

HUMBOLDT-UNIVERSITÄT ZU BERLIN

MENSCH-TECHNIK-INTERAKTION



ARBEITSGRUPPE
SOFTWARETECHNIK
(INSTITUT FÜR INFORMATIK)



ARBEITSGRUPPE
INGENEURPSYCHOLOGIE
(INSTITUT FÜR PSYCHOLOGIE)

Smalltalk, Squeak & You

Dipl.-Inf. Michael Hildebrandt

Dipl.-Inf. Nicolas Niestroj

Gliederung

- » 1. Einführung Smalltalk
 - » **1.1 Grundlegende Eigenschaften**
 - » 1.2 syntaktische Besonderheiten
- » 2. Einführung Squeak
 - » 2.1 Tools in Squeak
 - » System Browser
 - » Workspace
 - » Transcript
 - » Message Names
 - » Debugger
 - » File List
 - » 2.2 ATEO Developer Tools
 - » ATEO Menu

Grundlegende Eigenschaften

- » Smalltalk 80
 - » streng objektorientiert
 - » Virtuelle Maschine
 - » garbage collector
 - » Dynamische Typisierung
 - » alle Klassen von Smalltalk zugänglich und einsehbar
 - » Geschriebener Code sofort ausführbar (just in time compiler) → „lebendige Welt“

Smalltalk Regeln

- » 1. Alles ist ein Objekt
- » 2. Jedes Objekt ist eine Instanz einer Klasse
- » 3. Alles passiert via Nachrichten
- » 4. Jede Klasse besitzt eine Super-Klasse
- » 5. Methoden-Lookup folgt der Vererbungskette

Smalltalk: streng objektorientiert – Regel 1: Alles in Smalltalk ist ein Objekt

- » Regel 1: Alles in Smalltalk ist ein Objekt
- » Beispiele
 - » Die Zahl „1“: Objekt der Klasse Integer
 - » Die Klasse „Object“ ist ein Objekt der Metaklasse „Object“

Smalltalk: streng objektorientiert - Regel 2: Jedes Objekt ist eine Instanz einer Klasse

- » Regel 2: Jedes Objekt ist eine Instanz einer Klasse
 - » Instanz wird beschrieben durch eine Klasse
 - » Instanzvariablen sind immer private
 - » Klassenvariablen
 - » Methoden (immer public)

<code>1 class</code>	<code>⇒ SmallInteger</code>
<code>,hello' class</code>	<code>⇒ ByteString</code>
<code>#(1 2 3) class</code>	<code>⇒ Array</code>
<code>(4@5) class</code>	<code>⇒ Point</code>

Smalltalk: streng objektorientiert – Variablen und Methoden

» Instanzvariablen vs. Klasseninstanzvariablen

- » Sichtbarkeit auf Instanz bzw. Klasseninstanz beschränkt

» Methoden vs. Klassenmethoden

- » Methoden werden an Objekten gerufen
- » Klassenmethoden am Klassenobjekt
 - » Utility-Methoden

```
(Float pi/2) sin.
```

- » erstellen bestimmter Instanzen

```
Color red. anstatt Color r:1 g:0 b:0.
```

Smalltalk: streng objektorientiert – Regel 3: Alles passiert via Nachrichten

- » Nachricht = Methodenruf
- » Empfänger = Objekt an dem die Methode gerufen wird
- » nahezu alles wird über **Nachrichten** realisiert
 - » Schleifen, Verzweigungen, Operatoren

3 + 4

Sende Nachricht + mit Argument 4 an
Objekt 3

- » Ausnahmen
 - » Variablendeklarationen, Zuweisungen, Rückgaben, Primitiven

Smalltalk: streng objektorientiert – Nachrichtenarten

» Unäre Nachrichten

- » Involvieren nur ein Objekt: den Empfänger
- » Beispiel:

Float pi ⇒ 3.141592653589793

» Binäre Nachrichten

- » Involvieren zwei Objekte: Empfänger und Argument
- » Selektor besteht aus:
+, -, *, /, &, =, >, |, <, ~, @
- » Beispiel:

4 + 5 * 2 ⇒ 18

» **Keyword-Nachrichten**

- » Involvieren zwei und mehr Objekte: Empfänger und Argumente
- » Selektor besteht aus einem oder mehr Schlüsselwörtern, welche mit einem : enden
- » Beispiel

```
Color r:1 g:0 b:0 ⇒ die Farbe rot  
1 to: 10           ⇒ Intervall [1,10]
```

Smalltalk: streng objektorientiert – Auswertungsreihenfolge

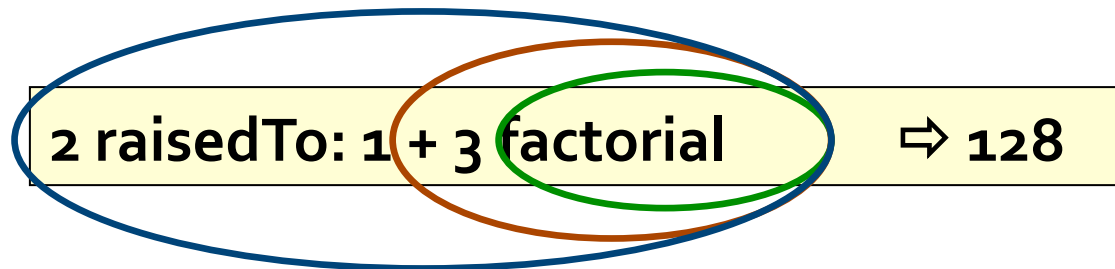
» Reihenfolge

- » Unäre Nachrichten
- » Binäre Nachrichten
- » Keyword Nachrichten

» **Klammern** werden immer zuerst gesendet

» Nachrichten des selben Typs werden **links nach rechts** ausgewertet

» **Beispiel:**



Gliederung

- » 1. Einführung Smalltalk
 - » 1.1 Grundlegende Eigenschaften
 - » **1.2 syntaktische Besonderheiten**
- » 2. Einführung Squeak
 - » 2.1 Tools in Squeak
 - » System Browser
 - » Workspace
 - » Transcript
 - » Message Names
 - » Debugger
 - » File List
 - » 2.2 ATEO Developer Tools
 - » ATEO Menu

Smalltalk: Syntaktische Besonderheiten: Blöcke

- » **Blöcke** sind Objekte, welche „aufgeschobene“ Anweisungen enthalten
- » Blöcke werden vor allem in Kontrollkonstrukten benutzt
- » **Syntax**
[Anweisung1. Anweisung2 ... AnweisungN]
[:par | Anweisung1 ... AnweisungN]

Smalltalk: Syntaktische Besonderheiten: Blöcke (cont.)

