Java ist heiß

Das Lieblingsgetränk hyperaktiver Programmierer stand Pate: Java ist die altenglische Bezeichnung für Kaffee. Die gleichnamige Programmiersprache gilt als Muntermacher für das Internet. CHIP erläutert das Konzept.

Der Geist aus der Flasche hat einen modernen Bruder bekommen: das Java-Männchen aus der Tasse. In der Sprache Java geschriebene Programme werden in absehbarer Zeit auf allen Internet-fähigen PC ihre Wunder vollbringen.

Bislang wurden die bunten Web-Seiten mit Grafiken, Tabellen und Audio aufgepeppt. Doch um Internet-Anwendungen, die ein hohes Maß an Interaktion verlangen, war es bis vor kurzem schlecht bestellt. Gängige Applikationen liefen auf einem Internet-Server ab, der bei gleichzeitigem Zugriff vieler Anwender in die Knie ging. Die oft umfangreichen Ausgaben der Anwendung mußten übers Internet immer aufs neue um die halbe Welt geschickt werden. Einen Ausweg aus diesem Dilemma bringt Java.

Die Software-Tochter der Firma Sun entwickelte die Programmiersprache ursprünglich für interaktives Fernsehen und Set-Top-Boxen. Der Name stammt vom altenglischen Wort für Kaffee, dem Standardgetränk der Programmierer.

Seit 1995 treten Java-Applikationen ihren Siegeszug im Internet an. Java-Interpreter, also die Programme, die den Java-Code ausführen, sind in die Web-Browser, etwa in den Netscape-Navigator und den Internet-Explorer von Microsoft, integriert. Viele Betriebssystemhersteller planen ebenfalls, Java in ihre Produkte zu implantieren. Besonders flott schreitet dabei Sun mit seinem Java-Betriebssystem JavaOS vornweg.

Java-Programm-Module, sogenannte Applets, sind in HTML-Seiten integriert. Mit einem Java-fähigen Web-Browser ausgerüstet, braucht sich der Nutzer mit keinen technischen Details auseinanderzusetzen. Die Java-Programme werden einfach innerhalb des Browsers ausgeführt.

Applets sind architekturunabhängig

Ein Java-Programmierer entwickelt auf seiner Maschine den Quellcode. (Quelldateien sind an der Endung *.java* zu erkennen.) Nach einer Sicherheitsverifizierung übersetzt der Java-Compiler daraus einen maschinenunabhängigen Byte-Code (s. Grafik S. 270). Dieser Java-Byte-Code ist kein ausführbares Programm, sondern repräsentiert die Abfolge des Programms (Dateien mit Java-Code haben die Dateiendung *.class*).

Auf der Benutzerseite setzt ein Java-Interpreter diese Byte-Folge in ein ablauffähiges Programm um. Java ist also für die interpretative Ausführung konzipiert. Das hat den Vorteil, daß Java-Applikationen unabhängig von der Zielplattform sind. Einen Java-fähigen Browser vorausgesetzt, ist das Applet vom Großrechner bis zur Spielekonsole einsetzbar. Der Benutzer muß sich nur einmal in den Besitz eines plattformabhängigen Java-Interpreters kommen. Den ersten Interpreter *Hotjava* entwickelte ebenfalls die Firma Sun. Hotjava ist selbst in Java geschrieben.

Der Nachteil des interpretativen Konzepts ist die geringe Abarbeitungsgeschwindig-

Fiorito

keit. Diesem Umstand wirken Just-In-Time-Compiler (JIT) entgegen. Beim Laden des Programms erzeugen sie durch Kompilierung aus dem architekturneutralen Java-Zwischencode einen plattformspezifischen Binärcode. Dieser Code wird dann um ein Vielfaches schneller abgearbeitet, als es mit einem Interpreter möglich wäre.

0 Objektorientierte Programmiersprache

Java ist bezüglich des Sprachkonzepts sehr an C++ angelehnt. Java erhebt jedoch den Anspruch, einfacher erlernbar zu sein als sein großer Bruder. C++ und Java sind objektorientierte Programmiersprachen. Im Unterschied zum herkömmlichen, imperativen Sprachkonzept (erkennbar an Funktionen und Prozeduren) sind hier Objekte, Klassen, Vererbung, Methoden und Instanzen implementiert.

