i) Generics

ii) Collections (intro)

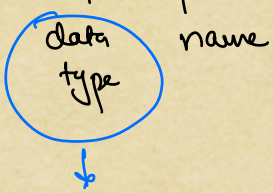
iii) Hashmap ← how it works

* Coding for Semaphore

i) Generics

⇒ Generic data

Variable ⇒ int x

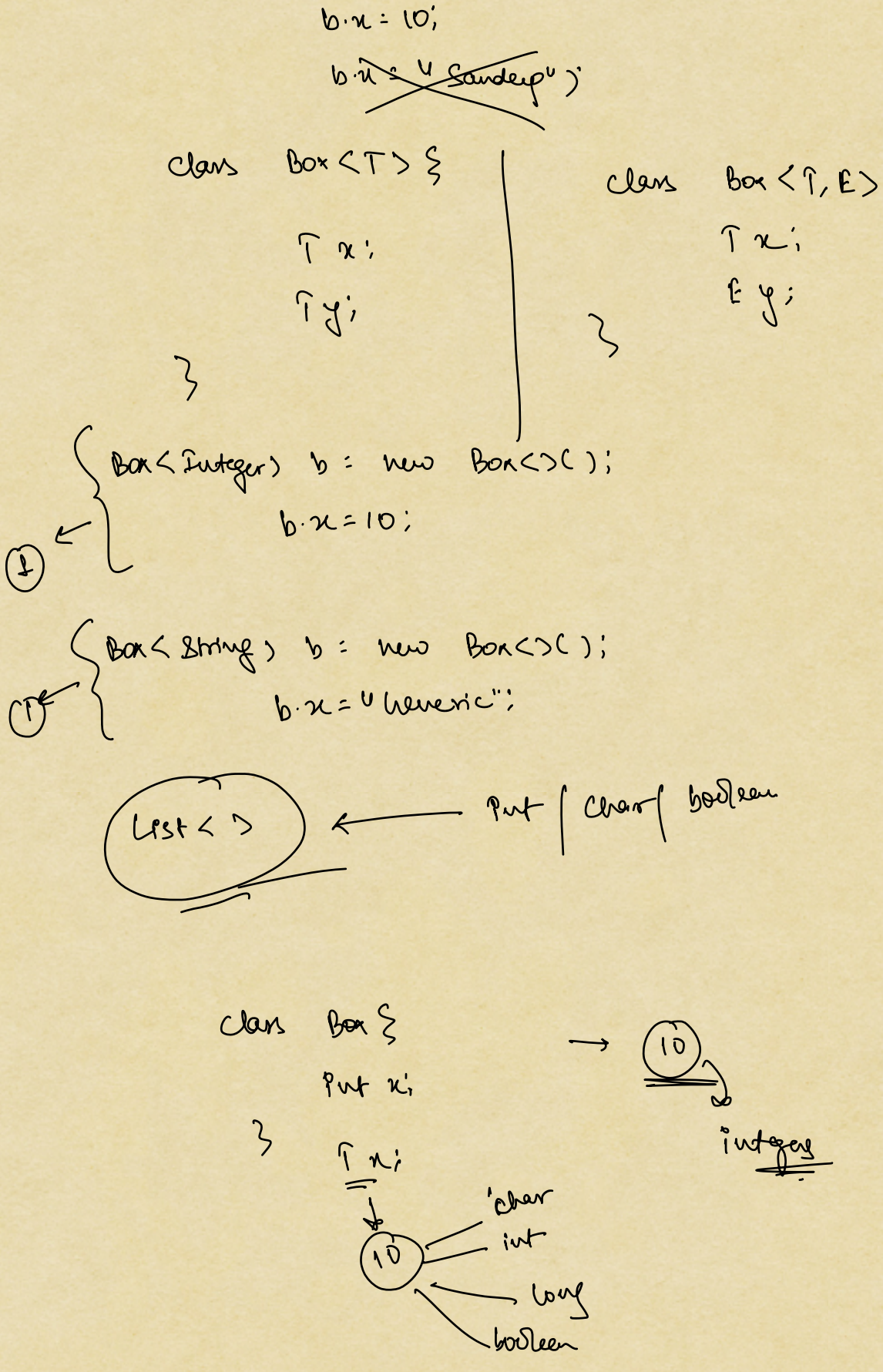


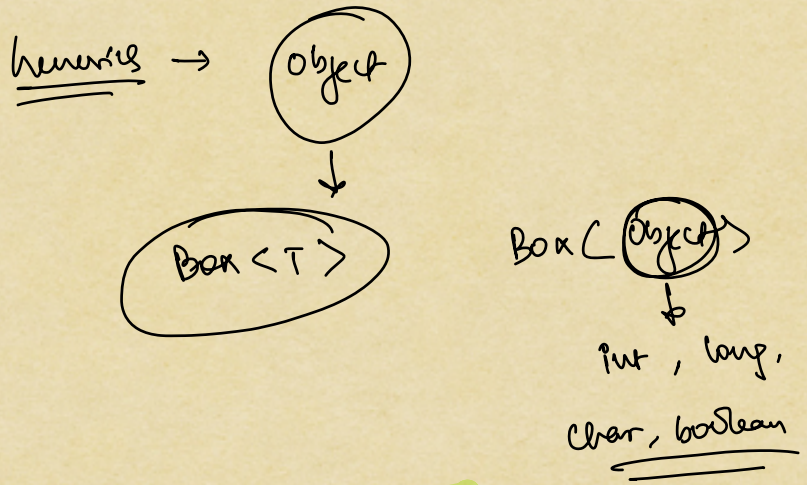
with the help of generics the constraint of data type goes away

```
class Box {
    int x | String x | char x | boolean x

}

Box b = new Box();
```


b.x = 10;
b.x = "Sandeep";
class Box<T> {
T x;
T y;
}
class Box<T, E>
T x;
E y;
}
Box<Integer> b = new Box<>();
b.x = 10;
1
Box<String> b = new Box<>();
b.x = "Generic";
List< >
Put
Char
boolean
class Box {
Put x;
}
T x;
10
integer
'char
int
long
boolean



Generic can only be an Object.

