Null Checks and Error Handling

Harit Himanshu
@harittweets

Overview

Why null checks are bad?

Option as a better alternative

Error handling in imperative languages

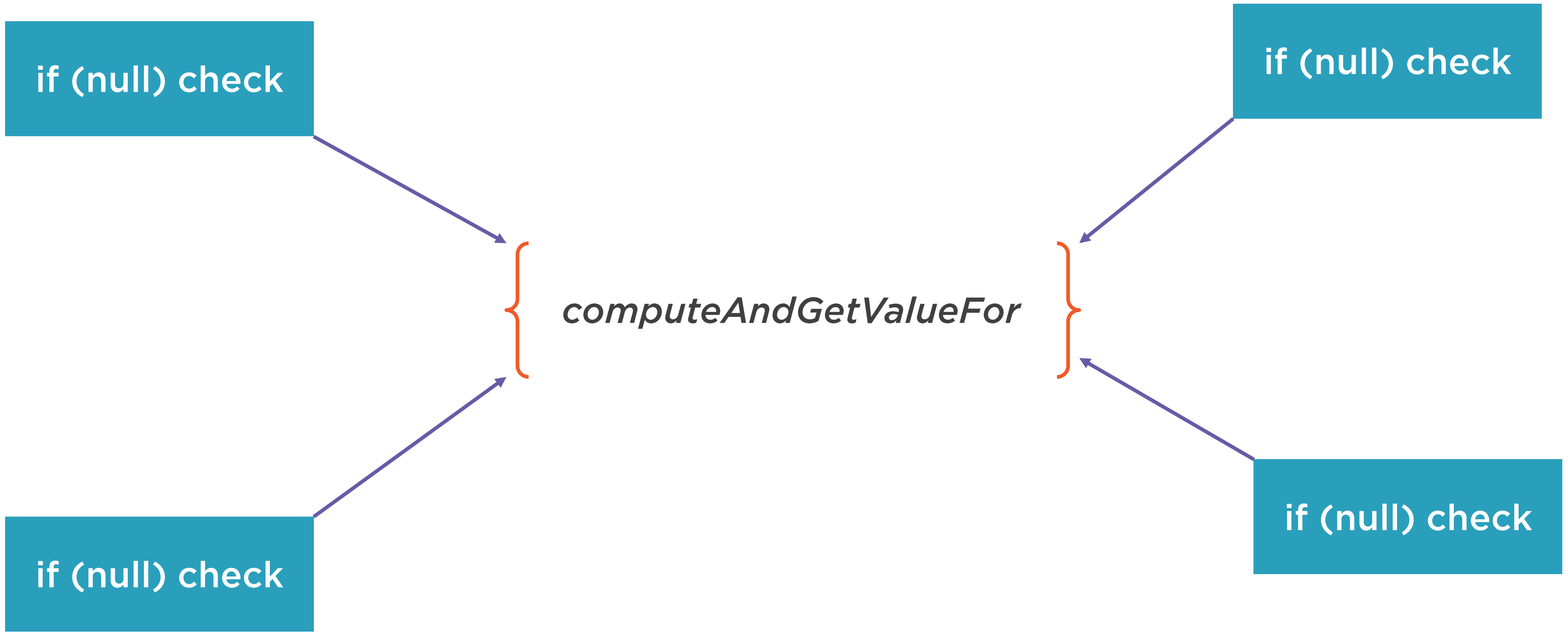
Error handling with Try

Error handling with Either

```
val valueFromComputation = computeAndGetValueFor(someInput)
if (valueFromComputation ! = null) {
 // do something with this value
}
```

