THRIVER ASHISH

JAVA COLLECTIONS INTERNALS

For a Detailed Explanation -- Refer my Youtube video tutorials

ARRAYLIST
HASHMAP
HASHSET
LINKED HASHMAP
TREEMAP

https://www.youtube.com/c/thriverashish

https://www.linkedin.com/in/thriverashish

@thriverashish

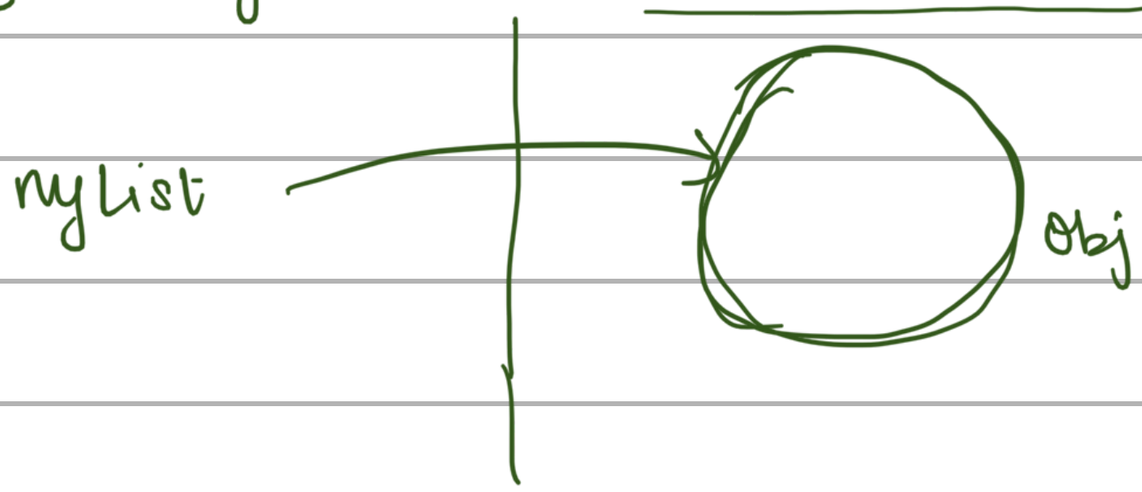
ARRAYLIST INTERNALS

@thriverashish

https://www.linkedin.com/in/thriverashish

Video Link -- https://youtu.be/JzD0VCX3ql4

```
List<String> myList = new ArrayList<String>();
```




```
transient Object[] elementData;

private static final Object[] EMPTY_ELEMENTDATA = {};
```

Non-Parameterized Construction

```
public ArrayList() {
    super();
    this.elementData = EMPTY_ElementData;
}

private static final int DEFAULT_CAPACITY = 10;
```

```
public ArrayList(int initialCapacity) {
    super();
    if(initialCapacity<0) throw IllegalArgumentException;
    this.elementData = new Object[initialCapacity];
}
```

How ArrayList Grows Dynamically.

Integer | 5 | 8 | 12 | 2 | 18 | 15 | 17 | 11 | 20 | 25 | → capacity = 10

20 → ensureCapacity

The capacity is full now

↓ grow.

newCapacity = 50% more of OlderCapacity

10 + 5 = 15

* `int newCapacity = OldCapacity + (OldCapacity >> 1);`

Dep Dive into Righ Shift op.

OldCap = 10 ⇒ 1 0 1 0

2^3 2^2 2^1 2^0

↓ >>1

..... 0 1 0 1

2^3 2^2 2^1 2^0

4 1 ⇒ 5

= (5)

OldCap + (OldCap >> 1)

10 + 5

newCapacity = (15)

Eligible for GC.

OldCap Array → | 5 | 8 | 12 | 2 | 18 | 15 | 17 | 11 | 20 | 25 |

newCap Array → | | | | | | | | | | 25 | | | | | |

0 1 2 3 4 5 6 7 8 9 10

new Element.

Arrays.copyOf

↖ used internally.

