

# Smalltalk FAQ

Chris Burkert

16. Juli 2003

# Inhaltsverzeichnis

<b>1</b>	<b>Einführung</b>	<b>3</b>
1.1	Smalltalk in a Nutshell . . . . .	3
1.1.1	Sprachkonzept . . . . .	3
1.1.2	Produktivität . . . . .	4
1.1.3	Wartbarkeit . . . . .	4
1.1.4	Portierbarkeit . . . . .	4
1.1.5	Zuverlässigkeit . . . . .	4
1.2	Smalltalk's Geschichte . . . . .	4
<b>2</b>	<b>Einsteigerfragen</b>	<b>6</b>
2.1	Welche Dateien benötige ich ? . . . . .	6
2.2	Was ist die 'Virtual Machine' ? . . . . .	6
2.3	Was ist das 'Image' ? . . . . .	6
2.4	Was sind die 'Sources' ? . . . . .	6
2.5	Was sind die 'Changes' ? . . . . .	6
2.6	Wo ist Main ? . . . . .	7
2.7	Ist Smalltalk ernst zu nehmen ? . . . . .	7
2.8	Ist Squeak ernst zu nehmen ? . . . . .	7
2.9	Wann und Wo sollte ich Smalltalk einsetzen ? . . . . .	7
<b>3</b>	<b>Die Sprache</b>	<b>9</b>
3.1	Allgemeine Fragen . . . . .	9
3.1.1	Gibt es einen Standart ? . . . . .	9
3.1.2	Wie schreibt man in Smalltalk Programme ? . . . . .	9
3.1.3	Was ist der 'Zuweisungsoperator' ? . . . . .	9
3.1.4	Was ist der 'Returnoperator' ? . . . . .	9
3.2	zentrale Begriffe der Sprache . . . . .	9
3.2.1	Was ist eine 'Variable' ? . . . . .	9
3.2.2	Was ist eine 'Pseudovvariable' ? . . . . .	10
3.2.3	Was ist eine 'Instanzvariable' ? . . . . .	10
3.2.4	Was ist eine 'Klassenvariable' ? . . . . .	10
3.2.5	Was ist eine 'Klasseninstanzvariable' ? . . . . .	10
3.2.6	Was ist ein 'Objekt' ? . . . . .	10
3.2.7	Was ist eine 'Methode' ? . . . . .	10
3.2.8	Was ist eine 'Klasse' ? . . . . .	11
3.2.9	Was ist 'Vererbung' ? . . . . .	11
3.2.10	Was ist eine 'Subklasse' ? . . . . .	11
3.2.11	Was ist eine 'abstrakte Klasse' ? . . . . .	11
3.2.12	Was ist eine 'Metaklasse' ? . . . . .	11
3.2.13	Was ist eine 'Instanz' ? . . . . .	11

3.2.14	Was ist 'Polymorphismus' ?	11
3.2.15	Was ist eine 'Message' ?	11
3.2.16	Welche Priorität haben Messages ?	12
3.2.17	Was ist 'Kaskadierung' ?	12
3.2.18	Was ist ein 'Literal' ?	13
3.2.19	Was ist 'Kapselung' ?	13
3.3	die Klassenhierarchie	13
3.3.1	Wie ist die Klassenhierarchie aufgebaut ?	13
3.3.2	Ist eine Klasse ein Objekt ?	13
3.4	reservierte Wörter	13
3.4.1	Was ist 'nil' ?	13
3.4.2	Was ist 'self' ?	13
3.4.3	Was ist 'super' ?	14
3.4.4	Was ist 'true' ?	14
3.4.5	Was ist 'false' ?	14
3.5	Blöcke	14
3.5.1	Was ist ein 'Block' ?	14
3.5.2	Wann wird der Wert eines Blockes bestimmt ?	15
3.5.3	Ist ein Block ein Objekt ?	15
3.5.4	Kann ich Blöcken Argumente übergeben ?	15
<b>4</b>	<b>Grafik und die Programmierumgebung</b>	<b>16</b>
4.1	die Tools	16
4.1.1	Was ist das 'Workspace' ?	16
4.1.2	Was ist das 'Transcript' ?	16
4.1.3	Was ist der 'System Browser' ?	16
4.1.4	Was ist der 'Hierarchy Browser' ?	16
4.1.5	Was ist der 'Method Finder' ?	16
4.1.6	Was ist der 'Inspector' ?	16
4.2	MVC	17
4.2.1	Was ist 'MVC' ?	17
4.2.2	Was ist ein 'Dependent' ?	17
4.2.3	Wie löse ich alle Abhängigkeiten ?	17
<b>5</b>	<b>Smalltalk's Einflüsse</b>	<b>18</b>
<b>6</b>	<b>Smalltalk-Implementierungen</b>	<b>19</b>
6.1	frei, OpenSource, non-commercial	19
6.2	kommerziell	19
<b>7</b>	<b>Smalltalk im Netz</b>	<b>20</b>
7.1	Webseiten	20
7.2	Newsgroups	20
7.3	Usergroups	20
7.4	Was ist ein 'Wiki' ?	20
<b>8</b>	<b>Wer setzt Smalltalk ein ?</b>	<b>21</b>
8.1	Produktion	21
8.2	Finanzwesen	21
8.3	Logistik	21

# Kapitel 1

## Einführung

Die objektorientierte Sprache Smalltalk ist nicht nur eine Programmiersprache sondern vielmehr eine integrierte, grafische und interaktive Programmierumgebung, die eine mächtige Basis für komplexe Applikationen zur Verfügung stellt. Der Umfang und die Bedienung lassen fast auf ein Betriebssystem schließen, was gar nicht so abwegig ist. Für viele ist es sogar eine Vision, eine Entwicklungsumgebung, basierend auf wenigen Konzepten. Nebenbei ... wenn ich hier von Smalltalk spreche, beziehe ich mich im Wesentlichen auf die freie Implementierung Squeak, da ich mit dieser die meiste Erfahrung habe und sich die Implementierung in den grundlegenden Prinzipien kaum unterscheiden.

