Smalltalk

Gespräche über alltägliche, allgemeine und unwichtige Dinge.

Wikipedia

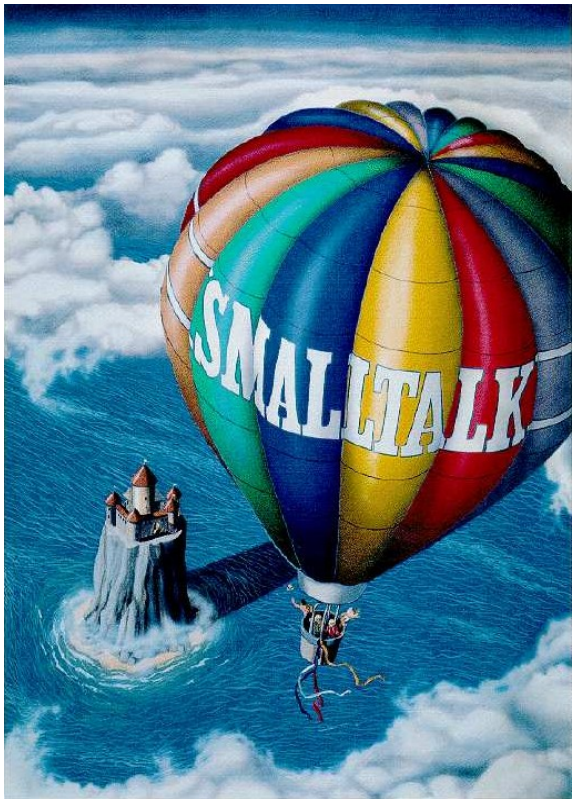
Smalltalk ist die Kunst zu reden, ohne zu denken.

Andreas Tenzer, Dozent für Philosophie

Smalltalk

Inhalt:

- Vorwort
- Geschichte
- Eigenschaften
- Sprachkonstrukte
- die IDE
- im Vergleich
- Quellen und Referenzen



Smalltalk

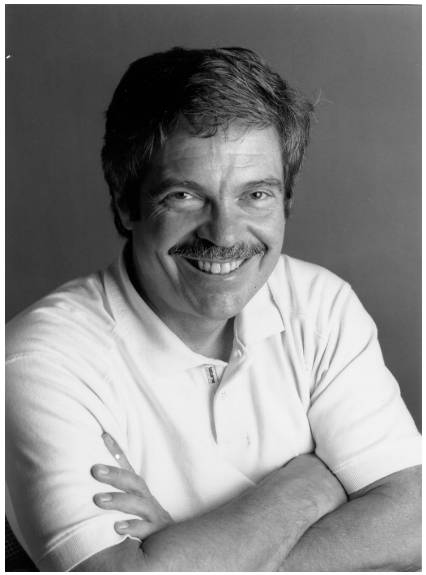
Vorwort:

- **Was ist Smalltalk?**
  - erste komplett objektorientierte Programmiersprache
  - Ausgangspunkt viele heute gängiger Konzepte
  - „Informatiker-Allgemeinbildung“
- **Was ist Smalltalk nicht?**
  - Standardisiert
  - in der Praxis relevant

Smalltalk

Geschichte:

- 70er Jahren im Palo Alto Research Center (PARC) von Xerox entwickelt
- Xerox wollte auf dem umkämpften Druckermarkt Boden gut machen
- gesucht wurde eine einfache Sprache, um mit Computern zu interagieren
- es entstanden die ersten Versionen und letztendlich Smalltalk-80



Alan Kay, Teamleiter

Smalltalk

Geschichte:

- Smalltalk war die erste komplett objektorientierte Sprache
- bereits moderne Konzepte *(VM, Byte-Code, Garbage Collector, IDE)*
- lange Zeit zu hohe Ansprüche für die Hardware
- dann in den 90er in vielen große Unternehmen zu finden

  (*z.B. Daimler Chrysler, BMW, Deutsche Bank, IBM*)
- Internet-Boom bestärkt die Java-Gemeinde
- von Firmenpleiten in Mitleidenschaft gezogen
- nie ein Standard durchgesetzt
- von Java und C++ fast komplett verdrängt

