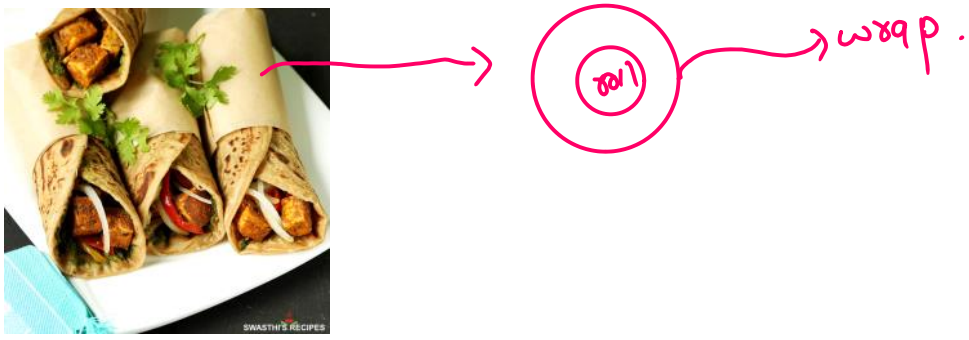
Collection

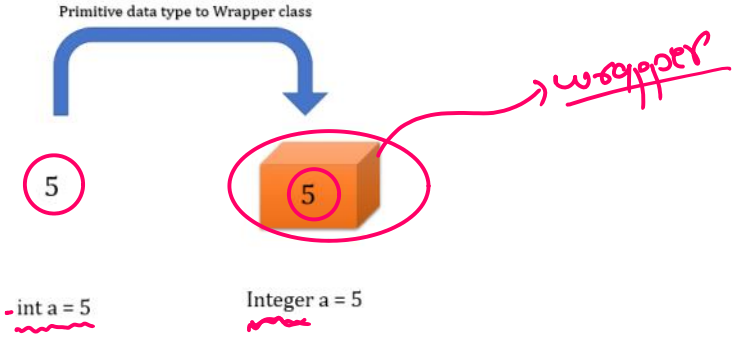
22 June 2025 09:01

1. Why we need it ?
2. Interfaces
3. implementation

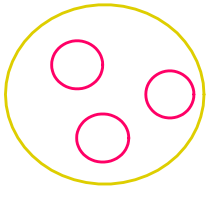
What wrapper classes ?

Wrapper classes :
Wrapper classes in java are used to wrap primitive data types





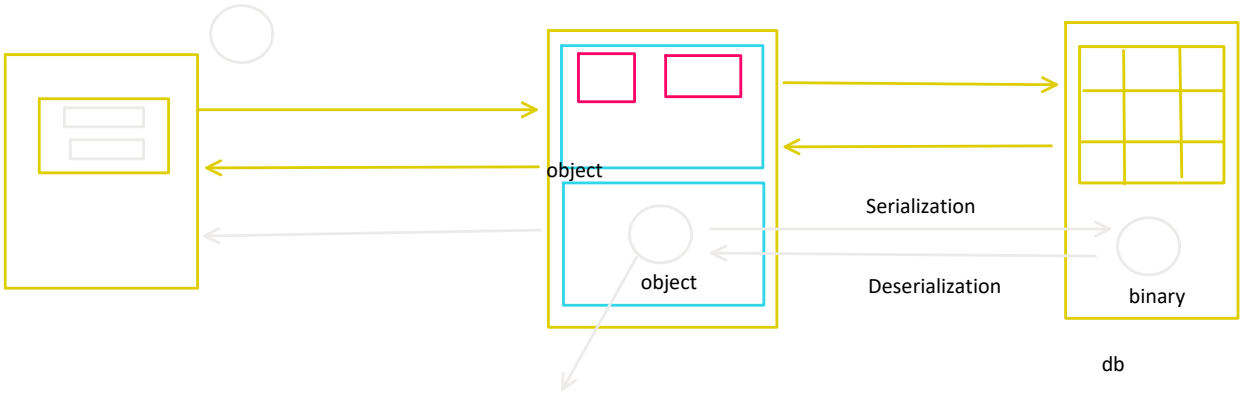
byte --> Byte
short --> Short
int -->Integer
long ---> Long
float -> Float
double --> Double
char --> Character
boolean --> Boolean



Why Wrapper classes ?

