$b = f(a)$

$c = g(b)$

$d = h(c)$

[ ]

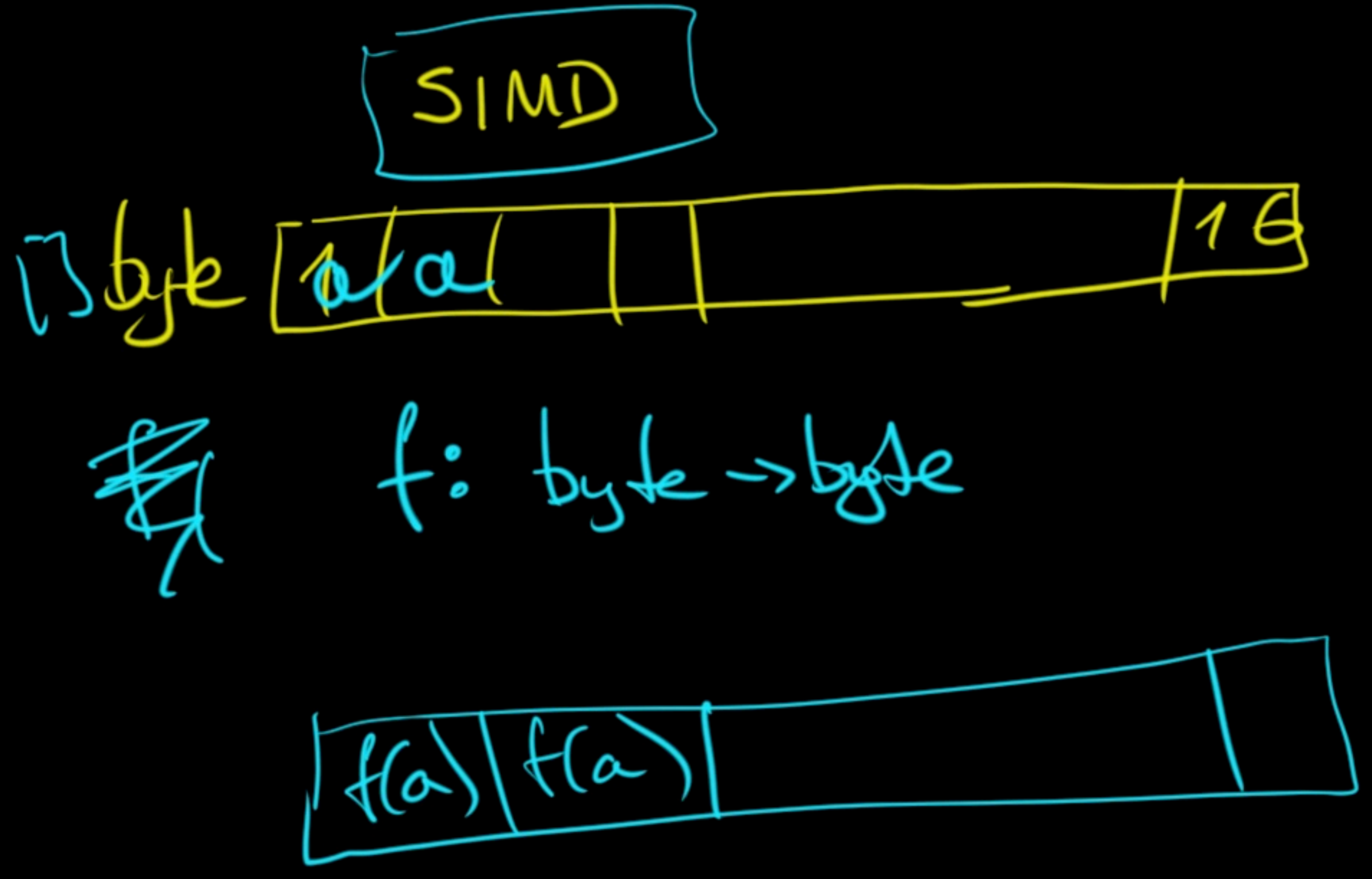
1+

0+

bw 128

byte[]

