Collections:

It is growable in nature.

1. List 🡺 Interfaces
2. Duplicates are allowed
3. Null values are allowed
4. Insertion order.

* ArrayList 🡪 implantation class of list

1. Whenever you want to fetch data from the list it is best choice 🡺 Random Access Interface
2. You should not use when adding and deleting data at the middle of list.
3. By default size is 10.
4. Threshold/load factor 🡺 75%/0.75 🡺 new capacity = current capacity \*3/2+1
5. Asynchronous

1,2,3,4,5,11,6,7,8,9,10

* LinkedList

1. Whenever you have addition and deletion of data in the middle of list, you can use linkedlist
2. You should not use when you want fetch the data from list.

* Vector 🡺 Synchronous 🡺 size is 10 🡺threashold 75%==> new capacity = current capacity \* 2

Diagram

Description automatically generated

1. Set 🡺 Interfaces
2. It will remove duplicates.
3. Null values not allowed.
4. Search operations is the best choice

* HashSet 🡪 classes

1. Data will be print in random order

* TreeSet -🡪 Classes

1. Data will be print in sorted order

* LinkedHashSet -🡪 Classes

1. Data will be print in insertion order
2. Map 🡺 Interfaces
3. Data will be in key value pair
4. Keys will be unique and only 1 null key is allowed.
5. Values can be duplicated, and null allowed.
6. You can’t iterate map directly, it should be converted to set interface.

* HashMap

1. Data will be random order

* LinkedHashMap

1. Data will be in insertion order

* TreeMap

1. Data will be ascending order

ConcurrentHashMap:

Implementation **HashMap**:

* Default size will be 16 index.
* Each index value we will call as bucket.
* Each index is internally a linkedlist.

Map<Integer, String> map = new HashMap<>();

map.put(1,”Hi”); 🡺 hashcode will be calculated for key 🡺1 🡺 12345678 🡺index = 12345678%16 🡺 index value will be 0-15 🡺 1 🡺 in 1st index hashcode, key,value, address of next node

map.put(4,”Hello”)

map.put(29,”Sunny”);

map.get(29); 🡺 63281940 🡺 4 index

