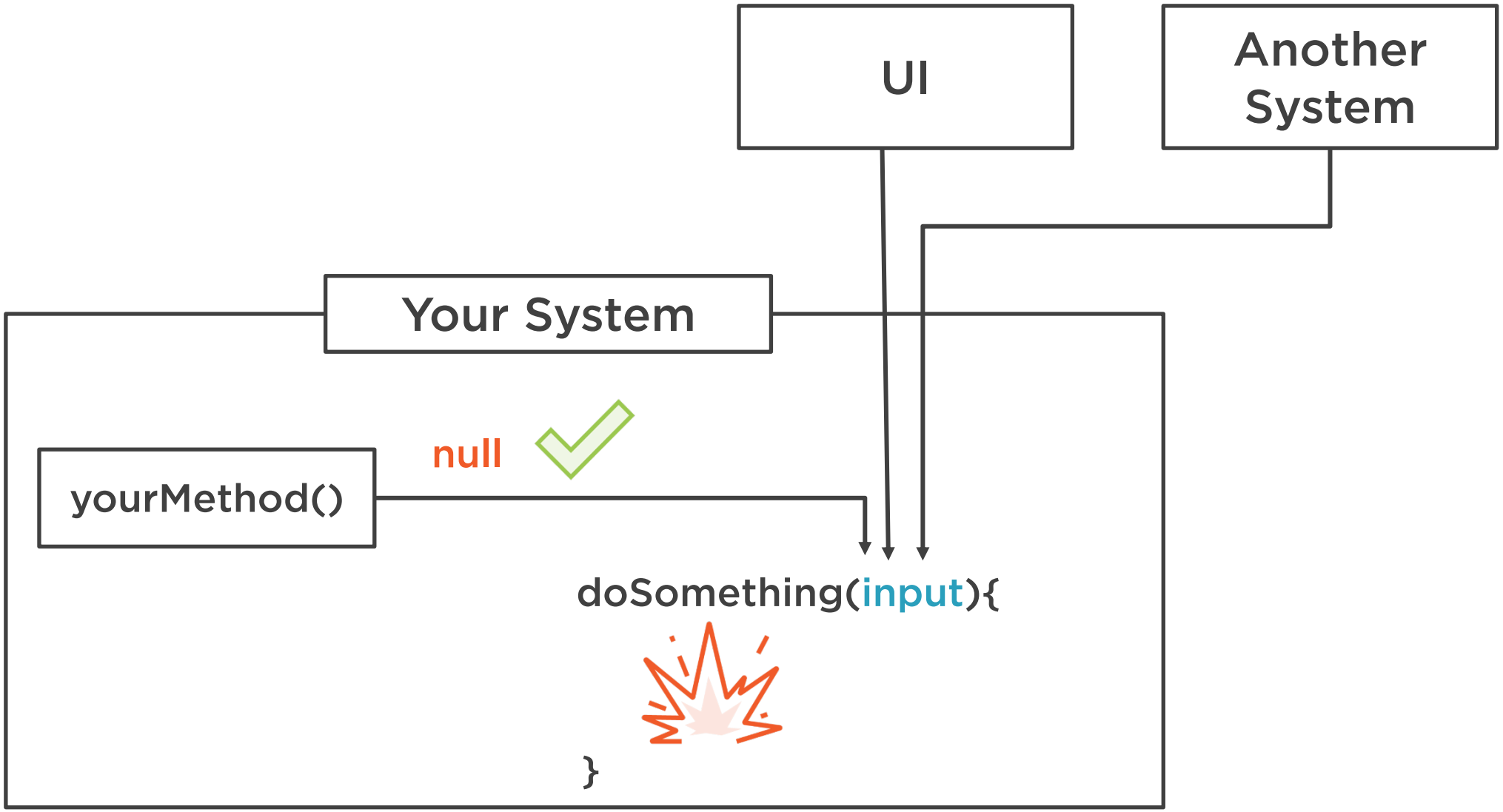
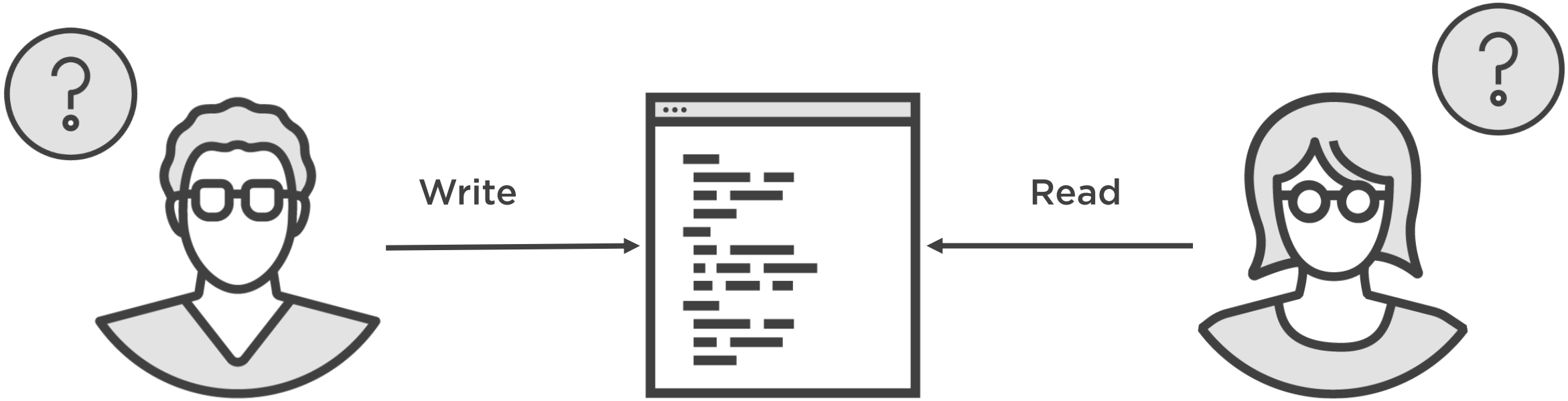
Improving Method Return Values

