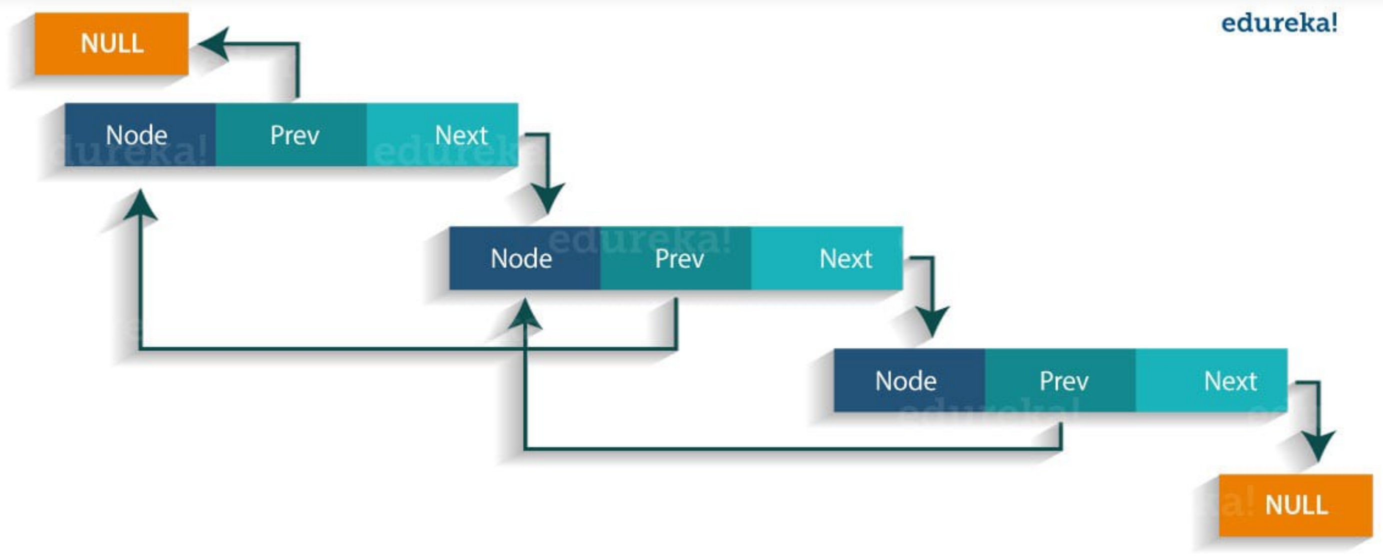
Syntax:

```
Linkedlist object = new Linkedlist();
```

Java LinkedList class uses two types of LinkedList to store the elements:

- ***Singly Linked List:*** In a singly LinkedList, each node in this list stores the data of the node and a pointer or reference to the next node in the list.