Note : Java was designed to be object - oriented , but primitive are not object so we can't perform some operations so we need wrapper classes

1. Collections : ArrayList<Integer>
2. Sterilization
3. Synchronization
4. Generics

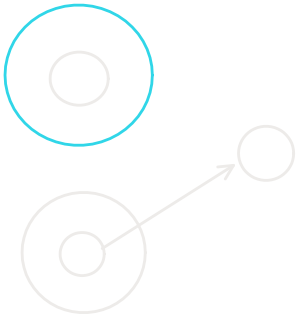


Autoboxing & Unboxing

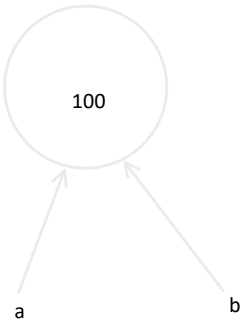
Int n=100;
Integer I = new Integer(10);

Integer i2=10; //autoboxing

Integer i=20;
Int n=I;



```
int obj1 = Integer.parseInt("2323");   //"2323"--->
```



What is collection ?
In java, collection is an object that group multiple element into single entity.

Eg: it is like container that hold multiple object into one place.

Why ?

Array is fixed size and cannot grow and shrink dynamically. But collection can grow or shrink dynamically

In Array only similar type of values we can store but in collection multiple type of object we can store

Collection core package : java.util

Collection Vs Collections

Collection ---> Super Interface for List, Set, Queue
Collections ---> utility/helper class with static methods

Collection Hierarchy

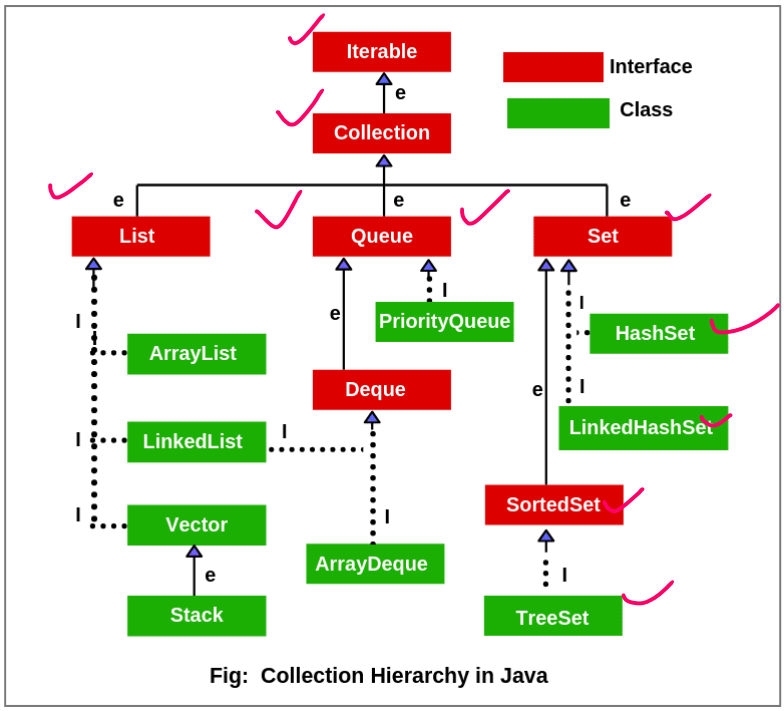
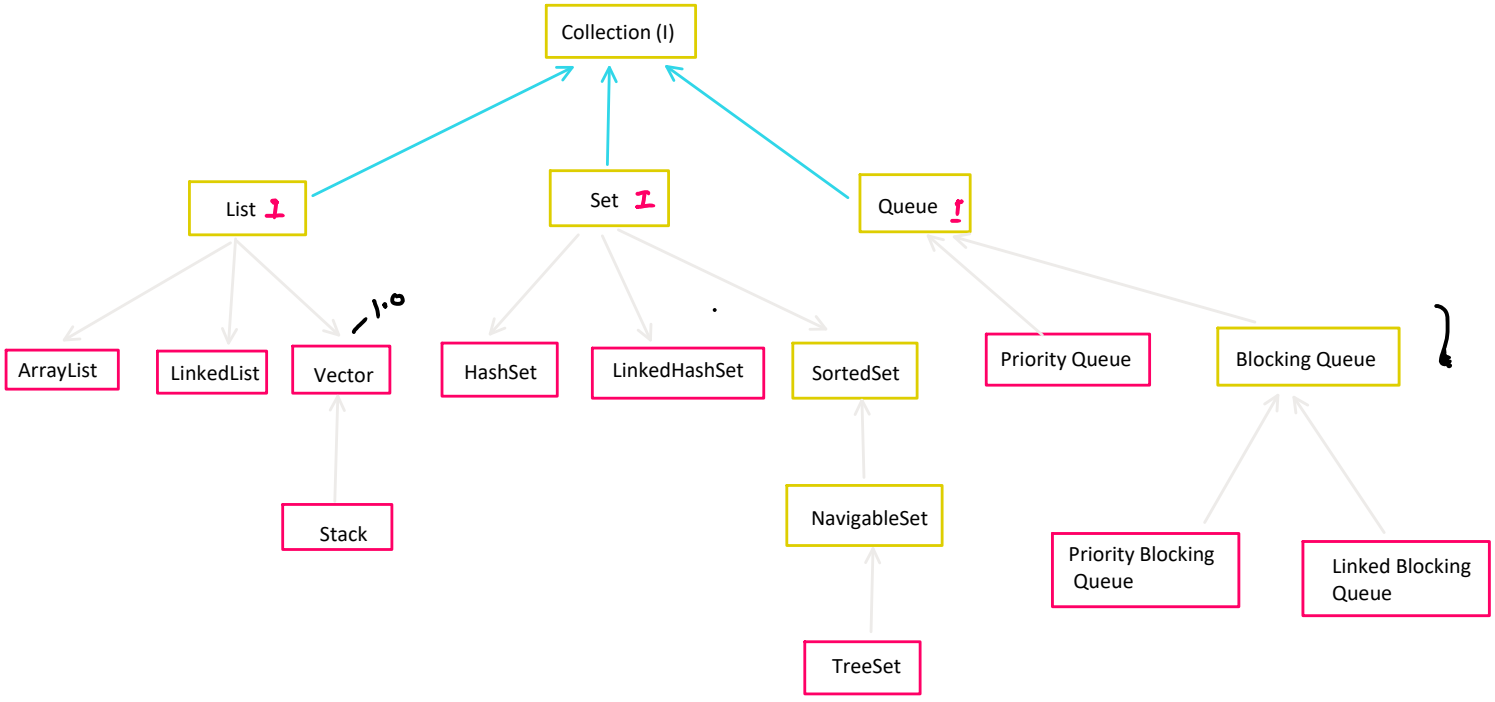


