ArrayList

An array that automatically resizes.

default size 10

grows into 1.5 times + 1 - old items copies to new place, old place eligible for GC

shrinks in size on deletion too

https://www.youtube.com/watch?v=IEqvmsqjpT0 (Derek Banas)

add(index, value) – inserts a new value at that position. Doesn’t replace.

set(index, value) – replaces value in index.

HashMap

default size 16

An array of linkedlists, each node of linkedlist is an instance of the class Entry

load factor 0.75 when this size is reached, the hashmap is doubled.

Not synchronized.

One null key, multiple null values allowed.

Use Iterator to iterate.

Hashtable (legacy)

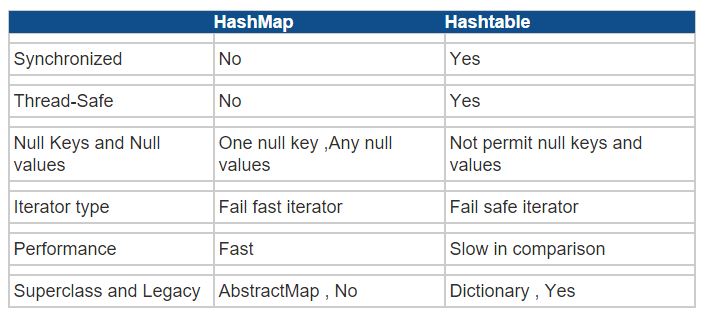
https://www.youtube.com/watch?v=B4vqVDeERhI (Derek Banas)

Array. Value stored in index given by a hash function. Collision? find next available slot from index.

Synchronized.

No null key, null values allowed.

Use Enumerator to iterate.



ConcurrentHashMap

Very high concurrency. It is thread safe without synchronizing the whole map. Locking at bucket level.

Collections.synchronizedMap(Map<K,V> map)

Collections.synchronizedList(List<K > list)

Collections.synchronizedSet(Set<K > set)

Collections.synchronizedSortedMap(SortedMap<K,V> map)

Collections.synchronizedSortedSet(SortedSet<K > set)

A function to synchronize an un-synchronized implementation of the corresponding argument. (Synchronizes it at an object level. Not at a bucket level)

LinkedHashMap

Order of insertion remains the same.

Iterator - Interaface vs Enumeration - Interaface (legacy)

Both iterate over Collections.

Iterator can remove while traversing. Enumeration can't.

Iterator doesn't allow ConcurrentModificationException - it's synchronized. Enumeration isn't.

Iterator functions – hasNext(), next(), remove()

Enumeration functions – hasMoreElements(), nextElement()

Use Enumeration when you want a read-only access to a Collection object.

Comparable - Interface vs Comparator – Interface

Comparable is to be used with a custom object, or when you need a custom sorting algorithm. You implement the Comparable interface in your own class, and overwrite the compareTo(Object o ) method and make it return -1 or 0 or 1. You can then call Collections.sort(yourClassObjectList) which will sort your custom objects for you in the way you defined.

Comparator is used for implementing custom sorting (or comparison) algorithms on predefined classes like String, etc. You need to define compare(Object o1, Object o2) in this case.

SortedMap – Interface

Provides a total ordering of the contents wrt the key in their natural order or the order specified by a Comparator.

TreeMap – implements SortedMap

The contents are ordered in the natural order of their keys or according to the Comparator provided at creation.

Set – Interface

A Collection that contains no duplicate elements. Only one null element.

SortedSet - Interface

Provides a total ordering of the elements in their natural order or the order specified by a Comparator.

TreeSet – implements SortedSet apart from Set

The contents are ordered in the natural order or according to the Comparator provided at creation.

HashSet – implements just Set

Set backed by a HashMap. Order not preserved.

LinkedHashSet – implements just Set

Set backed by a HashMap and a doublylinkedlist. Insertion order *is* preserved.

Map - Interface

Vector

AbstractMap

AbstractList