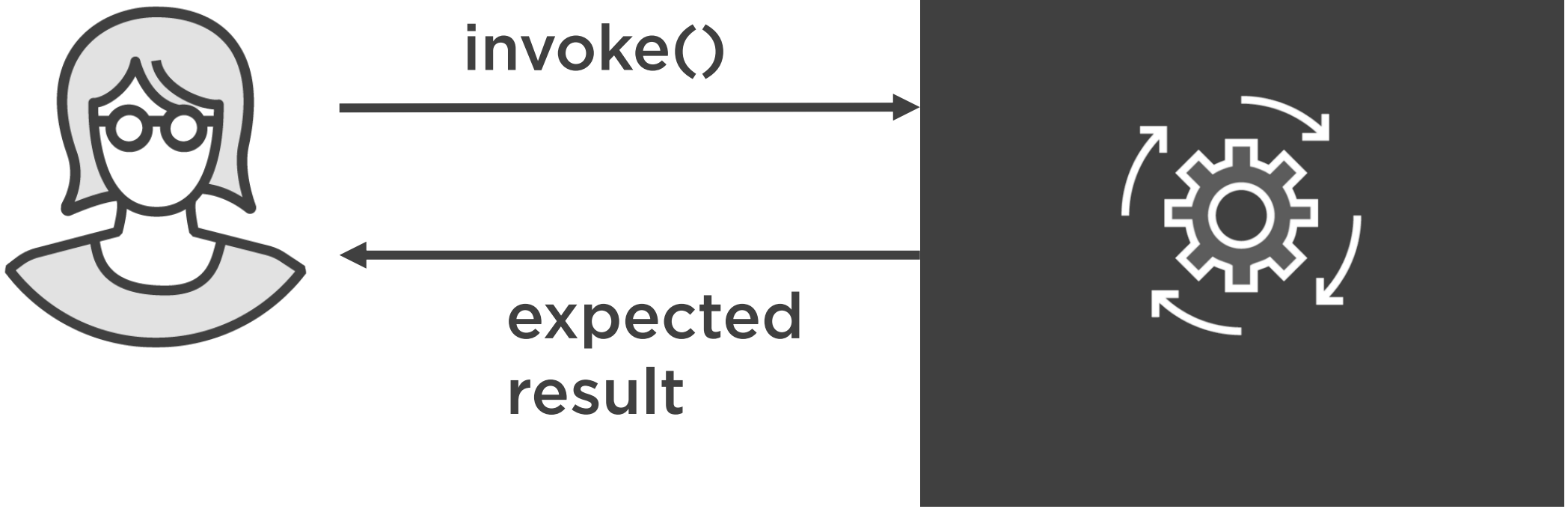
Andrejs Doronins
TEST AUTOMATION ENGINEER