Null Checks

Callers making null checks



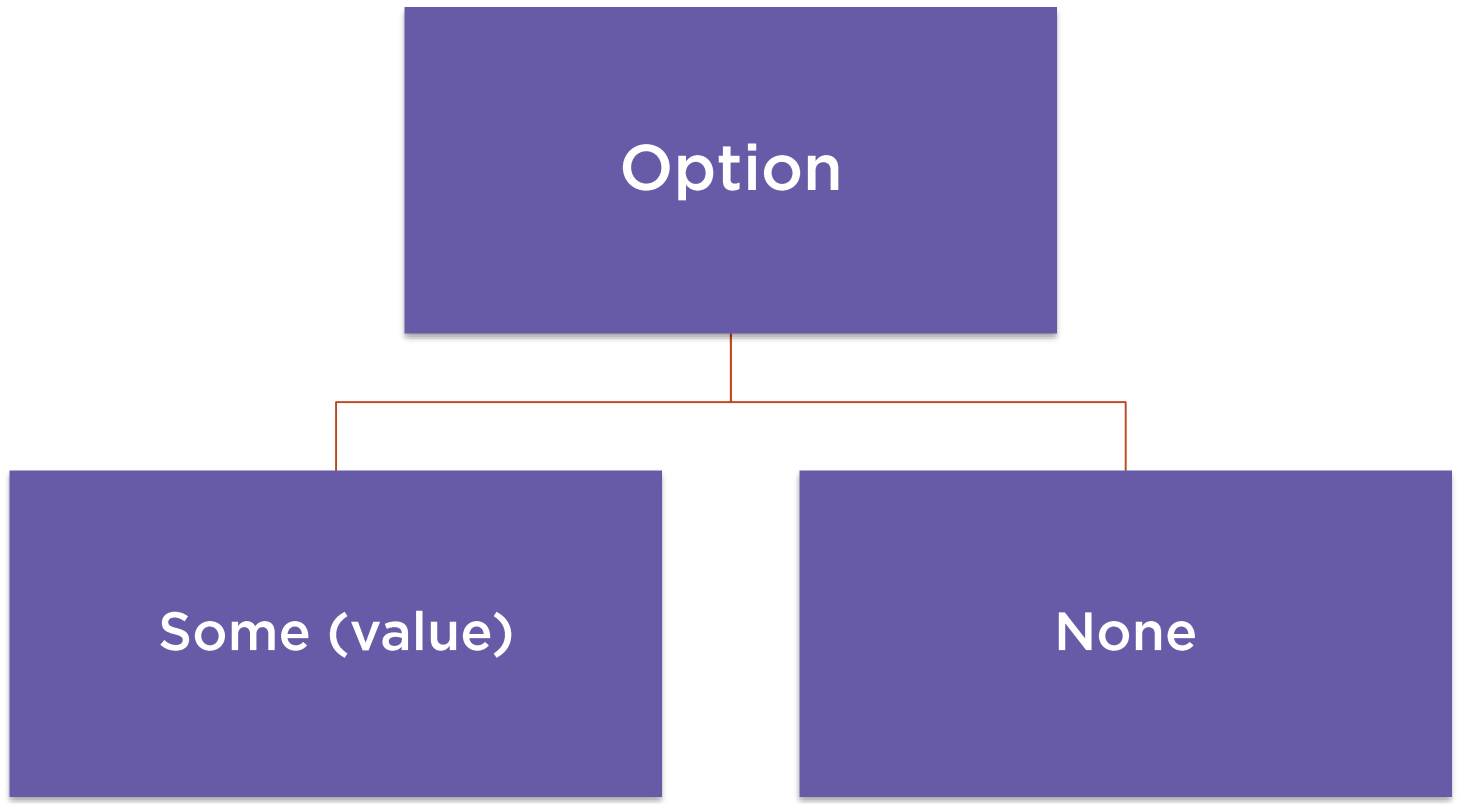
Interface

```
val employeeFromDatabase = getEmployeeByName(someName)
```

