Agenda

→ Comparable vs Comparators

→ Collection·Framework .

4 1 2 5 3

↓

1 2 3 4 5]

Natural sorting order

↓

'e' 'f' 'd' 'a' 'b' } → alphabetical order

```
class Student {
    gradYear;
    age;
    psp;
    :
}
```

st1 ____

st2 ____

st3 ____

⇑

Sort these student objects .

Natural order ? ✗

↓

Comparable

any sorting algo → always comparing 2 things at a time

comparable

↓

compareTo

↓

natural sorting order

obj1.compareTo(obj2)

comparator

↓

compare

↓

explicit sorting order.

compare(obj1, obj2);

collection Framework

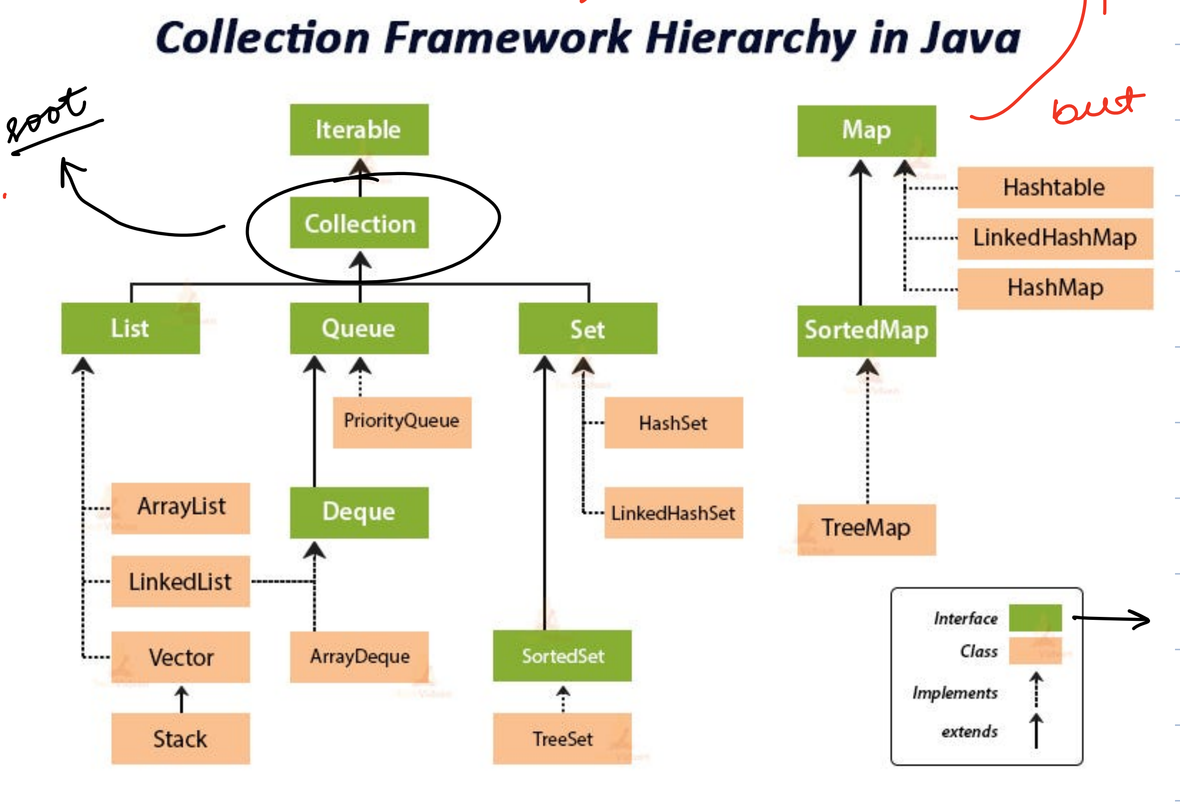
array ↑

↓

group of something

↓

multiple classes & interfaces that implements commonly used DS .



root

part of

but doesn't implement Collection interface

<< Collection >>

common methods.

size()

add()

List :- stores the elements in a sequence.

insertion order preserved.

fast random access → index.

allow duplicates.