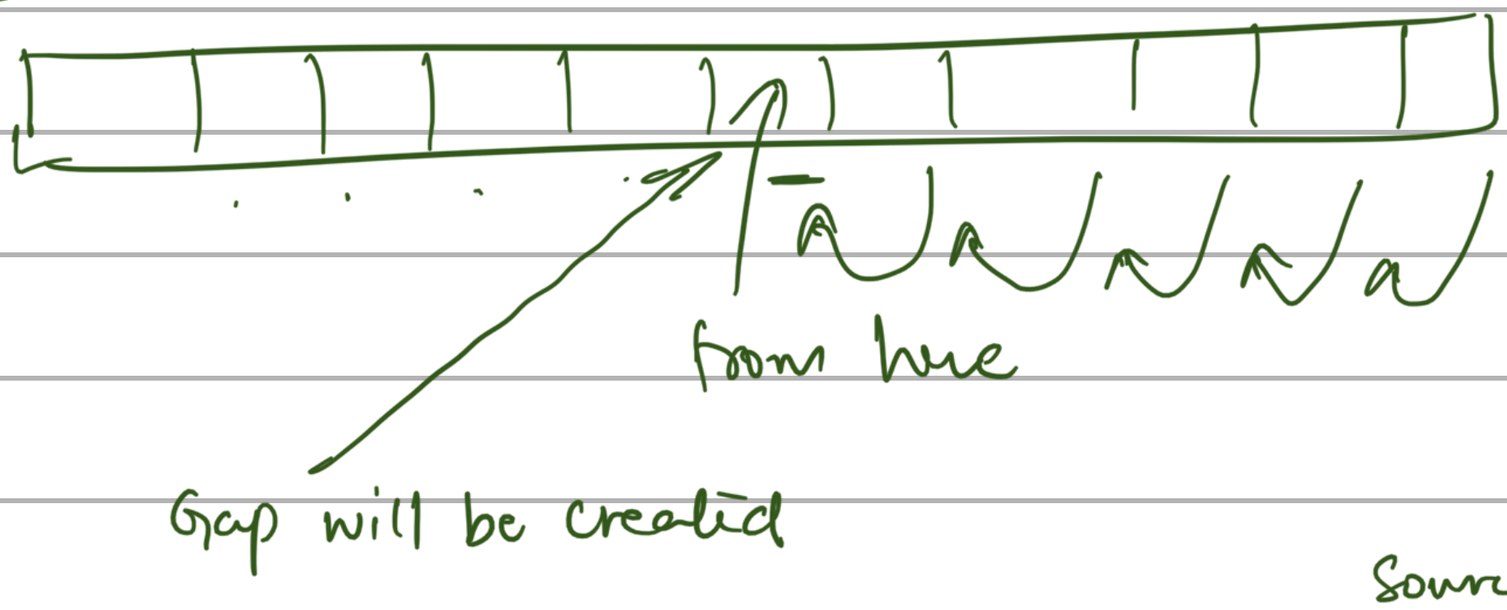
At index 9 → O(1)

But now for index 10 ⇝ it has to re-grow

copy from old to new Array.

Amortized Complexity

Remove



Source position

System.arrayCopy (elementData, index+1, elementData,

index, numMoved);

Destination

gap is created

Arraylist

non-synchronised

Fast

Vector

Synchronised

Slow.

Even if the Environment is Single Threaded.

Vector → Always Slower than ArrayList.

ArrayList → Faster.

* CopyOnWriteArrayList ← Read this

@thriveashish
- ↳ insta
- ↳ twitter
- ↳ linkedin

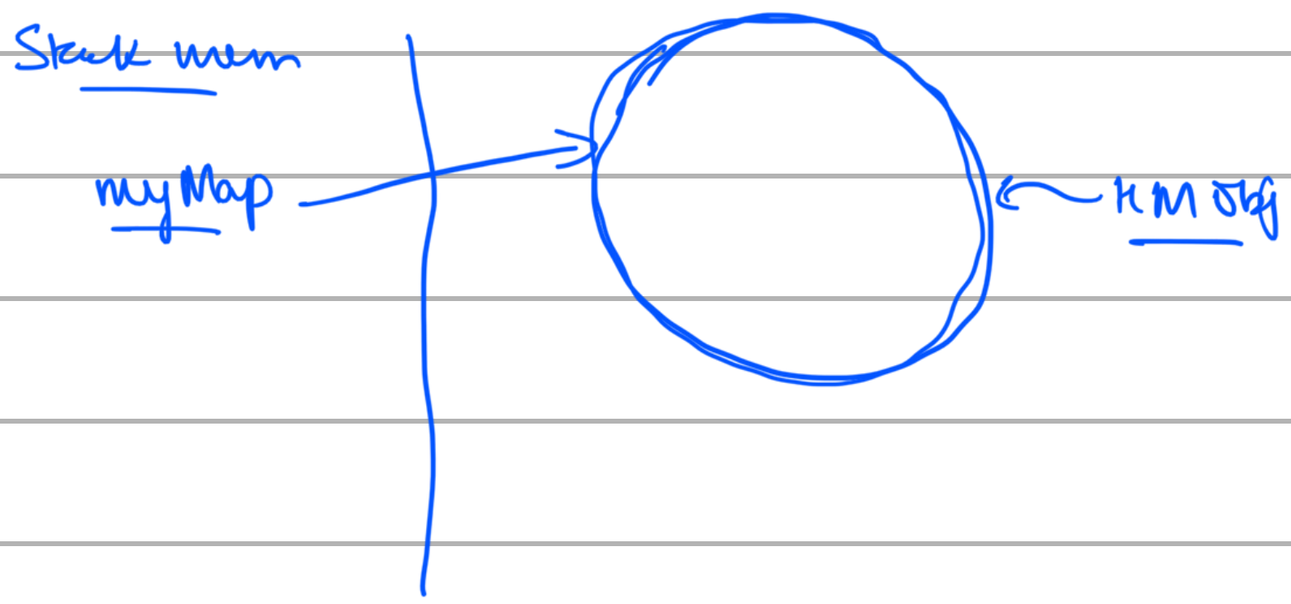
HASHMAP INTERNALS

@thriverashish

https://www.linkedin.com/in/thriverashish

Video Link -- https://youtu.be/TkM6-NLvAR8

```
Map<String, String> myMap = new HashMap<>();
```

