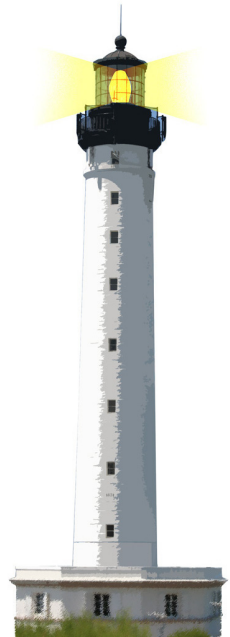
A Glance at Numbers

Damien Cassou, Stéphane Ducasse and Luc Fabresse

WXSYY

Pharo

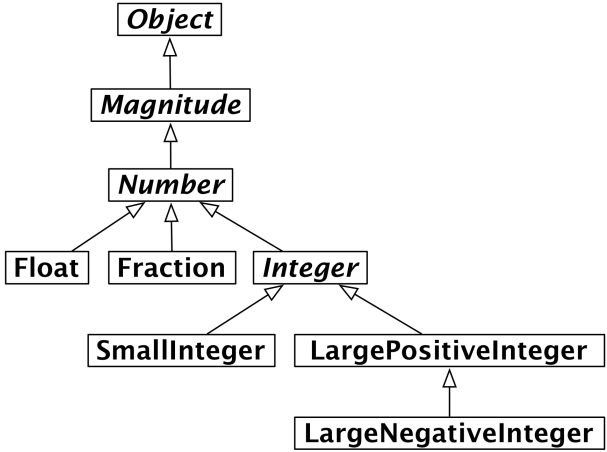
http://www.pharo.org



Numbers

- SmallInteger, Integer,
  - 4, 2r100 (4 in base 2), 3r11 (4 in base 3)
- Automatic coercion
  - 1 + 2.3 -> 3.3
  - 1 class -> SmallInteger
  - 1 class maxVal class -> SmallInteger
  - (1 class maxVal + 1) class -> LargePositiveInteger
- Fraction, Float
  - 3/4, 2.4e7
  - (1/3) + (2/3) -> 1
  - 1000 factorial / 999 factorial -> 1000
  - 2/3 + 1 -> (5/3)

Numbers


Object
Magnitude
Number
Float
Fraction
Integer
SmallInteger
LargePositiveInteger
LargeNegativeInteger

Small Integers are Real Objects

- Small ints are real objects: instance of class SmallInteger
- No need for boxing or unboxing

```
1 class
> SmallInteger
```

- Operations on small integers are plain messages (no exception to the rule)

Small integers are optimized

But small integers are heavily optimized

- Small integers encoded on 30 bits
- The pointer itself is the small integer

```
2 raisedTo: 30
> 1073741823
```

```
SmallInteger maxVal
> 1073741823
```

Automatic Coercion

What is the largest small integer?

```
1 class maxVal
> 1073741823
```

What is the smallest large integer?

```
1 class maxVal +1
> 1073741824

(1 class maxVal + 1) class
> LargePositiveInteger
```

Fun With Numbers

Fun With Large Numbers

factorial is not optimized

```
1000 factorial numberOfDigits
> 2568
```

```
100000 factorial numberOfDigits
> 456574
```

- Takes some seconds :)

Fraction

Fraction

- denominator, numerator, ...
- Can handle large numbers

1000 factorial / 999 factorial
> 1000

Messages Sent to Objects

- Operations on numbers are plain messages
- No exceptions

```
(1 / 3) + (2 / 3)
>1
```

```
2 / 3 + 1
> 5 / 3
```

- Numbers are objects
- Mathematical operations are messages sent to objects

Message Limits

- There is no notion of precedence

```
2 * 3 + 5
> 11
```

```
2 + 3 * 5
> 25 !!!
```

- Use parenthesis to enforce precedence

```
2 + (3 * 5)
> 11
```

Points

- A point has an x and y
- Points are created using the message @ or x:y:

```
10 @ 100
(10 @ 100) x
> 10
(10 @ 100) y
>100
```

Alternative

```
Point new x: 100 y: 100
```

Class written in a functional way: Most Point methods are returning new points

Rectangles

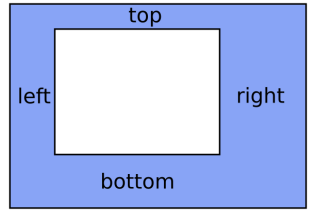
- Geometric and expected operations
- areasOutside:, expandBy:, insetBy:, intersect:

```
Rectangle left: 10 right: 100 top: 20  bottom: 40
> (10@20) corner: (100@40)
```

```
Rectangle origin: 20@20 corner: 100@200
```

Margins

- Delta to apply to a rectangle
- Useful for GUI
- Encodes 4 numbers, expressed in 3 different ways
  - a number (same space all around)
  - two numbers (same left/right and top/bottom)
  - four numbers
- Used to compute the extended (or inner) rectangle



Summary

- Numbers are real objects
- Automatic coercion
- Number can be infinitely large
- Abstractions are built on top: Point, Rectangle, Margin

A course by

and

in collaboration with