What are the two ways to iterate the elements of a collection?

What is the difference between ArrayList and LinkedList classes in collection framework?

What is the difference between ArrayList and Vector classes in collection framework?

What is the difference between HashSet and HashMap classes in collection framework?

What is the difference between HashMap and Hashtable class?

What is the difference between Iterator and Enumeration interface in collection framework?

How can we sort the elements of an object? What is the difference between Comparable and Comparator interfaces?

What does the hashcode() method?

What is the difference between Java collection and Java collections?

[How to Sort ArrayList in Java](https://www.javatpoint.com/how-to-sort-arraylist-in-java)

Collections.sort(list); Collections.sort(list, Collections. reverseOrder());

[Difference between Array and ArrayList](https://www.javatpoint.com/difference-between-array-and-arraylist)

Array is static in size. ArrayList dynamic in size.

Array can be multi-dimensional. ArrayList is always single dimensional.

An array can store both **objects** and **primitive** type. ArrayList cannot store primitive type. It automatically converts primitive type to object.

[When to use ArrayList and LinkedList in Java](https://www.javatpoint.com/when-to-use-arraylist-and-linkedlist-in-java)

ArrayList is better to access data whereas LinkedList is better to manipulate data. Both classes implements List interface.

[Difference between ArrayList and LinkedList](https://www.javatpoint.com/difference-between-arraylist-and-linkedlist)

ArrayList internally uses a **dynamic array** to store the elements. LinkedList internally uses a **doubly linked list** to store the elements.

ArrayList is **better for storing and accessing** data. LinkedList is **better for manipulating** data.

[Difference between ArrayList and Vector](https://www.javatpoint.com/difference-between-arraylist-and-vector)

ArrayList is **not synchronized**. Vector is **synchronized**.

[How to Compare Two ArrayList in Java](https://www.javatpoint.com/how-to-compare-two-arraylist-in-java)

There are following ways to compare two ArrayList in Java:

Java **equals()** method

**boolean** boolval = firstList.equals(secondList);  //returns true because lists are equal

Java **removeAll()** method

secondList.removeAll(firstList);

Java **retainAll()** method

secondList.retainAll(firstList);

Java **ArrayList.contains()** method

secondList.contains(tempList) //tempList is a variable

Java **contentEquals()** method

//converts List into String and checks string are equal or not

return ls1.toString().contentEquals(ls2.toString())?true:false;

Java **Stream** interface

Common elements: - firstList.stream().filter(secondList::contains).collect(Collectors.toList())

[How to reverse ArrayList in Java](https://www.javatpoint.com/how-to-reverse-arraylist-in-java)

Collections.reverse(list);

[How to make ArrayList Read Only](https://www.javatpoint.com/how-to-make-java-arraylist-read-only)

 List<String>unmodifiableList= Collections.unmodifiableList(List);

[How to Synchronize ArrayList in Java](https://www.javatpoint.com/how-to-synchronize-arraylist-in-java)

// Synchronizing ArrayList in Java

        furitList = Collections.synchronizedList(fruitList);

        // we must use synchronize block to avoid non-deterministic behavior

**synchronized** (fruitList) {

            Iterator<String> itr = fruitList.iterator();

**while** (itr.hasNext()) {

                System.out.println(itr.next());

            }

        }

[How to convert ArrayList to Array and Array to ArrayList in java](https://www.javatpoint.com/how-to-convert-arraylist-to-array-and-array-to-arraylist-in-java)

Converting ArrayList to Array

String[] item = fruitList.toArray(**new** String[fruitList.size()]);

Converting Array to ArrayList

List<String>l2 = **new** ArrayList<>();

         l2 =  Arrays.asList(item);

[How to remove duplicates from ArrayList in Java](https://www.javatpoint.com/how-to-remove-duplicates-from-arraylist-in-java)

Convert ArrayList to HashSet.

Set<String> s = **new** LinkedHashSet<String>(list);

How to Traverse the list of elements in reverse order for LinkedList

Iterator it = list.descendingIterator()

HashSet

HashSet stores the elements by using a mechanism called **hashing.**

HashSet contains unique elements only.

HashSet allows null value.

HashSet class is non synchronized.

HashSet doesn't maintain the insertion order. Here, elements are inserted on the basis of their hashcode.

HashSet is the best approach for search operations.

The initial default capacity of HashSet is 16, and the load factor is 0.75.

LinkedHashSet

Java LinkedHashSet class contains unique elements only like HashSet.

Java LinkedHashSet class provides all optional set operation and permits null elements.

Java LinkedHashSet class is non synchronized.

Java LinkedHashSet class maintains insertion order.

TreeSet

Java TreeSet class contains unique elements only like HashSet.

Java TreeSet class access and retrieval times are quite fast.

Java TreeSet class doesn't allow null element.

Java TreeSet class is non synchronized.

Java TreeSet class maintains ascending order.

ArrayDeque class

The ArrayDeque class provides the facility of using deque and resizable-array. It inherits AbstractCollection class and implements the Deque interface.

