

Vorwort

»Writing a book is an adventure. To begin with, it is a toy and an amusement; then it becomes a mistress, and then it becomes a master, and then a tyrant. «

–Winston Churchill über seine Memoiren.

Die Welt der Softwareentwicklung ändert sich. Einerseits weitet sich die Durchdringung von **rechnergestützten Systemen** in zunehmendem Maße aus – selbst in Chipkarten, Waschmaschinen und Toaster ist inzwischen eine CPU. Andererseits nimmt die **Vernetzung von Rechnern**, nicht nur, aber in besonderem Maße durch das Internet zu. Dadurch erhält die Entwicklung von **Anwendungen in verteilten Systemen** eine neue Bedeutung.

Verteilte Systeme sind in der Industrie und in der Wirtschaft schon lange im Einsatz. Doch während diese Systeme früher für hochspezialisierte Anwendungen entwickelt wurden und ein sehr hoher Aufwand für die Erstellung gerechtfertigt werden mußte, entwickeln sich verteilte Systeme heutzutage langsam zu einer allgegenwärtigen Infrastruktur, die in sehr vielen Bereichen Einzug hält. Während früher nur große Unternehmen, wie Banken und Luftfahrtunternehmen, in der Lage waren, diesen hohen Aufwand zu betreiben, stehen heute der breiten Masse der Firmen und sogar Privatleuten die Möglichkeiten zur Verfügung, verteilte Anwendungen zu entwickeln.

Mit der zum Teil seit Jahrzehnten bewährten Internettechnologie, die durch das WWW populär, weitverbreitet und einfach zu benutzen geworden ist und durch Intra- und Extranets und Electronic Commerce auch eine große wirtschaftliche Bedeutung gewonnen hat, steht eine Technologie und eine Infrastruktur zur Verfügung, die die Entwicklung verteilter Anwendungen ermöglicht.

Der Programmiersprache **Java** kommt in diesem Zusammenhang eine große Bedeutung zu. Einerseits bietet Java die Möglichkeit, Anwendungen durch seine Portabilität und die »Download«-barkeit zu verteilen, d.h., Java-Code kann als Applet von einem Server über das Internet in einen Browser geladen und dort ausgeführt werden. Andererseits bietet sie die Möglichkeit, verteilte Anwendungen zu ent-

wickeln, bei denen die Applikation nicht nur auf einem Rechner, sondern verteilt über mehrere abläuft.

Durch die Kombination dieser beiden Aspekte stehen in Java Möglichkeiten zur Verfügung, die es bisher in keiner anderen weitverbreiteten Programmiersprache gegeben hat. Der erste Aspekt, das Herunterladen von Java-Code als Applet, wurde bereits in vielen Büchern beschrieben. Doch die **Techniken zur Erstellung verteilter Anwendungen mit Java** wurden bisher nur in einzelnen Aspekten und fast ausschließlich in der englischen Literatur behandelt. Dieses Buch stellt in einem Überblick die vorhandenen Techniken zur Entwicklung verteilter Anwendungen mit Java vor.

Aufbau des Buches

Nach einer **Einleitung**, in der die Bedeutung von Java und von verteilten Systemen besprochen und deren Zusammenhänge erläutert werden, gliedert sich dieses Buch in zwei Teile.

- Teil I, **Java in verteilten Systemen**, gibt eine breite Übersicht über die heutzutage verfügbaren Techniken, die in verteilten Systemen Verwendung finden. Dabei wird ein Schwerpunkt auf die drei Aspekte Nebenläufigkeit, Verteilung und Persistenz gelegt, die in fast jedem verteilten System zum Tragen kommen. Die Problematik der Nebenläufigkeit wird mit Java Threads behandelt. Die vorgestellten Techniken zur verteilten Kommunikation reichen von Sockets über RMI und CORBA bis hin zu mobilen Agenten. Die persistente Speicherung wird mit relationalen ebenso wie mit objektorientierten Datenbanken und mit einer Java-Sprachumgebung, die Persistenz unmittelbar in die Sprache integriert, diskutiert. Alle drei Aspekte werden von sogenannten Tupelräumen, einem verteilten Speichermechanismus für nebenläufige Systeme, abgedeckt. Und schließlich wird Jini, eine Infrastruktur, die das einfache Finden von Diensten in verteilten Systemen erlaubt, vorgestellt.
- In Teil II, **Ein verteiltes Java**, wird untersucht, ob sich die Programmierung verteilter Systeme durch eine geeignete Weiterentwicklung der Sprache Java vereinfachen lässt. Es werden zunächst andere verteilte Programmiersprachen untersucht und daraus Grenzen und Möglichkeiten eines verteilten Javas hergeleitet. Danach wird ein Konzept diskutiert, das die Programmierung verteilter Systeme erheblich vereinfacht. Anschließend wird die Programmiersprache »Dejay« – ein verteiltes Java – vorgestellt, die auf diesem Konzept beruht, und mit Beispielen demonstriert.

