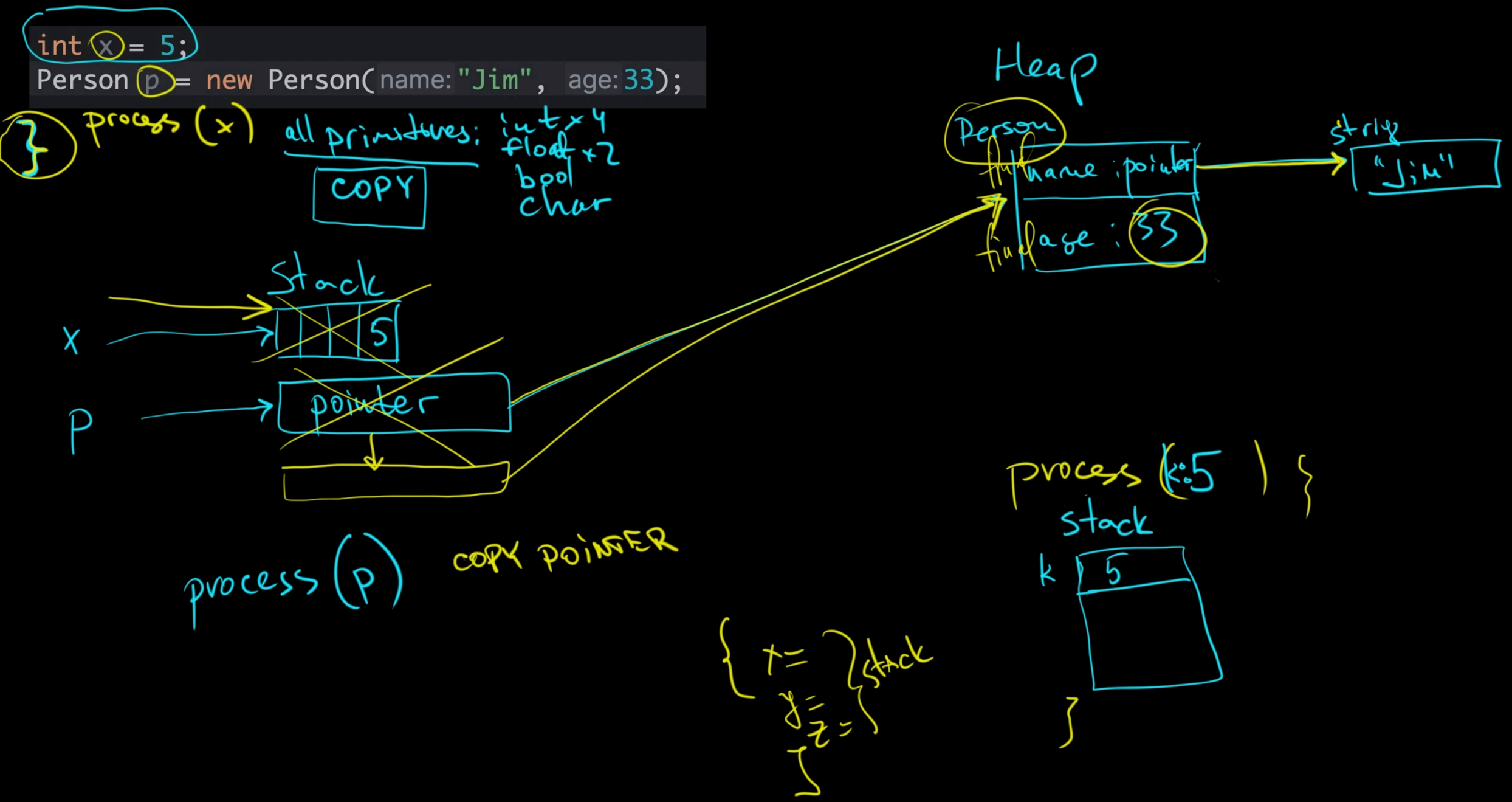
```
record Person(String name, int age)
```

```
int x = 5;
Person p = new Person(name: "Jim", age: 33);
```



```java
static void process(int k) {
  System.out.printf("Processing %d\n", k);
  k = 100;
  System.out.printf("Processing %d\n", k);
}

public static void main(String[] args) {
  int x = 5;
  System.out.printf("Processing main %d\n", x);
  process(x);
  System.out.printf("Processing main %d\n", x);
}
```