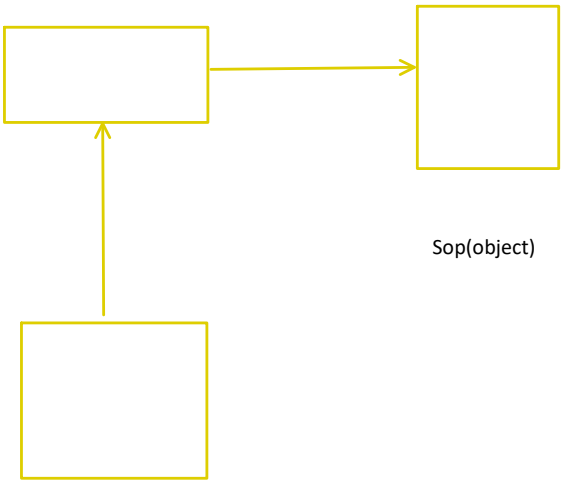
Fig: Collection Hierarchy in Java



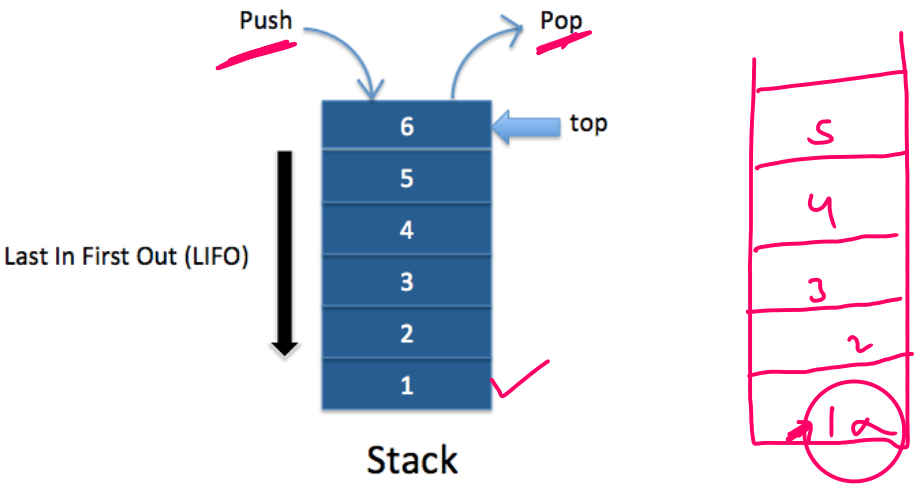
Size()
Add()
Remove()
Clear()
Add(idx,ele)
isEmpty()

Program to remove all element from list
Instruction:
Create list with 5 element
Display list is not empty
Remove all element using clear()
Display list is empty