List<int>

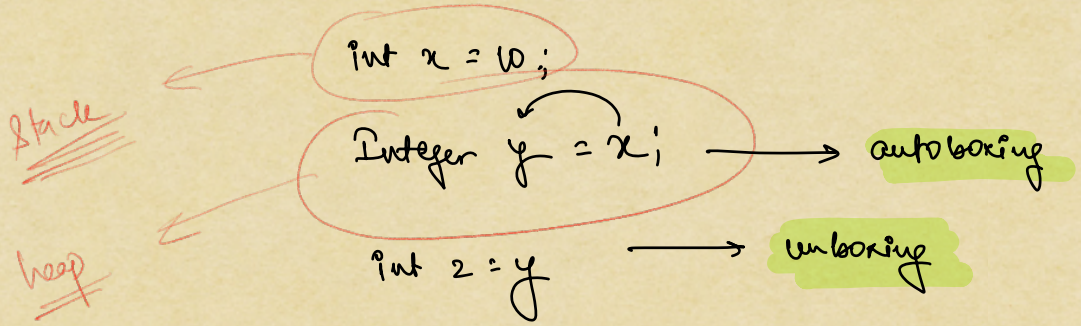
* Wrapper Class

⇒ Class that holds primitive values

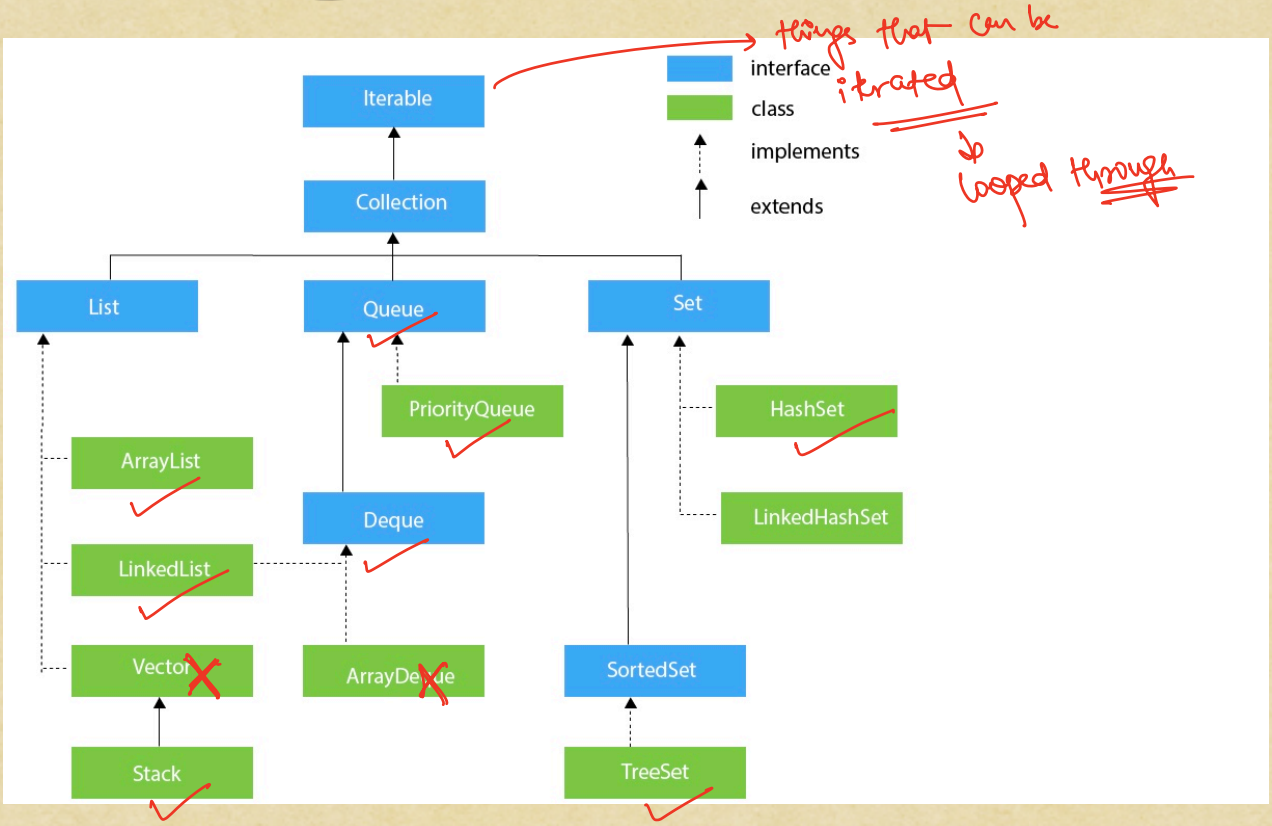
primitive → int long boolean char double float

wrapper → Integer Long Boolean Character Double Float

AutoBoxing / UnBoxing



⇒ Collections



Collection ⇒ group of items | items kept together

↓

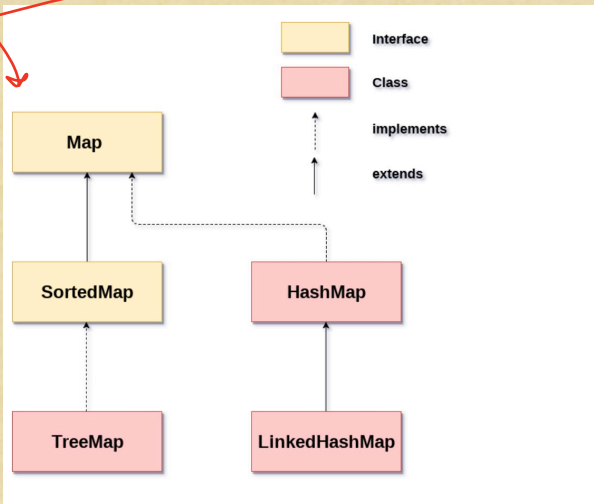
[ready to use data structures]

ArrayList | HashSet | HashMap

LL | Stack | Queue

PriorityQueue

Map is not part of collection in Java



⇒ How does Hashmap work?