» Beispiele:

```
x := Dictionary new.  
x at: #key1 put: 10; at: #key2 put: 20.  
x do: [:value | Transcript show: value.].  
x keysDo: [:key | Transcript show: key.].
```

» **Weitere Beispiele** unter <http://wiki.squeak.org/squeak/5699>

Smalltalk: Syntaktische Besonderheiten: If-Then-Else

» Java

```
if (x < 10) {  
    a := 200;  
}
```

```
else {  
    a := 0;  
}
```

» Smalltalk

```
(x < 10) ifTrue: [  
    a := 200.  
]
```

```
ifFalse: [  
    a := 0.  
].
```

Smalltalk: Syntaktische Besonderheiten: If-Then-Else (cont.)

- » ifTrue: und ifFalse → Nachrichten
- » Als Parameter wird ein Block (Menge von Nachrichten) erwartet

Smalltalk: Syntaktische Besonderheiten: While-Loop

» Java

```
while (x < 10) {  
    a := a + 1;  
}
```

```
while (!(x < 10)) {  
    a := a + 1;  
}
```

» Smalltalk

```
[x < 10] whileTrue: [  
    a := a + 1.  
]
```

```
[x < 10] whileFalse:[  
    a := a + 1.  
]
```

Smalltalk: Syntaktische Besonderheiten: weitere Schleifen

» **n-Mal – Schleife**

n timesRepeat: [a := a+1.].

» **for – Schleife** (Inkrement = 1)

m to: n do: [:i | a := i+1.].

» **for – Schleife** (Inkrement variabel)

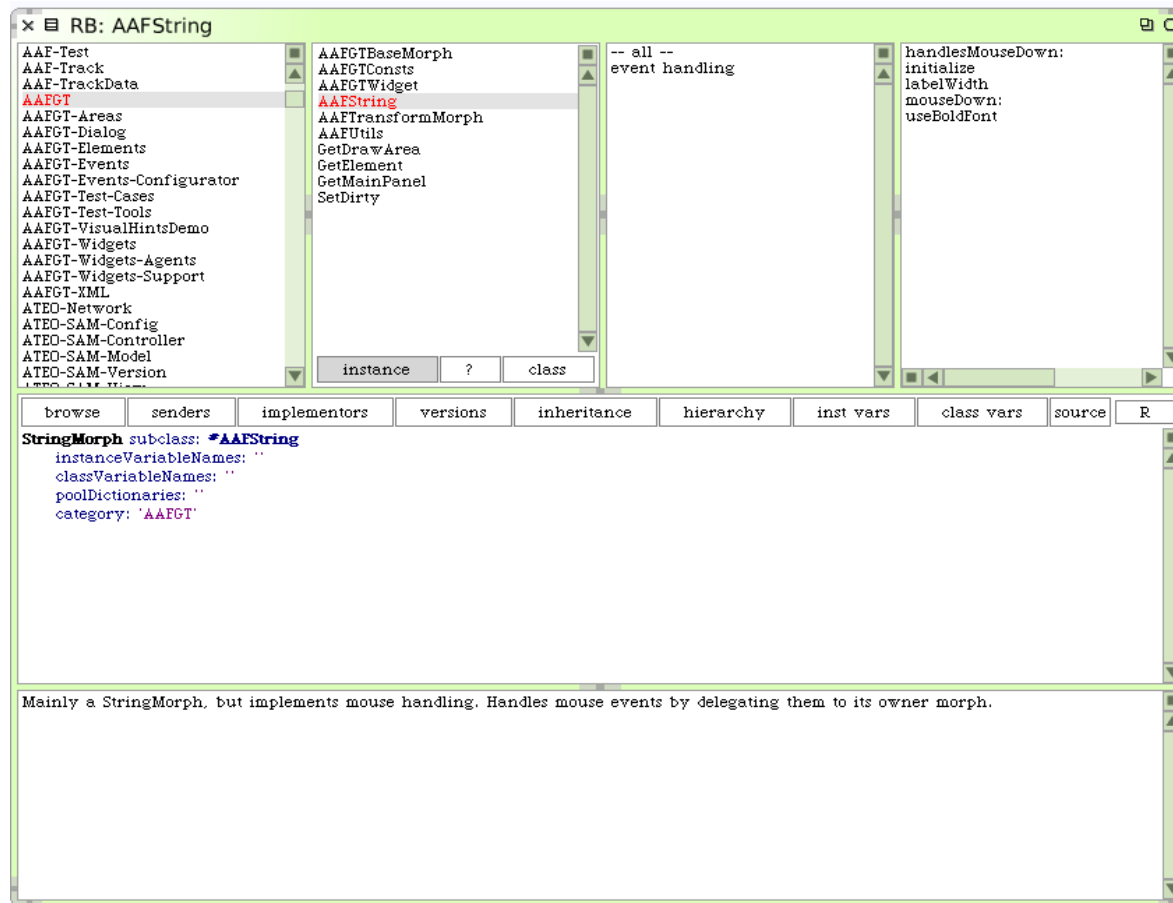
m to: n by: incr do: [:i | a := i+1.].