Smalltalk

Eigenschaften: „die wichtigsten“

- komplett objektorientiert, d.h.: **alles ist ein Objekt**
  - keine primitiven Datentypen wie **`int`** oder **`char`**
  - alle Operationen werden durch Methoden realisiert
  - Klassen, Rückgabewerte, Threads und Blöcke sind selbst stets Objekte
- reflexive Sprache
  - Smalltalk ist in Smalltalk implementiert (*Klassen, IDE...*)
  - wenige echte Primitive der VM
- keine statische Typisierung

Smalltalk

Eigenschaften: „...und“

- Image, als Speicher für alle Objekte
  - nicht dateibasiert
  - hier werden alle Objekte (Klassen etc.) gespeichert
  - auch die IDE liegt komplett im Image
- Just-in-Time Compilierung am laufenden System
  - beim Speichern der Klasse findet die Compilierung statt
  - es wirkt sich sofort auf das System aus
  - aufbrechen des Write-Compile-Test-Zyklus
- Außerdem: (nur) Einfachvererbung, Meta-Klassen, feste Sichtbarkeiten, Byte-Code der in der VM ausgeführt wird, IDE, umfangreiche und offene Klassen-bibliothek, ein Garbage Collector...

Smalltalk

Grundlegende Sprachkonstrukte:

- **„alle Operationen sind durch Methoden realisiert“**
  - **+ - * /** sind Methoden der jeweiligen Klasse
  - sie können vom Programmierer überschrieben werden
- **„keine syntaktischen Schleifen“**
  - auch Schleifen werden durch Methoden-Aufrufe realisiert
- **„alles ist ein Objekt“**
  - Threads, Rückgabewerte und Klassen sind selbst Objekte

Smalltalk

Grundlegende Sprachkonstrukte:

- es gibt nur 5 reservierte Schlüsselwörter:
  - **`self`**: bezieht sich auf das aktuelle Objekt
  - **`super`**: es wird ab der Super-Klasse nach einer Methode gesucht
  - **`true`**: repräsentiert einen wahren Boolean-Wert
  - **`false`**: repräsentiert einen falschen Boolean-Wert
  - **`nil`**: das „UndefindedObject“
- es gibt nur wenige unveränderliche Operatoren (*Auszug*):
  - **`:=`** dient der Zuweisung an Variablen
  - **`^`** dient der Rückgabe aus Methoden
  - **`[ ]`** markiert einen Block

Smalltalk

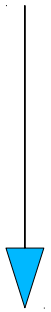
Wir haben jetzt ALLES!

Smalltalk

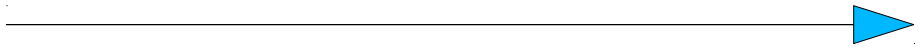
„alle Operationen sind durch Methoden realisiert“

```
|a|                          „Deklaration von lokalen Variablen“
a := 5 + 3 * 2.              „Methoden + und *“
Transcript print: a.         „Ausgabe ist: 16“
a inspect.                   „dazu später ein paar Worte“
a := 'Hallo'.                „keine statische Typisierung“
Transcript print: a.         „Ausgabe ist: Hallo“
```

- Unäre Nachrichten (*a inspect.*)
- Binäre Nachrichten *(5 + 3.)*
- Schlüsselwortnachrichten *(Transcript print: a.)*



Präzedenz



Evaluation

Smalltalk

„keine syntaktischen Schleifen“

```
|buchstaben|
buchstaben := OrderedCollection new.
buchstaben add: 'a'; add: 'b'; add: 'c'.
buchstaben add: 'd' after: 'a'.
buchstaben do:[:each | Transcript cr; show: each].
```

- Ausgabe:

```
a
d
b
c
```

Smalltalk

„keine syntaktischen Schleifen“

```
|buchstaben|
buchstaben := OrderedCollection new.
buchstaben add: 'a'; add: 'b'; add: 'c'.
buchstaben add: 'd' after: 'a'.
buchstaben do:[:each | Transcript cr; show: each].
```

- dem Objekt **`OrderedCollection`** (einer Klasse) wird die Nachricht **`new`** gesendet
- mit **`;`** können mehrere Nachrichten an ein Objekt geschickt werden
- **`buchstaben`** hat eine Methode **`add:after:`** mit zwei Parametern
- Blöcke **`[ ]`** sind Objekte und können als Parameter übergeben werden
  - ein Block konserviert Code
  - Smalltalk arbeitet mit Closures

