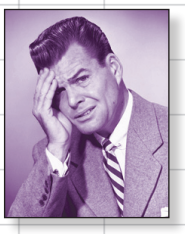
Ein Buch zum Mitmachen und Verstehen

Behandelt Java 5.0

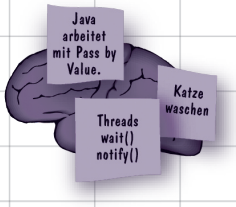
Java™ von Kopf bis Fuß

Erfahren Sie, wie Threads Ihr Leben verändern können



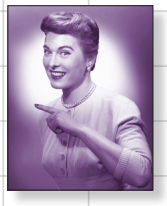
Vermeiden Sie peinliche OO-Fehler

Sorgen Sie dafür, dass die Java-Konzepte auch wirklich in Ihrem Hirn haften bleiben



Stöbern Sie in der Java-Bibliothek herum

Trainieren Sie Ihren Kopf mit 42 Java-Rätseln



Machen Sie Ihre GUIs hübsch und nützlich

O'REILLY®

Kathy Sierra & Bert Bates
Deutsche Übersetzung von Elke Buchholz und Lars Schulten

Kommentare und Fragen können Sie gerne an uns richten:
O'Reilly Verlag
Balthasarstr. 81
50670 Köln
Tel.: 0221/9731600
Fax: 0221/9731608
E-Mail: kommentar@oreilly.de

Die Originalausgabe erschien 2005 unter dem Titel
Head First Java, 2nd Edition bei O'Reilly Media, Inc.

Bibliografische Information Der Deutschen Bibliothek
Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.ddb.de abrufbar.

Übersetzung und deutsche Bearbeitung: Lars Schulten, Köln & Elke Buchholz, Aachen
Lektorat: Christine Haite, Köln
Fachgutachten: Karsten Loesing, Bamberg, Judith Baur, München & Michael Just, Freiburg
Korrektorat: Sibylle Feldmann, Düsseldorf
Satz: Conrad Neumann, München
Umschlaggestaltung: Edie Freedman, Boston
Produktion: Andrea Miß, Köln
Belichtung, Druck und buchbinderische Verarbeitung: Media-Print, Paderborn

ISBN-10 3-89721-448-2
ISBN-13 978-3-89721-448-4

Dieses Buch ist auf 100% chlorfrei gebleichtem Papier gedruckt.

Für unsere Gehirne – dafür, dass sie immer da sind.

(trotz schwacher Beweise)

Die Köpfe hinter dieser Reihe

Kathy interessiert sich seit ihrer Zeit als Spiele-Designerin (sie hat Spiele für Virgin, MGM und Amblin' geschrieben) für Lerntheorie. Sie hat einen großen Teil des Formats der amerikanischen Originalreihe »Head First» entwickelt, aus der dieses Buch stammt, während sie im Rahmen des berufsbegleitenden Studienprogramms der UCLA im Studiengang Unterhaltung das Fach »New Media Authoring» unterrichtet hat. Später war sie Master Trainer bei Sun Microsystems und hat Java-Dozenten darin ausgebildet, wie man die neuesten Java-Techniken vermittelt, außerdem war sie leitende Entwicklerin der Java-Programmer- und Java-Developer-Zertifizierungsprüfungen von Sun. Gemeinsam mit Bert Bates hat sie die Konzepte von »Java von Kopf bis Fuß» aktiv im Unterricht für Hunderte von Trainern, Entwicklern und sogar Nicht-Programmierern eingesetzt. Zudem hat sie eine der größten Java-Community-Websites der Welt, *javaranch.com*, und das Blog »Creating Passionate Users» begründet.

Kathy war neben diesem Buch auch Koautorin von Head First Servlets & JSP, Head First EJB und Head First Design Patterns, das unter dem Titel »Entwurfsmuster von Kopf bis Fuß» auch auf Deutsch erschienen ist.

In ihrer Freizeit beschäftigt sie sich gern mit ihrem neuen Islandpferd, Skifahren, Laufen und der Lichtgeschwindigkeit.

kathy@wickedlysmart.com

Bert arbeitet als Software-Entwickler und -Architekt, aber die jahrzehntelange tägliche Arbeit mit künstlicher Intelligenz hat sein Interesse an Lerntheorie und computergestütztem Lernen gefördert. Seitdem bringt er Kunden das Programmieren bei. In letzter Zeit hat er im Entwicklerteam für einige von Suns Java-Zertifizierungsprüfungen mitgearbeitet.

Im ersten Jahrzehnt seiner Software-Karriere hat er die Welt bereist und für Rundfunk-Kunden wie Radio New Zealand, den Weather Channel und das Arts & Entertainment Network (A & E) gearbeitet. Eines seiner absoluten Lieblingsprojekte war der Aufbau einer kompletten Eisenbahnsystem-Simulation für Union Pacific Railroad.

