

# CPS506 - Comparative Programming Languages

## Smalltalk

Dr. Dave Mason  
Department of Computer Science  
Ryerson University

©2022 Dave Mason 

RYERSON  
UNIVERSITY

## Smalltalk

In this section we will:

- Look at Smalltalk history
- Explore Smalltalk Syntax and Pharo

## History

- Alan Kay at Xerox PARC, with Adele Goldberg and Dan Ingalls, along with Ted Kaehler, Larry Tesler
- wanted to build Dynabook
- Sketchpad, LOGO, Lisp, Simula
- Smalltalk-71
- Smalltalk-72 on the ALTO
- Smalltalk-76
- Smalltalk-80 - standard to today
- extensions, but forward compatible

## Overview

- the prototypical class-based object-oriented language
- minimal - no reserved words
- control structures as methods
- typically high-performance byte-code interpreter

## Paradigm

- pure Object-Oriented
- class based
- simple, metacircular, reflective

## Syntax

```
exampleWithNumber: x
  | y |
  true & false not & (nil isNil) ifFalse: [self halt].
  y := self size + super size.
  #($a #a 'a' "a" 1 1.0)
    do: [ :each |
          Transcript show: each class name;
          show: ' '].
  ^x < y
```

# Syntax Rules

## 1 literals

- numbers: `-17`  
`3.141592`  
`2r101`  
`16r2c4f`
- characters: `$a`  
`$(`
- strings: `'this isn't "hard"'`
- symbols: `#asymbol` `#'a symbol'`  
`#aSymbol:`
- arrays:  
`$( abc nil #nil 3 ** 'string' (a subarray) $! )`
- blocks: `[ 3 ] [ : arg | arg-4 ]`

## 2 variables

- upper/lower case, digits; case sensitive; camel-case
- arguments to methods and blocks
- temporaries | `a b` | at beginning of methods and blocks
- instance variables
- global variables - includes class names

No reserved words; only `self super nil true false`

predefined

## 3 messages

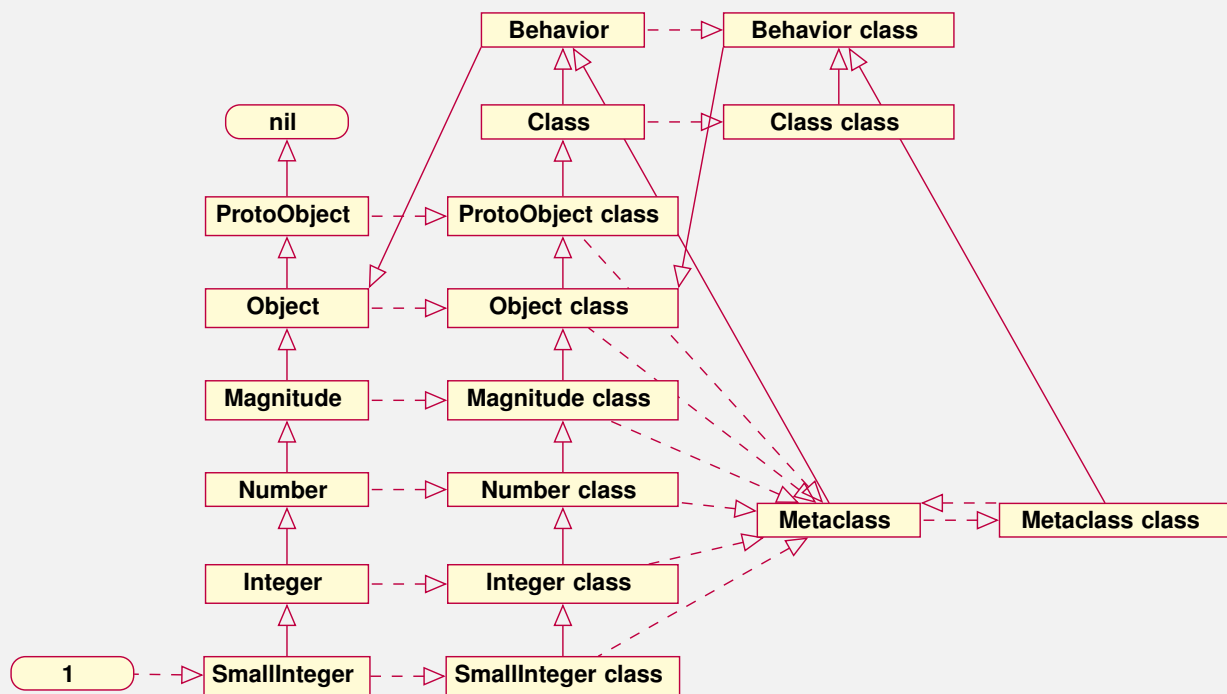
# Semantics

- everything is an object
- only 3 operations
  - assignment
  - message send
  - return result
- class-based method tables
- intrinsic rich meta-environment

