Dynamic Object-Oriented Programming with Smalltalk

1. Introduction

Prof. O. Nierstrasz
Autumn Semester 2009
# Smalltalk

<table>
<thead>
<tr>
<th><strong>Lecturer</strong></th>
<th>Prof. Oscar Nierstrasz</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Assistants</strong></td>
<td>David Röthlisberger, Fabrizio Perin</td>
</tr>
<tr>
<td></td>
<td>Timur Altun</td>
</tr>
<tr>
<td><strong>Lectures</strong></td>
<td>IWI 001, Wednesdays @ 10h15-12h00</td>
</tr>
<tr>
<td><strong>Exercises</strong></td>
<td>IWI 001, Wednesdays @ 12h00-13h00</td>
</tr>
<tr>
<td><strong>WWW</strong></td>
<td><a href="http://scg.unibe.ch/teaching/smalltalk">http://scg.unibe.ch/teaching/smalltalk</a></td>
</tr>
</tbody>
</table>

Selected material courtesy Stéphane Ducasse
Smalltalk is still today one of the few fully reflective, fully dynamic, object-oriented development environments.

We will see how a simple, uniform object model enables live, dynamic, interactive software development.
Roadmap

- Course schedule, goals, resources
- What is Smalltalk?
- Origins of Smalltalk
- Smalltalk key concepts
- The Smalltalk environment
Roadmap

> Course schedule, goals, resources
> What is Smalltalk?
> Origins of Smalltalk
> Smalltalk key concepts
> The Smalltalk environment
<table>
<thead>
<tr>
<th>Week</th>
<th>Date</th>
<th>Lecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>16-Sep-09</td>
<td>Introduction</td>
</tr>
<tr>
<td>2</td>
<td>23-Sep-09</td>
<td>Smalltalk Basics</td>
</tr>
<tr>
<td>3</td>
<td>30-Sep-09</td>
<td>Standard Classes</td>
</tr>
<tr>
<td>4</td>
<td>07-Oct-09</td>
<td>Smalltalk Coding Idioms</td>
</tr>
<tr>
<td>5</td>
<td>14-Oct-09</td>
<td>Seaside</td>
</tr>
<tr>
<td>6</td>
<td>21-Oct-09</td>
<td>Debugging</td>
</tr>
<tr>
<td>7</td>
<td>04-Nov-09</td>
<td>Best Practice Patterns</td>
</tr>
<tr>
<td>8</td>
<td>28-Oct-09</td>
<td>Refactoring and Design Patterns</td>
</tr>
<tr>
<td>9</td>
<td>11-Nov-09</td>
<td>Understanding Classes and Metaclasses</td>
</tr>
<tr>
<td>10</td>
<td>18-Nov-09</td>
<td>Reflection</td>
</tr>
<tr>
<td>11</td>
<td>25-Nov-09</td>
<td>Working with ByteCode</td>
</tr>
<tr>
<td>12</td>
<td>02-Dec-09</td>
<td>Virtual Machines</td>
</tr>
<tr>
<td>13</td>
<td>09-Dec-09</td>
<td>Traits and Classboxes</td>
</tr>
<tr>
<td>14</td>
<td><strong>16-Dec-09</strong></td>
<td><strong>Final Exam</strong></td>
</tr>
</tbody>
</table>
Pharo by Example (preview)

Special preview edition prepared for this course
Goals of this Course

> Some history
> A pure object-oriented model
> Classes and metaclasses
> Reflection (not just introspection)
> Design and implementation of dynamic languages
> Advanced object-oriented design concepts
What is surprising about Smalltalk

> Everything is an object
> Everything happens by sending messages
> All the source code is there all the time
> You can't lose code
> You can change everything
> You can change things without restarting the system
> The Debugger is your Friend
A Word of Advice

You do not have to know everything!!!

Try not to care — Beginning Smalltalk programmers often have trouble because they think they need to understand all the details of how a thing works before they can use it. This means it takes quite a while before they can master Transcript show: ‘Hello World’. One of the great leaps in OO is to be able to answer the question “How does this work?” with “I don’t care”.

—Alan Knight. Smalltalk Guru
Resources

- Pharo
  - www.pharo-project.org
- History
  - en.wikipedia.org/wiki/Smalltalk
  - www.smalltalk.org/smalltalk/history.html
- Free books
  - stephane.ducasse.free.fr/FreeBooks.html
- European Smalltalk Users Group
  - www.esug.org
Recommended Books


> Sherman Alpert et al., *The Smalltalk Design Pattern Companion*, Addison-Wesley, 1998
Roadmap

- Course schedule, goals, resources
- **What is Smalltalk?**
- Origins of Smalltalk
- Smalltalk key concepts
- The Smalltalk environment
Why Smalltalk?