The important points about ArrayDeque class are:

* Unlike Queue, we can add or remove elements from both sides.
* Null elements are not allowed in the ArrayDeque.
* ArrayDeque is not thread safe, in the absence of external synchronization.
* ArrayDeque has no capacity restrictions.
* ArrayDeque is faster than LinkedList and Stack.

**Map Interface**

HashMap is the implementation of Map, but it doesn't maintain any order.

LinkedHashMap is the implementation of Map. It inherits HashMap class. It maintains insertion order.

TreeMap is the implementation of Map and SortedMap. It maintains ascending order.

**HashMap Class**

* Java HashMap contains values based on the key.
* Java HashMap contains only unique keys.
* Java HashMap may have one null key and multiple null values.
* Java HashMap is non synchronized.
* Java HashMap maintains no order.
* The initial default capacity of Java HashMap class is 16 with a load factor of 0.75.

**Difference between HashSet and HashMap class**

In HashMap we store a **key-value pair**. It maintains the mapping of key and value. In HashSet, we store **objects**.

It does not allow **duplicate keys**, but **duplicate values** are **allowed**. It can contain **a single null value**.

HashMap is **faster/ than HashSet because values are associated with a unique key.** HashSet is **slower** than HashMap because the member object is used for calculating hashcode value, which can be same for two objects.

**Difference between HashMap and TreeMap**

Java **HashMap** is a hashtable based implementation of Map interface. Java **TreeMap** is a Tree structure-based implementation of Map interface.

HashMap allows a **single** null key and **multiple** null values. TreeMap does not allow **null** keys but can have **multiple** null values.

HashMap does not maintain any order. The elements are sorted in **natural order** (ascending).

**LinkedHashMap Class**

* Java LinkedHashMap contains values based on the key.
* Java LinkedHashMap contains unique elements.
* Java LinkedHashMap may have one null key and multiple null values.
* Java LinkedHashMap is non synchronized.
* Java LinkedHashMap maintains insertion order.
* The initial default capacity of Java HashMap class is 16 with a load factor of 0.75.

**TreeMap class**

* Java TreeMap contains values based on the key. It implements the NavigableMap interface and extends AbstractMap class.
* Java TreeMap contains only unique elements.
* Java TreeMap cannot have a null key but can have multiple null values.
* Java TreeMap is non synchronized.
* Java TreeMap maintains ascending order.

**Hashtable class**

* A Hashtable is an array of a list. Each list is known as a bucket. The position of the bucket is identified by calling the hashcode() method. A Hashtable contains values based on the key.
* Java Hashtable class contains unique elements.
* Java Hashtable class doesn't allow null key or value.
* Java Hashtable class is synchronized.
* The initial default capacity of Hashtable class is 11 whereas loadFactor is 0.75.

**Difference between HashMap and Hashtable**

HashMap and Hashtable both are used to store data in key and value form. Both are using hashing technique to store unique keys. /// Similarities

HashMap is **non synchronized**. It is not-thread safe and can't be shared between many threads without proper synchronization code. Hashtable is **synchronized**. It is thread-safe and can be shared with many threads.

HashMap **allows one null key and multiple null values**. Hashtable **doesn't allow any null key or value**.

HashMap is **fast**. Hashtable is **slow**.

We can make the HashMap as synchronized by calling this code

**Map m = Collections.synchronizedMap(hashMap);**

**Hashtable** is internally synchronized and can't be unsynchronized.

**Java EnumSet Class**

Java EnumSet class is the specialized Set implementation for use with enum types. It inherits AbstractSet class and implements the Set interface.

**Java Comparable interface**

Java Comparable interface is used to order the objects of the user-defined class. This interface is found in java.lang package and contains only one method named compareTo(Object). It provides a single sorting sequence only, i.e., you can sort the elements on the basis of single data member only. For example, it may be rollno, name, age or anything else.

* positive integer, if the current object is greater than the specified object.
* negative integer, if the current object is less than the specified object.
* zero, if the current object is equal to the specified object.

**Java Comparator interface**

Java Comparator interface is used to order the objects of a user-defined class.

This interface is found in java.util package and contains 2 methods compare(Object obj1,Object obj2) and equals(Object element).

It provides multiple sorting sequences, i.e., you can sort the elements on the basis of any data member, for example, rollno, name, age or anything else.

1. Comparator<Student> cm1=Comparator.comparing(Student::getName);
2. Collections.sort(al,cm1);

**Difference between Comparable and Comparator Interface**

* 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 sorting sequences. In other words, we can sort the collection on the basis of multiple elements such as id, name, and price etc.

* Comparable affects the original class, i.e., the actual class is modified.

Comparator doesn't affect the original class, i.e., the actual class is not modified.

* Comparable provides compareTo() method to sort elements.

Comparator provides compare() method to sort elements.

* Comparable is present in java.lang package.

A Comparator is present in the java.util package.

* We can sort the list elements of Comparable type by Collections.sort(List) method.

We can sort the list elements of Comparator type by Collections.sort(List, Comparator) method.

What all collection you used in your project?

List 🡪 ArrayList, LinkedList, CopyOnWriteArrayList

Set 🡪 HashSet, LinkedHashSet, TreeSet, CopyOnWriteHashSet

Map 🡪 HashMap, LinkedHashMap, TreeMap, ConcurrentHashMap