Gliederung

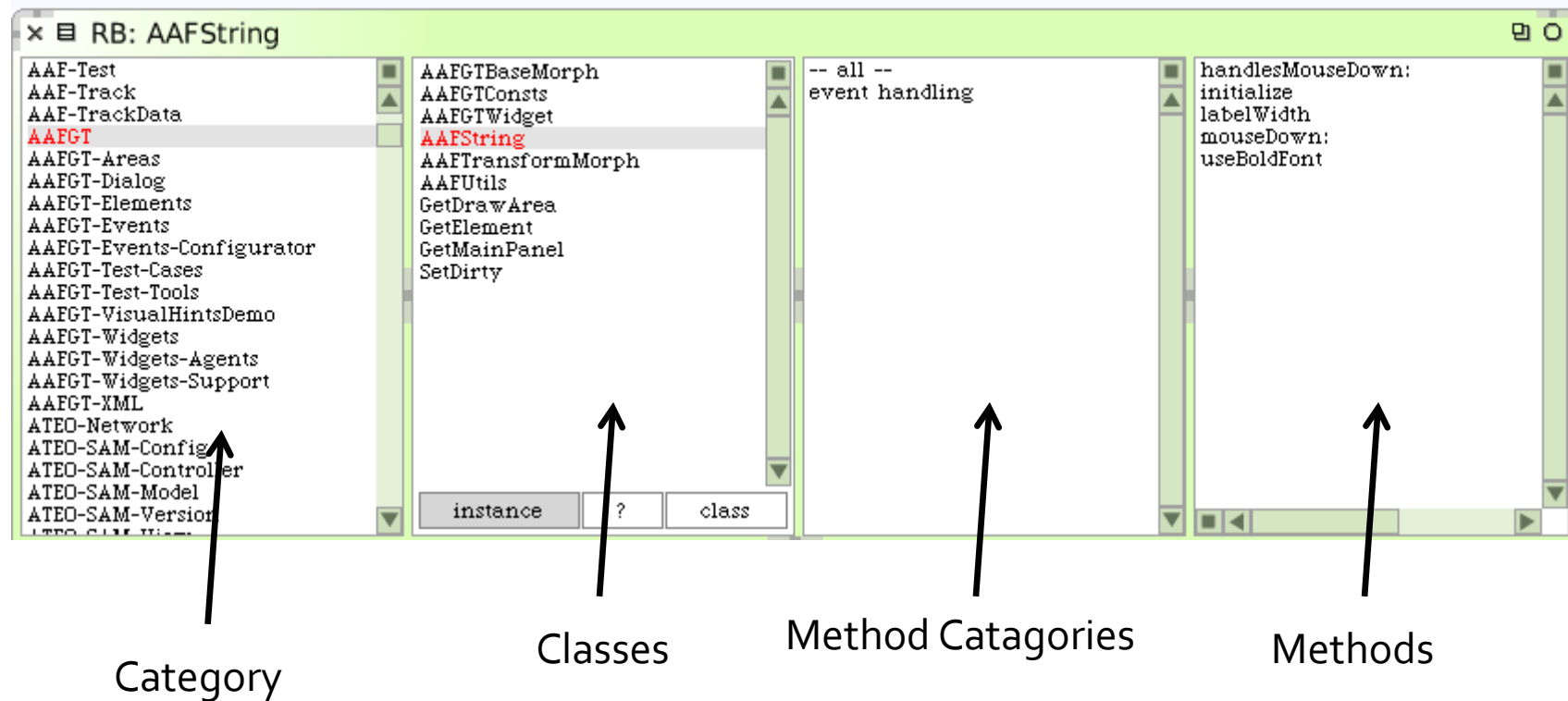
- » 1. Einführung Smalltalk
 - » 1.1 Grundlegende Eigenschaften
 - » 1.2 syntaktische Besonderheiten
- » 2. Einführung Squeak
 - » **2.1 Tools in Squeak**
 - » System Browser
 - » Workspace
 - » Transcript
 - » Message Names
 - » Debugger
 - » File List
 - » 2.2 ATEO Developer Tools
 - » ATEO Menu

Tools in Squeak: Der Systembrowser

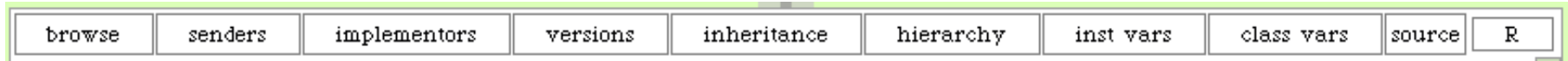
» Browser für alle Klassen in Smalltalk



Tools in Squeak: Der Systembrowser (cont.)

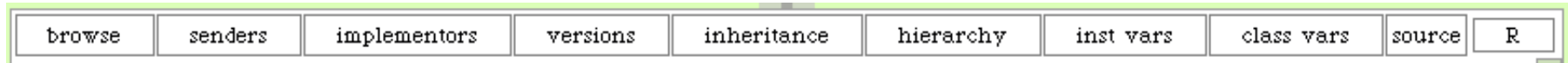


Tools in Squeak: Der Systembrowser (cont.)



- » **browse**
 - » neuer Systembrowser
- » **senders**
 - » welche Klasse sendet die Nachricht
- » **implementors**
 - » von welchen Klassen wird die Methode implementiert
- » **versions**
 - » Vergleich der aktuellen mit alten Versionen einer Methode
- » **inheritance**
 - » Vererbungsbaum der Methoden
- » **hierarchy**
 - » Vererbungsbaum der Klassen

Tools in Squeak: Der Systembrowser (cont.)



- » **inst vars**
 - » Referenzen auf Instanzvariablen
- » **class vars**
 - » Referenzen auf Klassenvariablen
- » **source**
 - » Formatierung des Code Fensters
- » **R**
 - » Funktionen für das Refactoring

Tools in Squeak: Der Systembrowser (cont.)