- Array
- LL [objects]

wrapper

Hashmap

Array ⇒ entry type object

⇒ Entry<K,V>

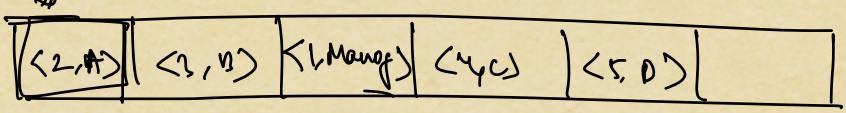
K key

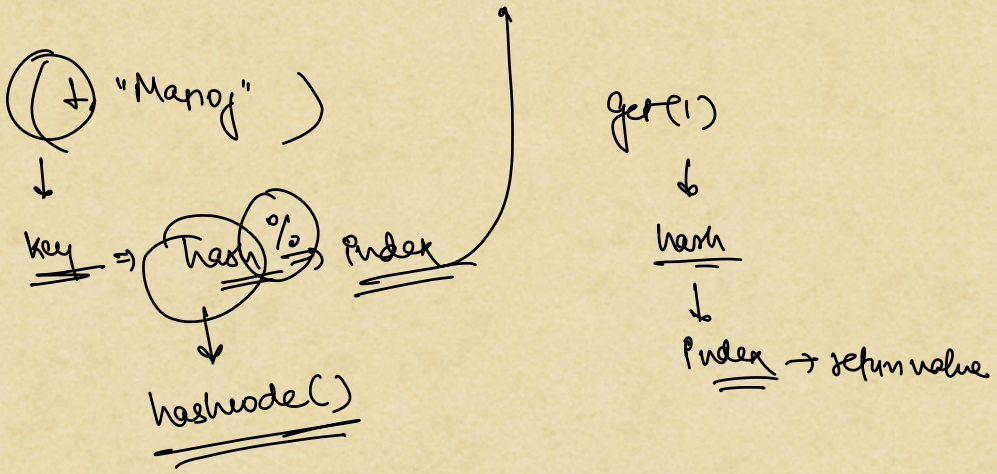
V value

HashMap<Integer, String> map :——

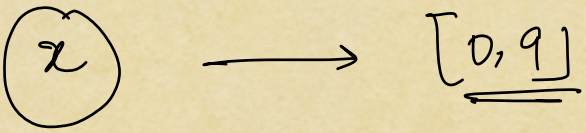
roll name

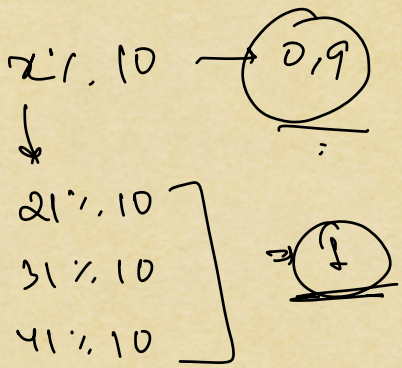
Entry<Integer, String>



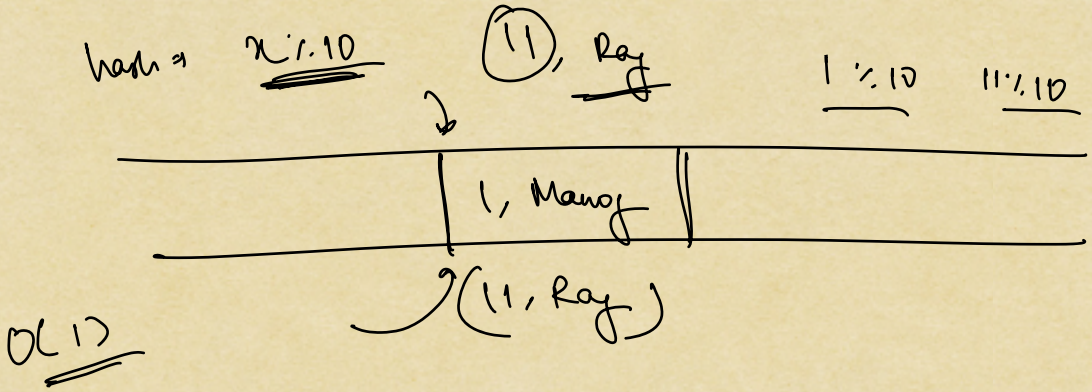


* every class in Java, has parent class → Object
  - hashcode