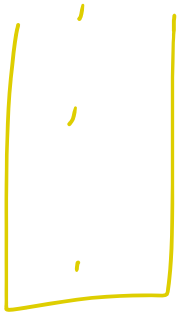
Break till 10:45

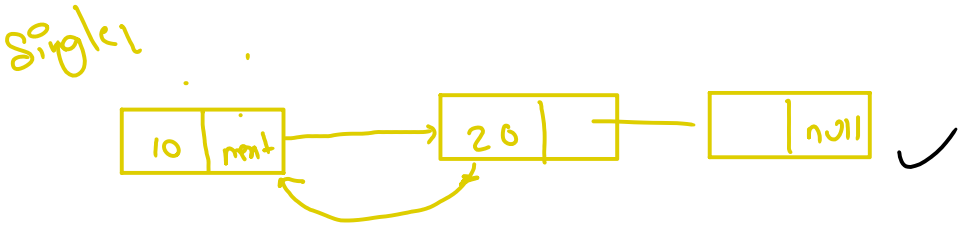


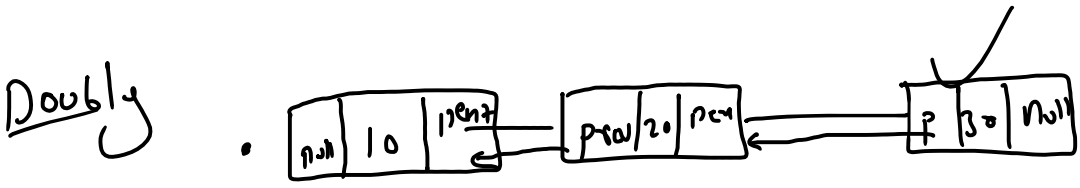
Sop(object) → toString()