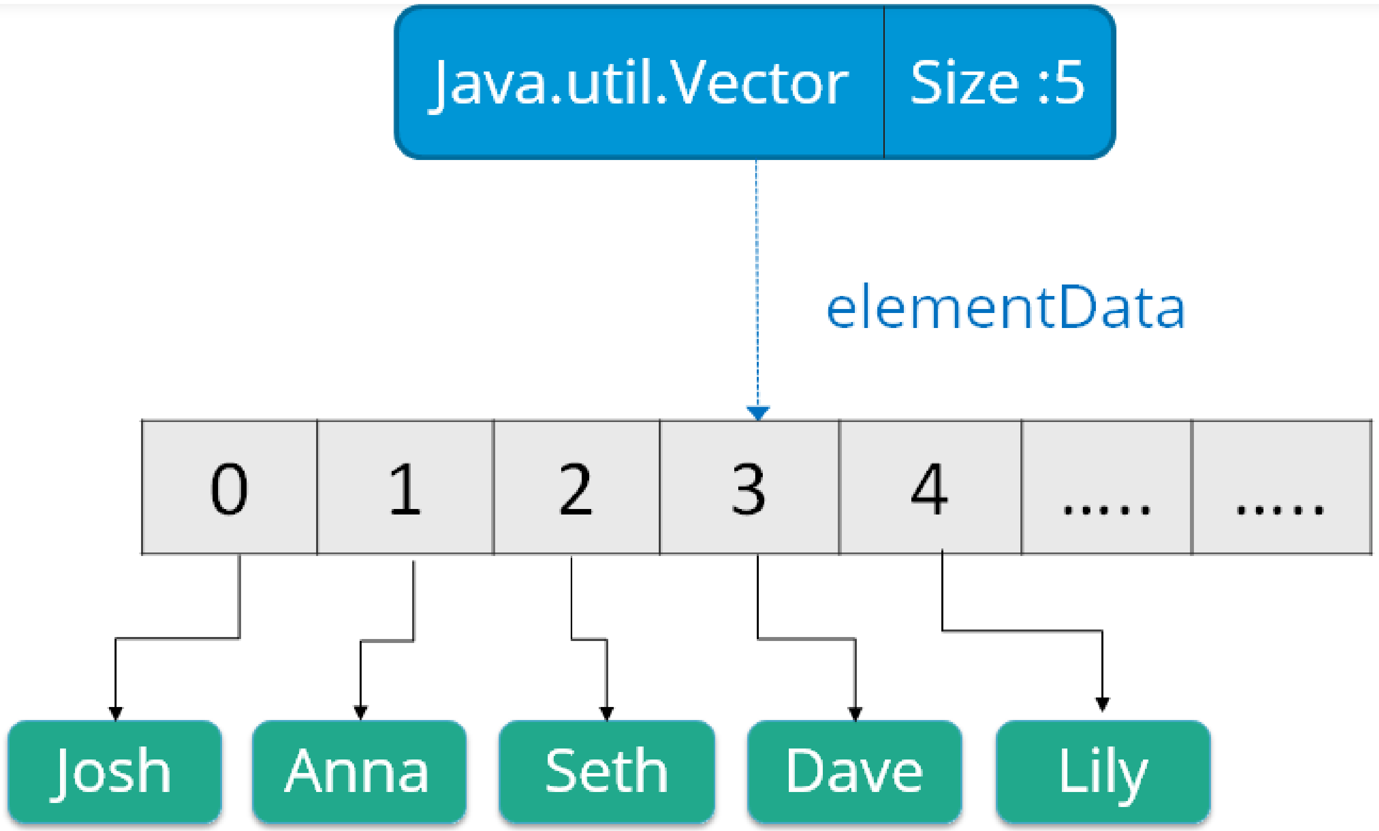




16. What is a Vector in Java?

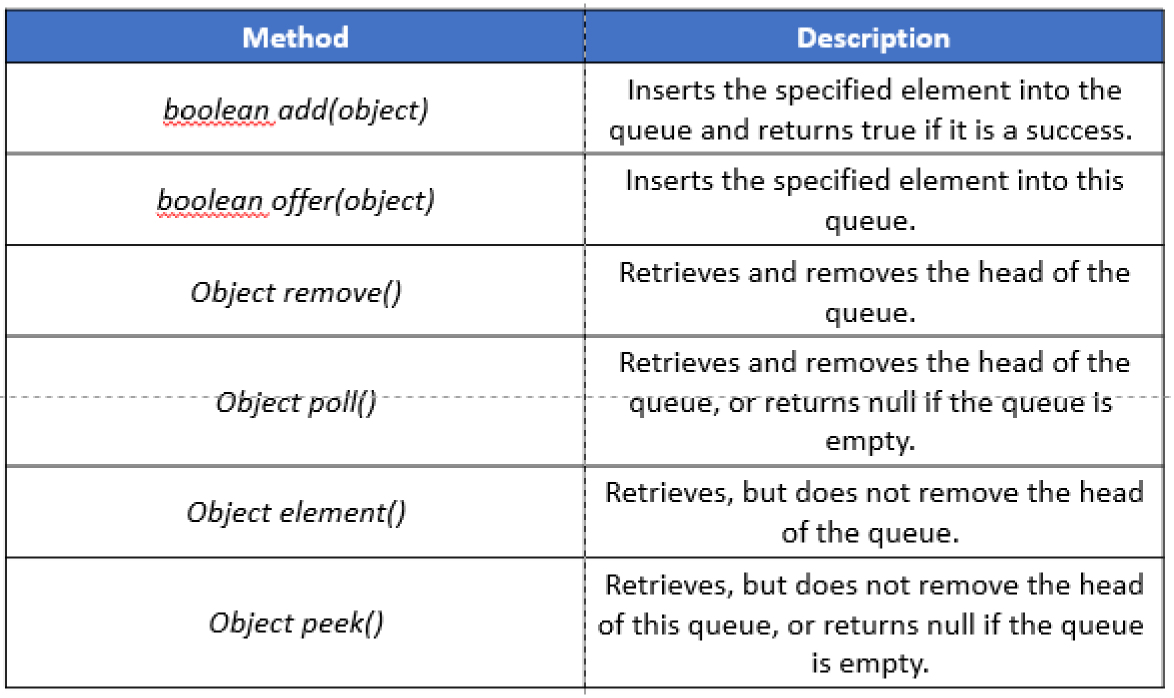
Vectors are similar to arrays, where the elements of the vector object can be accessed via an index into the vector. Vector implements a dynamic array. Also, the vector is not limited to a specific size, it can shrink or grow automatically whenever required. It is similar to ArrayList, but with two differences :

- Vector is synchronized.
- The vector contains many legacy methods that are not part of the collections



```
Vector object = new Vector(size,increment);
```

Below are some of the methods of Java Queue interface:

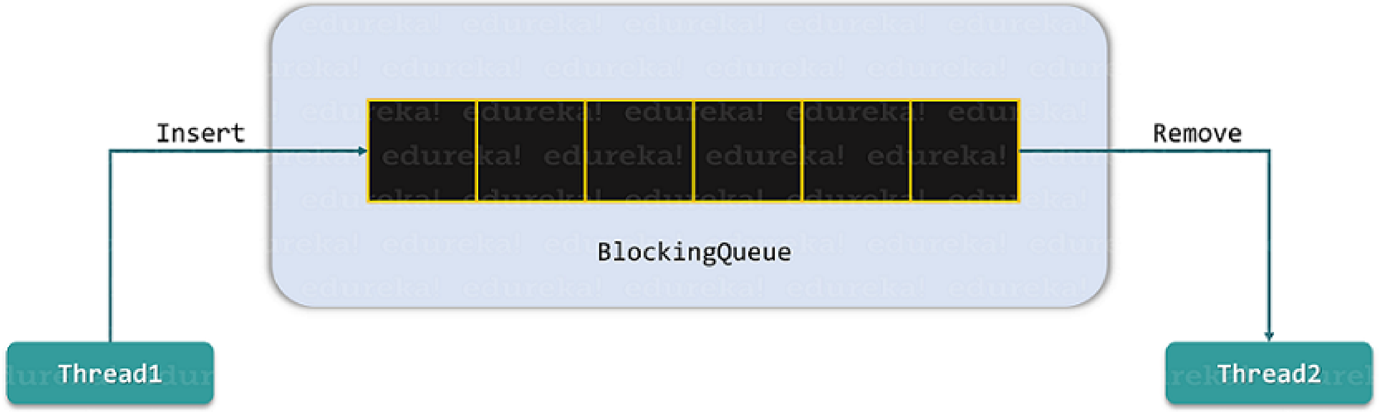


Method	Description
boolean add(object)	Inserts the specified element into the queue and returns true if it is a success.
boolean offer(object)	Inserts the specified element into this queue.
Object remove()	Retrieves and removes the head of the queue.
Object poll()	Retrieves and removes the head of the queue, or returns null if the queue is empty.
Object element()	Retrieves, but does not remove the head of the queue.
Object peek()	Retrieves, but does not remove the head of this queue, or returns null if the queue is empty.

18. What do you understand by BlockingQueue?

BlockingQueue interface belongs to the **java.util.concurrent** package. This interface enhances flow control by activating blocking, in case a thread is trying to dequeue an empty queue or enqueue an already full queue. While working

NullPointerException. The below figure represents the working of the BlockingQueue interface in Java.

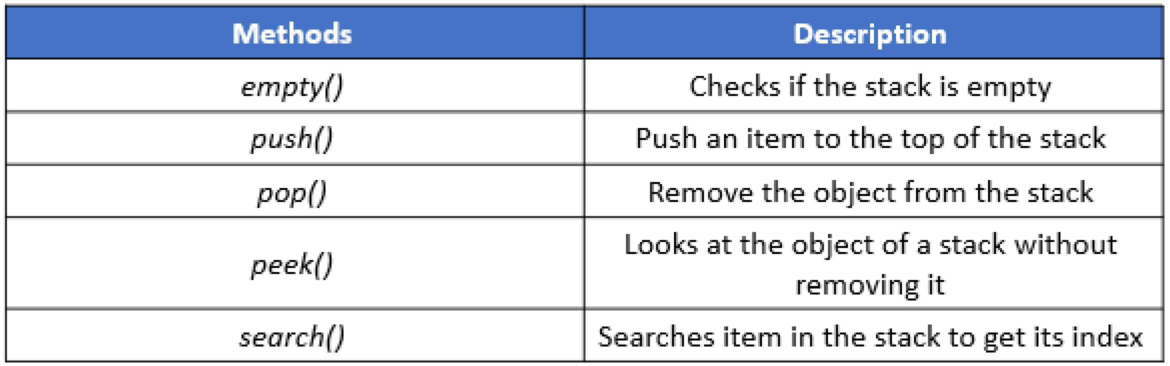


19. What is a priority queue in Java?

A priority queue in Java is an abstract data type similar to a regular queue or stack data structure but has a special feature called priority associated with each element. In this queue, a high priority element is served before a low priority element irrespective of their insertion order. The PriorityQueue is based on the priority heap. The elements of the priority queue are ordered according to the natural ordering, or by a Comparator provided at queue construction time.