• `x := 3. y:=x*5`

- Squeak version of Smalltalk built from scratch at Apple, Walt Disney Imagineering by Dan Ingalls and Alan Kay
- built to explore Dynabook ideas
- 2 significant forks:
  - Pharo for “business”
  - Cuis for teaching
- Currently Pharo 10 pharo.org

## Class Structure

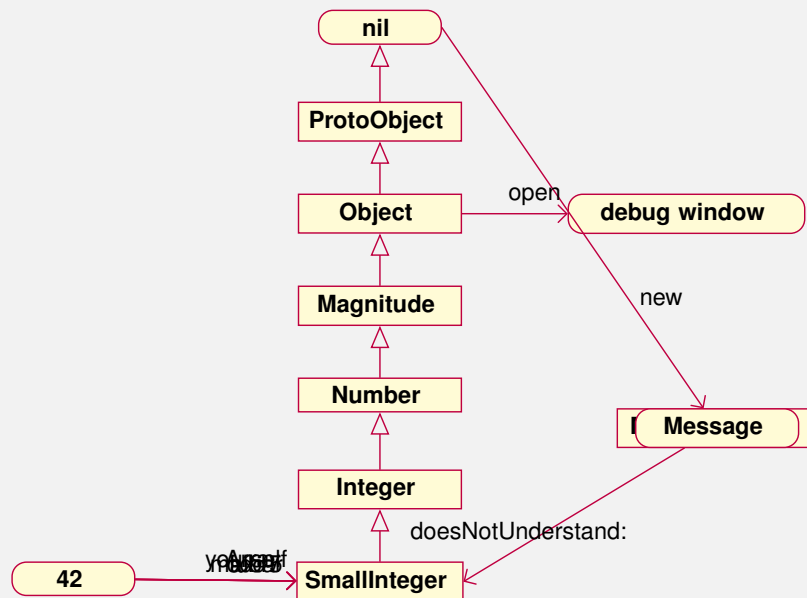


```

1 1 class 1 class superclass 1 class superclass superclass superclass
superclass superclass superclass 1 class class 1 class class superclass
= 1 class superclass class 1 class class superclass superclass
superclass superclass superclass 1 class class superclass superclass superclass

```

## Method Dispatch



42 42 odd 42 max: 5 42 yourself 42 fubar 42 Array *(after defining a DNU for SmallInteger)*

## Method Dispatch...

- many special cases
- “implementing a language like Smalltalk efficiently requires the implementor to cheat... but that’s okay as long as you don’t get caught” - Peter Deutsch (creator of JITs and many other language optimizations)
- `ifTrue:ifFalse: do: whileTrue:, etc.`
- primitives
- hashed dispatch

## Types

- hardware level - instructions act on register of bits
- statically typed
  - types determined and instructions chosen at compile time
  - **variables** and **expressions** have types
  - less flexible, sometimes much less
  - safety variable
- dynamically typed
  - types determined and instructions chosen at run time
  - **values** have types
  - safety assured

## Type determination

- values tagged - size of a register
- fallback is to heap-allocated object
- simplest tagging is just differentiating `SmallInteger`
- description for AST Smalltalk - alternate dispatch
- documentation for GNU Smalltalk
- code for OpenSmalltalk

## Pragmatics

- garbage-collected
- usually image-based development
- best-in-class IDE
- optimized VM (JIT)

## Environment and IDE

- 1 image-based
- 2 IDE that others aspire to
- 3 class browser, playground, debugger, inspector, senders, receivers, refactoring, transcript, unit-test runner, code critic, method versions, interruptable
- 4 even if crashes, changes recoverable



# Evaluation

- Simplicity
  - Size of the grammar
  - complexity of navigating modules/classes
- Orthogonality
  - number of special syntax forms
  - number of special datatypes
- Extensibility
  - functional
  - syntactically
  - defining literals
  - overloading