Smalltalk: OOP allo stato puro

Giovanni Corriga

gcorriga@gulch.crs4.it

Cagliari, 27/11/2004

GULCh – Gruppo Utenti Linux Cagliari

Cos'è Smalltalk?

The purpose of the Smalltalk project is to provide computer support for the creative spirit in everyone.

Dan Ingalls

We aim to make simple things simple and complex things possible.

Alan Kay

In essence, Smalltalk is a programming language focused on human beings rather than the computer.

Alan Knight

I principi

Vediamo quali sono i concetti base di Smalltalk:

- Tutto è un oggetto
- L'immagine del sistema
- Il ciclo di sviluppo
- Linguaggio semplice, libreria sofisticata

Tutto è un oggetto

Smalltalk è un linguaggio Object Oriented *puro*.
Ogni oggetto è una *instanza* di una qualche *classe*.

`12` è una instanza della classe SmallInteger.
`'stringa'` è una instanza della classe String.
Il compilatore è una instanza della classe Compiler.
Un metodo è una instanza di CompiledMethod.
Una finestra è una instanza di SystemWindow.

E' un approccio semplice, facile da comprendere.

Tutto è un oggetto

Gli oggetti interagiscono tra loro spedendo dei *messaggi*.

```
anArray size
100 squared

12 + 27.3
800 @ 600

aDictionary at: #key put: 'value'
Compiler evaluate: aString
```

In Smalltalk, anche gli operatori aritmetici sono messaggi.

L'immagine del sistema

Un ambiente o una singola applicazione in Smalltalk può essere visto come un insieme di oggetti, ciascuno instanza della propria classe. Poiché tutto è un oggetto, questo insieme comprende anche le classi ed i metodi di queste ultime.

Questo insieme è contenuto in una unica *immagine*.

Generalmente l'immagine è salvata in un unico file.
E' l'approccio opposto a quello usato da Java.

L'immagine del sistema

Smalltalk è un ambiente altamente riflessivo. Ogni aspetto del sistema può essere indagato o manipolato spedendo opportuni messaggi.

L'immagine può quindi essere vista come un database da interrogare e modificare a piacere:

```
SystemNavigation new browseSendersOf: #nextPutAll:

Collection allSubclasses
```

Il ciclo di sviluppo

In linguaggi come Java o C#, il ciclo di sviluppo è il tradizionale *Edit-Compile-Run-Debug*:

- *Edita* i file con il codice sorgente
- *Compila* il codice sorgente
- *Esegui* il programma
- Effettuane il *Debug*

In pratica, il sistema offre dei mattoni con cui costruire il programma, secondo il progetto rappresentato dal sorgente.

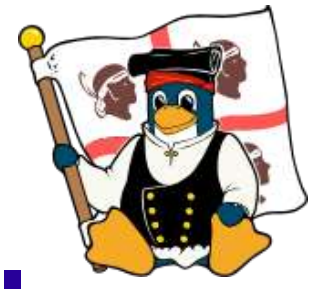
Il ciclo di sviluppo

L'approccio di Smalltalk è molto diverso.

Si modifica l'immagine dell'ambiente, creando nuove classi, aggiungendo metodi ad esse, modificando i metodi delle classi già esistenti (anche quelle di sistema!), e così via.

Una volta che l'applicazione è pronta, si *riduce* l'immagine eliminando tutte le classi ed i metodi inutili.

Creare una applicazione in Smalltalk è come scolpire un blocco di roccia malleabile.



Linguaggio semplice...

Talk small, and carry a big class library.

James Robertson

Smalltalk, come linguaggio, è molto semplice:

- Un solo operatore (di assegnazione, `:=`)
- Cinque parole chiave: `self`, `super`, `nil`, `true`, `false`.
- Una sintassi molto semplice, che si impara in due ore.
- Tipizzazione dinamica e *late binding*
- Ereditarietà singola

E tutto il resto? E' nella *class library*...

...librerie sofisticate

La class library di Smalltalk è molto sofisticata.

Molte delle sue classi implementano funzionalità che in altri linguaggi sono fissate nella sintassi:

```
[self isNotEmpty] whileTrue: [self keepWorking]
```

A parte questo, Smalltalk ha funzionalità per il networking, parsing XML, interfacce utente... tutto quello che ci si aspetta da una *framework* moderno.

Il linguaggio

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

(from Ralph Johnson)

Pseudovariabili

Smalltalk ha cinque *pseudovariabili*: variabili con un significato particolare.

`self` rappresenta l'instanza corrente.