The screenshot displays the Squeak System Browser interface. At the top, there is a menu bar with buttons for 'browse', 'senders', 'implementors', 'versions', 'inheritance', 'hierarchy', 'inst vars', 'class vars', 'source', and 'R'. The main window is divided into two sections. The top section, labeled 'Code', contains the following Smalltalk code for the `StringMorph` class:

```
StringMorph subclass: #AAFString
instanceVariableNames: ''
classVariableNames: ''
poolDictionaries: ''
category: 'AAFMT'
```

The bottom section, labeled 'Kommentar', contains the following comment text:

```
Mainly a StringMorph, but implements mouse handling. Handles mouse events by delegating them to its owner morph.
```

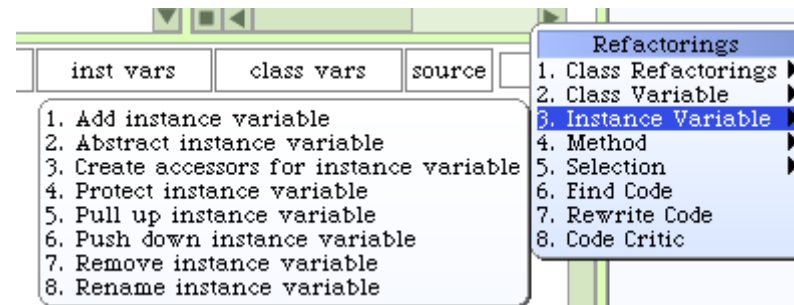
Arrows point from the labels 'Code' and 'Kommentar' to their respective sections in the browser window.

Tools in Squeak: Der Systembrowser (cont.)

» Syntaxhighlighting

» Autovervollständigung

» Refactoring

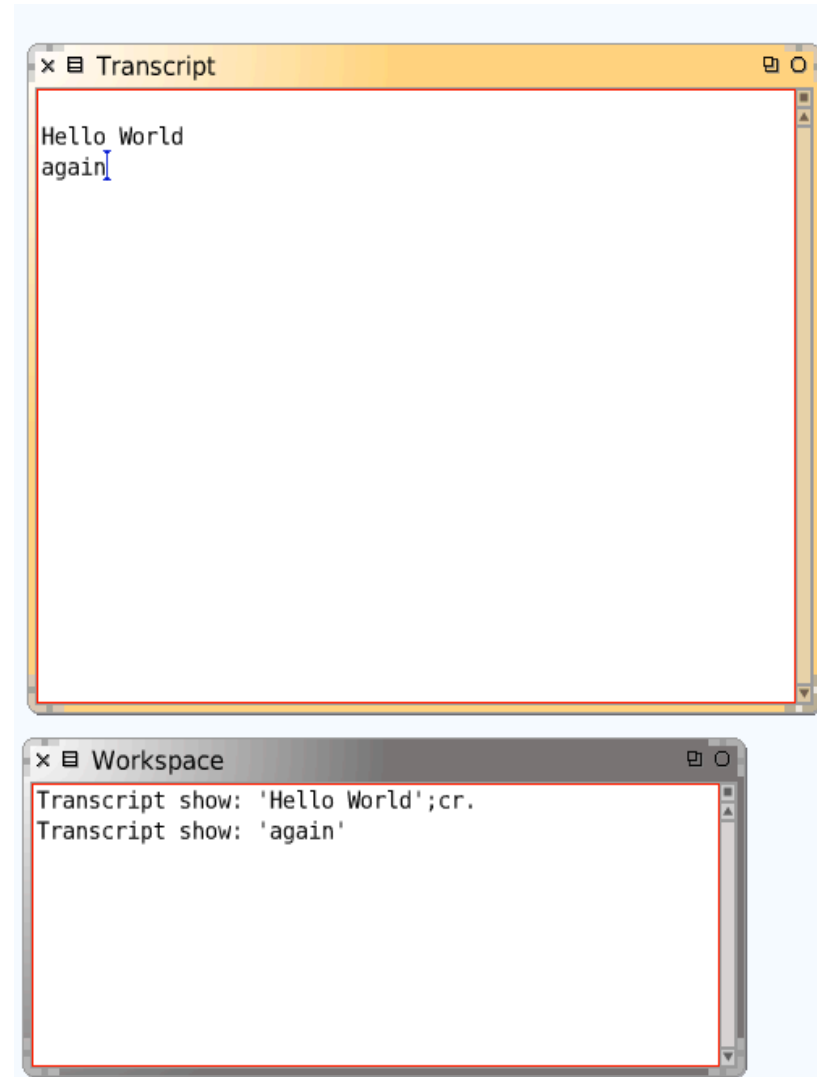


Tools in Squeak: Workspace

- » Vielseitiges Werkzeug
 - » Notizblock
 - » Testentwicklungsumgebung
 - » Quelltext programmieren und testen
 - » Wirkungsweise von Klassen und Methoden ausprobieren
 - » inspect oder explore von Objekten (strg+i oder strg+e bei markierten Objekten)
 - » strg+shift+n → Referenzen auf globale Variablen
 - » Merktzettel für Codeschnipsel
 - » Dokumentation