Die einzelnen Techniken werden mit praktisch relevanten aber einfachen Beispielen vorgestellt. Um eine möglichst gute Vergleichbarkeit verwandter Techniken zu erreichen, wird mehrfach dieselbe Problemstellung in verschiedenen Kapiteln mit unterschiedlichen Techniken gelöst. Die dabei unveränderten Programmteile werden jeweils im Anhang aufgeführt.

Zielgruppe

Dieses Buch ist aus mehreren Lehrveranstaltungen zum Thema Verteilung, Internet und Java an der Universität Hamburg sowie der Forschungsarbeit des Autors auf dem Gebiet Programmiersprachen für verteilte Systeme entstanden. Es richtet sich daher an verschiedene Leserkreise:

- ❑ Studenten mit Vorkenntnissen in Java, die sich für die Programmierung verteilter Systeme interessieren. Das Buch kann sehr gut als begleitendes Material für eine Vorlesung, ein Praktikum oder ein Seminar genutzt werden.
- ❑ Programmierer, die sich die verschiedenen Programmier Techniken für verteilte Systeme anschauen möchten. Das Buch präsentiert den State-of-the-Art in der Programmierung verteilter Systeme und bietet eine gute Vergleichsmöglichkeit.
- ❑ Entscheider mit programmiertechnischem Hintergrund, die ein Gefühl für das Spektrum der verschiedenen zur Verfügung stehenden Alternativen beim Entwurf eines verteilten Systems bekommen möchten.
- ❑ Forscher, die sich mit dem Stand der Technik vertraut machen und neue innovative Techniken kennenlernen wollen.

Zu allen vorgestellten Techniken werden die wichtigsten Konzepte aufgeführt und an einfachen Beispielen verdeutlicht. Es wird aber in keinem Fall eine Programmierschnittstelle in ihrer vollen Tiefe und Breite abgedeckt. Dieses Buch ersetzt kein Referenzhandbuch. Vielmehr bietet es, was Referenzhandbücher nicht bieten können: einen breiten Überblick über ein Themengebiet in angemessener Tiefe.

Konventionen

Dies ist ein Buch über Java, daher taucht natürlich auch sehr viel Java-Code auf. Die meisten Beispiele sind möglichst kurz gehalten, um das Wesentliche zu verdeutlichen und nicht in einem Wust von

Code untergehen zu lassen. Daher haben die meisten Beispiele keine grafischen Oberflächen, sondern arbeiten mit der Textkonsole als Ein- und Ausgabemedium.

Java-Code und -Bezeichner, wie etwa Klassen- oder Methodennamen innerhalb des Textes, wenn zum Beispiel von einer `TestClass` die Rede ist, oder Quellcodeauszüge und Bildschirmein- und -ausgaben, werden mit dem Font `Sanserif` dargestellt. Methoden werden im Fließtext mit leeren Klammern (`someMethod()`) aufgeführt, unabhängig davon, ob sie Parameter erfordern. Wenn sich der Text auf gewisse Bereiche innerhalb eines Quellcodeauszuges bezieht oder ein bereits zitierter Quellcodeauszug um neue Zeilen erweitert wird, so wird dieser zusätzlich **fett** gedruckt.

Darstellung von Code

```
class TestClass {
    public static void main (String[] args) {
        System.out.println("Hello World");
    }
}
```

Codesegmente, deren Breite über den normalen Textfluß hinausreicht oder die über eine Seite hinausgehen, werden in zwei horizontale Striche gefaßt.