`super` rappresenta l'instanza corrente, ma considerata come instanza della superclasse, e non della classe.

`self` e `super` vengono in genere usate nei metodi, con qualche interessante eccezione.

Pseudovariabili

`nil` è l'unica instanza della classe UndefinedObject.
E' equivalente al `null` di Java (ma a `nil` possono essere spediti messaggi).

`true` è l'unica instanza della classe True.
`false` è l'unica instanza della classe False.

True e False sono a loro volta sottoclassi di Boolean.

`nil`, `true` e `false` sono tre esempi dell'uso del pattern Singleton nel linguaggio Smalltalk.

Precedenza

I vari messaggi da spedire vengono spediti secondo questo ordine:

1. messaggi unari
2. messaggi binari
3. messaggi *keyword*

Per cambiare l'ordine di precedenza si devono usare parentesi: `( )`.

Ma allora, quanto fa `1 + 2 * 3` ?

Blocchi

Un blocco è un insieme di invii di messaggi (di istruzioni) compilato, che può essere eseguito quante volte si vuole mandandogli opportuni messaggi (`#value`, `#value:`, etc).

Un blocco può avere zero o più argomenti, zero o più variabili temporanee, e la sua valutazione restituisce sempre un valore che dipende dal blocco stesso. Può essere passato come argomento di un messaggio o restituito da un messaggio.

```
[:each | Transcript show: each asString]
```

Blocchi

Un blocco è una *closure*, simile alle lambda di LISP e Python, agli oggetti Proc di Ruby, alle `function` di JavaScript.
Le inner class anonime di Java sono delle versioni meno potenti dei blocchi di Smalltalk.

I blocchi sono uno dei componenti fondamentali su cui si basa un framework Smalltalk.
Sono impiegati per le strutture di controllo, l'iterazione su collezioni, la gestione delle eccezioni e del multithreading.

Strutture di controllo

```
aBooleanExpression
   ifTrue: aBlock
   ifFalse: anotherBlock
```

`aBooleanExpression` è un'espressione che restituisce `true` oppure `false`. Se restituisce `true`, viene valutato `aBlock`, altrimenti viene valutato `anotherBlock`.
`#ifTrue:ifFalse:` e le relative varianti sono metodi implementati nelle classi True e False.

```
(network isOn)
   ifTrue: [self sendPendingData]
   ifFalse: [self wait]
```

Strutture di controllo

```
[aBooleanTest] whileTrue: aBlock
```

`aBooleanTest` è un blocco che dà come risultato `true` oppure `false`. Se restituisce `true`, `aBlock` viene valutato, e poi il messaggio `#whileTrue:` viene di nuovo spedito a `aBooleanTest`. Una valutazione di `aBlock` può anche modificare il risultato di `aBooleanTest`.

Collezioni

Smalltalk offre una ricca gamma di collezioni:

Array	Dictionary
OrderedCollection	Set
SortedCollection	SharedQueue

Tutte queste classi derivano dalla classe Collection.

Collection è una classe astratta che offre i metodi base per interagire con una collezione.

Collezioni

Accesso agli elementi: `#at:` e `#at:put:`

```
anArray at: 2 put: 'number 2'
aDictionary at: #key put: aValue
```

Aggiungere elementi: `#add:` e `#addAll:`

```
anArray add: 1
aDictionary addAll: aCollectionOfAssociations
```

Dimensioni: `#size`

```
#(1 2 3 4 5) size
```

Collezioni

Sono numerosi anche i metodi per iterare su una collezione: `#do:`, `#select:` e `#reject:`, `#detect:` e altri.

Questi metodi prendono come argomento un blocco che ha a sua volta un argomento.

Iterare su tutti gli elementi: `#do:`

```
aCollection do: [:each | Transcript show: each]
```