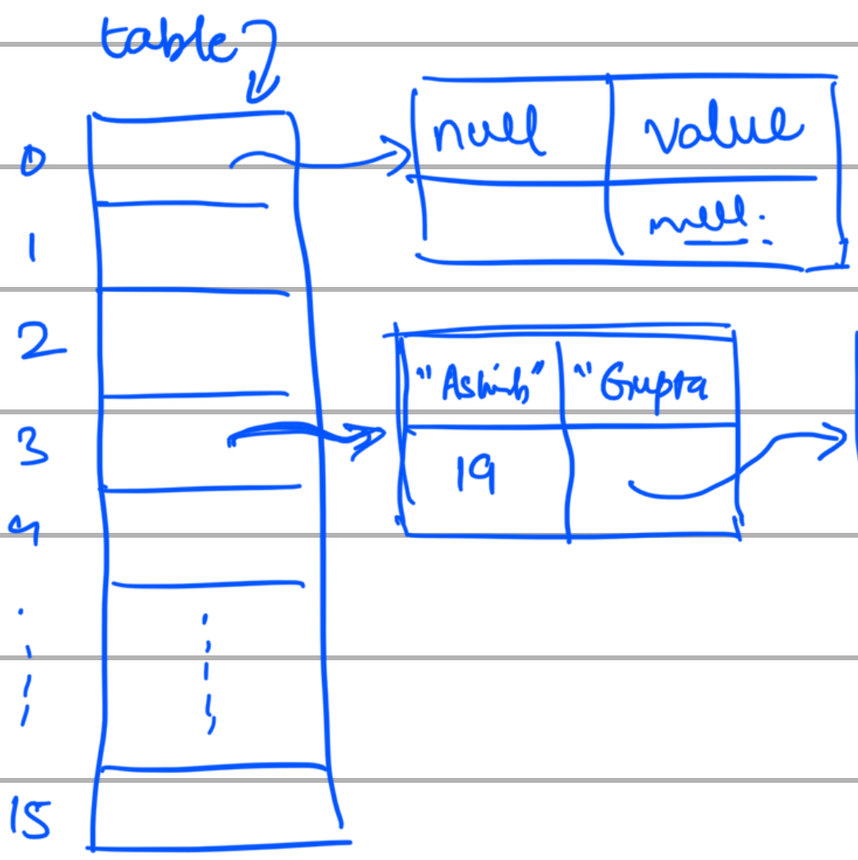


get(__) , put(key, value)

O(1)

table ← array of type Node ← inner class which extends Map.Entry

transient Node[] table



Size = 16
capacity

Case 1: Key is not null.

mymap.put("Ashish", "Gupta");

Key

Hash of Ashish is 19

19% capacity

Kept growing. 19 % 16 = 3

mymap.put("XYZ", "Pqr")

hash of XYZ = 35

35 % 16 = 3

index.

Node ____ {

Key
Value
hash

THRIVE WITH ASHISH

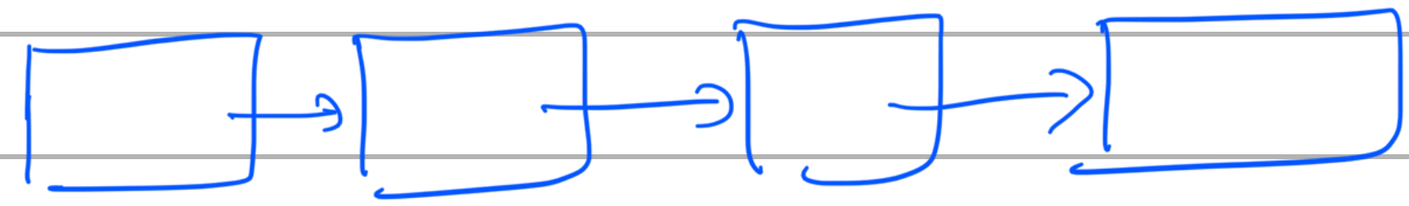
Node (next)

}

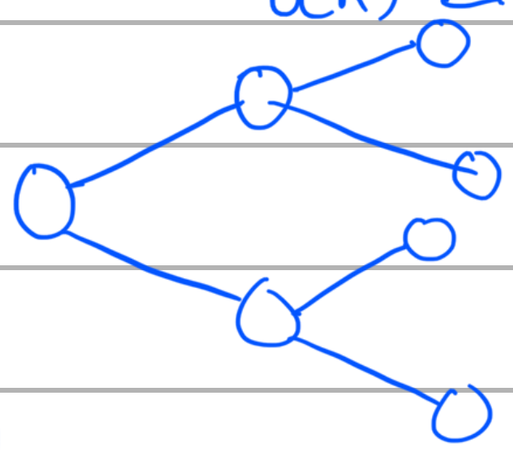
Case 2 : Key is null

If Size of LL is grown upto Threshold.

from Java 8 → collision is further handled via Self Balancing Binary Search Tree



O(K) ≃ O(n)



Self Bal BST

O(log n)

Tree.Node

loadfactor → 75% By default

If more and more elements are inserted
⇒ increases collision.

Capacity of HM → 16 loadfactor 75%

75%

12 → Size of HM. in

Capacity of HM will increase.

newCapacity = Oldcapacity << 1

16 << 1

32

How 16 up in Binary = ...10000 = 16

2^4 2^3 2^2 2^1 2^0

left Shift by 1

newCap. ---100000 = 32
v v v v v v
5 4 3 2 1 0

Max Capacity = 1 << 30

Get Operation

Key is null

table[0]

Key is not null.