  - equals



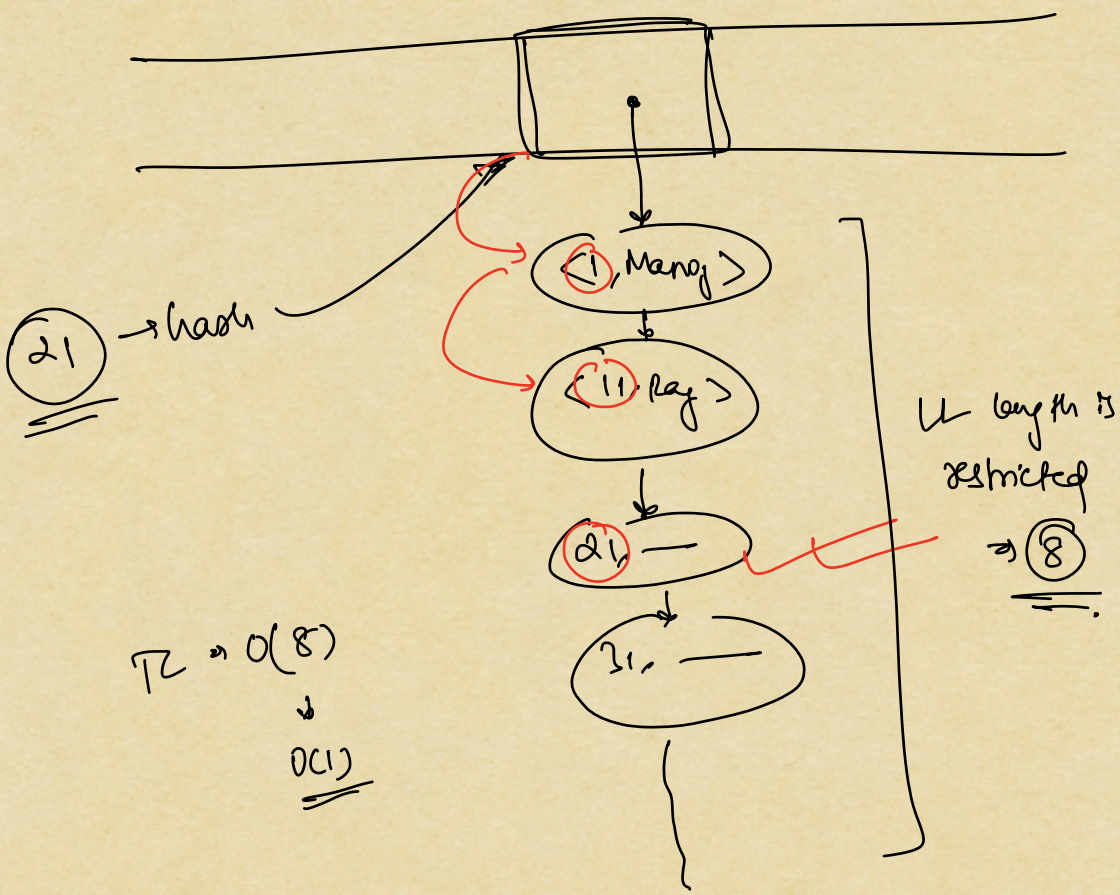


* collision ⇒ hash function returns same value for two diff inputs, its called collision.



O(1)

* collision ⇒ LinkedList at idx.



TC ⇒ O(8)

↓

O(1)

After the LL length becomes >8 → converts itself

↓

BBST

Balanced BST

TC → O(log N)

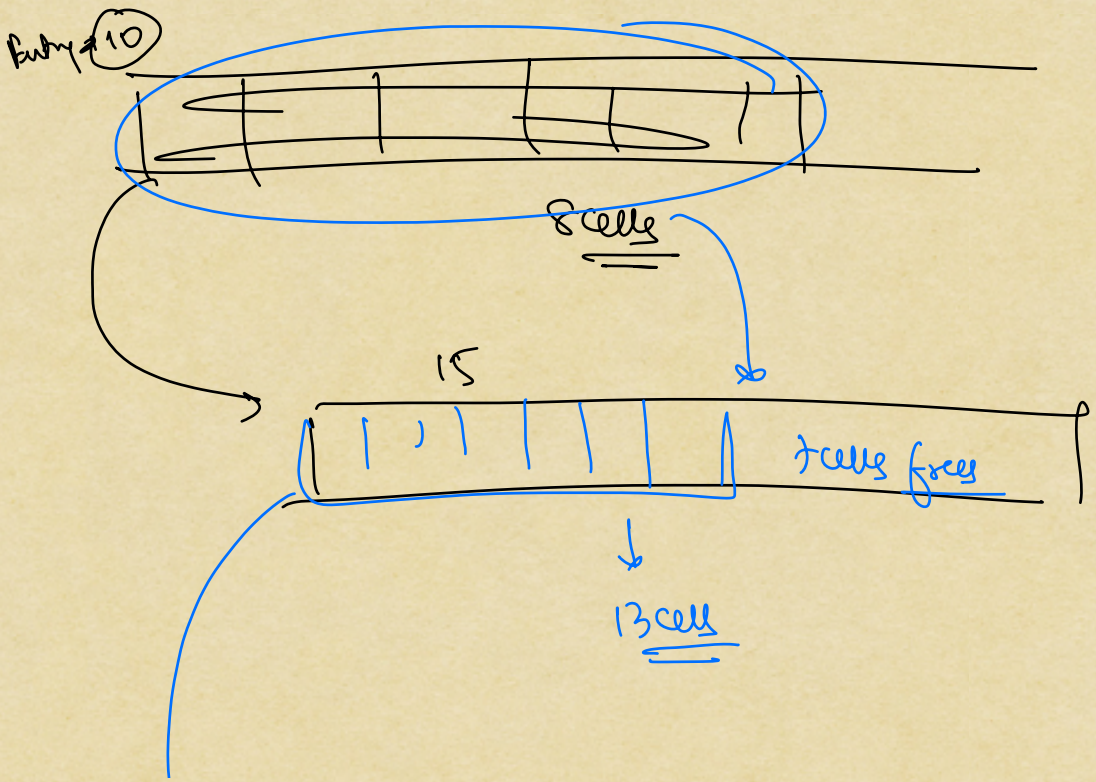
⇒ initial capacity ⇒ (10)

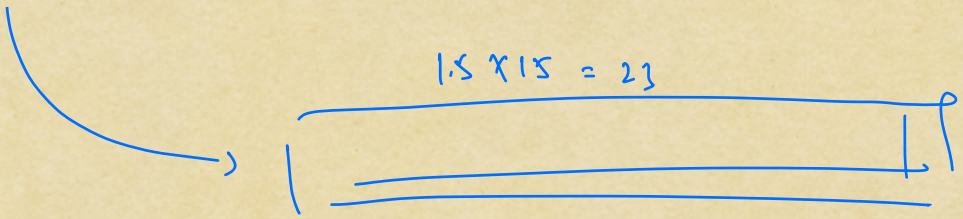
⇒ load factor ⇒ 0.75 ~ 75%.

