Java (J2SE, J2EE, J2ME)

Viktor Styrbul

In dieser Ausarbeitung geht es um Softwarearchitektur von Java. Ihre Merkmale und ihre Eigenschaften. Welche Entscheidungen wurden von den Entwicklern beim Design der Sprache getroffen und warum.

Inhaltsverzeichnis:

1 Historischer Hintergrund

Anfang der 1990er Jahre wollte Bill Joy eine neue Programmiersprache schaffen, die alle Vorteile von C und MESA vereinigen sollte. Seine Gedanken darüber, wie eine neue objektorientierte Sprache aussehen könnte, beschrieb er in dem Artikel „Further“. Diese neue Sprache sollte in ihren Grundzügen auf C++ aufbauen. Erst später wird ihm klar, dass C++ als Basissprache nicht geeignet und für große Programme unhandlich ist.

Zu dieser Zeit ist James Gosling mit der Entwicklung von SGML-Editor Imagination beschäftigt. Zu seinen Zwecken nutzte er C++ und war mit dieser Sprache nicht zufrieden. Aus diesem Unmut heraus entstand die neue Sprache Oak (Object Application Kernel). Der Legende nach hatte der Name Oak seinen Ursprung in einer Eiche (engl. oak), die James Gosling erblickte, als er aus seinem Arbeitszimmer schaute. Daraufhin startete Patrick Naughton im Dezember 1990 das Green-Projekt, in das Gosling und Mike Sheridan involviert waren. Überbleibsel aus dem Green-Projekt ist der Duke, der zum bekannten Maskottchen wurde.

Die Idee hinter diesem Projekt war, eine Software zu entwickeln, die für interaktives Fernsehen und andere Geräte der Konsumelektronik zuständig war. Das Betriebssystem Green-OS, Interpreter Oak und einige Hardwarekomponenten waren die Bestandteile dieses Projekts. Die Mitglieder des Green-Projekts orientierten sich nach dem Further-

Aufsatz von Joy. Als erstes schrieb Gosling den Original-Compiler in C, und anschließend wurde Runtime-Iterpreter von Naughton, Gosling und Sheridan ebenfalls in C entworfen. Die Sprache C++ kam bei dem Projekt nie zum Einsatz. Im August 1991 könnte Oak bereits die ersten Programme ausführen. So wurde von dem Green-Dream-Team ein Gerät mit dem Namen *7 (Star Seven) entwickelt, das im Herbst 1992 intern vorgestellt wurde. Damaliger Sun-Chef Scott McNealy war von dem Gerät sehr beeindruckt, und so wurde aus dem Team im November dieses Jahrs die Firma First Person Inc und beschäftigte sich mit der Vermarktung von Star Seven.

Am Anfang des Jahrs 1993 erfährt das Team von einer Anfrage von Time Warner wegen eines Systems für Set-Top-Boxen[1]. So richtete die Firma ihren Blick von Verbrauchermarkt auf die Set-Top-Boxen. Nach einiger Zeit war Time Warner nicht mehr an den Geräten interessiert, aber First Person entwickelte sich weiter. Zu dieser Zeit breitete sich das Internet langsam aus und so nach vielen Richtungswechseln konzentrierte sich die Entwicklung auf das World Wide Web. Dabei sollte die Programmiersprache Programmcode über das Netzwerk aufnehmen können, und schadhafte Programme sollten kein Schaden verursachen. Da sah man ein, dass die meisten Konzepte aus C und C++ dafür ungeeignet waren – Zugriffe über ungültige Zeiger, über die man den Speicher beliebig beschreiben kann, sind ein Beispiel.

Man erkannte, dass Oak alle Eigenschaften hatte, die nötig waren, um es im Web einzusetzen, obwohl die Sprache zuerst für einen ganz anderen Zweck entwickelt wurde. Da es bereits eine Software dieses Namens gab, war man aufgrund rechtlicher Probleme gezwungen den Namen zu verwerfen. Man entschied sich für den Namen Java nach einer Kaffee-Sorte, die von den Entwicklern bevorzugt getrunken wurde. In Java entwickelte Patrick Naughton den Prototyp des Browsers mit den Namen WebRunner. Nach einigen Überarbeitungen durch Jonathan Payne wurde der Browser HotJava genannt und im Mai auf der SunWorld ’95 der Öffentlichkeit vorgestellt.

