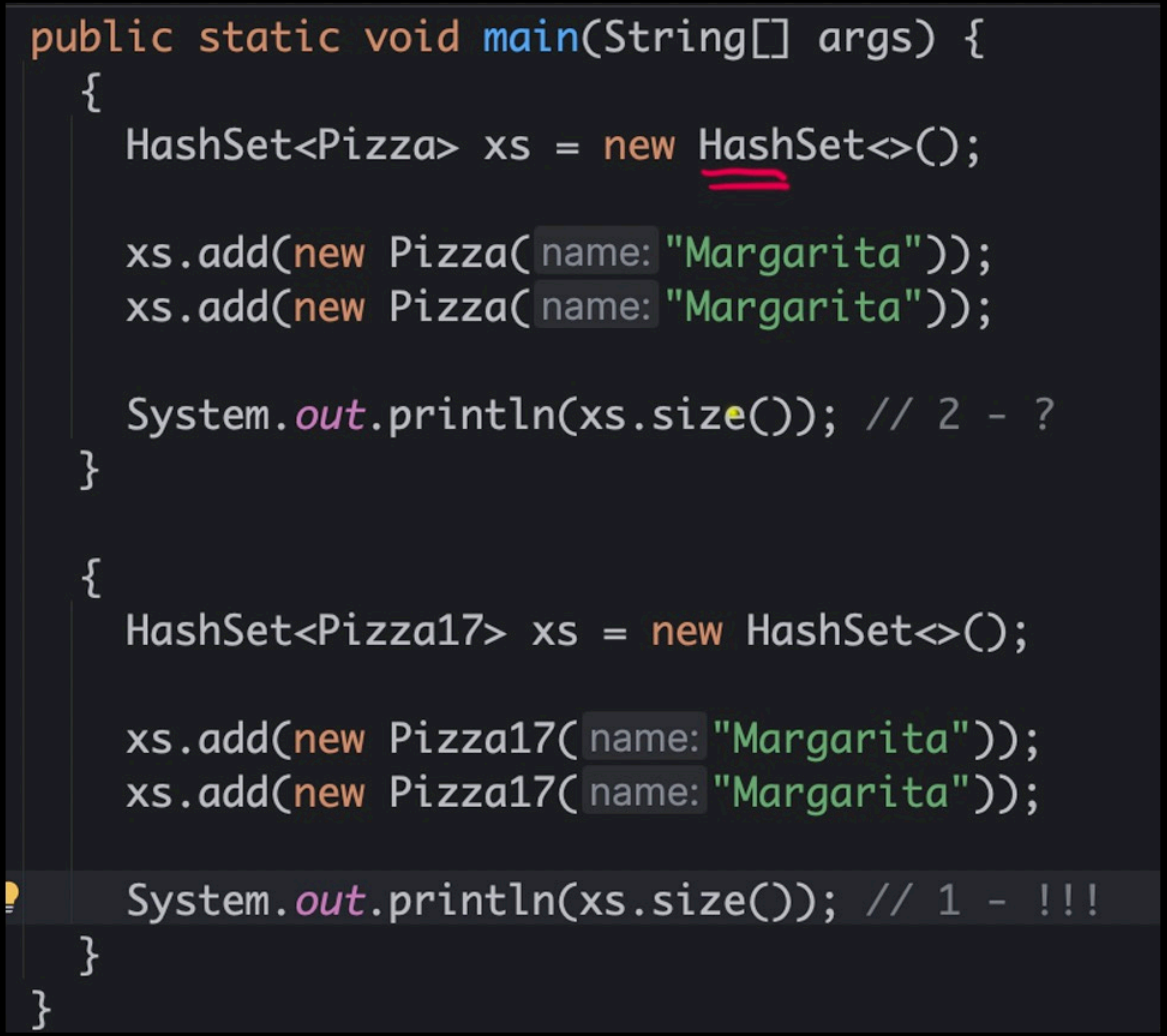
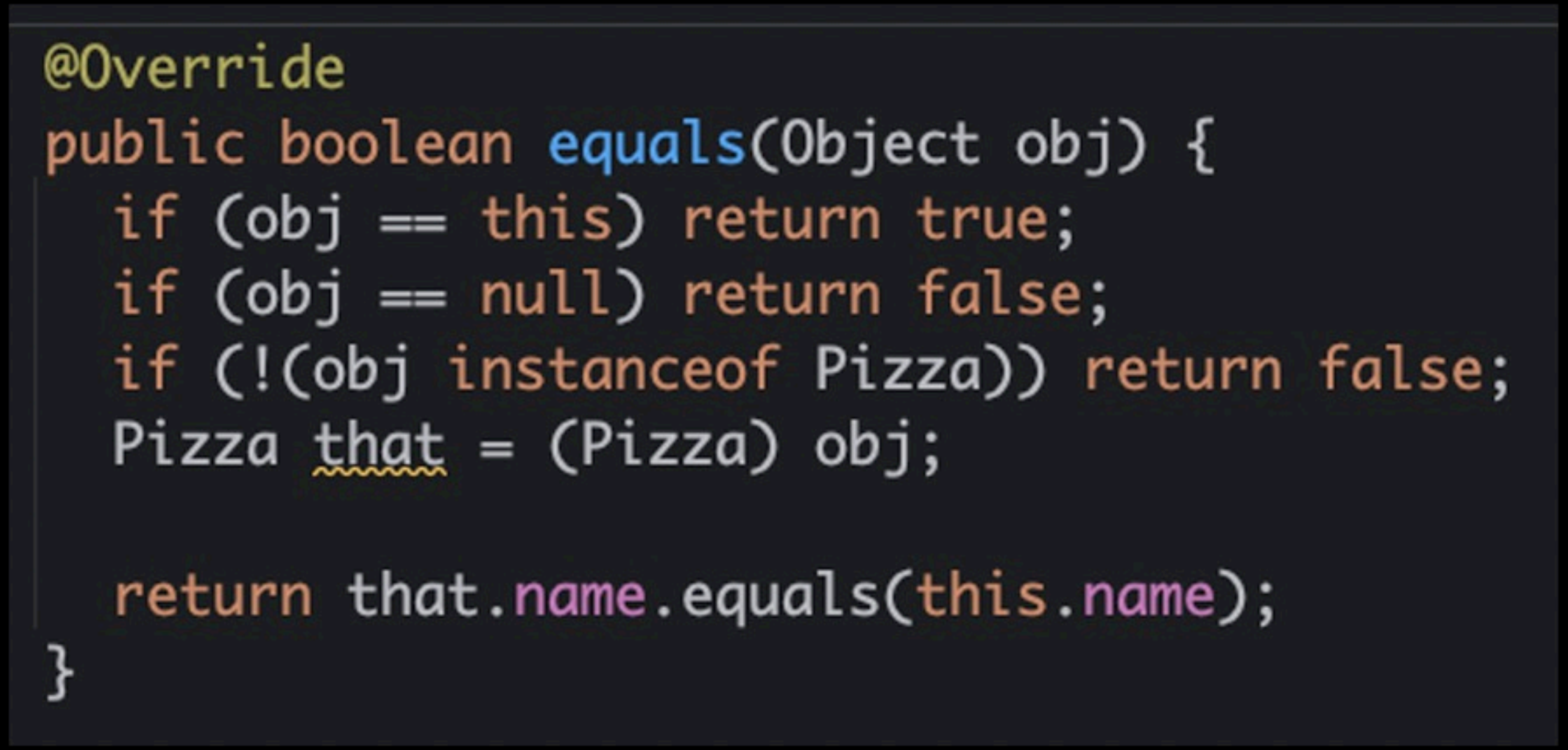