ArrayList X

↓

dynamic array

O(1) random access.

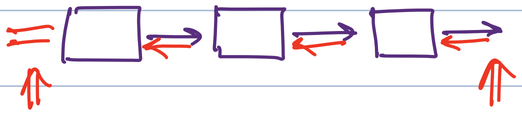
↓

Arrays

LinkedList X not thread-safe

↓

implementation of doubly linked list DS.



legacy

Vector

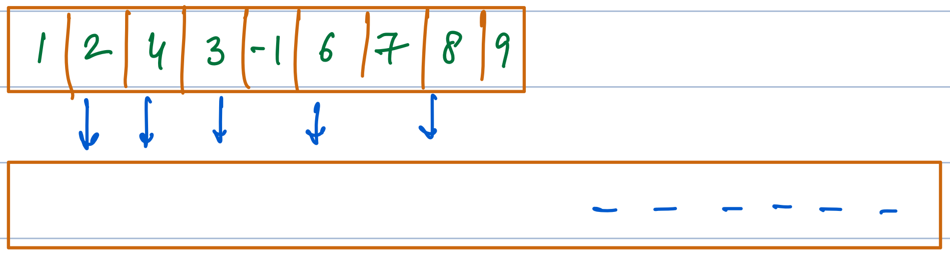
dynamic array.

synchronized

thread-safe

↑

Stack



Queue → collection where you keep elements to make them wait for turn

↓

FIFO

↑

Deque

↑ ↑

LinkedList ArrayDeque

PriorityQueue

↑

you can decide the priority via natural sorting order or comparator.

Set: doesn't contain duplicate.

↳ Hashing → search becomes better.

HashSet

randomOrder

Hashing + arrays

TreeSet

Sorted Order

BBST:

Red Black Trees

↓

$O(H) \log_2 N$

LinkedHashSet

Insertion Order

+ HM

[] ⇌ [] ⇌ [] ⇌ []

HashMap

Key, Value

NULL

Map: Key, Value

HashMap, TreeMap, LinkedHashMap

HashMap

✗

one null-key,

multiple

null values.

HashTable

thread-safe

ConcurrentHashMap

thread-safe.

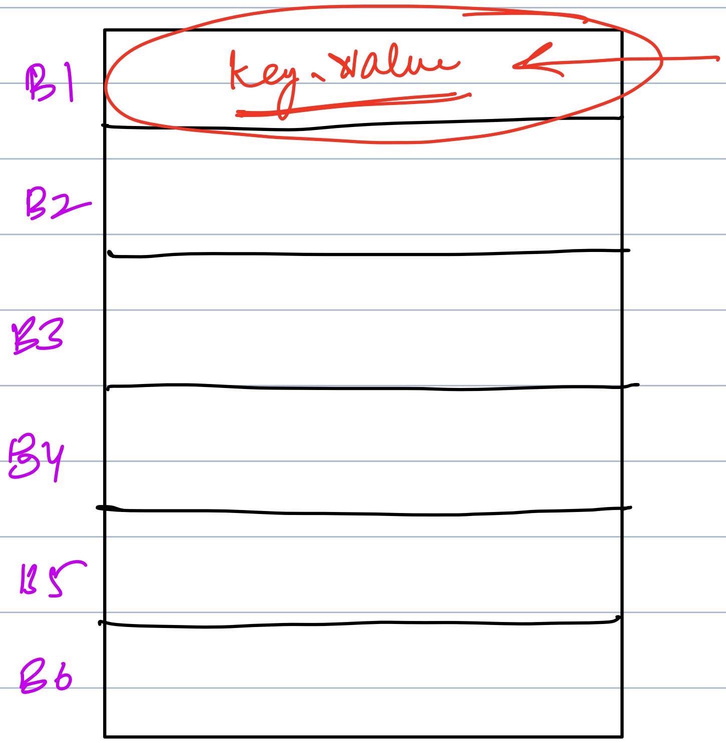
No null keys

& no null values

Synchronized

↓

block the complete Hashtable



Bucket + CAS

<< Collection >>

Collections

↓

class

sort