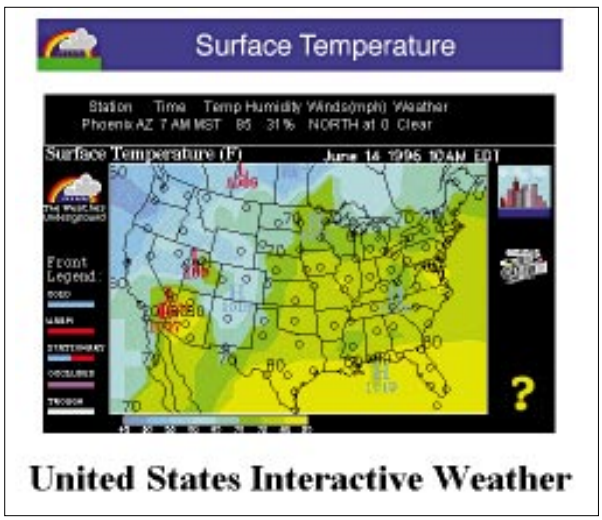
Ein Java-Programm ist ein Objekt, das wiederum aus Objekten besteht. Die Eigenschaften der Objekte werden durch Klassen beschrieben. Jede Klasse definiert eine Anzahl Methoden und Variablen. Die Methoden eines Objekts sind das Pendant zu den Prozeduren der imperativen Sprachen. Sie steuern die Handlungsweise des Objekts.

Klassen bilden üblicherweise Klassenhierarchien. Innerhalb der Hierarchie erbt eine Unterklasse die Methoden und Variablen der Überklasse. Alltagsbeispiel: Die Überklasse „Computer-Magazine“ besitzt die Methoden „Beschäftigung mit Computern“ und „Kioskverkauf“. Leiten wir die Unterklasse „CHIP“ aus der Klasse „Computer-Magazine“ ab, erbt „CHIP“ die Eigenschaften „Beschäftigung mit Computern“ und „Kioskverkauf“. Die Deklaration der Unterklasse in Java würde so aussehen: class CHIP extends Computer-Magazine { Deklaration der Methoden }

Die Beziehung zwischen Ober- und Unterklasse ist gerichtet. Um bei dem Beispiel zu bleiben: Jede „CHIP“ ist ein „Computer-Magazin“; aber nicht jedes „Computer-Magazin“ eine „CHIP“. Die Vererbung ist eine der herausragenden Fähigkeiten objektorientierter Sprachen.

In manchen objektorientierten Programmiersprachen gibt es die sogenannte Mehrfachvererbung. Eine Klasse kann so von mehreren Klassen Eigenschaften erben. Das kann im Einzelfall die Effizienz steigern, macht aber den Code sehr unübersichtlich. Die Entwickler von Java sahen deshalb keine Mehrfachvererbung direkt vor.

Mit Pauken und Trompeten: Laufschrift auf einer HTML-Seite ist eine häufig anzutreffende Anwendung von Java. Hier ist die Seite zusätzlich mit Musik untermalt.



Wetterkarte: Dank Java-Applet reicht es, den Cursor über einer Stadt zu positionieren, und die Klimadaten werden angezeigt. Auch Zoomen ist möglich.

Java bietet dafür das Konzept der Interfaces. Mit ihnen können Mehrfachvererbungen abgebildet werden. Interfaces sind abstrakte Klassen. Die Methoden dieser Klassen werden deklariert, aber nicht implementiert. Der Klassen-Vererbungsmechanismus ist auch auf Interfaces anwendbar.

0 Java-Syntax

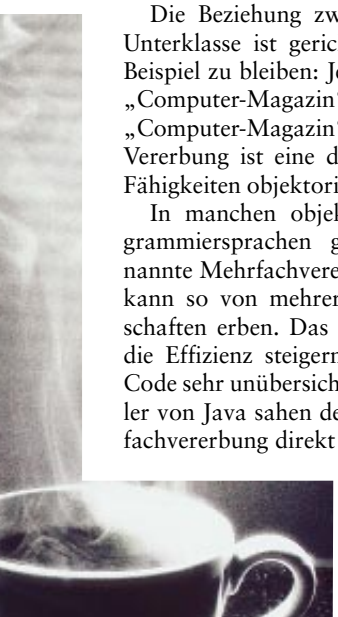
Wie schon erwähnt, lehnt sich die Syntax stark an C++ an. Java ist eine typisierte Programmiersprache. Variablen und Ausdrücke besitzen einen festen Typ. Während des Compiler-Laufes wird diese Syntax getestet. Bei den Variablen gibt es einfache Typen wie *int* (integer = ganze Zahl) und *float* (floating point = Gleitkommazahl). Felder zählen nicht zu den einfachen Typen, sie sind Objekte. Sie sind eine Ansammlung einfacher, gleicher Variablen und können in Java nur dynamisch (mit dem Schlüsselwort *new*) angelegt werden.