- Cal HashCode of key.
- maps index & search.
- iterates over table[index]

Checks hashCode & equals

if matches return value.
else null. (return).

equals given key & present key.

@thriverashish
roadmap.
insta
twitter
linkedin.

Ashish

HASHSET INTERNALS

@thriverashish

https://www.linkedin.com/in/thriverashish

video link -- https://youtu.be/tK5N1bW4YLo

04 DAYS

```
Set<String> mySet = new HashSet<String>();
```

String → Integer or etc

HashSet internally uses HashMap.

HashSet → Value

HashMap → (Key, Value)

↳ it stores unique values and no Duplicates

myset.add("Ashish")

↳ Stored as Key in HashMap.

```
private transient HashMap<E, Object> map;
```

E → generic

Object → Always bc of Object type.

```
private static final Object PRESENT = new Object();
```

PRESENT → Stored against each and every key.

↳ treated as Value in HashSet.

mySet.add("thriverashish")

Key in HM but treated as Value in HS

thriver ashish	PRESENT
hash code.	null

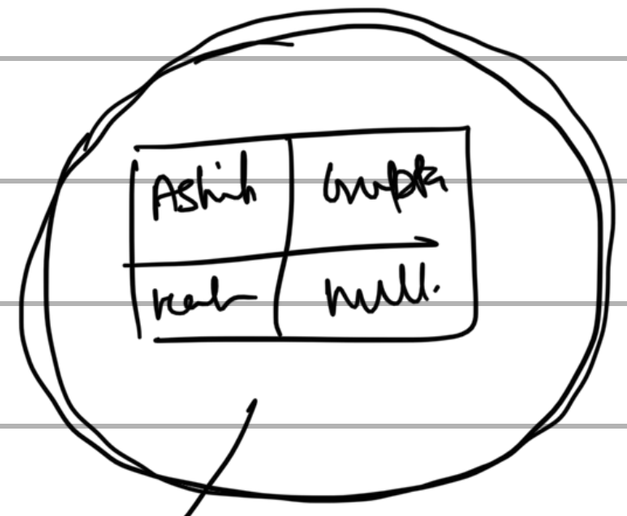
Important

```
public boolean add(E e) {
    return map.put(e, PRESENT) == null;
}
```

1st time:

```
map.put("Ashish", "Gupta");
```

null

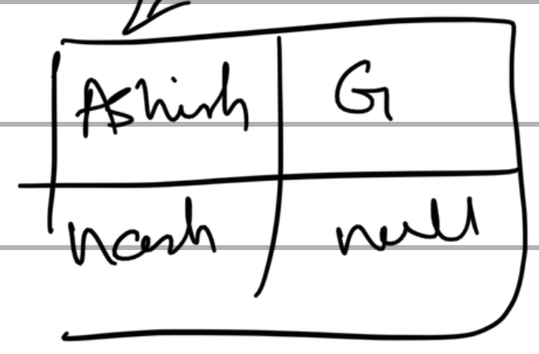


2nd time

```
map.put("Ashish", "G");
```

return

Gupta




```
public boolean contains(Object o) {
    return map.containsKey(o);
}
```

