

Abstract Classes

- Should not be instantiated (abstract in Java)
- But can defined 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`.
- Advanced tools check this situation and exploit it.

Example

- 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.

“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:”

Case Study - Boolean, True, and False

```
Object ()  
  Boolean (&, not, |, and:, or:, ifTrue:,  
  ifFalse:, ifTrue:ifFalse:, ifFalse:ifTrue:)  
  False ()  
  True ()
```

Boolean

```
eqv:, xor:, storeOn:,  
shallowCopy
```

False

```
and:, or:, ifTrue:, ifFalse:,  
ifTrue:ifFalse:, ifFalse:ifTrue:  
&, not, |
```

True

```
and:, or:, ifTrue:, ifFalse:,  
ifTrue:ifFalse:, ifFalse:ifTrue:  
&, not, |
```

Case Study - Boolean, True, and False (II)

- 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 will be defined, #xor: is automatically defined

Case Study - Boolean, True, and False (III)

False>>not

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

^true

True>>not

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

^false

False>>ifTrue: trueBlock ifFalse: falseBlock

"Answer the value of falseBlock. This method is typically not invoked because ifTrue:/ifFalse: expressions are compiled in-line for literal blocks."

^falseBlock value

True>>ifTrue: trueBlock ifFalse: falseBlock

"Answer the value of trueBlock. This method is typically not invoked because ifTrue:/ifFalse: expressions are compiled in-line for literal blocks."

^trueAlternativeBlock value

Implementation Note

- Note that the Virtual Machine short cuts 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.

Case Study - Magnitude

• $1 > 2 = 2 < 1 = \text{false}$

Magnitude>> < aMagnitude
^self subclassResponsibility

Magnitude>> = aMagnitude
^self subclassResponsibility

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

Magnitude>> > aMagnitude
^aMagnitude < self

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

Magnitude>> between: min and: max
^self >= min and: [self <= max]

$1 <= 2 = (1 > 2) \text{ not}$
 $= \text{false not}$
 $= \text{true}$

Case Study - Date

Date>>< aDate

"Answer whether the argument, aDate, precedes the date of the receiver."

year = aDate year

if True: [^day < aDate day]

if False: [^year < aDate year]

Date>>= aDate

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

self species = aDate species

if True: [^day = aDate day & (year = aDate year)]

if False: [^false]

Date>>hash

^(year hash bitShift: 3) bitXor: day

Date

- Subclass of Magnitude

Date today > Date newDay: 15 month: 10 year:
1998

-> false

What you should know

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