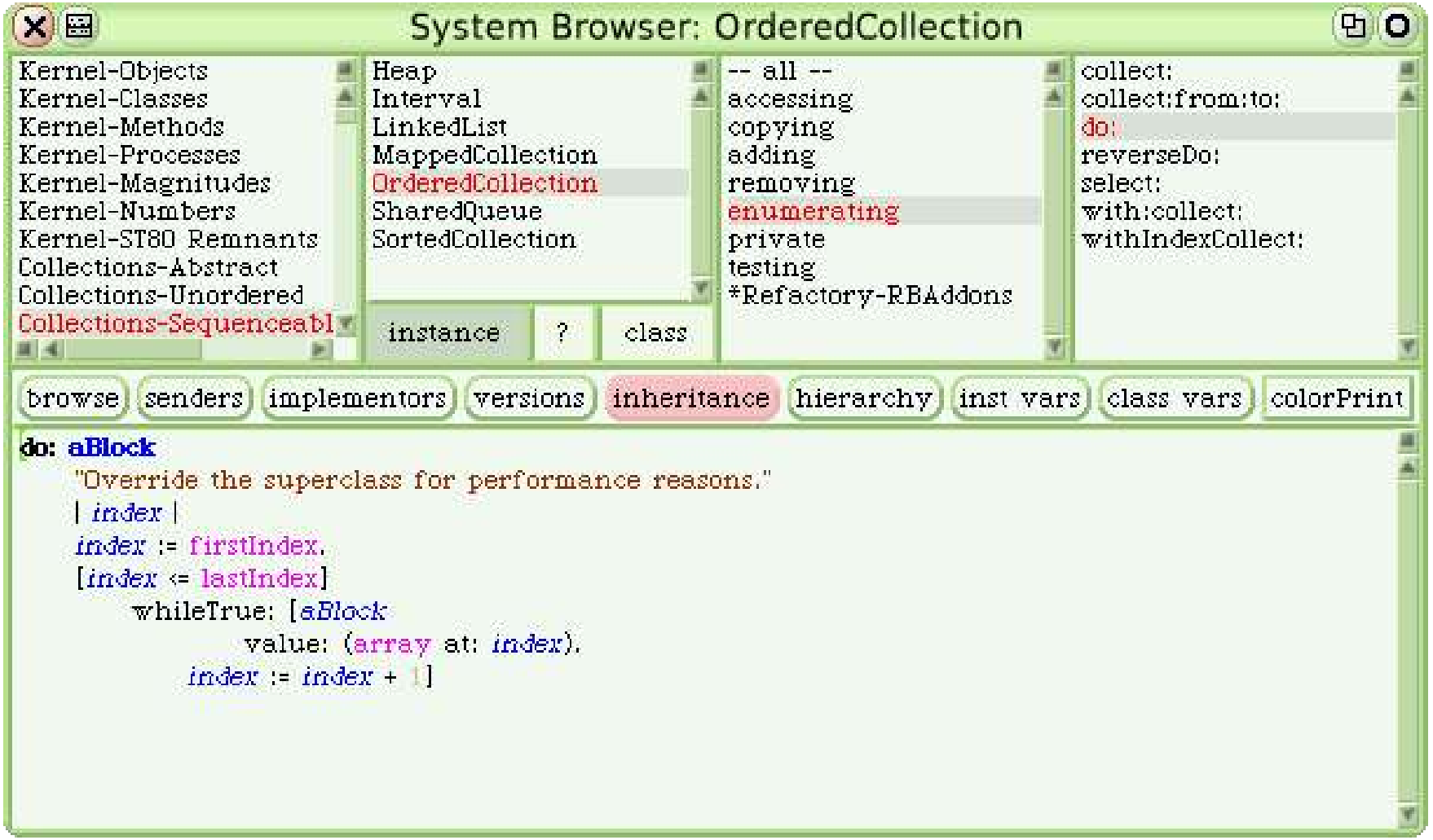
Eccezioni

I blocchi vengono utilizzati anche per la gestione delle eccezioni.