Bert ist hoffnungslos dem *Go*-Spiel verfallen und arbeitet – schon etwas zu lange – an einem Go-Programm. Er spielt ganz gut Gitarre und versucht sich jetzt am Banjo. Zu seinen Lieblingsfreizeitvergnügen gehören Skifahren, Laufen und die Versuche, seinem Islandpferd Andi etwas beizubringen (oder von ihm zu lernen).

Bert ist Koautor der gleichen Bücher wie Kathy und bereits schwer mit dem nächsten Stoß Bücher beschäftigt.

Manchmal können Sie ihn auf dem IGS-Go-Server erwischen (unter dem Login-Namen jackStraw).

terrapin@wickedlysmart.com

Kathy und Bert versuchen, so viele E-Mails zu beantworten wie möglich, aber auf Grund des hohen Mail-Aufkommens und ihrer vielen Reiseverpflichtungen ist das schwierig. Der beste (schnellste) Weg, fachliche Hilfe zu diesem Buch zu erhalten, ist das *sehr* aktive Forum für Java-Anfänger auf javaranch.com.

Über die Übersetzer dieses Buchs
(die bei aller Berühmtheit nicht auf der Straße erkannt werden wollen)

Lars Schulten ist freier Übersetzer für IT-Fachliteratur und hat für den O'Reilly Verlag schon unzählige Bücher zu ungefähr allem übersetzt, was man mit Computern so anstellen kann. Eigentlich hat er mal Philosopie studiert, aber mit Computern schlägt er sich schon seit den Zeiten herum, in denen Windows laufen lernte. Die Liste der Dinge, mit denen er sich beschäftigt, ist ungefähr so lang, launenhaft und heterogen wie die seiner Lieblingsessen oder Lieblingsbücher.

Lars legt sich eben nicht gern fest. »Eine Klasse, eine Verantwortlichkeit», das ist sicher nicht seine Sache. Am besten funktioniert er, wenn man ihn als Universaladapter betrachtet und verdachtsweise einfach mal eine Methode aufruft und sich dann vom Ergebnis überraschen lässt.

Allein tritt er eigentlich nur auf, wenn er mal wieder versucht, den körperlichen Verfall mit sportlicher Betätigung aufzuhalten. Sonst ist er immer in Begleitung eines Buchs, seines Laptops oder Frederics unterwegs. Frederic ist vier Jahre alt und setzt gerne eine sehr kritische Miene auf, wenn Papa die Spielerei mit dem Conpuuta als Arbeit bezeichnet.

Elke Buchholz arbeitet als freiberufliche Übersetzerin und Lektorin. Sie hat Naturwissenschaften studiert und anschließend einige Jahre lang für verschiedene Verlage Fachliteratur übersetzt, unter anderem auch Lehrbücher. Nach einer mehrjährigen Tätigkeit als Angestellte in einem Aachener Verlag ist sie nun seit 2004 wieder selbstständig tätig.

Besonders gern übersetzt Elke inzwischen Informatikfachliteratur, denn seit sie vor einigen Jahren entdeckt hat, wie spannend die Java-Programmierung sein kann, hat sie sich von diesem Fachgebiet nicht wieder losreißen können. Nach ihrer Sun-Zertifizierung zum Java-Developer und zum Web Component Developer beschloss sie daher, statt weiterer Fremdsprachen in Zukunft lieber Programmiersprachen zu lernen.

Passend zu ihrem Beruf und ihrem Lieblingshobby trägt Elke BuchHolz einen »sprechenden Variablen(nach)-namen«. Als geborener Bücherwurm musste sie sich zwangsläufig auch beruflich irgendwann den Büchern zuwenden. Damit auch der zweite Teil ihres Namens nicht zu kurz kommt, spielt sie seit einigen Jahren in ihrer Freizeit begeistert ein Holzblasinstrument, die Klarinette.

Zur deutschen Übersetzung

Das leidige Thema der deutschen Umlaute im Code haben wir natürlich auch gründlich diskutiert. Der besseren Lesbarkeit halber haben wir schließlich in allen Arten von Bezeichnern Umlaute verwendet – auch in Klassennamen. Solange Sie innerhalb Ihres Systems bleiben und die Klassen nur dort kompilieren und ausführen, funktioniert dies in der Regel. Aber sobald Sie JARs erzeugen und/oder Ihre Programme auf anderen Plattformen einsetzen möchten, könnte es zu Problemen kommen. Sie sollten dann unbedingt in allen Verzeichnis- und Klassennamen (auch bei inneren Klassen!) auf Umlaute, auf das »ß» und Ähnliches verzichten. Profis, die meist sowieso zu englischen Bezeichnern neigen, empfinden Umlaute oft als irritierend und praxisfern – unser Anliegen ist es aber, dem lernenden Leser den Text, auch den Code, so leicht verdaulich zu präsentieren wie möglich.