> Pure object-oriented language and environment
  — “Everything is an object”

> Origin of many innovations in OO development
  — RDD, IDE, MVC, XUnit …

> Improves on many of its successors
  — Fully interactive and dynamic
What is Smalltalk?

- **Pure OO language**
  - Single inheritance
  - Dynamically typed

- **Language and environment**
  - Guiding principle: “Everything is an Object”
  - Class browser, debugger, inspector, …
  - Mature class library and tools

- **Virtual machine**
  - Objects exist in a persistent image [+ changes]
  - Incremental compilation
## Smalltalk vs. C++ vs. Java

<table>
<thead>
<tr>
<th></th>
<th><strong>Smalltalk</strong></th>
<th><strong>C++</strong></th>
<th><strong>Java</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Object model</strong></td>
<td>Pure</td>
<td>Hybrid</td>
<td>Hybrid</td>
</tr>
<tr>
<td><strong>Garbage collection</strong></td>
<td>Automatic</td>
<td>Manual</td>
<td>Automatic</td>
</tr>
<tr>
<td><strong>Inheritance</strong></td>
<td>Single</td>
<td>Multiple</td>
<td>Single</td>
</tr>
<tr>
<td><strong>Types</strong></td>
<td>Dynamic</td>
<td>Static</td>
<td>Static</td>
</tr>
<tr>
<td><strong>Reflection</strong></td>
<td>Fully reflective</td>
<td>Introspection</td>
<td>Introspection</td>
</tr>
<tr>
<td><strong>Concurrency</strong></td>
<td>Semaphores, Monitors</td>
<td>Some libraries</td>
<td>Monitors</td>
</tr>
<tr>
<td><strong>Modules</strong></td>
<td>Categories, namespaces</td>
<td>Namespaces</td>
<td>Packages</td>
</tr>
</tbody>
</table>
Smalltalk: a State of Mind

> **Small and uniform language**
> — Syntax fits on one sheet of paper

> **Large library of reusable classes**
> — Basic Data Structures, GUI classes, Database Access, Internet, Graphics

> **Advanced development tools**
> — Browsers, GUI Builders, Inspectors, Change Management Tools, Crash Recovery Tools, Project Management Tools

> **Interactive virtual machine technology**
> — Truly platform-independent

> **Team Working Environment**
> — Releasing, versioning, deploying
Smalltalk in industry

> **Worldwide:**
  - [http://www.esug.org/companiesdevelopinginsmalltalk/](http://www.esug.org/companiesdevelopinginsmalltalk/)
  - [http://www.stic.org/companies/companies.htm](http://www.stic.org/companies/companies.htm)

> **In Bern:**
  - Netstyle.ch
  - DVBern AG
  - Mobiliar (in-house)
  - Pulinco
ST — Introduction

Roadmap

- Course schedule, goals, resources
- What is Smalltalk?
- Origins of Smalltalk
- Smalltalk key concepts
- The Smalltalk environment
Origins of Smalltalk

> **Project at Xerox PARC in 1970s**
  — Language and environment for new generation of graphical workstations (target: “Dynabook”)

> **In Smalltalk-72, every object was an independent entity**
  — Language was designed for children (!)
  — Evolved towards a meta-reflective architecture

> **Smalltalk-80 is the standard**
Smalltalk — The Inspiration

> **Flex** (Alan Kay, 1969)
> **Lisp** (Interpreter, Blocks, Garbage Collection)
> Turtle graphics (The **Logo** Project, Programming for Children)
> Direct Manipulation Interfaces (**Sketchpad**, Alan Sutherland, 1960)
> **NLS**, (Doug Engelbart, 1968), “the augmentation of human intellect”
> **Simula** (Classes and Message Sending)
> Xerox PARC (Palo Alto Research Center)
> **DynaBook**: a Laptop Computer for Children

— [www.smalltalk.org/smalltalk/TheEarlyHistoryOfSmalltalk_Abstract.html](http://www.smalltalk.org/smalltalk/TheEarlyHistoryOfSmalltalk_Abstract.html)
Dynabook Mockup

www.artmuseum.net/w2vr/archives/Kay/01_Dynabook.html
Alto: a Machine to Run Smalltalk

