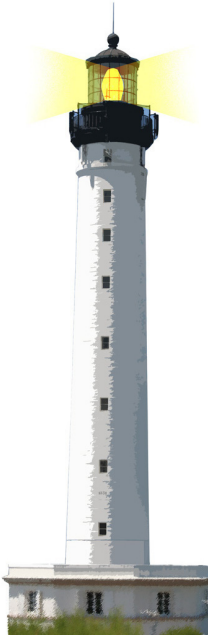
Learning Object-Oriented Programming and Design with TDD

Avoid Null Checks

Stéphane Ducasse

http://stephane.ducasse.free.fr



Pharo

http://www.pharo.org

W7S06

What You Will Learn

- AntiIf Campaign
- If you do not want to be forced to test nil, do not generate nil
- NullObject Design Pattern

Anti If Campaign

```
Main >> showHappiness: animal
  animal isDog
    ifTrue: [ animal shakeTail ].
  animal isDuck
    ifTrue: [ animal quack ].
  animal isCat ifTrue: [ ... ].
```

Branching (with `if`) based on the type of an object is bad:

- adding a new type requires modifying all such code
- methods will become very long and full of details

Instead, send messages!

Anti-IF Campaign

Anti If Campaign

```
Dog >> showHappiness
  self shakeTail
```

```
Duck >> showHappiness
  self quack
```

```
Cat >> showHappiness
  ...
```

nil or anObject?

When you get a variable that can be nil or anObject

- Forced to check before doing anything
- Every access should be controlled

How to Avoid nil?

- Initialize well your objects (see Lectures Instance Initialization)
- When possible, do not return nil!
- When possible, apply NullObject Design Pattern

Clients do not have to check anymore

```
ToolPalette >> initialize
    self selectedTool: NoTool new
```

```
ToolPalette >> nextAction
    self selectedTool attachHandles
```

```
ToolPalette >> previousAction
    self selectedTool detachHandles
```

Conclusion

- A message acts as a better `if`
- Avoid null checks, return polymorphic objects instead
- Initialize your variables
- If you can, create objects representing default behavior

Anti-IF Campaign

Resources

- Pharo mooc - Videos W7S07: `http://mooc.pharo.org`

A course by Stéphane Ducasse
http://stephane.ducasse.free.fr

Reusing some parts of the Pharo Mooc by

Damien Cassou, Stéphane Ducasse, Luc Fabresse
http://mooc.pharo.org

Except where otherwise noted, this work is licensed under CC BY-NC-ND 3.0 France
https://creativecommons.org/licenses/by-nc-nd/3.0/fr/