UI
Another System
Your System
yourMethod()
null
doSomething(input){
}


Write
Read

The user of a component should never be surprised by its behavior

com.nosurprises.api



(Don’t return)
Magic numbers

Return
combinations

Alternatives to
null

Magic Numbers

-1 or 0

Java

```
System.exit(0);
```

void

boolean

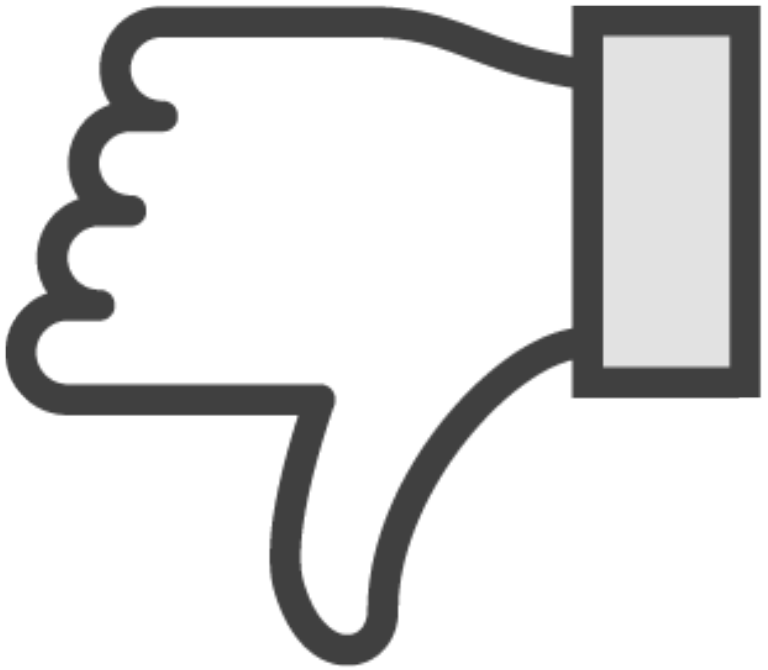
int

write(String txt)

```
int write(String txt)
```

```
int result = write(text);
// write failed
if(result == -1){
} else if (result == 0){


}
```

Magic Numbers



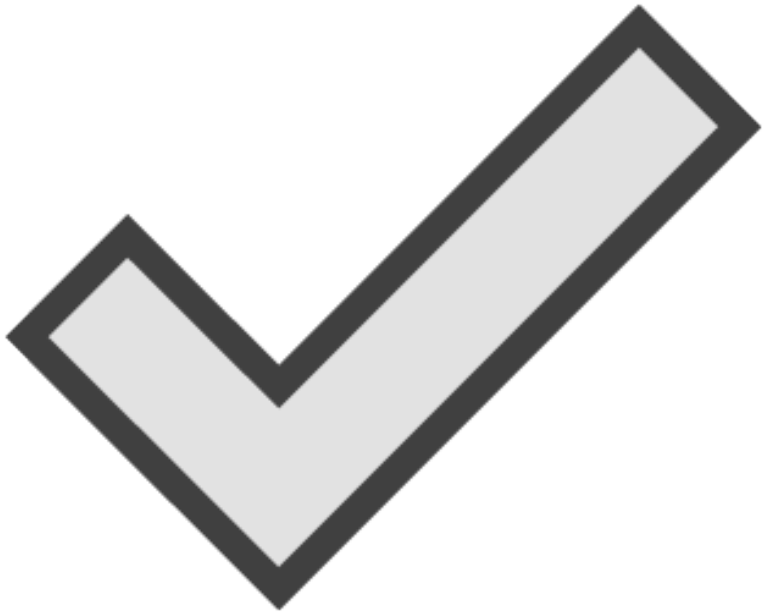
Force programmers to learn their meaning

Results in very poor client code

```
double calculate(a,b);

LocalDate parse(stringDate);

Person newPerson("John");

boolean isValid(whatever);
```

```
? doThisButItMightFail(input);
```

Valid Return Values



1) true/false

- True: success
- False: failure

2) void/throw

- Nothing happens: success
- Exception: failure

Throw in exceptional circumstances

(situations when the function cannot fulfill its purpose)

Bad Return Values

3) Don’t mix!

true/false (and maybe an exception)

