Date: 20-02-2023

Day 1. Interview Preparation.

Collections

-------------

1)What is the difference between Array and Collection classes?

Ans- In order to store multiple values or objects of the same type, Java provides two types of data structures namely Array and Collection.

The following are the important differences between Arrays and Collection.

| **Sr. No.** | **Key** | **Arrays** | **Collection** |
| --- | --- | --- | --- |
| 1 | Size | Arrays are fixed in size i.e once the array with the specific size is declared then we can't alter its size afterward. | The collection is dynamic in size i.e based on requirement size could be get altered even after its declaration. |
| 2 | Memory Consumption | Arrays due to fast execution consumes more memory and has better performance. | Collections, on the other hand, consume less memory but also have low performance as compared to Arrays. |
| 3 | Data type | Arrays can hold the only the same type of data in its collection i.e only homogeneous data types elements are allowed in case of arrays. | Collection, on the other hand, can hold both homogeneous and heterogeneous elements. |
| 4 | Primitives storage | Arrays can hold both object and primitive type data. | On the other hand, collection can hold only object types but not the primitive type of data. |
| 5 | Performance | Arrays due to its storage and internal implementation better in performance. | Collection on the other hand with respect to performance is not recommended to use. |

2)Name the core Collection interfaces & their implementing Collection classes

Ans- The **Collection**interface is a member of the [Java Collections Framework](https://www.geeksforgeeks.org/collections-in-java-2/). 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](https://www.geeksforgeeks.org/list-interface-java-examples/), [Queue](https://www.geeksforgeeks.org/queue-interface-java/), and [Set](https://www.geeksforgeeks.org/set-in-java/).

3)What is the root interface in collection hierarchy?List its methods

Ans- java.util.Collection is the root interface of Collections Framework. It is on the top of the Collections framework hierarchy. It contains some important methods such as size(), iterator(), add(), remove(), clear() that every Collection class must implement.

Some other important interfaces are java.util.List, java.util.Set, java.util.Queue and java.util.Map. The Map is the only interface that doesn’t inherit from the Collection interface but it’s part of the Collections framework. All the collections framework interfaces are present in java.util package.

4)Explain difference between

5)ArrayList & Vector

Ans-

|  |  |  |
| --- | --- | --- |
| **S.No.** | **ArrayList** | **Vector** |
| 1. | ArrayList is a resizable or dynamic array. | Vectors are also a form of dynamic array. |
| 2. | ArrayList is not synchronised. | Vector is synchronised. |
| 3. | Syntax of ArrayList:  ArrayList<T> al = new ArrayList<T>(); | Syntax of Vector:  Vector<T> v = new Vector<T>(); |
| 4. | If the number of elements overextends its capacity, ArrayList can increase the 50% of the present array size. | If the number of elements overextends its capacity, Vector can increase the 100% of the present array size, which means double the size of an array. |
| 5. | It is not an inheritance class. | It is an inheritance class. |
| 6. | It is faster than Vector. | It is slow as compared to the ArrayList. |
| 7. | It is not a legacy class. | It is a legacy class. |
| 8. | It prefers the Iterator interface to traverse the components. | It prefers an Enumeration or Iterator interface to traverse the elements. |

6)ArrayList & LinkedList

Following are the important differences between ArrayList and LinkedList method.

| **Sr. No.** | **Key** | **ArrayList** | **LinkedList** |
| --- | --- | --- | --- |
| 1 | Internal Implementation | ArrayList internally uses a dynamic array to store its elements. | LinkedList uses Doubly Linked List to store its elements. |
| 2 | Manipulation | ArrayList is slow as array manipulation is slower. | LinkedList is faster being node based as not much bit shifting required. |
| 3 | Implementation | ArrayList implements only List. | LinkedList implements List as well as Queue. It can acts as a queue as well. |
| 4 | Access | ArrayList is faster in storing and accessing data. | LinkedList is faster in manipulation of data. |

7)HashSet & TreeSet

## Ans- **Difference between Hashset and Treeset in Java**

| **S. No.** | **Hash Set** | **Tree Set** |
| --- | --- | --- |
| 1 | The Hash set is executed with the help of a HashTable. | The tree set is executed with the help of a tree structure. |
| 2 | It does not authorise a heterogeneous object. | It authorises a heterogeneous object. |
| 3 | It permits a null object. | It does not permit the null object. |
| 4 | To compare two objects, we use the equals method. | To compare two objects, we use the compare method. |
| 5 | It does not support any order | TreeSet supports an object in sorted order. |