Find Employee by Name

There may or may not be such an employee

Option



Option example

Online delivery

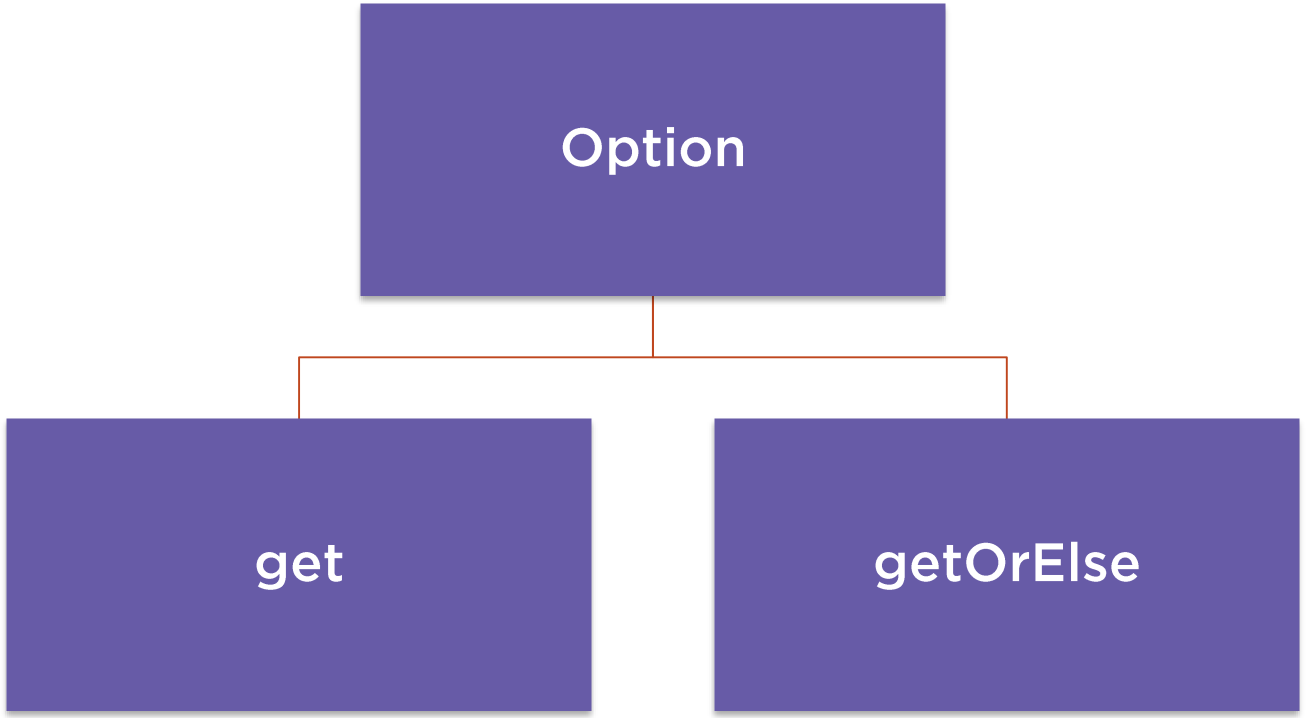
Lost lottery ticket

```
def find(p: (A) ⇒ Boolean): Option[A]
```