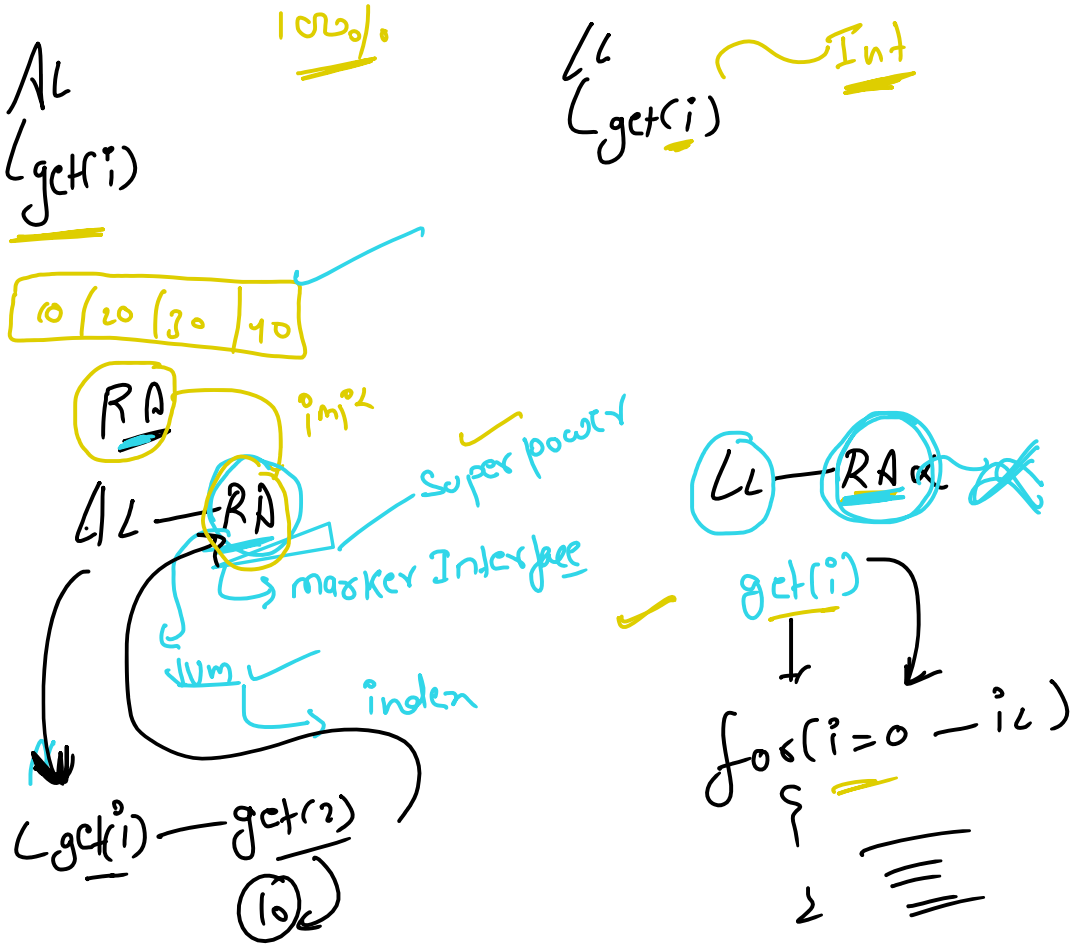


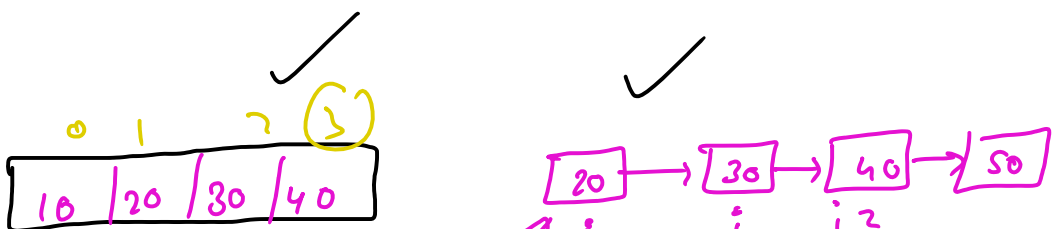
10,20,30,60,70
10 20 30 60 70
10,20,30,60,70

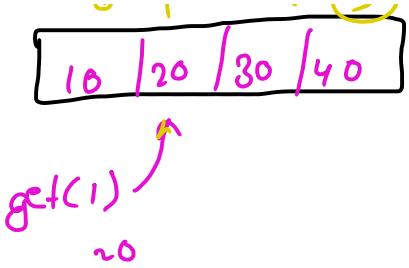


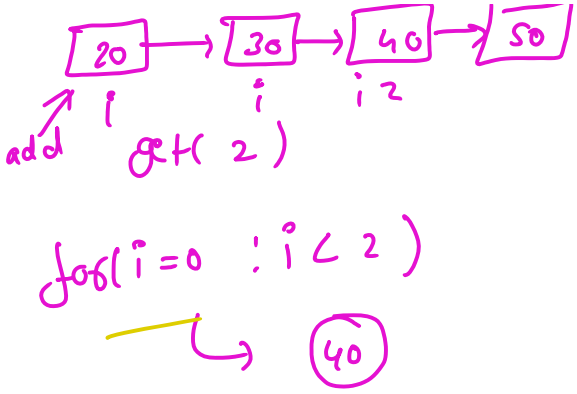

Singly
10
next
20
null

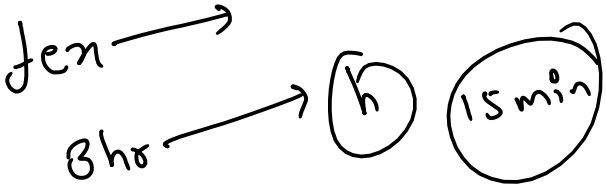

Doubly
null
10
next
prev
20
prev
30
null


AL
get(i)
10
20
30
40
RA
impl
AL
RA
super power
marker Interface
VM
index
get(i)
get(1)
10
LL
get(i)
Int
LL
RA
get(i)
for(i=0 - it)
{

