Using the Null Object Pattern

Esteban Herrera
JAVA ARCHITECT

@eh3rrera www.eherrera.net

Using Null to Represent the Absence of Data

```
ReadingLevel readingLevel = book.getReadingLevel();

if (readingLevel != null) {
    // Use reading level object
} else {
    // Supply default behavior
}
```

Using Null to Represent the Absence of Data

```java
ReadingLevel readingLevel = book.getReadingLevel();

if (readingLevel != null) {
    // Use reading level object
}
```

Wouldn’t It Be Great?

```java
ReadingLevel readingLevel = book.getReadingLevel();

// Use reading level object either way
System.out.println(readingLevel.getGrade());
```

Null Object pattern

Instead of using a null reference to represent the absence of an object, it uses an object that implements the expected interface but does nothing, hiding the details from its collaborators.

What “do nothing” means is subjective.

About the Null Object Pattern

Not a GoF pattern

- Bobby Woolf
  - 1996 article “The Null Object Pattern”
  - Pattern Languages of Program Design Vol. 3

Also known as

- Active Nothing
- Stub

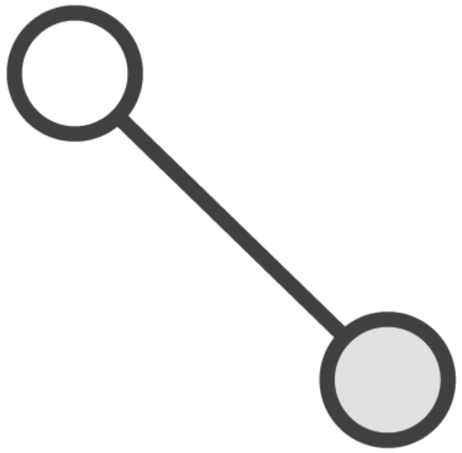
```
s = "";

l = Collections.emptyList();

n = 0;
```

◀ For strings

◀ For collections

◀ For numbers

Relation to Other Patterns



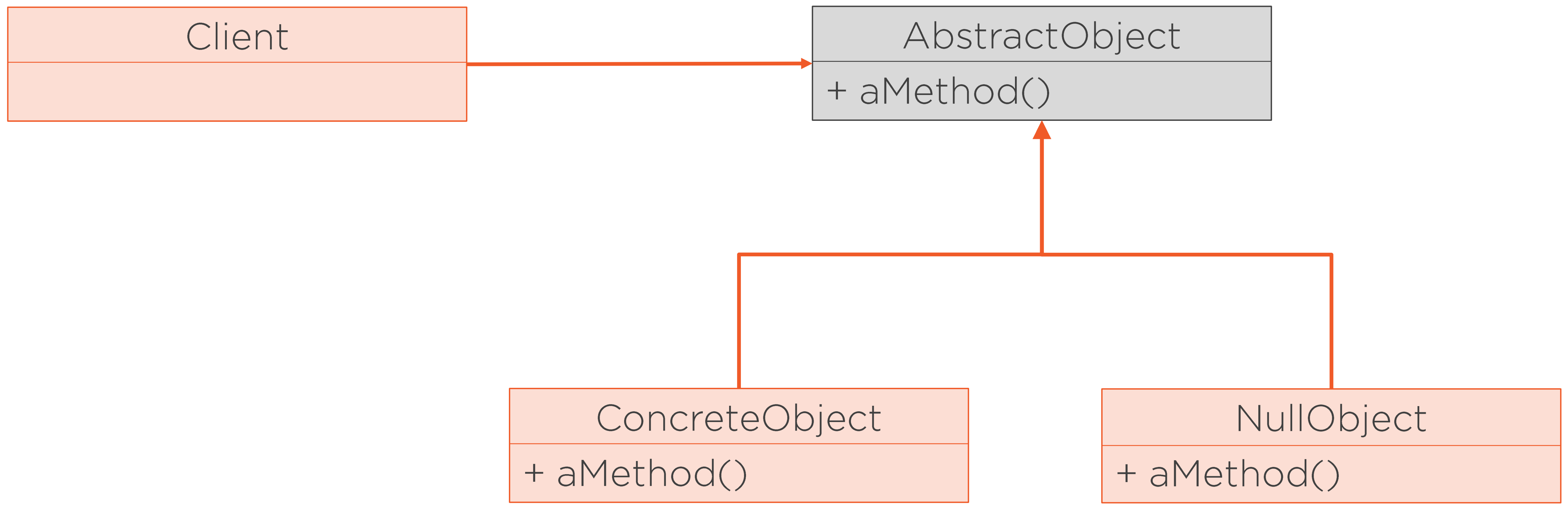
A Null Object doesn't work like a proxy

It can be the State pattern if it can transform into something else

A Null Object can be seen as a special case of the Strategy pattern

Implementing the Null Object Pattern

The Null Object Pattern



Demo

Implementing the Null Object pattern

Criticisms of the Null Object Pattern

An incorrect implementation can make bugs harder to detect

- Null objects can fail slowly
- Do not to implement it just to avoid null checks
- Best suited when a default value can be assigned or a default action can be taken

Creating a proper Null Object may not be easy

- Should it do nothing? Or should it fail with an exception?
- What if you still have to check for the null object?
- What if the parent class is final?

Summary

The Null Object pattern replaces nulls with objects that implement

- A default behavior
- A do-nothing behavior

This allows you to avoid null checks

To implement it:

- Abstract class that defines the behavior for all objects of this type
- The Null Object is a subclass of the abstract class

Summary

Disadvantages

- An incorrect implementation can make bugs harder to detect
- Creating a proper Null Object may not be easy

In the next module:
Using Optional instead of null