4) Null

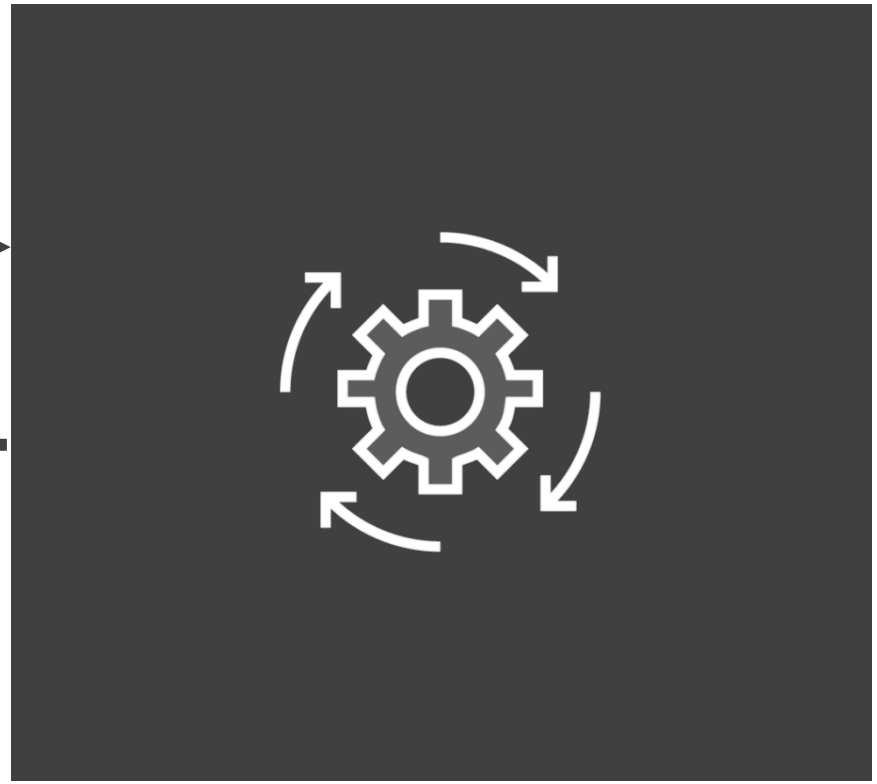
Should get true or false...

What the...

isValid(...)

SurpriseException!