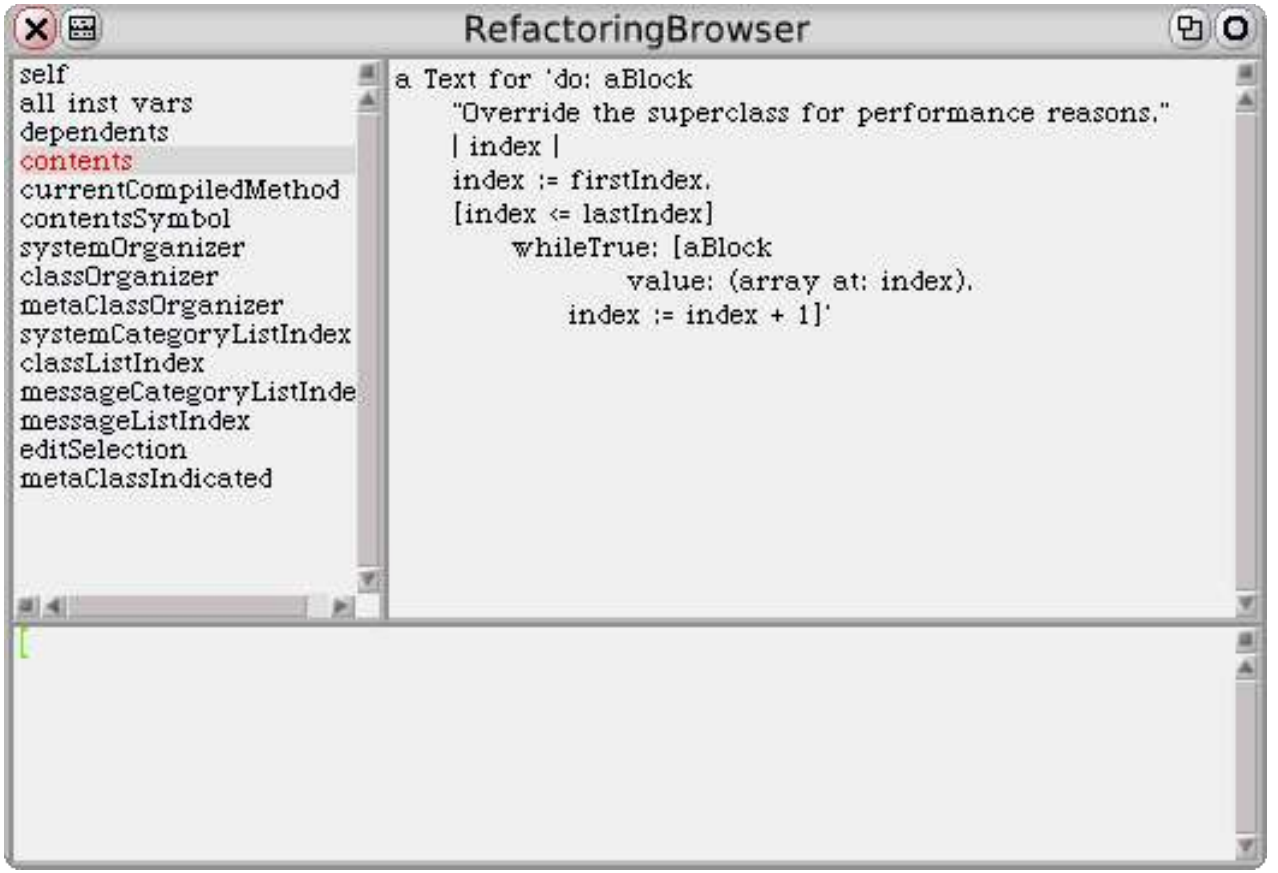
La classe BlockContext ha un metodo #on:do: che viene utilizzato per definire i gestori di eccezioni.

```
[ 10 / 0 ]
   on: ZeroDivide
   do: [:ex | Transcript show: ex asString]
```

