

Alexander Ziegler

Programmierung mit Servlets und Applets in Java

Am Beispiel der grafischen Darstellung von Prozessdaten

Diplomarbeit

Bibliografische Information der Deutschen Nationalbibliothek:

Bibliografische Information der Deutschen Nationalbibliothek: Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de/> abrufbar.

Dieses Werk sowie alle darin enthaltenen einzelnen Beiträge und Abbildungen sind urheberrechtlich geschützt. Jede Verwertung, die nicht ausdrücklich vom Urheberrechtsschutz zugelassen ist, bedarf der vorherigen Zustimmung des Verlanges. Das gilt insbesondere für Vervielfältigungen, Bearbeitungen, Übersetzungen, Mikroverfilmungen, Auswertungen durch Datenbanken und für die Einspeicherung und Verarbeitung in elektronische Systeme. Alle Rechte, auch die des auszugsweisen Nachdrucks, der fotomechanischen Wiedergabe (einschließlich Mikrokopie) sowie der Auswertung durch Datenbanken oder ähnliche Einrichtungen, vorbehalten.

Copyright © 2001 Diplomica Verlag GmbH
ISBN: 9783832452414

Alexander Ziegler

Programmierung mit Servlets und Applets in Java

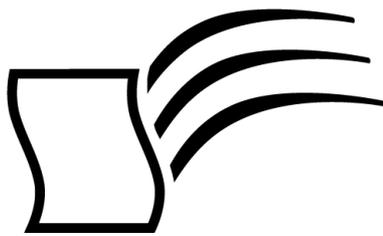
Am Beispiel der grafischen Darstellung von Prozessdaten

Alexander Ziegler

Programmierung mit Servlets und Applets in Java

Am Beispiel der grafischen Darstellung von Prozessdaten

Diplomarbeit
an der Fachhochschule Reutlingen
Fachbereich Wirtschaftsinformatik
Januar 2001 Abgabe



Diplom.de

Diplomica GmbH _____
Hermannstal 119k _____
22119 Hamburg _____

Fon: 040 / 655 99 20 _____
Fax: 040 / 655 99 222 _____

agentur@diplom.de _____
www.diplom.de _____

ID 5241

Ziegler, Alexander: Programmierung mit Servlets und Applets in Java: Am Beispiel der grafischen Darstellung von Prozessdaten / Alexander Ziegler - Hamburg: Diplomica GmbH, 2002

Zugl.: Reutlingen, Fachhochschule, Diplom, 2001

Dieses Werk ist urheberrechtlich geschützt. Die dadurch begründeten Rechte, insbesondere die der Übersetzung, des Nachdrucks, des Vortrags, der Entnahme von Abbildungen und Tabellen, der Funksendung, der Mikroverfilmung oder der Vervielfältigung auf anderen Wegen und der Speicherung in Datenverarbeitungsanlagen, bleiben, auch bei nur auszugsweiser Verwertung, vorbehalten. Eine Vervielfältigung dieses Werkes oder von Teilen dieses Werkes ist auch im Einzelfall nur in den Grenzen der gesetzlichen Bestimmungen des Urheberrechtsgesetzes der Bundesrepublik Deutschland in der jeweils geltenden Fassung zulässig. Sie ist grundsätzlich vergütungspflichtig. Zuwiderhandlungen unterliegen den Strafbestimmungen des Urheberrechtes.

Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Werk berechtigt auch ohne besondere Kennzeichnung nicht zu der Annahme, dass solche Namen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürften.

Die Informationen in diesem Werk wurden mit Sorgfalt erarbeitet. Dennoch können Fehler nicht vollständig ausgeschlossen werden, und die Diplomarbeiten Agentur, die Autoren oder Übersetzer übernehmen keine juristische Verantwortung oder irgendeine Haftung für evtl. verbliebene fehlerhafte Angaben und deren Folgen.

