COLLECTIONS:

1. What are the main differences between array and collection?

Arrays:

*Arrays are fixed in size that is once we create an array we can not increased or decreased based on our requirement.

*With respect to memory Arrays are not recommended to use.

*With respect to performance Arrays are recommended to use.

*Arrays can hold only homogeneous data types elements.

*There is no underlying data structure for arrays and hence ready made method support is not available.

*Arrays can hold both object and primitive.

Collection:

*Collection are growable in nature that is based on our requirement. We can increase or decrease of size.

*With respect to memory collection are recommended to use.

*With respect to performance collection are not recommended to use.

Collection can hold both homogeneous and and heterogeneous elements.

*Every collection class is implemented based on some standard data structure and hence for every requirement ready made method support is available being a performance. we can use these method directly and We are not responsible to implement these methods.

*Collection can hold only object types but primitive.

2. Explain various interfaces used in Collection framework?

Java Collection framework provides many interfaces **(Set, List, Queue, Deque)** and classes (ArrayList, Vector, LinkedList, PriorityQueue, HashSet, LinkedHashSet, TreeSet).

3. What is the difference between ArrayList and Vector?

Vector and ArrayList both uses Array internally as data structure. They are dynamically resizable. ... But, **ArrayList increases by half of its size when its size is increased**. Therefore as per Java API the only main difference is, Vector's methods are synchronized and ArrayList's methods are not synchronized

4. What is the difference between ArrayList andLinkedList?

*ArrayList internally uses a dynamic array to store the elements.

*LinkedList internally uses a doubly linked list to store the elements. ...
*LinkedList class can act as a list and queue both because it implements List and Deque interfaces.

*__ArrayList is better for storing and accessing data__.

5. What is the difference between Iterator and ListIterator?

Iterator can traverse only in forward direction whereas **ListIterator traverses both in forward and backward directions**.

ListIterator can help to replace an element whereas Iterator cannot.

6. What is the difference between List and Set?

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 duplicate elements.
2.	The insertion order is maintained by the List.	It doesn't maintain the insertion order of elemen
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, 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 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 collectio elements.
7.	The method of List interface listiterator() is used to iterate the List elements.	The iterator is used when we need to ite elements.

7. What is the difference between HashSet and TreeSet?

1	Hash set is implemented using HashTable	The tree set is implemented using a tree structure.

2	HashSet allows a null object	The tree set does not allow the null object. It throws the null pointer exception.
3	Hash set use equals method to compare two objects	Tree set use compare method for comparing two objects.
4	Hash set doesn't now allow a heterogeneous object	Tree set allows a heterogeneous object
5	HashSet does not maintain any order	TreeSet maintains an object in sorted order

8. What is the difference between HashSet andHashMap?

Basis	**HashMap**	**HashSet**
Definition	Java HashMap is a hash table based implementation of Map interface.	HashSet is a Set. It creates a collection that uses for storage.
Implementation	HashMap implements **Map, Cloneable, and Serializable** interface es.	HashSet implements **Set, Cloneable, Iterable** and **Collection** interfaces.
Stores	In HashMap we store a **key-value pair**. It maintains the mapping of key and value.	In HashSet, we store **objects**.
Duplicate values	It does not	It does not allow **duplicate values**.

	allow **duplicate keys**, but **duplicate values** are **allowed**.	
Null values	It can contain a **single null key** and **multiple null values**.	It can contain **a single null value**.
Method of insertion	HashMap uses the **put()** method to add the elements in the HashMap.	HashSet uses the **add()** method to add elem HashSet.
Performance	HashMap is **faster/ than HashSet because values are associated with a unique key.**	HashSet is **slower** than HashMap because object is used for calculating hashcode value, same for two objects.
The Number of objects	Only **one** object is created during the add operation.	There are **two** objects created during put op for **key** and one for **value**.
Storing Mechanism	HashMap internally uses **hashing** to store objects.	HashSet internally uses a **HashMap** object to st
Uses	Always prefer when we do not maintain the **uniqueness**.	It is used when we need to maintain the **unique**
Example	**{a->4, b->9, c->5}** Where **a, b, c** are **keys** and **4, 9, 5** are **values** associated with key.	**{6, 43, 2, 90, 4}** It denotes a set.