10 ⇒ 8 cells

↓

1.5x



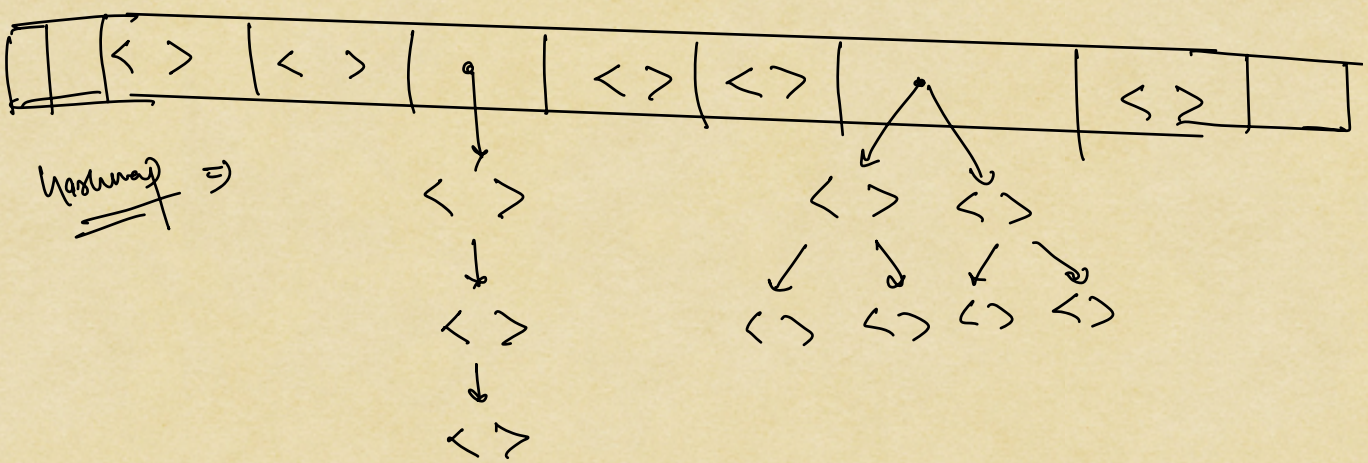


⇒ if no of calls (↑) collision (↓)

⇒ collision ⇒ O(1) ⇒ with LL ⇒ O(1)

⇒ BBST ⇒ O(log N)

⇒ Amortised ⇒ O(1)



i) put( ) ⇒ duplicate keys allowed ⇒ No.

↓

overrides the value

ii) get(key) ⇒ equals

i) Scenario ⇒ every insertion to collide

Class A

HashMap<A, B>

hashcode( )

return 1

ii) Set of attributes

table → time

↓ later timestamp

A

B

C

O(1)

O(1)

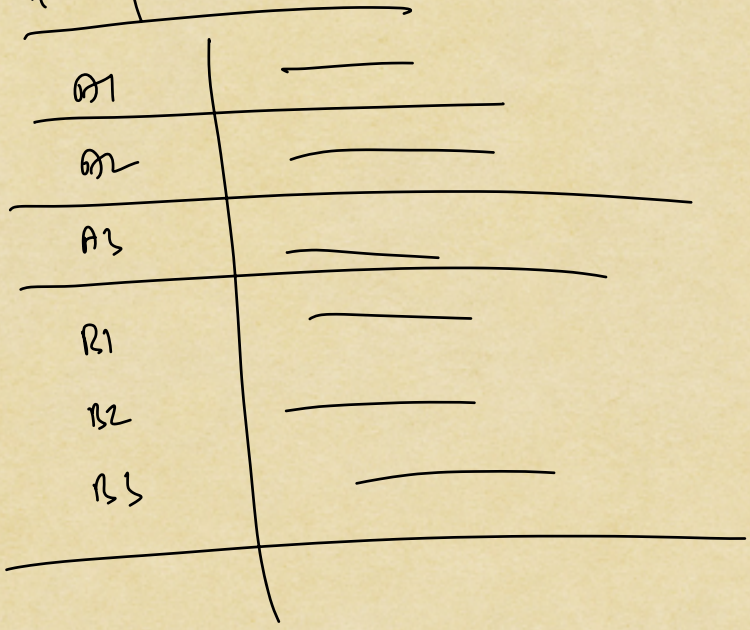
O(1)