COPY VALUE

```
Processing main 5
Processing 5
Processing 100
Processing main 5
```

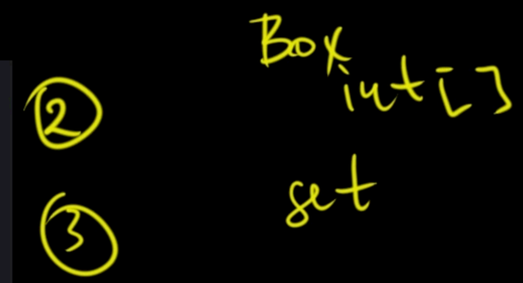
int x4
float x2
bool
char

```java
static void process(Person k) {
  System.out.printf("Processing main %s\n", k);
  k.age = 100;
  System.out.printf("Processing main %s\n", k);
}

public static void main(String[] args) {
  Person p = new Person( name: "Jim",  age: 33);
  System.out.printf("Processing main %s\n", p);
  process(p);
  System.out.printf("Processing main %s\n", p);
}
```

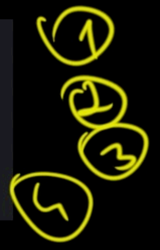
COPY POINTER

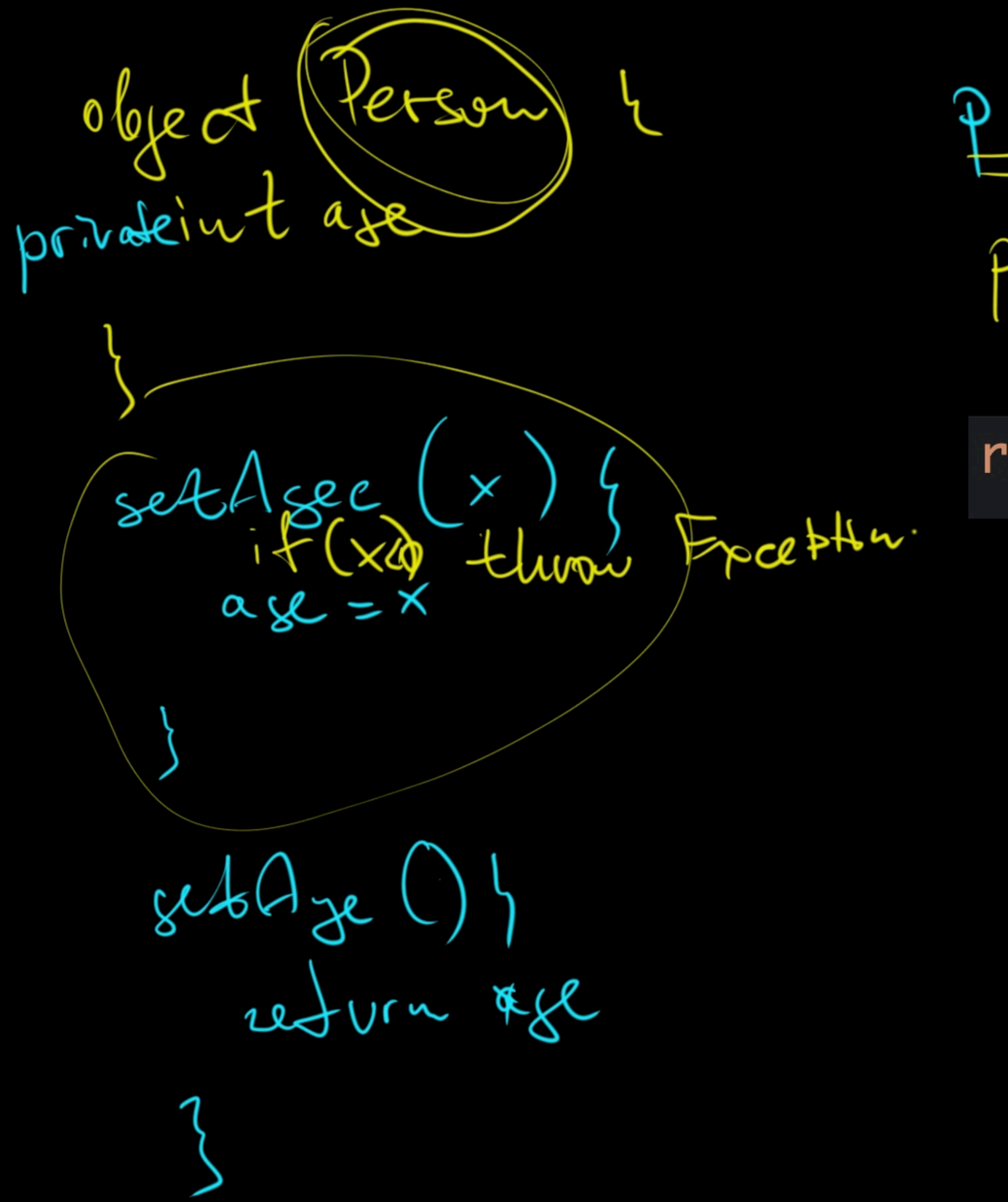
```
Processing main Person{name='Jim', age=33}
Processing main Person{name='Jim', age=33}
Processing main Person{name='Jim', age=100}
Processing main Person{name='Jim', age=100}
```

```java
static void process(int[] ks) {
  System.out.printf("Processing main %s\n", Arrays.toString(ks));
  ks[1] = 100;
  System.out.printf("Processing main %s\n", Arrays.toString(ks));
}

public static void main(String[] args) {
  int[] xs = {1,2,3};
  System.out.printf("Processing main %s\n", Arrays.toString(xs));
  process(xs);
  System.out.printf("Processing main %s\n", Arrays.toString(xs));
}
```




```
Processing main [1, 2, 3]
Processing main [1, 2, 3]
Processing main [1, 100, 3]
Processing main [1, 100, 3]
```