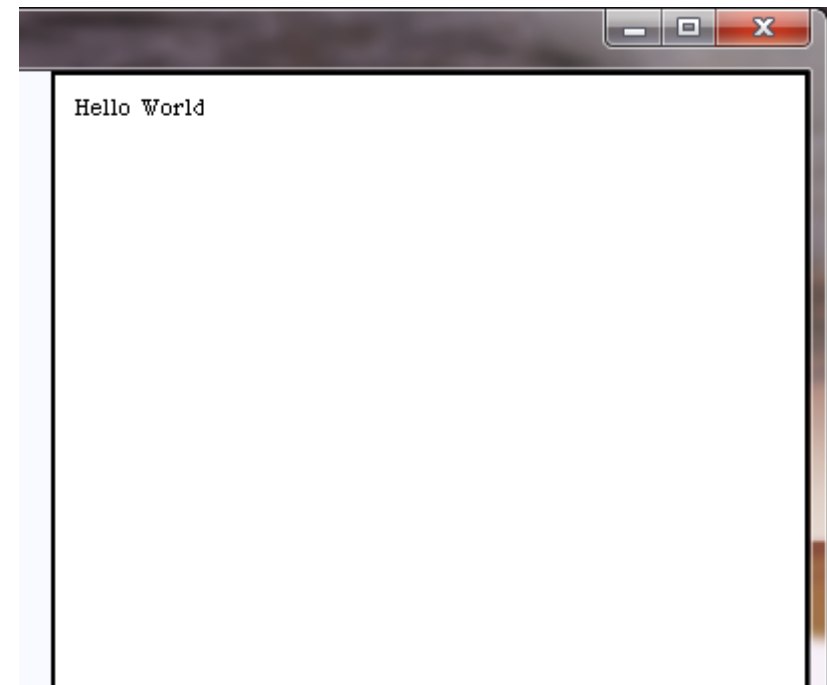
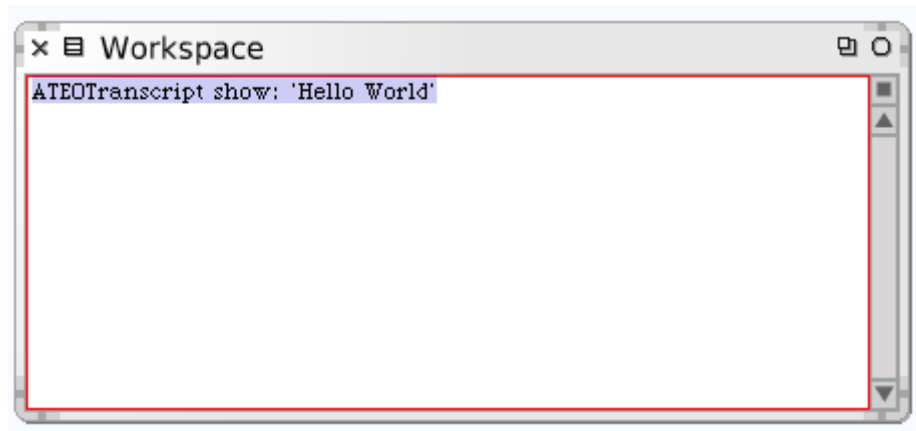
Tools in Squeak: Transcript

- » Systemconsole für Debug-Ausschriften



Tools in Squeak: ATEO Transcript

- » Ausgabe direkt auf dem Bildschirm
 - » → schneller als Transcript



Tools in Squeak: Message Names

» Suchen von geeigneten Nachrichten

Suchfeld

Treffer

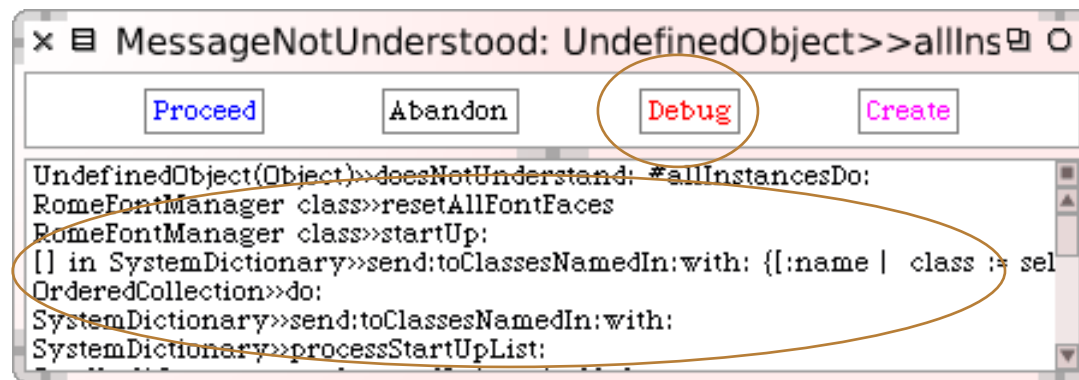
Quelltext

In welchen Klassen implementiert

```
Message names containing "addmorph"
Search: addMorph
addMorph:
addMorph:after:
addMorph:asElementNumber:
addMorph:asElementNumber:inGlobalFlapSatisfyi
addMorph:asElementNumber:inGlobalFlapWithID:
addMorph:behind:
addMorph:centeredNear:
addMorph:frame:
addMorph:fullFrame:
addMorph:inFrontOf:
addMorphBack:
addMorphCentered:
FlashPlayerMorph addMorph:
Morph addMorph:
ScreeningMorph addMorph:
browse senders implementors versions inheritance hierarchy inst vars class vars source
addMorph: aMorph
self addMorphFront: aMorph.
```

Tools in Squeak: Debugger

» startet sich bei Fehlern automatisch



Tools in Squeak: Debugger (cont.)

The screenshot displays the Squeak Debugger interface with the following components:

- Stack Trace:** Located at the top, it shows the call stack starting with `UndefinedObject(Object)>>doesNotUnderstand: #allInstancesDo:` and `RomeFontManager class>>resetAllFontFaces`. An arrow points to the `ifAbsent` method in the stack trace.
- Debug-Aktionen:** A row of buttons for debugging actions: `Proceed`, `Restart` (highlighted with an orange oval), `Into`, `Over`, `Through`, `Full Stack`, and `Where` (also highlighted with an orange oval). An arrow points to this row.
- Code:** The source code for the `resetAllFontFaces` method is shown below the buttons. The line `RomeFreetypeFont allInstancesDo: [:each |` is highlighted in blue. An arrow points to this code block.
- Objekt-Inspektor:** The bottom section contains four panels for inspecting objects:
 - `self`: Lists instance variables like `all inst vars`, `superclass`, `methodDict`, `format`, and `instanceVariables`.
 - `superclass: Object`: Shows the superclass and its dictionary.
 - `thisContext`: Shows the current context with `all temp vars` and `each`.
 - `RomeFontManager`: Shows the current method being executed: `class>>resetAllFontFaces`.An arrow points to this section.

Tools in Squeak: Debugger (cont.)