Smalltalk wurde in den Siebziger entwickelt und ist mit Smalltalk-80 eines der heute noch erfolgreichsten Entwicklertools. Wer heute die Prinzipien von Java kennenlernt, wird merken, das vieles von Smalltalk abgeschaut wurde, aber im Original wesentlich besser implementiert ist. Dazu zählen Plattformunabhängigkeit, Objekt-Orientierung, Garbage-Collection und vieles mehr.

Das war aber noch nicht alles. Smalltalk bietet Unterstützung für Datenbanken jeder Art. Client/Server-Anwendungen profitieren von CORBA und ähnlichen Technologien. Auch XML und andere reguläre Sprachen lassen sich spielend leicht verarbeiten. Auch die Webservices sind ausgereift und bieten mehr als z.B. Java-Applets.

### 1.1 Smalltalk in a Nutshell

#### 1.1.1 Sprachkonzept

- pures Zeigerkonzept - Smalltalk arbeitet nur mit OOP's (Object Oriented Pointer).
- pure Objektorientierung - In Smalltalk ist alles ein Objekt mit dazugehörigen Methoden und der Klassendefinition.
- Polymorphismus - Unterschiedliche Objekte, die auf die gleichen Methoden reagieren, nutzen Polymorphismus um die korrekte Methode auszuwählen.
- Kapselung - Auf die Daten eines Objektes kann nur durch seine Methoden zugegriffen werden.
- Vererbung - Smalltalk unterstützt einfache Vererbung.
- Reflexivität - Smalltalk ist in Smalltalk implementiert.

- dynamisch getypte Sprache - Der Typ jedes Objektes steht erst zur Laufzeit zur Verfügung.

### 1.1.2 Produktivität

- Garbage Collection - Die Garbage Collection löscht nicht mehr benötigte Objekte. Mehr Zeit für das Wesentliche.
- simples Sprachkonzept - Die einfache Syntax erlaubt eine schnelle Einarbeitung in objektorientierte Techniken, ohne sich lang an der Sprache selbst aufzuhalten.
- schnelle Entwicklung - Die Applikation wird während sie läuft ohne Kompile- und Linkzyklen programmiert.
- leistungsfähige Tools - Die vielen Tools und Browser unterstützen den Programmierer.
- umfangreiche Klassenhierarchie - Klassen sind hierarchisch geordnet und bieten alle wichtigen Funktionen und leistungsfähige Datenstrukturen.
- Smalltalk vs. C++ - Offizielle Untersuchungen zeigen, das die Produktivität eines Smalltalk Entwicklers ca. 2 bis 4 Mal so hoch ist, wie die des C++ Entwicklers.

### 1.1.3 Wartbarkeit

- Wartbarkeit des Quellcodes - Wesentlich weniger Code in Smalltalk als in anderen Sprachen führt zu mehr Übersicht.
- Lesbarkeit des Quellcodes - Saubere und einfache Syntax dient der besseren Lesbarkeit.
- schnelle Änderungen - Die Konzepte von Smalltalk erlauben Änderungen in erstaunlich kurzer Zeit mit minimalem Aufwand.

### 1.1.4 Portierbarkeit

- Virtual Machine - Die Virtual Machine erlaubt eine vom Betriebssystem unabhängige Programmierumgebung.

### 1.1.5 Zuverlässigkeit

- Reife - Smalltalk ist seit 1980 erfolgreich in der Industrie in Verwendung. Die Sprache ist ausgereift, stabil und erfreut sich immer noch größter Beliebtheit.

## 1.2 Smalltalk's Geschichte

Grundgedanke für die Entwicklung von Smalltalk war es, ein System zu schaffen, welches das menschliche Denken und die verfügbare Hardware verbindet. Dieses System sollte einfach und effizient sein und mit seinen Einsatzgebieten mitwachsen können. Damit entstand auch die erste grafische Benutzeroberfläche, die später Apple und Microsoft inspirierte.

- 1960  
Die Programmiersprache Simula ist eine der ersten, die Prinzipien der Objektorientierung einführt. Dazu zählen Objekte, Vererbung und dynamische Bindung.