com.badsurprises.api



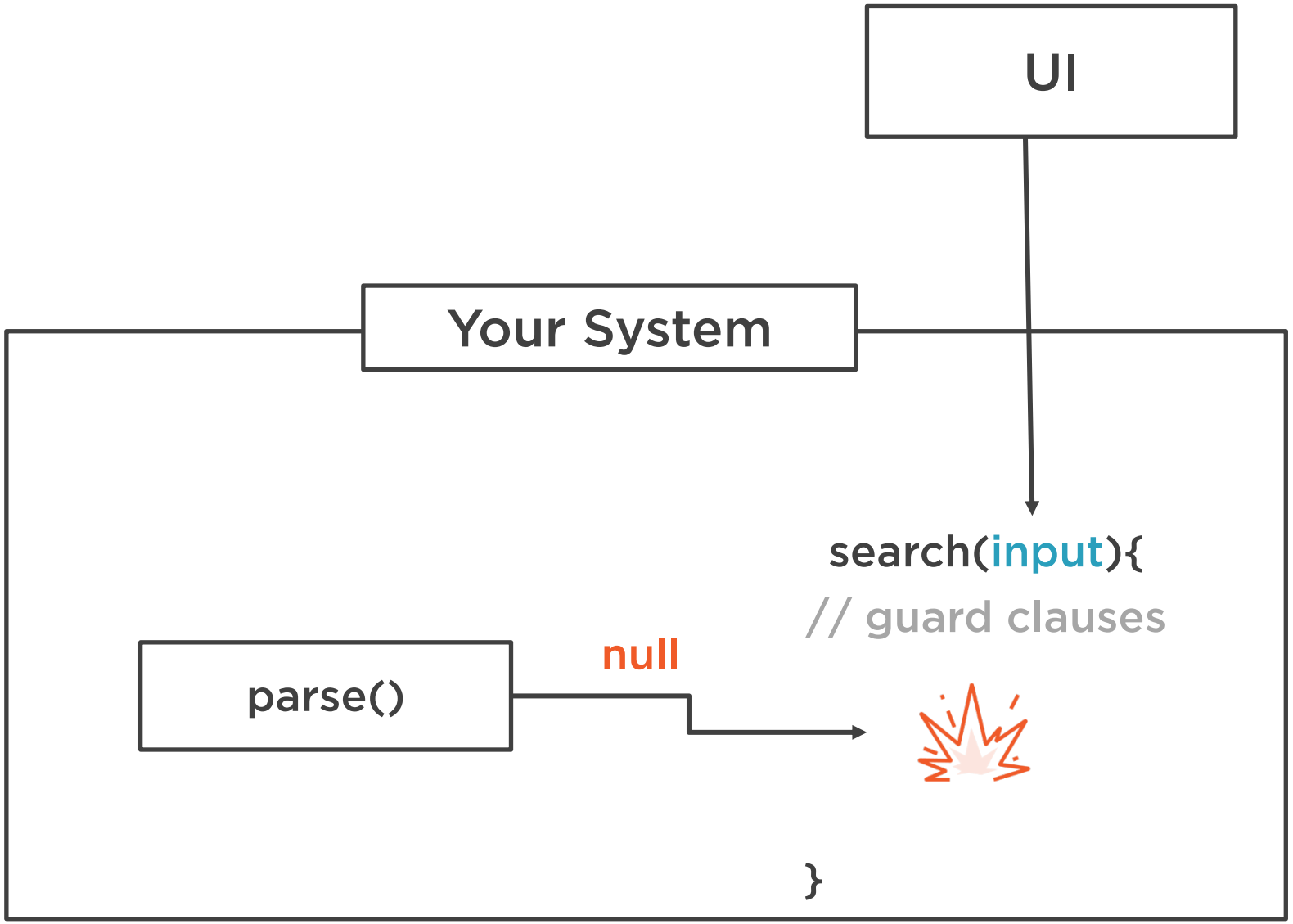
Usage of Null

Value for uninitialized fields

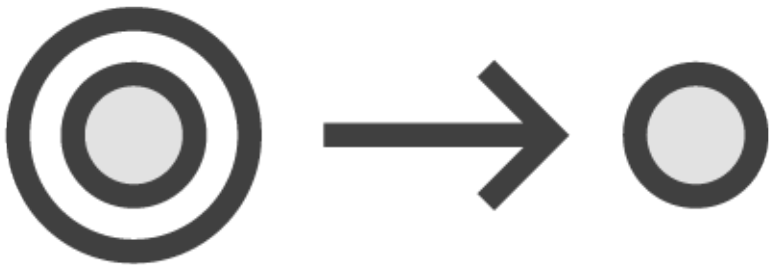
(necessary)

To indicate a missing value or result

(not necessary)