Zunächst zeigten wenige Anwender ihre Begeisterung für HotJava. So hatten die Entwickler großes Glück, als Netscape sich entschied, Java-Technologie zu lizenzieren und wurde damit in die Version 2.0 des Nescape Navigators implementiert. Der Navigator wurde im Dezember 1995 auf den Markt gebracht. Schon im Januar nächsten Jahres wurde das JDK 1.0 freigegeben, was den Entwicklern ermöglicht hatte, Java-Applikationen und Web-Applets[2] zu programmieren. Kurz bevor das JDK 1.0 fertig gestellt wurde, gründeten die verbliebenen Mitarbeiter des Green-Teams die Firma JavaSoft. Und so begann Javas Siegeszug.

2 Grundkonzepte der Sprache

Als man Java entwickelte, strebte man hauptsächlich nach fünf Zielen:

- Sie sollte sicher und robust sein.

[1] Set-Top-Boxen sind elektronische Geräte für Endbenutzer
[2] Applet: »A Mini Application« (deutsch „Eine kleine Anwendung“)

- Sie sollte eine objektorientierte, verteilte, einfache und vertraute Architektur haben.
- Sie sollte interpretierbar, parallelisierbar und dynamisch sein.
- Sie sollte portabel und architekturneutral sein.
- Sie sollte sehr leistungsfähig sein

2.1 Portabilität
Zusätzlich dazu, dass die Java Architektur neutral ist, ist Java auch portabel. Das heißt, dass alle primitiven Datentypen sowohl in ihrer interner Darstellung und ihrer Größe als auch in ihrem arithmetischen Verhalten standardisiert sind. Zum Beispiel ist so ein primitiver Datentyp wie float immer ein IEEE 754 Float[3] von 32 Bit. Das gilt auch für die Klassenbibliotheken, die für GUI[4] zuständig sind. Sie kann dasselbe GUI unabhängig vom Betriebsystem erzeugen.

2.2 Architekturneutralität
Man hat Java so entwickelt, dass ein Programm, das man nur einmal schreiben muss, auf jeder beliebigen Computerhardware funktioniert, unabhängig von ihrem Prozessor oder anderen Hardwarebestandteilen.

2.3 Vertrautheit
Wegen seiner syntaktischen Ähnlichkeit mit C++, der ursprünglichen Ähnlichkeit der Klassenbibliothek zu Smalltalk-Klassenbibliotheken und der Einsatz von Design Patterns[5] in den Klassenbibliotheken hat man als erfahrener Programmierer sofort den Durchblick bei Java.

2.4 Sicherheit
Sicherheit wird bei Java durch folgenden Konzepte realisiert: der Class-Loader steuert die sichere Zuführung von Klasseninformation zu JVM[6], und Security-Manager stellt sicher, dass nur auf Programmobjekte zugegriffen, für die entsprechende Rechte vergeben wurden.

2.5 Objektorientierung
Java gehört zu den objektorientierten Sprachen.

2.6 Einfachheit
Da Java einen reduzierten Sprachumfang besitzt, ist sie im Vergleich zu anderen objektorientierten wie C++ oder C# einfach. Zum Beispiel unterstützt sie Operatorüberladung und Mehrfachvererbung nicht.

---

[3] Das bekannteste und häufigste Gleitkommasystem wurde 1985 von **I**nstitute of **E**lectrical and **E**lectronics **E**ngineers konzipiert

[4] Grafische Benutzeroberfläche

[5] Entwurfsmuster sind bewährte Lösungsschablonen für wiederkehrende Entwurfsprobleme in Softwarearchitektur und Softwareentwicklung.

[6] Java Virtual Machine ist Teil der Java-Laufzeitumgebun für Java-Programme, der für die Ausführung des Java-Bytecodes verantwortlich ist.

2.7 Interpretierbarkeit
Java gehört zu den interpretierbaren Sprachen. Java-Bytecode wird von der Java Virtual Machine der Firma Sun interpretiert, bevor sie ihn aus Leistungsgründen optimiert und übersetzt.

2.8 Verteilt
Java enthält eine Reihe einfacher Möglichkeiten für Netzwerkkommunikation, von TCP/IP[7]-Protokolen über Remote Method Invocation[8] bis zu Webservices.