@thriverashish
↳ insta
twitter
LinkedIn

LINKED HASHMAP INTERNALS

@thriverashish

https://www.linkedin.com/in/thriverashish

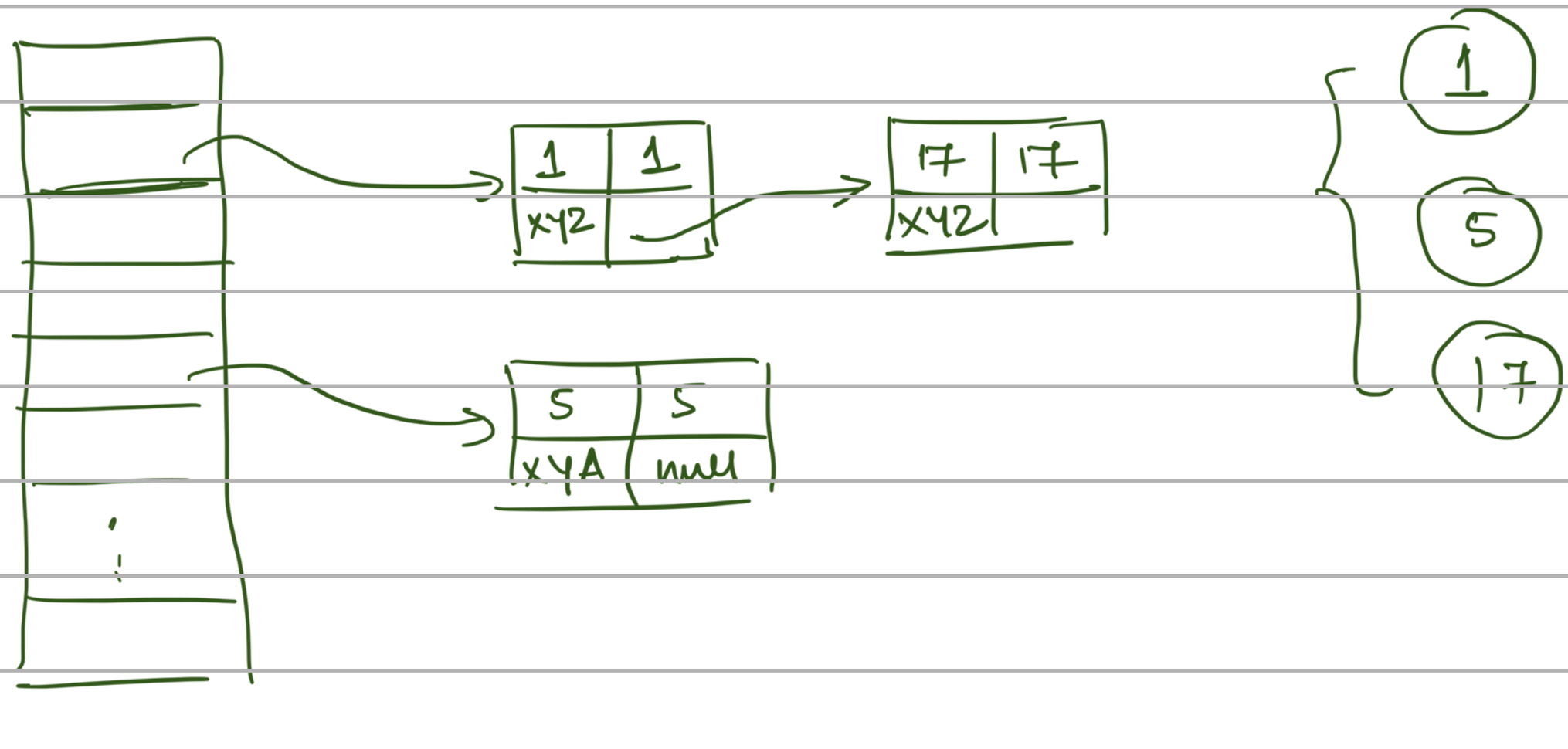
video link -- https://youtu.be/xFWkXKRtIMI

```
public class LinkedHashMap<K,V>
        extends HashMap<K,V>
        implement Map<K,V> {

    static class Entry<K,V> extends
    HashMap.Node<K,V> {

        Entry<K,V> before, after;

        Entry(int hash, K key, V value, Node<K,V> next) {
            super(hash, key, value, next);
        }
    }
}
```

