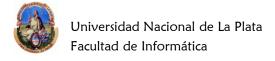
Más clases

Bloques (BlockClosure)
Boolean





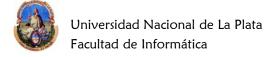
Motivación

'dibujar un cuadrado'

robotech brushDown.

4 timesRepeat: [robotech move:10; direction:90].

robotech brushUp

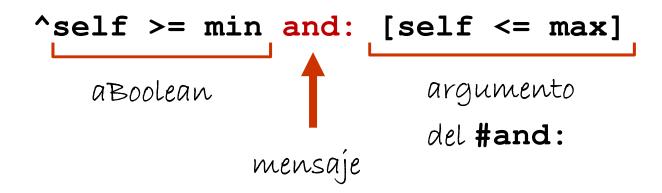


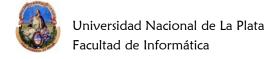


Motivación

Magnitude>>between: min and: max

"Answer whether the receiver is less than or equal to the argument, max, and greater than or equal to the argument, min."

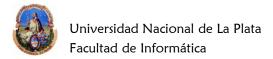






Bloques

Clase: BlockClosure





Motivación

• Pensemos en definir una función:

$$f(x) = x * x + x$$

 $f(2) = 6$
 $f(20) = 420$

• En Smalltalk

$$|f|$$

f:= [:x | x * x + x].





Otros éjemplos

$$[2 + 3 + 4 + 5]$$
 value

[:x |
$$x + 3 + 4 + 5$$
] value: 2

[:x:y |
$$x + y + 4 + 5$$
] value: 2 value: 3





Bloques: clase BlockClosure

- Los *bloques* son instancias de la clase *BlockClosure* que contienen una secuencia de sentencias
- es esencialmente una función anónima.
- se evalúa mediante el envío del mensaje #value
 - devuelve el valor de la última expresión en su cuerpo
 - al menos que haya una sentencia de retorno ^anObject
- son definidos como literales usando los [...]
 - » [30 squared]
 - » [Transcript clear. Transcript show: 'Hello']





Bloques: clase BlockClosure (2)

 Representan una ejecución retardada, las sentencias serán ejecutadas sólo si el programa lo requiere explícitamente cuando se le envía el mensaje #value

```
[30 squared] value
[30 squared. 2 + 5] value

[ Transcript clear.
   Transcript show: 'Hello'] value
```





Bloques: clase BlockClosure (3)

- como los bloques son objetos, entonces
 - pueden ser ligados a variables





Bloques: clase BlockClosure (3)

- como los bloques son objetos, entonces
 - puede tener argumentos, en ese caso se usa el mensaje #value: que liga su argumento al argumento del bloque antes de su ejecución

```
sizer := [ :array | array size < 3].

sizer value: \#(a b c d) \rightarrow 4

sizer value: \#(1 2) \rightarrow 2
```





Bloques: clase BlockClosure (3)

Un bloque puede tener variables temporales:

```
[:x :y | | z | z := x+ y. z ] value: 1 value: 2
```

 Pueden contener variables definidas afuera del bloque

```
| x |
x := 1.
[ :y | x + y ] value: 2 -> 3
```





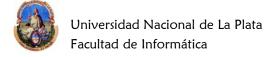
uso de bloques para íterar: #timesRepeat:

```
| amount |
amount:=0.
4 timesRepeat: [amount :=amount + 1]
```





La Clase Boolean





Clase Boolean

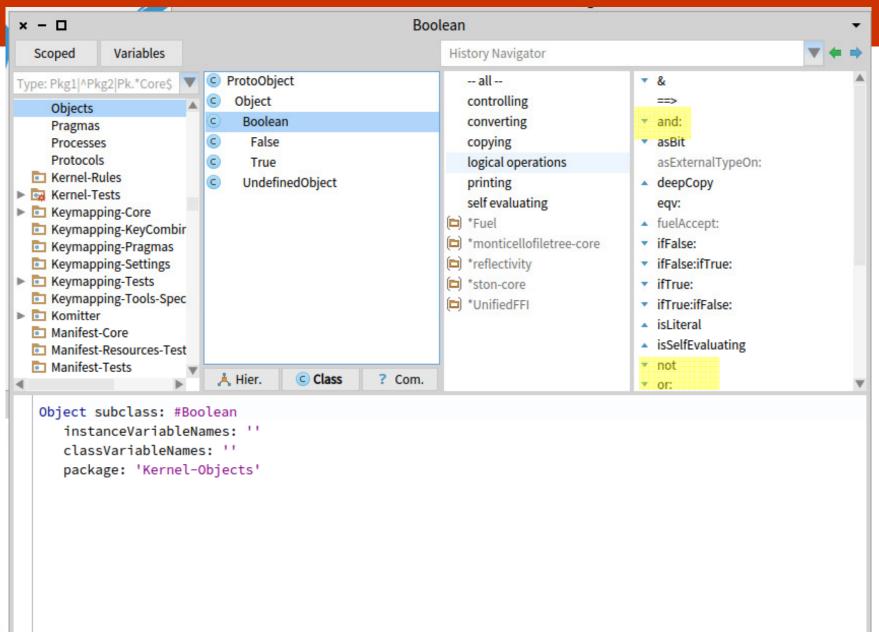
- es una clase abstracta cuya clases concretas son True y False
- Boolean
- C False
- C True