20. What is the Stack class in Java and what are the various methods provided by it?

Java Stack class is an important part of the Java Collection framework and is based on the basic principle of last-in-first-out. In other words, the elements are added as well as removed from the rear end. The action of adding an element to a stack is called push while removing an element is referred to as pop. Below are the various methods provided by this class:



Methods	Description
empty()	Checks if the stack is empty
push()	Push an item to the top of the stack
pop()	Remove the object from the stack
peek()	Looks at the object of a stack without removing it
search()	Searches item in the stack to get its index

Set — Java Collections Interview Questions

A Set refers to a collection that cannot contain duplicate elements. It is mainly used to model the mathematical set abstraction. The Java platform provides three general-purpose Set implementations which are:

1. HashSet
2. TreeSet
3. LinkedHashSet

22. What is the HashSet class in Java and how does it store elements?

java.util.HashSet class is a member of the Java collections framework which inherits the AbstractSet class and implements the Set interface. It implicitly implements a hashtable for creating and storing a collection of unique elements. Hashtable is an instance of the HashMap class that uses a hashing mechanism for storing the information within a HashSet. Hashing is the process of converting the informational content into a unique value that is more popularly known as hash code. This hashcode is then used for indexing the data associated with the key. The entire process of transforming the informational key into the hashcode is performed internally.

In HashSet, only one null element can be added but in TreeSet it can't be added as it makes use of NavigableMap for storing the elements. This is because the NavigableMap is a subtype of SortedMap that doesn't allow null keys. So, in case you try to add null elements to a TreeSet, it will throw a NullPointerException.

24. Explain the emptySet() method in the Collections framework?

The Collections.emptySet() is used to return the empty immutable Set while removing the null elements. The set returned by this method is serializable. Below is the method declaration of emptySet().

Syntax:

```
public static final <T> Set<T> emptySet()
```

25. What is LinkedHashSet in Java Collections Framework?

A java.util.LinkedHashSet is a subclass of the HashSet class and implements the Set interface. Itis an ordered version of HashSet which maintains a doubly-linked

Syntax:

```
LinkedHashSet<String> hs = new LinkedHashSet<String>();
```

Map — Java Collections Interview Questions

26. What is Map interface in Java?

The java.util.Map interface in Java stores the elements in the form of keys-values pairs which is designed for faster lookups. Here every key is unique and maps to a single value. These key-value pairs are known as the map entries. This interface includes method signatures for insertion, removal, and retrieval of elements based on a key. With such methods, it's a perfect tool to use for key-value association mapping such as dictionaries.

27. Why Map doesn't extend the Collection Interface?

The Map interface in Java follows a key/value pair structure whereas the doesn't support the key-value pair like Map interface's put(K, V) method. It might not extend the Collection interface but still is an integral part of the Java Collections framework.

28. List down the different Collection views provided by the Map interface in the Java Collection framework?

The Map interface provides 3 views of key-value pairs which are:

- key set view
- value set view
- entry set view

All these views can be easily navigated through using the iterators.

29. What is the ConcurrentHashMap in Java and do you implement it?

ConcurrentHashMap is a Java class that implements ConcurrentMap as well as to Serializable interfaces. This class is the enhanced version of HashMap as it doesn't perform well in the multithreaded environment. It has a higher

Below is a small example demonstrating the implementation of ConcurrentHashMap:

```
package edureka;
import java.util.concurrent.*;

public class ConcurrentHashMapDemo {
    public static void main(String[] args)
    {
        ConcurrentHashMap m = new ConcurrentHashMap();
        m.put(1, "Welcome");
        m.put(2, "to");
        m.put(3, "Edureka's");
        m.put(4, "Demo");

        System.out.println(m);

        // Here we cant add Hello because 101 key
        // is already present in ConcurrentHashMap object
        m.putIfAbsent(3, "Online");
        System.out.println("Checking if key 3 is already present in
the ConcurrentHashMap object: "+ m);

        // We can remove entry because 101 key
        // is associated with For value
        m.remove(1, "Welcome");
        System.out.println("Removing the value of key 1: "+m);
```

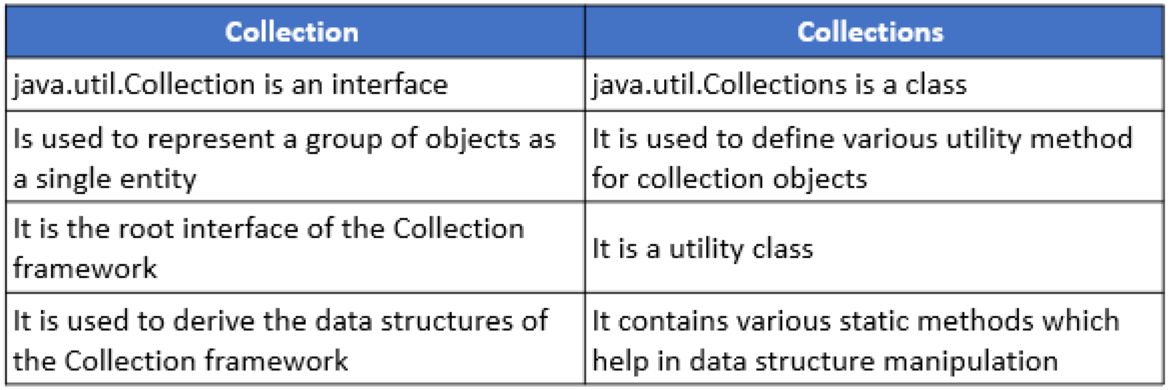
```
            System.out.println("Replacing value of key 1 with Welcome: "+
m);
      }
}
```

