Collections:

* List

1. It will have insertion order
2. Duplicates are allowed
3. Null values are allowed
4. Java.util.\* package
5. ArrayList (Asynchronous)
6. Whenever your having operation like fetching data 🡺 Random Access Interface
7. Whenever you want to add or delete data in middle arraylist is worst choice
8. Default size capacity is 10 🡺 Threshold /load factor = 75% or 0.75

New capacity = Current capacity \* 3/2+1= 16

1. LinkedList
2. Whenever you’re having operations like adding or deleting then use it
3. Whenever you want to fetch the data linkedlist worst choice
4. Vector (deprecated)
5. Synchronous
6. Default size capacity is 10 🡺 Threshold /load factor = 75% or 0.75
7. New capacity = 2 \* current capacity

* Set

1. Set will remove all the duplicates.
2. If your operations are search, then you can use set interface
3. Null values won’t allow
4. HashSet
5. HashSet internally uses HashMap. All the data in Hashset will store as keys in HashMap.
6. Data will print randomly.
7. LinkedHashSet
8. LinkedHashSet uses LinkedHashMap internally. All the data in LinkedHashset will store as keys in LinkedHashMap.
9. Whenever you want to remove the duplicates and print in Insertion order.
10. TreeSet
11. TreeSet used TreeMap internally, All the data in Treeset will store as keys in TreeMap
12. Whenever you want to remove duplicates and print data in Ascending order.

* Map

1. Key value pairs
2. All the map implementation classes you can directly iterate the data, you need to convert them to set interface.
3. Asynchronous
4. HashMap
5. It will print data randomly,
6. TreeMap
7. It will print data in sorting order.
8. LinkedHashMap
9. It will print data in insertion order

* HashTable == synchronous , null values, keys not allowed
* CuncurrentHashMap🡺 add /delete it will be synchronous and get operations all are asynchronous

1,2,3,4,6,7,8,9,10 ,11,12

HashMap:

Map<Integer,String> map = new HashMap<>(); // 16 capacity

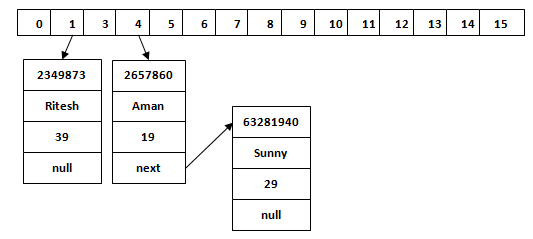
Each index value we will call it as buckets. 🡺 each bucket is internally a linkedlist.

Map.put(1,”Sujay”); 🡺 1 and calculates hashcode 🡺 12345678🡺 it calculates index value by 12345678 % 15 🡺 1

Map.put(2,”Aman”) 🡺 2 it calculates hashcode🡺 2657860🡺 index 4 🡺

Map.put(3,”sunny”)🡺 3 it calculates hashcode 🡺 63281940 🡺 index 4🡺

Map(null,data)



Map.get(3)🡺 hashcode == 63281940 🡺 index 4.

* concurrentModification exception

Fail safe and fail fast iterators:

Fail fast: hashmap concurrent modification

Fail safe: is concurrent hashmap

* Threshold/loadfactor 🡺 8 🡺 hashmap will convert into complete balanced tree