Collections:

* List 🡺 Interface

1. Duplicates are allowed
2. Null values are allowed
3. Insertion order
4. ArrayList 🡺 Asynchronous
5. Whenever we have fetch operations, we need to use it 🡺 it have internally implements Random access interface.
6. Addition or deletion operations in the middle of the list.
7. By default, capacity is 10, load factor/ threshold limit = 0.75 or 75%
8. New capacity = current capacity \* 3/2+1 = 10\*3/2+1 = 16
9. LinkedList
10. Addition or deletion operations into the list.
11. Whenever we have fetch operations, we do not use it.
12. Vector
13. New capacity=2 current capacity 🡺 after 75% filling
14. Synchronous

* Set 🡺 interface

1. Duplicates will be removed
2. Null values not allowed
3. Search operation it is best
4. HashSet
5. It will print data in random order
6. Hashset internally uses hashmap implementations
7. LInkedHashSet
8. It wii print data in insertion order.
9. TreeSet
10. It will print data in sorting order.

* Map

1. Data will be stored in key value pair
2. Keys will be unique and 1 null value allowed
3. Values can be duplicates and null values are allowed.
4. Asynchronous.
5. HashMap
6. It will print data in random order.
7. LinkedHashMap
8. Insertion order it will print.
9. TreeMap
10. Sorting order data will print

* Hashtable

1. It is synchronous
2. It wont allowed null keys and null values
3. It wont allow duplicates.

* ConcurrentHashMap 🡺 add, delete operations it will be synchronous, get operations it will have asynchronous.
* Fail safe and fail fast iterators

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

Map<Integer, String> map = new HashMap<>(); // deafulat 16 🡺each index is bucket🡺 each bucket is linkedlist internally.

Map.put(1,”chandu”); 🡺 hashcode will get calculated for key 1 🡺 hashcode 1 = 12345678 🡺 12345678%16 🡺 10

Map.put(2,”pranith”); 🡺 hashcode 23456781 🡺 10 🡺

Map.put(3,”shiva”); 🡺 index 2

Map.get(2); 🡺 pranith

Threshold hold limit 8 🡺 load factor is 8 🡺 complete balanced tree

