INRIA

RMod

Abstract Classes

Stéphane Ducasse
stephane.ducasse@inria.fr
http://stephane.ducasse.free.fr/

Goal

Abstract classes
Examples

Abstract Classes

- Should not be instantiated (abstract in Java)
- But can define complete methods.
- Defines a protocol common to a hierarchy of classes that is independent from the representation choices.
- A class is considered as abstract as soon as one of the methods to which it should respond to is not implemented (can be a inherited one).

Abstract Classes in Smalltalk

- Depending of the situation, override new to produce an error.
- No construct: Abstract methods send the message self subclassResponsibility

- Tools check this situation and exploit it.
- Abstract classes are not syntactically different from instantiable classes, but a common convention is to use class comments: So look at the class comment and write in the comment which methods are abstract and should be specialized.

Example

Boolean>>not
"Negation. Answer true if the receiver is false, answer false if the receiver is true."

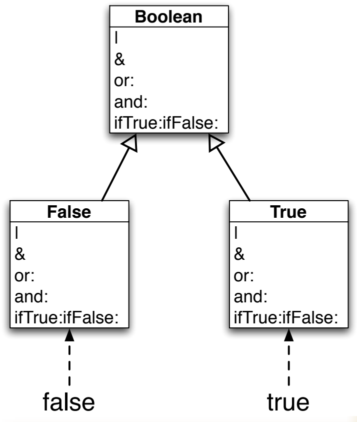
self subclassResponsibility

Goal

Abstract classes
Examples

Boolean Objects

false and true are objects described by classes Boolean, True and False



Conditional: messages to booleans

- aBoolean **ifTrue:** aTrueBlock **ifFalse:** aFalseBlock
- aBoolean **ifFalse:** aFalseBlock **ifTrue:** aTrueBlock
- aBoolean **ifTrue:** aTrueBlock
- aBoolean **ifFalse:** aFalseBlock

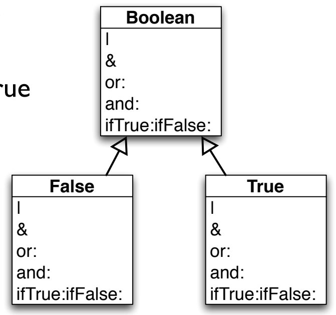
(thePacket isAddressedTo: self)
    ifTrue: [self print: thePacket]
    ifFalse: [super accept: thePacket]

- Hint: Take care — true is the boolean value and True is the class of true, its unique instance!

Boolean Hierarchy

- How to implement in OO true and false without conditional?

- Late binding: Let the receiver decide!
- Same message on false and true produces different results



Example

“Class Boolean is an abstract class that implements behavior common to true and false. Its subclasses are True and False. Subclasses must implement methods for logical operations &, not, controlling and:, or:, ifTrue:, ifFalse:, ifTrue:ifFalse:, ifFalse:ifTrue:”

```
Boolean>>not
    "Negation.  Answer true if the receiver is false, answer
    false if the receiver is true."
```

Not

false not -> true
true not -> false

```
Boolean>>not
    "Negation.  Answer true if the receiver is false, answer  false if
    the receiver is true.”
   self subclassResponsibility

False>>not
    "Negation -- answer true since the receiver is false."
    ^true
True>>not
    "Negation--answer false since the receiver is true."
    ^false
```

| (Or)

- **true** | true -> **true**
- **true** | false -> **true**
- **true** | anything -> **true**

- false | **true -> true**
- false | **false -> false**
- false | **anything -> anything**

Boolean>> | aBoolean

```
Boolean>> | aBoolean
  "Evaluating disjunction (OR).  Evaluate the argument.
  Answer true if either the receiver or the argument is
  true."

  self subclassResponsibility
```

False>> | aBoolean

false | **true -> true**
false | **false -> false**
false | **anything -> anything**

```
False>> | aBoolean
  "Evaluating disjunction (OR) -- answer with the
  argument, aBoolean."

  ^ aBoolean
```