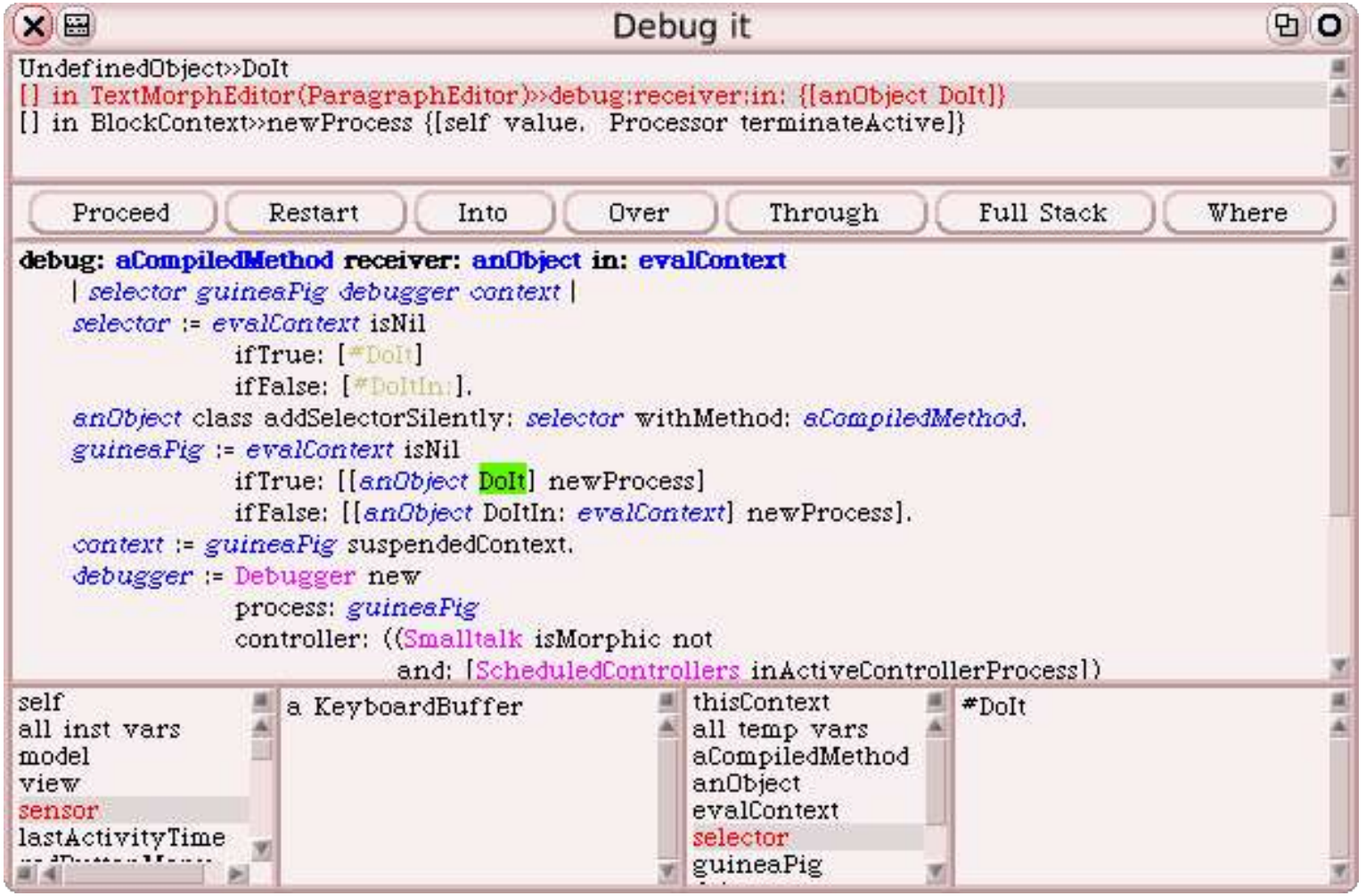
Il Browser



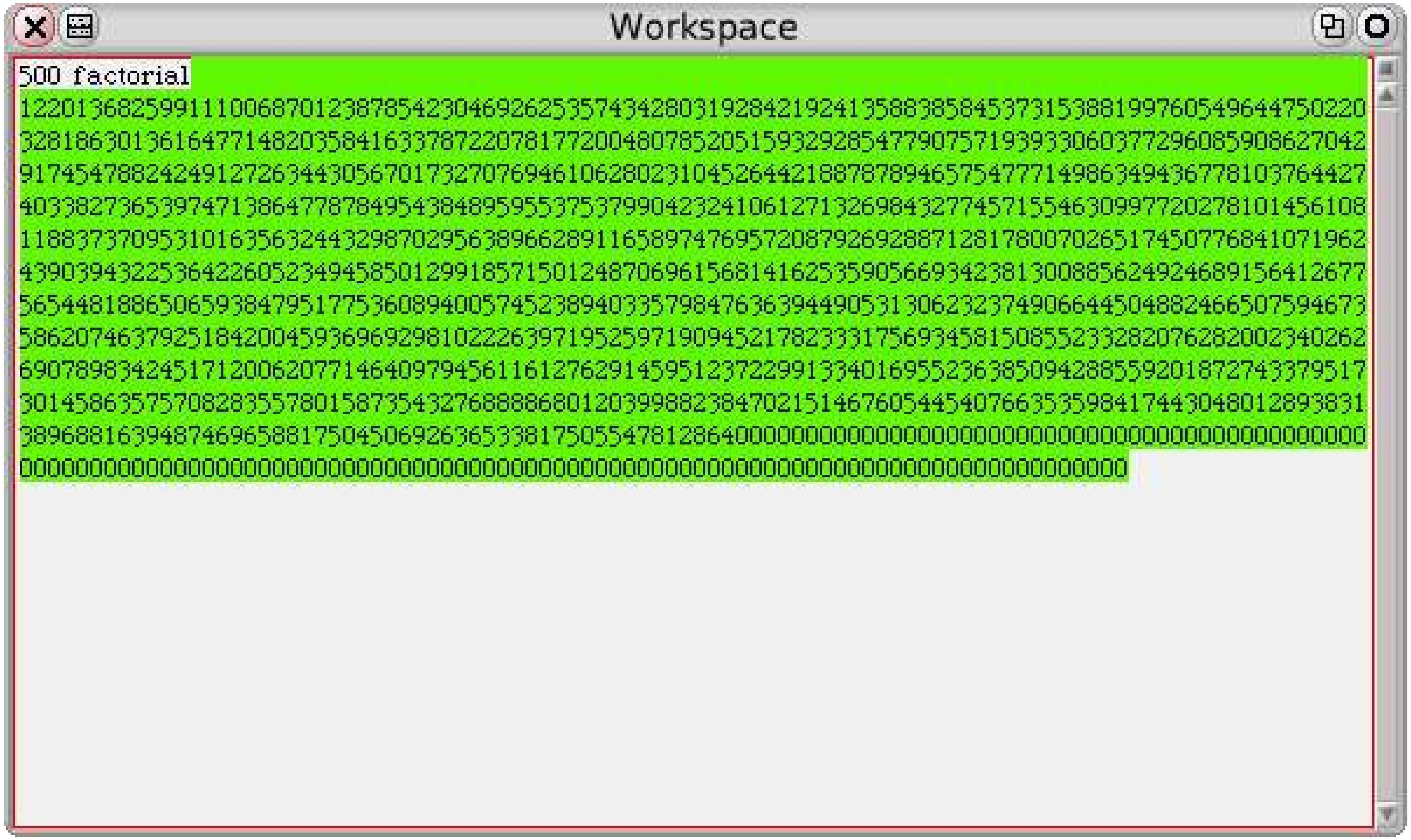
L'Inspector


RefactoringBrowser
self
all inst vars
dependents
contents
currentCompiledMethod
contentsSymbol
systemOrganizer
classOrganizer
metaClassOrganizer
systemCategoryListIndex
classListIndex
messageCategoryListInde
messageListIndex
editSelection
metaClassIndicated
a Text for 'do: aBlock
"Override the superclass for performance reasons."
| index |
index := firstIndex.
[index <= lastIndex]
whileTrue: [aBlock
value: (array at: index).
index := index + 1]'

Il Debugger



Il Workspace



Tutti i gusti di Smalltalk

Esistono varie implementazioni di Smalltalk:

Commerciali:

Ambrai Smalltalk (beta) *	Gemstone/S *
Cincom VisualWorks *	IBM VisualAge
Cincom ObjectStudio *	Smalltalk MT
Dolphin Smalltalk *	S# *

* per questa versione è disponibile una licenza gratuita

Tutti i gusti di Smalltalk

Open Source e/o free:

GNU Smalltalk (GPL)

Altro:

Squeak Smalltalk
Smalltalk/X

La licenza di Squeak non è né GPL-compatible né OSI-approved per via di alcune clausole secondarie. E' comunque Open Source in spirito, se non nella lettera.

La comunità Smalltalk

La comunità degli Smalltalkers è una delle comunità più aperte che esistano.

Dato il particolare approccio del linguaggio, concetti come la libera distribuzione del codice sono sempre stati di casa nella comunità Smalltalk.