p.age = 33 -13

p.setAge(-13)

final final

```
record Person17(String name, int age)
```

no setters !!!

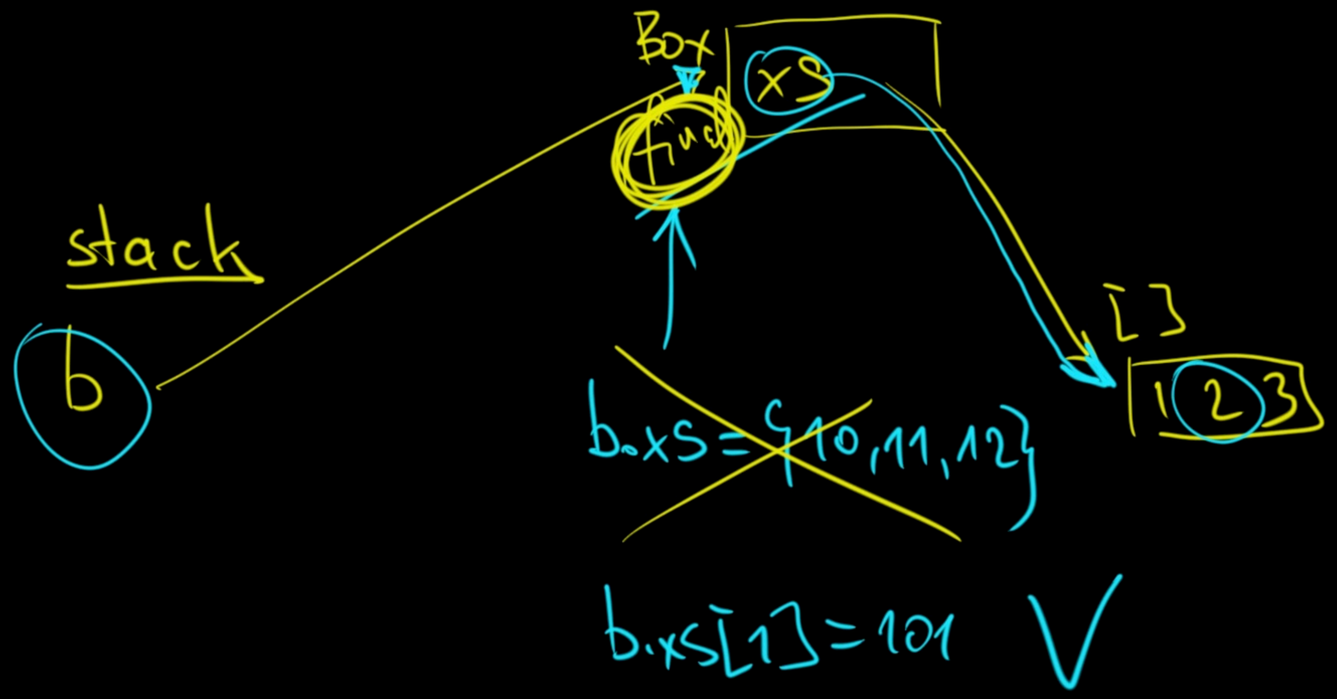
```
public class Box {
  final int[] xs = {1,2,3};
}
```