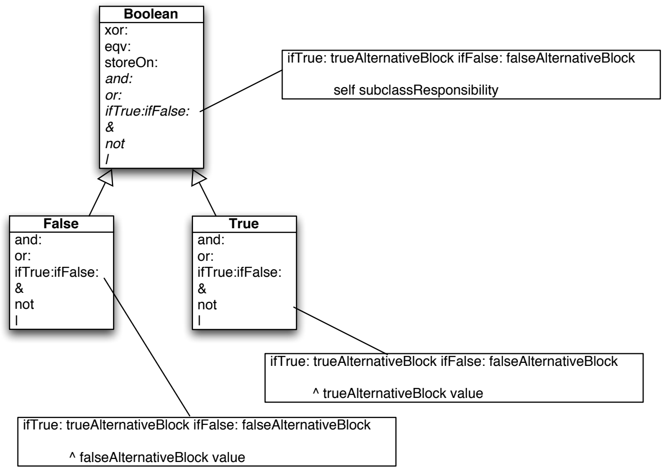
True>> | aBoolean

true | true -> **true**
true | false -> **true**
true | anything -> **true**

```
True>> | aBoolean
  "Evaluating disjunction (OR) -- answer true since the
  receiver is true."

  ^ self
```

Boolean, True and False



```
ifTrue: trueAlternativeBlock ifFalse: falseAlternativeBlock
        self subclassResponsibility
```

```
ifTrue: trueAlternativeBlock ifFalse: falseAlternativeBlock
        ^ trueAlternativeBlock value
```

```
ifTrue: trueAlternativeBlock ifFalse: falseAlternativeBlock
        ^ falseAlternativeBlock value
```

Abstract/Concrete

Abstract method

Boolean>>not
   "Negation. Answer true if the receiver is false, answer false if the receiver is true."
    self subclassResponsibility

Concrete method defined in terms of an abstract method

Boolean>>xor: aBoolean
   "Exclusive OR. Answer true if the receiver is not equivalent to aBoolean."
            ^(self == aBoolean) ***not***

When not is be defined in subclasses, xor: is

Block Use in Conditional?

- Why do conditional expressions use blocks?
- Because, when a message is sent, the receiver and the arguments of the message are *always* evaluated. Blocks are necessary to avoid evaluating both branches.

Implementation Note

Note that the Virtual Machine shortcuts calls to boolean such as condition for speed reason.

Virtual machines such as VisualWorks introduced a kind of macro expansion, an optimisation for essential methods and Just In Time (JIT) compilation. A method is executed once and afterwards it is compiled into native code. So the second time it is invoked, the native code will be executed.

Magnitude

I'm abstract class that represents the objects that can be compared between each other such as numbers, dates, numbers.

My subclasses should implement

```
< aMagnitude
= aMagnitude
hash
```

Here are some example of my protocol:

```
3 > 4
5 = 6
```

Magnitude

```
Magnitude>> < aMagnitude
   ^self subclassResponsibility

Magnitude>> = aMagnitude
        ^self subclassResponsibility

Magnitude>> hash
        ^self subclassResponsibility
```

Magnitude

```
Magnitude>> <= aMagnitude
        ^(self > aMagnitude) not

Magnitude>> > aMagnitude
        ^aMagnitude < self

Magnitude>> >= aMagnitude
        ^(self < aMagnitude) not

Magnitude>> between: min and: max
```

Date

Subclass of Magnitude

```
Date today < Date newDay: 15 month: 10 year: 1998
-> false
```

Date

Date>>< aDate

```
  "Answer whether the argument, aDate, precedes
  the date of the rec."

     year = aDate year
        ifTrue: [^day < aDate day]
        ifFalse: [^year < aDate year]
```

Date

Date>>= aDate

```
  "Answer whether the argument, aDate, is the
  same day as the receiver. "

     self species = aDate species
        ifTrue: [^day = aDate day & (year = aDate
year)]
        ifFalse: [^false]
```

Date>>hash

What you should know

- What is an abstract class?
- What can we do with it?