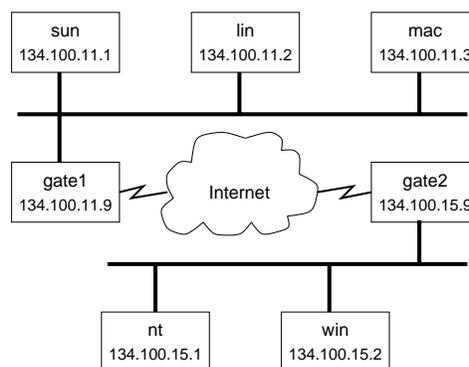
Darstellung von Ein- und Ausgaben

Eingaben, wie zum Beispiel Aufrufe des Java-Compilers `javac`, werden durch ein vorangestelltes Größerzeichen gekennzeichnet.

```
> javac TestClass.java
```

Ausgaben des Systems auf den Bildschirm, die häufig aus Eingaben resultieren, werden durch einen vorangestellten senkrechten Strich gekennzeichnet.

Abbildung 1
Die in Beispielen
verwendeten Rechner und
die Netzstruktur



```
> java TestClass  
| Hello World
```

Da es aber in diesem Buch um verteilte Systeme geht, werden für viele Beispiele mehrere Rechner zur Ausführung einer Anwendung benötigt. Es wird daher hier ein beispielhaftes heterogenes Netz, bestehend aus zwei Teilnetzen verbunden über das Internet, mit verschiedenen Rechnern aufgezeichnet. In diesem Beispielnetz, dargestellt in Abbildung 1, haben die Rechner eine (fiktive) IP-Adresse und einen Namen.

Diese Namen stehen symbolisch für verschiedene Hardwareplattformen und die dazugehörigen Betriebssysteme, die tatsächlich alle als Testrechner für die Programmbeispiele zur Verfügung standen. Die verwendeten Namen sind `sun`, `lin`, `mac`, `nt` und `win` sowie `gatel` und `gate2` als Gateways in das Internet. Wo dies zur Unterscheidung notwendig ist, werden diese Namen oder diese IP-Adressen im Text, in den Programmbeispielen und vor der Eingabe bzw. Ausgabe verwendet. Das folgende Beispiel, in dem eine Nachricht vom Rechner `sun` an den Rechner `lin` geschickt und ausgegeben wird, soll dies verdeutlichen:

```
sun> java Talker  
sun| sending Message "Hello World" to 134.100.11.2
```

```
lin> java Listener  
lin| received Message "Hello World" from 134.100.11.1
```

Homepage und Sourcecode

Wie es sich für ein Buch, in dem das Internet eine große Rolle spielt, gehört, gibt es eine Homepage zu diesem Buch, die unter der URL <http://www.dpunkt.de/produkte/dejay> erreichbar ist. Dort sind Auszüge dieses Buches, die vermutlich unvermeidlichen Errata sowie der Sourcecode zu den Beispielen zugreifbar.

Zu Dejay (ein verteiltes Java, das im zweiten Teil des Buches vorgestellt wird) gibt es eine eigene Domäne. Unter <http://www.dejay.org> sind Informationen, weitere Dokumentation und der Sourcecode frei erhältlich.

Danksagung

Dieses Buch ist vollständig mit frei verfügbarer Software entstanden. Als Betriebssystem wurde Linux mit der neu entwickelten grafischen Oberfläche KDE eingesetzt. Als Textverarbeitungsprogramm

war `kLyX`, ein Programm, das die vollen Möglichkeiten von `LATEX` ausschöpft, aber seine Komplexität hinter einer sehr gut bedienbaren und hilfreichen Oberfläche verbirgt, von großem Wert. Die Grafiken wurden mit `xfig` erstellt. Die Programme entstanden im wesentlichen mit dem JDK von Sun in der Version 1.1.7 und sind durchgehend kompatibel zum JDK 1.2 (bzw. der Java-Plattform 2). Für einige Kapitel (Jini und Teile von Tupelräume) ist das JDK 1.2 Voraussetzung. All den Menschen, die an der Entwicklung dieser Software beteiligt waren und ihr Wissen und ihre Arbeitskraft in so selbstloser Weise der Allgemeinheit zur Verfügung stellen, bin ich zutiefst dankbar und möchte ihnen hier eine Würdigung ihrer Arbeit aussprechen.

