Smalltalk – Eine mini Einführung

Fediverse: ckeen@vernunftzentrum.de

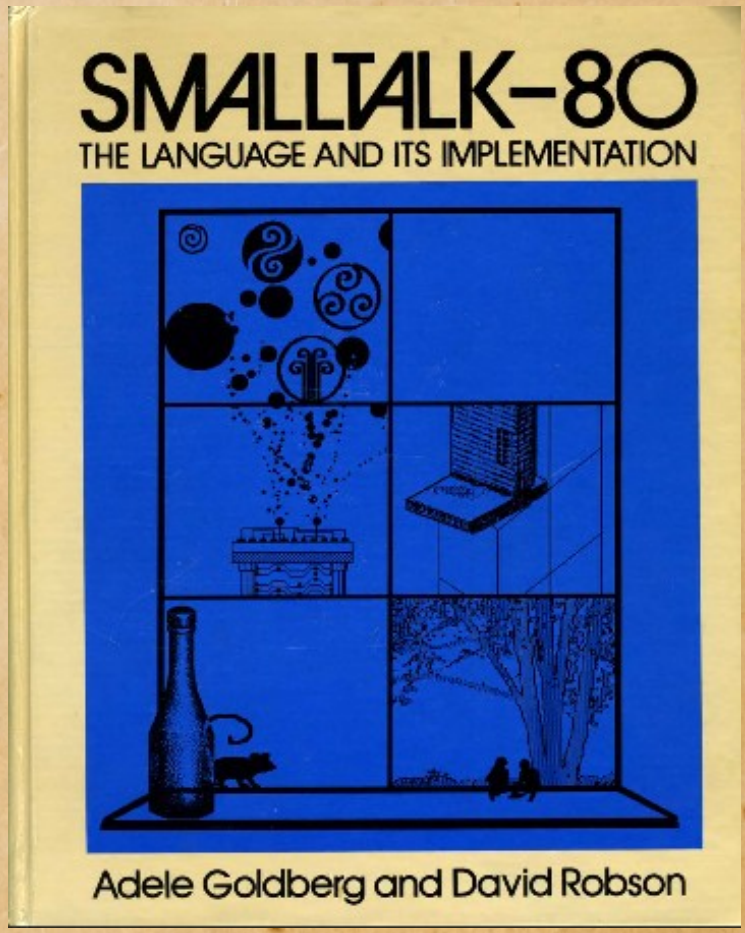
Mail: ckeen@pestilenz.org

Geschichte

- Geboren ~1970/71 aus einem Projekt am Xerox PARC
- Ziel war es, ein System zu bauen, mit denen man Kindern programmieren beibringen kann
- Damit wurde mehrere Jahre lang 10-12 jähr-igen Kindern programmieren beigebracht
- Implementiert für Hardware, die es noch nicht gab
- Später dann reimplementierung auf VM

Geschichte

- Modernes Smalltalk mit VM 1976-1980: Smalltalk-80, worauf heute die Systeme basieren
- Release 1983 mit dem “blue book”
- Die Spec wurde dann nochmal nachimplementiert um sie zu testen.
- Kommerzielle Anwendungen u.a. von IBM und Techtronix
- IBM hat sich später zu Gunsten Java entschieden. Die ersten Java VM wurde in Smalltalk geschrieben.



Implementierungen

- Kommerzielle Smalltalks: Gemstone, Dolphin (jetzt GPL), Cincom ...
- Freie Implementierungen: Squeak, Pharo, Cuis, Scratch
- Freie Sonderlinge: Gnu Smalltalk, A little smalltalk
- Freie Neuentwicklungen: newspeak, lively kernel

Implementierungen

- Windows, MacOS, Linux, *BSD, Android, Javascript, bare metal
- Freie Implementierungen benutzen mittlerweile diesselbe VM: opensmalltalk-vm
- Implementierung der VM in Javascript: SqueakJS: https://github.com/bertfreudenberg/SqueakJS

Implementierungen

- Squeak ist das älteste System
- Pharo ist ein fork, der mehr auf ‘business’ Anwendung und Modernisierung ausgelegt ist
- Cuis ein fork, der explizit Komplexität reduzieren will
- SqueakJS, naja…