A tutt'oggi c'è un vivace scambio di idee, esperienze e soluzioni tra le varie anime (accademica, commerciale e hobbistica) della comunità.

Gli eventi Smalltalk

Gli eventi Smalltalk principali sono la ESUG Conference in Europa (a cura dello European Smalltalk User Group) e lo Smalltalk Solutions negli Stati Uniti (a cura dello Smalltalk Industry Council).

A fianco di questi ed altri eventi si tengono i *Camp Smalltalk*: riunioni di Smalltalker che collaborano a svariati progetti, in totale libertà.

Bibliografia

Alcuni link utili su Smalltalk

http://www.cincomsmalltalk.com
http://www.whysmalltalk.com
http://www.smalltalk.org
http://www.squeak.org
http://directory.fsf.org/devel/prog/smalltalk.html

http://www.esug.org
http://www.stic.org

Bibliografia

Mailing list:

La mailing list per gli sviluppatori di Squeak Smalltalk
squeak-dev@lists.squeakfoundation.org

La mailing list per gli utenti VisualWorks (Non Commercial)
vwnc@cs.uiuc.edu

La comunità italiana degli utenti Smalltalk
smalltalkit@lists.unica.it

Bibliografia

Numerosi libri su Smalltalk sono disponibili in formato elettronico al sito del Prof. Stephane Ducassé:
http://www.iam.unibe.ch/~ducasse/FreeBooks.html
Queste copie elettroniche sono state autorizzate dagli autori e dagli editori.

Domande?