30. Can you use any class as a Map key?

Yes, any class can be used as Map Key as long as the following points are considered:

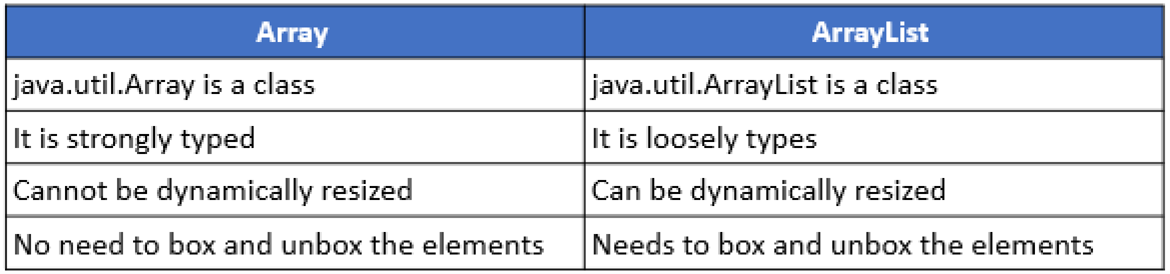
- The class overriding the equals() method must also override the hashCode() method
- The class should adhere to the rules associated with equals() and hashCode() for all instances
- The class field which is not used in the equals() method should not be used in hashCode() method as well
- The best way to use a user-defined key class is by making it immutable. It helps in caching the hashCode() value for better performance. Also if the class is made immutable it will ensure that the hashCode() and equals() are not changing in the future.

31. Differentiate between Collection and Collections.

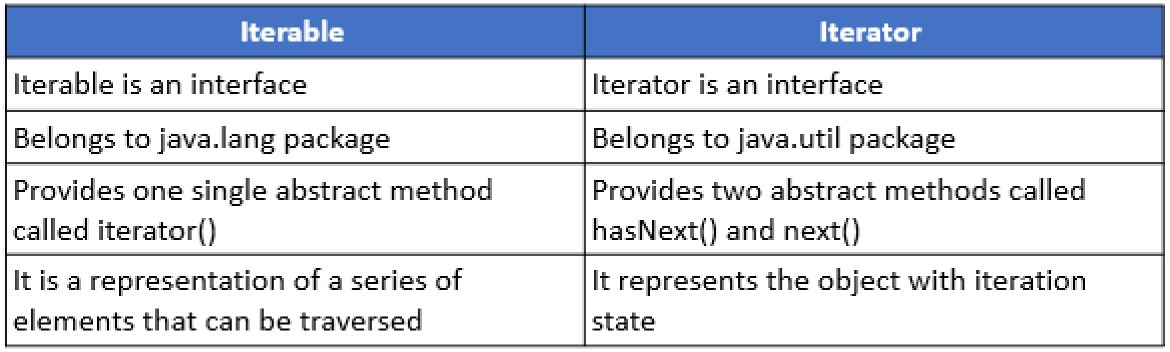


Collection	Collections
java.util.Collection is an interface	java.util.Collections is a class
Is used to represent a group of objects as a single entity	It is used to define various utility method for collection objects
It is the root interface of the Collection framework	It is a utility class
It is used to derive the data structures of the Collection framework	It contains various static methods which help in data structure manipulation

32. Differentiate between an Array and an ArrayList.

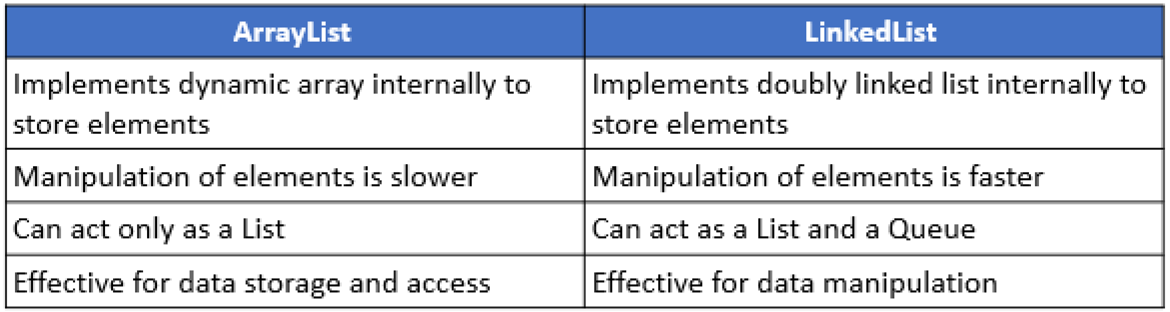


Array	ArrayList
java.util.Array is a class	java.util.ArrayList is a class
It is strongly typed	It is loosely types
Cannot be dynamically resized	Can be dynamically resized
No need to box and unbox the elements	Needs to box and unbox the elements

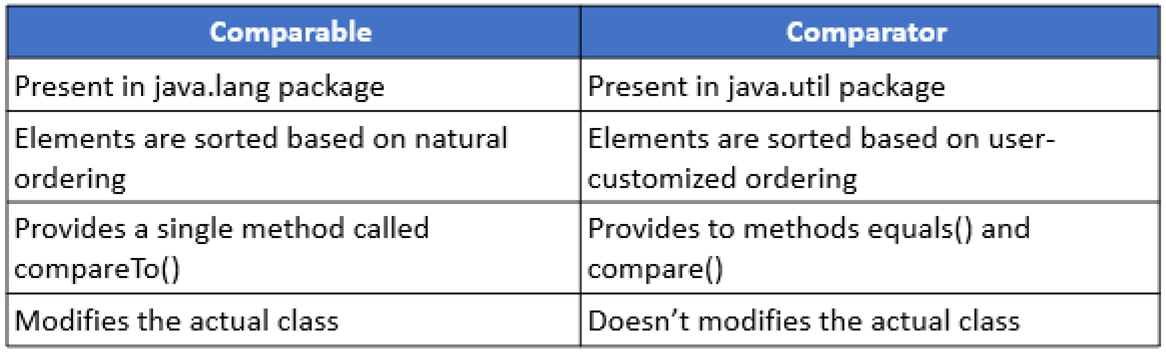


Iterable	Iterator
Iterable is an interface	Iterator is an interface
Belongs to java.lang package	Belongs to java.util package
Provides one single abstract method called iterator()	Provides two abstract methods called hasNext() and next()
It is a representation of a series of elements that can be traversed	It represents the object with iteration state