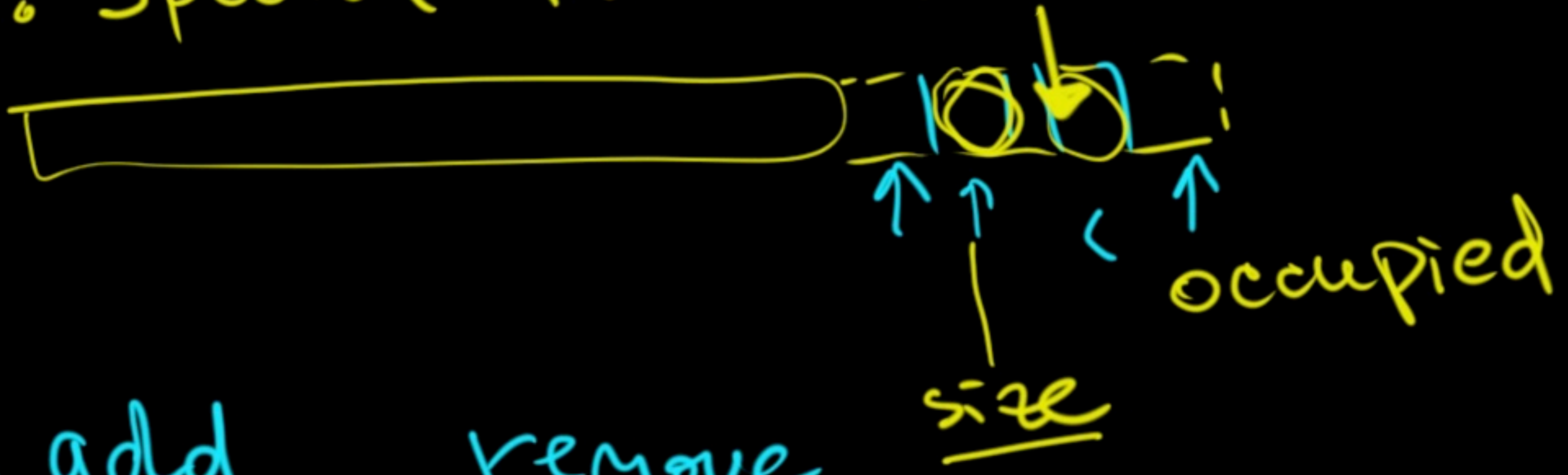
for



Java17

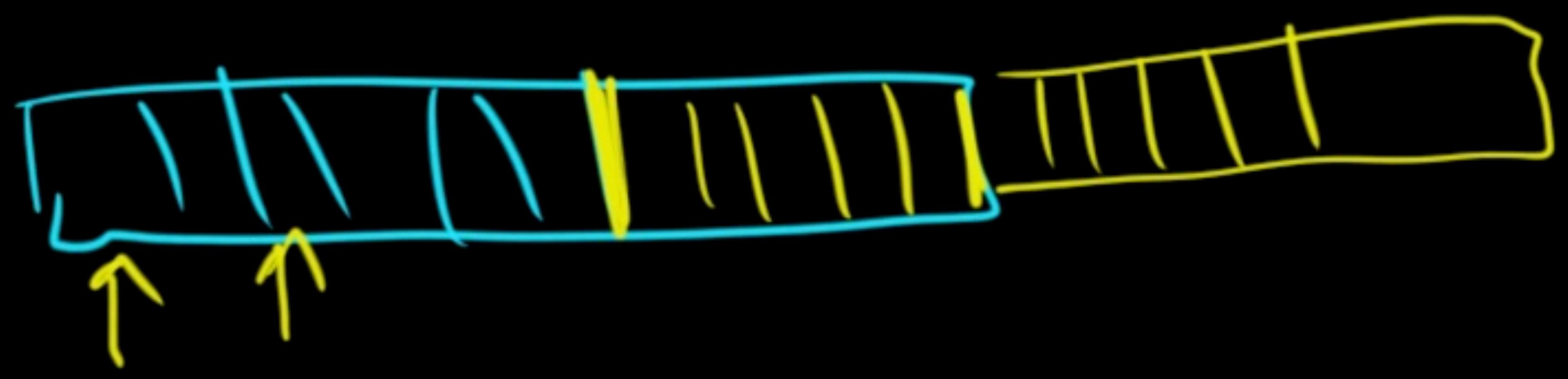
Collections

1. fixed size => auto grow
2. special functions

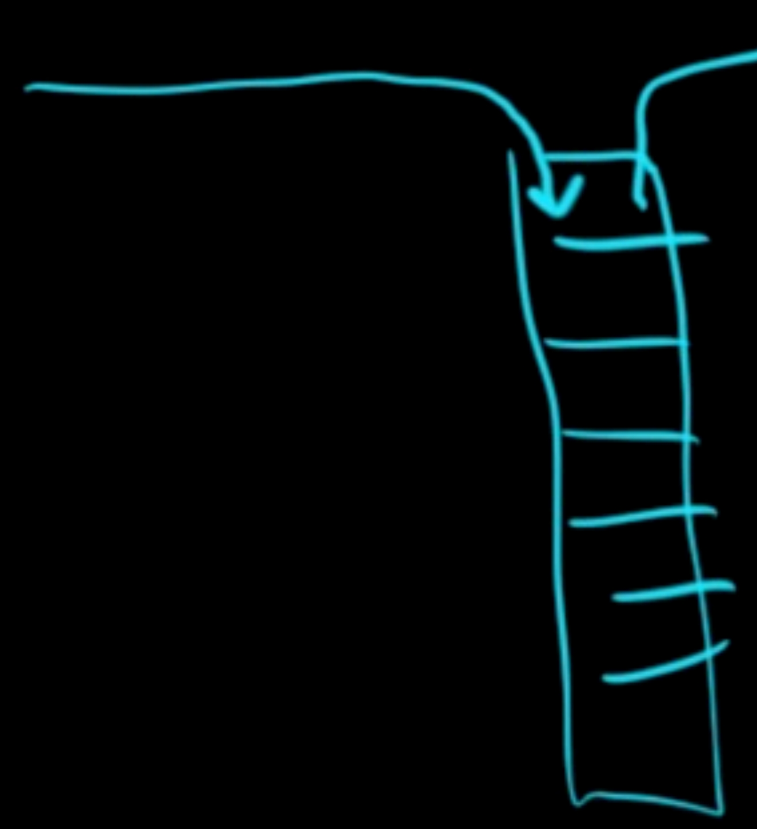


- . add . remove