» **Proceed**

- » Programm an der aktuellen Stelle fortsetzen
- » Debugger wird beendet

» **Restart**

- » Erneute Ausführung des aktuellen Kontexts

» **Into**

- » Hineingehen in die aktuelle Methode

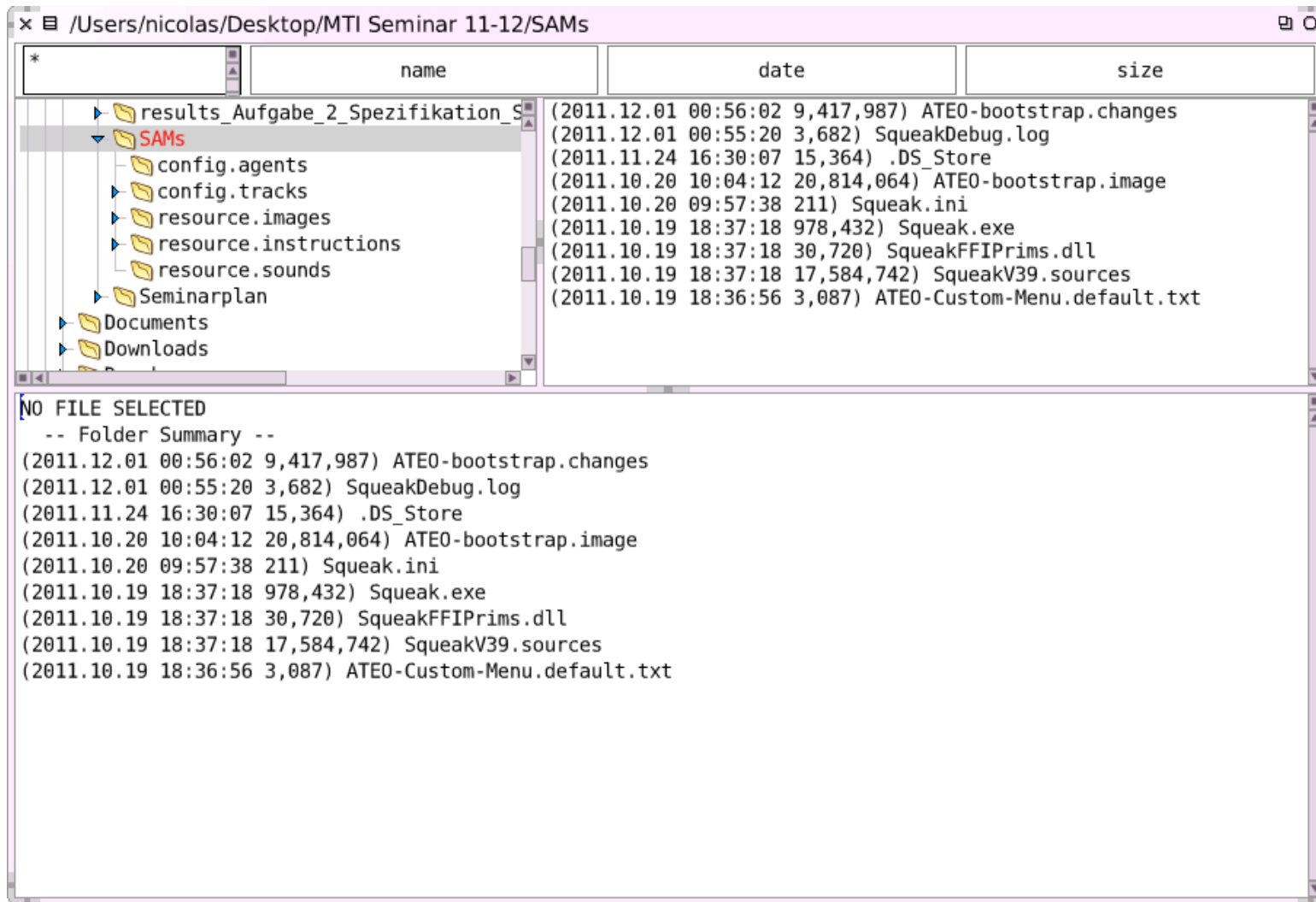
» **Over**

- » Senden der aktuellen Nachricht

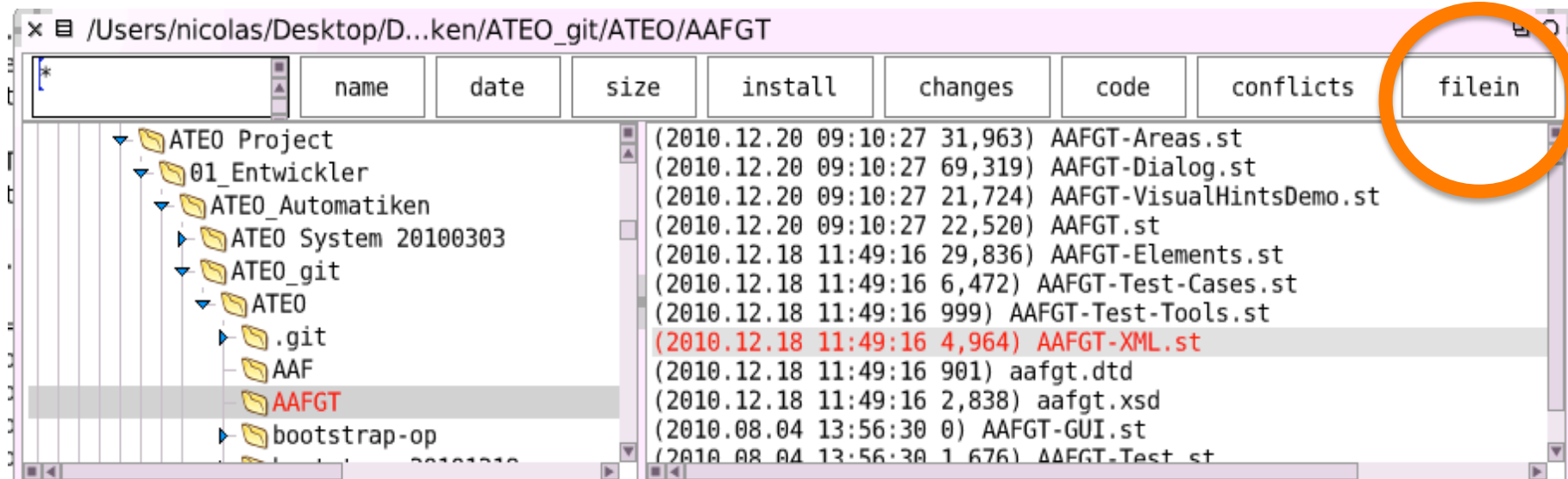
» **Through**

- » Hineingehen in den aktuellen Block

Tools in Squeak: File List



Tools in Squeak: File List (cont.)