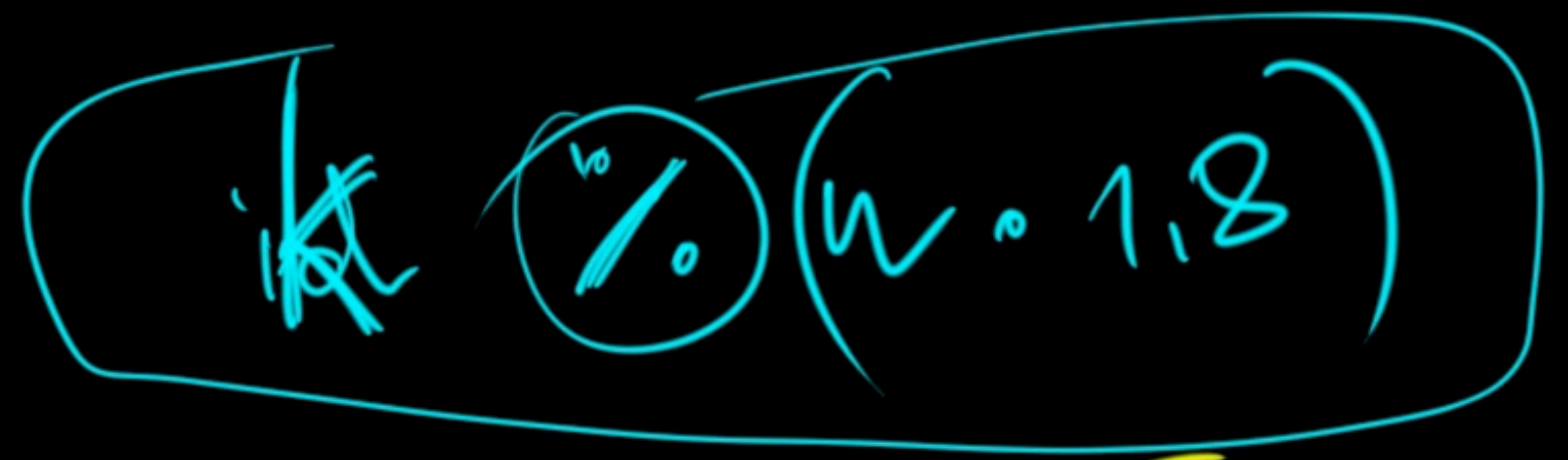
```java
public static void main(String[] args) {
    {
        HashSet<Pizza> xs = new HashSet<>();

        xs.add(new Pizza( name: "Margarita"));
        xs.add(new Pizza( name: "Margarita"));

        System.out.println(xs.size()); // 2 - ?
    }

    {
        HashSet<Pizza17> xs = new HashSet<>();

        xs.add(new Pizza17( name: "Margarita"));
        xs.add(new Pizza17( name: "Margarita"));

        System.out.println(xs.size()); // 1 - !!!
    }
}
```