- 1970  
Am Xerox Palo Alto Research Center (Xerox PARC) gründet sich eine Forschungsgruppe unter der Führung von Alan Kay. Früchte dieser Arbeit sind Smalltalk-71, Smalltalk-72 (Objekte, Klassen, Messages), Smalltalk-74 (erweiterbare Klassen), Smalltalk-76 (Vererbung, die ersten Werkzeuge) und Smalltalk-78 (effiziente und portierbare Interpreter).
- 1983  
Smalltalk 80, der heutige Quasistandard, wird herausgegeben. Die erste verbreitete Implementierung von ParcPlace Inc., der Adele Goldberg vorsteht, ist noch heute in Cincom's VisualWorks eines der wichtigsten Werkzeuge eines Smalltalkers. Es ist die erste objektorientierte Programmiersprache mit leistungsfähiger Entwicklungsumgebung, Plattformunabhängigkeit durch die Virtual Machine, einem effizienten Speichermanagement durch die Garbage Collection, hochauflösender 2D Rastergrafik und der Unterstützung für Zeigergeräte wie die Maus. Das Problem ist noch die Ressourcenanforderung, welche die damalige Hardware nur teilweise befriedigen kann.
- 1986  
Smalltalk/V von Digital wird veröffentlicht. Auch werden mit verbesserter Hardware Smalltalkentwickler gesucht und die ersten erfolgreichen Projekte veröffentlicht. Smalltalk beeinflusst ausserdem viele Sprachen wie C++ oder Eiffel und leitete neue Wege in der künstlichen Intelligenz ein.
- 1993  
Das NCITS J20-Komitee wird gegründet, welches die Standardisierung von Smalltalk und seiner Klassenbibliothek zum Ziel hat. Unterstützt wird es vom Smalltalk Industry Council (STIC) das von den führenden Smalltalk-Produzenten gegründet wird. Resultat ist die Standardisierung von Smalltalk nach ANSI.
- 1995  
Bei Apple wird das Projekt Squeak ins Leben gerufen, das von Alan Kay und Dan Ingalls geleitet wird. Am 1 Oktober 1996 wird es in der Newsgroup comp.lang.smalltalk bekannt gegeben. Innerhalb weniger Wochen entstehen die ersten Unix und Windows Portierungen. Es steht heute unter vielen, teils exotischen Plattformen zur Verfügung.
- 1996  
ParcPlace und Digital fusionieren zur Firma ParcPlace-Digital welche 1997 zu Objectshare umbenannt wird. Ein Jahr später veröffentlichen sie ihr Produkt VisualWorks auch als non-commercial Version, das damit für den Heimgebrauch kostenlos zur Verfügung steht.
- 1999  
VisualWorks wird von Cincom aufgekauft. Es wird heute noch erfolgreich in der Version 7 vertrieben und beeinflusste entscheidend die Verbreitung von Smalltalk.

# Kapitel 2

## Einsteigerfragen

### 2.1 Welche Dateien benötige ich ?

Smalltalk wird im Wesentlichen in die 'Virtual Machine' und das 'Image' untergliedert. Desweiteren findet man noch die 'Sources' und die 'Changes'. Eine Smalltalkimplementierung wird, unter anderem, immer diese vier Komponenten als Dateien enthalten.

### 2.2 Was ist die 'Virtual Machine' ?

Die 'Virtual Machine' ist ein plattformabhängiges Programm im Binärformat, welches plattformunabhängigen Bytecode aus dem 'Image' interpretiert und ausführt. Es gibt VM's für fast jedes Betriebssystem und fast jede Rechnerarchitektur. Sogar für PDA sind VM's verfügbar.

### 2.3 Was ist das 'Image' ?

Das Image ist eine Datei, die aus dem gesamten Bytecode (die kompilierte Form der Methoden) und den Objekten des Programmes besteht. Diese Datei ist plattformunabhängig.

### 2.4 Was sind die 'Sources' ?

Da im 'Image' vorkompilierter Bytecode gespeichert ist und die Quellcodesourcen nur während der Entwicklung benötigt werden, werden sie in einer extra Datei gespeichert. Diese Datei wird in der Regel über ein Network Filesystem vielen Nutzern zur Verfügung gestellt.

### 2.5 Was sind die 'Changes' ?

Die Datei 'Changes' enthält alle persönlichen Änderungen, die im eigenen Image vorgenommen wurden. Sollte das System crashen, so kann man mit Hilfe dieser Datei die Arbeit bis kurz vor dem kritischen Punkt wiederherstellen.

## 2.6 Wo ist Main ?

Bei Smalltalk ist eine Method, Procedure oder Function, die als Einstiegspunkt in das Programm dient, nicht direkt spezifiziert. Dies ist auch prinzipiell nicht nötig, da das 'Image', exakt so, wie es gespeichert wurde, beim nächsten Start wiederhergestellt wird. Jedoch lässt sich z.B. für den Aufbau eines Sockets eine Methode definieren, die beim 'Start' des Programmes ausgeführt wird. Dies ist jedoch abhängig von der jeweiligen Smalltalkimplementierung. Squeak kann man beim Start z.B. ein Script übergeben (squeak squeak.image myScript.st) welches eingelesen wird (file in), oder in seiner Applikation die Klassenmethode MyApp>>startUp definieren und diese Klasse mittels SystemDictionary>>addToStartUpList: in den Startprozess einbinden.

