Objects and Design in Smalltalk

Smalltalk Basics

Éric Tanter
etanter@dcc.uchile.cl





Elementary Objects

- Some elementary Smalltalk objects have their own notation

Sample Instance	Its Class
37	Integer (SmallInteger)
'To be or not to be'	String
2.7188	Float
$p	Character
true	Boolean (True)
false	Boolean (False)

Messages and Precedence

- evaluation from left to right
- 3 kinds of messages
  - unary: 'iahcac' reversed 4 factorial
  - binary: 17 <= 14 4 + 3 'hello ' , 'world'
  - keyword:
    - HomeBudget spend: 229 on: 'Gas'
    - (HomeBudget spend: 229) on: 'Gas'
- precedence rule: unary -> binary -> keyword
  - 12 between: 7 and: 9 + 2
  - (12 between: 7 and: 9) + 2

Cascading Messages

- For conveniently sending various messages to the same object

```
SomeObject msg1.
SomeObject msg2.
SomeObject msg3.
```

```
SomeObject msg1;
           msg2;
           msg3.
```

Assignments

- Smalltalk assignments are not messages
  - (you can view them as such if you want to)
  - right side is processed first
  - `X := 12 between: 7 and: 9 + 2`

Symbols, Strings, Arrays

- Strings and symbols
  - 'rosebud' is a String
  - #rosebud is a Symbol (subclass of String)
  - two different strings with the same sequence can exist
  - a symbol is unique! (#rosebud copy -> same!)
    - fundamental role in object identity
    - class names are symbols!
- Strings and characters
  - 'rosebud', '', 'b' are String objects
  - $b is a Character object
- Arrays: #(5 'rosebud' 7) #(5 #(2 11 13) 7)

Exercise

- What is returned by these expressions?

```
3 - 5 * 2
Integer superclass
#(me you they) at: 2
```

- What is the effect of these messages?

```
#zero at: 1 put: $h
'zero' at: 1 put: $h
'zero' copy at: 1 put: $h
#zero copy at: 1 put: $h
```


Variables

- Naming conventions
  - restricted visibility: lowercase
    - formal arguments: anObject, aTable, etc.
    - local / temporary variables: last, next, etc.
    - instance variables: numerator, children, etc.
  - shared variables: uppercase
    - global variables: Transcript, X, class names!
    - class variables and class instance variables
- Variable ≠ Object
  - variable X can point to any type of object
  - variables are untyped, objects are strongly typed

Example

- What does this method do?
- What is a class for which it would make sense?

```
replaceLastBy: anObject
  |last|
  last := self size.
  self at: last put: anObject.
  ^self
```

Classes are Objects

- Remember: "everything is an object"
  - classes (factories) are also objects
  - "Smalltalk classes are first-class objects"
    - this is not the case in most OO languages!!

```
Whale new talk
```

- two kinds of methods
  - apply to instances: instance methods
  - apply to classes: class methods
  - see Smalltalk browser instance/class switch

Blocks

- Smalltalk has no language statements for conditionals (if-then) and loops (while/for)?!
  - how? with the only think Smalltalk knows, messages
- Blocks
  - special kind of object: a chunk of code without a name

```
MyBlock := [Whale new talk] -> nothing
MyBlock value -> 'I spout and sing!'
```


Blocks

- Conditional and loops rely on blocks!

```
MyValue < 17
  ifTrue: [Whale new talk]
```

```
6 timesRepeat: [K := K + 1]
```

- the Smalltalk object model is extreme
  - even integers have rich behavior!

Blocks

the lambdas of object orientation

- Blocks can also have arguments

```
MyProduct := 1.
S := Set with: 5 with: 3 with: 4.
S do: [ :num | MyProduct := MyProduct * num].
MyProduct
```

- How do you 'emulate' blocks in Java??
  - how many lines of code do you need for something like do: aBlock?