



Learning Object-Oriented Programming and Design with TDD

Essence of Dispatch

Taking Pharo Booleans as Example

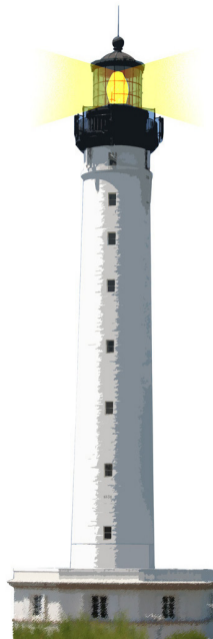
Stéphane Ducasse

<http://stephane.ducasse.free.fr>



<http://www.pharo.org>

W4S01



Objectives

- Understanding of message passing (late binding)
- The heart of OOP
- Insight at how beautiful Pharo's implementation is



Context: Booleans

In Pharo, Booleans have a superb implementation!

- `&`, `|`, `not` (**eager**)
- `or:`, `and:` (**lazy**)
- `ifTrue:ifFalse:`, `ifFalse:ifTrue:`



Three Exercises

1. Implement not (Not)
2. Implement | (Or)
3. What is the goal of these exercises?



Exercise 1: Implement Not

Propose an implementation of Not in a world where:

- You have: true, false
- You only have objects and messages
- How would you implement the message not?

```
false not  
>>> true
```

```
true not  
>>> false
```

Hint 1: No conditionals

The solution does not use conditionals (i.e., no if)

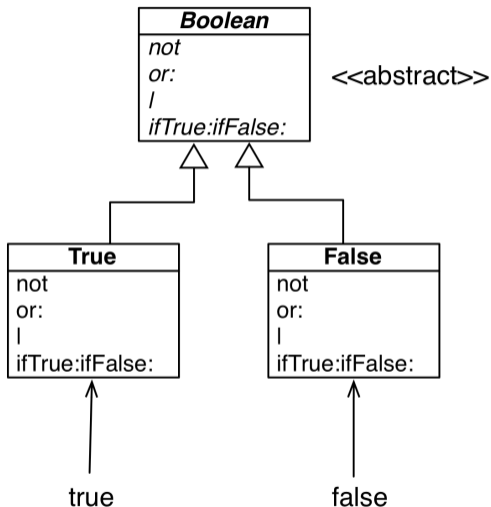


Hint 2: With Two (Three) Classes

- The solution uses three classes:
 - True and False (and Boolean abstract superclass but not mandatory)
- true is the singleton instance of True
- false is the singleton instance of False



Hint 2: Two (Three) Classes



Hint 3: How do We Express Choice in OOP?

In OOP, choice is expressed

- By defining classes with compatible methods
- By sending a message to an instance of such class

Example

```
x.open
```

- `x` can be a file, a window, a tool,...
- The method is **selected** based on `x`'s class

Implementation of Not in Two Methods

False >> not

"Negation -- answer true since the receiver is false."