9.What is the difference between HashMap andHashtable?

- HashMap is non-synchronized. It is not thread-safe and can't be shared between many threads without proper synchronization code whereas Hashtable is synchronized. It is thread-safe and can be shared with many threads.
- HashMap allows one null key and multiple null values whereas Hashtable doesn't allow any null key or value.
- HashMap is generally preferred over HashTable if thread synchronization is not needed

10. What is the difference between Comparable and Comparator?

Comparable	**Comparator**
1) Comparable provides a **single sorting sequence**. In other words, we can sort the collection on the basis of a single element such as id, name, and price.	The Comparator provides **multiple sortin** In other words, we can sort the collectior of multiple elements such as id, name, and
2) Comparable **affects the original class**, i.e., the actual class is modified.	Comparator **doesn't affect the original c** actual class is not modified.
3) Comparable provides **compareTo() method** to sort elements.	Comparator provides **compare() met** elements.
4) Comparable is present in **java.lang** package.	A Comparator is present in the **java.util** pa
5) We can sort the list elements of Comparable type by **Collections.sort(List)** method.	We can sort the list elements of Com by **Collections.sort(List, Comparator)** meth

11. How to synchronize List, Set and Map elements?

ArrayList, *HashSet* and *HashMap* are three most frequently used data structures in java. As they are most used, they are not synchronized for the sake of performance. But, java provides the methods to make them synchronized as and when the need arises. These methods are introduced in *java.util.Collections* class. *Collections* class is an utility class which has some useful methods helpful for operations on collection types. In this post, we will see how to synchronize *ArrayList*, *HashSet* and *HashMap* in java.

12. What do you understand by fail-fast?

Fail fast is a philosophy that values extensive testing and incremental development to determine whether an idea has value. ... Failing fast seeks to take the stigma out of the word "failure" by emphasizing that the knowledge gained from a failed attempt actually increases the probability of an eventual success.

13.What is the difference between Array and ArrayList?

Basis	Array	ArrayList
Definition	An array is a dynamically-created object. It serves as a container that holds the constant number of values of the same type. It has a contiguous memory location.	The ArrayList is a class of Java Collections f contains popular classes like Vector, and HashMap.
Static/ Dynamic	Array is **static** in size.	ArrayList is **dynamic** in size.
Resizable	An array is a **fixed-length** data structure.	ArrayList is a **variable-length** data structur resized itself when needed.
Initialization	It is mandatory to provide the size of an array while initializing it directly or	We can create an instance of ArrayList witho its size. Java creates ArrayList of default size.

	indirectly.	
Performance	It performs **fast** in comparison to ArrayList because of fixed size.	ArrayList is internally backed by the array resize operation in ArrayList slows performance.
Primitive/ Generic type	An array can store both **objects** and **primitives** typ e.	We cannot store **primitive** type in A automatically converts primitive type to obje
Iterating Values	We use **for** loop or **for each** loop to iterate over an array.	We use an **iterator** to iterate over ArrayList.
Type-Safety	We cannot use generics along with array because it is not a convertible type of array.	ArrayList allows us to store only **generic/ typ it is type-safe.**
Length	Array provides a **length** variable which denotes the length of an array.	ArrayList provides the **size()** method to deter of ArrayList.
Adding Elements	We can add elements in an array by using the **assignment** operator.	Java provides the **add()** method to add ele ArrayList.
Single/ Multi-Dimensional	Array can be **multi-dimensional**.	ArrayList is always **single-dimensional**.

14. How to remove duplicates from ArrayList?

To remove the duplicates from a list, you can make **use of the built-in function set()**. The specialty of the set() method is that it returns distinct elements. You can remove duplicates from the given list by importing OrderedDictfrom collections.

15. Write a Java program to copy one array list into another?

In order to copy elements of ArrayList to another ArrayList, we use **the Collections. copy() method**. It is used to copy all elements of a collection into another. where src is the source list object and dest is the destination list object.

```
package com.demo;

import java.util.ArrayList;
import java.util.LinkedList;
import java.util.List;

public class LinkedListIterate {

	public static void main(String[] args) {

		List<Integer> list = new LinkedList<Integer>();
			list.add(56);
			list.add(25);
			list.add(64);
			list.add(25);
			list.add(12);
			list.add(53);

		System.out.println("Linked List : "+list);

		List<Integer> list1 = new ArrayList<Integer>(list);

		System.out.println("Array List : "+list1);
	}
}
```

Output:

```
Linked List : [56, 25, 64, 25, 12, 53]
Array List : [56, 25, 64, 25, 12, 53]
```

16. Write a Java program of swap two elements in an array list?

We can swap two elements of Array List using Collections.swap() method. This method accepts three arguments. The first argument is the ArrayList and the other two arguments are the indices of the elements. This method returns nothing.