- . get
- . set
- . isEmpty.
- . contains (x) -> boolean

- ArrayList



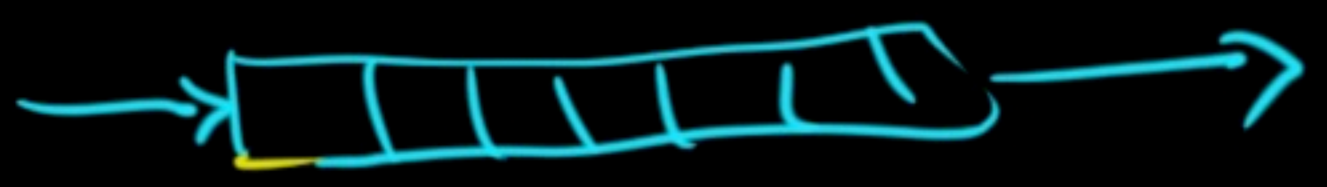
- stack



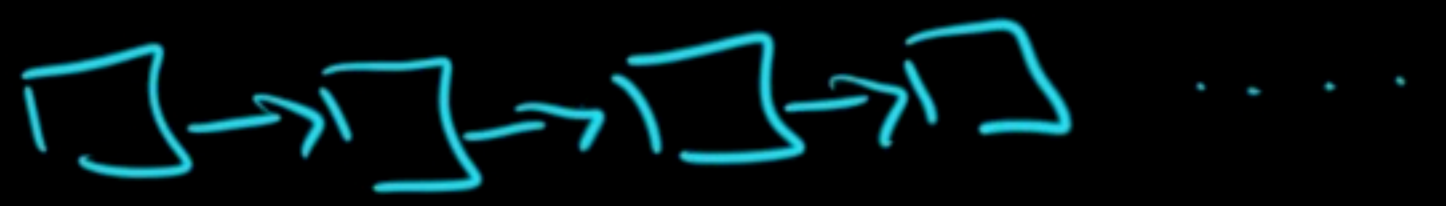
LIFO

FIFO

- Queue



- linkedlist