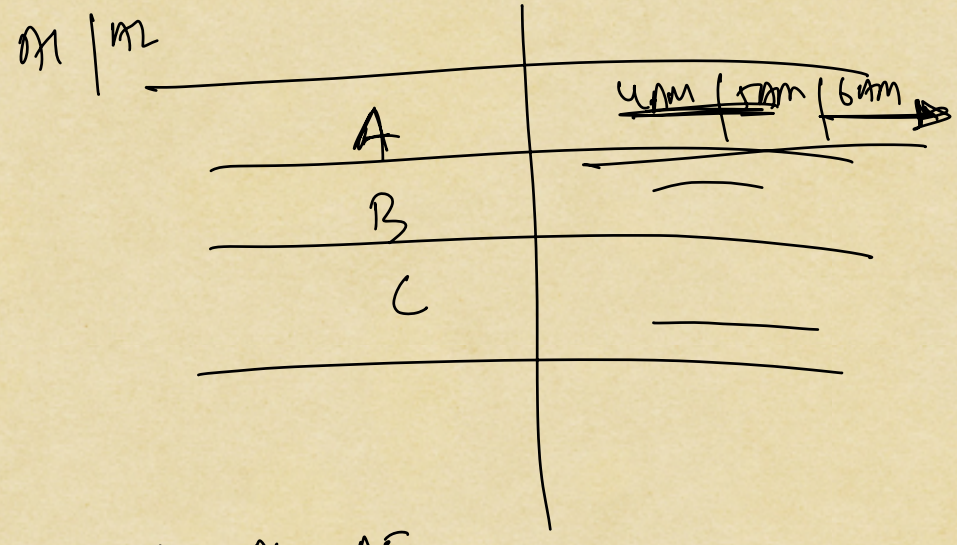
A1 A2 A3 A4

B1 B2 B3 B4

C1 C2 C3 C4 C5

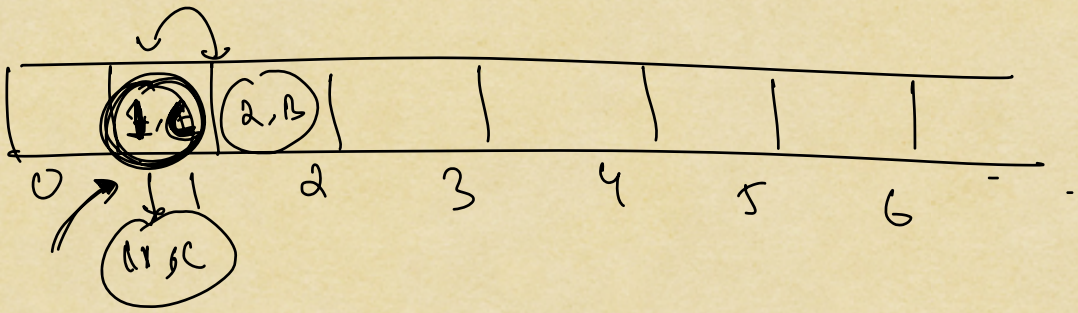
Hashmap





A1 A2 A3 A4 A5

equals (true)



hash() {

key % 10

}

1 % 10 ⇒ 1

2 % 10 ⇒ 2

11 % 10 ⇒ 1

map.put(1, A)

(2, B)

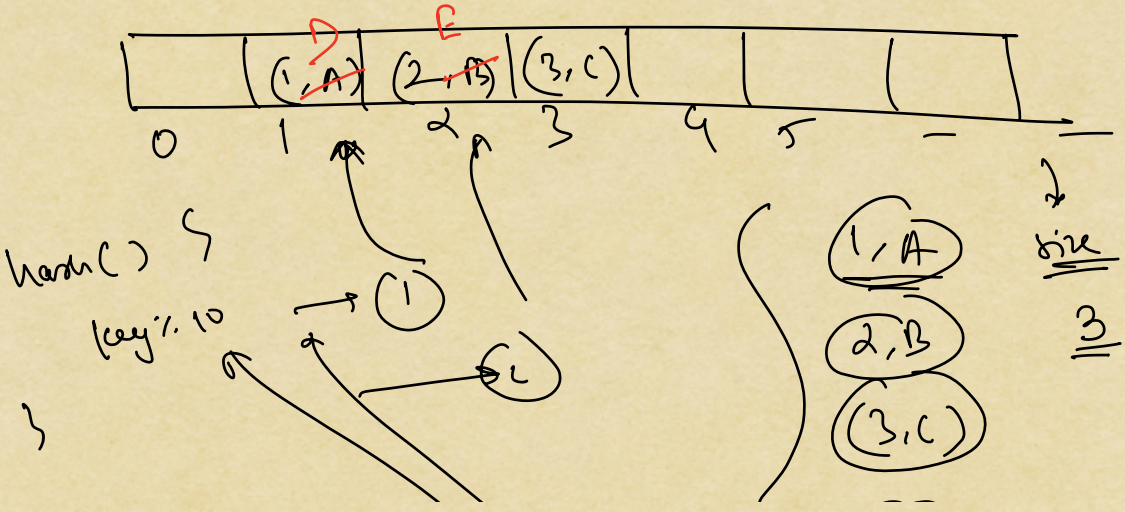
(11, C)