Dieses Buch ist zu großen Teilen aus meiner Lehrtätigkeit auf dem Gebiet Verteilte Systeme an der Universität Hamburg entstanden. Besonders erwähnen möchte ich das Praktikum »Internet«, das Seminar »Objekte in verteilten Systemen« und das Projektseminar »Realisierung offener verteilter Softwareanwendungen«. Die vielen Studenten, die daran teilgenommen haben, mich das Lehren gelehrt, meine Vorträge konstruktiv und kritisch kommentiert, mich durch ihre eigenen Beiträge weitergebracht, mich zu neuen Ideen inspiriert und schlechte Ideen identifiziert haben, meine Manuskripte gelesen, Fehler gesucht und gefunden, Schwachstellen angemahnt und Vorschläge zur Verbesserung gemacht haben, all denen bin ich mit großem Dank verbunden. Einen ganz besonderen Dank möchte ich denen aussprechen, die mich durch Studien- und Diplomarbeiten ganz unmittelbar in meiner Arbeit unterstützt haben. Bei der Entwicklung von Dejay waren Jan Raap, Thorsten Sturm, Tobias Baier, Nils Poppendiek und Per Fragemann maßgeblich beteiligt. Besondere Erwähnung verdient Marco Kaiser, der mich in vielerlei Hinsicht tatkräftig und zuverlässig unterstützt hat. Für die kritische Durchsicht des Manuskripts möchte ich mich bei Prof. Dr. Claudia Linnhoff-Popien, Dr. Christian Zeidler, Stefan Middendorf und Uta Arnold bedanken.

Bei meinen Kollegen der Arbeitsgruppe Verteilte Systeme möchte ich mich für eine sehr gute Atmosphäre, für viele Gespräche und Diskussionen, für Beistand und Hilfe bedanken. Hochachtung und Dank möchte ich Prof. Dr. Winfried Lamersdorf aussprechen. Er hat mit hohem Engagement und Weitsicht meine Arbeit ermöglicht, begleitet und gefördert.

Schließlich möchte ich dem Team des `dpunkt.verlages` und insbesondere Christa Preisendanz für die gute Zusammenarbeit bei der Erstellung dieses Buches danken.

Hamburg, im August 1999, Marko Boger

Inhaltsverzeichnis

1	Einleitung	1
1.1	Die Bedeutung von Java	1
1.2	Die Bedeutung verteilter Systeme	7
1.3	Nebenläufigkeit, Verteilung und Persistenz	10
1.4	Literaturhinweise	15
I	Java in verteilten Systemen	17
2	Nebenläufigkeit in Java	19
2.1	Threads in Java	22
2.2	Synchronisationsmechanismen	30
2.2.1	Monitor	30
2.2.2	Semaphore	31
2.2.3	ReadWriteLock	33
2.3	Nebenläufigkeit und Verteilung	36
2.3.1	Server und Handler	36
2.3.2	Asynchrone Aufrufe	38
2.4	Literaturhinweise	41
3	Java Sockets	43
3.1	TCP/IP	43
3.1.1	Die Protokollschichten	44
3.1.2	Ports	46
3.2	Sockets	47
3.3	Streams	48
3.4	Filter	51
3.5	Ein Chat-System mit Sockets	53
3.6	Multicast	58
3.6.1	Ein Chat-System mit IP-Multicast	61
3.7	iBus: Ein zuverlässiger Multicast	62
3.7.1	Der Protokollstack	65
3.7.2	Ein Chat-System mit iBus	69
3.8	Literaturhinweise	71