- Priority Queue 1,7,5,3

pop

- Set unique

1,2,3,1 ≡ 1,2,3

- Map

K	V
a	1,5,7
b	1,2
c	

abstract class

+ interfaces

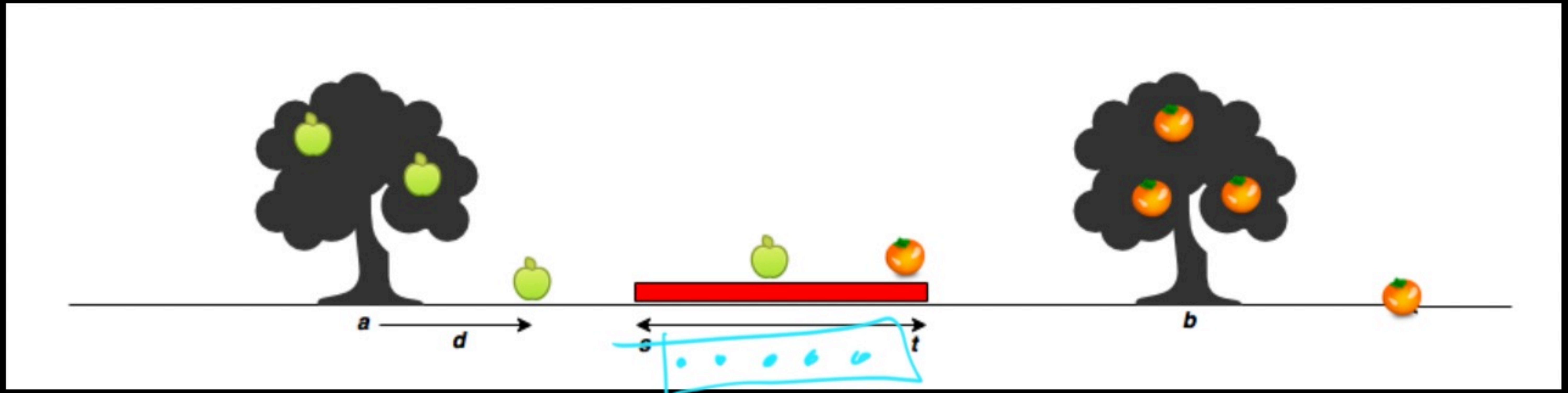
+ generics

(center , distanes []) → positions []