Smalltalk

„alles ist ein Objekt“

```
|s a|
s := Semaphore new.
a := [Transcript show: 'A'].
a fork.
[Transcript show: 'B'. s wait. Transcript show: 'C']forkAt: 3.
[Transcript show: 'D'. s signal. Transcript show: 'E'; cr]forkAt: 2.
```

- Blöcke sind Objekte denen die Nachricht **`fork`** geschickt werden kann
- **`fork`** (bzw. **`forkAt: aNumber`**) erzeugt einen Thread (mit einer Priorität)

- Ausgabe:

```
A B D C E
```

Smalltalk

Die Entwicklungsumgebung:

- zu Smalltalk gehört immer eine Entwicklungsumgebung:

Smalltalk = Sprache + Klassenbibliothek + Entwicklungsumgebung

- die IDE ist selbst in Smalltalk programmiert
- sie läuft auf der Smalltalk-VM und ist im Image gespeichert
- sie wird aus dem fertigen Image gelöscht

Smalltalk

Die Entwicklungsumgebung:

- zu einer Smalltalk-Entwicklungsumgebung gehört immer:
  - **Workspace**

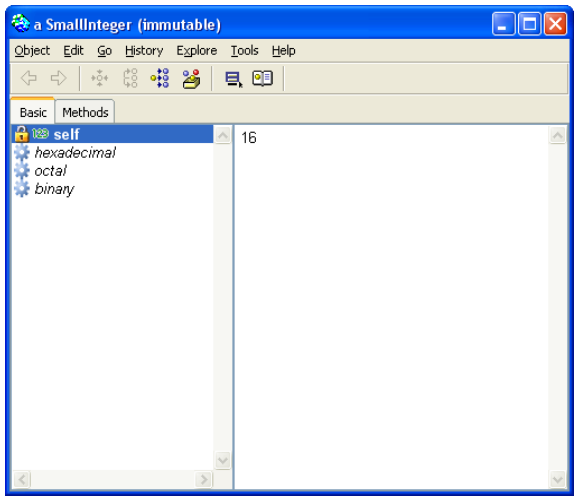
    eine Art Texteditor in dem auch Code ausgeführt werden kann
  - **Inspector**

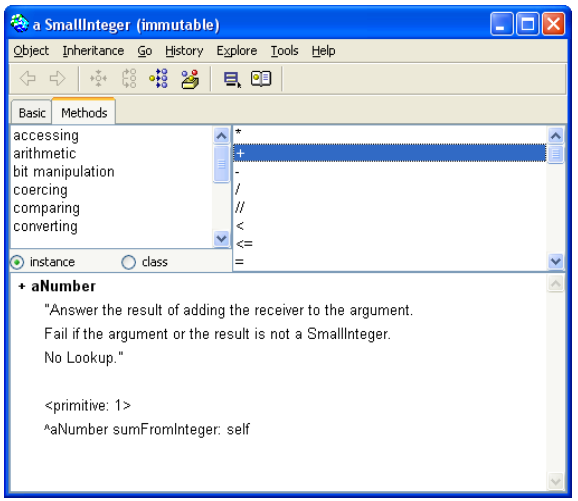
    mit ihm können Zustände von Objekten „*live*“ eingesehen werden
  - **Klassenbrowser**
  - **Debugger**

Smalltalk

Die Entwicklungsumgebung:

- **Der Inspector**
  - Visualisierung von Objekten
  - hilfreich bei der Fehlersuche






```
a := 5 + 3 * 2.

a inspect.
```

Smalltalk

Die Entwicklungsumgebung:

- **Die Fehlerbehandlung zur Compile-Zeit**
  - Syntaxfehler werden (erst) beim Compilieren erkannt/ markiert
- **Die Fehlerbehandlung zur Laufzeit**
  - Programmunterbrechung und Start des Debuggers
  - relativ häufig durch fehlende Typisierung und „Tippfehler“
- **Exceptions** (*nur kurz angeschnitten*)
  - Unterscheidung zw. **`Notification`** und **`Error`**
  - Unterscheidung zw. Lokalen Exceptions und globalen Exceptions
  - Vom Konzept grundsätzlich ähnlich wie in Java

Smalltalk

Zeit für ein Fazit!

