Collections:

* It is growable nature
* We can add objects data.

1. List 🡪 Interface
2. It allows adding duplicate data
3. It allows Null values to be added
4. It has insertion order

* ArrayList

1. Whenever your operations are fetching arrayList is best choice. 🡺 RandomAccess Interface
2. Whenever you want to add data or delete data In the middle of the list arraylist is worst choice.
3. By default size is 10.
4. It is having load factor/Threshold= 75%/0.75 🡺
5. new capacity = current capacity\*3/2+1 🡺 16
6. It is asynchronous

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

* LinkedList

1. Whenever we have adding and deleting of data in the middle of the list then linked list is best choice.
2. Whenever we have fetch operations linked list is worst choice.
3. Asynchronous

* Vector

1. Synchronous
2. Default size is 10.
3. Load factor is 75% or 0.75
4. New capacity = 2\* current capacity
5. Set 🡪 Interface
6. It wont add duplicates
7. Null vales not allowed to be added.
8. Asynchronous
9. Search operations

* HashSet

1. It will print based on hash value and mostly it will be random order

* LinkedHashSet

1. It will print data in insertion order.

* TreeSet

1. It will print data in Sorting order.
2. Map 🡪 Interface
3. Map is the Key value pair
4. Keys will be unique, only 1 null key is allowed
5. Values can be null, duplicate values.
6. It is Asynchronous.

* HashMap 🡺 random order

1. By default size is 16. Each index size is called as bucket, each bucket is a internally linked list

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

map.put(19,”Aman”); 🡺 key will be fetched and hash code will be calculated internally 🡺12345678 = hashcode 🡺 hashcode%16 = index number (0-15) 🡺 12345678%16 = 4 🡺 key,value,hashcode, address of next node

map.put(29,sunny); 🡺 12345676 🡺 index value 4

map.get(29); 🡺 sunny 🡺 12345676 🡺 index value 4

* LinkedHashMap 🡺 insertion order
* TreeMap 🡺 sorting order
* HashTable 🡺 synchonous 🡺 no null keys allowed and values also. Keys will be unique and values can be duplicate.
* ConsurrentHashMap 🡺 for adding/ updating/removing 🡺 synchronous

For fetching it will be asynchronous.

Diagram

Description automatically generated