Diplomica GmbH

<http://www.diplom.de>, Hamburg 2002

Printed in Germany

Inhaltsverzeichnis

1	ZIEL DER STUDIENARBEIT	8
2	JAVA – DIE PROGRAMMIERSPRACHE	8
2.1	Geschichte von Java	8
2.2	Merkmale der Programmiersprache	9
2.2.1	Portabilität	9
2.2.2	Objektorientierung	9
2.2.3	Multithreading	9
2.2.4	Verteilte Programmierung	9
2.2.5	Robustheit	9
2.2.6	Sicherheit	9
2.2.7	Garbage Collector	10
2.2.8	Unterschiede zu C++	10
2.3	Das objektorientierte Konzept.....	10
2.3.1	Einführung	10
2.3.2	Das Konzept	11
2.3.3	Abstraktion.....	11
2.3.4	Botschaften.....	12
2.3.5	Kapselung.....	12
2.3.6	Klassen und Exemplare.....	12
2.3.7	Vererbung und Taxonomie.....	12
2.3.8	Polymorphie.....	12
2.3.9	Überladung	13
2.4	Objektorientierte Programmierung mit Java	13
2.4.1	Die Java Virtual Machine (JVM)	13
2.4.2	Der Unicode-Zeichensatz	13
2.4.3	Klassen	14
2.4.3.1	Definition	14
2.4.3.2	Zugriffsrechte	14
2.4.3.3	Modifier.....	14
2.4.4	Exemplare.....	15
2.4.4.1	Instanziierung.....	15
2.4.4.2	Konstruktoren	15
2.4.4.3	Vererbung.....	16
2.4.5	Interfaces	17
2.4.5.1	Allgemeines.....	17
2.4.5.2	Beschreibung	17
2.4.5.3	Modifier.....	18
2.4.6	Datenelemente	18
2.4.6.1	Beschreibung	18
2.4.6.2	Zugriffsrechte	19
2.4.6.3	Modifier.....	19
2.4.7	Methoden.....	20
2.4.7.1	Methodenkopf und -rumpf	20
2.4.7.2	Lokale Variablen.....	21
2.4.7.3	Methodenaufruf	21
2.4.7.4	Methodenüberladung	22
2.4.7.5	Zugriffsrechte	23
2.4.7.6	Modifier.....	23
2.4.8	Pakete.....	24
2.4.8.1	Allgemeines.....	24
2.4.8.2	Importieren von Paketen und Klassen	25

2.4.9	Archive.....	25
2.4.9.1	Beschreibung	25
2.4.9.2	Archivarten	25
3	JAVA DATABASE CONNECTIVITY (JDBC).....	27
3.1	JDBC – Allgemeines.....	27
3.2	Die Struktur von JDBC	27
3.3	Treiber.....	27
3.3.1	Der Treibermanager	27
3.3.2	Treiberarten	28
3.3.3	Aufgaben des Treibers	28
3.3.4	Laden eines Treibers.....	29
3.4	Die Datenbankverbindung	30
3.4.1	Das Connection-Objekt	30
3.4.2	Der JDBC-URL	30
3.4.3	Schließen einer Verbindung	31
3.4.4	Die Schnittstelle PooledConnection	32
3.5	Escape-Klauseln	33
3.6	SQL-Anweisungen.....	34
3.6.1	Das Statement-Objekt	34
3.6.2	Die execute-Methoden	34
3.6.3	Ergebnisse von Anfragen	35
3.6.3.1	Die Schnittstelle Statement	35
3.6.3.2	Die Schnittstelle PreparedStatement	36
3.6.3.3	Die Schnittstelle CallableStatement	37
3.6.4	Abfragen mit den Methoden der Schnittstelle ResultSet.....	38
3.6.5	Abbildung von SQL-Typen	39
3.6.5.1	Typabbildungen.....	39
3.6.5.2	getXXX-Methoden	41
3.6.5.3	setXXX()-Methoden.....	43
3.7	Transaktionen in JDBC	44
3.8	Unterschiede der JDBC-Versionen.....	45
4	APPLETS UND APPLIKATIONEN	46
4.1	Beispiel einer Applikation: <i>ApplicationTest.class</i>	47
4.2	Java-Applets erstellen.....	47
4.3	Sicherheit bei Applets	48
4.4	Der Lebenszyklus eines Applets.....	49
4.4.1	Init().....	49
4.4.2	Start().....	49
4.4.3	Stop().....	49
4.4.4	Destroy().....	49
4.5	Das Vererbungsdiagramm der Klasse Applet	51
4.6	Aufruf eines Applets aus der HTML-Seite.....	52
4.6.1	Informationen an Applets übergeben	53
4.6.2	Weitere Applet-Attribute	55