2.9 Parallelisierbarkeit
Auf Grund dessen, dass Java Multithreading[9] unterstützt wird, ist es möglich den parallelen Ablauf von eigenständigen Programmabschnitten durchzuführen. Für die Zwecke bietet die Sprache selbst die Keywords synchronized und volatile Konstrukte, die das „Monitor & Condition Variable Paradigma“ von C.A.R Hoare[10] unterstüzen. Die Klassenbibliothek unterstützt parallele Programmierung mit Treads[11]. Bei modernen JVMs wird ein Java-Tread auf Betriebsystem-Thread abgebildet. Dadurch kann ein Programm von Prozessoren mit mehreren Rechenkernen profitieren.

2.10 Robustheit
Als Java entwickelt wurde, hatte man bei vielen der Designentscheidungen bei der Definition der Sprache drauf geachtet, dass die Wahrscheinlichkeit von ungewollten Systemfehlern so weit wie möglich ist zu reduzieren. Als Beispiel sind die starke Typisierung, Garbage Collection, Ausnahmebehandlungen als auch Verzicht auf Zeigerarithmetik zu nennen.

2.11 Leistungsfähigkeit
Dank der dynamischen Optimierungen der Virtual Maschine ist Java eine der effizientesten Programmiersprachen und erbringt ähnliche Geschwindigkeiten wie C++- oder C#-Programme.

2.12 Dynamisch
Man hat Java so aufgebaut, dass es sich an dynamisch ändernde Rahmenbedingungen anpassen lässt. Es können zum Beispiel Bibliotheken neu ausgeliefert werden, ohne die restlichen Programmteile anpassen zu müssen, weil die Module erst bei der Laufzeit gelinkt werden. Als Basis für Kommunikationen unter zwei Modulen können Interfaces eingesetzt werden, die eigentliche Implementierung kann aber dynamisch und auch während der Laufzeit geändert werden.

---

[7] Transmission Control Protocol / Internet Protocol ist eine Familie von Netzwerkprotokollen und hat eine große Bedeutung für das Internet.

[8] Remote Method Invocation (deutsch „Aufruf entfernter Methoden“). „Entfernt“ bedeutet dabei, dass sich das Objekt auf einem entfernten Rechner befindet. Dabei sieht der Aufruf fur das aufrufende Objekt für den Programmierer genauso aus, als ob es ein lokaler Aufruf wäre.

[9] Auch Nebenläufigkeit oder Mehrfädigkeit genannt. Bezeichnet das gleichzeitige Abarbeiten mehrerer

[10] Sir **C**harles **A**ntony **R**ichard **Hoare** besser bekannt als Tony Hoare oder C.A.R. Hoare, ist ein britischer Informatiker.

[11] Ausführungsstränge

3 Eigenschaften von Java

Java ist eine objektorientierte Sprache, die sich durch einige zentrale Eigenschaften auszeichnet. Diese Eigenschaften machen sie universell anwendbar und für die Industrie als robuste Programmiersprache interessant. Da Java objektorientiert ist, können Entwickler wieder verwertbare und moderne Softwarekomponenten programmieren.

3.1 Bytecode und die virtuelle Maschine

Zuerst ist Java eine Programmiersprache wie jede andere. Doch im Vergleich mit herkömmlichen Übersetzer einer Programmiersprache, die in der Regel Maschinencode für eine spezielle Plattform (z.B. Windows oder Linux) und einen bestimmten Prozessor (x86- oder ARM-Architektur) generieren, wird von dem Java-Compiler Bytecode erzeugt, für eine virtuelle Maschine. Bytecode kann man mit Mikroprozessorcode für einen erdachten Prozessor vergleichen. Der ist im Stande die Anweisungen wie arithmetische Operationen, Sprünge und weiteres auszuführen. Der Java-Compiler von Oracle und der Java-Compiler der Entwicklungsumgebung Eclipse sind selbst in Java implementiert und generieren diesen Bytecode (es existiert aber auch Java-Compiler die in C++ implementiert sind).
Um den Programmcode des virtuellen Prozessors auszuführen, wird der Bytecode nach der Übersetzungsphase von der Laufzeitumgebung (auch Runtime-Interpreter genannt) ausgeführt. Von der Hardwaremethode abgesehen ist Java somit eine compilierbare, aber auch interpretierbare Programmiersprache. Die virtuelle Maschine und Bibliotheken sind aber in C++ programmiert.