Smalltalk

Stärken von Smalltalk:

- leichte und schlanke Syntax, lesbarer Code
  - ```
    buchstaben add 'D' after 'a'.
    ```
    ```
    buchstaben add('D', 'a');
    ```
- es ist (im Vergleich zu Java) weniger Code nötig
  - dafür sind Konventionen nötig
  - Kommentare zur Methodenbeschreibung sind Standard
- in sich konsistentes Sprachkonzept
- gut für große und komplexe Projekte
    ```

Smalltalk

Stärken von Smalltalk:

Wozu sollte man sich heutzutage noch mit einer fast dreißig Jahre alten Programmiersprache beschäftigen?

- Die Antwort auf diese Frage liegt eigentlich auf der Hand: Es gibt nur wenige Techniken auf dem IT-Markt die so viel Zeit hatten zu reifen.

Joachim Tuchel, objektfabrik.de

Smalltalk

Schwächen von Smalltalk sprachlich:

- eine Formalisierung für Protokolle fehlt
  - keine Interfaces
  - Convention-Over-Code
  - „sprechende Namen“: `Objekt meth: aNumber.`
- keine Packages
- feste Sichtbarkeiten
  - Instanzvariablen sind stets **`private`**
  - Instanzmethoden stets **`public`**
- keine statische Typisierung

Smalltalk

Schwächen von Smalltalk konzeptionell:

- schlecht für Teamarbeit
  - Smalltalk ist seiner Architektur nach eine IDE für einen Einzelnen
- sehr Ressourcen intensiv
- viele Dialekte, kein echter Standard