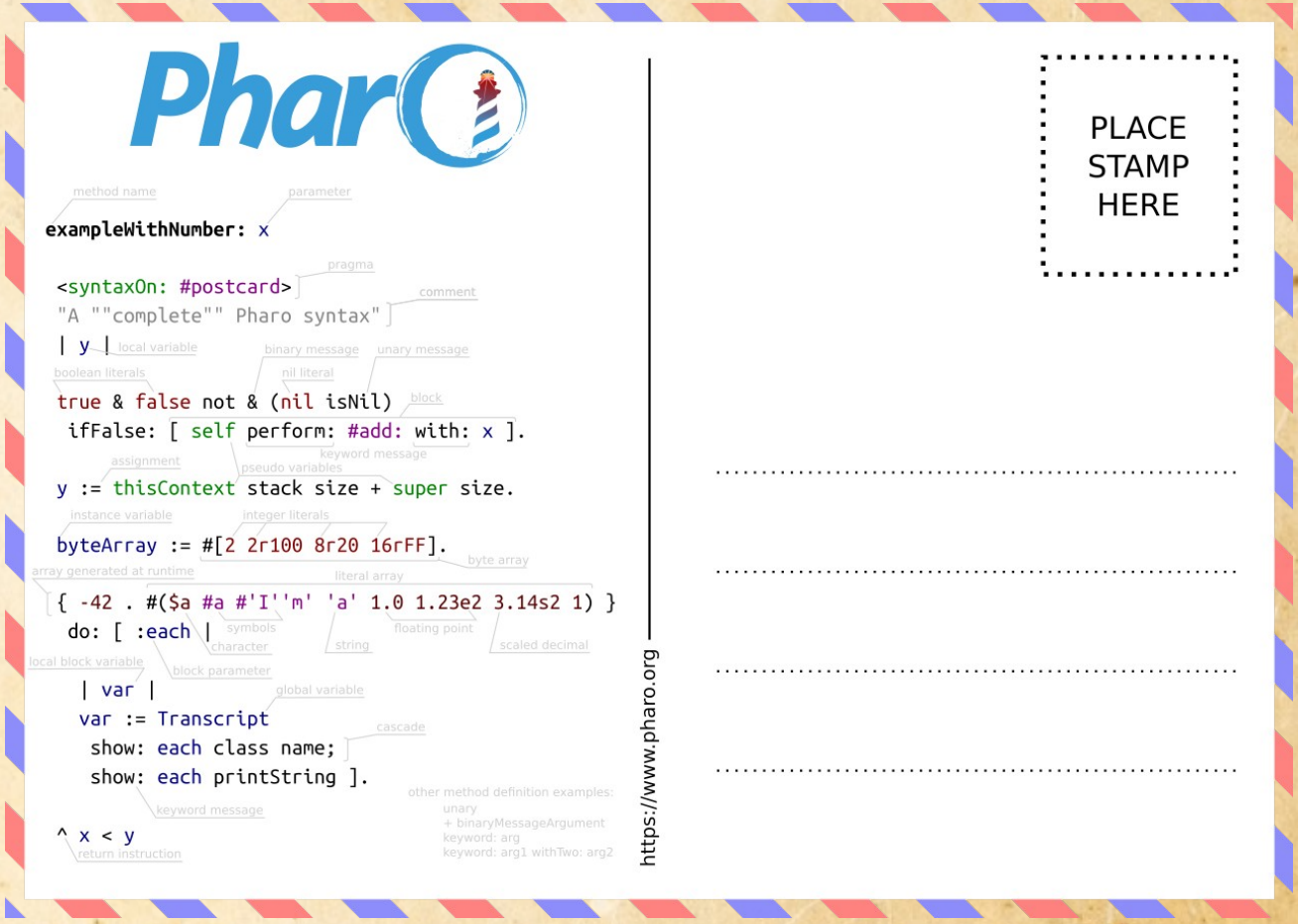
Die Virtuelle Maschine

- aktiv entwickelt, C code, compiler wird in Smalltalk generiert.
- Schnell
- Leicht portierbar
- Erweiterbar durch plugins: git, sdl, etc...

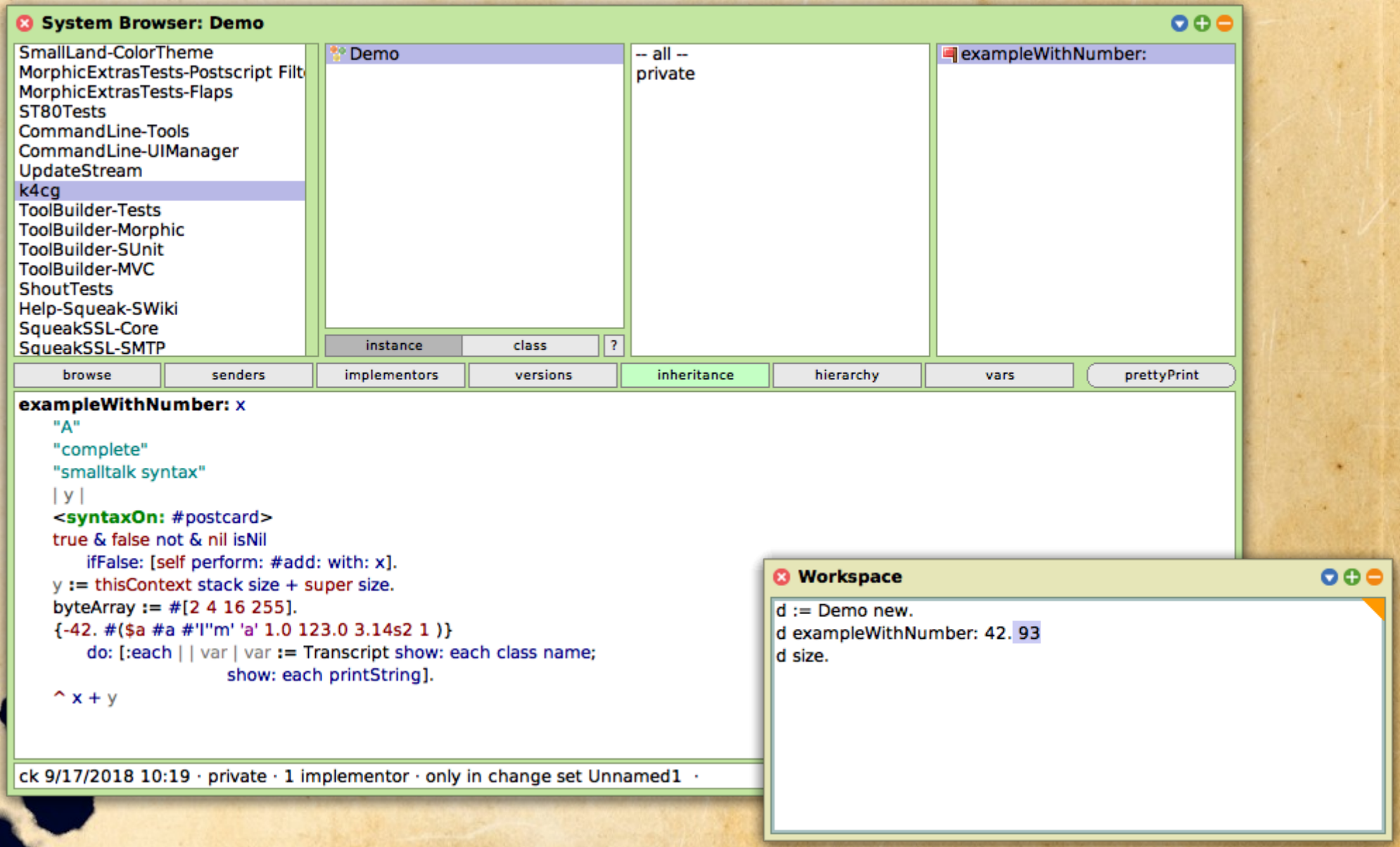
Features

- Objektorientierung: Alles ist ein Objekt, alles kommuniziert über Nachrichten
- Einfache Vererbung von Klassen
- Einfache Klassenhierarchie der Sprache
- Wenig Syntax
- Relektion von *allem möglich*