3.2 Just-In-Time Compilation und Optimierung

Da der Code interpretiert wird gibt es natürlich Geschwindigkeitsprobleme, weil das Erkennen, Dekodieren und Ausführen der Befehle Zeit kostet. Die ersten Java-Programme waren im Vergleich zu übersetzten C++-Programme deutlich langsamer. Das wurde durch die Technik der Just-in-Time-Compiler gelöst. Durch einen JIT-Compiler wird die Ausführung von Programmen beschleunigt, indem er zur Laufzeit die Programmanweisungen der virtuellen Maschine in Maschinencode der entsprechenden Plattform übersetzt. So entsteht ein an die Architektur angepasstes Programm im Speicher, das der physikalische Prozessor ohne Interpretation ziemlich schnell ausführen kann. Durch diese Technik kommt er der Geschwindigkeit der anderen übersetzten Sprachen entgegen.

4 Merkmale der Sprache

Java wurde als Sprache entwickelt, die es möglich machen sollte ein fehlerfreies Programm zu schreiben. In C-Programmen tritt statistisch gesehen alle 55 Programmzeilen ein Fehler auf. Auch in großen Softwarepaketen mit einigen Million Codezeilen tritt, unabhängig von der Programmiersprache, im Schnitt alle 200 Programmzeilen ein Fehler. Natürlich gilt es, diese Fehler zu beheben. Man hat aber bis heute noch keine umfassende Strategie, wie man dieses Problem in der Softwareentwicklung lösen kann. Viele Arbeiten der Informatik beschäftigen sich mit der Frage, wie Tausende Programmierer über Jahrzehnte miteinander arbeiten und Software entwerfen können. Dieses Problem ist nicht einfach zu lösen und wurde im Zuge der Softwarekrise in den 1960er Jahren heftig diskutiert[CU11].

Die Sprache Java ist nicht bis zur letzten Konsequenz objektorientiert. Es gibt eine Reihe an primitiven Datentypen wie Ganzzahlen oder Fließkommazahlen, die nicht als Objekte verwaltet werden. Vermutlich war der Grund für dieses Design, dass die Laufzeitumgebung und der Compiler mit der Trennung besser in der Lage waren, die Programme zu optimieren.

Von der Basisklasse „Object“ werden alle andere Klassen und Schnittstellen abgeleitet. Der Objektzugriff in Java wurde über Verweise auf Objekte implementiert, die den aus C++ bekannten Zeiger ähnlich sind. Die Verweise werden von der Sprachdefinition als „Reference Values“ bezeichnet, um zu verdeutlichen, dass sie als Call by Value[12] übergeben werden. Aus Sicherheitsgründen erlauben diese weder den Verweis auf Methoden oder andere Verweise, noch die tatsächliche Speicheradresse rauszufinden und zu modifizieren. Zeigerarithmetik[13] wurde aus denselben Gründen auch nicht implementiert. So wird bei Java bereits beim Design der Sprache ein häufiger Typ von Fehlern, die in anderen Programmiersprachen auftreten, von vornherein verhindert.

Die Unions, Strukturen und Aufzählungstypen (z.B. enum) können in Java, wie in jeder objektorientierten Programmiersprache, durch geeignete Klassen ersetzt werden. Deswegen werden explizite Typendefinitionen überflüssig. Durch diese Maßnahme ist die Syntax bei Java wesentlich schlanker als z.B. bei C++.

Alleinstehende Funktionen, die keine Methoden einer Klasse sind, werden bei Java nicht unterstützt. Dadurch werden die Widersprüche und Mehrdeutigkeit in der Klassenhierarchie vermieden.

Es wird nur Klassenvariablen und Klassenmethoden unterstützt, das allein stehende Funktionen überflüssig macht.

---

[12] Wertparameter sind Parameter von Unterprogrammen, die Kopien der beim Aufruf übergebenen Argumente speichern.

[13] Das Verringern oder Erhöhen eines Zeigers, um einen festen Wert oder das Subtrahieren zweier Zeiger wird als Zeigerarithmetik bezeichnet.