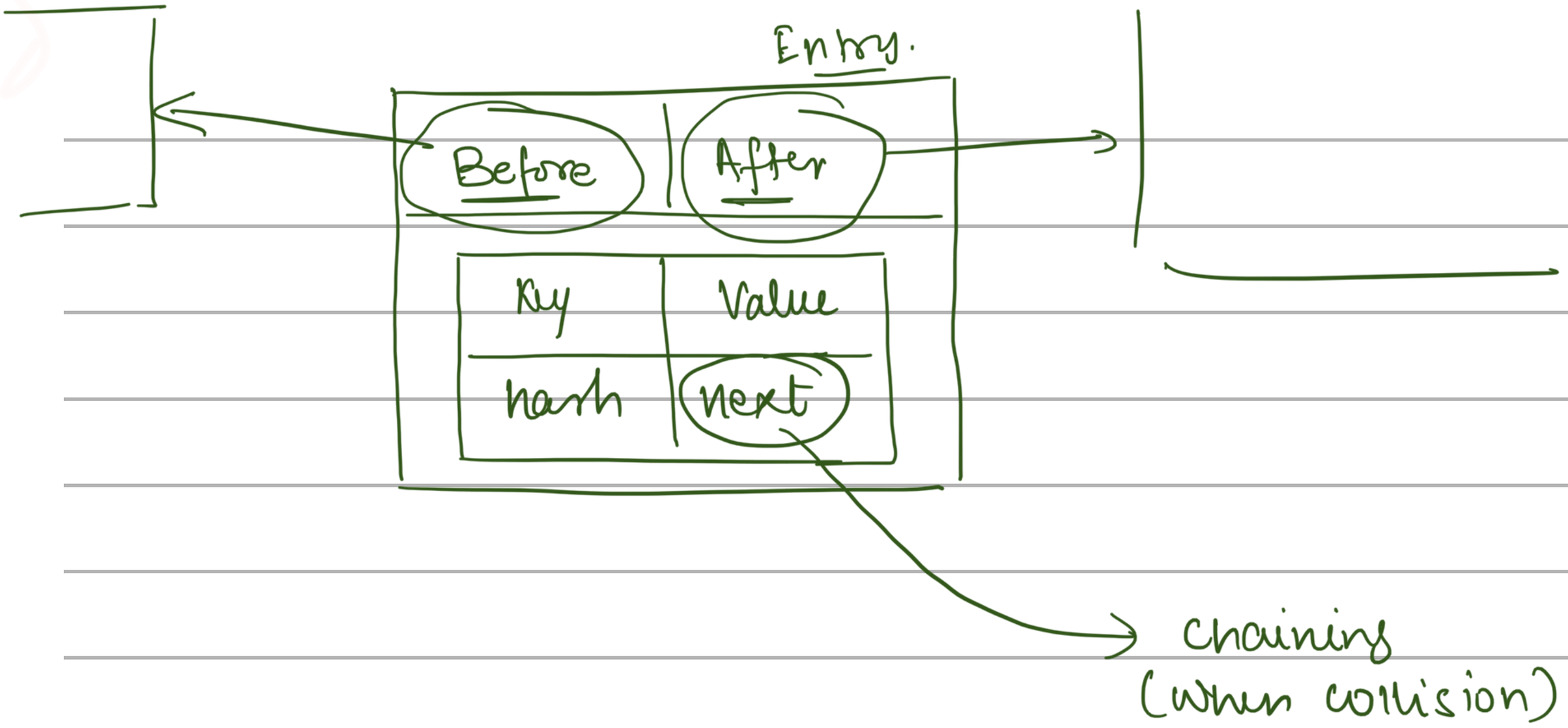
inner

inner class of HashMap

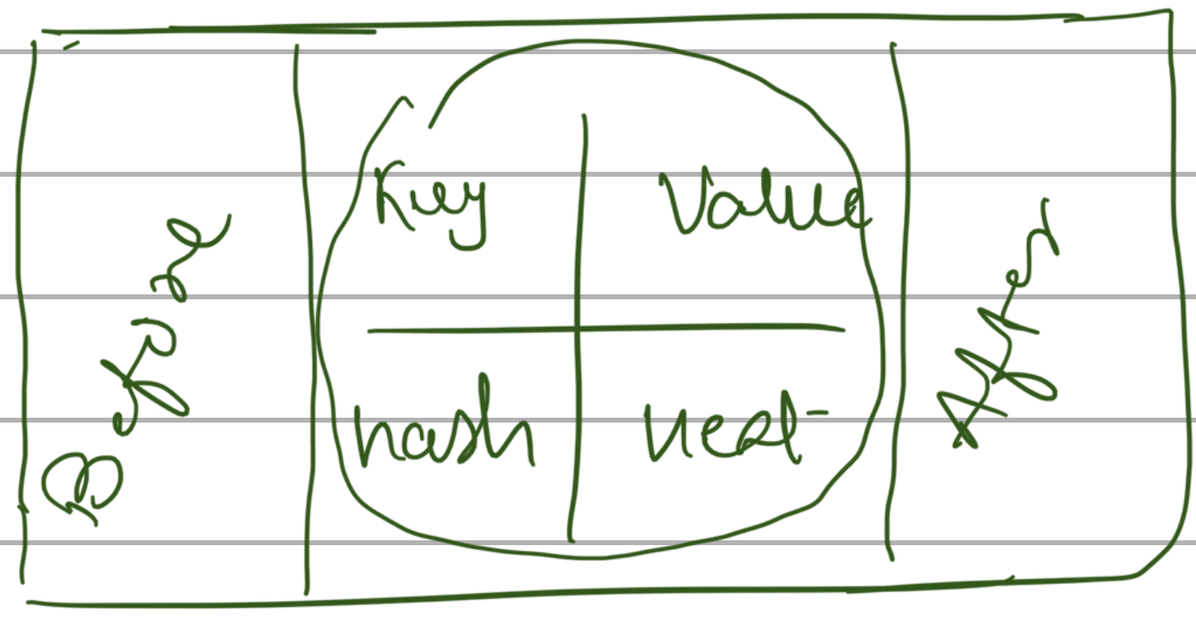
LinkedHM ⟶ Maintains Insertion Order

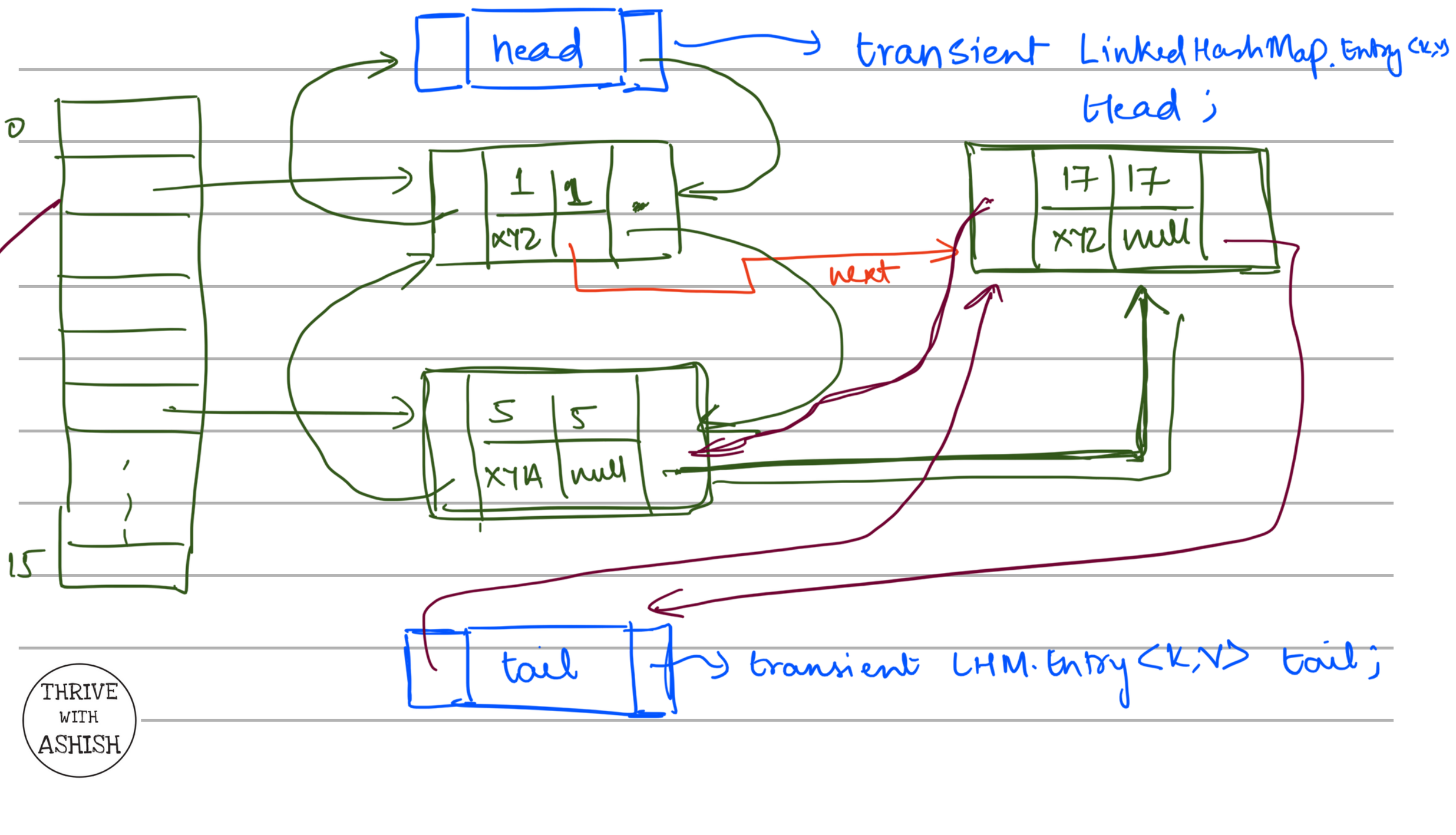


