SmallTalk - Eine kurze Einführung

Andreas Lochbihler

Lehrstuhl Programmierparadigmen
Universität Karlsruhe

30. April 2008

SmallTalk

- Entwickelt seit Anfang der 70er bei XEROX PARC
- Alan Kay, Dan Ingalls, Adele Goldberg, ...
- Beeinflußt von SIMULA
- Smalltalk-80 (1980)
- Programmiersprache und Entwicklungsumgebung
- Zahlreiche kommerzielle Varianten
- Standardisiert als ANSI Smalltalk 1998

- Im Folgenden: VisualWorks 7.5 von Cincom

Eigenschaften

- "Alles ist ein Objekt" :
  Strings, Integers, ... sind vollwertige Objekte
- Objekte und Botschaften
- Klassen und Vererbung
- Garbage Collection
- dynamisch typisiert

Das “Hello World” Programm

```
'Hello, World!' printOn: Transcript.
```

- `'Hello, World!'`
  Erzeugt ein Objekt der Klasse *String* mit dem entsprechenden Text
- `printOn`
  sendet die Botschaft *printOn* an das vorangestellte Objekt mit Parameter Transcript.
- `.`
  Ende des Ausdrucks

Methodenaufrufe

- `anObject message.`
  Dem Object `anObject` wird die Botschaft `message` geschickt.
- `anObject selector.` Unäre Botschaft.

  `3 negated.`
- `anObject selector: aParameter.` Binäre Botschaft.

  `3 raisedTo: 5.`
- `anObject operator aParameter.`

  `3 + 6.`
- `anObject select1: param2 select2: param2.`
  Schlüsselwort-Botschaft.

  `3 between: 5 and: 10.`
- `anObject message1; message2; message3.`
  Aufruf mehrerer Methoden desselben Objekts.
  `3 negated; + 10; between: 5 and: 20.` liefert false
  `3 negated + 10 between: 5 and: 20.` ergibt true

Arithmetik

```
> 42 printOn: Transcript.
42

> 6 * 8 printOn: Transcript.
48

> 2 + 6 * 8 printOn: Transcript.
64

> 6 + 8 negated printOn: Transcript.
-2
```

$\Rightarrow$ Arithmetische Operatoren sind normale Messages
$\Rightarrow$ Unär vor Binär vor Schlüsselwort-Message

Variablen

```
> Smalltalk at: #x put: 0.
> x := 5.
> Smalltalk at: #y put: 0.
> y := 7.
> x * y printOn: Transcript.
35
```

`Smalltalk` ist das System Dictionary, eine Art Verzeichnis aller Variablen.

Datenstrukturen: Arrays

```
> x := Array new: 20.
> (x at: 1) printOn: Transcript.
nil
> x at: 1 put: 5; at: 2 put: 7.
> (x at: 1) printOn: Transcript.
5
```

$\Rightarrow$ Keine eigene Syntax

Datenstrukturen: Mengen

```
> x := Set new.
> x add: 5.
> x add: 7.
> x printOn: Transcript.
Set (7 5)
> x add: 'foo'.
> x printOn: Transcript.
Set (7 'foo' 5)
> x add: 3; add: 9; add: 'bar'.
> x printOn: Transcript.
Set (3 5 'bar' 7 9 'foo')
```

Überladene Methoden

- Bei Überladung in Java/C++: gleicher Methodenname, verschiedene Parametertypen
- In Smalltalk ohne statische Typen nicht möglich
- Alternative: Schlüsselwortbotschaft als mehrteiliger Methodenname
  $\Rightarrow$ Identifikation über Namen der Parameter

```
> x set: 1 at: 5.
> x set: 1 from: 5 to: 7.
```

Java-Äquivalent:

```
x.set(1,5);
x.set(1,5,7);
```

$\Rightarrow$ Deutlich erhöhter Dokumentationswert!

Eigene Klassen

```
Smalltalk.Examples defineClass: #Account
    superclass: #(Core.Object)
    indexedType: #none
    private: false
    instanceVariableNames: 'balance'
    classInstanceVariableNames: ''
    imports: ''
    category: '(none)'

initialize
    balance := 0.
    ^self

> x := Account new.
> x printOn: Transcript.
an Examples.Account
```

Schönere Ausgabe

Methoden werden in Protokollen gruppiert: printing

```
printOn: stream
    super printOn: stream.
    stream nextPutAll: ' with balance: '.
    balance printOn: stream
> x printOn: Transcript.
an Examples.Account with balance: 0
```

Methoden-Definition

```
withdraw: amount
     balance := balance - amount.

> x withdraw: 10.
> x printOn: Transcript.
an Examples.Account with balance: -10
```

Kontrollstrukturen: IF

```
withdraw: amount
    (amount < 0)
        ifTrue: [ ^self error: 'no no no' ].

    balance := balance - amount
```

```
> x withdraw: -100.
Unhandled exception: no no no
Examples.Account(Object)>>error:
Examples.Account>>withdraw:
UndefinedObject>>unboundMethod
UndefinedObject(Object)>>performMethod:arguments:
UndefinedObject(Object)>>performMethod:
```

Kontrollstrukturen: WHILE

```
withdraw2: amount
    | a |
    a := amount.
    [ a > 0 ]
       whileTrue: [ balance := balance - 1 .
                    a := a - 1 ].
```

Code-Blöcke als Objekte

```
callback: handler
     handler value: balance.

> x callback: [ :b | 'balance is ' printOn: Transcript.
                     b printOn: Transcript ]
'balance is '5
```

Interaktive Erweiterung bestehender Klassen

```
> y := Account new.
> y greet.
Unhandled Exception: Message not understood: #greet
...

greet
     'Hello, World!' printOn: Transcript.

> y greet.
'Hello, World!'
```

Erweiterung bestehender Klassen

```
> x isInteger printOn: Transcript.
false
> 3 isInteger printOn: Transcript.
true
> x isAccount printOn: Transcript.
Unhandled Exception: Message not understood: #isAccount
...
```

Reflektion

```
> x class printOn: Transcript.
Account
> x class superclass printOn: Transcript.
Object
> 3 class printOn: Transcript.
SmallInteger
> x class class printOn: Transcript.
Examples.Account class
> x class class class printOn: Transcript.
Metaclass
> x class class class class printOn: Transcript.
Metaclass class
> x class class class class class printOn: Transcript.
Metaclass
```

Entwicklungsumgebung

- üblich: Quelltexte werden im Klassenbrowser eingegeben
- Bytecode und virtuelle Maschine
- Interaktiver Interpreter
- Quelltexte sind offen und können live geändert werden
- Gesamter Systemzustand kann als Image gespeichert werden

Zusammenfassung

- “Alles ist ein Objekt”
- Objekte und Botschaften
- Einfach strukturiert, wenig Syntax $\Rightarrow$ Einheitlichkeit
- Arbeit am Live-System
- Relevant in der Industrie

Weiterführendes

- VisualWorks von Cincom
  `http://www.cincomsmalltalk.com/userblogs/cincom/blogView?content=vwfactsheet`
- GNU Smalltalk
  `http://www.gnu.org/software/smalltalk/smalltalk.html`
- Dan Ingalls "Design Principles Behind Smalltalk" (1981)
  `http://users.ipa.net/~dwighth/smalltalk/byte_aug81/design_principles_behind_smalltalk.html`
- Wilf LaLonde "I can Read C++ and Java But I can't Read Smalltalk" (2001)
  `http://www.eli.sdsu.edu/courses/spring01/cs635/readingSmalltalk.pdf`