Smalltalk on Alto III
Precursor, Innovator & Visionary

> First to be based on Graphics
  — Multi-Windowing Environment (Overlapping Windows)
  — Integrated Development Environment: Debugger, Compiler, Text Editor, Browser
> With a pointing device ☞ yes, a Mouse
> Ideas were taken over
  — Apple Lisa, Mac
  — Microsoft Windows 1.0
> Platform-independent Virtual Machine
> Garbage Collector
> Just-in-time Compilation
> Everything was there, the complete Source Code
History

1950

1960
- Simula 67
- Smalltalk 72
- ALGOL 60
- ALGOL 68
- FORTRAN
- COBOL

1970
- Smalltalk 80
- Objective C
- C
- C++
- awk
- Pascal
- Clu

1980
- Self
- Java
- JavaScript
- Ruby
- Perl
- Modula-2
- Eiffel
- Ada
- Oberon

1990

2000
- Squeak
- Java
- JavaScript
- Ruby
- C#
- Groovy

2010
- Pharo
The History (Internal)

> **1972 — First Interpreter**
  > More Agents than Objects
  > (every object could specify its own syntax!)

> **1976 — Redesign**
  > A hierarchy of classes with a unique root, fixed syntax, compact bytecode, contexts, processes, semaphores, browsers, GUI library.
  > Projects: ThingLab, Visual Programming Environment, Programming by Rehearsal.

> **1978 — NoteTaker Project**
  > Experimentation with 8086 Microprocessor with only 256 KB RAM.
The History (External)

> **1980 — Smalltalk-80**
  - ASCII, cleaning primitives for portability, metaclasses, blocks as first-class objects, MVC.
  - Projects: Gallery Editor (mixing text, painting and animations) + Alternate Reality Kit (physics simulation)

> **1981 — Books + 4 external virtual machines**
  - Dec, Apple, HP and Tektronix
  - GC by generation scavenging

> **1988 — Creation of Parc Place Systems**

> **1992 — ANSI Draft**

> **1995 — New Smalltalk implementations**
  - MT, Dolphin, **Squeak**, Smalltalk/X, GNU Smalltalk

> **2000 — Fscript, GNU Smalltalk, SmallScript**

> **2002 — Smalltalk as OS: 128k ram**
What are Squeak and Pharo?

> Squeak is a modern, open-source, highly portable, fast, full-featured Smalltalk implementation  
   — Based on original Smalltalk-80 code

> Pharo is a lean and clean fork of Squeak  
   — www.pharo-project.org
Roadmap

- Course schedule, goals, resources
- What is Smalltalk?
- Origins of Smalltalk
- Smalltalk key concepts
- The Smalltalk environment
Smalltalk — Key Concepts

> **Everything is an object**
  — numbers, files, editors, compilers, points, tools, booleans …

> Everything happens by **sending messages**

> Every object is an instance of one class
  — which is also an object
  — A class defines the structure and the behavior of its instances.

> Objects have private (protected) state
  — Encapsulation boundary is the object

> Dynamic binding
  — Variables are dynamically typed and bound
Objects and Classes

> *Every object is an instance of a class*

— A class specifies the structure and the behaviour of all its instances
— Instances of a class share the same behavior and have a specific state
— *Classes are objects* that create other instances
— *Metaclasses* are classes that create classes as instances
— Metaclasses describe class behaviour and state (subclasses, method dictionary, instance variables...)
Messages and Methods

> **Message** — which action to perform

```smalltalk
aWorkstation accept: aPacket
aMonster eat: aCookie
```

> **Method** — how to carry out the action

```smalltalk
accept: aPacket
    (aPacket isAddressedTo: self)
    ifTrue: [
        Transcript show:
            'A packet is accepted by the Workstation ',
            self name asString
    ]
    ifFalse: [super accept: aPacket]
```
Smalltalk Run-Time Architecture

> Virtual Machine + Image + Changes and Sources

- All the objects of the system at a moment in time
- Image = bytecodes
- Sources and changes = code (text)

- IMAGE1.IM
- IMAGE1.CHA
- IMAGE2.IM
- IMAGE2.CHA

A byte-code interpreter: the virtual machine interpretes the image

Standard SOURCES

Shared by everybody
Smalltalk Run-Time Architecture

> Byte-code is translated to native code by a just-in-time compiler
   — Some Smalltalks, but not Pharo
> Source and changes are not needed to interpret the byte-code.
   — Just needed for development
   — Normally removed for deployment
> An application can be delivered as byte-code files that will be executed with a VM.
   — The development image is stripped to remove the unnecessary development components.
Roadmap

> Course schedule, goals, resources
> What is Smalltalk?
> Origins of Smalltalk
> Smalltalk key concepts
> The Smalltalk environment
Mouse Semantics

Select

Operate

Window
World Menu
“Hello World”
The Smalltalk Browser
The Debugger

/ aNumber
"Primitive. This primitive (for /) divides the receiver by the argument and returns the result if the division is exact. Fail if the result is not a whole integer. Fail if the argument is 0 or is not a SmallInteger. Optional. No Lookup. See Object documentation whatIsAPrimitive."