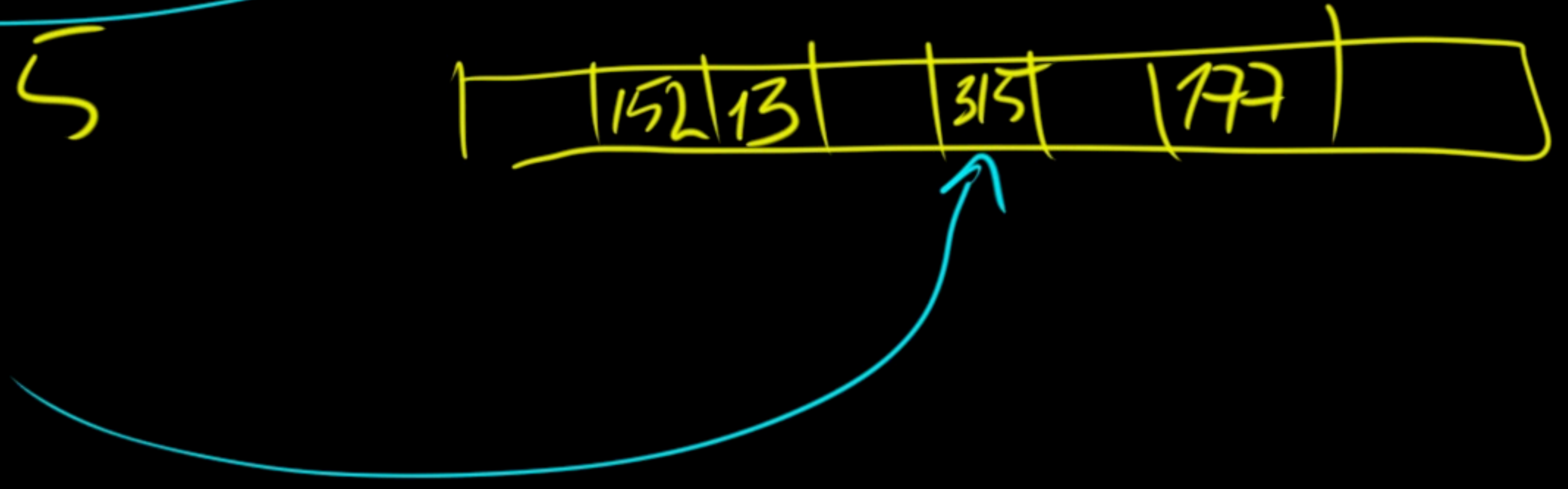
```java
@Override
public boolean equals(Object obj) {
    if (obj == this) return true;
    if (obj == null) return false;
    if (!(obj instanceof Pizza)) return false;
    Pizza that = (Pizza) obj;

    return that.name.equals(this.name);
}
```

HashMap <K,V>
HashSet <K> $O(1)$

.contains(x)
.get(K)