## 2.7 Ist Smalltalk ernst zu nehmen ?

Wer in die Vergangenheit schaut, wird erkennen, das Smalltalk eine bemerkenswerte Erfolgsgeschichte zu verzeichnen hat. Wer in die Zukunft sieht und die Smalltalkprinzipien kennengelernt und nachvollzogen hat, wird erkennen, das Smalltalk für komplexe, große Projekte prädestiniert ist. Zum gegenwärtigen Zeitpunkt wird Smalltalk vor allem von Firmen aus den USA, aber auch mehr und mehr in Europa und Deutschland eingesetzt. Schwerpunkt ist dabei die Steuerung von Produktionsanlagen und Finanzsysteme wie sie in Banken eingesetzt werden. Wahrscheinlich wird in Bereichen, die auf optimiertes Laufzeitverhalten, oder wenige Byte an Code angewiesen sind, weiterhin Assembler und C den Ton angeben, weil damit ein maschinenoptimiertes Binary erzeugt werden kann <sup>1</sup>. Wer jedoch an Plattformunabhängigkeit, kurze Entwicklungszeiten und sehr gute Wartbarkeit des Quellcodes denkt, wird mit Smalltalk mehr als GUT bedient werden.

## 2.8 Ist Squeak ernst zu nehmen ?

Squeak hat es innerhalb weniger Jahre geschafft ein stabiles und umfangreiches System zu entwickeln, was andere Smalltalkimplementierungen, geschweige denn andere Sprachen, bei weitem noch nicht bieten können. Squeak ist vielleicht für den Anfang etwas gewöhnungsbedürftig, wenn man noch nie mit Smalltalk gearbeitet hat, aber es ist sicherlich dem System am nächsten, was Smalltalk einmal werden sollte, was nicht zuletzt damit zusammenhängt, das führende Köpfe in der Smalltalkgeschichte wie Alan Kay, Dan Ingalls und andere Pioniere aus dem Siebzigern an Squeak arbeiten.

## 2.9 Wann und Wo sollte ich Smalltalk einsetzen ?

Wem z.B. ein umfangreiches Softwareprojekt vorschwebt, dem sei hier Smalltalk empfohlen. Die folgenden Gründe könnten auch für dich ausschlaggebend sein.

- 'write once, run anywhere' ... Smalltalk ist von Grund auf plattformunabhängig und läuft bitidentisch auf allen Systemen. Eine unter Solaris geschriebene Applikation wird also auch unter Windows, BSD, Linux und vielen anderen Betriebssystemen laufen.
- Smalltalk ist strenge und kompromisslose Objekt-Orientierung.

---

<sup>1</sup>Produkte wie Smalltalk MT und Smalltalk/X erlauben es plattformabhängige und optimierte Bibliotheken und/oder Executables zu erstellen

- Smalltalk kommt mit einer Vielzahl an Entwicklungswerkzeugen. Angefangen bei verschiedensten Browsern über Debugger bis zu kleinen Helfern wie dem Method-Finder.
- Smalltalk ist sehr alt und damit ausgereift und stabil. Die ersten Implementierungen kamen in der 80ern heraus und haben Kinderkrankheiten, wie sie Java heute hat, schon längst aus dem Weg geräumt.
- Smalltalk ist sehr leicht zu lernen, weil es sich wie normale Sprache liest und nur 5 reservierte Wörter benötigt.
- Smalltalk schreibt man in der laufenden Applikation (im Image). Es sind keine Compile-Link Zyklen vonnöten.
- Smalltalk selbst ist in Smalltalk geschrieben. So kann man fast jeden Winkel des Systems zur Laufzeit begutachten, ändern und erweitern.
- Smalltalk besitzt die sogenannte 'Garbage-Collection'. Das bedeutet, das nicht mehr benötigte Objekte, vom System selbst aus dem Speicher entfernt werden. Damit hat der Entwickler mehr Zeit für das Wesentliche.
- Es macht Spaß !!!

# Kapitel 3

## Die Sprache

### 3.1 Allgemeine Fragen

#### 3.1.1 Gibt es einen Standard ?

Ja ... der ANSI Standard. Er läuft unter der Bezeichnung 'X3J20' des Smalltalk Standard Development Committee. Zu finden ist er in voller Schönheit unter:

`ftp://www.smalltalksystems.com/sts-pub/x3j20/`

Er beschreibt nur die Schnittstellen zwischen elementaren Objekten, keinerlei interne Implementierung und auch keine Grafikklassen.

#### 3.1.2 Wie schreibt man in Smalltalk Programme ?

Programme werden entwickelt, indem man Klassen definiert, Methoden hinzufügt und Instanzen miteinander kommunizieren lässt. Dazu öffnet man sich einen Browser legt Klassen an, fügt Instanz- und, wenn nötig, Klassenvariablen hinzu, definiert die Schnittstellen durch Methoden und instanziiert sie mit der Message **new**.

#### 3.1.3 Was ist der 'Zuweisungsoperator' ?

Der typische Zuweisungsoperator ist in Smalltalk `:=`. Mit ihm kann man die Speicheradressen von Variablen ändern und damit auf andere Objekte zeigen. Implementierungen, die sich am ursprünglichen Smalltalk orientieren (z.B. Squeak), verwenden zusätzlich einen Pfeil nach links `←`, der durch den Unterstrich `_` erreicht wird. Auch mehrfach Zuweisungen sind möglich: `a := b := c`

#### 3.1.4 Was ist der 'Returnoperator' ?

Der Rückgabewert einer Methode, durch `^` dargestellt, wird als Pfeil nach oben realisiert `↑`. Ist kein Rückgabewert definiert, wird innerhalb einer Methode immer `self` zurückgegeben.