Existen sólo 2 instancias:

Las siguientes expresiones devuelven un booleano









Boolean: #and:

¿Dónde se implementa el #and:?

- Boolean
- False
- True

Boolean>> and:aBlock



si el receptor es false \Rightarrow ^false $sino \Rightarrow$ ^aBlock value

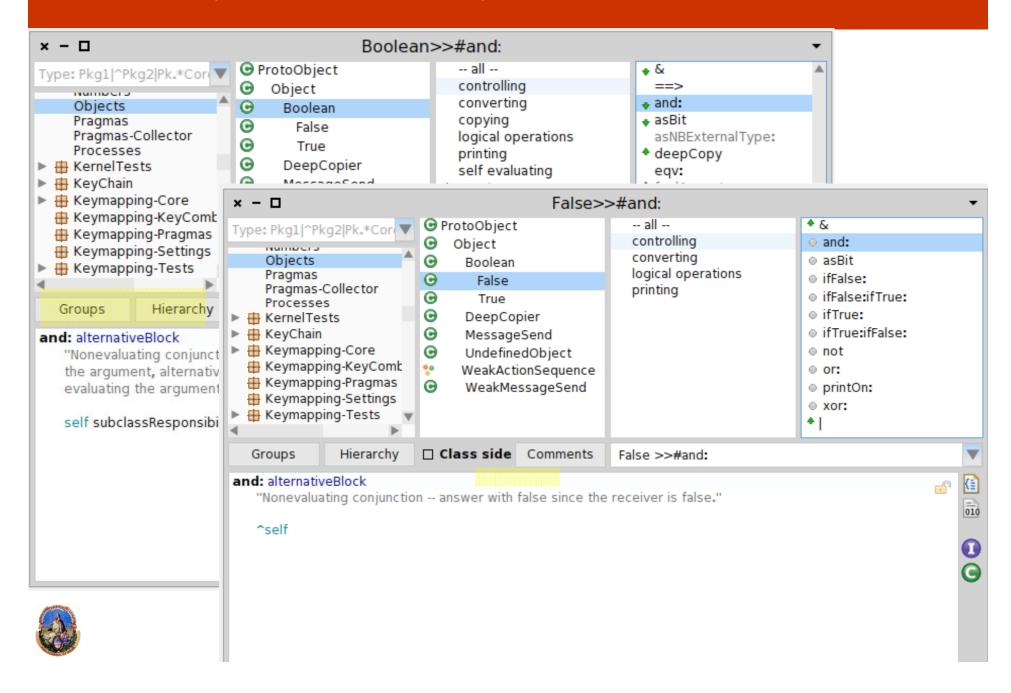
¿Hace falta preguntar si es false o true?

True>> and:aBlock
^aBlock value





La clase **Boolean** es abstracta



Boolean: más comportamiento

```
(3 < 4) & (5 < 6)
                  "logical AND"
(3 < 4) | (5 < 6)
                  "logical OR"
#ifTrue:
  a < b ifTrue:[Transcript clear;</pre>
                show: 'a is less than b'].
#ifFalse:
  a < b ifFalse:[Transcript clear;</pre>
                 show: 'a is NOT less than b'].
#ifTrue:ifFalse:
  a < b ifTrue: [Transcript clear;</pre>
                 show: 'a is less than b'l
        ifFalse:[Transcript clear;
                 show: 'a is NOT less than b'].
#ifFalse:ifTrue:
```



Implementación del #ifTrue:aBlock

True>> ifTrue: aBlock

"Evaluate the given Block if I am an instance of class True. Otherwise, do nothing."

^aBlock value

Hay que implementarlo en la clase False?

False>> ifTrue: aBlock

"Evaluate the given Block if I am an instance of class True. Otherwise, do nothing."

Ejercicio: implemetar el #ifTrue:ifFalse:





Boolean

#whileTrue: aBlock

En qué clase esta implementado?

BlockClosure>> whileTrue: aBlock

"Ordinarily compiled in-line, and therefore not overridable.

This is in case the message is sent to other than a literal block.

Evaluate the argument, aBlock, as long as the value of the receiver is true."

```
self value ifTrue: [aBlock value.
```

self whileTrue: aBlock]



