Smalltalk - die Programmiersprache

die uralte Programmiersprache mit aktuellen Sprach-Features

Karsten Pietrzyk

26.01.2012

Smalltalk ist alt

- Wie alt genau?

Smalltalk ist alt

- Wie alt genau? $\rightarrow$ 1971!
- The first version, known as Smalltalk-71, was created by Ingalls in a few mornings on a bet that a programming language based on the idea of message passing inspired by Simula could be implemented in "a page of code". englische Wiki-Seite zu Smalltalk

Die Syntax

- „Alles ist ein Objekt!“ deutsche Wiki-Seite zu Smalltalk
  - Auch Integer- und Boolean-Werte
  - und Klassen
  - und Code-Blöcke.
- Die grundlegene Syntax ist einfach und aussagekräftig.

$$
\begin{aligned}
\langle \textit{Objekt} \rangle &::== \langle \textit{Literal} \rangle \mid \langle \textit{Variable} \rangle \mid \langle \textit{Ausdruck} \rangle \\
\langle \textit{Ausdruck} \rangle &::== \langle \textit{Objekt} \rangle \ \langle \textit{Nachricht} \rangle \\
&\quad \mid \langle \textit{Objekt} \rangle \ (\langle \textit{Nachricht} \rangle : \langle \textit{Ausdruck} \rangle)^{+} \\
\langle \textit{Anweisung} \rangle &::== \langle \textit{Objekt} \rangle .
\end{aligned}
$$

Beispiel

```
14 factorial.
Transcript show: 'Hallo, Welt!'.
1 to: 10 do: [:i | Transcript show: i * i ].
'the-raven.txt' asFilename contents.
```

Features moderner Programmiersprachen

Ruby: Klassen sind Objekte

Man kann Klassen als Parameter übergeben und Methoden, wie z.B. **new** daran aufrufen.

```
Object.new
Array.superclass
String.instance_methods
temp = Hash
temp.new.to_s == "{}"
```

In Smalltalk ist alles ein Objekt, auch Klassen.
Die zweit-wichtigste Nachricht (neben new), die man einer Klasse in Smalltalk schicken kann, ist **subclass:...**

Unterklassenbildung in Smalltalk

Beispiel für subclass:...

Object sublass: #Lebewesen
instanceVariableNames: 'alter'
classVariableNames: 'anzahlAllerLebewesen'
poolDictionaries: ' '
category: 'Eigenes'.

Alle Klassen lassen sich mit einer Dokumentation versehen.

Lebewesen comment: 'Meine Exemplare stellen Lebewesen dar.'.

Die lassen sich auch wieder auslesen:

UndefinedObject comment
$\rightarrow$ 'I describe the behavior of my sole instance, nil...'

Funktionen-Features

C#: Funktionen sind Objekte

Man kann Funktionen als Parameter übergeben sie mit Argumenten ausführen.

```
Action<string> put = text => Console.Write(text);
put("Hallo, Lambda-Ausdruck!");
```

In Smalltalk sind sogar Kontrollstrukturen wie **if** und **while** mit Funktions-Objekten realisiert.

Beispiel mit **if**

7 factorial > 1337
ifTrue: [#greater]
ifFalse: [#smaller].

Der Code wird zum Symbol #greater ausgewertet.

Funktionen-Features

C#: Funktionen sind Objekte

Das führt zu tollen Möglichkeiten für funktionale Programmierung:

```
Array.FindAll(new int[] {5, 17, 42, 194}, i => i % 2 == 0);
```

oder ab C# 3.0 mit Linq sowas wie:

```
Enumerable.Range(1553, 3741)
.Where(item => item % 5 > 2)
.OrderBy(item => item.ToString().Length)
.Select(item => item / 2.0)
.ToList();
```

Funktionen-Features

In Smalltalk?

In Smalltalk schon lange vertreten:

Map (als collect:), Filter (als select:) und Fold (als inject:into:).

In Smalltalk kann man übrigens Objekte als Konstante-Funktionen auffassen, denn Funktionen sind Objekte, die die Nachricht value und valueWithArguments: verstehen.

Diese sind in Object (in Squeak! jedenfalls) mit ↑self (`return this` in Java) implementiert.

Sehr schön gelöst ist die Iteration über Sammlungen (oder Intervalle, wie z.B. 1 to: 29 by: 3)...
Jedes „iterierbare“ Objekt versteht die Nachricht do:, der als Parameter eine unäre Funktion mitgegeben wird, die auf jedes Objekt angewendet werden soll.

Arithmetik-Features

Ruby: Zahlen sind Objekte, $+, -, \cdot, /$ sind Methoden

Zahlen laufen nicht mehr über!

```
1312312931923912 3.class == Fixnum
(1312312931923912 3 * 129381928391823 9).class == Bignum
```

In Smalltalk auch

30129222 class == SmallInteger
(30129222 * 30129222) class = LargePositiveInteger

Das geht, weil * eine Methode ist, die entscheiden kann, ob ein als Rückgabe ein Objekt mit höherer Genauigkeit geliefert werden sollte.

(Die Methode wird optimiert und prüft vermutlich das Overflow-Flag. Boolean»ifTrue: wird auch optimiert. . . Doku dazu in Squeak!)

Features

Java: void ist Blödsinn

Am Ende von StringBuffer.append steht `return this`, sodass Nachrichten-Kaskaden möglich sind.

Beispiel:

```
class Point
{
  public int x, y;
  public String toString()
  {
    return new StringBuffer()
    .append("[X=").append(x)
    .append(", Y=").append(y)
    .append("]").toString();
  }
}
```

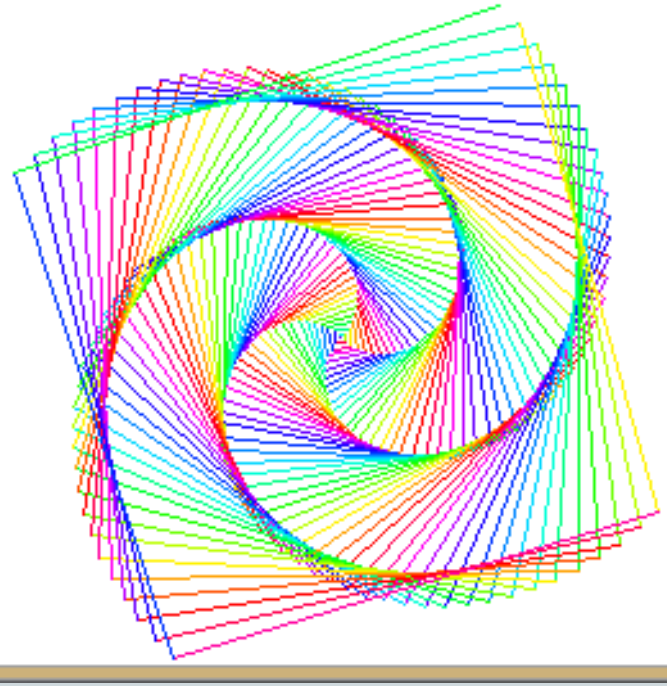
Nachrichten-Kaskade

Malen in Squeak!

Workspace

```
|pen|
pen := Pen new.
1 to: 200 do:
  [:step |
    pen color: step * 2;
        go: step;
        turn: 89.]
```

Transcript



» End of line.