### 3.2 zentrale Begriffe der Sprache

#### 3.2.1 Was ist eine 'Variable' ?

Eine Variable ist in Smalltalk ein 'Zeiger', also eine Referenz auf eine Speicheradresse. An dieser Adresse kann ein Objekt abgelegt werden. Bezeichner für Variablen bestehen aus

einer Folge von Buchstaben und/oder Zahlen die mit einem Buchstaben beginnen muss. Beginnt eine Variable mit einem Großbuchstaben, so ist sie global verfügbar (alle Klassen beginnen mit Großbuchstaben), beginnt sie mit einem Kleinbuchstaben, so ist sie nur im aktuellen Namensraum gültig. Jede neue Variable zeigt auf nil, ist also initialisiert.

### 3.2.2 Was ist eine 'Pseudovvariable' ?

Pseudovariablen sind z.B. die 5 reservierten Wörter (nil, true, false, self, super). Ihr Wert kann nicht durch Zuweisung geändert werden, da sie in einem Kontext immer genau ein Objekt repräsentieren.

### 3.2.3 Was ist eine 'Instanzvariable' ?

Dies sind Variablen, die von den Instanzen verwaltet werden. Sie sind privat, da man nur durch Instanzmethoden zugreifen kann.

### 3.2.4 Was ist eine 'Klassenvariable' ?

Dies sind Variablen, die von der Klasse verwaltet werden. Sie sind nicht ganz privat, da man neben Klassenmethoden auch durch Instanzmethoden zugreifen kann. Auch Klassen weiter unten in der Hierarchie teilen sich diese Variablen.

### 3.2.5 Was ist eine 'Klasseninstanzvariable' ?

Dies sind Variablen, die von der Klasse verwaltet werden. Sie sind privat, da man nur durch Klassenmethoden zugreifen kann und auch andere Klassen weiter unten in der Hierarchie ihre eigenen Klasseninstanzvariablen verwalten.

### 3.2.6 Was ist ein 'Objekt' ?

Ein 'Objekt' ist eine 'Ansammlung' von privatem Speicher, also Daten und zugehörigen Operationen also Methoden. Während bei der prozeduralen Programmierung Daten und Code getrennt sind, geht man in der Objektorientierung (OO) eine Abstraktionsebene höher und verbindet beide zu einem 'Objekt'. Die Daten sind dabei nach aussen hin nicht sichtbar und können nur indirekt durch Aufruf der Methoden verändert werden. Dies bedeutet also, das man den Zustand des Objekts, also seiner Daten, nur durch Aufruf seiner Methoden ändern kann. In Smalltalk besteht alles aus Objekten. Bezeichner von Objekten (Variablen) werden, wenn sie aus mehreren Worten bestehen, traditionell durch Großbuchstaben getrennt. Bsp.: **aSortedCollection**.

### 3.2.7 Was ist eine 'Methode' ?

'Methoden' sind im Prinzip Subroutinen, Funktionen oder Prozeduren; wie man sie auch immer nennen will, sie bleiben Vorschriften etwas zu tun. Sie sind immer zu einer Klasse zugehörig und können so indirekt über das Objekt referenziert werden. Eine Methode wird immer durch Senden einer Message an ein Objekt mit mindestens einem Parameter (**self**) aufgerufen. **self** wird jedoch immer, auch wenn es nicht im Quelltext steht, übergeben. Weiterhin liefert eine Methode einen Rückgabewert, was wieder ein Objekt ist; meist **self**. Bezeichner von Methoden (Messages) beginnen traditionell mit einem Kleinbuchstaben. Bsp.: **thisIsAMethod**.

### 3.2.8 Was ist eine 'Klasse' ?

Eine 'Klasse' ist eine Objektdefinition. Sie definiert zum einen die Daten, die ein Objekt enthalten kann und das Interface (die Messages), um auf das Objekt und die Klasse selbst zuzugreifen.

### 3.2.9 Was ist 'Vererbung' ?

'Vererbung' ist eine Möglichkeit bestimmte Fähigkeiten, die eine Klasse A schon hat, in Klasse B zu nutzen, ohne den Code zweimal zu schreiben. Smalltalk unterstützt einfache Vererbung, was bedeutet, das eine Klasse nur von **einer** anderen Klasse abgeleitet werden kann.

### 3.2.10 Was ist eine 'Subklasse' ?

Eine 'Subklasse' ist (relativ gesehen) eine Klasse, bei deren Definition eine andere Klasse erhalten musste. Die Subklasse 'erbt' alle Daten und Methoden der übergeordneten Klasse.

### 3.2.11 Was ist eine 'abstrakte Klasse' ?

Eine 'abstrakte Klasse' ist eine Klasse, die selbst nicht instanziiert wird, sondern von ihr nur ein andere, speziellere Klasse abgeleitet wird. Sie dient damit als eine Art Vorlage für eine gewisse Grundfunktionalität von mehreren Objekten.

### 3.2.12 Was ist eine 'Metaklasse' ?

Eine 'Metaklasse' ist ein Objekt, dessen Instanzen wiederum Klassen sind. Die Metaklasse antwortet auf die Klassenmethoden, die Klasse auf die Instanzmethoden. Metaklassen werden automatisch mit dem Erzeugen von Klassen angelegt.

### 3.2.13 Was ist eine 'Instanz' ?

Eine 'Instanz' einer Klasse ist einfach ein Objekt dieser Klasse. Sie wird in der Regel durch die Message **new** erzeugt.