- optisch wenig ansprechende Entwicklungsumgebung („Feeling“)
  - *meine persönliche Meinung...*

Smalltalk

Smalltalk im Vergleich:

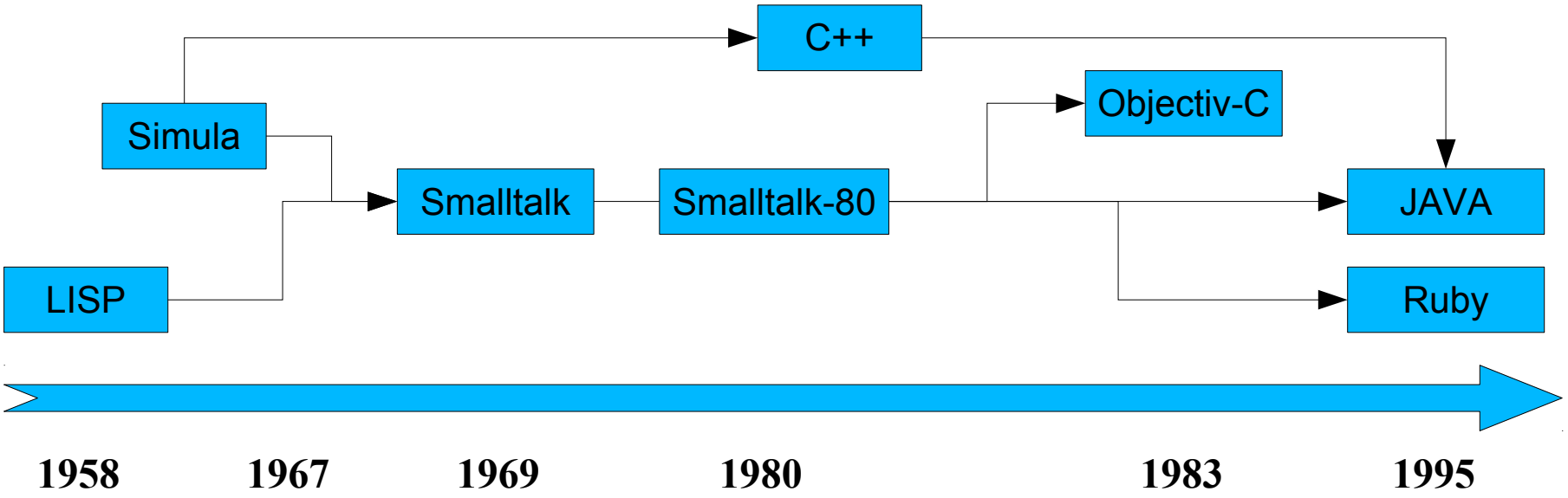
Sprache	CPU secs	Memory kB
C	11,95 sec	664 kB
Java	15,91 sec	12.564 kB
JavaScript	81,15 sec	13.880 kB
Smalltalk	112,19 sec	14.524 kB
Ruby	11 min	3.560 kB

- Benchmarktest (spectral-norm) von http://shootout.alioth.debian.org/
- besonders „*früher*“ ein echtes Problem für Smalltalk!

Smalltalk

Smalltalk heute:

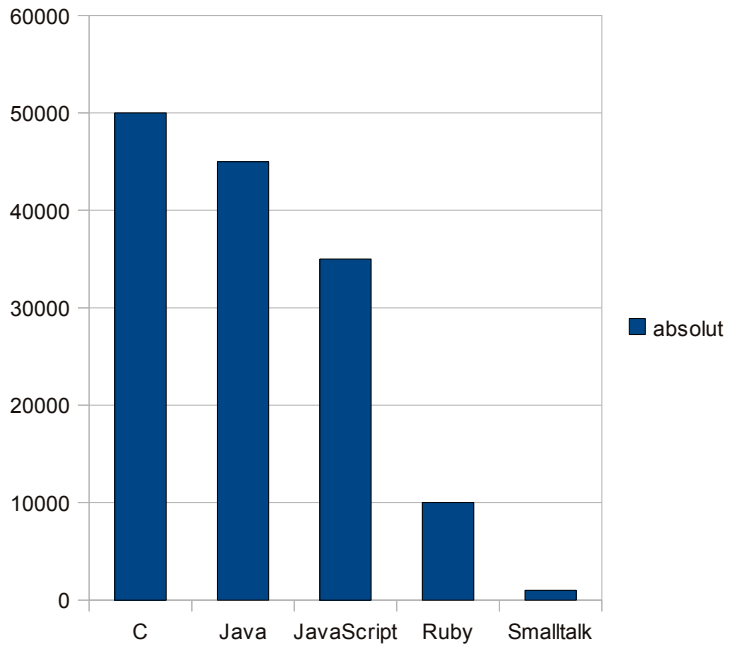
- keine praktische Relevanz mehr
- immer noch ein sehr hohes Potenzial
- wichtig für theoretische Betrachtungen und Lehre
- „Informatiker-Allgemeinbildung“, *denn*:

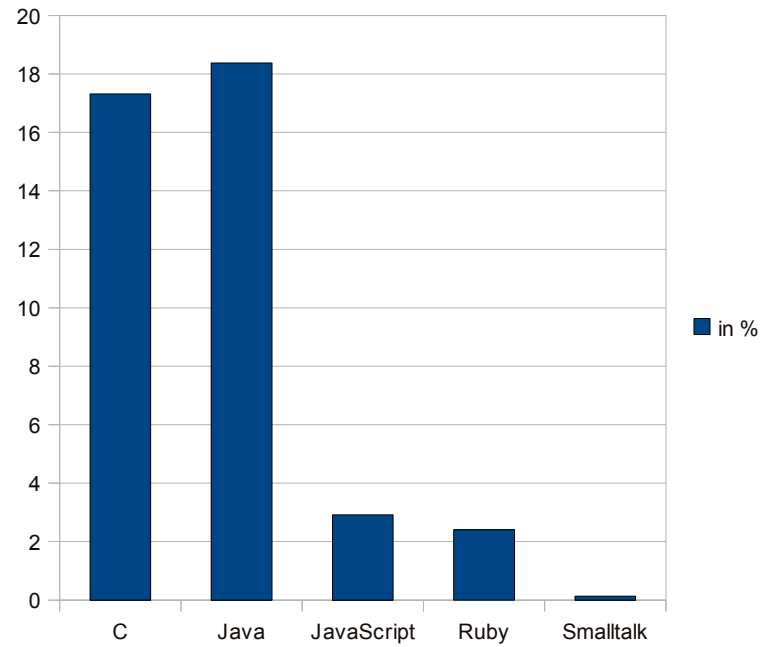


Smalltalk

Smalltalk heute:

- „*keine praktische Relevanz mehr*“, soll heißen:





- Laut http://langpop.com, gemessen an der Verwendung in OpenSource-Projekten
- Laut http://tiobe, gemessen an Entwicklern

Smalltalk

Quellen und Referenzen:

- **Joachim Tuchel:** Lebendes Objekt. c't, Heft 2/2003, S. 188-193.
- **Johannes Brauer:** Grundkurs Smalltalk. Teubner Verlag, 2009.
- **Josef Mittendorfer:** Objekt. Programmierung. Addison-Wesley, 1998
- **Wikipedia**. http://de.wikipedia.org/wiki/Smalltalk-80.

- VisualWorks von Cincom, unter: www.cincomsmalltalk.com

Zeit für Smalltalk...