//To swap two elements in an array list.

```
package com.te.demo;

import java.util.ArrayList;
import java.util.List;

public class ArraySawp {

	public static void main(String[] args) {

		List<Integer> list = new ArrayList<Integer>();
			list.add(56);
			list.add(25);
			list.add(64);
			list.add(25);
			list.add(12);
			list.add(53);
		System.out.println("Array List Before Swap = "+list);

		Collections.swap(list, 0, 4);
```

```
            System.out.println(("Array List After Swap = "+list);
        }
}
```

Output:

```
Array List Before Swap = [56, 25, 64, 25, 12, 53]
Array List After Swap = [12, 25, 64, 25, 56, 53]
```

17. Write a Java program to iterate through all elements in a linked list starting at the specified position?

LinkedList in java is basically a part of the collection framework present in java.util package. It is the implementation of the LinkedList data structure that stores elements in a non-contiguous manner inside the memory.

```
package com.te.demo;

import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;

public class LinkedListIterate {

        public static void main(String[] args) {

                List<Integer> list = new LinkedList<Integer>();
                        list.add(56);
                        list.add(25);
                        list.add(64);
                        list.add(25);
                        list.add(12);
                        list.add(53);

                System.out.println("----------Iterate using iterator()-----");
```

```
                Iterator<Integer> itr =     list.iterator();
                while (itr.hasNext()) {
                        System.out.println(itr.next());
                }
        }
}
```

Output:

```
----------Iterate using iterator()-----
56
25
64
25
12
53
```

18. Write a Java program to get the first and last occurrence of the specified elements in a linked list.

```
package com.demo;

import java.util.LinkedList;
import java.util.List;

public class LinkedListIterate {

        public static void main(String[] args) {

                List<Integer> list = new LinkedList<Integer>();
                        list.add(56);
                        list.add(25);
                        list.add(64);
                        list.add(25);
                        list.add(12);
                        list.add(53);
```

```
            System.out.println("First(Index) Occurance of 25 :
"+list.lastIndexOf(25));
            System.out.println("Last(Index) Occurance of 25 : "+list.indexOf(25));

      }
}
```

Output:

```
First(Index) Occurance of 25 : 3
Last(Index) Occurance of 25 : 1
```

19.Write a Java program to retrieve but does not remove, the first element of a linked list.

```
package com.demo;

import java.util.LinkedList;
import java.util.List;

public class LinkedListIterate {

     public static void main(String[] args) {

          List<Integer> list = new LinkedList<Integer>();
               list.add(56);
               list.add(25);
               list.add(64);
               list.add(25);
               list.add(12);
               list.add(53);


          System.out.println("Orginal List : "+list);
```

```
        int ival = list.get(0);
        System.out.println("Retriving first element :
"+ival);
        System.out.println("After Retriving first element :
"+list);

    }
}
```

Output:

```
Orginal List : [56, 25, 64, 25, 12, 53]
Retriving first element : 56
After Retriving first element : [56, 25, 64, 25, 12, 53]
```

20 .Write a Java program to convert a linked list to array list.

```
package com.demo;

import java.util.ArrayList;
import java.util.LinkedList;
import java.util.List;

public class LinkedListIterate {

    public static void main(String[] args) {

        List<Integer> list = new LinkedList<Integer>();
            list.add(56);
            list.add(25);
            list.add(64);
            list.add(25);
            list.add(12);
            list.add(53);

        System.out.println("Linked List : "+list);

        List<Integer> list1 = new ArrayList<Integer>(list);
```

```
        System.out.println("Array List : "+list1);
    }
}
```

Output:

```
Linked List : [56, 25, 64, 25, 12, 53]
Array List : [56, 25, 64, 25, 12, 53]
```