### 3.2.14 Was ist 'Polymorphismus' ?

'Polymorphismus' heißt im Weitesten einem Begriff mehrere Bedeutungen zuzuodnen. In Smalltalk erkennt man dies, das z.B. unterschiedliche Zahlenformate, wie ganze und gebrochene Zahlen, das gleiche Zeichen für die Additon verwenden, nämlich '+'. Dies wird dadurch möglich, das jedes Zahlenformat eine Methode '+' in ihrem Interface implementiert hat, die auf die rechnerinterne Darstellung angepasst und optimiert ist.

### 3.2.15 Was ist eine 'Message' ?

Eine 'Message' ist im Prinzip der Aufruf einer Methode. Objekte kommunizieren über 'Messages'. Unterschieden wird dabei in drei verschiedene Arten von 'Messages':

- Unary Message - Sie besteht nur aus Empfänger und Methode.

```
myNumber squared.  
myString size.
```

- Binary Message - Sie besteht aus Empfänger, Methode (1 bis 2 nichtalphanumerische Zeichen als Bezeichner) und einem Parameterobjekt.

```
1 + 2.
myNumber <= 5.
```

- Keyword Message - Sie besteht aus Empfänger, Methode (mit Doppelpunkt) und Parameterobjekt(en).

```
myString at: 1 put: 'text'.
myCollection add: 'Hallo Welt'.
```

Diese Art der Parameterübergabe macht es damit in Smalltalk möglich, Programmcode zu schreiben, der sich selbst erklärt und wie normale Sprache gelesen werden kann, vorausgesetzt man wählt seine Bezeichner geschickt. Jetzt ist auch klar wie Smalltalk zu seinem Namen kam.

### 3.2.16 Welche Priorität haben Messages ?

Eine Zeile Smalltalk-Code wird (fast wie bei funktionalen Sprachen) in einer eindeutigen Reihenfolge abgearbeitet. Unary Messages haben immer Priorität, danach werden binary Messages abgearbeitet, danach Keyword Messages. Treten mehrere Messages desselben Typen auf, so werden diese von links nach rechts abgearbeitet. Darum ist es wichtig auf Klammerung zu achten. So ergibt z.B.  $2 + 3 * 4$  20 und nicht wie erwartet 14. Dies erhält man mit  $2 + (3 * 4)$ .

### 3.2.17 Was ist 'Kaskadierung' ?

Man kann auf einfache Art und Weise einem Objekt mehrere Messages senden indem man sie kaskadiert.

```
aStream
  nextPutAll: 'abc';
  nextPutAll: 'def';
  nextPutAll: 'ghi'.
```

” ist einfacher und übersichtlicher als: ”

```
aStream nextPutAll: 'abc'.
aStream nextPutAll: 'def'.
aStream nextPutAll: 'ghi'.
```

### 3.2.18 Was ist ein 'Literal' ?

Literale sind Zahlen, Symbole, Zeichen, Zeichenketten und Arrays, die vom Compiler und der Virtual Machine sofort in ein Objekt umgesetzt werden. Streng genommen, fallen damit auch die reservierten Wörter mit in die Definition. Beispiele:

```
" Beispiele für Literale in Smalltalk "  
  'Wie geht's'  " String "  
    #bla       " Symbol "  
    -12 4e3    " Integer "  
    -5.4       " Float "  
    5/4        " Fraction "  
    2r101      " Binäre Darstellung "  
    8r174      " Oktale Darstellung "  
    16rAF4     " Hexadezimale Darstellung "  
    $M         " Character "  
    #(1 2 3)   " Array aus Integern "  
    #((1 2 3) (4 5 6) (7 8 9)) " Array aus Arrays "
```

### 3.2.19 Was ist 'Kapselung' ?

'Kapselung' bedeutet, das die Daten des Objektes nicht von aussen verändert werden können. Dies kann nur über die Schnittstelle, die Methoden des Objektes, geschehen.

## 3.3 die Klassenhierarchie

### 3.3.1 Wie ist die Klassenhierarchie aufgebaut ?

Die Smalltalkklassenhierarchie ist wie ein Baum aufgebaut. Die Wurzel ist dabei die Klasse **Object**. Alle anderen Klassen sind Subklassen von **Object**, oder indirekt über weitere Subklassen von **Object** abgeleitet.

### 3.3.2 Ist eine Klasse ein Objekt ?

Ja. Auch wenn es schwer vorstellbar ist; Klassen sind Objekte weil sie Instanzen von Metaklassen sind. Immer dran denken. In Smalltalk besteht alles aus Objekten.

## 3.4 reservierte Wörter

### 3.4.1 Was ist 'nil' ?

**nil** steht für 'nichts' und ist eine Instanz der Klasse **UndefinedObject**. Wenn eine Variable den Wert **nil** hat, so bedeutet dies lediglich, das sie auf (noch) kein Objekt zeigt.

### 3.4.2 Was ist 'self' ?

**self** repräsentiert das Objekt selbst. So kann man in einer Methode das Objekt selbst referenzieren und damit eigene Methoden aufrufen.