13 152 177 315



k = Pizza
hashcode

class Pizza

Set<Pizza>

String a

int b

double c

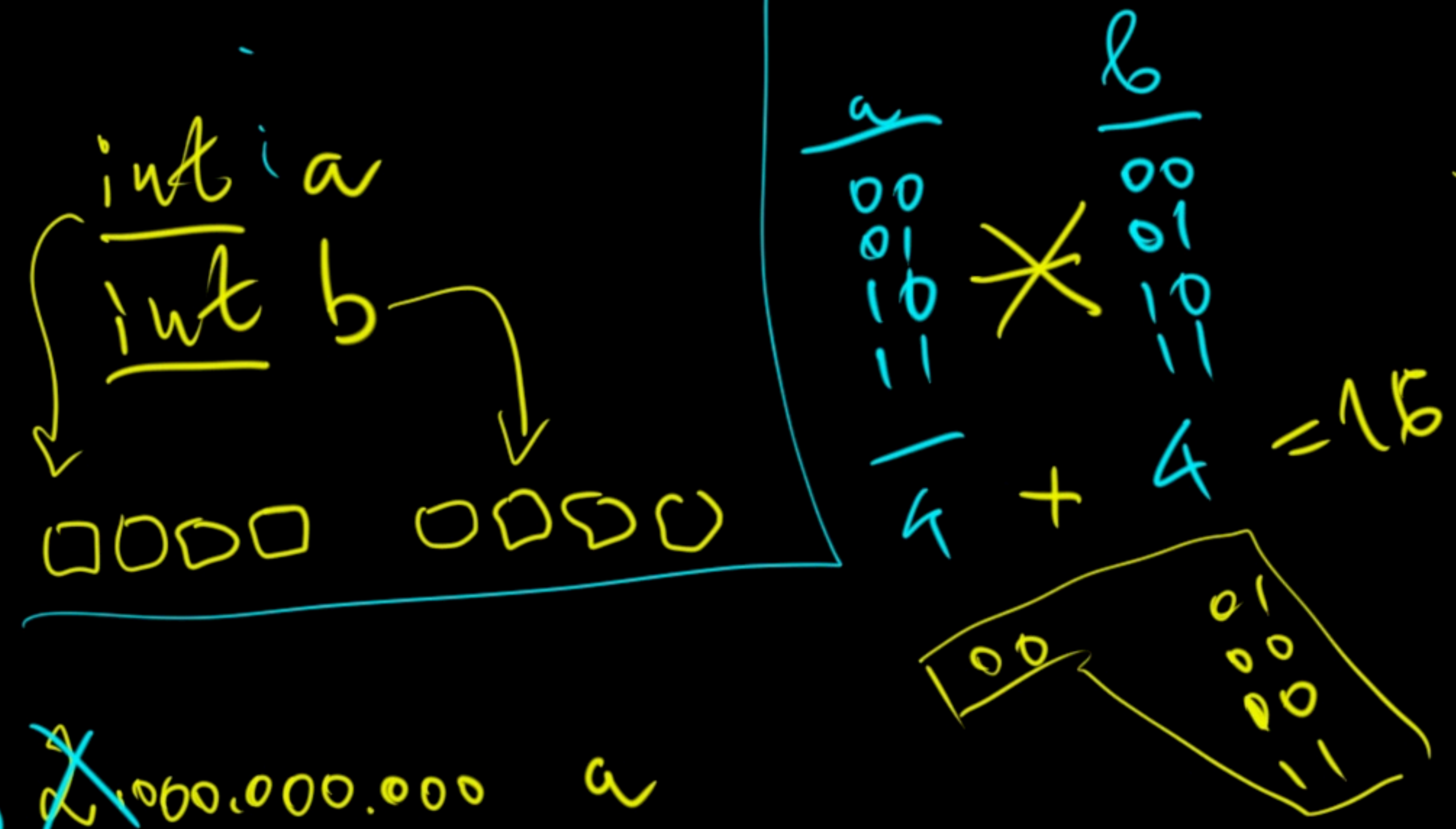
...

int hashcode ≈ equals

+ fast

- not precise

hash



00

01

10

11

4

2^{32}=int

-2.000.000.000 - 2.000.000.000

1~~2~~.000.000.000 a

1~~2~~.000.000.00 b

p1 = "M"

p2 = "M"

p1.a == p2.a

&&

p1.b == p2.b

&&

...

long, slow

Hash *

$O(1)^*$

$\boxed{\text{hashcode}}$ int

+ fast

- not precise

equals

- slow

+ precise

HashMap

HashSet

h1 vs h2

$h1 \neq h2 \longrightarrow o1 \neq o2$

$\boxed{h1 = h2}$ + $\boxed{\text{equals} \begin{array}{l} \nearrow = \\ \searrow \neq \end{array}}$

0,1%

```
HashSet<Pizza> xs = new HashSet<>();

Pizza p1 = new Pizza( name: "Margarita",  size: 30);
Pizza p2 = new Pizza( name: "Margarita",  size: 30);
xs.add(p1);
xs.add(p2);

System.out.println(xs.size());
System.out.println("---------");
```

```
calculating hashcode...
calculating hashcode...
calculating equals...
```