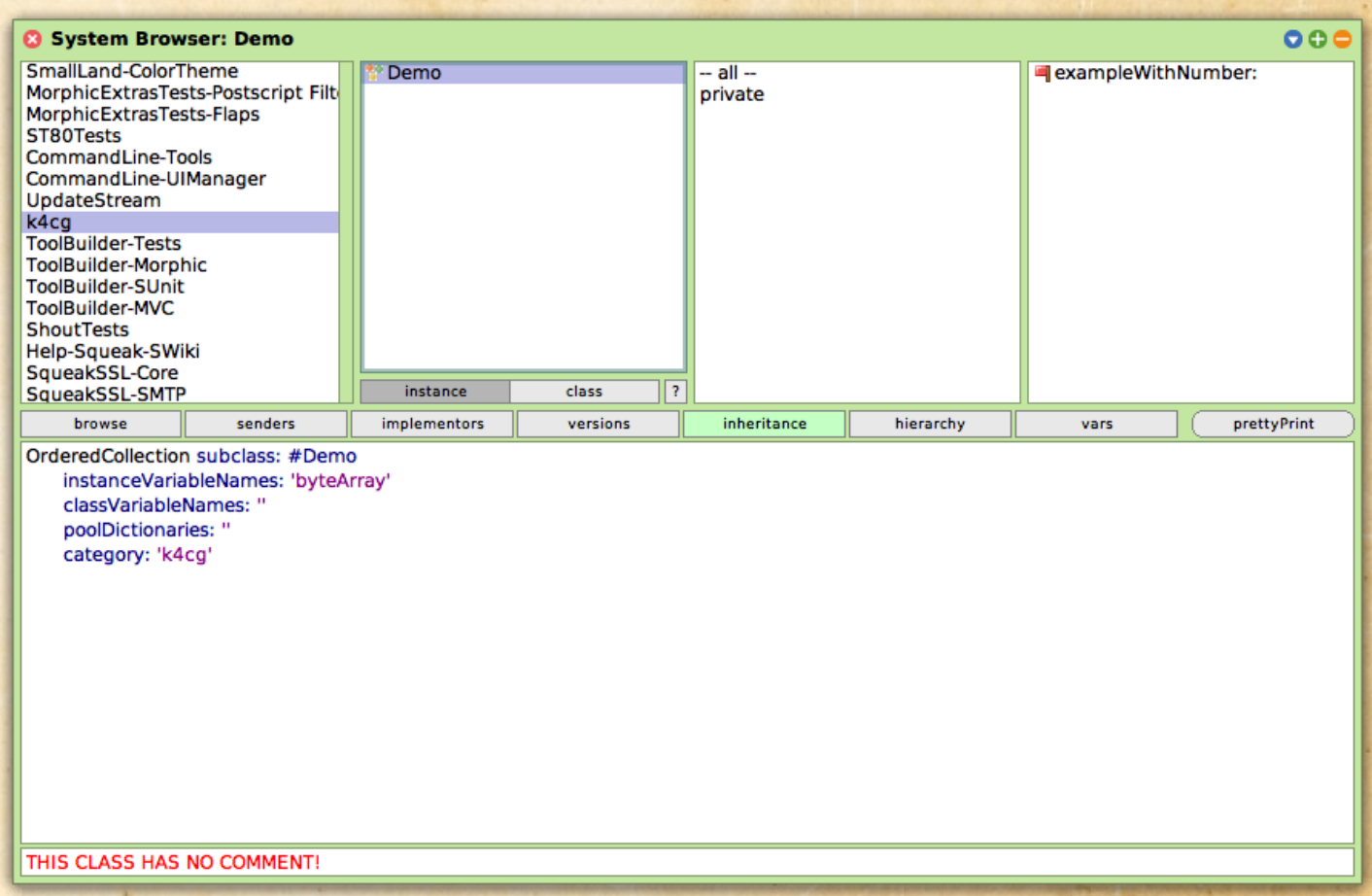
Syntax auf einer Postkarte



Syntax (im System)



Syntax


System Browser: Demo
SmallLand-ColorTheme
MorphicExtrasTests-Postscript Filt
MorphicExtrasTests-Flaps
ST80Tests
CommandLine-Tools
CommandLine-UIManager
UpdateStream
k4cg
ToolBuilder-Tests
ToolBuilder-Morphic
ToolBuilder-SUnit
ToolBuilder-MVC
ShoutTests
Help-Squeak-SWiki
SqueakSSL-Core
SqueakSSL-SMTP
Demo
instance
class
?
-- all --
private
exampleWithNumber:
browse
senders
implementors
versions
inheritance
hierarchy
vars
prettyPrint
OrderedCollection subclass: #Demo
instanceVariableNames: 'byteArray'
classVariableNames: ''
poolDictionaries: ''
category: 'k4cg'
THIS CLASS HAS NO COMMENT!

Syntax Fallstricke

- “” sind Kommentarzeichen
- Arrays sind 1-based!
- Statements werden durch einen Punkt ‘.’ getrennt
- Kaskaden durch ein semicolon ‘;’
- Messages werden in einer bestimmten Reihenfolge ausgewertet.
- Spezielle ‘Konstanten’: `true, false, nil, self, super`

Messages

- Unary messages:

  ```
  'abc' asUppercase. → 'ABC'
  ```

- Binary messages:

  ```
  5 + 2
  ```

- Keyword messages:

  ```
  #('a' 'b' 'c') at: 1 → 'a'
  ```

- Reihenfolge der Auswertung: unary → binary → keyword

  ```
  3 + 2 raisedTo: 2 squared + 7 → 48828125
  (3+2) raisedTo: ((2 squared) + 7) →  48828125
  ```

- Wenn anders gewünscht: Klammern
- Alle messages liefern ein Ergebnis!

Blöcke

- Verzweigungen, Schleifen, neue Prozesse usw. macht alles ein Objekt: Der Block
- Auch nur ein Objekt, wie jedes andere. Syntax: []

```
3 even ifFalse: [ 'Haha, ne' ].

#( 23 42 2 77) select: [:i | i even]. → #(42 2)

#( 23 42 2 77) collect: [:i | i even]. → #(false true true false)
```

Objekte

- Werden aus Klassen mit einer ‘new’ Nachricht erzeugt
- Haben eigenen state in Instanzvariablen
- Implementieren Nachrichtenempfänger (receiver) als Methode
- Alle Methoden sind public
- Um die Intention von Methoden zu erkennen, werden Methoden in einzelne Gruppen eingeteilt (“Protokolle”)

Klassen

- Klassen sind auch Objekte
- Kann eine Unterklasse einer andere Klasse sein
- Haben auch eigenen state, in Klassenvariablen
- Können eigene Methoden implementieren, um z.B. eigene Konstruktoren zu bauen oder globale Einstellungen abzufragen, ...
- ‘new’ erzeugt eine Instanz dieser Klasse

“Das System”

- Sammlung von Klassen und der Programmier Umgebung (Editor, Compiler, Debugger, Profiler, Anwendungen...)
- Größe unterscheidet sich stark von System zu System.
- Grundtypen sind allerdings alle gleich (aus dem “blue book”)

  Smalltalk80: 236

  Cuis 5: 550

  Squeak 5.1: 2261
- Die Schwierigkeit ist es, die richtige zu finden

  → method Finder

Systemkomponenten (UI)

- Fürs arbeiten, am häufigsten benutzt:

  SystemBrowser, Workspace, Transkript, Debugger

- Für Fortgeschrittene:

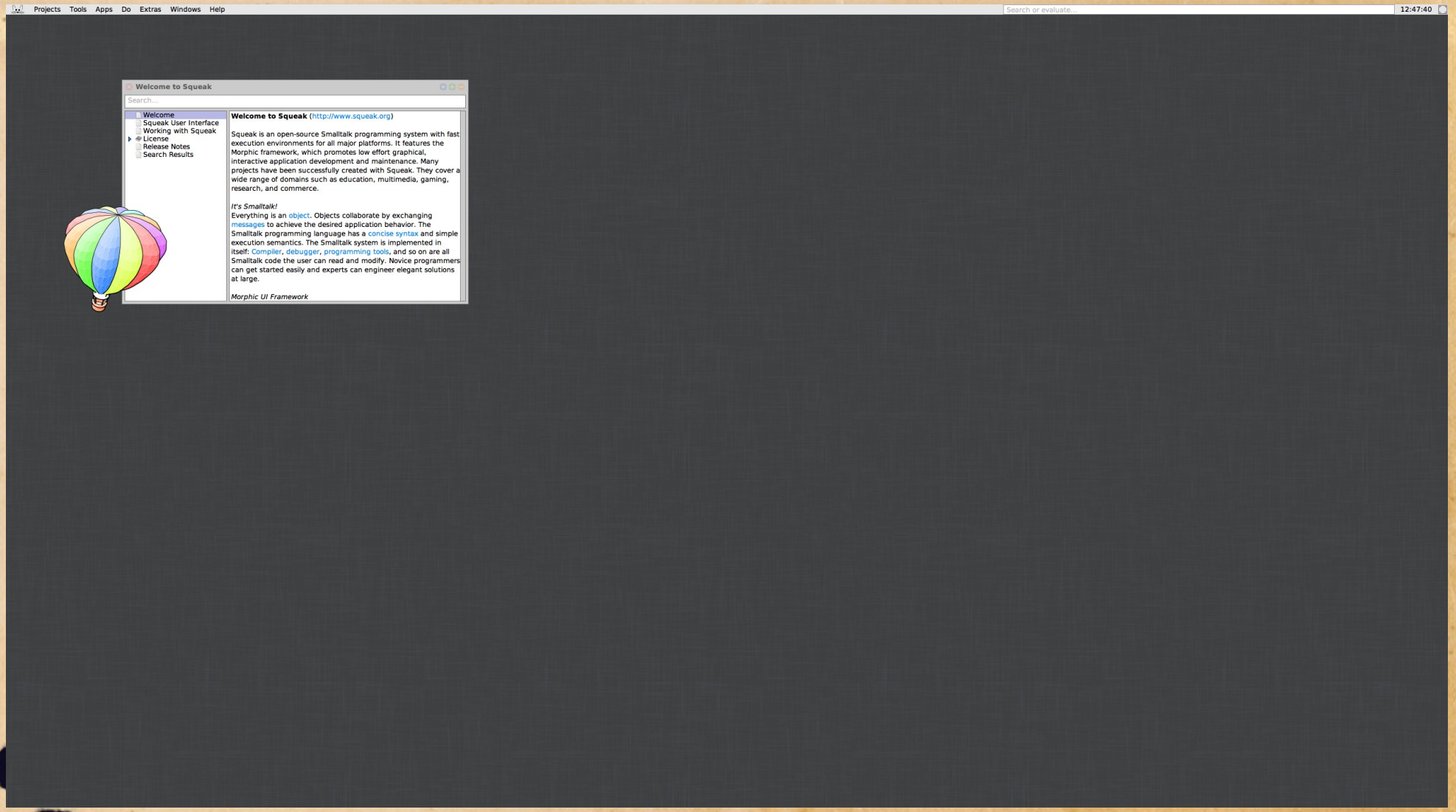
  Refactoring Browser, Source Code Verwaltung (monticello, git, changes)

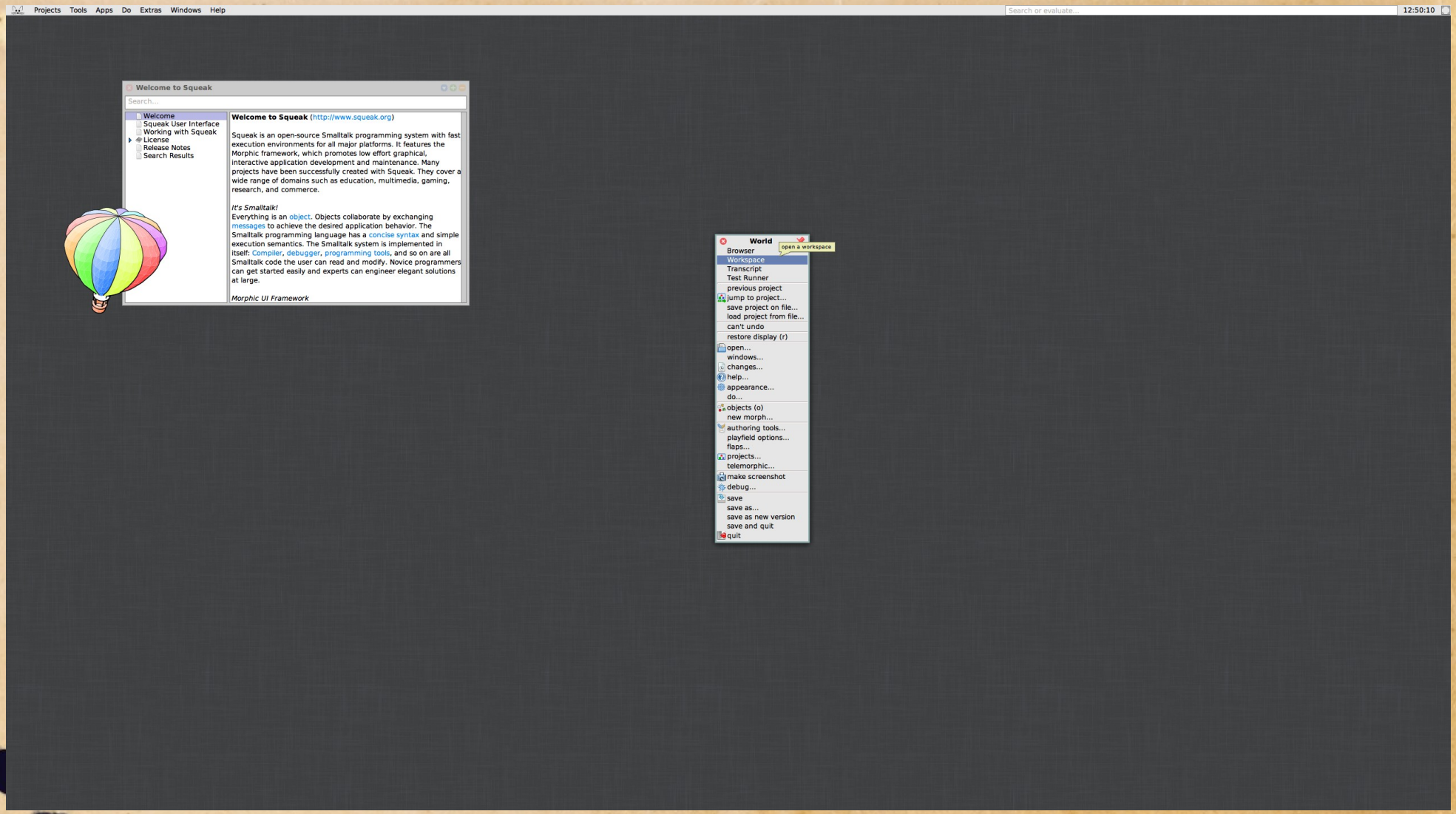
- Und dann noch der Rest:

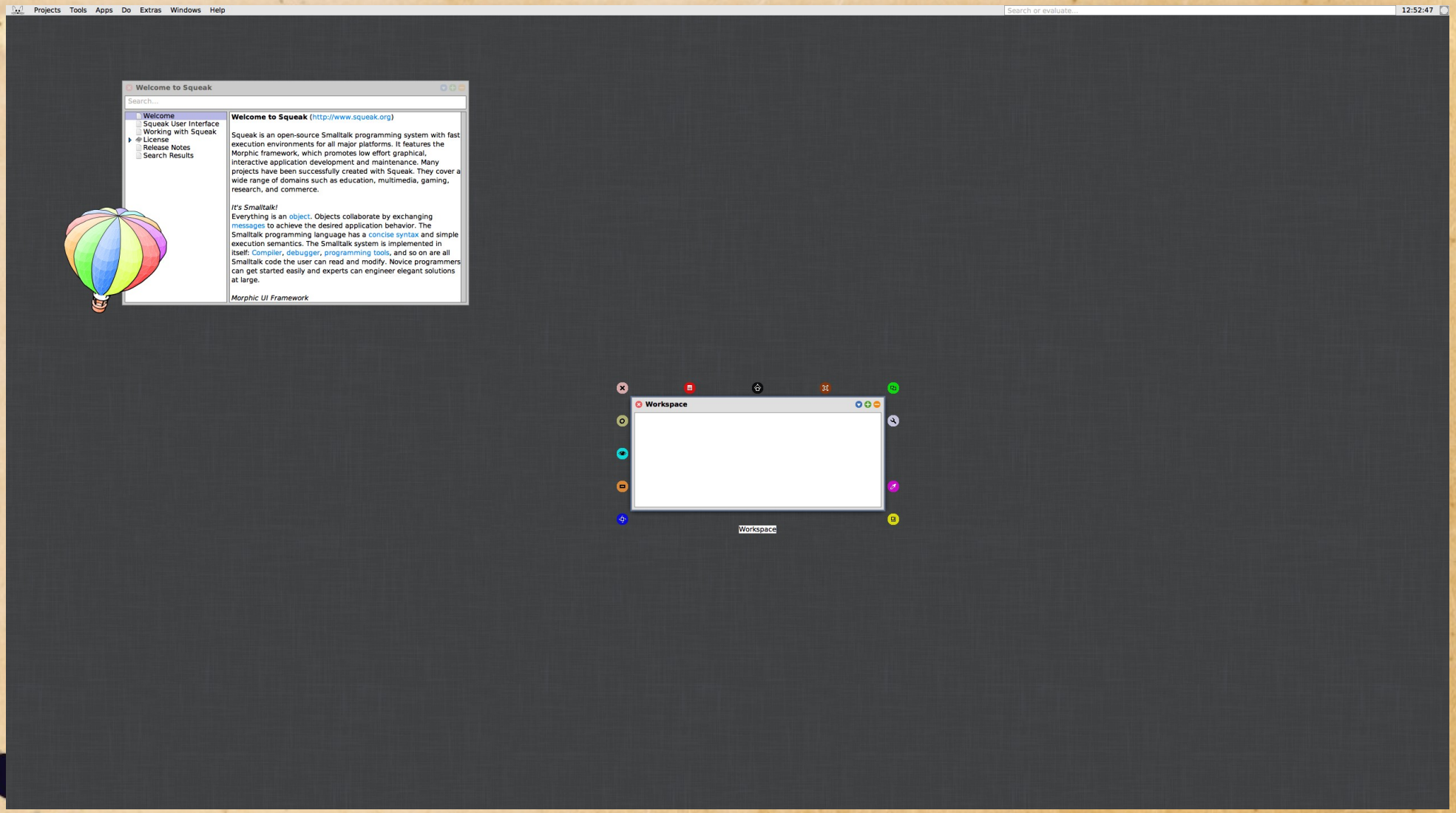
  Profiler, File list, Change sorter, ….

Benutzen des Systems

- Bedienung gedacht für eine Maus, allerdings gibt es überall short cuts für schnelles arbeiten
- Code markieren und Alt-d (doIt) evaluiert code
- Code markieren und Alt-p (printIt) gibt das Ergebnis aus
- Klasse / methode markieren und Alt-b (browseIt) öffnet den SystemBrowser für die Klasse / Methode
- Alt-i (inspect) zeigt den state einer Klasse / eines Objektes


Projects Tools Apps Do Extras Windows Help
Search or evaluate...
12:47:40
Welcome to Squeak
Search...
Welcome
Squeak User Interface
Working with Squeak
License
Release Notes
Search Results
Welcome to Squeak (http://www.squeak.org)
Squeak is an open-source Smalltalk programming system with fast execution environments for all major platforms. It features the Morphic framework, which promotes low effort graphical, interactive application development and maintenance. Many projects have been successfully created with Squeak. They cover a wide range of domains such as education, multimedia, gaming, research, and commerce.
It's Smalltalk!
Everything is an object. Objects collaborate by exchanging messages to achieve the desired application behavior. The Smalltalk programming language has a concise syntax and simple execution semantics. The Smalltalk system is implemented in itself: Compiler, debugger, programming tools, and so on are all Smalltalk code the user can read and modify. Novice programmers can get started easily and experts can engineer elegant solutions at large.
Morphic UI Framework