Entry.

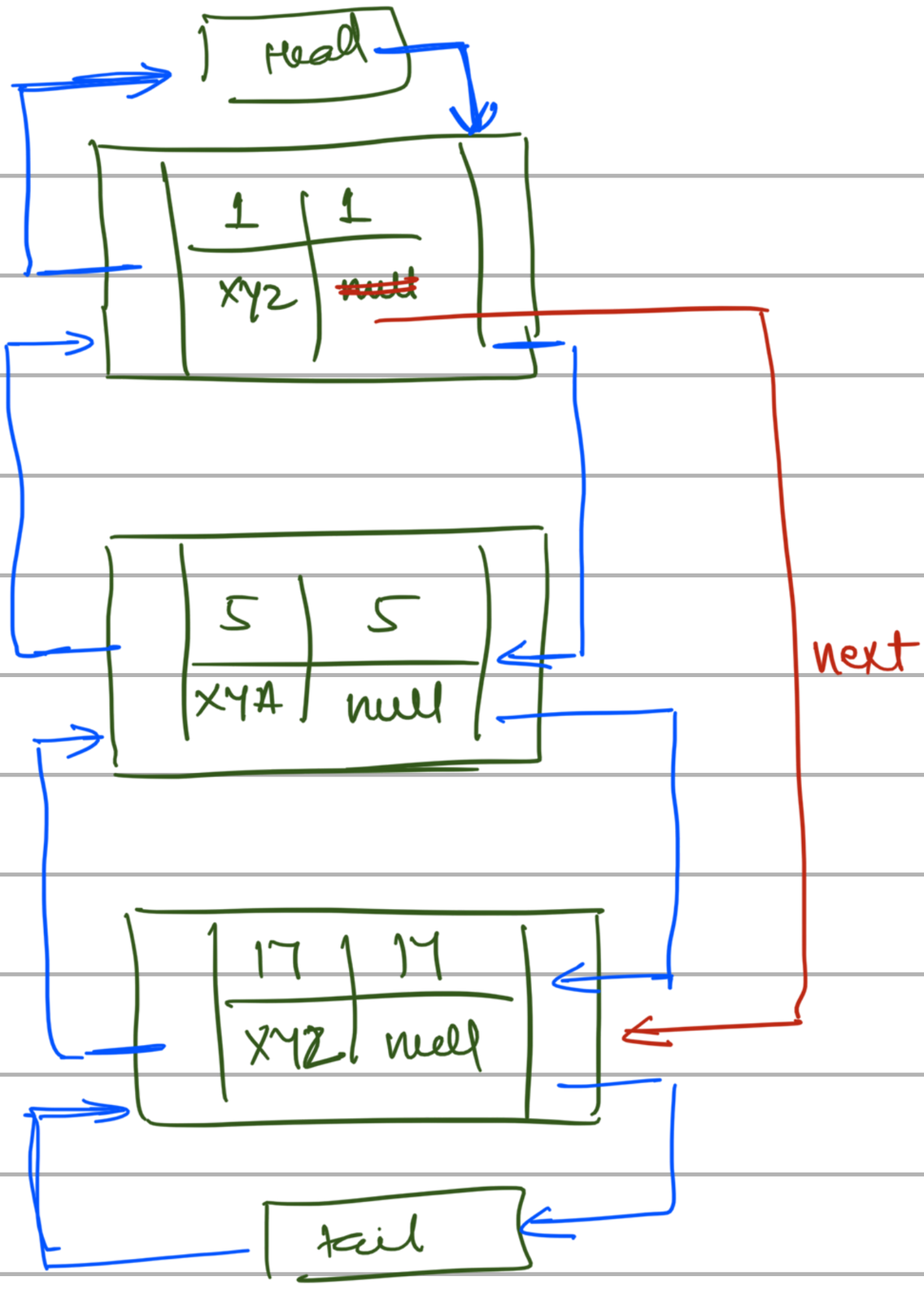


Chaining
(when collision)





THRIVE WITH ASHISH



Simplified Representation of How it Stores Insertion Order.