4.7	Grafikprogrammierung	55
4.7.1	Striche zeichnen	56
4.7.2	Figuren	56
4.7.3	Füllfiguren	56
4.7.4	3D-Figuren	56
4.7.5	Text zeichnen (Zeichensätze definieren)	56
4.8	Benutzeroberfläche des Applets	57
4.8.1	Die gebräuchlichsten AWT-Komponenten	57
4.8.2	Swing-Komponenten (Jcomponent)	59
4.8.3	Container	59
4.8.4	Layout-Manager	60
4.9	Multimedia mit Applets	60
4.10	Der Applet-Kontext	61
4.11	Kommunikation zwischen verschiedenen Applets	61
4.12	Objekte im Browser anzeigen	61
4.13	Zugriff auf JavaScript	62
4.14	Auftretende Probleme bei der Entwicklung und Ausführung von Applets	63
4.15	Doppelnutzung als Applet und Applikation	64
4.16	Signierte Applets / JAR-Dateien signieren	65
4.16.1	Signierte Intranet-Applets	65
4.16.2	Signierte Internet-Applets	67
4.17	Kommunikation über einen URL (Netzwerkverbindungen)	67
4.17.1	Grundbegriffe der Netzwerkprogrammierung	67
4.17.1.1	IP-Adresse	67
4.17.1.2	Hostname	67
4.17.1.3	Client	67
4.17.1.4	Server	67
4.17.1.5	Domain	67
4.18	Die Streams	69
4.19	Die URLConnection	70
5	SERVLETS IN JAVA	72
5.1	CGI – Skripte und Servlets	72
5.2	Aufgaben von Servlets	72
5.3	Vorteile von Servlets gegenüber den herkömmlichen CGI-Skripten	73
5.4	Der Kommunikationsprozess im WWW	74
5.5	Die Servlet – Engine	74
5.6	Der Lebenszyklus von Servlets	74
5.6.1	Die init()-Methode	74
5.6.2	Die service()-Methode	75
5.6.3	Das SingleThreadModel	75
5.6.4	Die destroy()-Methode	75

5.7	Datenfluss bei Applet – Server – Servlet – Kommunikation	77
5.8	Interaktion zwischen dem Server und dem Client.....	77
5.8.1	Aus der Sicht des Clients:	77
5.8.2	Aus der Sicht des Servers:.....	77
5.9	Grundstruktur der Servlets	78
5.9.1	Der Aufruf aus der HTML-Datei	79
5.9.2	Client-Anfragen bearbeiten – Formulardaten	79
5.9.3	Formulardaten aus einem Servlet lesen.....	80
5.9.4	Server Antwort generieren:.....	80
5.10	Cookies	83
5.10.1	Vorteile von Cookies.....	83
5.10.2	Probleme mit Cookies	83
5.10.3	Cookies vom Client lesen	84
5.10.4	Cookies mit einem angegebenen Namen finden	85
5.11	Sessions.....	85
5.11.1	HttpSession-Objekt zur akt. Anfrage nachschauen.....	85
5.11.2	Informationen zu einer Sitzung nachschauen	85
5.12	Applets als Frontend für Servlets	87
5.12.1	Daten mit GET senden und Ergebnisse direkt verarbeiten (HTTP-Tunneling).....	87
5.12.2	Serialisierte Datenstrukturen lesen	89
5.12.3	Daten mit POST senden und Ergebnisse direkt verarbeiten	89
5.13	JDBC (Java Database Connectivity)	91
5.13.1	Datenbanktreiber laden	91
5.13.2	Die Verbindungs-URL definieren.....	91
5.13.3	Die Verbindung herstellen mit getConnection().....	91
5.13.4	Anweisungen erzeugen	91
5.13.5	Abfrage ausführen	92
5.13.6	Ergebnisse verarbeiten.....	92
5.13.7	Verbindung schliessen	92
5.14	Threads	93
5.14.1	Was sind Threads.....	93
5.14.2	Zustände von Threads.....	93
5.14.3	Ausführen und starten des Threads	95
5.14.4	Verwendung von Threads in Applets	95
5.14.5	Synchronisierung von Threads.....	95
5.14.6	Deadlocks	97
6	PROJEKTDESCHEIBUNG.....	98
6.1	Randbedingungen für die einzutragenden Werte:	98
6.2	Das ER-Model.....	100
6.3	Darstellung mit Hilfe des Applet auf der Client-Seite	101
6.3.1	Daten eingeben und absenden	102
6.3.2	Ergebnisse in graphischer Form darstellen.....	104
6.3.3	Datenfluss der Applet – Server – Servlet-Kommunikation des Projektbeispiels.....	106
6.3.3.1	Aufruf der Datei AppletCon.class aus der HTML-Datei	107
6.3.4	Realisierung des Applet.....	108
6.3.4.1	Eingabedaten des Clients übernehmen und URL-Verbindung aufbauen.....	110
6.3.4.2	Ergebnisdaten einlesen.....	110
6.3.4.3	Maximalen Wasserstand ermitteln	112
6.3.4.4	Zeichnen der X-Achse.....	113
6.3.4.5	Zeichnen der Y-Achse.....	113
6.3.4.6	Zeichnung der XY-Werte bzw. Zeichnung des Graphen	114