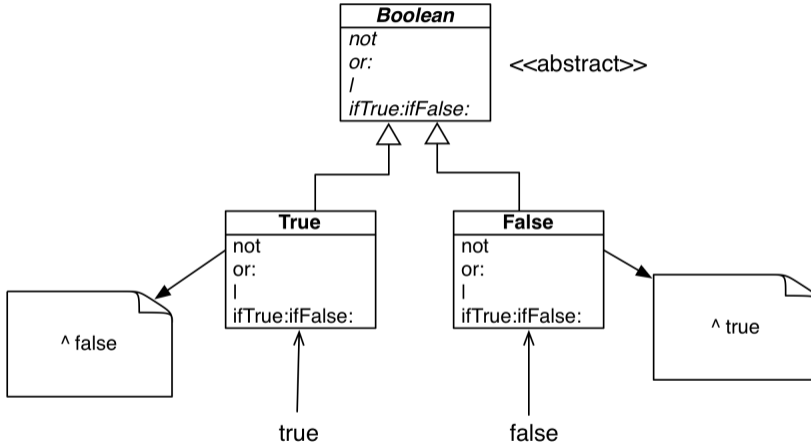
^ true

True >> not

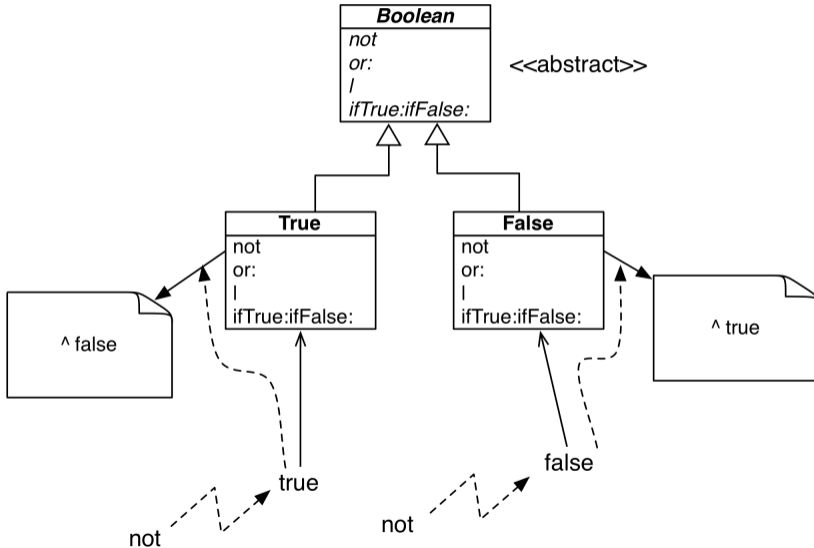
"Negation -- answer false since the receiver is true."

^ false

Implementation Hierarchy



Message Lookup is Choosing the Right Method



Boolean Class

- Boolean is abstract
 - does not create instances
 - specify abstract methods that subclasses should implement
 - specify concrete method that subclass instances may use

`Boolean>>not`

"Abstract method. Negation: Answer true if the receiver is false, answer false if the receiver is true."

`self subclassResponsibility`

- Subclasses are `True` and `False` and implement
 - logical operations `&`, `not`
 - control structures `and:`, `or:`, `ifTrue:`, `ifFalse:`, `ifTrue:ifFalse:`, `ifFalse:ifTrue:`



Behavior of Or

```
true | true -> true  
true | false -> true  
true | anything -> true
```

```
false | true -> true  
false | false -> false  
false | anything -> anything
```

Implementation of Or in Boolean

`Boolean >> | aBoolean`

"Abstract method. Evaluating Or: Evaluate the argument.
Answer true if either the receiver or the argument is true."

`self subclassResponsibility`

Implementation of Or in Class False

```
false | true -> true  
false | false -> false  
false | anything -> anything
```

```
False >> | aBoolean
```

```
"Evaluating Or -- answer with the argument, aBoolean."
```

```
^ aBoolean
```


Implementation of Or in Class True

```
true | true -> true  
true | false -> true  
true | anything -> true
```

```
True >> | aBoolean  
"Evaluating Or -- answer true since the receiver is true."  
^ true
```

Real Implementation of Or in Class True

The object `true` is the receiver of the message!

```
True>> | aBoolean
```

```
"Evaluating disjunction (Or) -- answer true since the receiver is true."
```

```
^ true
```

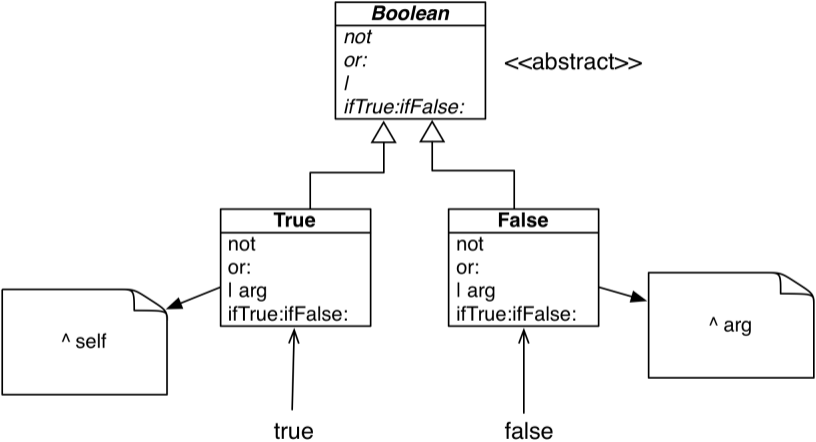
So we can write it like the following:

```
True >> | aBoolean
```

```
"Evaluating disjunction (Or) -- answer true since the receiver is true."
```

```
^ self
```

Or Implementation in Two Methods



Summary

- The solution to implement booleans' operations:
 - does NOT use conditionals (if)
 - lets the receiver decide
- Do not ask, tell



Resources

- Pharo Mooc - W3S01 Videos
- Pharo by Example <http://books.pharo.org>
- Learning Object-Oriented Programming, Design and TDD with Pharo

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

Reusing some parts of the Pharo Mocc by

Damien Cassou, Stéphane Ducasse, Luc Fabresse
<http://mocc.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/>