34. Differentiate between ArrayList and LinkedList.

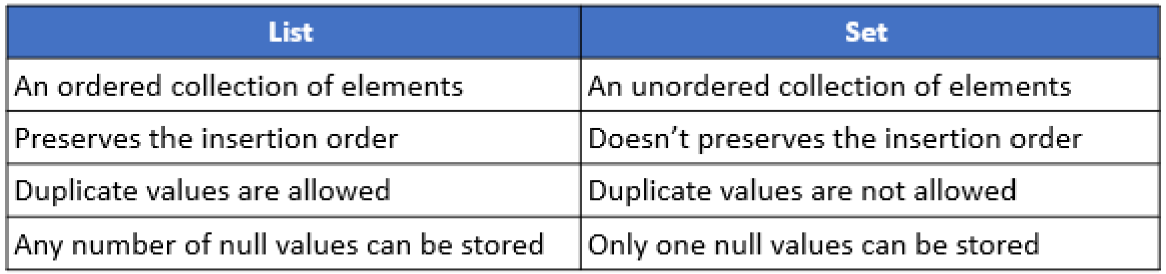


ArrayList	LinkedList
Implements dynamic array internally to store elements	Implements doubly linked list internally to store elements
Manipulation of elements is slower	Manipulation of elements is faster
Can act only as a List	Can act as a List and a Queue
Effective for data storage and access	Effective for data manipulation



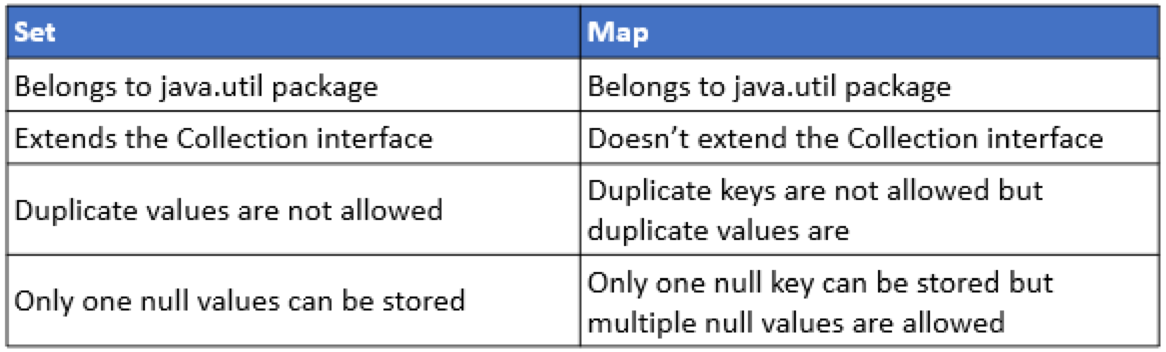
Comparable	Comparator
Present in java.lang package	Present in java.util package
Elements are sorted based on natural ordering	Elements are sorted based on user-customized ordering
Provides a single method called compareTo()	Provides to methods equals() and compare()
Modifies the actual class	Doesn't modifies the actual class

36. Differentiate between List and Set.



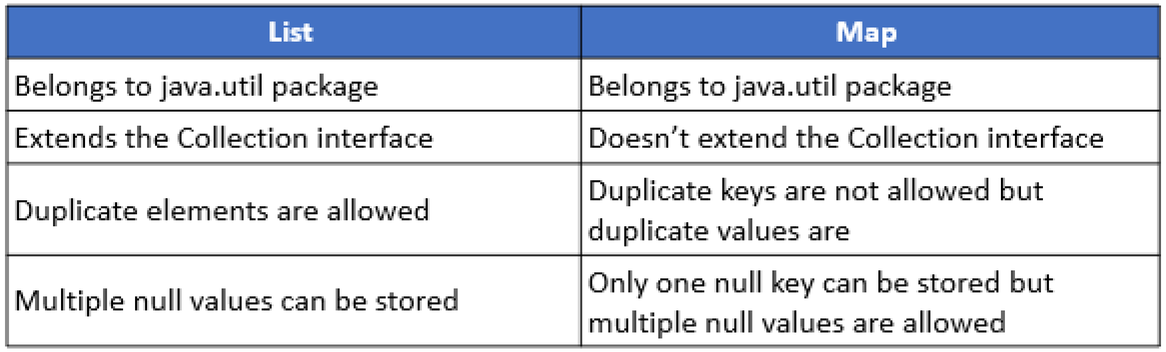
List	Set
An ordered collection of elements	An unordered collection of elements
Preserves the insertion order	Doesn't preserves the insertion order
Duplicate values are allowed	Duplicate values are not allowed
Any number of null values can be stored	Only one null values can be stored

37. Differentiate between Set and Map.



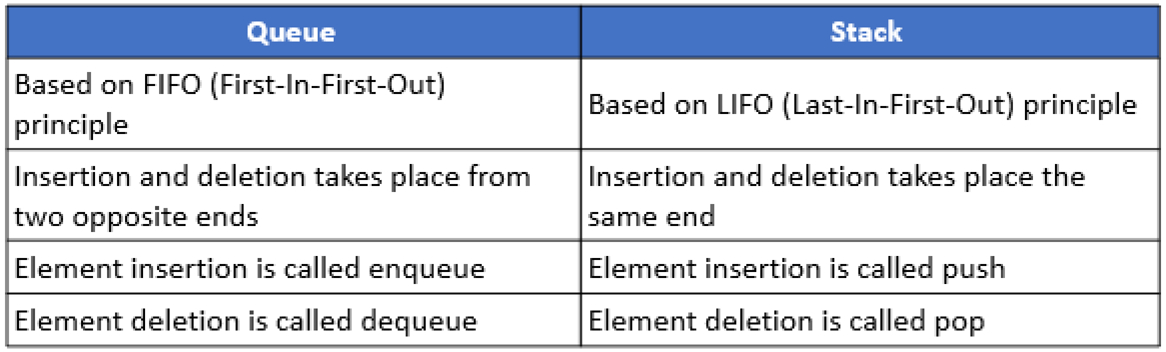
Set	Map
Belongs to java.util package	Belongs to java.util package
Extends the Collection interface	Doesn’t extend the Collection interface
Duplicate values are not allowed	Duplicate keys are not allowed but duplicate values are
Only one null values can be stored	Only one null key can be stored but multiple null values are allowed