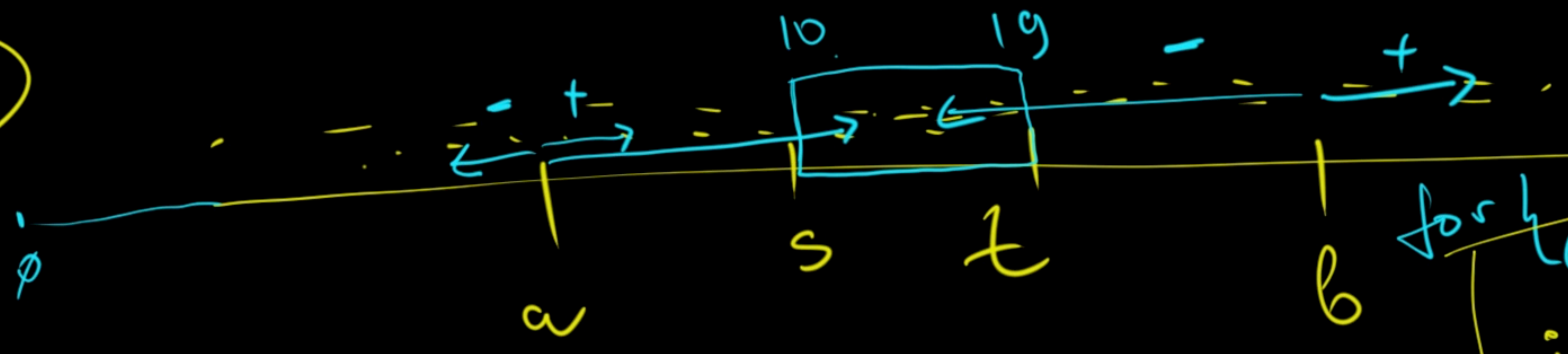
$s \le P_i \le t$

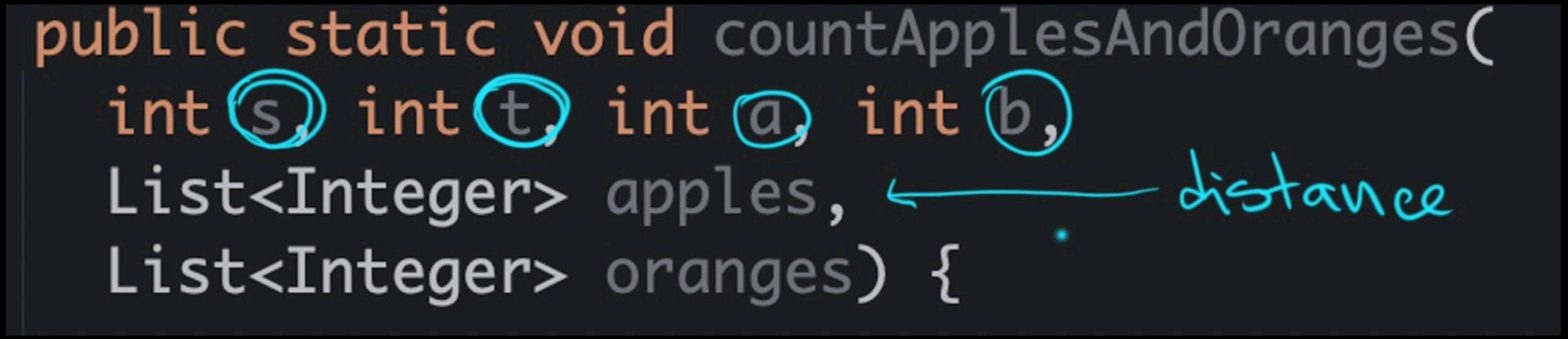
center ↓ 20	
-5	15
-2	18
-1	19
1	21
2	22