Projects Tools Apps Do Extras Windows Help
Search or evaluate...
12:50:10
Welcome to Squeak
Search...
Welcome
Squeak User Interface
Working with Squeak
License
Release Notes
Search Results
Welcome to Squeak (http://www.squeak.org)
Squeak is an open-source Smalltalk programming system with fast execution environments for all major platforms. It features the Morphic framework, which promotes low effort graphical, interactive application development and maintenance. Many projects have been successfully created with Squeak. They cover a wide range of domains such as education, multimedia, gaming, research, and commerce.
It's Smalltalk!
Everything is an object. Objects collaborate by exchanging messages to achieve the desired application behavior. The Smalltalk programming language has a concise syntax and simple execution semantics. The Smalltalk system is implemented in itself: Compiler, debugger, programming tools, and so on are all Smalltalk code the user can read and modify. Novice programmers can get started easily and experts can engineer elegant solutions at large.
Morphic UI Framework
World
open a workspace
Browser
Workspace
Transcript
Test Runner
previous project
jump to project...
save project on file...
load project from file...
can't undo
restore display (r)
open...
windows...
changes...
help...
appearance...
do...
objects (o)
new morph...
authoring tools...
playfield options...
flaps...
projects...
telemorphic...
make screenshot
debug...
save
save as...
save as new version
save and quit
quit


Projects Tools Apps Do Extras Windows Help
Search or evaluate...
12:52:47
Welcome to Squeak
Search...
Welcome
Squeak User Interface
Working with Squeak
License
Release Notes
Search Results
Welcome to Squeak (http://www.squeak.org)
Squeak is an open-source Smalltalk programming system with fast execution environments for all major platforms. It features the Morphic framework, which promotes low effort graphical, interactive application development and maintenance. Many projects have been successfully created with Squeak. They cover a wide range of domains such as education, multimedia, gaming, research, and commerce.
It's Smalltalk!
Everything is an object. Objects collaborate by exchanging messages to achieve the desired application behavior. The Smalltalk programming language has a concise syntax and simple execution semantics. The Smalltalk system is implemented in itself: Compiler, debugger, programming tools, and so on are all Smalltalk code the user can read and modify. Novice programmers can get started easily and experts can engineer elegant solutions at large.
Morphic UI Framework
Workspace
Workspace

Hello World

- Schreibt mal eine

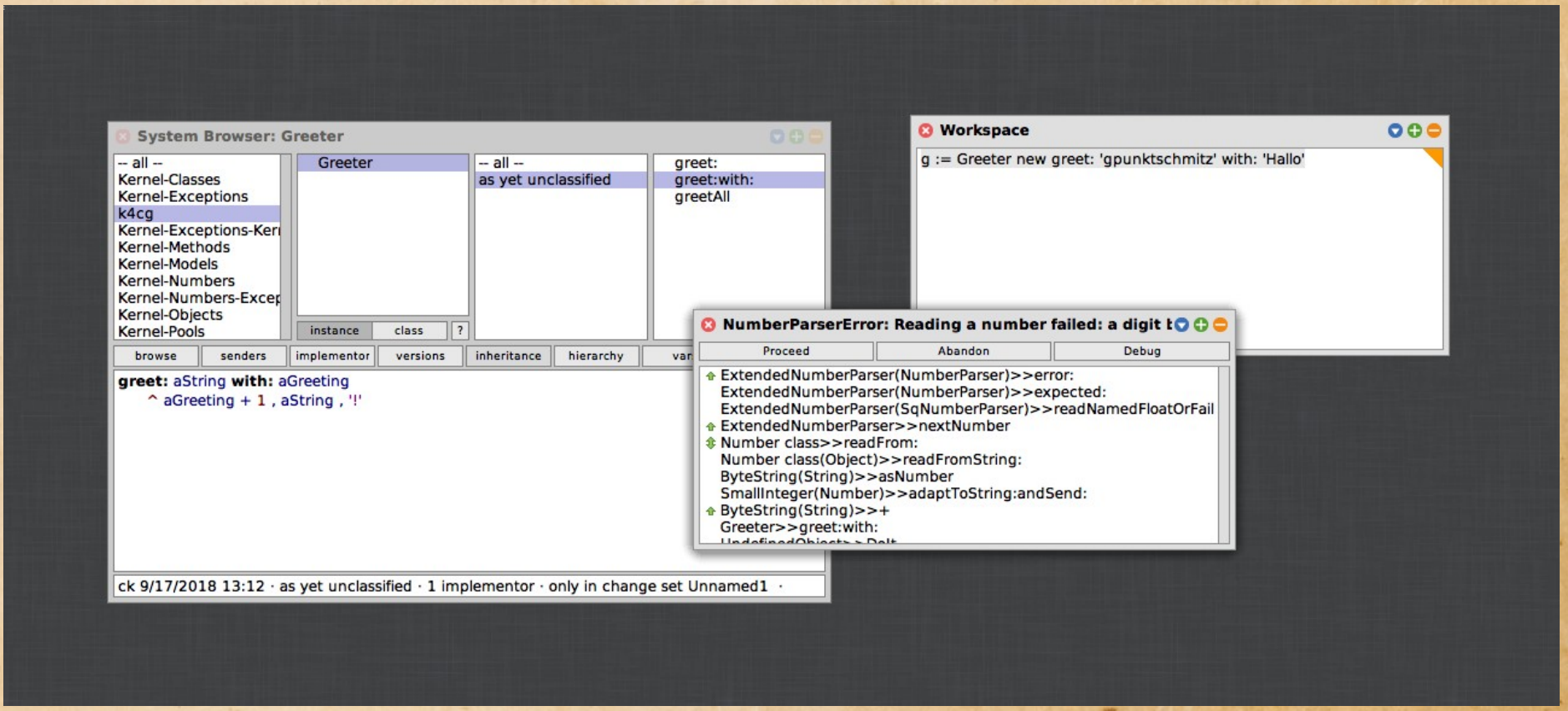
  Greeter Klasse

```
Greeter>>greetAll → ‘Ciao a tutti!’

Greeter>>greet: aName → ‘Hi aName!’

Greeter>>greet: aName with:
aGreeting → ‘aString name’
```

