Goals

Syntax in a Nutshell

OO Model in a Nutshell

Smalltalk in a Nutshell

Complete Syntax on a PostCard

Language Constructs

Syntax

Syntax in a Nutshell (II)

Class Definition in St-80

Method Definition

Smalltalk OO Model

- Everything is an object
- Only message passing
- Only late binding
- Instance variables are private to the object
- Methods are public
- Everything is a pointer
- Garbage collector
- Single inheritance between classes
- Only message passing between objects

Class Definition in St-80

```smalltalk
Node>>accept: thePacket
| result |
\[\text{result := (thePacket isAddressedTo: self)} \]
\[\text{ifTrue: \{self print: thePacket\}} \]
\[\text{ifFalse: \{super accept: thePacket\}} \]
```

Method Definition

- Normally defined in a browser or (by directly invoking the compiler)
- Methods are public
- Always return self

Node>>accept: thePacket
- "If the packet is addressed to me, print it. Else just behave like a normal node"
- (thePacket isAddressedTo: self)
- ifTrue: [self print: thePacket]
- ifFalse: [super accept: thePacket]
- ‘1’, ‘abc’
- Basic class creation messages are `new`, `new:`, `basicNew`, `basicNew:`
- Monster new
- Class specific message creation (messages sent to classes)
  Tomagoshi with hunger: 10

**Instance Creation: Messages Too!**

- Anonymous method
  - Passed as method argument or stored
  - Functions
    - `fct(x) = x^2 + 3, fct(2)`, `fct := [x | x * x + 3]`.
    - `fct value: 2`

**Messages and their Composition**

- Three kinds of messages
  - Unary: `Node new`
  - Binary: `1 + 2, 3@4`
  - Keywords: `T omagoshi eat: #cooky furiously: true`

**Message Priority**

- `(Msg) > unary > binary > keywords`
- Same Level from left to right
- Example:
  - `(10@0 extent: 10@100) bottomRight`
  - `s isNil ifTrue: [ self halt ]`

**Blocks**

- Integer>>factorial
  - `tmp := 1`
  - `2 to: self do: [i | tmp := tmp * i]`
  - `#(1 2 3) do: [each | Transcript show: each printString ; cr ]`

**Yes ifTrue: is sent to a boolean**

- Weather `isRaining`
  - `ifTrue: [self takeMyUmbrella]
  - `ifFalse: [self takeMySunglasses]`
  - `ifTrue: ifFalse is sent to an object: a boolean!`

**Yes a collection is iterating on itself**

- `#(1 2 -4 -86)`
  - `do: [each | Transcript show: each abs printString ; cr ]`
  - `> 1
  - `> 2
  - `> 4
  - `> 86

**Yes we ask the collection object to**

**Goals**

- Syntax in a Nutshell
  - *OO Model in a Nutshell*

**Summary**

- Objects and Messages
  - Three kinds of messages
    - Unary
    - Binary
    - Keywords
  - Block: a.k.a innerclass or closures or lambda
  - `Unary > Binary > Keywords`

**Instance and Class**

- Only one model
- Uniformly applied
- Classes are objects too

**Lookup…Class + Inheritance**

- `Object`
- `Node`
- `msg`
Classes are objects too

- Instance creation is just a message send to a Class
- Same method lookup than with any other objects
- A Class is the single instance of an anonymous class
- Point is the single instance of Point class

Class Parallel Inheritance

- Node class
- new withName: aString
- instance of Node

- Workstation
- originate: aPacket
- accept: aPacket

aWorkstation (BigMac)

Workstation

class

instance of Object class

Object

withName: 'BigMac'

instance of name

About the Buttons

- Instance of Button

Summary

- Everything is an object
- One single model
- Single inheritance
- Public methods
- Protected attributes
- Classes are simply objects too
- Class is instance of another class
- One unique method lookup
- Look in the class of the receiver

- Look in the class of the receiver