6.3.4.7	Ablaufplan des Exchange-Sort-Verfahren ermitteln.....	115
6.4	Beschreibung der Serverseite bzw. der Servletverarbeitung	116
6.4.1	Datenbanktreiber laden und Datenbankverbindung herstellen	116
6.4.2	Parameterwerte aus der Umgebungsvariablen einlesen	117
6.4.3	Datenbankabfrage ausführen	117
6.4.4	Abfrageergebnis einlesen und Daten an das Applet senden	118
7	REALISIERUNG DES SERVLETS (ZUFALLSWERTE ERZEUGEN)	118
7.1	Aufgabendefinition	119
7.2	Programmablauf	119
7.3	Klassenbeschreibung	122
7.3.1	Allgemeines	122
7.3.2	servlet WerteErzeugen	122
7.3.3	Klasse Verarbeitung	128
7.3.4	Klasse Wasserstand	132
7.3.5	Klasse Zufallszahlen.....	133
7.3.6	Klasse DatumZeit	136
8	INSTALLATION UND INBETRIEBNAHME	138
9	GLOSSAR	139
10	LITERATURHINWEISE:.....	142

Abbildungsverzeichnis

1	Abb. Generationen der Programmiersprachen.....	11
2	Abb. Prozedurale und objektorientierte Programmierung	11
3	Abb. Java Virtual Machine	13
4	Abb. Datenbankanfrage in JDBC	27
5	Abb. Treiberarten.....	28
6	Abb. Connection-Pool	33
7	Abb. Applet und Applikation.....	46
8	Abb. Lebenszyklus eines Applets	50
9	Abb. Vererbungsdiagramm.....	51
10	Abb. AppletBeispiel	54
11	Abb. Lebenszyklus eines Servlets.....	76
12	Abb. Datenfluss bei Client – Server Kommunikation	77
13	Abb. Zustände von Threads	93
14	Abb. Nicht synchronisierte und synchronisierte Threads	96
15	Abb. Entity - Relationship - Model.....	100
16	Abb. Darstellung des Applets auf der Clientseite	101
17	Abb. Daten über Applet eingeben und Senden	102
18	Abb. Ausgabe (a) Applet	103
19	Abb. Auswertung der Daten in graphischer Form (a).....	104
20	Abb. Auswertung der Daten in graphischer Form (b).....	105
21	Abb. Datenfluss des Projektbeispiels.....	106
22	Abb. Exchange-Sort-Verfahren	115
23	Abb. Servletaufruf über HTML-Seite.....	119
24	Abb. Datenfluss des Servlets WerteErzeugen.....	120

1 Ziel der Studienarbeit

Ziel dieser Studienarbeit war es, Erfahrungen im Umgang mit der Programmiersprache im Umfeld des Internets zu sammeln. Im Besonderen wollten wir dabei auf die Bereiche Applets, Servlets und Datenbankanbindungen mit der JDBC-API eingehen. Dabei wurden folgende Themenbereiche intensiv bearbeitet:

- Darstellung von grafischen Elementen in Applets
- Kommunikation zwischen Applets und Servlets
- Aufruf von Servlets über einen Webserver
- Funktionen eines Webservers
- Aufbau von Datenbankverbindungen
- Auslesen von Daten aus einer Datenbank
- Schreiben von Daten in eine Datenbank

Außerdem konnten wir Erfahrung in der Projektplanung sammeln, was für zukünftige Arbeit an der Hochschule und auch im Praxissemester sehr wichtig ist. Es war das erste Mal während des Studiums, dass wir ein Projekt dieser Größenordnung bearbeiten konnten. Deswegen gestaltete sich die zeitliche Planung und die Abschätzung des Aufwands etwas schwierig. Gäbe es die Möglichkeit, bereits in früheren Semestern Softwareprojekte durchzuführen, wären uns einige Schwierigkeiten erspart geblieben. Allerdings muss sich unsere Studienarbeit am Ergebnis messen lassen und das ist für ein Einsteigerprojekt meiner Meinung nach mehr als ausreichend.

