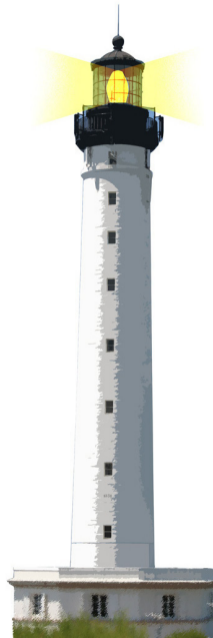


Essence of Dispatch

Taking Pharo Booleans as Example

S. Ducasse



Objectives

- Understanding of message passing (late binding) for **real** this time
- The **heart of OOP**
- Looking at a beautiful implementation in Pharo



Context: Booleans

In Pharo, Booleans have a superb implementation!
You get the classical messages

- &, |, not (eager)
- or:, and: (lazy)

And some less traditional ones

- ifTrue:ifFalse:, ifFalse:ifTrue:
 - Yes, conditionals are messages sent to boolean objects



Three exercises

- Exo 1: Implement not (Not)
- Exo 2: Implement | (Or)
- Exo 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: How do we express choices in OOP?

In OOP, the choice is expressed

- By defining classes with **compatible** methods
- By **sending** a message to an instance of such class
- Let the receiver decide

Example

x open

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



Hint 3: With at least two classes

- true is the singleton instance of True
- false is the singleton instance of False

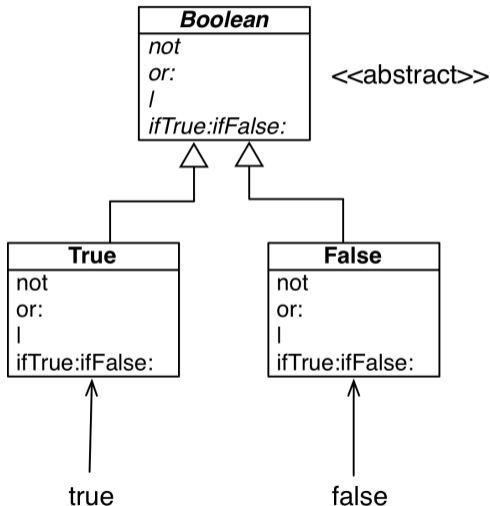
The Pharo implementation uses three classes:

- Boolean (**abstract**), True, and False



Hint 3: With at least two classes

Boolean is not needed per se but it improves reuse



Implementation of Not in two methods

False >> not

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

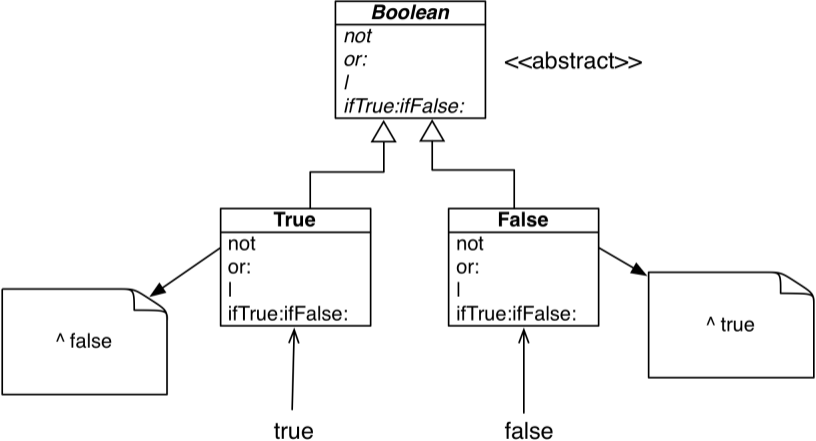
^ true

True >> not

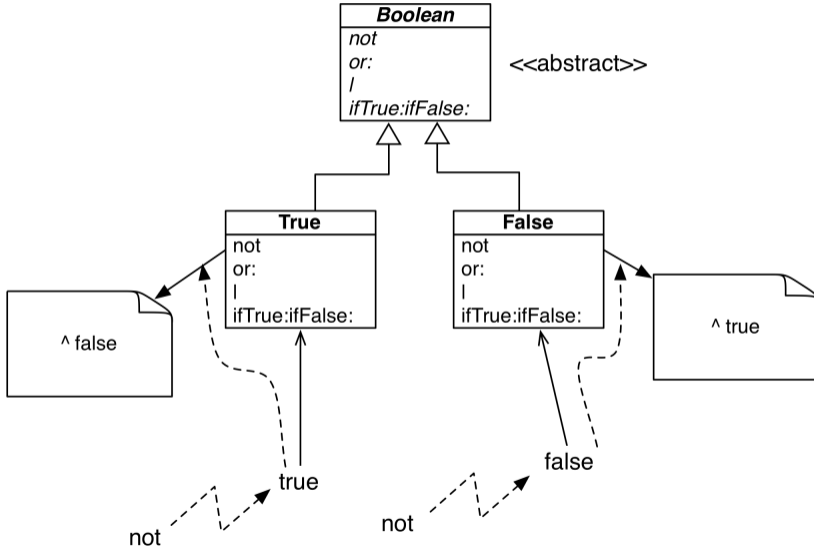
"Negation -- answer false since the receiver is true."

^ false

Implementation hierarchy



Message lookup is choosing the right method



Boolean implementation

- Boolean is abstract
- True and False implement
 - logical operations &, not
 - control structures and:, or:, ifTrue:, ifFalse:, ifTrue:ifFalse:, ifFalse:ifTrue:
 - reuse some logic from Boolean



Exercise 2: Implement 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
```


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
```

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:

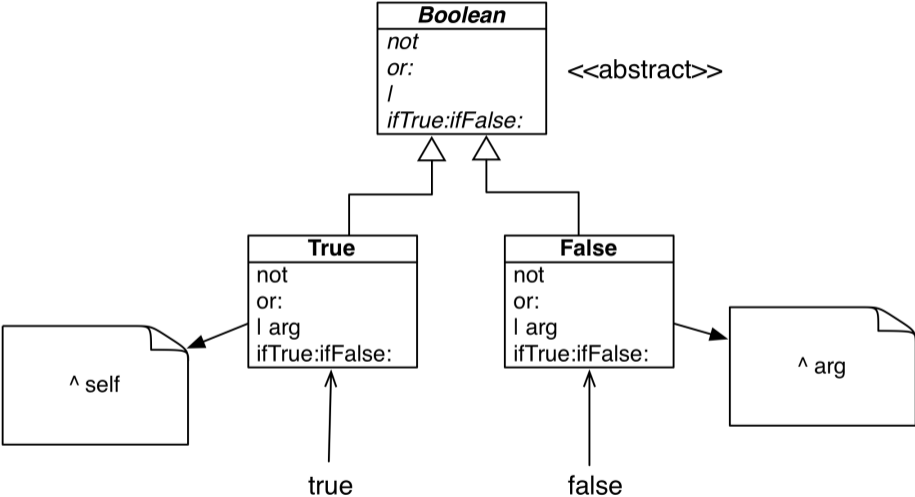
```
True >> | aBoolean
```

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

```
^ self
```



Or Implementation in two methods



Step back

Do not ask, tell

- we delegate to the correct Boolean object
- each subclass implements its own part



Summary

The solution to implement boolean operations:

- ○ does NOT use conditionals (if)
- **Do not ask, tell**
 - lets the receiver decide



A course by

S. Ducasse, G. Polito, and Pablo Tesone



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/>