8)HashSet & HashMap

Ans-

| **Sr. No.** | **Key** | **HashMap** | **HashSet** |
| --- | --- | --- | --- |
| 1 | Implementation | Hashmap is the implementation of Map interface. | Hashset on other hand is the implementation of set interface. |
| 2 | Internal implementation | Hashmap internally do not implements hashset or any set for its implementation. | Hashset internally uses Hashmap for its implementation. |
| 3 | Storage of elements | HashMap Stores elements in form of key-value pair i.e each element has its corresponding key which is required for its retrieval during iteration. | HashSet stores only objects no such key value pairs maintained. |
| 4 | Method to add element | Put method of hash map is used to add element in hashmap. | On other hand add method of hashset is used to add element in hashset. |
| 5 | Index performance | Hashmap due to its unique key is faster in retrieval of element during its iteration. | HashSet is completely based on object so compared to hashmap is slower. |
| 6 | Null Allowed | Single null key and any number of null value can be inserted in hashmap without any restriction. | On other hand Hashset allows only one null value in its collection,after which no null value is allowed to be added. |

9)Hashtable & HashMap

Ans-

* HashMap is non-syncronized and is not thread safe while HashTable is thread safe and is synchronized.
* HashMap allows one null key and values can be null whereas HashTable doesn't allow null key or value.
* HashMap is faster than HashTable.
* HashMap iterator is fail-safe where HashTable iterator is not fail-safe.
* HashMap extends AbstractMap class where HashTable extends Dictionary class.

10)HashMap & TreeMap

Ans-

## **Differences between TreeMap, HashMap in Java**

|  |  |  |
| --- | --- | --- |
| **Parameters** | **TreeMap** | **HashMap** |
| Insertion and Hookup | For insertion and hookup, it has O(logN) complexity. | For insertion and hookup, it has O(1) complexity. |
| Null Keys | It does not allow any null key. | It only allows a single null key. |
| Null Values | It allows null values. | It allows multiple numbers of null values. |
| Maintaining Order | The primary function of TreeMap is to maintain order. It helps us in the storage of keys in a sorted manner in ascending order. | The HashMap does not function to maintain any order. |

11)Iterator &ListIterator

Ans- An **Iterator**is an interface in Java and we can traverse the elements of a list in a **forward direction** whereas a **ListIterator**is an interface that extends the **Iterator** interface and we can traverse the elements in **both forward and backward directions.** An **Iterator**can be used in these collection types like**List, Set**, and **Queue**whereas **ListIterator**can be used in **List**collection only. The important methods of **Iterator**interface are**hasNext(), next()** and **remove()**whereas important methods of **ListIterator**interface are **add()**, **hasNext()**, **hasPrevious()**and **remove()**.

12)Iterator & Enumeration

| **Sr. No.** | **Key** | **Iterator** | **Enumeration** |
| --- | --- | --- | --- |
| 1 | Basic | In Iterator,  we can read and remove element while traversing element in the collections. | Using Enumeration, we can only read element during traversing element in the collections. |
| 2. | Access | It can be used with any class of the collection framework. | It can be used only with legacy class of the collection framework such as a Vector and HashTable. |
| 3. | Fail-Fast and Fail -Safe | Any changes in the collection, such as removing element from the collection during a thread is iterating collection then it throw concurrent modification exception. | Enumeration  is Fail safe in nature. It doesn’t throw concurrent modification exception |
| 4. | Limitation | Only forward direction iterating is possible | Remove operations can not be performed using Enumeration. |
| 5. | Methods | It has following methods − \*hasNext() \*next() \*remove() | It has following methods − \*hasMoreElements() \*nextElement() |

Ans-

13)Set & List

Ans-

|  |  |  |
| --- | --- | --- |
| **S.No** | **List** | **Set** |
| 1. | The list implementation allows us to add the same or duplicate elements. | The set implementation doesn't allow us to add the same or duplicate elements. |
| 2. | The insertion order is maintained by the List. | It doesn't maintain the insertion order of elements. |
| 3. | List allows us to add any number of null values. | Set allows us to add at least one null value in it. |
| 4. | The List implementation classes are LinkedList and ArrayList. | The Set implementation classes are TreeSet, HashSet and LinkedHashSet. |
| 5. | We can get the element of a specified index from the list using the get() method. | We cannot find the element from the Set based on the index because it doesn't provide any get method(). |
| 6. | It is used when we want to frequently access the elements by using the index. | It is used when we want to design a collection of distinct elements. |
| 7. | The method of List interface listiterator() is used to iterate the List elements. | The iterator is used when we need to iterate the Set elements. |