center 5 (dist)	positions
-1	4
-2	3
-3	2
+2	7
+4	9
+7	12



5   -5 -2 -1 20 +1 +2

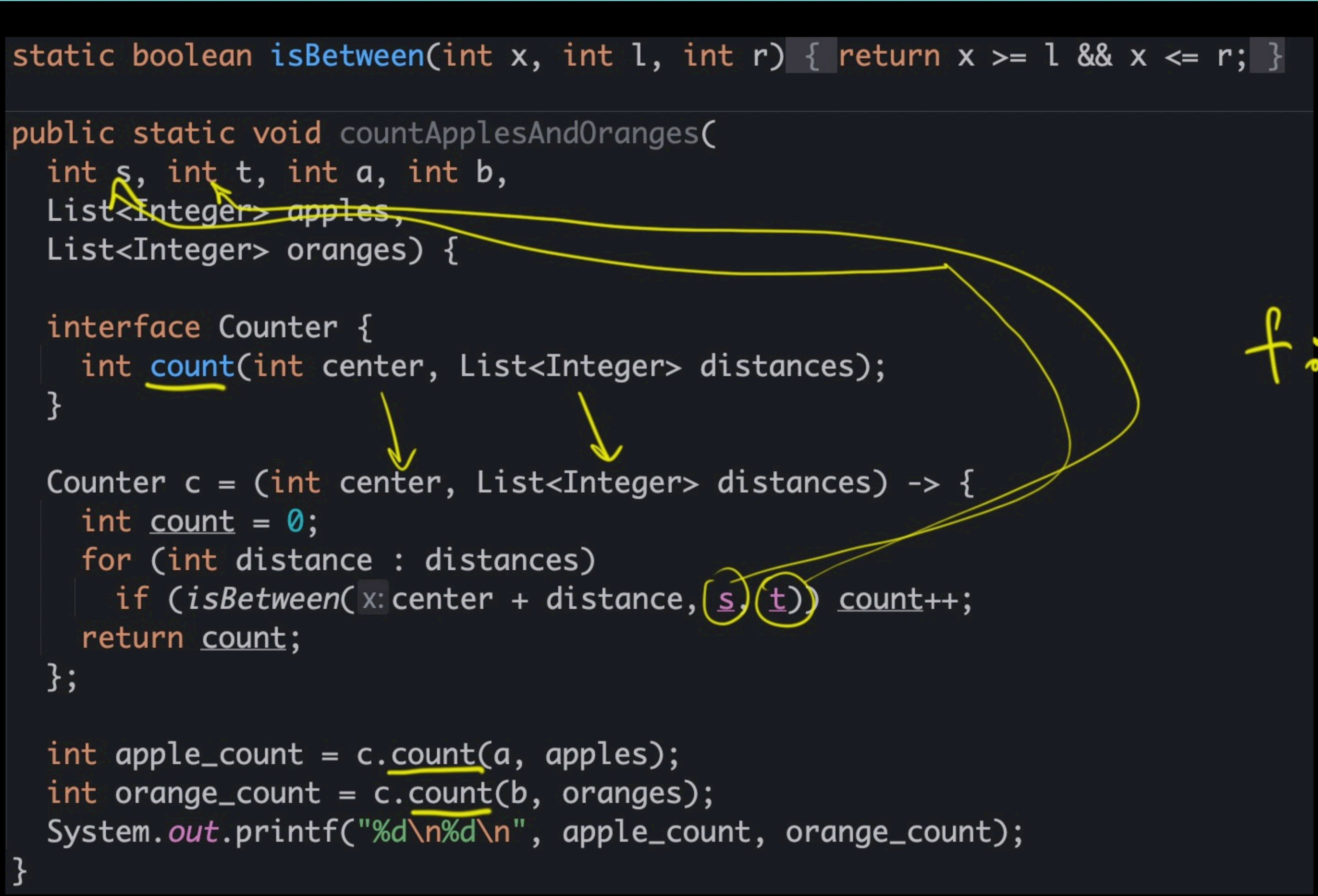




```java
public static void countApplesAndOranges(
  int s, int t, int a, int b,
  List<Integer> apples,
  List<Integer> oranges) {
```

← distance

```
for (c = a + ai :
  if (c >= s && c <= t)
    count++
}
```



```java
static boolean isBetween(int x, int l, int r) { return x >= l && x <= r; }

public static void countApplesAndOranges(
  int s, int t, int a, int b,
  List<Integer> apples,
  List<Integer> oranges) {

  interface Counter {
    int count(int center, List<Integer> distances);
  }

  Counter c = (int center, List<Integer> distances) -> {
    int count = 0;
    for (int distance : distances)
      if (isBetween(x:center + distance, s, t)) count++;
    return count;
  };

  int apple_count = c.count(a, apples);
  int orange_count = c.count(b, oranges);
  System.out.printf("%d\n%d\n", apple_count, orange_count);
}
```