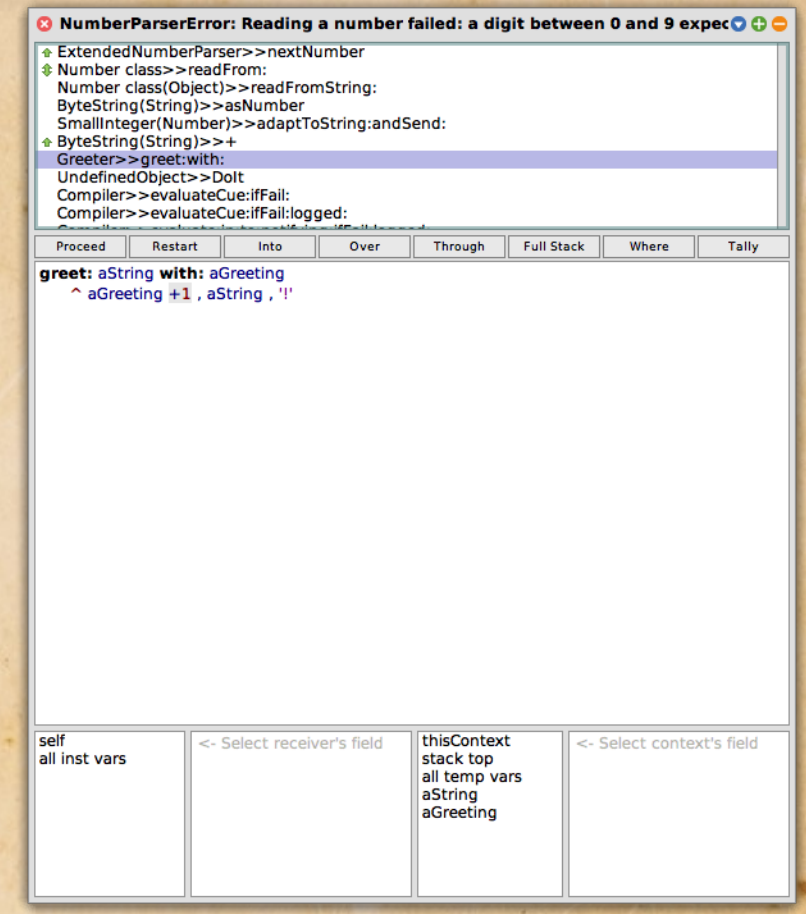
Debugger

- Kann immer noch einen Versuch starten
- Man kann hier seinen code ändern!
- Breakpoints kann man jederzeit selbst einfügen mit ‘self halt’.
- Alles kann (muss!) man debuggen!

Debugger


System Browser: Greeter
-- all --
Kernel-Classes
Kernel-Exceptions
k4cg
Kernel-Exceptions-Kerı
Kernel-Methods
Kernel-Models
Kernel-Numbers
Kernel-Numbers-Excep
Kernel-Objects
Kernel-Pools
Greeter
instance
class
?
-- all --
as yet unclassified
greet:
greet:with:
greetAll
browse
senders
implementor
versions
inheritance
hierarchy
greet: aString with: aGreeting
^ aGreeting + 1 , aString , '!'
ck 9/17/2018 13:12 · as yet unclassified · 1 implementor · only in change set Unnamed1 ·
Workspace
g := Greeter new greet: 'gpunktschmitz' with: 'Hallo'
NumberParserError: Reading a number failed: a digit t
Proceed
Abandon
Debug
ExtendedNumberParser(NumberParser)>>error:
ExtendedNumberParser(NumberParser)>>expected:
ExtendedNumberParser(SqNumberParser)>>readNamedFloatOrFail
ExtendedNumberParser>>nextNumber
Number class>>readFrom:
Number class(Object)>>readFromString:
ByteString(String)>>asNumber
SmallInteger(Number)>>adaptToString:andSend:
ByteString(String)>>+
Greeter>>greet:with:

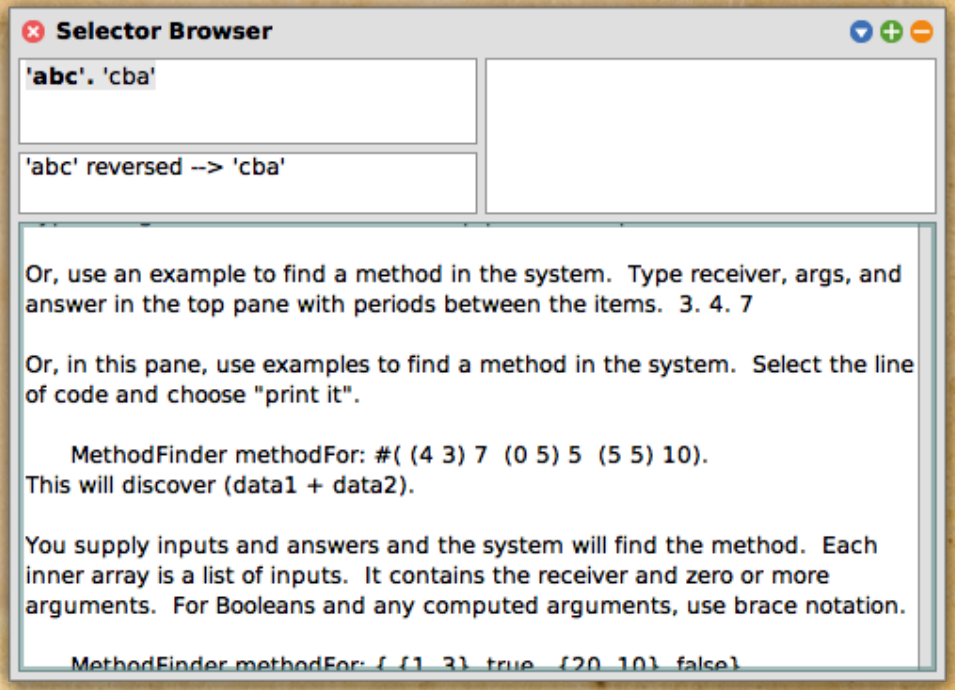
Debugger


NumberParserError: Reading a number failed: a digit between 0 and 9 expec
ExtendedNumberParser>>nextNumber
Number class>>readFrom:
Number class(Object)>>readFromString:
ByteString(String)>>asNumber
SmallInteger(Number)>>adaptToString:andSend:
ByteString(String)>>+
Greeter>>greet:with:
UndefinedObject>>DoIt
Compiler>>evaluateCue:ifFail:
Compiler>>evaluateCue:ifFail:logged:
Proceed
Restart
Into
Over
Through
Full Stack
Where
Tally
greet: aString with: aGreeting
^ aGreeting +1 , aString , '!'
self
all inst vars
<- Select receiver's field
thisContext
stack top
all temp vars
aString
aGreeting
<- Select context's field

Aufgabe

- Schreibe eine Klasse ReverseShouter, mit Methode ‘shout’, die einen String entgegennimmt und ihn verkehrt herum und ALL CAPS wieder ausgibt.
- Tipp:

  Tools → Method Finder

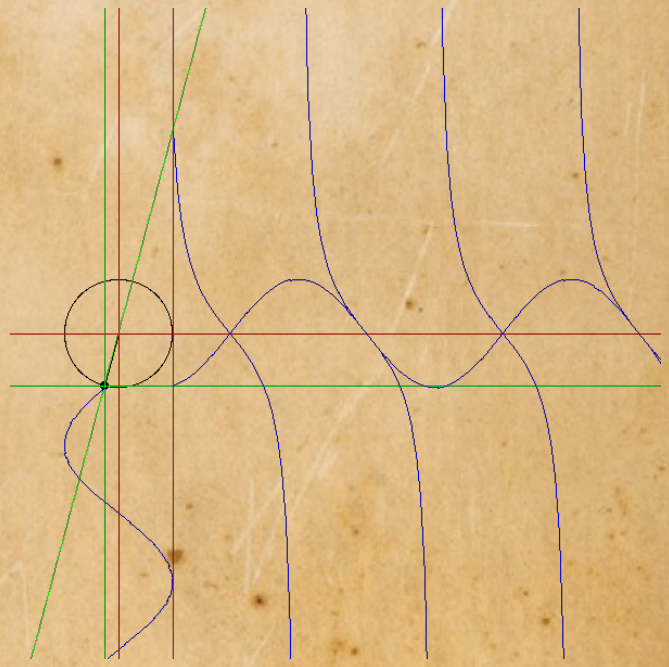


Grafische UI: Morphic

- Beispiel: https://gist.githubusercontent.com/tonyg/370749/raw/d0a3f0a12505c326fd130911a9ae8d02244b511f/SinCosTanMorph.st

```
(HTTPClient httpGet: 'http://pestilenz.org/~ckeen/SinCosTanMorph.st') fileIn
```

- Beispiel für einfachen Morph
- Die System Morphs sind komplexer, wegen der Vielfalt an Protokollen...
- Protokolle selten dokumentiert, sourcen lesen ist Pflicht... :-/



Implementierungen

- Squeak: https://squeak.org
- Pharo: https://pharo.org
- Cuis: https://cuis-smalltalk.org
- Die ganze Liste an Implementierungen:

  https://en.wikipedia.org/wiki/Smalltalk#List_of_implementations

Weiterführende Literatur

- Die meisten Bücher sind online verfügbar:

  https://open.umn.edu/opentextbooks/textbooks/squeak-by-example

  https://squeak.org/documentation/

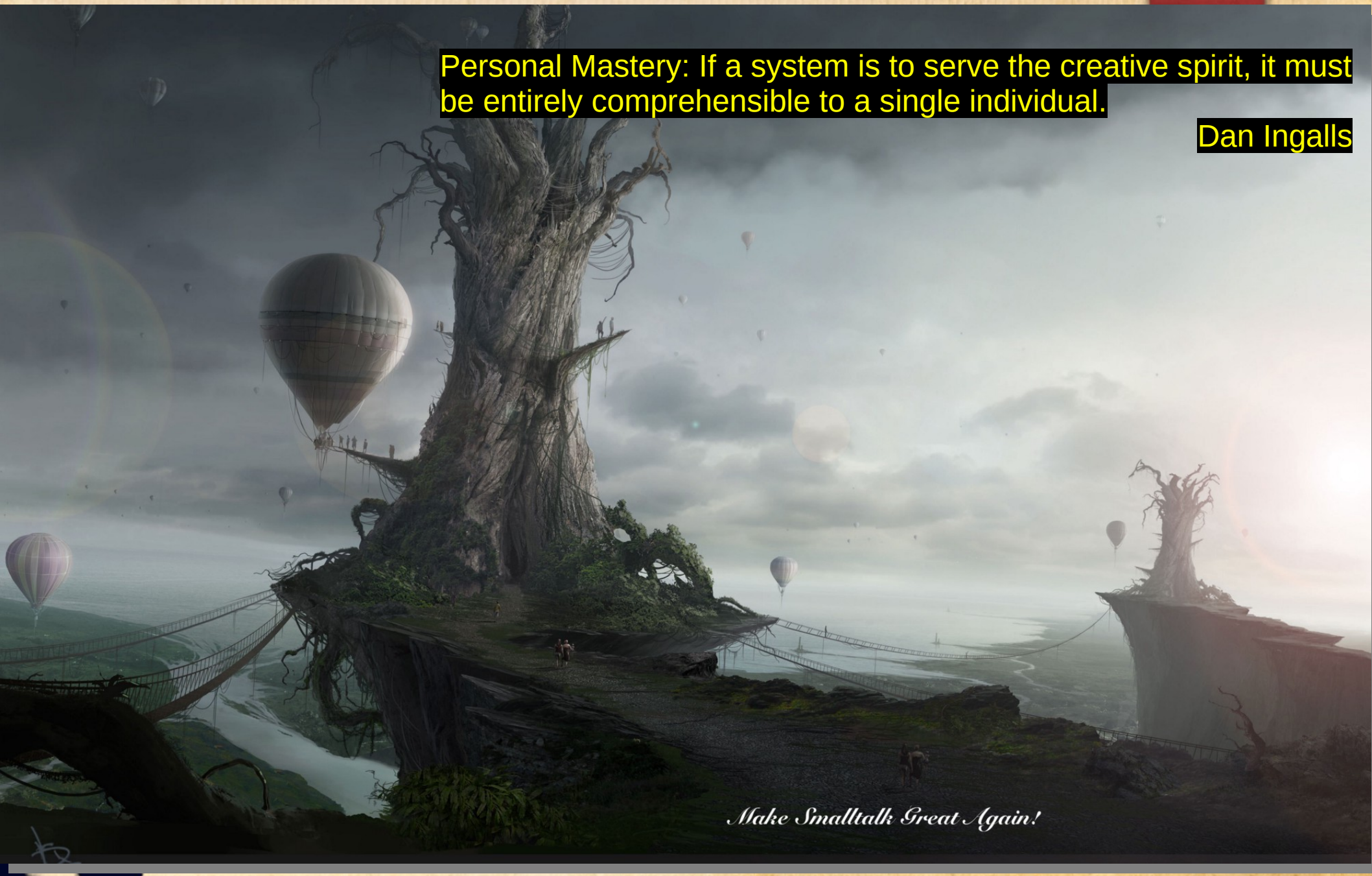
  http://stephane.ducasse.free.fr/FreeBooks.html

- Für Pharo gibt es einen kostenlosen online Kurs:

  https://mooc.pharo.org/

  Start 8. Oktober!

- Squeak mailing lists, slack:
  https://squeak.org/community/


Personal Mastery: If a system is to serve the creative spirit, it must be entirely comprehensible to a single individual.
Dan Ingalls
Make Smalltalk Great Again!