In Java, both the List and the Set are available in the [Collection framework](https://www.javatpoint.com/collections-in-java). In order to store the collection of objects as a single unit, Set and List interface are used. Apart from these similarities, both the interfaces have so many differences too, which are as follows:

14)Set & Map

Ans-

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Set** | **Map** |
| 1. | Set is used to construct the mathematical Set in Java. | Map is used to do mapping in the database. |
| 2. | It cannot contain repeated values. | It can have the same value for different keys. |
| 3. | Set doesn't allow us to add the same elements in it. Each class that implements the Set interface contains only the unique value. | Map contains unique key and repeated values. In Map, one or more keys can have the same values, but two keys cannot be the same. |
| 4. | We can easily iterate the Set elements using the keyset() and the entryset() method of it. | Map elements cannot be iterated. We need to convert Map into Set for iterating the elements. |
| 5. | Insertion order is not maintained by the Set interface. However, some of its classes, like LinkedHashSet, maintains the insertion order. | The insertion order is also not maintained by the Map. However, some of the Map classes like TreeMap and LinkedHashMap does the same. |

15)Queue and Stack

## Ans- **Difference Between Stack and Queue**

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Stack Data Structure** | **Queue Data Structure** |
| Basics | It is a linear data structure. The objects are removed or inserted at the same end. | It is also a linear data structure. The objects are removed and inserted from two different ends. |
| Working Principle | It follows the Last In, First Out (LIFO) principle. It means that the last inserted element gets deleted at first. | It follows the First In, First Out (FIFO) principle. It means that the first added element gets removed first from the list. |
| Pointers | It has only one pointer- the **top**. This pointer indicates the address of the topmost element or the last inserted one of the stack. | It uses two pointers (in a simple queue) for reading and writing data from both the ends- the **front** and the **rear**. The rear one indicates the address of the last inserted element, whereas the front pointer indicates the address of the first inserted element in a queue. |
| Operations | Stack uses **push** and **pop** as two of its operations. The pop operation functions to remove the element from the list, while the push operation functions to insert the element in a list. | Queue uses **enqueue** and **dequeue** as two of its operations. The dequeue operation deletes the elements from the queue, and the enqueue operation inserts the elements in a queue. |
| Structure | Insertion and deletion of elements take place from one end only. It is called the top. | It uses two ends- front and rear. Insertion uses the rear end, and deletion uses the front end. |
| Full Condition Examination | When top== max-1, it means that the stack is full. | When rear==max-1, it means that the queue is full. |
| Empty Condition Examination | When top==-1, it indicates that the stack is empty. | When front = rear+1 or front== -1, it indicates that the queue is empty. |
| Variants | A Stack data structure does not have any types. | A Queue data structure has three types- circular queue, priority queue, and double-ended queue. |
| Visualization | You can visualize the Stack as a vertical collection. | You can visualize a Queue as a horizontal collection. |
| Implementation | The implementation is simpler in a Stack. | The implementation is comparatively more complex in a Queue than a stack. |

16)Queue & List

Ans- **Here are some of the major differences between a Static Queue and a Singly Linked List**

|  |  |
| --- | --- |
| **Static Queue** | **Singly Linked List** |
| **Queue is a collection of one or more elements arranged in memory in a contiguous fashion.** | **A linked list is a collection of one or more elements arranged in memory in a dis-contiguous fashion.** |
| **Static Queue is always fixed size.** | **List size is never fixed.** |
| **In Queue, only one and single type of information is stored because static Queue implementation is through Array.** | **List also stored the address for the next node along with it’s content.** |
| **Static Queue is index based.** | **Singly linked list is reference based.** |
| **Insertion can always be performed on a single end called *REAR* and deletion on the other end called *FRONT*.** | **Insertion as well as deletion can performed any where within the list.** |
| **Queue is always based on FIFO.** | **List may be based on FIFI or LIFO etc.** |
| **Queue have two pointer FRONT and REAR.** | **While List has only one pointer basically called HEAD.** |

17)Collection and Collections

Ans- The **Collection**is an interface whereas **Collections**is a **utility**class in Java. The **Set, List,** and **Queue**are some of the subinterfaces of **Collection**interface, a **Map**interface is also part of the **Collections**Framework, but it doesn't inherit **Collection**interface. The important methods of **Collection**interface are **add(), remove(), size(), clear()**etc and **Collections**class contains only **static**methods like **sort(), min(), max(), fill(), copy(), reverse()** etc.