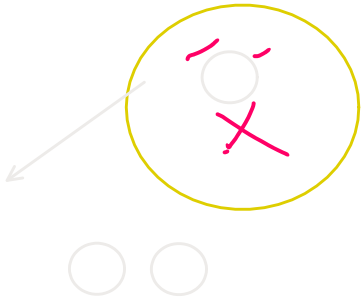

0
1
2
3
10
20
30
40
20
30
40
50



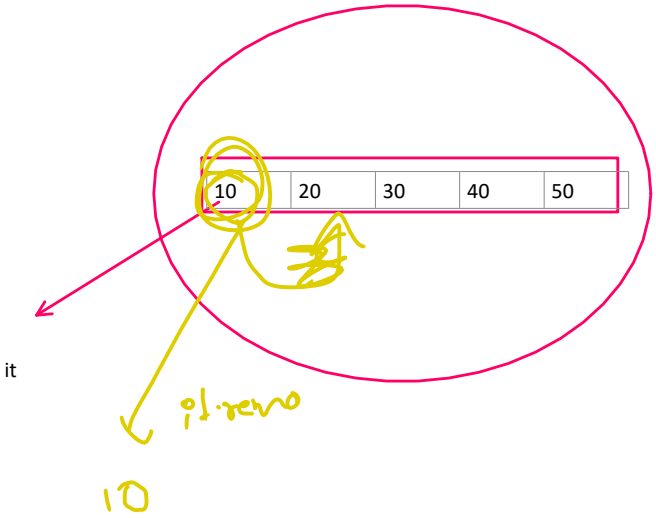


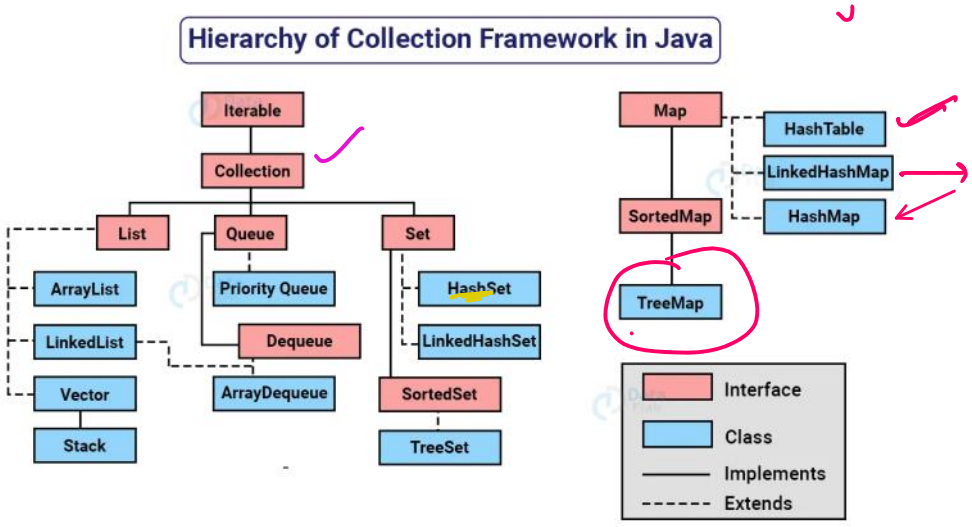


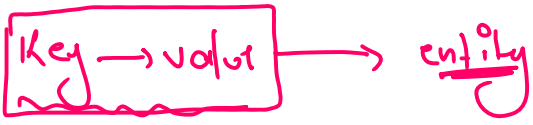
Enumeration	Iterator	ListIterator
Used for legacy classes Can't remove element	Works for all collection type Traverse and remove	Works only for List type Backword + forward Remove, update, add