b.xs[1]=101

```
static void process(Box b) {
  System.out.printf("Processing main %s\n", b);
  b.get()[1] = 101;
  System.out.printf("Processing main %s\n", b);
}
```

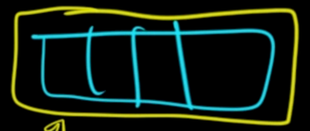
```
public static void main(String[] args) {
  Box b = new Box();
  System.out.printf("Processing main %s\n", b);
  process(b);
  System.out.printf("Processing main %s\n", b);
}
```

```
Processing main Box{xs=[1, 2, 3]}
Processing main Box{xs=[1, 2, 3]}
Processing main Box{xs=[1, 101, 3]}
Processing main Box{xs=[1, 101, 3]}
```



99%

final x = 5



final p = Person(33)

Person

final class X

class X extends X

memory

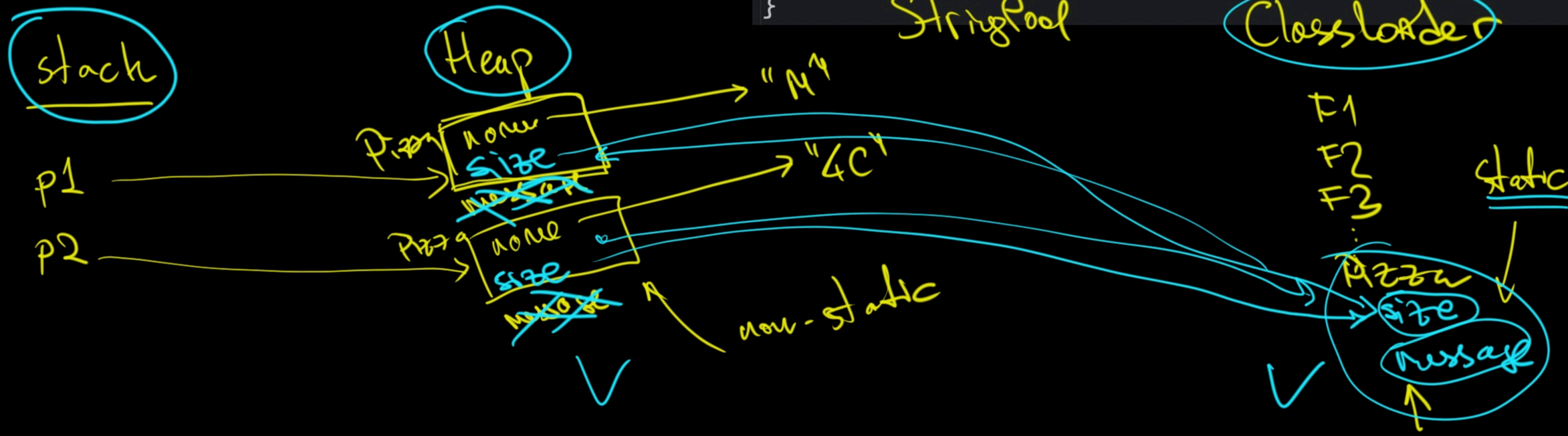
static

public final static

```
Pizza p1 = new Pizza( name: "M",  size: 30);
System.out.println(p1);

Pizza p2 = new Pizza( name: "4C",  size: 60);
System.out.println(p2);
System.out.println(p1);
// p1:Pizza{name='M',  size=30}
// p2:Pizza{name='4C', size=60}