4	RMI	73
4.1	Die Architektur von RMI	74
4.2	Ein einfaches Beispiel	76
	4.2.1 Der Server	77
	4.2.2 Der Client	79
4.3	Ein Chat-System mit RMI	80
4.4	Literaturhinweise	86
5	CORBA	87
5.1	Die Struktur der OMG	88
5.2	Der Standardisierungsprozeß	89
5.3	OMA – die Gesamtarchitektur	90
5.4	Die CORBA-Architektur	91
5.5	IDL	94
5.6	Der IDL-Compiler	98
5.7	Das IDL-Java-Mapping	102
5.8	Ein einfaches Beispiel	107
5.9	Ein Chat-System mit CORBA	109
5.10	Starten eines Servers	112
5.11	Starten eines Clients	113
5.12	Literaturhinweise	114
6	Voyager	117
6.1	Die Voyager-Laufzeitumgebung	120
6.2	Entfernte Objekte	121
6.3	Migration von Objekten	127
6.4	Entfernte Aufrufe	134
6.5	Gruppenkommunikation	137
	6.5.1 Ein Chat-System mit Voyager Space	139
6.6	Literaturhinweise	141
7	Mobile Agenten	143
7.1	Aglets	148
	7.1.1 Ein Chat-System mit Aglets	152
7.2	Mobile Agenten mit Voyager	156
	7.2.1 Ein Chat-System mit Voyager Agenten	157
7.3	Literaturhinweise	160
8	JDBC	163
8.1	Aufbau und Struktur von JDBC	164
8.2	Verbindungsaufbau zu einer Datenbank	167
8.3	Anfragen und Antworten	168
8.4	Ein Beispiel: BulletinBoard	170
8.5	Literaturhinweise	174

9	Objektorientierte Datenbanken	175
9.1	ObjectStore	176
9.2	Zugriff auf persistente Objekte	177
9.3	Der Postprozessor	182
9.4	Das BulletinBoard mit ObjectStore	183
9.5	Reaktivierung von Objekten	190
9.6	Literaturhinweise	195
10	Eine persistente Programmiersprache	197
10.1	PJama	197
10.2	Der persistente Store	199
10.3	Stabilisierung	202
10.4	Garbage Collection	203
10.5	Ein einfaches Beispiel: Ein persistenter Zähler	204
10.6	Literaturhinweise	206
11	Tupelräume in Java	207
11.1	JavaSpaces	209
11.2	TSpaces	212
11.3	Literaturhinweise	216
12	Jini	217
12.1	Voraussetzungen	218
12.2	Dienste	219
12.3	Discovery and Join	220
12.4	Lookup	224
12.5	Leasing	228
12.6	Jini starten	229
12.7	Literaturhinweise	230
II	Ein verteiltes Java	231
13	Verteilte Programmiersprachen und Nebenläufigkeit	233
13.1	Eine verteilte Programmiersprache: Emerald	234
13.2	Die Vision der <i>Unified Objects</i>	237
	13.2.1 Unterschiede zwischen lokaler und verteilter Programmierung	238
	13.2.2 Latenzzeit	239
	13.2.3 Speicherzugriff	240
	13.2.4 Teilausfälle	241
	13.2.5 Nebenläufigkeit	243
	13.2.6 Wahrung des Unterschieds	245
13.3	Ist Java eine verteilte Programmiersprache?	245
13.4	Ein Konzept für Nebenläufigkeit: Eiffel SCOOP	249

13.4.1	Nebenläufigkeit durch Prozessoren	250
13.4.2	Nebenläufigkeit und Synchronisation	252
13.4.3	Verteilung	253
13.5	Anforderungen an ein verteiltes Java	254
13.6	Literaturhinweise	256
14	Virtuelle Prozessoren	257
14.1	Das Konzept des virtuellen Prozessors	260
14.2	Migration	263
14.3	Verteilung und Nebenläufigkeit	265
14.4	Persistenz	267
14.5	Literaturhinweise	269
15	Dejay: Ein verteiltes Java	271
15.1	Ein einfaches Beispiel	271
15.2	Virtuelle Prozessoren	275
15.3	Entfernte Objekte	277
15.4	Migration	281
15.5	Namensdienst	283
15.6	Persistenz	284
15.7	Ausnahmebehandlung	285
15.8	Der Compiler dejayc	286
15.9	Programmstart	287
15.10	Einschränkungen	288
15.11	Literaturhinweise	291
16	Beispiele	293
16.1	Verteilung	293
16.2	Nebenläufigkeit	303
16.2.1	Implementationsvergleiche	309
16.3	Persistenz	314
16.4	Literaturhinweise	318
A	Die Chat-Oberfläche	319
B	Die BulletinBoard-Oberfläche	323
	Abbildungsverzeichnis	329
	Literaturverzeichnis	333
	Index	347