$(hc1, hc2) \rightarrow =$

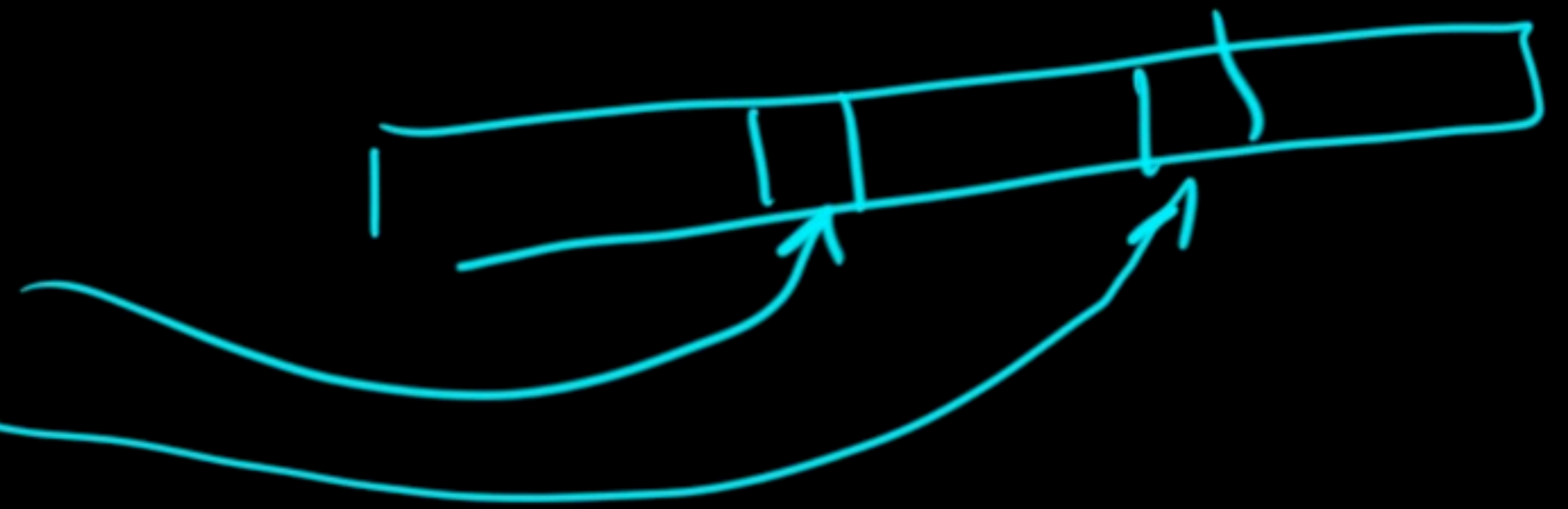
=

```
HashSet<Pizza> xs = new HashSet<>();

Pizza p1 = new Pizza( name: "Margarita",  size: 60);
Pizza p2 = new Pizza( name: "Margarita",  size: 30);
xs.add(p1);
xs.add(p2);

System.out.println(xs.size());
System.out.println("---------");
```

```
calculating hashcode...
calculating hashcode...
2
```

$(hc1, hc2) \neq$

no need to
run equals



Point
x: byte
y: byte
color: short
hash int
32bit
5
6.5
abcdef
a ~ 1 byte
a ... z
26 ~ 32
00001
00010
32 11111
5 bit

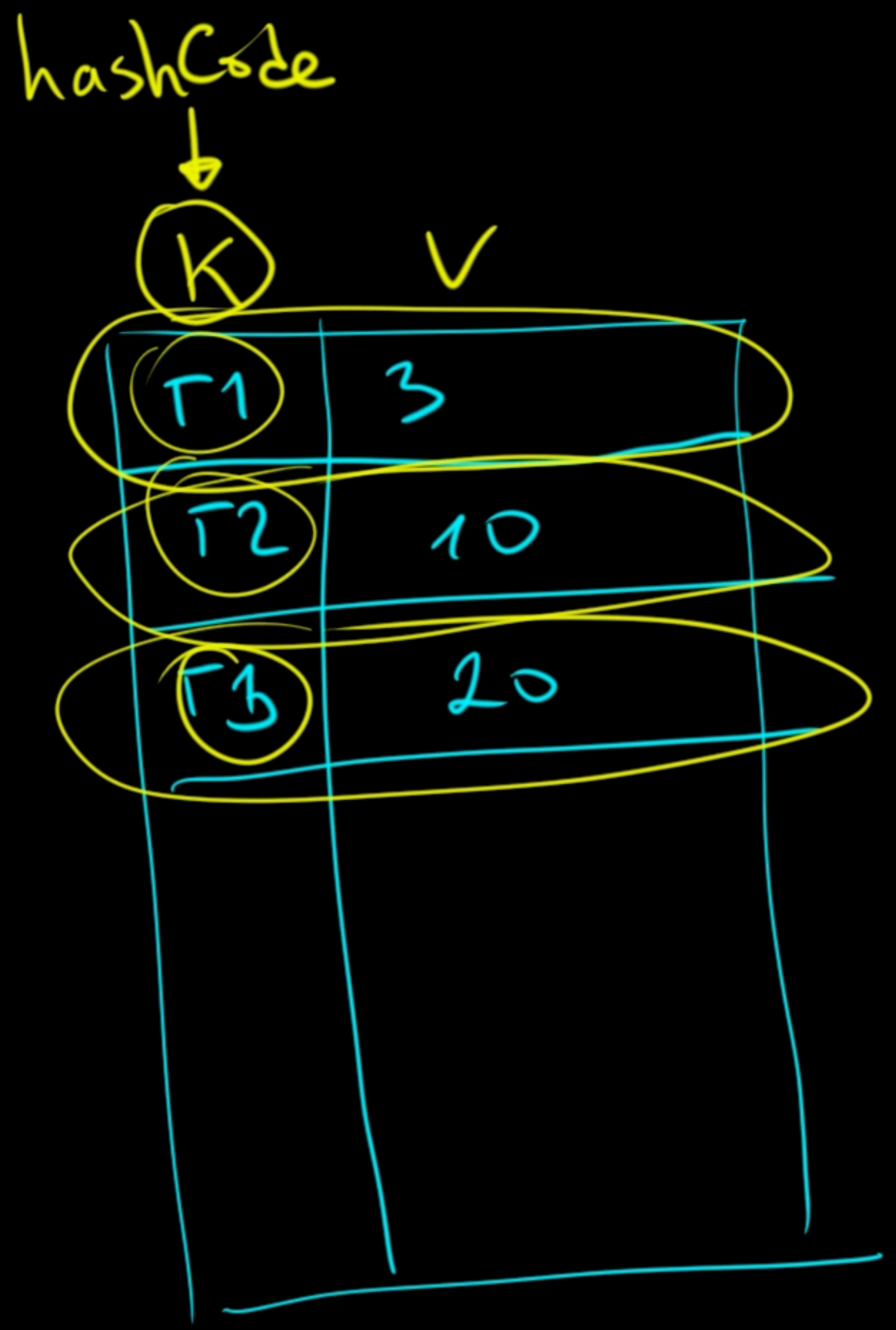
List $\rightarrow$ Hash<u>Set</u> $\rightarrow$ for {