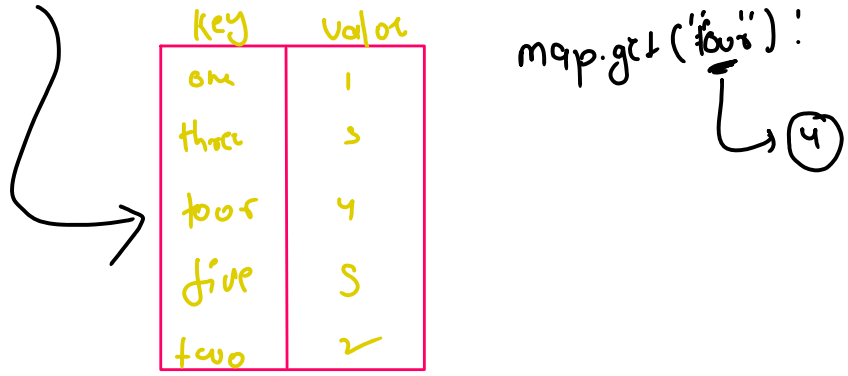


e

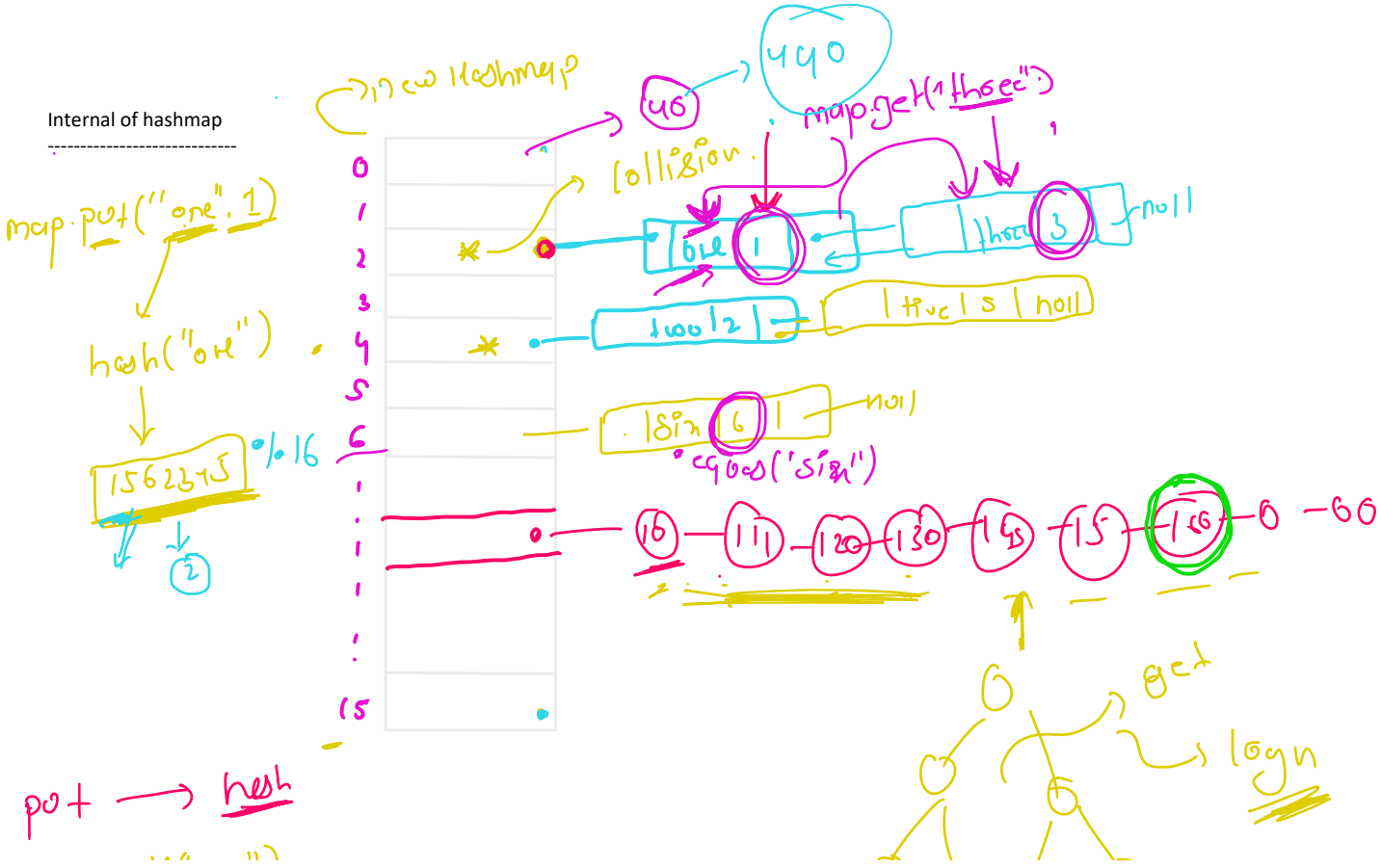

10
20
30
40
50
it
0
10

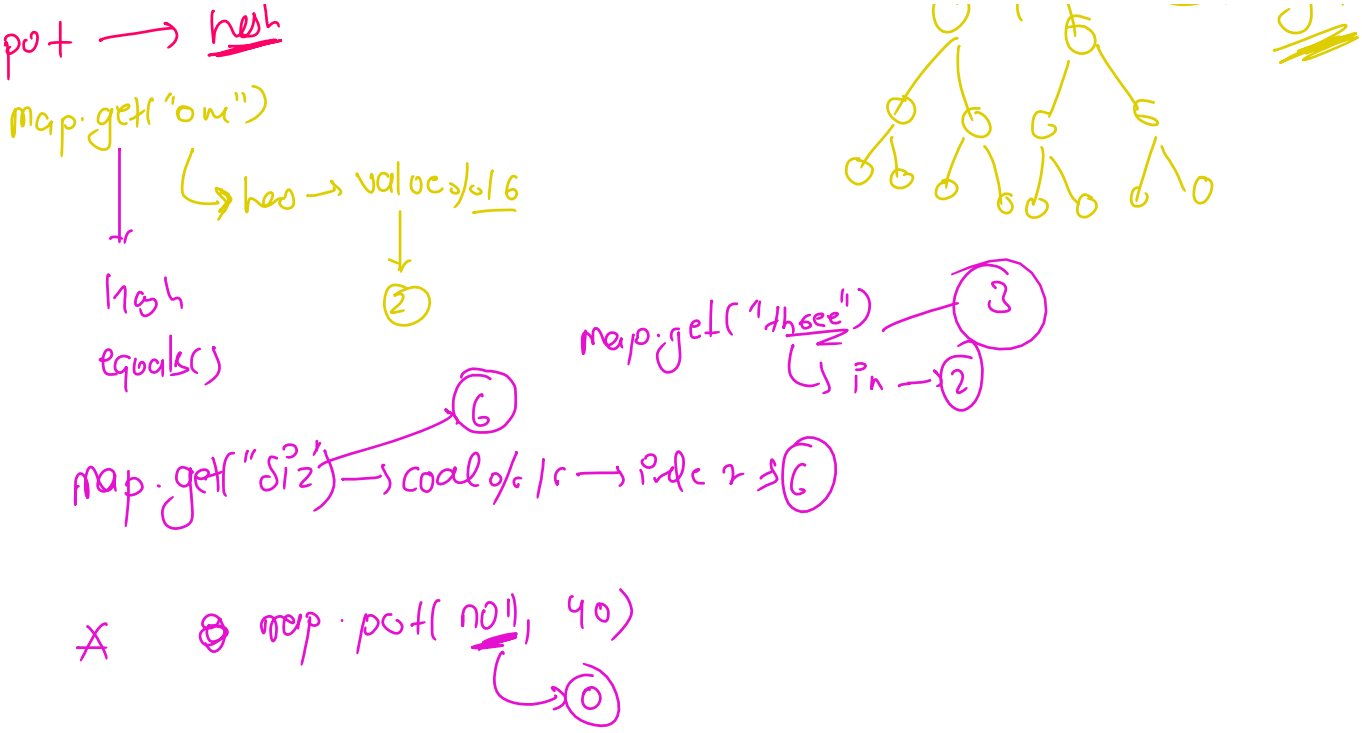

Hierarchy of Collection Framework in Java
Iterable
Collection
List
Queue
Set
ArrayList
LinkedList
Vector
Stack
Priority Queue
Dequeue
ArrayDequeue
HashSet
LinkedHashSet
SortedSet
TreeSet
Map
HashTable
LinkedHashMap
HashMap
SortedMap
TreeMap
Interface
Class
Implements
Extends


key → value → entity



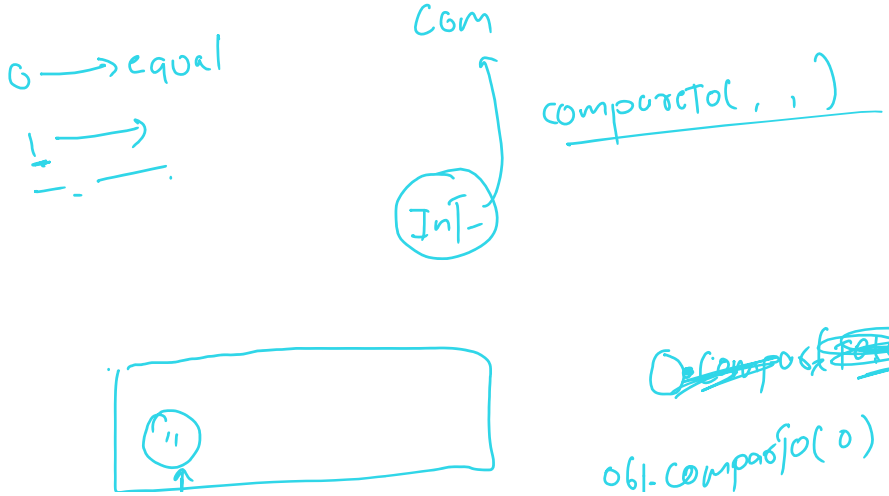
1. Get
2. Keys
3. Values
4. Entry k-v

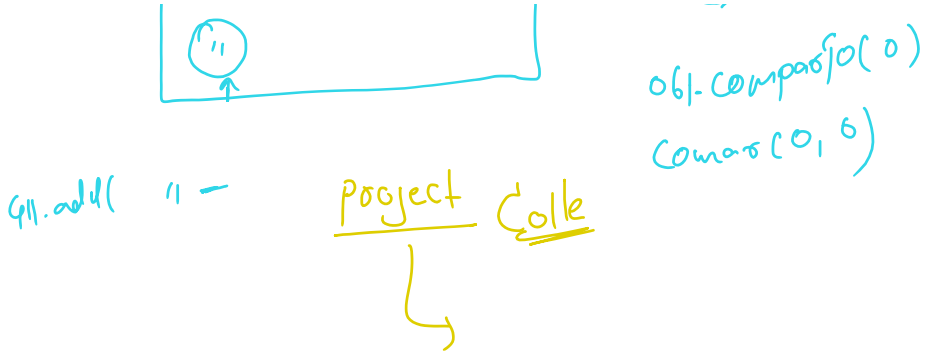




Hash → Tree → Like

Comparable : natural
Comparotor : customizer





String, Integer -> do we have anything to sort these object based on some para

Java introduce 2 functionality

1. Comparable : used for predefined classes, used for natural sorting order

All predefined wrapper classes implements Comparable and give implementation to compareTo()
It work on 3 parameter
x<y = -1
x==y = 0
x>y = 1

Can we use comparable for custom class ?
Yes we can use

5, 10 2

5

No sorting required for first element

Element 10

5	10	2

Collections.sort(al):

Internal working:

This =10
O =5

5	10

2=5
2=10
2 exact position

2	5	10

This sorting is called TimSort which perform by jvm

1. Comparator : used for customizer class, used for customizer sorting order

Can we use same sorting using comparable for custom class ?
Yes

Note:
Java is saying when you want to use custom sorting use comparator that's why developers are saying this for custom sorting

Then why two functionality ?
