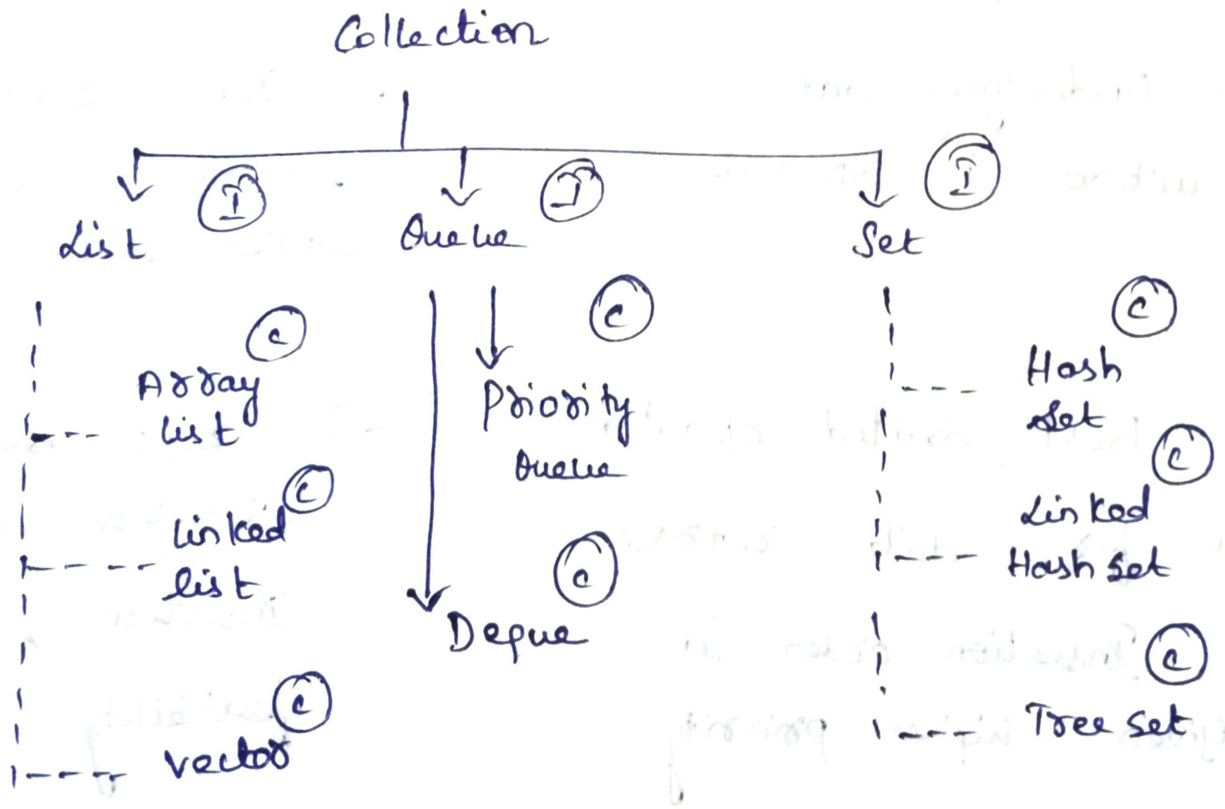
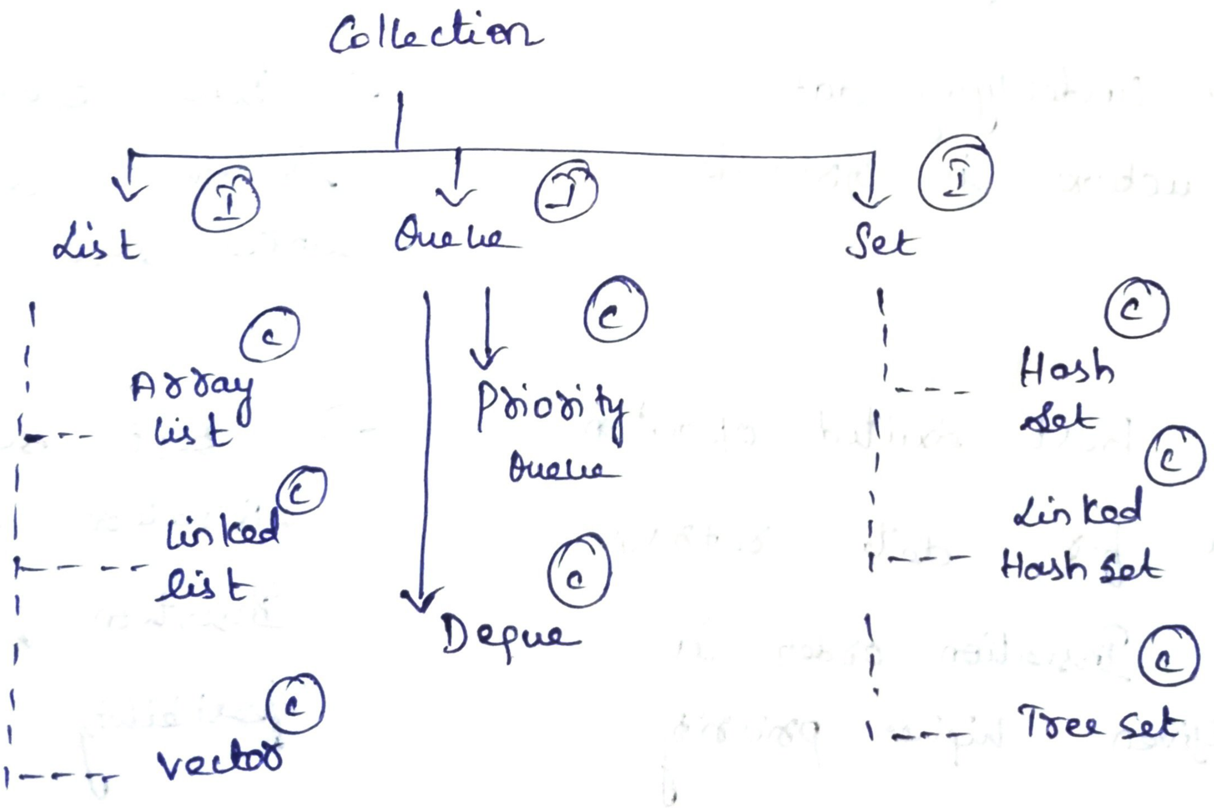
1) Array	Collection
*) Array itself a data structure and has some restriction for entering values.	*) Collection had Von as data structure available providing users for freedom to manipulation of objects
*) Array are not growable	*) Collection it is growable
*) Array elements Cannot be recorded	*) Elements Can be removed & Manipulated
*) Doesn't allow NULL	*) Allow NULL values