UI
Your System
search(input){
// guard clauses
null
parse()
}

Alternatives to Null

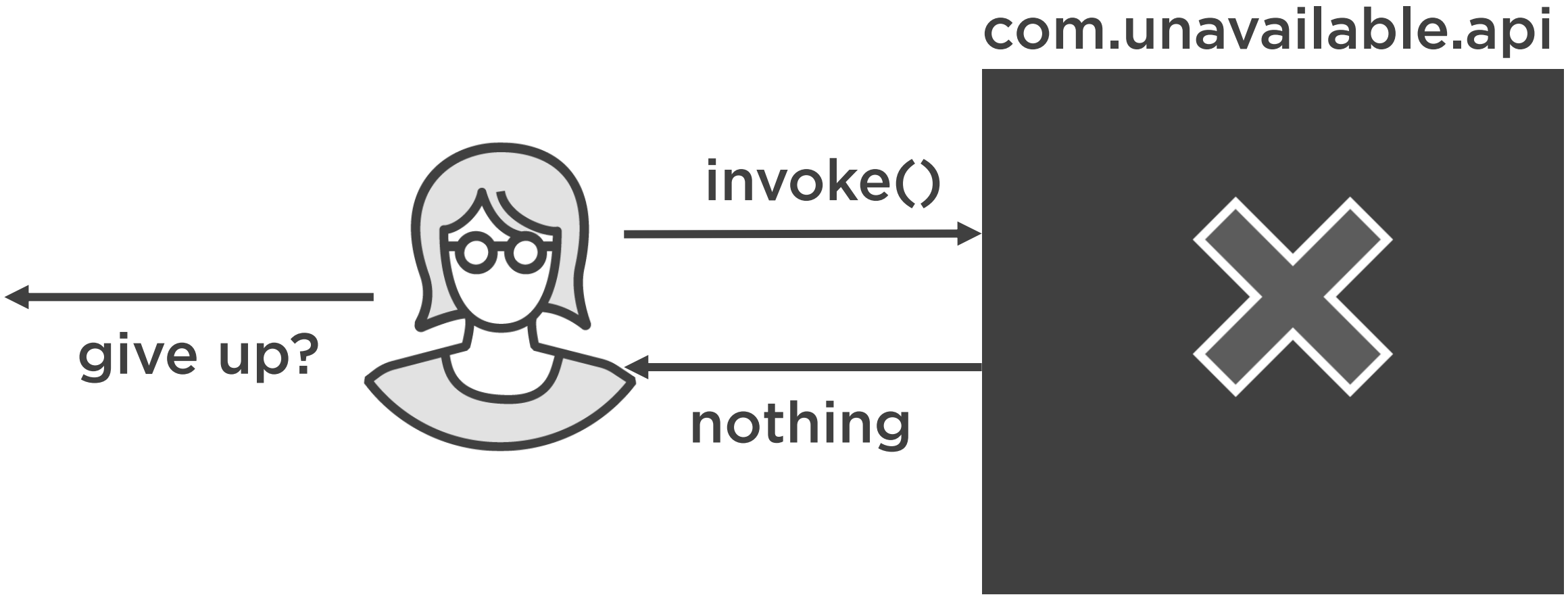


1) throw

2) Sensible default

3) Empty collection

4) Optional<T>


com.unavailable.api
invoke()
give up?
nothing

Absence of data is not necessarily an error

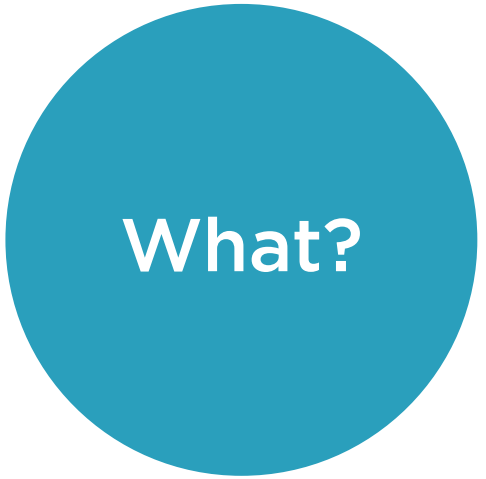
(and we should recover)

Collections

emptyList()

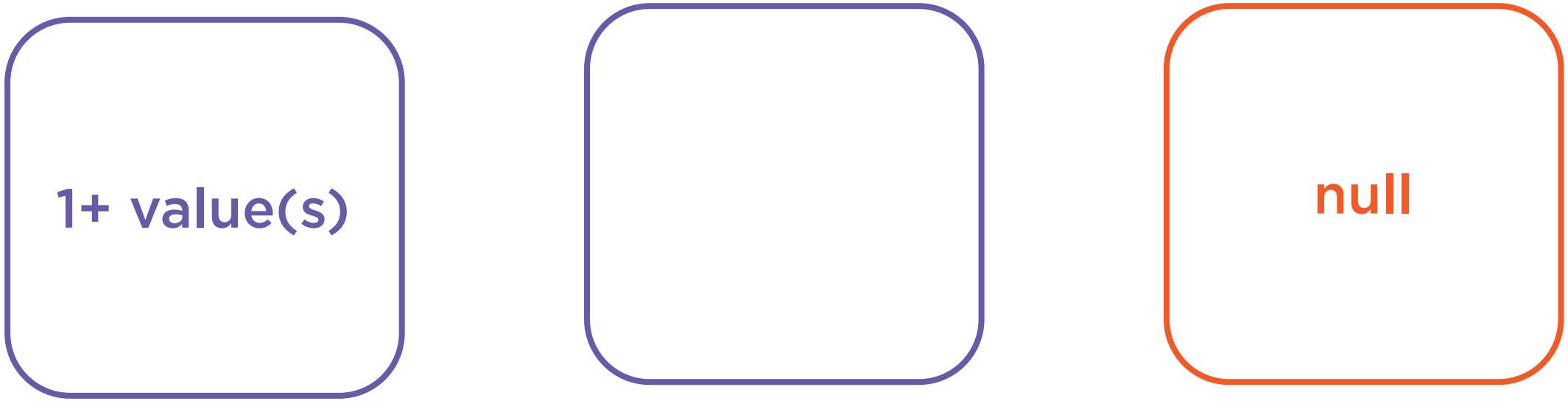
emptySet()

emptyMap()