2 Java – Die Programmiersprache

2.1 Geschichte von Java

Die Geschichte der Programmiersprache Java begann bereits im Jahre 1990, als ein Entwicklungsteam um James Gosling (er entwickelte den Texteditor Emacs) bei Sun Microsystems mit der Planung für eine neue Programmiersprache mit Namen **Oak** beauftragt wurde. Diese Programmiersprache wurde eigentlich nicht für den Einsatz im Internet, sondern zur Steuerung von Consumer-Electronic-Geräten, wie Fernseher und Stereo-Anlagen, aber auch Toaster oder Waschmaschinen, konzipiert. Leider setzte sich diese Sprache nie durch. Die Eigenschaften der neuen Programmiersprache (Plattformunabhängigkeit, geringer Umfang des Quellcodes) waren aber für einen anderen Einsatzzweck geradezu prädestiniert: Software für das World Wide Web.

Nachdem man die Sprache weiter verbessert und in **Java** umbenannt hatte, stellte Sun die Technologie im Jahr 1995 der Öffentlichkeit vor und entwickelte gleichzeitig einen Web-Browser (**HotJava**), der die Java-Virtual-Machine, die den Java-Bytecode interpretiert, implementiert hatte. Jetzt war es möglich, kleine Java-Programme, die sogenannten **Applets** im Browser auszuführen. Diese Mini-Applikationen ermöglichten es, die einstmaligen starren Web-Seiten auf der Basis von HTML lebendiger zu machen. Als dann Netscape mit dem Navigator 2.0 Ende 1995 nachzog und die JVM ebenfalls integrierte, hatte Java endgültig seinen Durchbruch.

Heute wird Java nicht nur für client-seitige Applets eingesetzt, sondern ebenso zur Datenbankanbindung und zur Programmierung in verteilten Systemen. Java ist eine vollwertige Programmiersprache, mit der sich alle Arten von Anwendungen realisieren lassen. Weil es aber seine Vorteile in Netzwerken voll ausspielt, wird es hauptsächlich dort eingesetzt. Welche Vorteile das sind und welche Merkmale Java des Weiteren besitzt, darauf soll in den nächsten Abschnitten eingegangen werden.

2.2 Merkmale der Programmiersprache

2.2.1 Portabilität

Java ist plattformunabhängig. D. h. es ist möglich, Java-Code der nach den Konventionen der Programmiersprache implementiert wurde, auf beliebigen Betriebssystemen auszuführen. Dies ist eine der wichtigsten Forderungen für den Einsatz im Internet.

Dies erreicht Java auf zweierlei Arten. Zum einen wird der Java-Code auf jeder Plattform mit Hilfe der Java-Virtual-Machine (JVM) interpretiert. Ein Java-Programm wird dafür zuerst in sogenannten Bytecode kompiliert und bei Ausführung von der JVM übersetzt. Zum anderen sind Datentypen in Java ebenso unabhängig von der jeweiligen Implementierung.

2.2.2 Objektorientierung

Java ist vollständig objektorientiert aufgebaut (im Gegensatz zu C++). So deckt es die Forderungen an objektorientierte Programmiersprachen, wie die Kapselung, die Vererbung, die Polymorphie und die Operatorüberladung weitestgehend ab.

2.2.3 Multithreading

Java unterstützt die gleichzeitige Verarbeitung von Prozessen. Oft ist es notwendig, verschiedene Aufgaben in einem Programm gleichzeitig auszuführen. Ein Beispiel hierfür sind Applets, die Eingaben des Benutzers am Client entgegenzunehmen, während sie auf eintreffende Nachrichten vom Server warten. Früher wurden solche Probleme mit Hilfe von Schleifen gelöst, was aber Schwierigkeiten macht, weil die Tasks nicht mehr sauber voneinander getrennt werden können. Mit sogenannten **Threads of Control** (Threads = Fäden) können diese Aufgaben effizienter und einfacher gehandhabt werden. Praktisch läuft folgendes ab: in einem Prozess können beliebige Threads gestartet werden, die keine eigenen Prozesse mehr sind, sondern Teilprozesse, die im selben Speicher arbeiten.