<primitive: 10>
aNumber isZero ifTrue: [([ZeroDivide dividend: self] signal).
^((aNumber isMemberOf: SmallInteger)

ifTrue: '((Fraction numerator: self denominator: aNumber) reduced]
ifFalse: [super / aNumber]

self
all inst vars

thisContext:
stack top
all temp vars
aNumber
The Inspector
The Explorer
Other Tools

- File Browser
  - *Browse, import, open files*

- Method Finder, Message Name tool
  - *Find methods by name, behaviour*

- Change Sorter
  - *Name, organize all source code changes*

- SUnit Test Runner
  - *Manage & run unit tests*
File Browser

Here is a list of packages included in this Pharo1.0beta (#10418):  
AST-damiencassou.171  
AutomaticMethodCategorizer-DF.25  
AutomaticMethodCategorizerOB-DF.1  
ImageForDevelopers-pharo-DamienCassou.189  
Installer-Core-kph.324  
NewInspector-DamienCassou.39  
Nile-All-damiencassou.144  
O2-Enhancements-DavidRoethlisberger.3  
O2-Morphic-DavidRoethlisberger.2  
O2-Standard-DavidRoethlisberger.2  
OB-Morphic-Ir.99  
OB-Refactory-Ir.159  
OB-Regexp-Ir.19  
OB-Standard-DamienCassou.429  
OCForOB-rr.2  
Ocompletion-damiencassou.33  
OmniBrowser-DamienCassou.459  
OmniBrowser2-DavidRoethlisberger.2  
Refactoring-Core-Ir.57  
Refactoring-Spelling-Ir.6  
RoelTyper-PF.74  
Shout.3.15-damiencassou.73  
ShoutWorkspace.1-tween.4
Message Name Finder

![Message Name Finder screenshot]

**match**: text

"Answer whether text matches the pattern in this string.
Matching ignores upper/lower case differences.
Where this string contains #, text may contain any character.
Where this string contains *, text may contain any sequence of characters."

```
^ self startingAt: 1 match: text startingAt: 1
```

"**'**
match: 'zort' true
'*baz' match: 'mobaz' true
'*baz' match: 'mobazo' false
"
Method Finder

Type a fragment of a selector in the top pane. Accept it.

Or, use an example to find a method in the system. Type receiver, args, and answer in the top pane with periods between the items. 3. 4. 7
Methods in ChangeSets & Versions
Preferences

If true, swaps some control- and alt-keys (making ctrl-c be copy instead of alt-c).
Cannot be true if duplicateControlAndAltKeys or duplicateAllControlAndAltKeys is true.
SUnit
Challenges of this Course

> **Mastering Smalltalk syntax**
  — Simple, but not Java-like

> **Pharo Programming Environment**
  — Requires some effort to learn at first, but worth the effort

> **Pharo Class Library**
  — Need time to learn what is there

> **Object-oriented thinking**
  — This is the hardest part!

> **Fully dynamic environment**
  — This is the most exciting part!

> **Smalltalk culture**
  — Best Practice Patterns (cf. book by Kent Beck)
What you should know!

- How does Smalltalk differ from Java or C++?
- Where are Smalltalk programs stored?
- Where are objects stored?
- What was the Dynabook?
- Is a class an object?
- What is dynamic binding?
- What is the difference between a message and a method?
Can you answer these questions?

- What ideas did Smalltalk take from Simula? From Lisp?
- Is there anything in Smalltalk which is not an object?
- What exactly is stored in the changes file?
- If objects have private state, then how can an Inspector get at that state?
- How do you create a new class?
- What is the root of the class hierarchy?
- If a class is an object, then what is its class? The class of its class? …
- If you don’t know, how would you find out?
License

http://creativecommons.org/licenses/by-sa/3.0/

Attribution-ShareAlike 3.0 Unported

You are free:
  to Share — to copy, distribute and transmit the work
  to Remix — to adapt the work

Under the following conditions:
  Attribution. You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).
  Share Alike. If you alter, transform, or build upon this work, you may distribute the resulting work only under the same, similar or a compatible license.

For any reuse or distribution, you must make clear to others the license terms of this work. The best way to do this is with a link to this web page.
Any of the above conditions can be waived if you get permission from the copyright holder.
Nothing in this license impairs or restricts the author's moral rights.