Zusammengehörige Klassen werden in Paketen (englisch packages) zusammengefasst. Durch die Pakete wird die Sichtbarkeit von Klassen eingeschränkt und eine Strukturierung von größeren Projekten sowie eine Trennung des Namensraums für verschieden Entwickler. Die Paketnamen sind hierarchisch aufgebaut und fangen meistens mit dem (umgekehrten) Internet-Domainnamen des Entwicklers. (Pakete, die von Sun entwickelt werden, fangen z.B. mit „com.sun“) Klassennamen müssen nur innerhalb eines Pakets eindeutig sein. Dadurch wird es ermöglicht, dass man Klassen von verschiedenen Entwicklern einfach kombinieren kann, ohne dass es zu möglichen Namenskonflikten kommt. Es gibt aber bei der Hierarchie der Paketnamen keine semantische Bedeutung. Bei der Sichtbarkeit zwischen den Klassen zweier Pakete spielt es keine Rolle, wo sich die Pakete in der Namenshierarchie befinden. Klassen sind entweder nur für Klassen des eigenen Paketes sichtbar oder für alle Pakete.

Bei Java wird es explizit zwischen Klassen und Schnittstellen unterschieden. So kann eine Klasse beliebig viele Schnittstellen implementieren, hat aber immer genau eine Basisklasse. Es wird kein direktes Erben von mehreren Klassen unterstützt, allerdings die Vererbung über mehrere Hiearchie-Ebenen[14].

In Java gibt es keine Header Files[15]. Alle Klassen werden komplett in einem File Codiert. Zu jeder Klasse wird ein eigenes Klassenname.class File erzeugt. Dadurch wird es verhindert, dass durch Änderungen in einer Zentralen Klasse in einem Header File das gesamte Projekt neu übersetzt werden muss.

4.1 Garbage-Collector

In Programmiersprachen wie C++ entsteht etwa die Hälfte der Fehler wegen falscher Speicher-Allokation. Mit Objekten zu arbeiten, heißt gewiss, sie anzulegen und zu löschen, wenn die nicht weiter gebraucht werden. In Java braucht man sich darum nicht zu kümmern, da die Laufzeitumgebung sich selbstständig um die Verwaltung dieser Objekte kümmert. Das hat die Wirkung, dass man nichts freigeben muss, ein so genannter Garbage-Collector macht das. Garbage-Collector ist ein Teil des Laufzeitsystems von Java. Das Generieren eines Objekts in einem Block mit anschließender Operation zieht eine Aufräumaktion des Garbage-Collectors nach sich. Wenn dann der Wirkungsbereich verlassen wird, wird das vom System als nicht mehr referenziertes Objekt erkannt. Ein weiterer Vorteil von Garbage-Collector ist , dass bei der Benutzung von Unterprogrammen oft Objekte zurückgegeben werden, und in herkömmlichen Programmiersprachen beginnt dann wieder die Diskussion, welcher Programmteil das Objekt jetzt löschen muss oder ob es nur ein Verweis ist. Bei Java spielt es keine Rolle, auch wenn ein Objekt nur der Rückgabewert einer Methode ist (anonymes Objekt). Garbage-Collector ist ein nebenläufiger Thread im Hintergrund, der nicht mehr referentierte Objekte markiert und von Zeit zu Zeit entfernt, da diese nicht gebraucht werden. So werden durch den Garbage-Collector die Funktionen free() aus C oder delete() aus C++ unnötig. Diese Technik macht es möglich, dass viele Probleme damit verschwinden. Nicht freigegebener Speicherplatz gibt es

[14] z.B. Klasse Kind erbt von Klasse Vater, die ihrerseits von Klasse Großvater erbt usw.
[15] eine Textdatei, die Deklarationen und andere Bestandteile des Quelltextes enthält.

in jedem größeren Programm, und falsche Destruktoren sind vielfach dafür verantwortlich [CU11].

4.2 Kein Präprozessor für Textersetzungen

Viele C(++)-Programme enthalten Präprozessor-Direktiven wie #include, #define oder auch #if zum Einbinden von Prototyp-Definitionen oder zur bedingten Übersetztung. Diesen Präprozessor hat man in Java aus unterschiedlichen Gründen nicht implementiert:

- Java besitzt die Datentypen einer festen Größe, die immer gleiche Länge haben, dadurch entfällt die Notwendigkeit, abhängig von der Plattform unterschiedliche Längen zu definieren.
- Da die virtuelle Maschine ohne äußere Steuerung Programmoptimierungen vornimmt, sind Pragma-Steuerungen im Programmcode unnötig.
- Da der Compiler die benötigten Informationen wie Methodensignaturen direkt aus den Klassendateien liest, sind Header-Dateien in Java nicht nötig.