equals() {

~~true~~

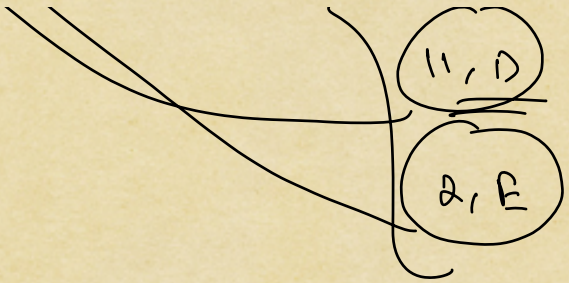
~~values = values~~

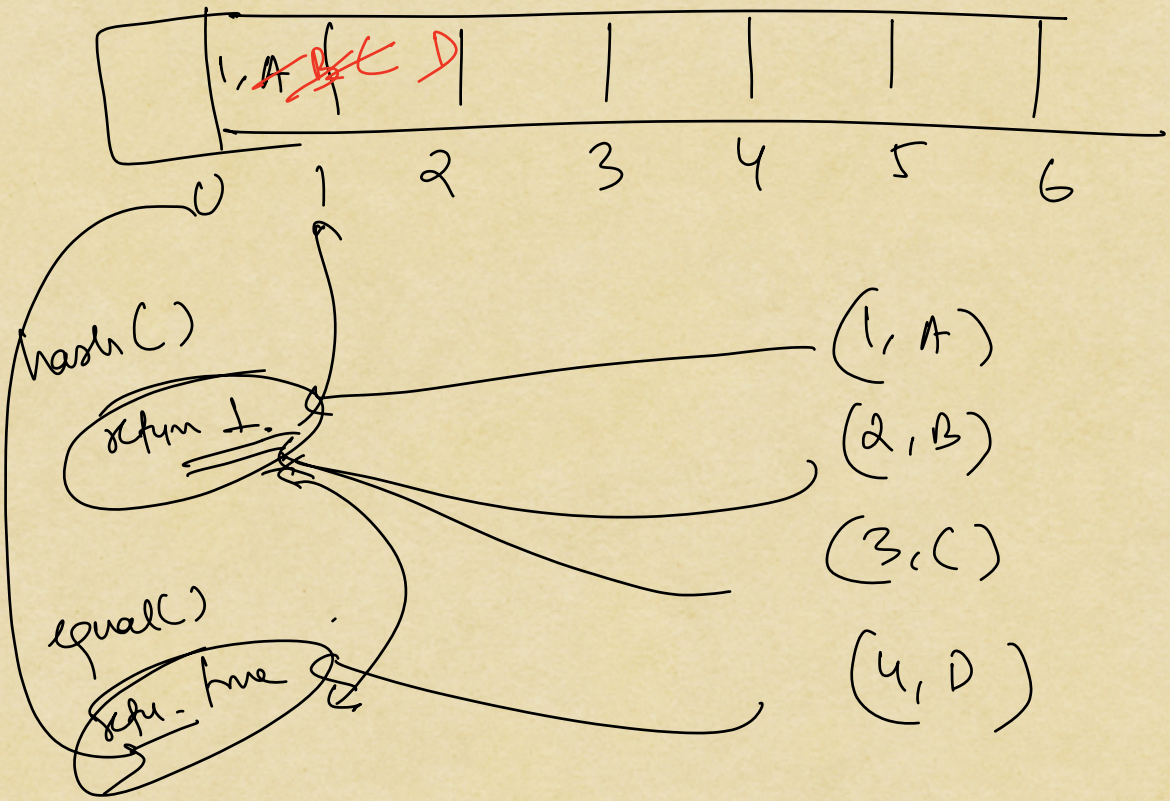
}



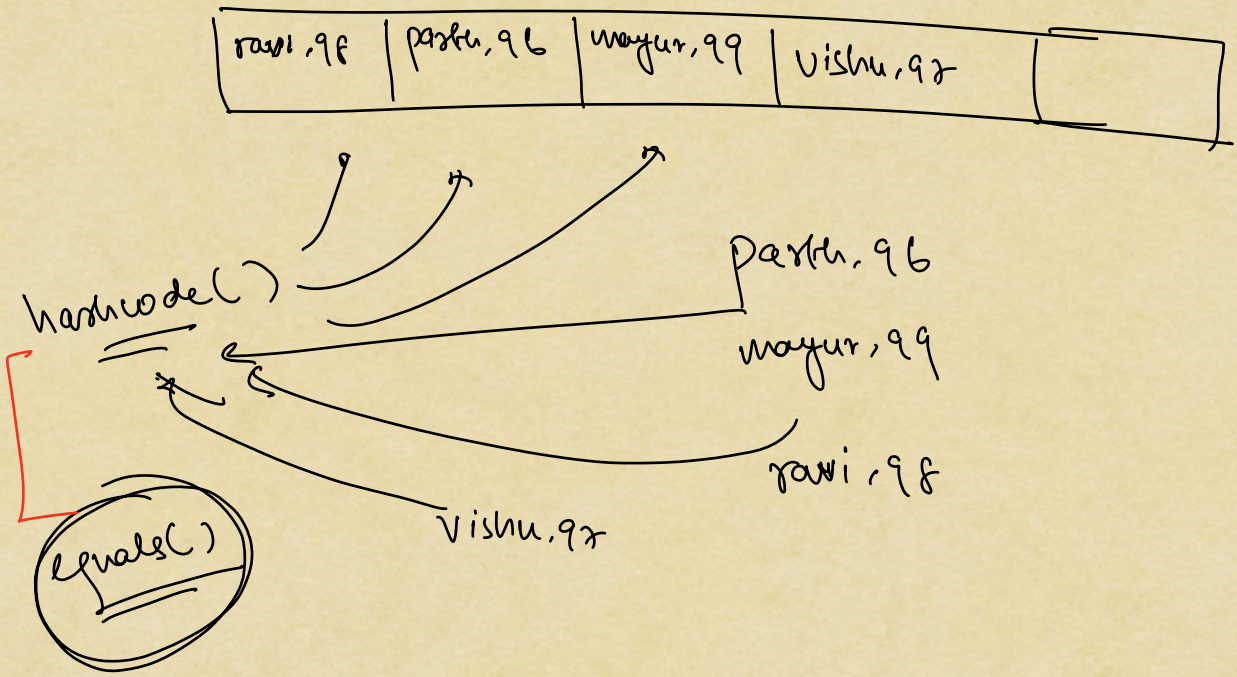
equals()