2)



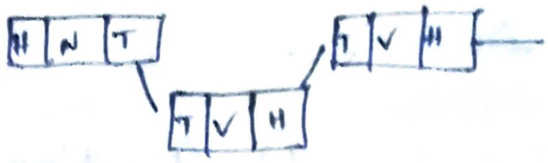
1) Array	Collection
*) Array itself a data structure and has some restriction for entering values.	*) Collection had Von as data structure available providing users for freedom to manipulation of objects
*) Array are not growable	*) Collection it is growable
*) Array elements Cannot be recorded	*) Elements Can be removed & Manipulated
*) Doesn't allow NULL	*) Allow NULL values

)



3) Array list	Vector list
-> Non-legacy class ie) from V.12	-> legacy class ie) from V.10
-> Not synchronized	-> Synchronized
-> Not thread safe	-> Thread safe
-> Initially the new capacity is meant to be (3/2 * old capacity) +1 later changed to 50% of load factor	-> Capacity is 2* old capacity ie) 100%

4) Array list	Linked list
-> Underlying data structure is Growable	-> Data structure is the doubly linked list
-> Best suited operation is for data retrival & Insertion order is given higher priority	-> Best suited operation is for Insertion & deletion feasibility

→ It implements the interfaces of serilizable, cloneable and random Access.	→  → T → Tail , N → Node , H → Head. It implements only finalizable & clonable.
5) Iterator	List iterator
→ It is the universal class that helps to fetch the data (among other things)	→ List iterator is the interface that extends iterator
→ Iterator can only print farward	→ List iterator can print farward & Backward
→ It prints from first to last	→ It can print for only specified location

6) List :-

The Child of the list are

→ Array list
→ Vector
→ linked List

-> List is adapted when,

a) Insertion order is of higher priority

b) When inserting or deleting the elements wanted to be favourite

c) When Sorting is of ~~for~~ last priori

Set :- The child of the Set are

-> Hash Set

-> Linked Hash Set

-> Tree set

Set is adopted when

a) Duplicate Value is to be removed

b) When we don't want insertion order

c) When we want to do sorting

7) Hash Set

-> It does not preserve insertion order, but allow Hetrogenous value & null pointer Acceptance

Tree Set

-> It does not allow anything

→ underlying data structure	→ underlying data structure is balanced tree
→ There is no guarantee that it will come in ascending order	→ 100% guarantee that o/p comes in ascending order.

8) Hash Set	Hash map
→ Comes under Collection	→ It doesn't belong to Collection
→ It is stored in single value	→ It is stored as key value pairs
→ add() Method is used to add values	→ Put() Method is used to add values
→ Data can be iterated through iterator directly	→ Cannot be iterated It should be Initialized set with the help of entry set method.

→ Hetrogeneous	→ yes
Duplicate	→ NO
Null Acceptance	→ 1
Insertion order	→ NO

	key	value
→ Hetrogeneous	yes	yes
Duplicate	NO	Yes
Null	1	yes
Insertion order	NO	yes

9) The main difference b/w Hashmap and Hash table is that hash map is not thread safe while latter is thread safe, ie) Hashmap is Not synchronized and Hashtable is synchronized.

Hence if we want for faster application we can go for Hashmap, if we want for thread safe application by sacrificing the speed of process, we can go for Hash table.

10) Comparable	Comparator
→ Used to compare single object	→ Compare two objects
→ Implements Compare TO method	→ Implements Compare method

|→ Comes from Java   → Comes from Java.
lang Package   util package

ii) To synchronize list we can use synchronize list ( ) method in Java

Eg:-

```
        import Java.util
        Public class program {
      PSVM ( String [] args) {
    list < integer > ref : new Arraylist ( );
        ref. add (10);
        ref. add (11);
        ref. add (15);
        ref. add (20);
  ref1 = Collections. Synchronized list (ref);
      Synchronized . (ref1) {
  Iterator <Integer > itr = ref. Iterator ( );
    while (itr. hasNext ( )) {
      S.o.p ( itr. next ( ) );
      }
  }
}
```

[illegible] that, if we use iterator it must
be use synchronized block, if we [illegible]
[illegible] print directly, instead of that ;

similar is the shown in synchronized
list and map we use synchronized set
and Map respectively

13) If we do any structural modification
to the list or set while iterating, then
JVM throws Concurrent Modification Exception
This is nothing because of fail-fast
iterators.

While we fed to add or remove any
element from Collection while a thread is
iterating over that Collection, then it a
fail-fast problem by Java JVM when i-t
throws Concurrent Modification Exception / Error.

13)

Array	Array list
→ Array itself a data structure	→ Array list works on a data structure of Resizable array
→ Does n't Accept null values.	→ Accept Null Values
→ Not Growable	→ It growable
→ Values cannot be removed	→ Values can be removed.