Die strenge Typisierung zwingt den Programmierer zu sauberer Arbeit. Semantische Fehler, die bei der automatischen Typkonvertierung in C und C++ zu den häufigsten Fehlerursachen zählen, werden so vermieden.

Direkte Zeiger und Zeigerarithmetik sind in Java, anders als in C++, nicht implementiert. Das führt zu einer deutlich besseren Lesbarkeit der Quelltexte und eröffnet keine Sicherheitslücken.

Ausdrücke liefern Werte eines bestimmten Typs. Ausdrücke bestehen aus sogenannten Operanden und Operatoren. Ein Beispiel: Der Ausdruck *13+7* enthält die Operanden *13* und *7* sowie den Additionsoperanden. Das Ergebnis *20* ist typisiert, hier der Typ Integer. Die Zuweisung des Ergebnisses eines oder mehrerer Ausdrücke an eine Variable wird als „Anweisung“ bezeichnet (*a=b*3*).

Natürlich gibt es auch in Java eine ganze Zahl von Schlüsselwörtern, welche die Programmabarbeitung steuern. Die *if*- und die *switch*-Anweisungen verzweigen die Programmausführung. Mit *for*, *while* und *do-while* lassen sich Schleifen programmieren.



Mehrere Threads

Moderne Betriebssysteme arbeiten im Multitasking-Betrieb. Mehrere Programme werden quasi zeitgleich ausgeführt. Eine Form des Multitaskings ist das Multithreading. Dabei werden autonome Teile eines Programms, die Threads, zeitgleich ausgeführt. Der Vorteil liegt in der Zeitersparnis bei der Programmabarbeitung: Während auf eine Benutzer-Interaktion gewartet wird, kann zeitgleich eine Berechnung oder eine Animation durchgeführt werden.

Die Sprache Java unterstützt die multithreaded Programmierung. Zur Erzeugung eines Threads in Java gibt es zwei Alternativen, und zwar erstens durch Implementierung des Interfaces *Runnable* und Instantiierung eines Objekts der Klasse *Thread*; zweitens durch die Definition einer Subklasse aus der Klasse *Thread* und Überschreiben der Methode *Run()*.

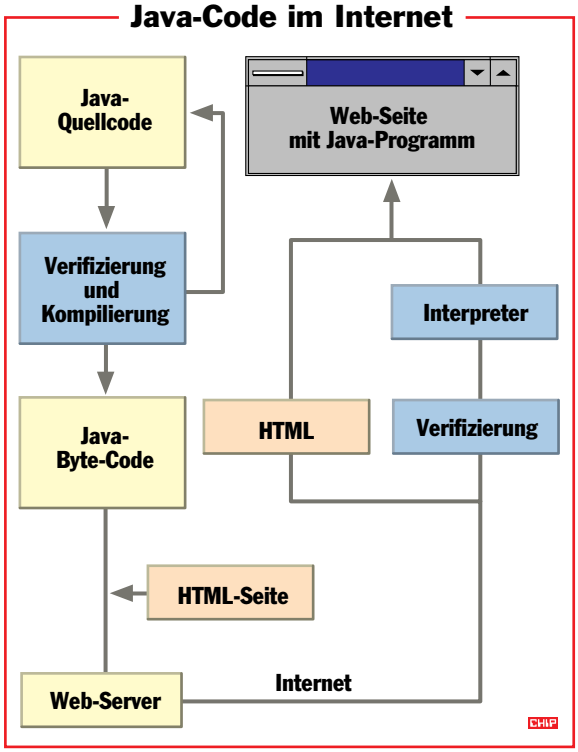
Ein Steuerprogramm läßt den Thread so lange laufen, bis der die Kontrolle selbst abgibt (per *sleep()* oder *yield()* oder *wait()*) oder von einem Thread mit höherer Priorität verdrängt wird.

In Java gibt es leistungsfähige Synchronisationsmechanismen, welche die Ablaufsteuerung der Threads untereinander regeln. Da der Ablauf der Threads untereinander nicht vorhersagbar ist, können Inkonsistenzen auftreten.