$f:(A,B)\to C$

Set

1,2,1 → 1,2

1,2,1 → 2,1

↘ {1,2}

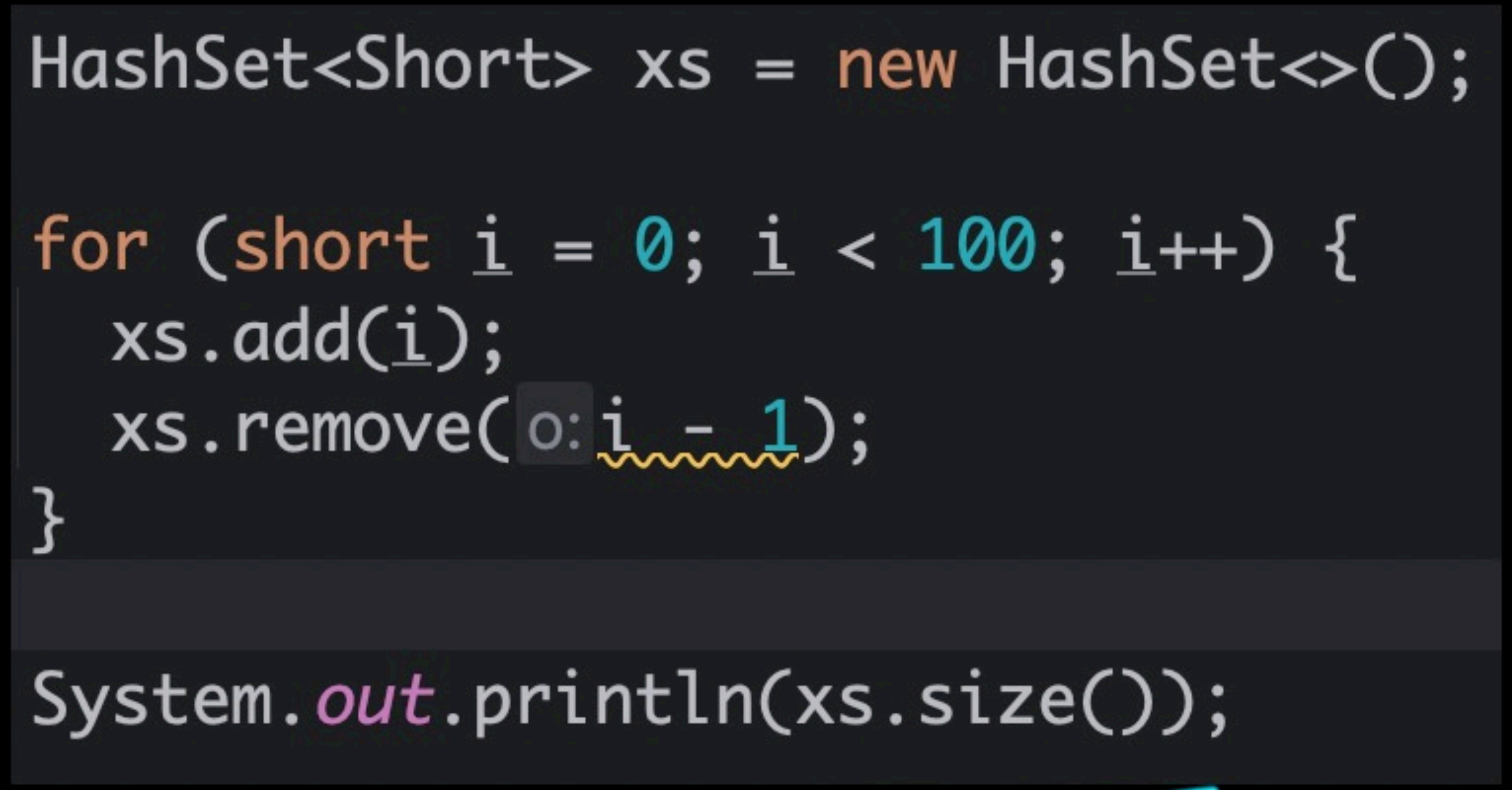
+ unique

- ordering

$\underbrace{[1,7,3,5,9,2]}_{\underline{\underline{n}}}$ .contains(x) → $O(\underline{\underline{n}})$

.contains(2)

HashSet .contains $O(\underline{\underline{1}})$



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

for (short i = 0; i < 100; i++) {
  xs.add(i);
  xs.remove( o: i - 1);
}

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

1

add	remove	
0	-1	[0]
1	0	[1]
⋮		
99	-98	[99]

Java!

$t1 \oplus t2 \rightarrow max(size\_of(t1), size\_of(t2), size\_of(int))$

$+$
$-$
$*$
$/$
$\%$
$\vdots$

long, _ => long

int, long => long

int, _ => int

_, _ => int

int = short_int

x-

Rust: $t \oplus t \rightarrow t$

$(\underset{i8}{x}\ \underline{\text{as } i32}) + (\underset{i16}{y}\ \underline{\text{as } i32})$