// p1:Pizza{name='M', | size=60}
```

```
public class Pizza {
  String name;
  static int size;
  String message = "size must be at least 10";

  public Pizza(String name, int size) {
    this.name = name;
    if (size < 10)
      throw new IllegalArgumentException(message);
    this.size = size;
  }
  public String toString() {
    return "Pizza{name='%s', size=%d}".formatted(name, s
  }
}
```

StringPool



Class SoldPizza

static var — one for ALL instances

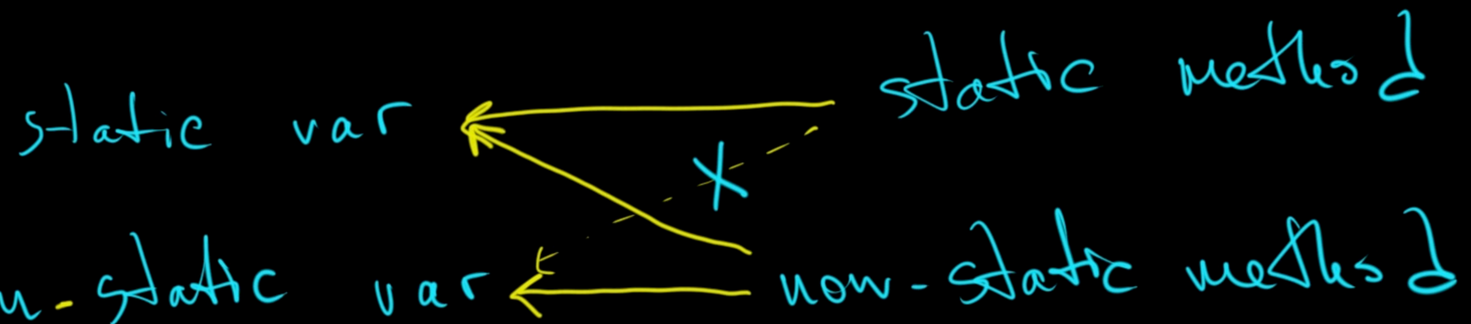
non-static var — one per instance

static method — one for ALL instances

non-static method — one per each instance

Calc.add(1,2)

c = new Calc
c.add(1,2)



```java
public class JavaMemory6 {

  int x;

  class Inside6 {
    int y;

    void whateverIwant() {
      int x1 = x;
    }
  }

  static class Inside6S {
    int y;

    void whateverIwant() {
      int x1 = x;
    }
  }
```

enum – special type ⟷ class Person (name String, age Int)

?