Ohne den Präprozessor werden solche Tricks wie #define private public oder Makros, die Fehler durch eine doppelte Auswertung erzeugen, von vornherein ausgeschlossen. Man kann den Private/Public-Hack im Quellcode von StarOffice finden. Die obere Definition ersetzt jedes Auftreten von private durch public – mit der Konsequenz, dass der Zugriffsschutz verletzt ist. Die bedingte Übersetzung mit #ifdef ist auch ohne Präprozessor nicht mehr möglich [CU11].

5 Java Editions

Am Anfang war Java mit dem Schlagwort „WORA“ (Write Once RunAnywhere) – „Einmal schreiben – läuft überall“ angetreten. Irgendwann stellte die Firma Sun fest, dass die Geräte, auf denen die Programme laufen sollen, doch zu verschieden sind. So sind die drei Editionen von Java entstanden:

- Java Standart Edition (Java SE) ist für gängigen PC mit Betriebsystem Windows, Linux, OS X, oder Solaris.
- Java Enterprise Edition (Java EE) ist für unternehmensweite Programmen mit sehr vielen Anwendern geeignet. Hier gibt es viele Erweiterungen zur Serverprogrammierung.
- Java Micro Edition (Java ME) hat man speziell für kleine Geräte mit kleiner Rechenleistung und wenig Speicher entwickelt. Das sind natürlich Mobiltelefone, PDAs, Empfangsgeräte für Fernseher oder Steuerungsmodule in Autos.

5.1 Java Standard Edition

Die Java Standard Edition oder auch Java SE oder auch J2SE ist eine Sammlung von Java-APIs. Die Java SE dient als Grundlage sowohl für die Java Enterprise Edition, mit denen Java Programme für Unternehmen erstellt werden, als auch für die einfachere Java Micro Edition für mobile Geräte. Die aktuelle Version 6 erschien am 11. Dezember 2006 und ist in der Version Update 26 seit dem 7. Juni 2011 verfügbar.

Die Zahl „2“ in J2SE steht für Java 2. Aufgrund der Fortschritte in der Version 1.2 von Java hat sich Sun Microsystems 1998 entschlossen, Java-Versionen ab Java 1.2 als „Java 2“ zu bewerben, kehrte mit Version 6 dann aber wieder zur ursprünglichen Bezeichnung „Java“ bzw. „Java SE“ ohne nachstehende „2“ zurück. Auch bei den Versionsnummern gab es mehrere Veränderungen: So wurde aus Version 1.5.0 (veröffentlicht im September 2004) durch Weglassen der vorstehenden „1“ „J2SE 5.0“, mit Version 6 verzichtete man außerdem noch auf das „.0“ am Ende der Versionsnummer.

5.1.2 Programmierschnittstellen

Das Java-SE-6-API besteht aus:

- Höheren Base Libraries (I/O, JNI, Serialisierung)
- User Interface Toolkits (AWT, Swing, Java 2D)
- Integration Libraries (IDL, JDBC, JNDI, RMI, RMI-IIOP)
- lang & util Base Libraries (siehe unten)

Die wichtigsten Klassenbibliotheken befinden sich in den Paketen java.lang und java.util, Ein- und Ausgaben im Paket java.io.

5.1.3 Fundamentale Klassen (`java.lang`)

In diesem Paket und dessen Unterpakete sind Klassen definiert, die für die grundlegendsten Mechanismen der Sprache Java benötigt werden.

- Die absolute Basisklasse Object, von der alle anderen Schnittstellen und Klassen abgeleitet sind.
- Die Metaklasse Class, die die Grundlage für den Mechanismus der Selbstreflexion[16] bildet.
- Wrapper[17] um grundlegende Datentypen, wie z. B. Boolean, Character oder Integer.
- Die Klasse String für Zeichenketten.
- Die Klasse System dient zum Abfragen von Umgebungsvariablen und sogenannten System-Properties[18].