2.2.4 Verteilte Programmierung

Java ist für die Programmierung in verteilten Systemen entwickelt worden. Es nutzt TCP/IP als Protokoll, wobei die Kommunikation entweder über Sockets, CGI-Programme (Servlets) oder Remote Method Invocation (RMI) erfolgt.

2.2.5 Robustheit

Java ist eine sehr robuste Programmiersprache, weil bewusst auf unsichere Elemente verzichtet wurde. Im Gegensatz zu C++ verzichtet Java ganz auf den Einsatz von Pointern. Die strenge Einhaltung der Objektorientierung und die Ausklammerung von Operatorüberladung, multipler Vererbung und automatischer Typumwandlung, tragen einen Großteil dazu bei.

2.2.6 Sicherheit

Ein mindestens ebenso wichtiger Punkt, wie die Portierbarkeit, ist die Forderung nach der Sicherheit von Java-Code. Dies gilt vor allem für Applets, die auf dem Client-Rechner ausgeführt werden. In Java wurde deshalb ein weitestgehender Schutz vor Viren und Zugriffen auf die Festplatte des Clients implementiert. Dies wird hauptsächlich durch den Einsatz der JVM erreicht. Sie unterbindet den Zugriff auf sicherheitskritische Ressourcen und führt vor dem Ablauf des Programms eine **Bytecode-Verifizierung** durch.

2.2.7 Garbage Collector

In Java gibt es nur sehr vereinzelt eine explizite Speicherfreigabe, weil sie sehr fehleranfällig ist. Java verwendet den **Garbage Collector**, der in die JVM integriert ist und sorgt während der Laufzeit dafür, dass nicht mehr benötigte Objekte aus dem Speicher gelöscht werden.

2.2.8 Unterschiede zu C++

Java ist mit Syntax und einigen Konventionen an C/C++ angelehnt. Allerdings gibt es einige wichtige Unterschiede. Java wurde auf Sicherheit und für den Einsatz im Internet konzipiert. In Java enthalten sind deshalb nicht:

- Typedefs, Defines und der Präprozessor
- Strukturen und Unions
- Funktionen außerhalb von Klassen
- Multiple Vererbung
- Goto-Befehle
- Automatische Typumwandlung
- Pointer
- Manuelle Speicherverwaltung

2.3 Das objektorientierte Konzept

2.3.1 Einführung

Das objektorientierte Programmierparadigma wurde in den späten Sechziger Jahren bzw. Anfang der Siebziger Jahre mit den Sprachen **Simula** und **Smalltalk** entwickelt. Man versprach sich damals eine einfachere Modellierung von Problemen der realen Welt in Programmen. Mittlerweile hat sich das objektorientierte Konzept durchgesetzt. Welche Gründe gab es für den enormen Aufschwung der objektorientierten Programmierung in den Neunziger Jahren?

Zu Beginn des Computerzeitalters waren die Sprachen maschinenorientiert, d.h. ganz nah an der Hardware ausgerichtet. Damit konnten die Eigenschaften der Bauteile sehr effizient genutzt werden, aber aufgrund dieser Nähe konnten keine komplexeren Programme entwickelt werden.

Anwendungsorientierte Sprachen wie Algol, Cobol oder Fortran sollten die Programmierung vereinfachen und schnellere Software-Entwicklung ermöglichen. Doch diese frühen Programme waren unstrukturiert und wenig effizient.

Bessere Ergebnisse versprach man sich von der Einführung von strukturierten Programmiersprachen wie Pascal oder C. Diese Art der Programmierung brachte echte Vorteile denn sie brachte übersichtlichere Programme und schuf die Möglichkeit, komplexere Sachverhalte zu implementieren. Leider kam man durch den prozeduralen Programmierstil dem eigentlichen Problem (dem Was) nicht näher, sondern entfernte sich immer weiter von der realen Problemlösung. Es stand das Konzept (das Wie) im Vordergrund, nicht aber die Modellierung eines Weltausschnitts.

Die objektorientierte Programmierung geht ein Stück weiter in die Richtung, die semantische Lücke zwischen Wirklichkeit und Code zu schließen. Es ist möglich ein reales Problem mit Objekten zu implementieren, die Eigenschaften und Fähigkeiten besitzen, untereinander kommunizieren und ihre Eigenheiten weitervererben können. Damit können sehr komplexe Sachverhalte abgebildet werden und die Flexibilität ist hoch genug, um mit