Diesem Umstand begegnet Java durch die Synchronisation des Ablaufs der Threads. Methoden werden dazu als *synchronized* deklariert. Wird eine solche Methode durch einen Thread ausgeführt, ist das Objekt für weitere Zugriffe gesperrt. Eine dritte Methode, die einen Zugriff wünscht, muß in diesem Falle warten. Nach der Beendigung der Methode wird die Sperrung wieder aufgehoben.

Der Nachteil dieser Vorgehensweise ist das mögliche Auftreten von Deadlocks. Das sind logische Verklemmungszustände, die dann auftreten, wenn mehrere Threads wechselseitig aufeinander warten müssen.

Java-Code im Internet



Sicherheitsaspekte

Die Abarbeitung fremden Codes birgt auch Gefahren. So könnten findige Virusprogrammierer auf die Idee kommen, einfach übers Internet auf die Rechner von Millionen Anwendern Viren zu schleusen. Auch könnten relevante Daten, zum Beispiel brisante Dokumente, ausgespäht werden.

Java begegnet dieser Gefahr auf mehrere Weise. Wie schon erwähnt, erlaubt die Struktur der Sprache (strenge Typisierung, keine Zeigerarithmetik) keinen versteckten Aufruf von Routinen. Außerdem überprüft der Compiler in der Entwicklungsphase der Applets, ob potentiell schädlicher Code vorhanden ist, und verhindert die Kompilierung im Fall des Falles.

Sollte sich ein Programmierer einen eigenen Compiler „stricken“, der diesen Sicherheitsmechanismus nicht enthält, greift der nächste Teil des Java-Sicherheitskonzepts: Auf der Anwenderseite wird der Java-Byte-Code beim Laden verifiziert. Das ist durch die Standardisierung des Codes und die strenge Typgebundenheit der Variablen vergleichsweise leicht möglich (siehe Grafik).

Weiter prüft der Browser nach dem Laden jeder Klasse, ob diese Namensräume anderer Klassen verletzt. Jedes Applet läuft also gewollt in einer beschränkten Umgebung.

Absolute Sicherheit im Internet gibt es jedoch nicht. Die Wahrscheinlichkeit aber, in nächster Zeit von einem Java-Virus behelligt zu werden, ist gering. Wie die Vergangenheit gezeigt hat, bessern Sun und die Browser-Hersteller etwaige Lücken im System sofort nach.

Das Werkzeug

Java-Programme werden mittels des Java Developer's Kit (JDK) von Sun entwickelt und übersetzt. Das JDK umfaßt: den Compiler (javac) zur Übersetzung, den Debugger (jdb) zur Fehlerlokalisierung, den Disassembler (javap) zur Rückübersetzung des Byte-Codes, einen Schnittstellengenerator (javah) zur Einbindung von C-Funktionen, ein Dokumentationsmodul (javadoc) für die Programmdokumentation .

Aktuell ist Version 1.0 des JDK für Win 95/NT, MacOS 7.5 und Solaris verfügbar. Außerdem gibt es Entwicklungstools von Symantec (Café), Microsoft (Jakarta) und für Borland (C++).

...und nicht stehlen?

Zur Beschaffung von Internet-fähigem Browser, von Java-Applets, von Java-Quelltexten und des Sun-JDK ist ein Anschluß ans Internet äußerst nützlich.

Erste Adresse dabei ist die Firma Sun, die ja die Erfinderin von Java ist. Bei ihr kann sich der geduldige Surfer das aktuelle JDK (rund 4 Megabyte) und die Dokumentation (800 Kilobyte) für den privaten Gebrauch kostenfrei abholen. Weiter findet er hier eine Entwicklerecke, die mit Dokumentationen für Programmierer und den Antworten auf häufig gestellte Fragen (FAQs), Beispielprogrammen und Neuigkeit rund um Java aufwartet.

Zweite Anlaufstelle ist der Gamelan-Server. Er ist die vielleicht längste Java-Links-Zusammenstellung der Welt.

Jan Kleinert

Anlaufstellen im Internet:

http://www.javasoft.com/
http://www.gamelan.com/
http://java.pages.de/
http://www.netscape.com/

Deutschsprachige Java-Slideshow über Java:
http://bilbo.rz.fh-wolfenbuettel.de/Seminare/Java/Sprache

Newsgroup zum Thema:
comp.lang.java

Das JDK ist auch in den CHIP-Foren von Compuserve und AOL verfügbar.