@thriverashish → insta / twitter / linkedin.

TREEMAP INTERNALS

@thriverashish

https://www.linkedin.com/in/thriverashish

video link -- https://youtu.be/QyAetW0mbEE

TreeMap implement NavigableMap.

Key, Value

~~Hashing~~

TreeMap is Red Black Tree based Navigable Map implementation

Self Balancing BT

In TreeMap entries are Sorted in Natural Ordering of its Keys

1, Ashish
2, Gupta
5, XYZ
3, PQR

Key: { String, Integer, Float }

Ashish, Gupta, PQR, XYZ.

Key as Custom Object → Employee {

3

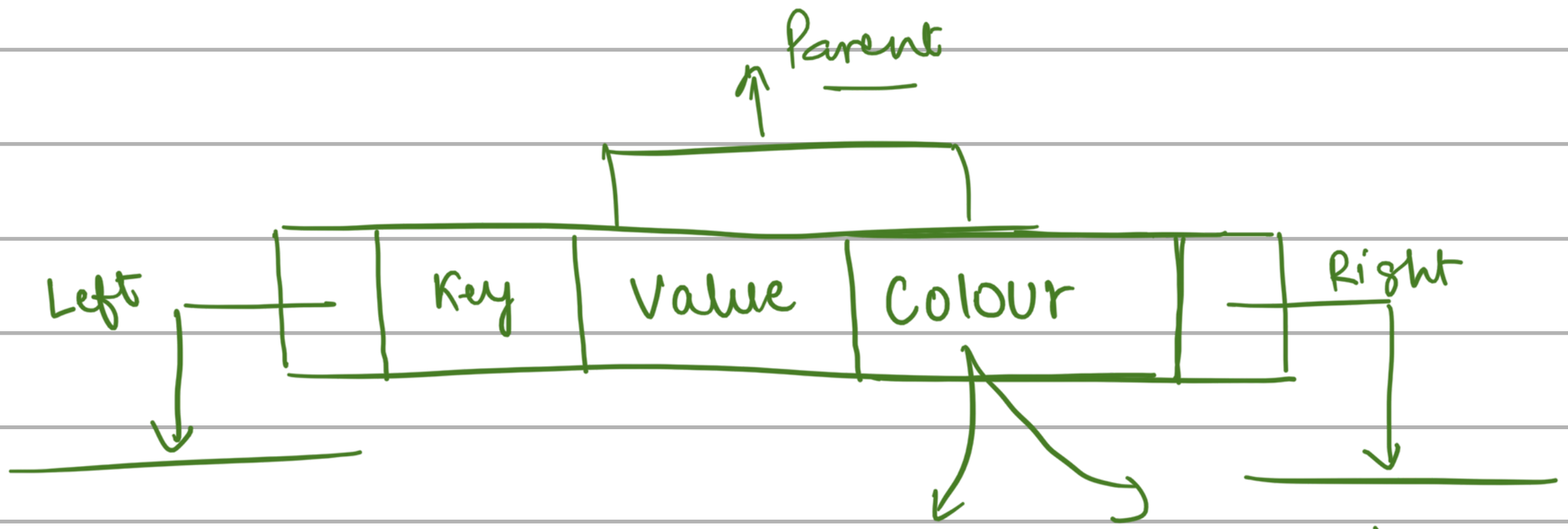
Two ways

1 Implement Comparable interface in the Class Employee used for Keys of TreeMap

(2) Supply an implementation of Comparator that would comparing outside key class.

= new TreeMap (new Comparator{

};);



Left will always be logically less than Parent

RED / BLACK

Right will always be logicall greater than parent.

~~equals()~~ and ~~hashCode()~~

Time Complexity

	Self Bal BST
Insertion	O(log n)
Lookup/Search →	O(log n)

TreeMap is not Synchronized

↳ do it using Collection.Synchronized SortedMap.

* TreeMap does not allow Null Keys

but multiple null values can be associated with diff Keys

1, null
2, null
3, null

* Initial Capacity ?

↳ does not allow an initial size because it can grow dynamically if need.

@Thriverashish → insta | twitter | linkedin.
Educational.

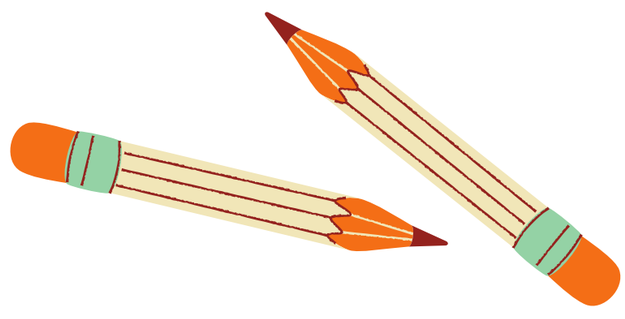
GOOD LUCK

@thriverashish

https://www.linkedin.com/in/thriverashish

https://www.youtube.com/c/thriverashish

All the best!



THRIVE WITH ASHISH