### 3.4.3 Was ist 'super' ?

**super** repräsentiert auch das Objekt selbst. Der Unterschied zu **self** ist jedoch, das Methoden nicht zuerst in der Klassendefinition des Objektes gesucht werden, sondern in der Klasse-definiton, die über der Klasse steht, welche die Methode implementiert.

```
" Bsp.: Drei Klassen A, B und C. "  
" C ist von B abgeleitet und B von A. "  
" B implementiert eine Methode initialize die wie folgt definiert ist: "  
  
initialize  
  super doSomething.  
  
" Wird nun initialize an C gesendet, so wird die Methode in B gefunden. "  
" Diese sendet danach ein doSomething an A, da A die übergeordnete "  
" Klasse derer ist, in der die Methode implementiert war. "
```

### 3.4.4 Was ist 'true' ?

**true** bedeutet 'Ja'. Es definiert einen Ausdruck als wahr.

### 3.4.5 Was ist 'false' ?

**false** bedeutet 'Nein'. Es definiert einen Ausdruck als falsch.

## 3.5 Blöcke

### 3.5.1 Was ist ein 'Block' ?

Smalltalk-Blöcke sind vollwertige Objekte, die vor allem in Kontrollstrukturen genutzt werden. Ein Block kann eine Sequenz von 'aufgeschobenen' Anweisungen enthalten und ist immer von eckigen Klammern umschlossen. Aufgeschoben bedeutet hierbei, das die Anweisungen nicht gleich ausgeführt werden, sondern erst wenn dem Block direkt oder indirekt die Message **value** gesendet wird.

```
[i := i+1]  
  " Block ohne Ausführung der Anweisung "  
  
[i := i+1] value  
  " Block mit Ausführung der Anweisung "  
  
[] value  
  " Block vom Wert nil "  
  
[1 to: 10000 do: [ :i | i factorial ]] fork  
  " simpler Benchmark "  
  " indirektes Senden der Message value durch to:do: und fork "
```

### 3.5.2 Wann wird der Wert eines Blockes bestimmt ?

Blöcke werden erst ausgewertet, wenn ihnen die Message **value** gesendet wird. So kann man mit Blöcken leistungsfähige Kontrollstrukturen erzeugen.

### 3.5.3 Ist ein Block ein Objekt ?

Ja. Blöcke sind Instanzen der Klasse **BlockContext**. Man kann ihnen Variablen zuweisen und Messages senden wie jedem anderen Objekt.

### 3.5.4 Kann ich Blöcken Argumente übergeben ?

Ja. Squeak implementiert z.B. bis zu vier mal **value:** hintereinander. Hat man mehr, so ist es sinnvoll, die Argumente mit der Message **valueWithArguments:** in einer Collection zu übergeben.

```
" ein Argument "  
[ :i | i factorial ] value: 5.  
  
" zwei Argumente "  
[ :i j | i + j ] value: 5 value: 7.  
  
" mehr Argumente "  
[ :i j | i + j ] valueWithArguments: #(5 7).
```

# Kapitel 4

## Grafik und die Programmierumgebung

### 4.1 die Tools

#### 4.1.1 Was ist das 'Workspace' ?

Das Workspace ist ein Texteditor. Man kann ihn zum Testen von neuem Code einsetzen und ist damit nicht gezwungen alte Klassen zu verändern, oder extra neue zum Testen anzulegen.

#### 4.1.2 Was ist das 'Transcript' ?

Das Transcript ist ein spezielles Workspace, das Texte 'sammelt' und anzeigt. Es dient meist als Informationsmonitor um dem Nutzer Dinge mitzuteilen, für die es sonst keine Visualisierungsmöglichkeit gab. So werden System- und Statusmeldungen ausgegeben und der Nutzer somit auf dem Laufenden gehalten.

#### 4.1.3 Was ist der 'System Browser' ?

Der System Browser ist ein spezielles Workspace, um die vorhandenen Klassen und Methoden, sortiert nach Kategorien, im System anzuzeigen und sie zu verändern. Er ist das wichtigste Werkzeug um in Smalltalk zu programmieren. Mit ihm schreibt man neue Klassen und definiert ihr Interface.

#### 4.1.4 Was ist der 'Hierarchy Browser' ?

Dieser Browser stellt die entsprechende Klassen in ihrer Vererbungshierarchie dar. So kann man schnell geerbte Eigenschaften finden, abwandeln oder auch unterbinden.

#### 4.1.5 Was ist der 'Method Finder' ?

Im Method Finder kann man Methoden finden, von denen man die zugehörige Klasse nicht kennt und sich Polymorphismus in Smalltalk verdeutlichen.

#### 4.1.6 Was ist der 'Inspector' ?

Der Inspector dient unter anderem dem debuggen von Programmen. So kann man jederzeit einem Objekt die Message **inspect** senden und so Informationen über dessen Zustand

sammeln.

## 4.2 MVC

### 4.2.1 Was ist 'MVC' ?

'MVC', Model-View-Controller, ist ein Konzept, was nicht direkt von Smalltalk abhängt, jedoch die wichtigste Technologie ist, um Grafik mit Daten zu verbinden. Es ist seit jeher Grundlage für Grafik in Smalltalk und auch Java setzt seit den Swing-Klassen auf MVC.

Das Konzept besteht aus drei Grundbestandteilen. 'Model' beschreibt die Daten (z.B. die aktuelle Uhrzeit), 'View' die grafische Darstellung (z.B. die Uhr) und 'Controller' ein Objekt, welches beide synchronisiert. Klickt man auf die Uhr, so kann daraufhin das Model geändert werden; ändert sich die Zeit, so kann daraufhin das View geändert werden.