[16] ist ein Programm, das seine eigene Struktur kennt und diese, wenn es nötig ist, modifizieren kann.
[17] ein Programm, das als Schnittstelle zwischen zwei Programmcodes dient.

5.1.4 Klassen für Datenstrukturen, Zeitangaben und Internationalisierung (`java.util`)

Im Paket java.util wurden verschiedene Klassen definiert, die zwar weniger grundlegend als die in java.lang definierten sind, aber auch für praktisch jedes Java-Programm gebraucht werden.

- Kontainer-Klassen.
- Die Klasse Date, die einen Zeitpunkt represäntiert und die Klasse Calender, die Kalenderberechnungen durchführen kann.
- Die Klasse ResourceBundle, die eine Menge sogenannter „Ressourcen" darstellt. Ressourcen in diesem Sinne sind Bilder, Texte und andere Objekte, die für verschiedensprachige Programmversionen angepasst werden können.
- Die Klasse Properties, die unter anderem benannte String-Werte aus einer Textdatei lesen kann. Sie liegen auch einer Implementierung der Ressourcenbündel zu Grunde.
- Das Paket java.util.jar unterstützt die Erzeugung und Verarbeitung von Java Archives (JAR- Dateien).
- Das Paket java.util.regex unterstützt Reguläre Ausdrücke zu Auffinden von Mustern in Strings

5.1.5 Klassen für Ein- und Ausgaben (`java.io`)

Das Paket java.io enthält Klassen für Ein und Ausgaben (englisch input und output , daher „io"), hauptsächlich Streams[19].

- Die abstrukte Klasse InputStream ist die Basisklasse aller Eingabeströme, um Eingaben von der Tastatur oder einer Datei zu lesen.
- Die abstrakte Klasse OutputStream ist die Basisklasse aller Ausgabeströme, um Zeichen auf den Bildschirm oder in eine Datei zu schreiben.
- Die Streams existierten schon in der ersten Version von Java und verarbeiten Bytes. Später wurden die Klassen Reader und Writer speziell für die Ein- und Ausgabe von Unicode-Zeichen eingeführt.

5.2 Java Platform, Micro Edition

Java Platform, Micro Edition, abgekürzt Java ME, früher auch als Java Platform 2, Micro Edition oder kurz J2ME bezeichnet, ist eine Umsetzung der Programmiersprache Java für Mobiltelefone oder PDAs etc. Die Java 2 Micro Edition ist gegenüber der Java 2 Standard Edition abgespeckt worden, um den Anforderungen der mobilen Endgeräte besser

---

[18] Systemeigenschaften, wie zum Beispiel die Version der Java-Laufzeitumgebung.
[19] Datenströme

gerecht zu werden. Im Gegensatz zu den Standart- und Enterpriseausgaben ist J2ME kein einheitliches Paket, sondern ist in mehrere Configurations und Profiles genannte Gruppen unterteilt, welche die unterschiedlichen Fähigkeiten der verschiedenen Engeräte voneinander abgrenzen.

Allerdings plant Sun die Java Micro Edition für mobile Geräte schrittweise durch die Standard Edition zu ersetzen und letztlich die Micro Edition einzustellen. Möglich wird dies, da moderne Smartphones anwachsend hohe Rechenleistung vorzuweisen haben. Auch soll so der Fragmentierung von Java entgegengewirkt werden.

5.3 Java Platform, Enterprise Edition

Java Platform, Enterprise Edition, abgekürzt Java EE oder früher J2EE, ist die Spezifikation einer Softwarearchitektur für die transaktionsbasierte Ausführung von in Java-Programmen und insbesondere Web-Anwendungen. Sie ist eine der großen Plattformen, die um den Middleware-Markt kämpfen. Größter Konkurrent ist dabei die .NET-Plattform von Microsoft.

5.3.1 Infrastruktur

Java-EE-Komponenten erfordern als Laufzeitumgebung einen Java EE Application Server. Dieser Server stellt verschiedene technische Funktionalitäten zur Verfügung:

- Sicherheit
- Transaktionsmanagement
- Namens- und Verzeichnidienste
- Kommunikation zwischen Java-EE-Komponenten
- Persistenzdienste zum lngfristigen Speichern von Java-Objekten
- Management der Komponenten über den gesamten Lebenszyklus
- Unterstützung für die Installation

Als Weiteres wird vom Server der Zugriff auf die Ressourcen des zugrunde liegenden Betriebsystems (Dateisystem, Netzwerk, …) gekapselt.