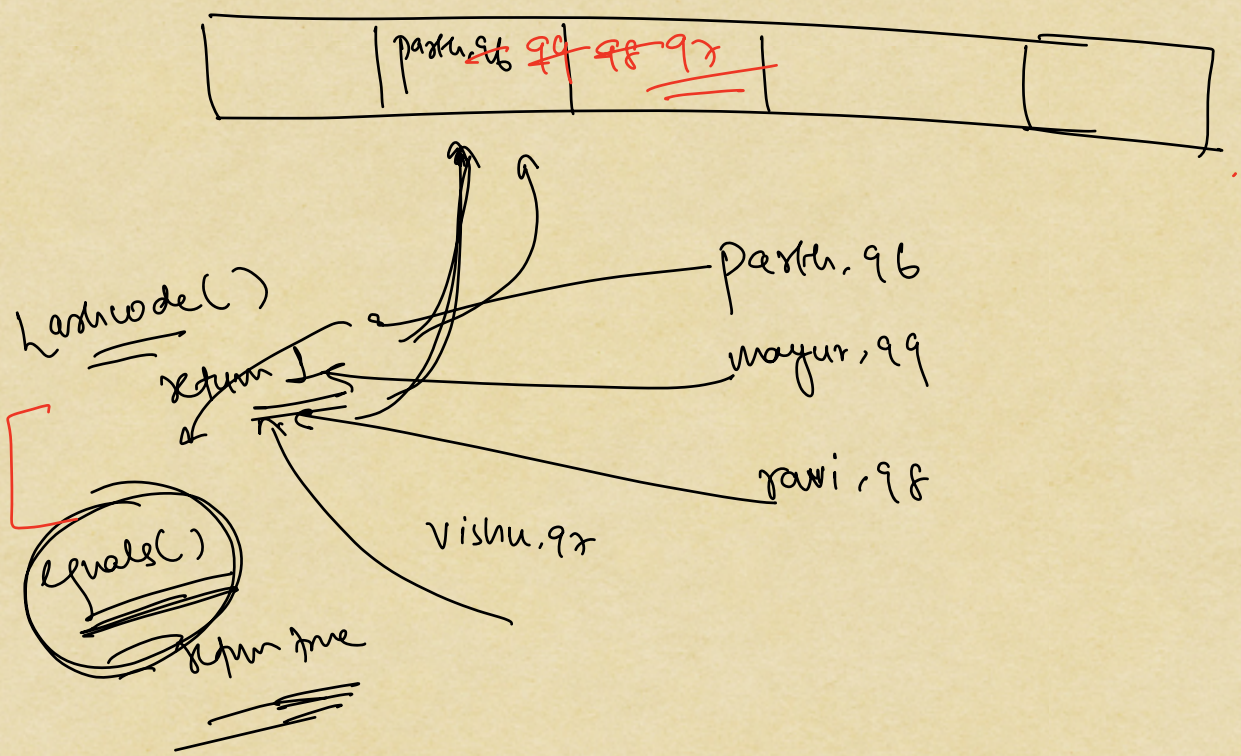
true

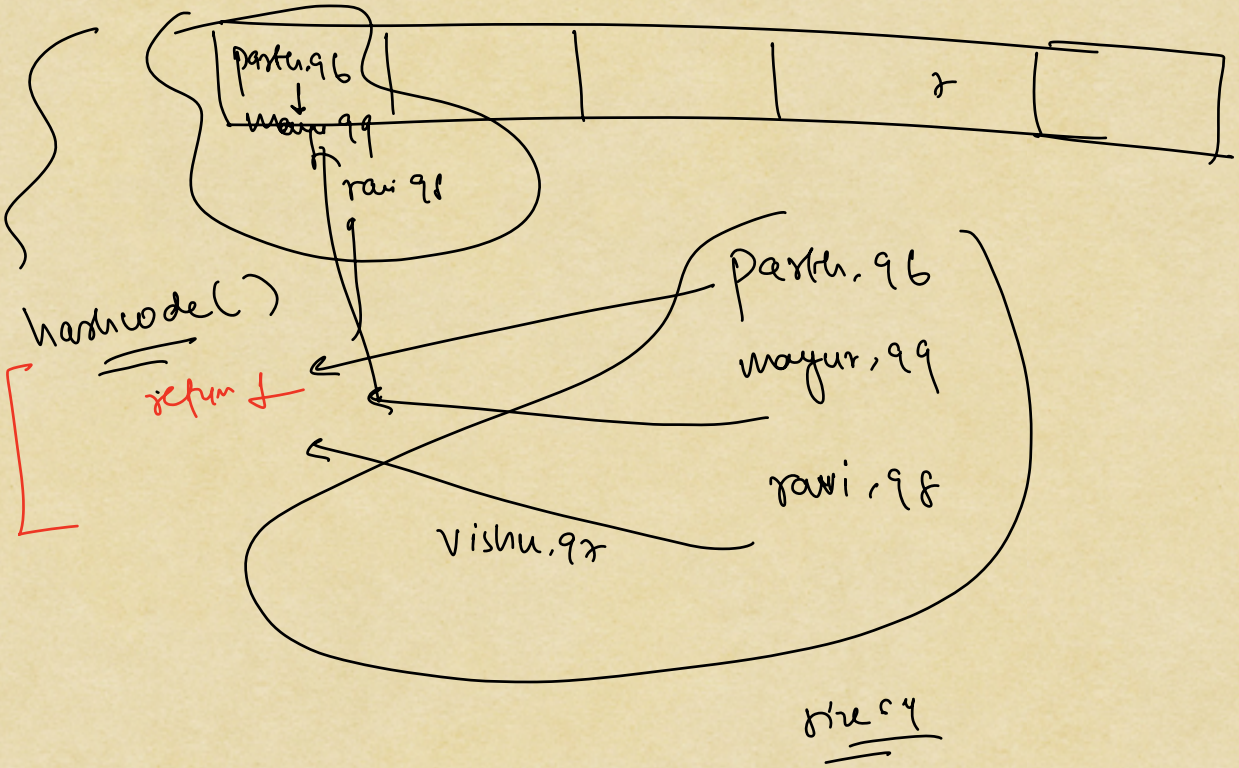




N numbers ⇒ $\frac{1}{=}$


ravi, 98
parth, 96
mayur, 99
vishu, 97
hashcode()
parth, 96
mayur, 99
ravi, 98
vishu, 97
equals()


parth, 96
hashcode()
return 1
parth, 96
mayur, 99
ravi, 98
vishu, 97
equals()
return true


parth,96
mayur,99
ravi,98
x
hashcode()
return 1
parth,96
mayur,99
ravi,98
vishu,97
size=4