38. Differentiate between List and Map.



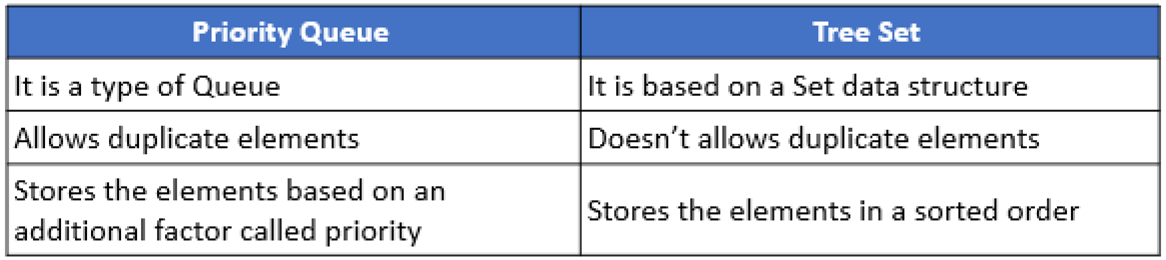
List	Map
Belongs to java.util package	Belongs to java.util package
Extends the Collection interface	Doesn’t extend the Collection interface
Duplicate elements are allowed	Duplicate keys are not allowed but duplicate values are
Multiple null values can be stored	Only one null key can be stored but multiple null values are allowed

39. Differentiate between Queue and Stack.



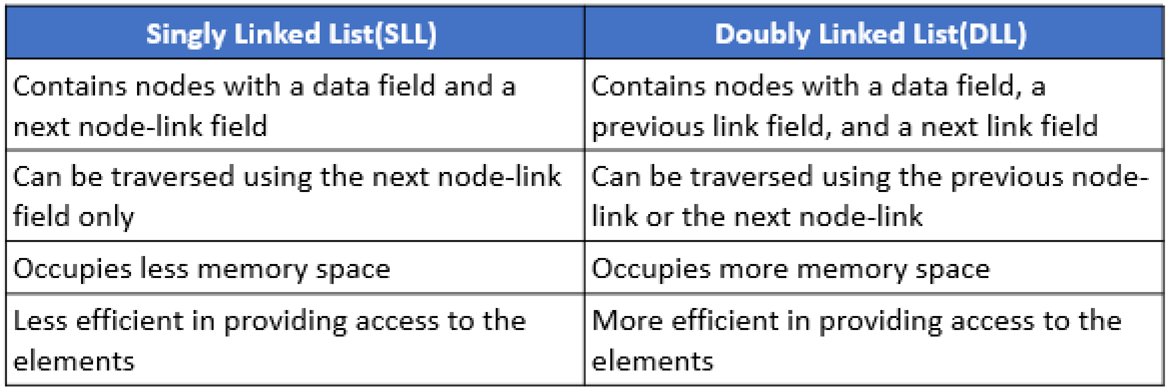
Queue	Stack
Based on FIFO (First-In-First-Out) principle	Based on LIFO (Last-In-First-Out) principle
Insertion and deletion takes place from two opposite ends	Insertion and deletion takes place the same end
Element insertion is called enqueue	Element insertion is called push
Element deletion is called dequeue	Element deletion is called pop

40. Differentiate between PriorityQueue and TreeSet.



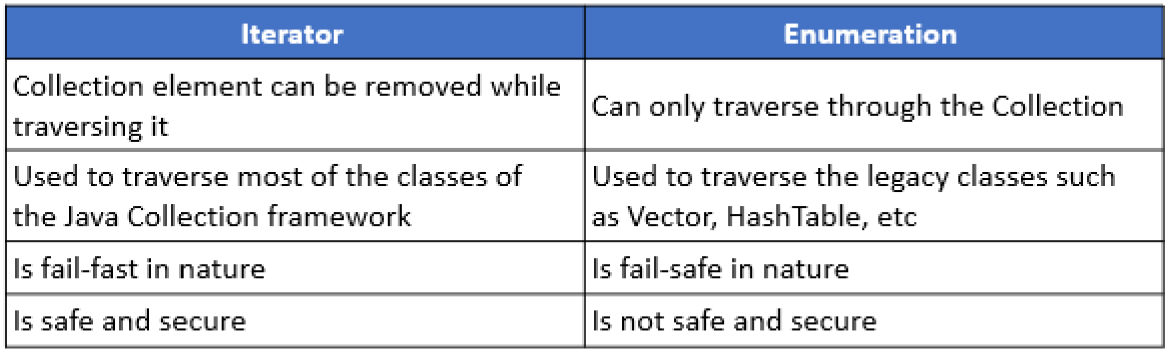
Priority Queue	Tree Set
It is a type of Queue	It is based on a Set data structure
Allows duplicate elements	Doesn't allows duplicate elements
Stores the elements based on an additional factor called priority	Stores the elements in a sorted order

41. Differentiate between the Singly Linked List and Doubly Linked List.

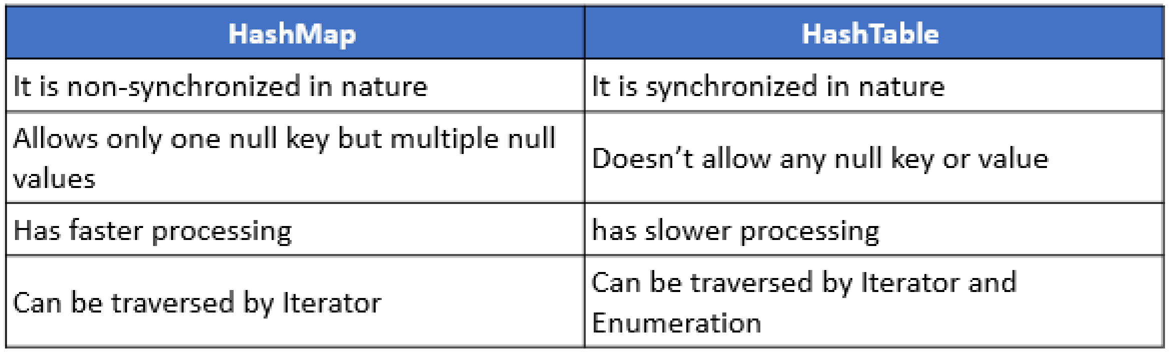


Singly Linked List(SLL)	Doubly Linked List(DLL)
Contains nodes with a data field and a next node-link field	Contains nodes with a data field, a previous link field, and a next link field
Can be traversed using the next node-link field only	Can be traversed using the previous node-link or the next node-link
Occupies less memory space	Occupies more memory space
Less efficient in providing access to the elements	More efficient in providing access to the elements