Ein Java-EE-Server wird in verschiedene logische Systeme unterteilt, die Container genannt werden. Die aktuelle Spezifikation erfordert die folgenden Container:

- einen EJB-Container als Laufzeitumgebung für Enterprise Java Beans
- einen Web-Container als Laufzeitumgebung für Servlets und JavaServer Pages
- einen JCA-Container als Laufzeitumbebung für JCA Connectoren.
- einen Java Message Service Provider als Verwaltungssystem für Nachrichtenwarteschlangen.

Als weitere Infrastrukturkomponente kommt für die persistente Speicherung von Daten ein Datenbankmanagementsystem zum Einsatz. Hierbei kann es sich um ein relationales System handeln, oder aber auch um ein vergleichbares System wie beispielsweise ein OODBMS[20]. Die Anbindung der Datenbankmanagementsysteme erfolgt meist über einen Java Database Connectivity Treiber. Der clientseitige Zugriff auf eine Java-EE-Anwendung erfolgt oft über einen Browser.

6 Wofür sich Java weniger eignet

Java wurde für allgemeine Probleme entworfen und deckt große Anwendungsgebiete ab.

Es gibt aber ausreichend viele Anwendundsfälle bei denen es deutlich bessere Programmiersprachen gibt. Zum Beispiel im Bereich Skripting, wo die Eigenschaft, dass jedes Java-Programm mindestens eine Klasse und eine Methode benötigt , eher störend ist, oder im Bereich von automatisierter Textverarbeitung, wo andere Programmiersprachen eleganter mit regulären Ausdrücken arbeiten können.

Auch wird es in Java umständlich, wenn extrem maschinen- und plattformabhängige Anforderungen bestehen. Java wurde plattformunabhängig entworfen, sodass alle Methoden auf allen Systemen lauffähig sein sollen. Sehr systemnahe Eigenschaften wie die Taktfrequenz sind nicht sichtbar, und sicherheitsproblematische Manipulationen wie der Zugriff auf bestimmte Speicherzellen (das PEEK und POKE)[21] sind ebenso untersagt.

Hier ist eine unvollständige Aufzählung von Dingen, die Java standardmäßig nicht kann:

- Cursor auf der Textkonsole positionieren, Bildschirm löschen und Farben setzen
- Zugriff auf USB oder Firewire
- CD auswerfen
- Microsoft Office fernsteuern
- auf niedrige Netzwerk-Protokolle wie ICMP[22] zugreifen

Aus oben genannten Nachteilen, dass Java nicht auf die Hardware zugreifen kann, folgt, dass Java nicht so ohne weiteres für die Systemprogrammierung eingesetzt werden kann. Treibersoftware, die Grafik-, Sound- oder Netzwerkkarten anspricht, lässt sich in Java nur über Umwege realisieren. Genau das Gleiche gilt für den Zugriff auf die allgemeinen Funktionen des Betriebssystems, die Windows, Linux oder ein anderes System bereitstellt.

---

[20] Objektorientierte Datenbankmanagementsystem.

[21] POKE ist ein Software-Kommando aus der Programmiersprache BASIC. Wird benutzt um direkt in Speicherzellen oder Hardware-Register des Rechners zu schreiben. Mit dem PEEK-Befehl wird aus den Speicherzellen direkt gelesen.

[22] Internet Control Message Protocol ist ein Internetprotokoll zum Austausch von Informations- und Fehlermeldungen

Literaturverzeichnis

[CU11] Christian Ullenboom: Java ist auch eine Insel

Internetquellen

[NK11] Nepomuk Karbacher: “Kommunikation und Austausch von Multimediadaten in spontanen Bluetooth Netzwerken mittels J2ME” http://www.karbacher.org/java-j2me/java-und-j2me-die-unterschiede/
[JS07] Julius Stiebert http://www.golem.de/0710/55533.html
[IQ01] http://www.referate10.com/referate/Informatik/7/Geschichte-und-Einfuhrung-in-Java-reon.php
[WP01] http://de.wikipedia.org/wiki/Java-Plattform
[WP02] http://de.wikipedia.org/wiki/Java_(Programmiersprache)
[WP03] http://de.wikipedia.org/wiki/Java_Standard_Edition