18)Comparable and Comparator

| **Sr. No.** | **Key** | **Comparable** | **Comparator** |
| --- | --- | --- | --- |
| 1 | Methods | The comparable interface has a method compareTo(Object a ) | The comparator has a method compare(Object o1, Object O2) |
| 2 | Sorting uses | Collection.sort(List) method can be used to sort the collection of Comparable type objects. | Collection.sort(List, Comparator) method can be used to sort the collection of Comparator type objects. |
| 3 | Sorting sequence | Comparable provides single sorting sequence. | The comparator provides a multiple sorting sequence. |
| 4 | Package | Comparable interface belongs to java.lang package. | Comparator interface belongs to java.util package. |

Ans- Comparable and comparator both are an interface that can be used to sort the elements of the collection. Comparator interface belongs to java.util package while comparable belongs to java.lang package. Comparator interface sort collection using two objects provided to it, whereas comparable interface compares" this" refers to the one objects provided to it.

19)What is the difference between Array and ArrayList?

## Ans- **Comparison between Array and ArrayList**

The following table highlights the major differences between an Array and an ArrayList −

| **Basis of Comparison** | **Array** | **Array List** |
| --- | --- | --- |
| Definition | A straightforward data structure with a continuous memory location, an array stores its contents with the same name but distinct index numbers for each element of the array it contains. It is imperative that all of the data stored in an array be of the same type. After an array has been declared, its size cannot be changed. | The Java collection framework contains a data structure known as an ArrayList, which is dynamic in nature. Additionally, it has components that are of the same type. In this case, it is not necessary for us to specify the length of the list. |
| Static/Dynamic | Arrays are static | ArrayList is dynamic |
| Resizable | Fixed Length | Can be Resizable |
| Initialization | When performing initialization for an array, it is required to specify the size of the array. | It is not necessary to mention the size of an ArrayList. |
| Performance | Arrays are faster | ArrayList is slower |
| Generic Type | An array can store primitive data as well as objects, but it cannot store generics. | ArrayList is able to store generics as well as objects, but it cannot store data of primitive types. |
| Iteration | Only loops are permitted in this area. | It is acceptable to use loops and iterators. |
| Type Safety | It is not type-safe. | It is type-safe. |
| Length | Makes use of the length object | Make use of size() function |
| Adding Elements | The additions are done with the assignment operator. | ArrayList uses add() method for performing additions |
| Single/Multi- Dimensional | Single and multiple dimensions are also a possibility. | You are only permitted to use single dimension. |

20)Why we override equals() & hashcode() method

Ans- HashMap and HashSet use the hashcode value of an object to find out how the object would be stored in the collection, and subsequently hashcode is used to help locate the object in the collection.

21)How to synchronize List, Set and Map elements

Ans- To synchronize ArrayList, we use Collections. synchronizedList() method. This method returns synchronized list backed by the specified list. There is an advise from javadocs that while iterating over the synchronized list, you must use it in a synchronized block.

22)What is the advantage of generic collection

Ans- The generic collections are introduced in **Java 5 Version.** The generic collections**disable**the**type-casting** and there is no use of **type-casting** when it is used in generics. The generic collections are**type-safe**and checked at **compile-time**. These generic collections allow the datatypes to pass as parameters to classes. The **Compiler**is responsible for checking the **compatibility**of the types.

23)How to convert ArrayList to Array and Array to ArrayList?

Ans- We can convert an array to arraylist using following ways.

* **Using Arrays.asList() method** - Pass the required array to this method and get a **List**object and pass it as a parameter to the constructor of the **ArrayList**class.
* **Collections.addAll() method** - Create a new list before using this method and then add array elements using this method to existing list.
* **Iteration method** - Create a new list. Iterate the array and add each element to the list.

24)How to reverse ArrayList?

Ans - To reverse an ArrayList in java, one can **use Collections class reverse method i.e Collections.** **reverse() method**. Collections reverse method reverses the element of ArrayList in linear time i.e time complexity is O(n). Collections reverse method accepts a List type as an argument.

25)How to sort ArrayList?

Ans- **An ArrayList can be sorted by using the sort() method of the Collections class in Java**. It accepts an object of ArrayList as a parameter to be sort and returns an ArrayList sorted in the ascending order according to the natural ordering of its elements.