42. Differentiate between Iterator and Enumeration.

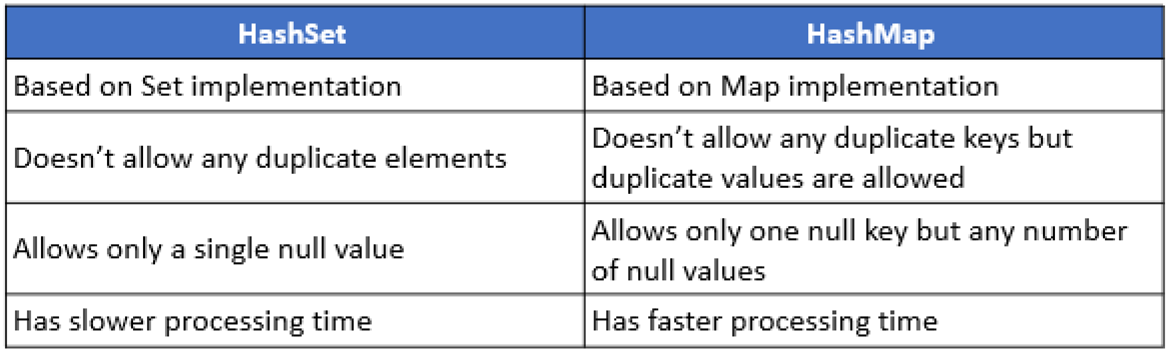


Iterator	Enumeration
Collection element can be removed while traversing it	Can only traverse through the Collection
Used to traverse most of the classes of the Java Collection framework	Used to traverse the legacy classes such as Vector, HashTable, etc
Is fail-fast in nature	Is fail-safe in nature
Is safe and secure	Is not safe and secure

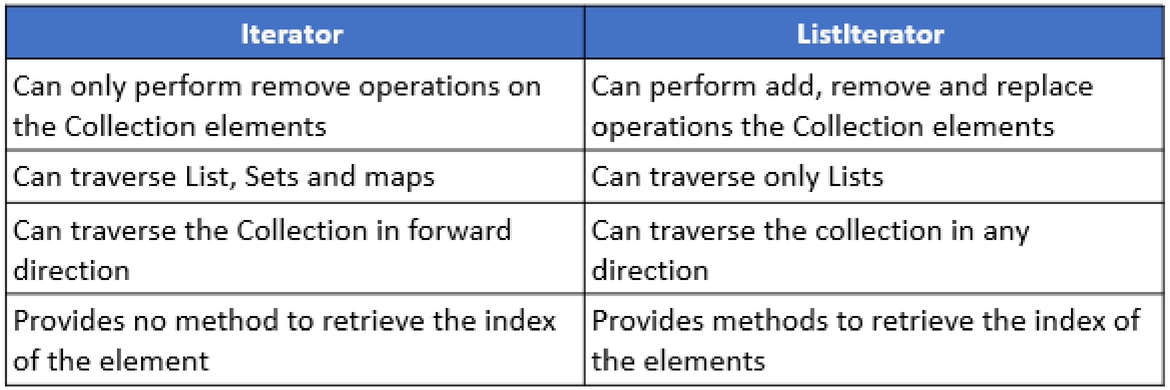


HashMap	HashTable
It is non-synchronized in nature	It is synchronized in nature
Allows only one null key but multiple null values	Doesn't allow any null key or value
Has faster processing	has slower processing
Can be traversed by Iterator	Can be traversed by Iterator and Enumeration

44. Differentiate between HashSet and HashMap.

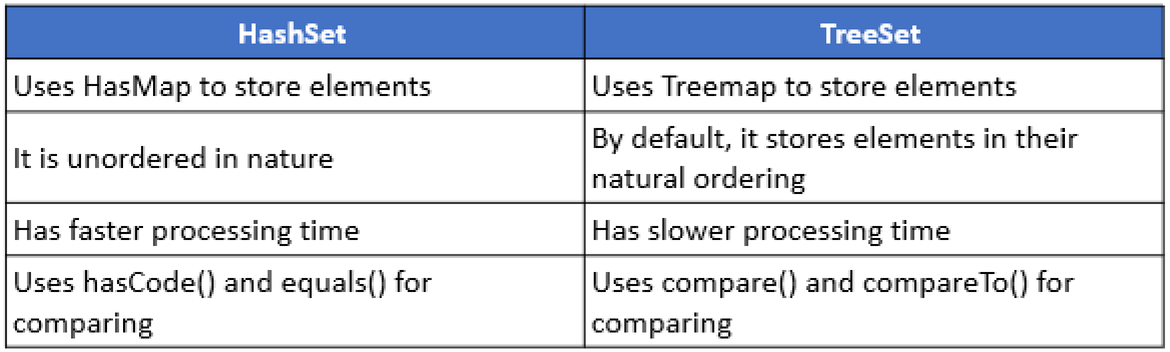


HashSet	HashMap
Based on Set implementation	Based on Map implementation
Doesn't allow any duplicate elements	Doesn't allow any duplicate keys but duplicate values are allowed
Allows only a single null value	Allows only one null key but any number of null values
Has slower processing time	Has faster processing time

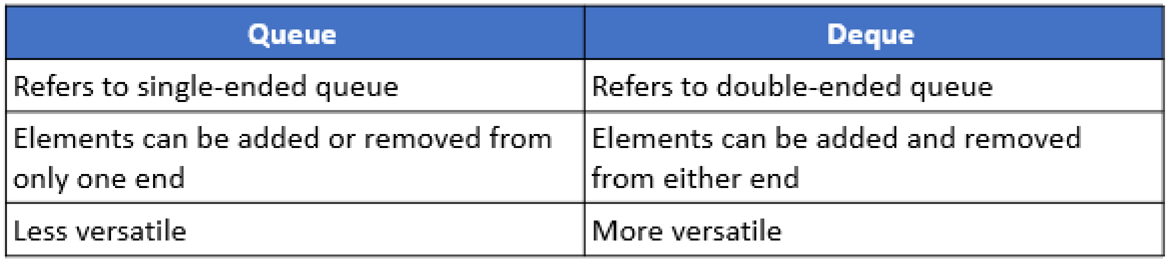


Iterator	ListIterator
Can only perform remove operations on the Collection elements	Can perform add, remove and replace operations the Collection elements
Can traverse List, Sets and maps	Can traverse only Lists
Can traverse the Collection in forward direction	Can traverse the collection in any direction
Provides no method to retrieve the index of the element	Provides methods to retrieve the index of the elements

