Understanding Messages

Damien Cassou, Stéphane Ducasse and Luc Fabresse

http://www.pharo.org
Objects, Messages and Closures

- We only manipulate **objects** (mouse, booleans, arrays, numbers, strings, ...)
- We only send them **messages** (@, +, not, getPng:, ifTrue:ifFalse:, new, ...)
- and we use **closures**
• Originally invented for kids
• Programs look like little sentences
• Try to minimize the number of parentheses
Example

(ZnEasy getPng: 'http://a.tile.openstreetmap.org/8/12/8.png')
asMorph openInWindow
Three Kinds of Messages to Minimize Parentheses

- **Unary message**: receiver selector
  - 9 squared, Date today

- **Binary message**: receiver selector argument
  - 1+2
  - 3@4

- **Keyword message**: receiver key1: arg1 key2: arg2
  - 2 between: 10 and: 20
Message Precedence

(Msg) > Unary > Binary > Keywords

- First we execute ()
- Then unary, then binary and finally keyword messages

This order minimizes () needs
But let us start with messages
Unary Message Examples

- `anObject aSelector`
- `1 class
  > SmallInteger`
- `false not
  > true`
- `Date today
  > 24 May 2009`
- `Float pi
  > 3.141592653589793`
Did you notice?

- We sent messages to any objects, including classes!
- There is no difference between sending a message to an object or to a class

1 class
> SmallInteger

Date today
> 27 June 2015
A Bit of Introspection

Point selectors

> #(#x #theta #quadrantOf: #onLineFrom:to:within: 
  #bitShiftPoint: #< #scaleFrom:to: #sideOf: #' \' #scaleTo: 
  #grid: #' // #asIntegerPoint #directionToLineFrom:to: ...)

- Returns all the messages the class understands
A Little Query

- Let us query the system and only filter the unary messages:

```
Point selectors select: #isUnary
> #(#x #theta #asIntegerPoint #r #negated #normalized #sign
 #degrees #isIntegerPoint #guarded #fourNeighbors
 #eightNeighbors #min #max #ceiling #normal #asPoint #y
 #abs #isPoint #angle #transposed #reciprocal
 #asFloatPoint #asNonFractionalPoint #rounded
 #leftRotated #floor #truncated #hash #deepCopy
 #fourDirections #rightRotated #isSelfEvaluating #asMargin
 #isZero)
```

- select: is an iterator (see Iterator lecture)

- Easy :-)
Binary Messages

- Used for arithmetic, comparison and logical operations
- One, two or three characters taken from:
  - + - / \ * ~ < > = @ % | & ! ? ,
Binary Message Examples

1 + 2
> 3

2 > 3
> false

10@200
> 10@200

'Black chocolate', 'is good'
> 'Black chocolate is good'
Keyword Messages

anObject keyword1: arg1 keyword2: arg2

equivalent to:

receiver.keyword1keyword2(arg1, arg2)
Test Yourself!

- 1 log
- Browser open
- 2 raisedTo: 5
- ’hello’, ’world’
- 10@20
- point1 x
- point1 distanceFrom: point2
Test Yourself!

- 1 log (unary)
- Browser open (unary)
- 2 raisedTo: 5 (keyword)
- ’hello’, ’world’ (binary)
- 10@20 (binary)
- point1 x (unary)
- point1 distanceFrom: point2 (keyword)
Example: Message setX:

10@20 setX: 2
> 2@20

- We change the x value of the receiver (a point)
- No parentheses required
Example: Message at:put:

```plaintext
#('Calvin' 'hates' 'Suzie') at: 2 put: 'loves'
> #(Calvin' 'loves' 'Suzie')
```

- `#(...) creates an array`
- `at:put:` changes the value of the array element.
- `arrays start at 1 in Pharo (i.e., first element is at index 1)`
Example: Message between:and:

12 between: 10 and: 20
> true

- The message between:and: is sent to an integer
- Takes two arguments 10 and 20
Summary

Three kinds of messages: unary, binary and keywords

- **Unary**
  - 5 factorial

- **Binary**
  - 2 + 3

- **Keywords-based messages**
  - 2 between: 0 and: 10