list $\rightarrow$ HashSet $\rightarrow$ list $\rightarrow$ sort }

HashMap   TreeMap.

Map < K , V >

Map<K,V> ≡ Set< X<K,V> >



```java
if (m.containsKey(s2)) {
  int c = m.get(s2);
  c = c + 1;
  m.put(s2, c);
} else {
  m.put(s2, 1);
}
```

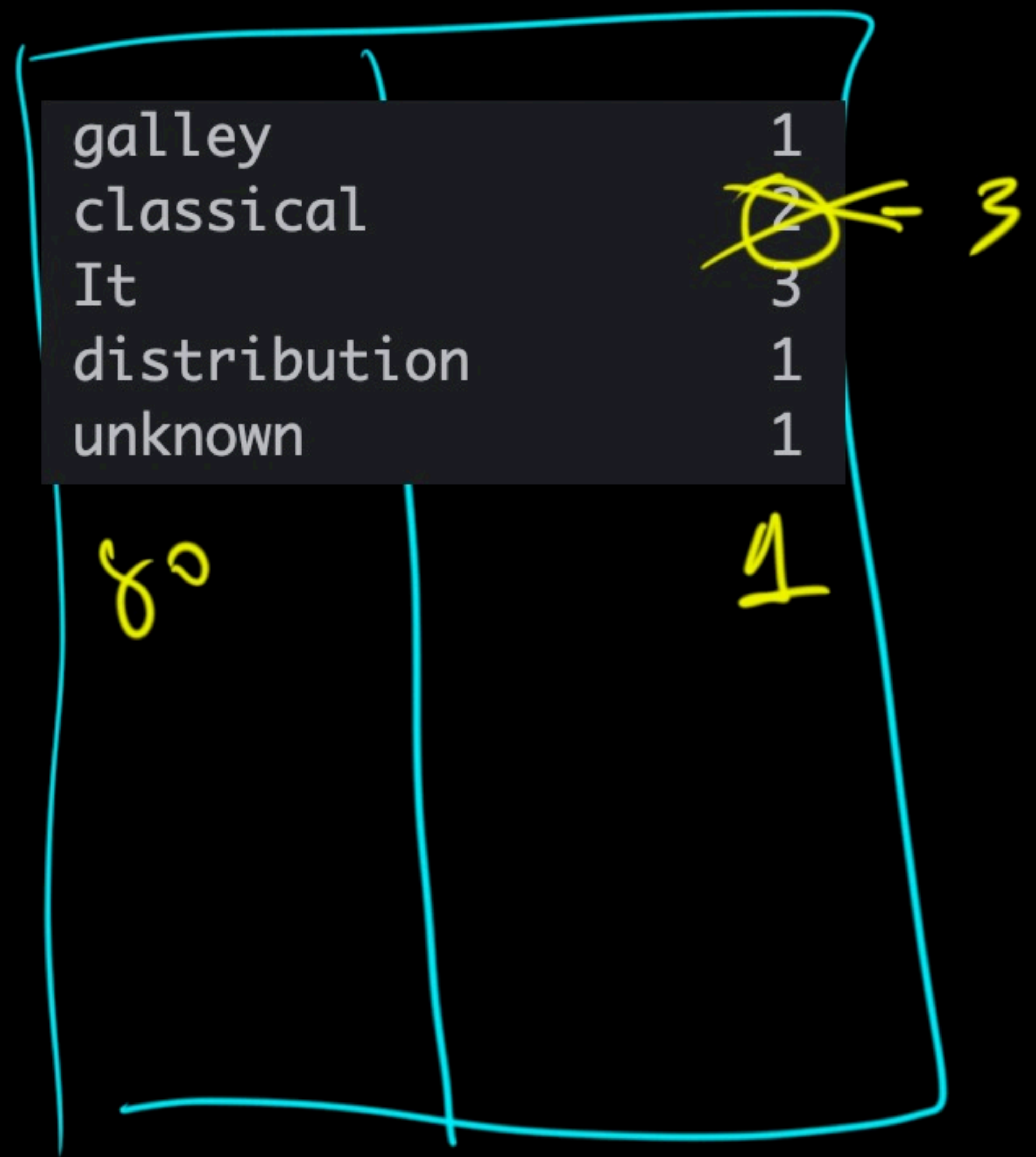
$Map<K,V> \approx Set<(K,V)>$

.containsKey

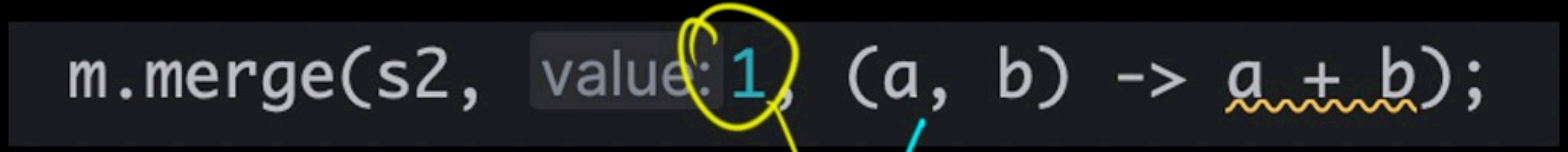
.contains

```java