46. Differentiate between HashSet and TreeSet.

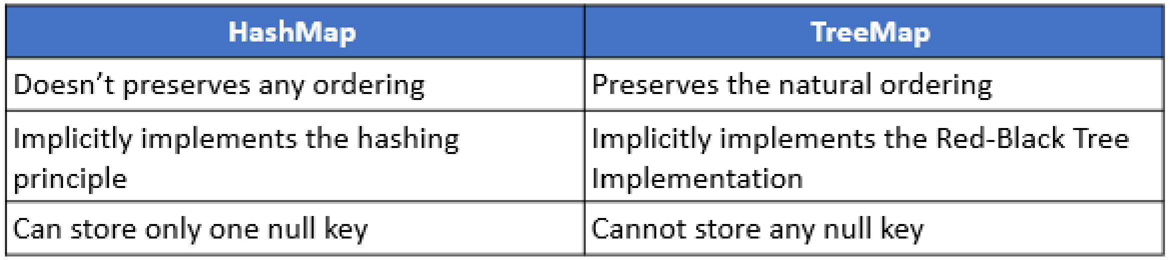


HashSet	TreeSet
Uses HasMap to store elements	Uses Treemap to store elements
It is unordered in nature	By default, it stores elements in their natural ordering
Has faster processing time	Has slower processing time
Uses hasCode() and equals() for comparing	Uses compare() and compareTo() for comparing



Queue	Deque
Refers to single-ended queue	Refers to double-ended queue
Elements can be added or removed from only one end	Elements can be added and removed from either end
Less versatile	More versatile

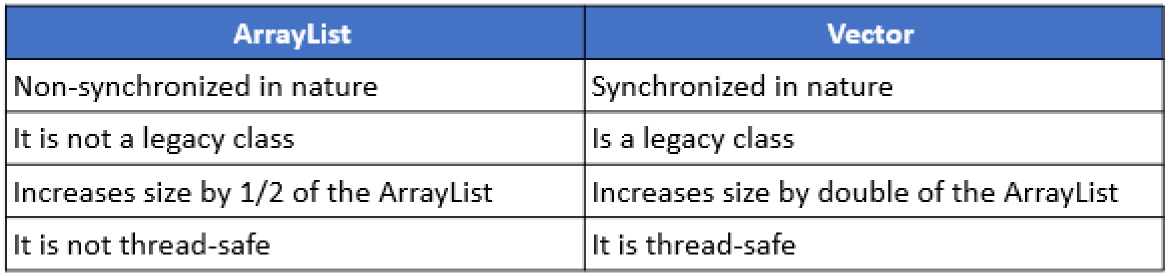
48. Differentiate between HashMap and TreeMap.



HashMap	TreeMap
Doesn’t preserves any ordering	Preserves the natural ordering
Implicitly implements the hashing principle	Implicitly implements the Red-Black Tree Implementation
Can store only one null key	Cannot store any null key

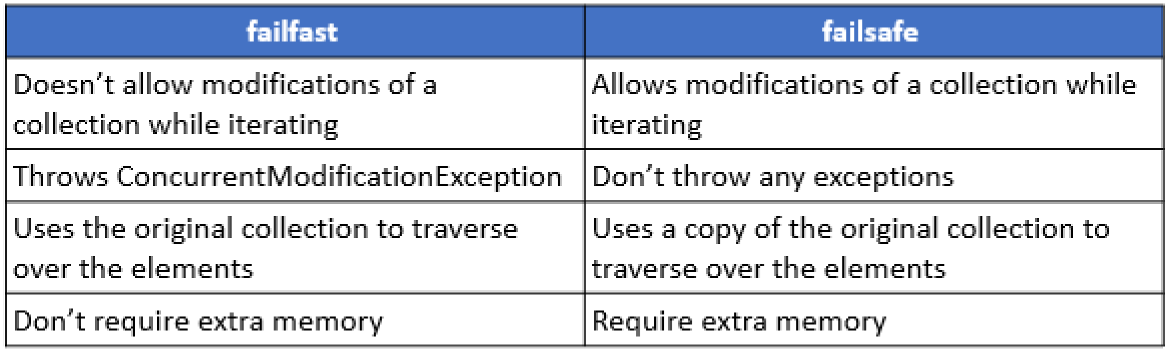
49. Differentiate between ArrayList and Vector.

Get unlimited access Open in app



ArrayList	Vector
Non-synchronized in nature	Synchronized in nature
It is not a legacy class	Is a legacy class
Increases size by 1/2 of the ArrayList	Increases size by double of the ArrayList
It is not thread-safe	It is thread-safe

50. Differentiate between failfast and failsafe.



failfast	failsafe
Doesn't allow modifications of a collection while iterating	Allows modifications of a collection while iterating
Throws ConcurrentModificationException	Don't throw any exceptions
Uses the original collection to traverse over the elements	Uses a copy of the original collection to traverse over the elements
Don't require extra memory	Require extra memory

So this brings us to the end of the Java Collections interview questions. The topics that you learned in this Java Collections Interview Questions are the most

Java Collection Interview Questions will definitely help you ace your job interview. **Good luck with your interview!**

If you wish to check out more articles on the market's most trending technologies like Artificial Intelligence, DevOps, Ethical Hacking, then you can refer to Edureka's official site.

Do look out for other articles in this series which will explain the various other aspects of Java.

> 1. Object Oriented Programming
>
> 2. Java Tutorial
>
> 3. Polymorphism in Java
>
> 4. Abstraction in Java
>
> 5. Java String

7. Java Collections

8. Java Threads

9. Introduction to Java Servlets

10. Servlet and JSP Tutorial

11. Exception Handling in Java

12. Advanced Java Tutorial

13. Java Interview Questions

14. Java Programs

15. Kotlin vs Java

16. Dependency Injection Using Spring Boot

18. Top 10 Java frameworks

19. Java Reflection API

20. Top 30 Patterns in Java

21. Core Java Cheat Sheet

22. Socket Programming In Java

23. Java OOP Cheat Sheet

24. Annotations in Java

25. Library Management System Project in Java

26. Trees in Java

27. Machine Learning in Java

29. Java Developer Skills

30. Top 55 Servlet Interview Questions

31. Top Java Projects

32. Java Strings Cheat Sheet

33. Nested Class in Java

34. Java Collections Interview Questions and Answers

35. How to Handle Deadlock in Java?

*36.*Java Tutorial

37. What is the concept of String Pool in Java?

38. What is the difference between C, C++, and Java?

40. *Top MVC Interview Questions and Answers You Need to Know*

41. *Top 10 Applications of Java Programming Language*

42. *Deadlock in Java*

43. *Square and Square Root in Java*

44. *Typecasting in Java*

45. *Operators in Java and its Types*

46. *Destructor in Java*

47. *Binary Search in Java*

48. *MVC Architecture in Java*

49. *Hibernate Interview Questions And Answers*

Originally published at https://www.edureka.co.