26)Which collection classes are synchronized or thread-safe

Ans- The collection classes that are thread-safe in Java are **Stack, Vector, Properties, Hashtable**, etc

27)What is the difference between peek(),poll() and remove() method of the Queue interface ?

Ans- The remove() and poll() methods differ only in their behavior when the queue is empty: **the remove() method throws an exception, while the poll() method returns null**. The element() and peek() methods return, but do not remove, the head of the queue.

28)Write java code showing insertion,deletion and retrieval of HashMap object ?

## Ans- **Example illustrating working with HashMap in Java**

**import** *java.util.*\*;

**class** HashMapDemo **{**

**public** **static** **void** main**(String** args**[])** **{**

// Create a hash map.

HashMap**<String**, **Double>** hm = new HashMap**<String**, **Double>()**;

// Put elements to the map

hm.put**(**"John Doe", new **Double(**3434.34**))**;

hm.put**(**"Tom Smith", new **Double(**123.22**))**;

hm.put**(**"Jane Baker", new **Double(**1378.00**))**;

hm.put**(**"Tod Hall", new **Double(**99.22**))**;

hm.put**(**"Ralph Smith", new **Double(**-19.08**))**;

// Get a set of the entries.

Set**<**Map.Entry**<String**, **Double>>** set = hm.entrySet**()**;

// Display the set.

**for(**Map.Entry**<String**, **Double>** me : set**)** **{**

System.out.print**(**me.getKey**()** + ": "**)**;

System.out.println**(**me.getValue**())**;

**}**

System.out.println**()**;

//Removing an element from HashMap

hm.remove**(**"Tod Hall"**)**;

// Deposit 1000 into John Doe's account.

**double** balance = hm.get**(**"John Doe"**)**;

hm.put**(**"John Doe", balance + 1000**)**;

System.out.println**(**"John Doe's new balance: " +

hm.get**(**"John Doe"**))**;

**}**

**}**

29)What is the difference between HashMap and ConcurrentHashMap ?

Ans- **the HashMap is a non-synchronized and non-Thread safe, while the ConcurrentHashMap is a synchronized and Thread-safe collection class**. Though the ConcurrentHashMap can not match the synchronization level of Hashtable, it performs well for most of the practical cases.

30) Why Map interface does not extend the Collection interface in Java Collections Framework ?

Ans- **Because they are of an incompatible type**. List, Set and Queue are a collection of similar kind of objects but just values where a Map is a collection of key and value pairs.

31)Write the code for iterating the list in different ways in java ?

Ans- **Methods:**

1. Using loops (Naive Approach)
   * For loop
   * [For-each loop](https://www.geeksforgeeks.org/for-each-loop-in-java/)
   * While loop
2. Using[Iterator](https://www.geeksforgeeks.org/iterators-in-java/)
3. Using [List iterator](https://www.geeksforgeeks.org/arraylist-listiterator-method-in-java-with-examples/)
4. Using lambda expression
5. Using [stream.forEach()](https://www.geeksforgeeks.org/stream-foreach-method-java-examples/)

32)Suppose there is an Employee class. We add Employee class objects to the ArrayList.

33)Mention the steps need to be taken , if I want to sort the objects in ArrayList using the

employeeId attribute present in Employee class.

### Ans- Different Ways  to Sort an ArrayList of Objects by Property

1. Using Comparator interface
2. Using Comparable interface

34)How can an ArrayList be synchronized without using Vector?

Ans- **To synchronize an ArrayList, we can use two JDK-provided methods.**

1. Collections. synchronizedList() method that returns a synchronized list backed by the specified list.
2. CopyOnWriteArrayList class that is a thread-safe variant of ArrayList.

35)What is NavigableMap in Java? What is a benefit over Map?

Ans- NavigableMap is an extension of the SortedMap collection framework. It is **used to arrange the elements in a uniform fashion**. NavigableMap has different methods to iterate over the elements in the Map. The NavigableMap interface is a member of the Java Collection Framework. It belongs to java. util package and It is an extension of SortedMap which **provides convenient navigation methods like lowerKey, floorKey, ceilingKey and higherKey, and along with this popular navigation method**.

36)What is Java Priority Queue ?

Ans- A priority queue in Java is **a special type of queue wherein all the elements are ordered as per their natural ordering or based on a custom Comparator supplied at the time of creation**. **A PriorityQueue is used when the objects are supposed to be processed based on the priority**. It is known that a Queue follows the First-In-First-Out algorithm, but sometimes the elements of the queue are needed to be processed according to the priority, that's when the PriorityQueue comes into play.