Color {
Red,
Green,
Yellow
}

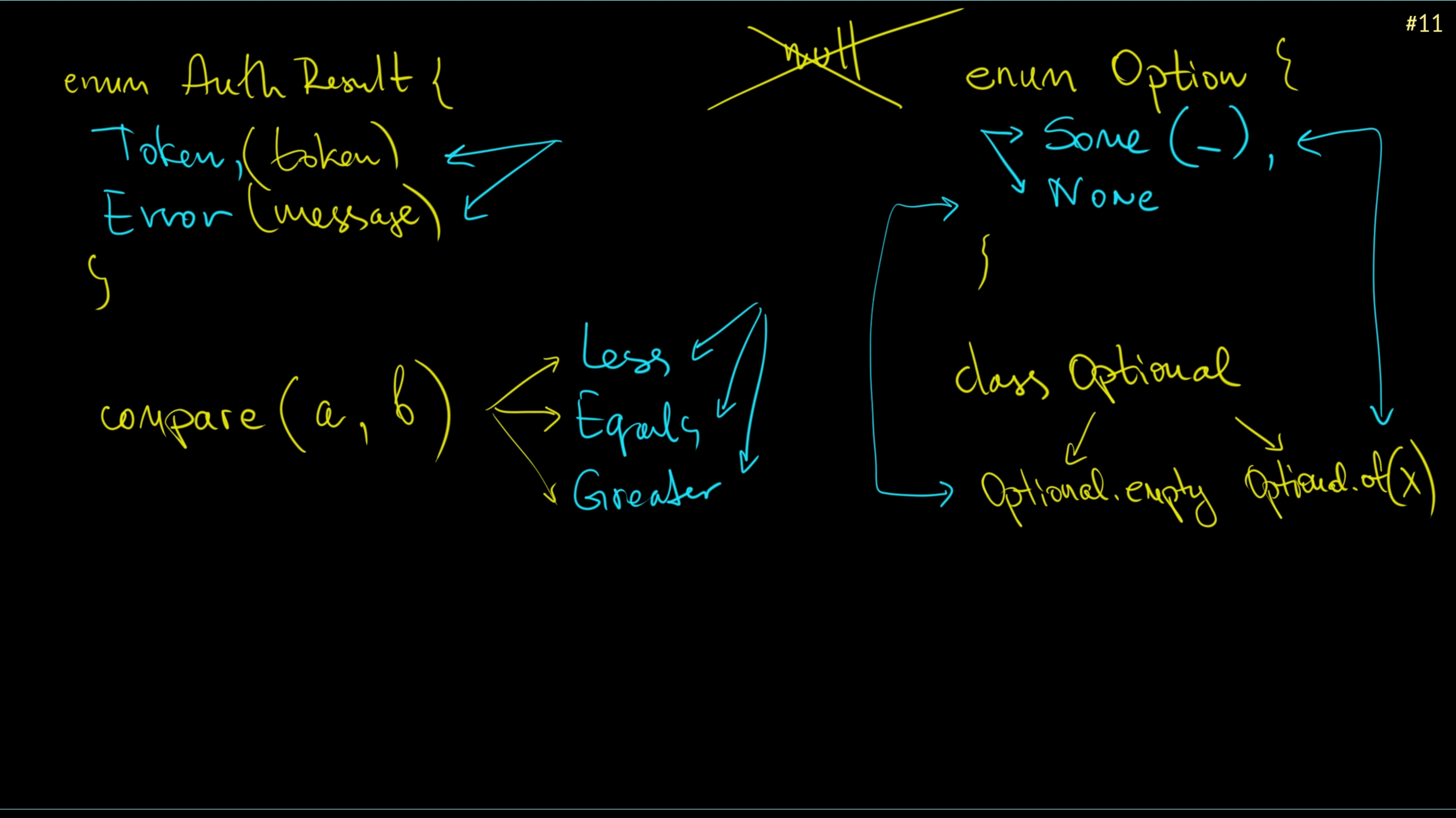
&

class Person ≡
name AND age AND skills AND ....

enum ≡ Red OR Green OR Yellow

CoProduct

Product


enum AuthResult {
Token(token)
Error(message)
}
null
enum Option {
Some(_),
None
}
compare(a, b)
Less
Equals
Greater
class Optional
Optional.empty
Optional.of(x)

readfile (f) -> String

Result<Error, String>

writeFile (f, String) -> void

Result<Error, void>

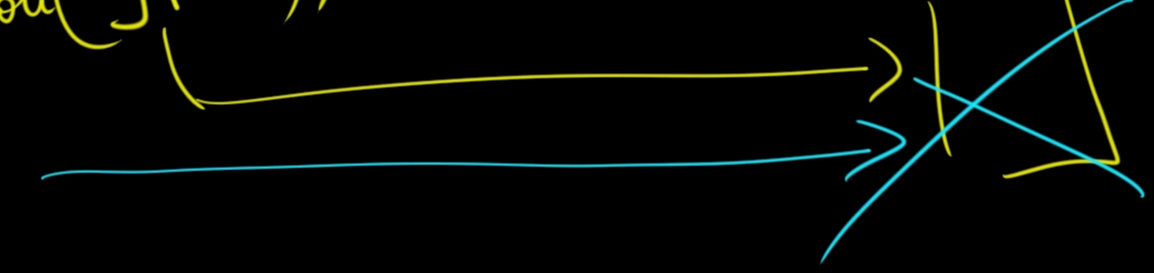
enum Result {

Ok(_)