Optional

Collection

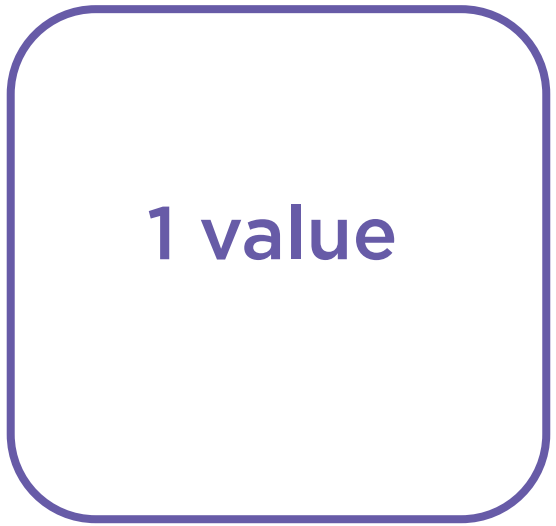


OK

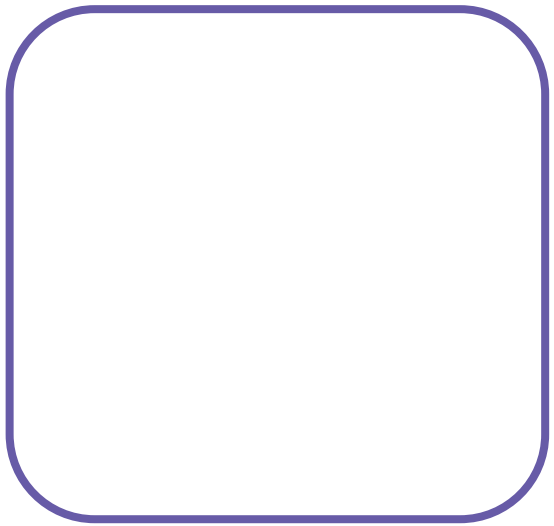
OK

Not OK

Optional<T>



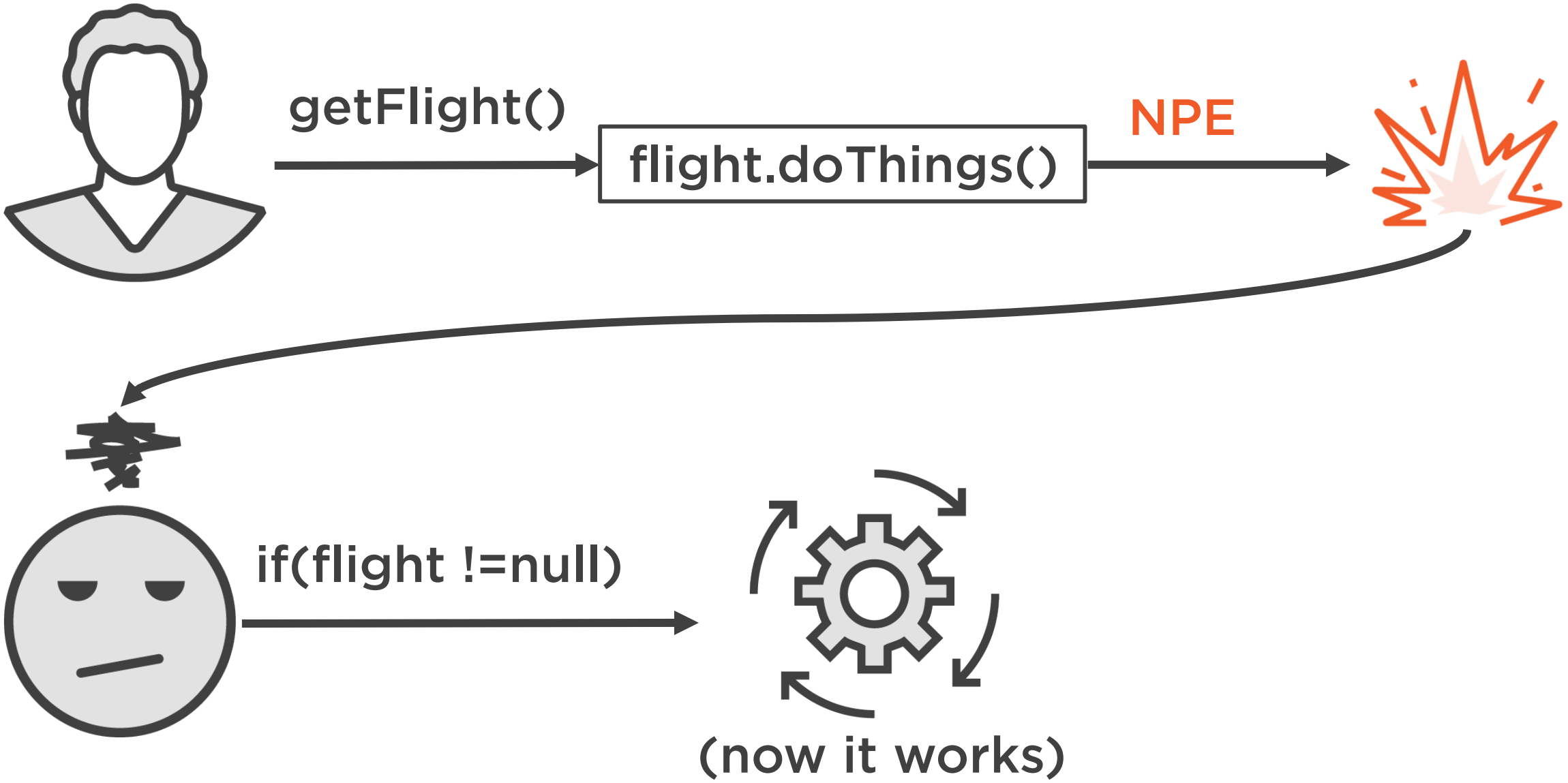
OK

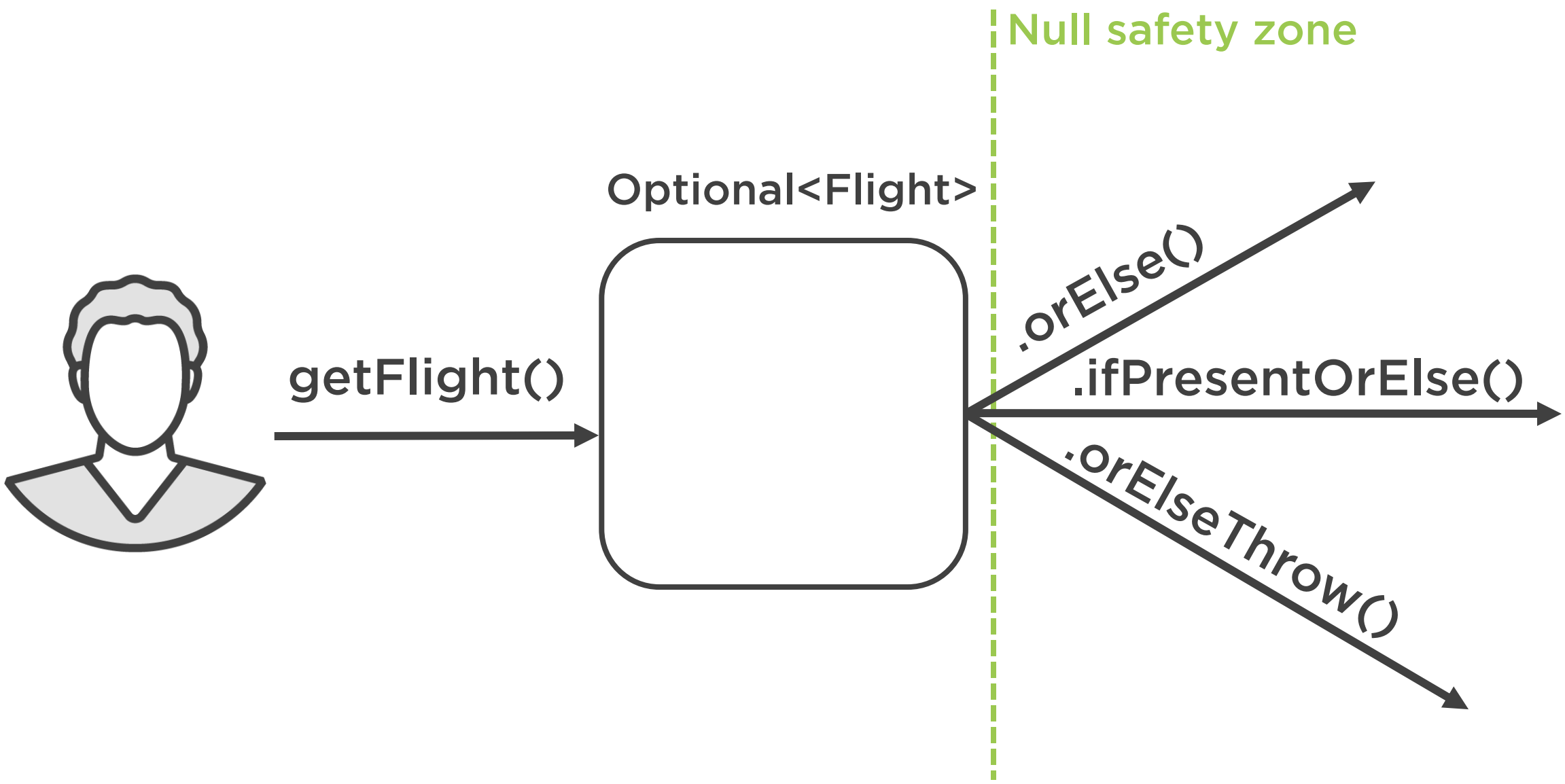


OK

Optional

A container object which may or may not contain a non-null value


getFlight()
flight.doThings()
NPE
if(flight !=null)
(now it works)


Null safety zone
Optional<Flight>
getFlight()
.orElse()
.ifPresentOrElse()
.orElseThrow()

Optional<T> forces the user to confront the fact that there may be no value

~~Flight~~

Optional<Flight> getFlight(String reference)

Optional<T>

.or()

.orElse()

.orElseGet()

.orElseThrow()

.ifPresentOrElse()

Summary

Don’t return magic numbers

Don’t return null

Do return:

- Default values
- Empty collection
- Optional

React

Fail early with a guard clause in methods

Prevent

Fail early with a guard clauses in constructors

Return only expected values

Don’t return nulls