Set *find* method

Clear interface

Reading value from an *Option*

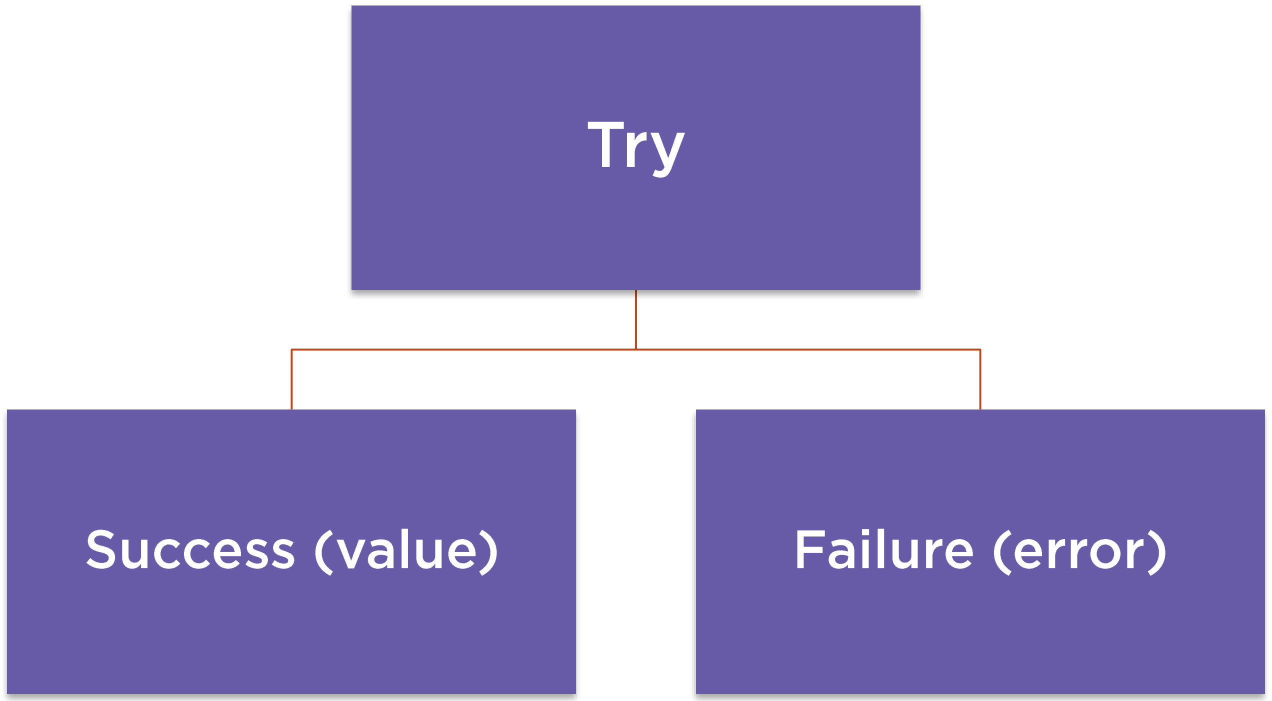


Errors and Exceptions

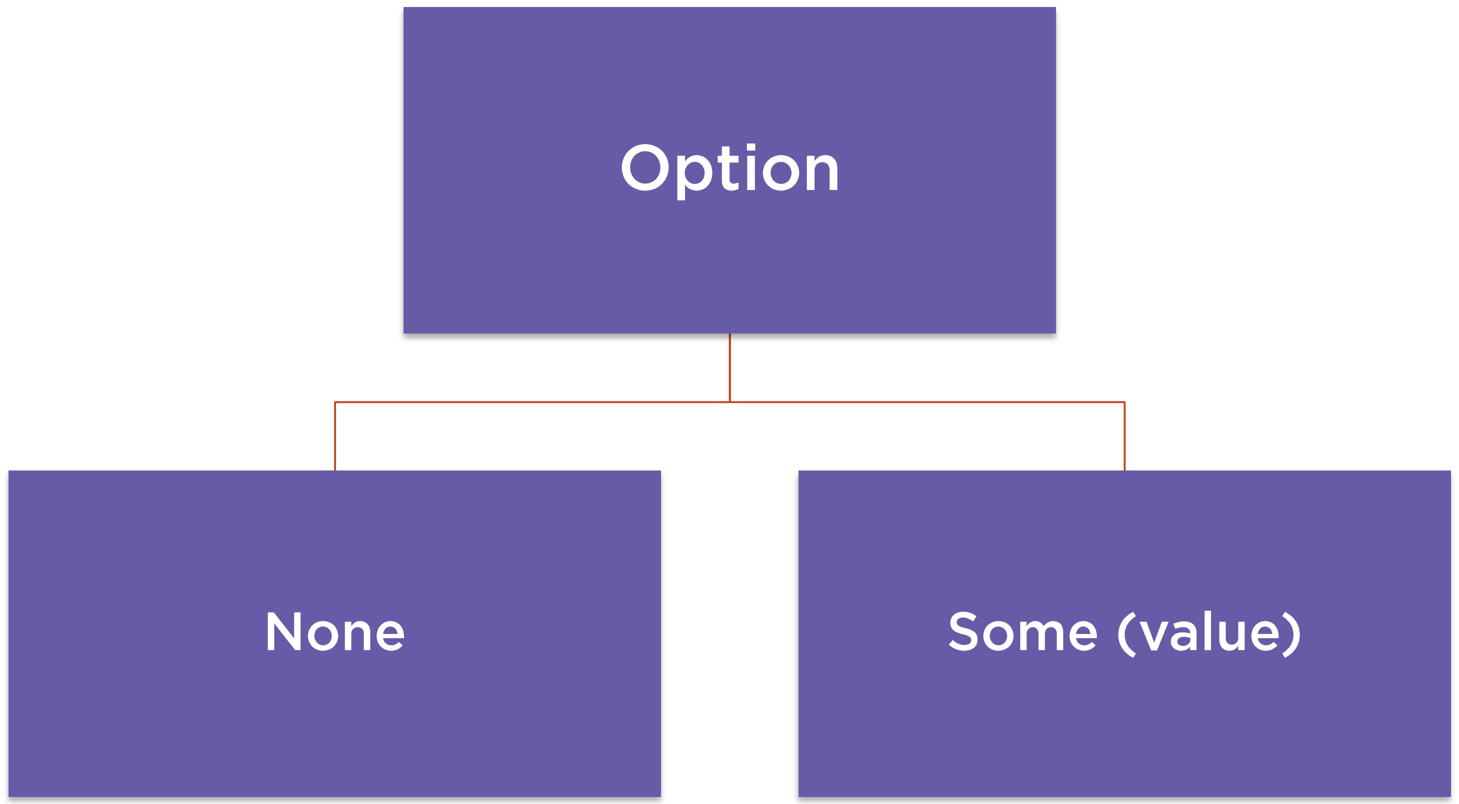
```
int value = 0;

try {

  value = numerator / denominator;

} catch (Exception e) {

  // do something with the exception

  // or re-throw the exception

} finally {

 // release resources that may leak memory

}
```