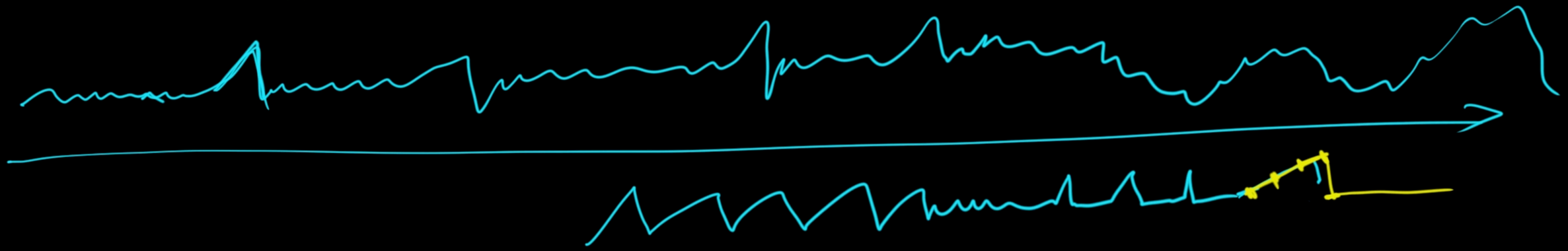
Error(_)

}

```
{
  p = new Person("John");
  free(p)
}
```

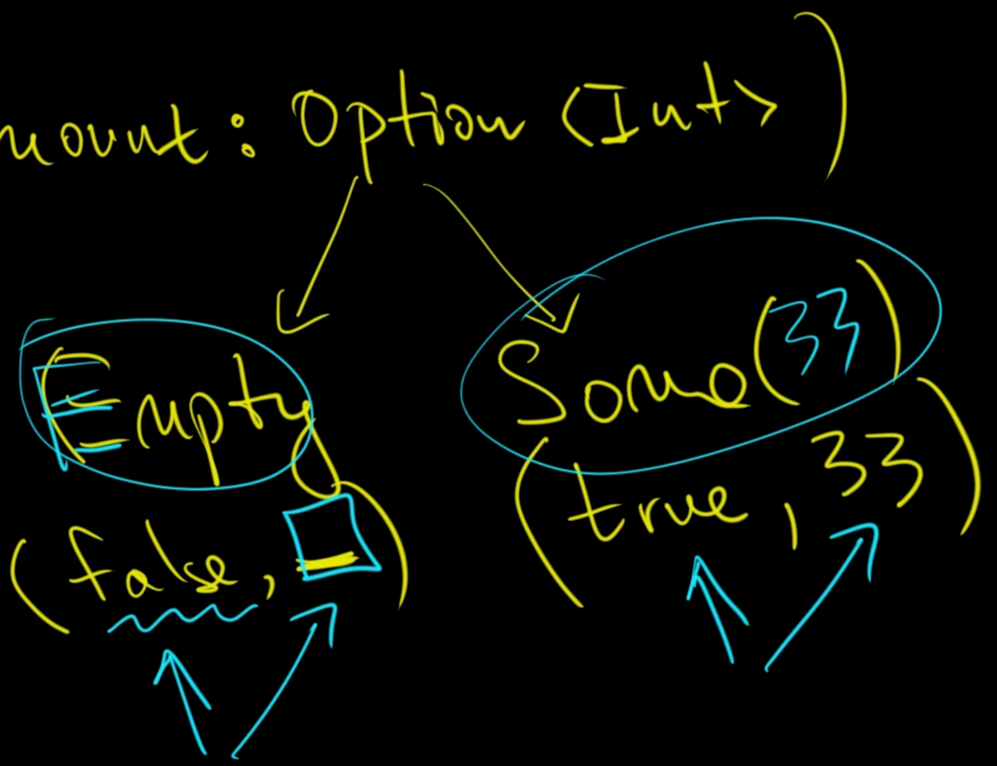


wride (x)



SQL amount INT NULLABLE

class Row (amount: Option<Int>)



~~int~~ max (int[])