HashMap<String, Integer> m = new HashMap<>();
```

```java
int c = m.getOrDefault(s2, defaultValue: 0);
m.put(s2, c + 1);
```

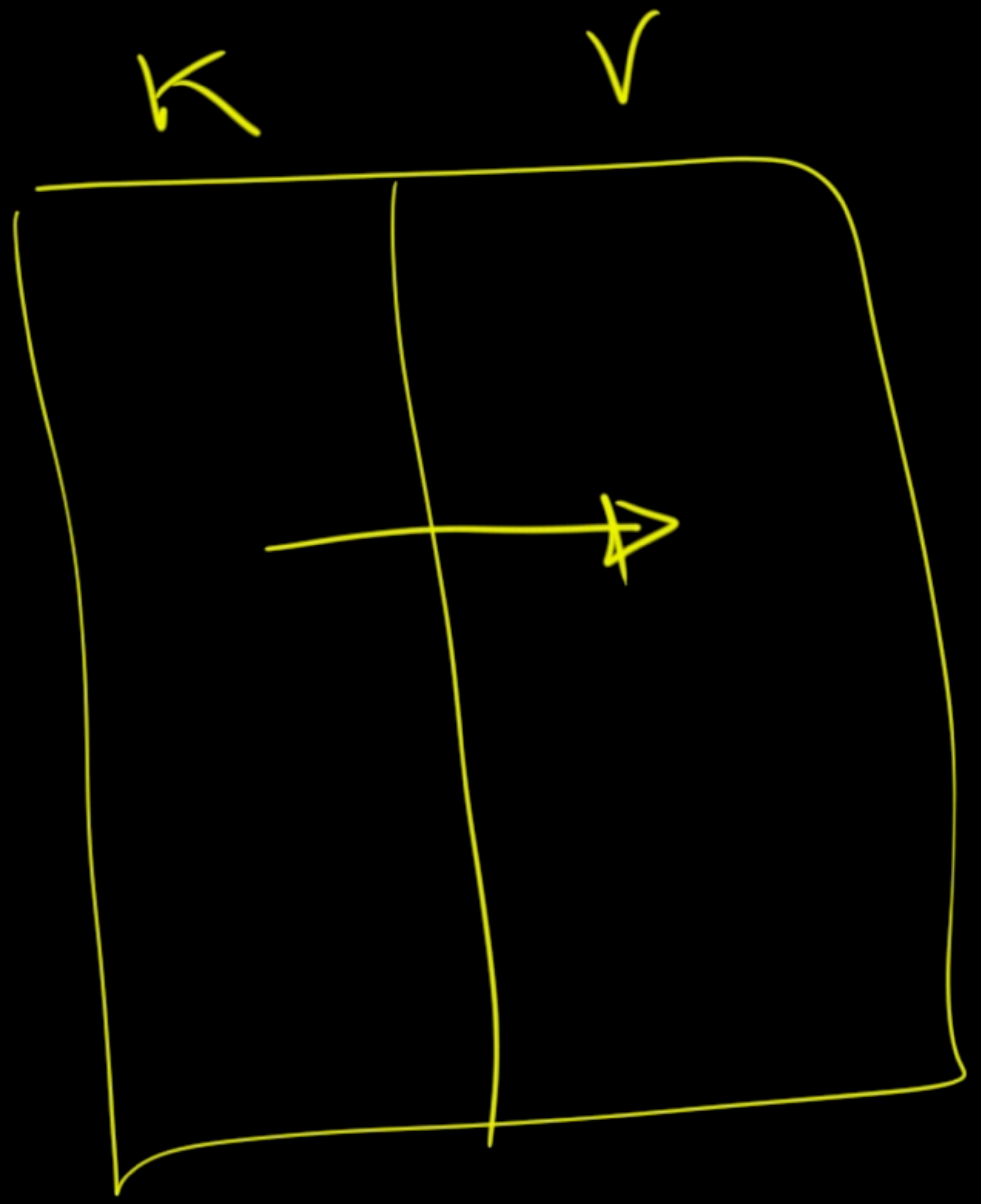


galley	1
classical	~~2~~ 3
It	3
distribution	1
unknown	1
80	1



```
m.merge(s2, value: 1, (a, b) -> a + b);
```

```
if (m.contains(s2))
    c = m.get(s2)
    c2 = c + 1
put(s2, c2)
```



```
Lorem Ipsum is simply dummy text of the printing
```