Divide number by zero throws exception

Scala Try

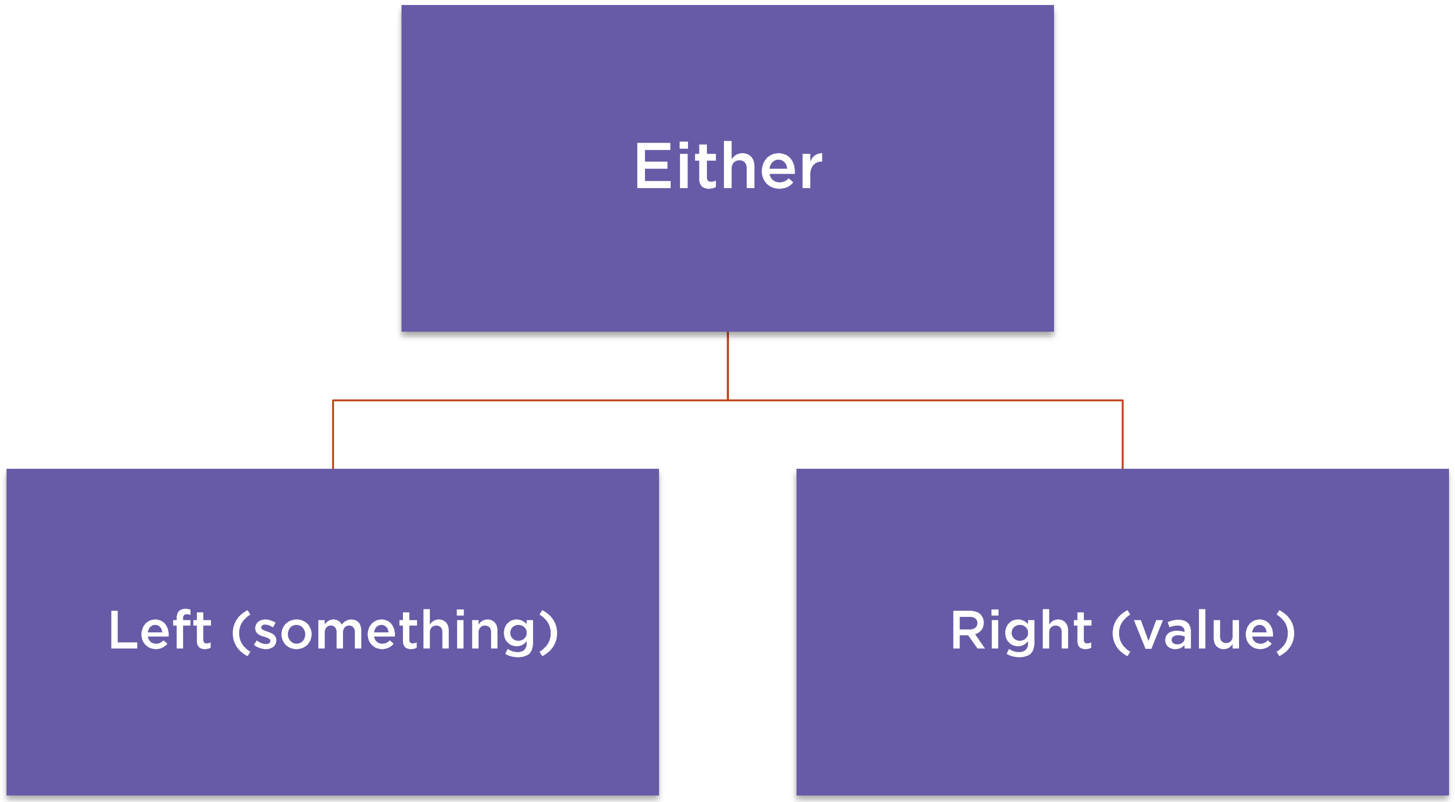


Option revisited



No way to return anything

Scala Either


Either
Left (something)
Right (value)