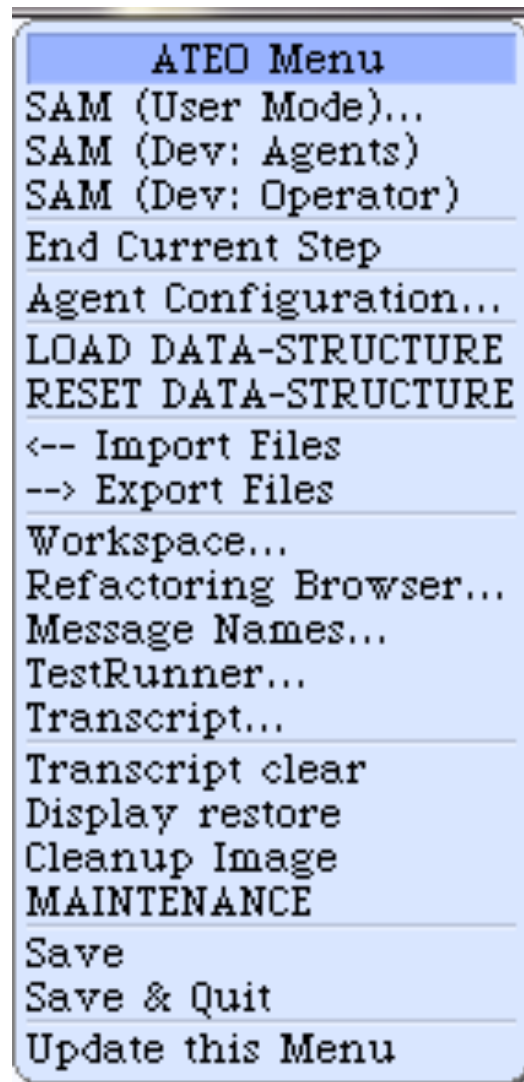
The screenshot shows a Squeak File List window with the following structure:

	name	date	size	install	changes	code	conflicts	filein
ATEO Project		(2010.12.20 09:10:27	31,963)			AAFGT-Areas.st		
01_Entwickler		(2010.12.20 09:10:27	69,319)			AAFGT-Dialog.st		
ATEO_Automatiken		(2010.12.20 09:10:27	21,724)			AAFGT-VisualHintsDemo.st		
ATEO System 20100303		(2010.12.20 09:10:27	22,520)			AAFGT.st		
ATEO_git		(2010.12.18 11:49:16	29,836)			AAFGT-Elements.st		
ATEO		(2010.12.18 11:49:16	6,472)			AAFGT-Test-Cases.st		
.git		(2010.12.18 11:49:16	999)			AAFGT-Test-Tools.st		
AAF		(2010.12.18 11:49:16	4,964)			AAFGT-XML.st		
AAFGT		(2010.12.18 11:49:16	901)			aafgt.dtd		
bootstrap-op		(2010.12.18 11:49:16	2,838)			aafgt.xsd		
		(2010.08.04 13:56:30	0)			AAFGT-GUI.st		
		(2010.08.04 13:56:30	1,676)			AAFGT-Test.st		

Gliederung

- » 1. Einführung Smalltalk
 - » 1.1 Grundlegende Eigenschaften
 - » 1.2 syntaktische Besonderheiten
- » 2. Einführung Squeak
 - » 2.1 Tools in Squeak
 - » System Browser
 - » Workspace
 - » Transcript
 - » Message Names
 - » Debugger
 - » File List
 - » **2.2 ATEO Developer Tools**
 - » ATEO Menu

ATEO Developer Tools: ATEO Menu



ATEO Developer Tools: ATEO Menu (cont.)

- » SAM (User Mode)
 - » Startet SAM „normal“ mit Konfigurationsmenü und Instruktionen
- » SAM (Dev: Agents)
 - » startet SAM mit den konfigurierten Automaten und ohne Konfigurationsmenü und Instruktionen
- » SAM (Dev: Operateur)
 - » Startet SAM im Operateur-Modus ohne Konfigurationsmenü und Instruktionen
- » EndCurrentStep
 - » Alt + . unterbricht SAM → EndOfCurrentStep beendet aktuellen Versuchsdurchlauf von SAM

ATEO Developer Tools: ATEO Menu (cont.)