### 4.2.2 Was ist ein 'Dependent' ?

Ein 'Dependent' ist meist ein 'View'. Es ist ein Objekt, welches sich als abhängig von einem anderen Objekt, meist ein Model, erachtet und sich deshalb in dessen Abhängigkeitsliste registriert. Bsp.: **aModel addDependent: aView**

### 4.2.3 Wie löse ich alle Abhängigkeiten ?

Mit der Message **release**.

# Kapitel 5

## Smalltalk's Einflüsse

- **Objektorientierung**  
Viele Sprachen wie Java, C++, Python oder auch ObjektPascal haben die Prinzipien der Objektorientierung in ihr Konzept integriert.
- **Klassenbibliothek**  
Klassenbibliotheken ermöglichen die schnelle Wiederverwendbarkeit von schon geschriebenem Code. Jede Sprache hat mittlerweile Bibliotheken in Form von Klassen oder Modulen.
- **Plattformunabhängigkeit**  
Die starke Abstraktion von der Hardware bringt Vorteile, die auch Java zu erreichen versucht.
- **Garbage Collection**  
Wenn man sich nicht um den Speicher kümmern muss hat man mehr Zeit für das Wesentliche. Auch Javaprogrammierer profitieren davon.
- **MVC**  
Das Konzept des Model-View-Controller wurde von Smalltalkentwicklern ins Leben gerufen und ist seit dem nicht mehr aus der Softwaretechnik wegzudenken.
- **grafische Oberflächen**  
Die Oberflächen von Apple, Unix und Microsoft sind zweifelsohne von Smalltalk inspiriert worden.

# Kapitel 6

## Smalltalk-Implementierungen

### 6.1 frei, OpenSource, non-commercial

- Squeak  
<http://www.squeak.org>
- Gnu Smalltalk  
<http://www.smalltalk.org/versions/GNUSmalltalk.html>
- Little Smalltalk  
<http://www.smalltalk.org/versions/LittleSmalltalk.html>
- Pocket Smalltalk  
<http://www.smalltalk.org/versions/PocketSmalltalk.html>
- Cincom Smalltalk  
<http://www.cincom.com/smalltalk>
- IBM VisualAge Smalltalk  
<http://www-3.ibm.com/software/ad/smalltalk/>
- Smalltalk/X  
<http://www.exept.de>

### 6.2 kommerziell

- Cincom Smalltalk  
<http://www.cincom.com/smalltalk>
- Dolphin Smalltalk  
<http://www.object-arts.com/>
- GemStone Smalltalk  
<http://smalltalk.gemstone.com/>
- IBM VisualAge Smalltalk  
<http://www-3.ibm.com/software/ad/smalltalk/>
- Smalltalk/X  
<http://www.exept.de>

# Kapitel 7

## Smalltalk im Netz

Grün unterlegte Referenzen sind in deutscher Sprache.

### 7.1 Webseiten

- <http://www.smalltalk.org/>
- <http://www.whysmalltalk.com/>
- <http://www.goodstart.com/>
- <http://www.squeak.org/>
  - <http://minnow.cc.gatech.edu/squeak>
  - <http://swiki.squeakfoundation.org/squeak-ev>

### 7.2 Newsgroups

- <news:comp.lang.smalltalk>
- <news:comp.object>

### 7.3 Usergroups

- <http://www.gsug.org/>
- <http://www.esug.org/>

### 7.4 Was ist ein 'Wiki' ?

Ein 'Wiki' ist ein kollaborativer Webserver, das bedeutet, jeder kann Seiten direkt über den Browser erstellen und ändern. Da zu jeder Zeit vorhergehende Versionen der Seiten verfügbar sind, macht ein Angriff auf dieses scheinbar nicht vorhandene Sicherheitskonzept keinen Sinn. Das Wort 'Wiki' stammt von 'wiki wiki' aus dem Hawaiianischen und bedeutet 'schnell'. Der Wiki unterstützt neben allem, was auf Clientseite verarbeitet werden kann (HTML, Javascript, ...) auch eine Art vereinfachtes HTML, was je nach Wiki unterschiedlich sein kann. Ein 'Swiki' ist ein 'Wiki', der unter Squeak läuft.

# Kapitel 8

## Wer setzt Smalltalk ein ?

### 8.1 Produktion

- AMD's Steuerung und Kontrolle der Chipproduktion in Dresden ist vollständig in Cincom VisualWorks implementiert.
- Fuji Research Inc. automatisiert Herstellungsmechanismen mit Hilfe von Smalltalk.
- Die Produktion des VW Beetle in Mexiko wird von Smalltalk gesteuert.
- Die Produktion von Honda und Mitsubishi wird von Smalltalk gesteuert.
- Texas Instruments nutzt Smalltalk zur Steuerung der Waferbearbeitung.
- ...

### 8.2 Finanzwesen

- Die Bayerische Landesbank, die Landesbank Baden-Württemberg, die Bayerische Vereinsbank und viele andere nutzen Smalltalk.
- Daimler Chrysler nutzt Smalltalk in der Finanzverwaltung.
- ...

### 8.3 Logistik

- Ericsson Radio Systems nutzt Smalltalk im Bestell- und Auftragssystem mit seinen Lieferanten.
- Die Deutsche Bahn AG nutzt Smalltalk in der Logistik.
- FedEx koordiniert seine Lieferungen mit Smalltalkanwendungen.
- ...