11

K	V
L	- 1
o	-2, 10, 15
r	- 3
e	- 4, 12
m	- 5, 20, 21

K = Character

V = List<Integer>

Int

String   x+1

Char   List<Int>
.add(x)

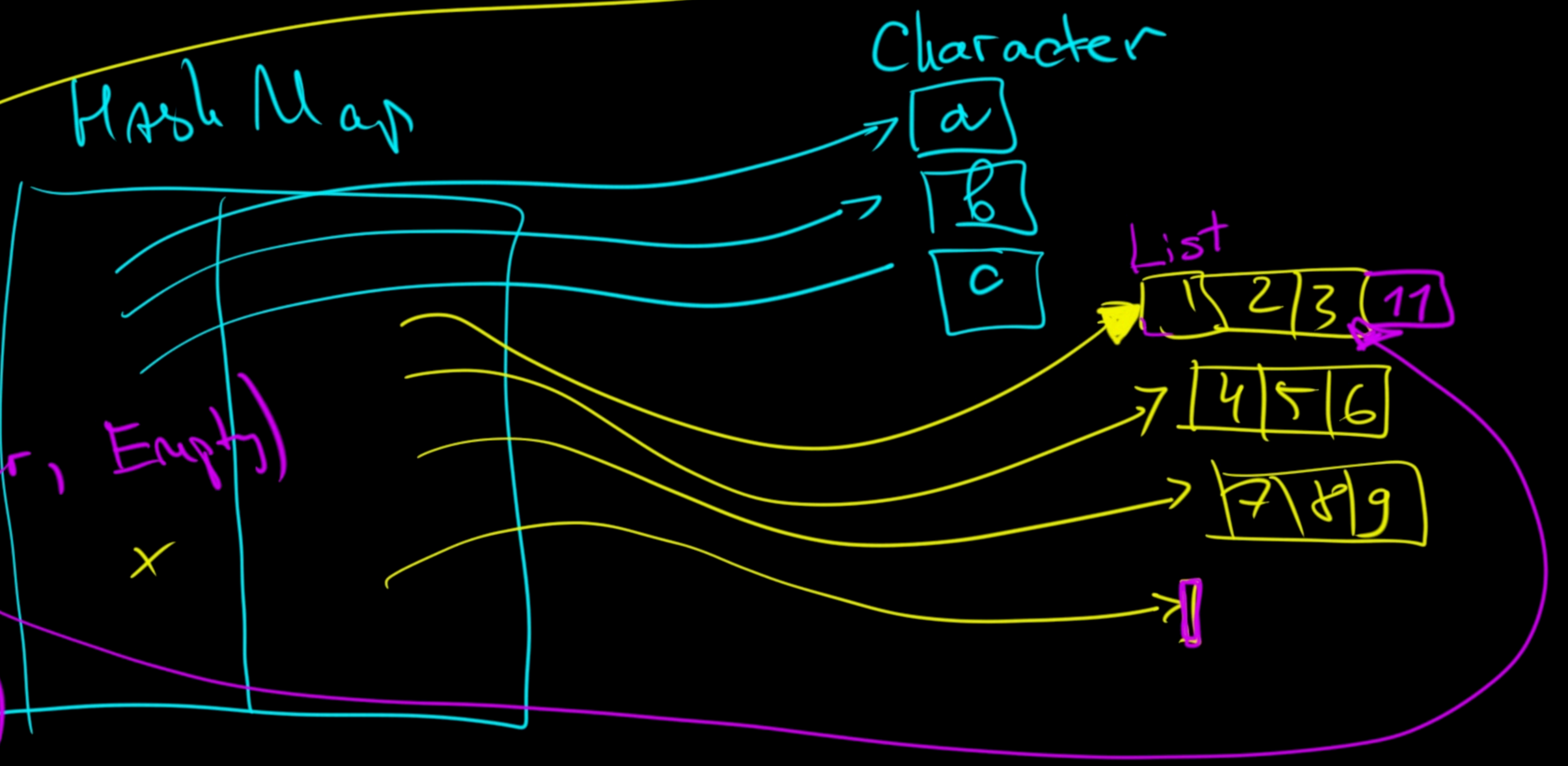
galley	1
classical	~~2~~ ← 3
It	3
distribution	1
unknown	1

```
List<Integer> positions = m.getOrDefault(c, new ArrayList<>());
positions.add(i);
m.put(c, positions);
```

Heap

```
if (!m.containsKey(c)) m.put(c, new ArrayList<>());
m.get(c).add(i);                                      ;
```

```
HashMap<Character, List<Integer>>
```



l = m.getOrDefault(letter, Empty)
l.add(i)
m.put(letter, l)