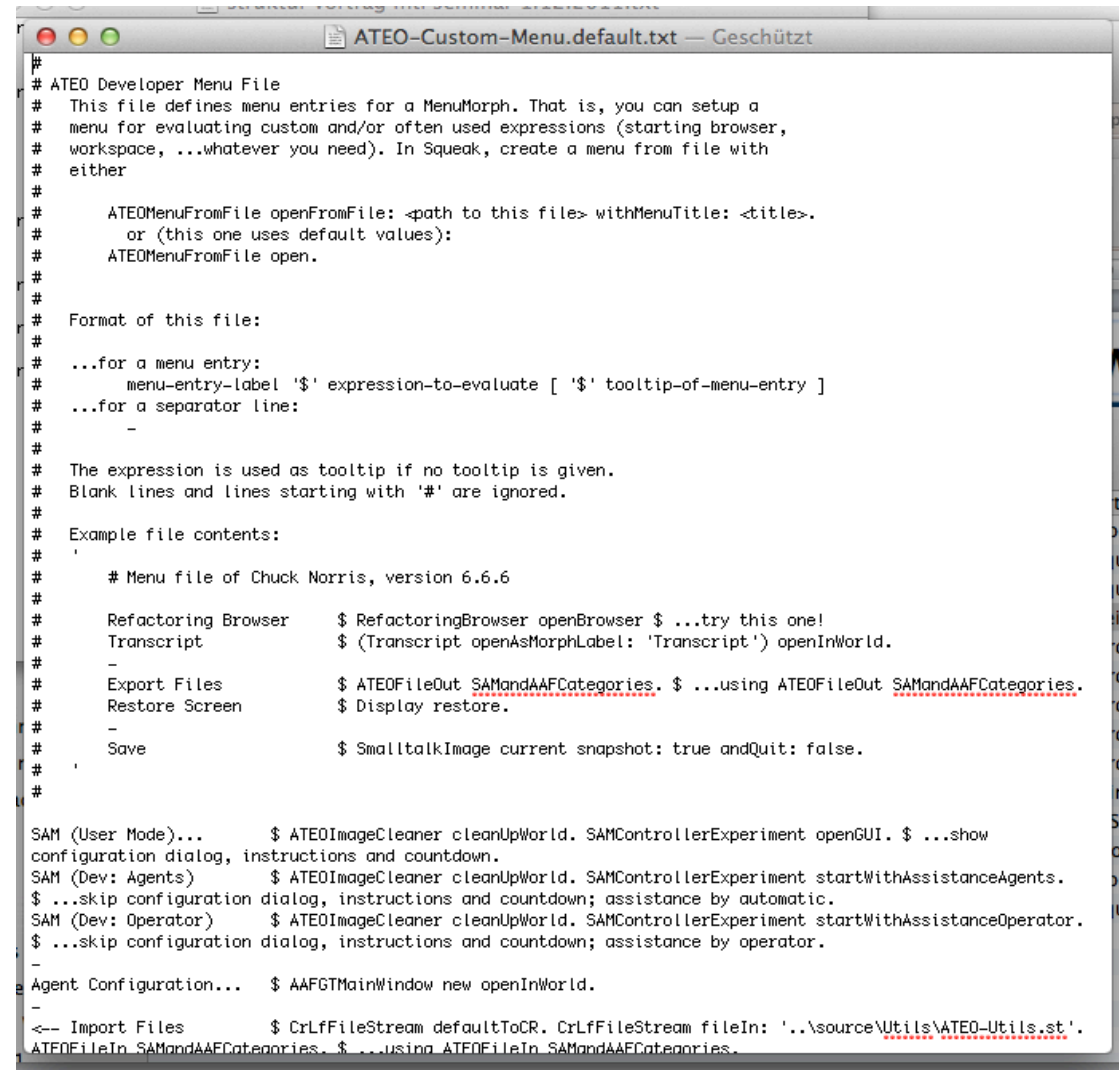
- » Agent Configuration
 - » startet das Konfigurationsfenster der Automaten
- » Load & Reset Data Structure
 - » externe Daten (Begrenzungen der Straße) werden geladen oder zurückgesetzt
- » Export/Import Files
 - » File-in/file-out von Source code
- » Workspace, Refactoring Browser, Message Names, Test Runner, Transcript
 - » startet die entsprechenden Tools
- » Transcript clear
 - » Löscht die Ausschriften im Transcript

ATEO Developer Tools: ATEO Menu (cont.)

- » Display Restore
 - » Löscht zurückgebliebene Bilder, die bspw. durch BitBlitern auf den Bildschirm gezeichnet wurden
- » Cleanup Image
 - » Löscht Objekte ohne Referenzen und vollzieht weitere speicherbereinigende Aktionen
- » Maintenance
 - » Ausführen von Transcript clear, Display restore, garbageCollect, cleanup Image
- » Save/Save & Quit
 - » Speichert bzw. speichert und schließt Squeak
- » Update Menu
 - » Bei Änderungen am Menü kann hiermit ein visuelles update ausgelöst werden und neue Einträge werden im aktuellen Image sichtbar

ATEO Developer Tools: ATEO Menu (cont.)

- » Konfiguration des Menus über Textdatei ATEO-Custom-Menu.default



```
#
# ATEO Developer Menu File
# This file defines menu entries for a MenuMorph. That is, you can setup a
# menu for evaluating custom and/or often used expressions (starting browser,
# workspace, ...whatever you need). In Squeak, create a menu from file with
# either
#
#     ATEOMenuFromFile openFromFile: <path to this file> withMenuTitle: <title>.
#     or (this one uses default values):
#     ATEOMenuFromFile open.
#
#
# Format of this file:
#
# ...for a menu entry:
#     menu-entry-label '$' expression-to-evaluate [ '$' tooltip-of-menu-entry ]
# ...for a separator line:
#     -
#
# The expression is used as tooltip if no tooltip is given.
# Blank lines and lines starting with '#' are ignored.
#
# Example file contents:
#
#     # Menu file of Chuck Norris, version 6.6.6
#
#     Refactoring Browser    $ RefactoringBrowser openBrowser $ ...try this one!
#     Transcript              $ (Transcript openAsMorphLabel: 'Transcript') openInWorld.
#     -
#     Export Files           $ ATEOFileOut SAMandAAFCategories. $ ...using ATEOFileOut SAMandAAFCategories.
#     Restore Screen         $ Display restore.
#     -
#     Save                   $ SmalltalkImage current snapshot: true andQuit: false.
#
#
# SAM (User Mode)...        $ ATEOImageCleaner cleanUpWorld. SAMControllerExperiment openGUI. $ ...show
# configuration dialog, instructions and countdown.
# SAM (Dev: Agents)         $ ATEOImageCleaner cleanUpWorld. SAMControllerExperiment startWithAssistanceAgents.
# $ ...skip configuration dialog, instructions and countdown; assistance by automatic.
# SAM (Dev: Operator)       $ ATEOImageCleaner cleanUpWorld. SAMControllerExperiment startWithAssistanceOperator.
# $ ...skip configuration dialog, instructions and countdown; assistance by operator.
#
# Agent Configuration...    $ AAFGMainWindow new openInWorld.
#
# <-- Import Files         $ CrLfFileStream defaultToCR. CrLfFileStream fileIn: '..\source\Utils\ATEO-Utils.st'.
# ATEOFileIn SAMandAAFCategories. $ ...using ATEOFileIn SAMandAAFCategories.
```


DANKE FÜR DIE AUFMERKSAMKEIT.