Der Inhalt (in der Übersicht)

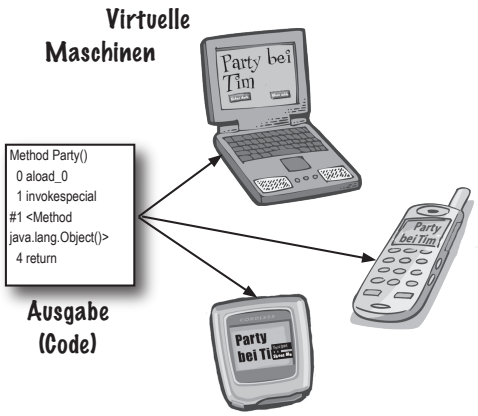
Der Inhalt (jetzt ausführlich)

Einführung

Ihr Gehirn und Java. Sie versuchen, etwas zu *lernen*, und Ihr *Hirn* tut sein Bestes, damit das Gelernte nicht *hängen bleibt*. Es denkt nämlich: »Wir sollten lieber ordentlich Platz für wichtigere Dinge lassen, z.B. für das Wissen, welche Tiere einem gefährlich werden könnten, oder dass es eine ganz schlechte Idee ist, nackt Snowboard zu fahren.« Tja, *wie* schaffen wir es nun, Ihr Gehirn davon zu überzeugen, dass Ihr Leben davon abhängt, etwas über Java zu wissen?

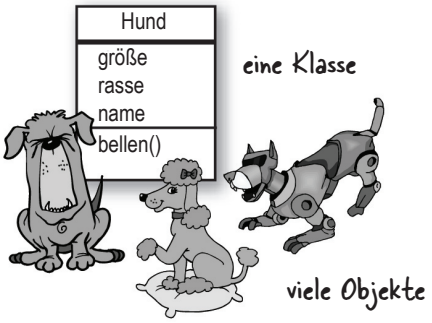
1 Die Oberfläche durchbrechen

Java führt Sie an neue Orte. Seit seinem bescheidenen Schritt an die Öffentlichkeit mit der (kümmerlichen) Version 1.02 hat Java mit seiner freundlichen Syntax, seinen objektorientierten Features, der Speicherverwaltung und natürlich mit dem Versprechen der Portabilität Programmierer verführt. Wir machen mal eine kleine Kostprobe und schreiben ein bisschen Code, kompilieren ihn und lassen ihn laufen. Wir sprechen über die Syntax, Schleifen, Verzweigungen und alles, was Java so cool macht. Springen Sie rein!



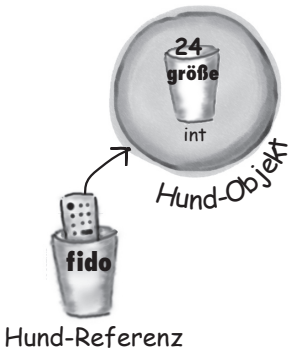
2 Ein Ausflug nach Objekthausen

Man hat mir gesagt, wir hätten es hier mit Objekten zu tun. Aber in Kapitel 1 haben wir unseren gesamten Code in die main()-Methode gesteckt. Das ist nicht sonderlich objektorientiert! Jetzt müssen wir diese prozedurale Welt hinter uns lassen und damit beginnen, eigene Objekte zu machen. Wir werden uns ansehen, was die objektorientierte Entwicklung in Java zu einem solchen Vergnügen macht. Wir werden uns den Unterschied zwischen einer Klasse und einem Objekt ansehen. Wir werden uns ansehen, wie Objekte Ihr Leben verbessern.



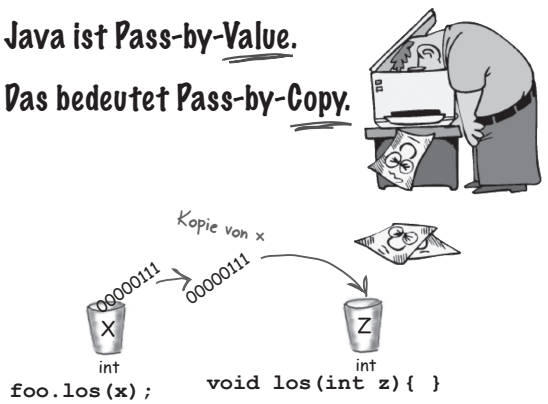
3 Verstehen Sie Ihre Variablen

Variablen gibt es in zwei Geschmacksrichtungen: elementare Typen und Referenztypen. Im Leben muss es doch mehr geben als Integer, Strings und Arrays. Was ist, wenn Sie ein HaustierBesitzer-Objekt mit einer Hund-Instanzvariablen haben? Oder ein Auto mit einem Motor? In diesem Kapitel werden wir die Mysterien von Java-Typen offen legen und uns ansehen, was Sie als Variable *deklarieren* können, was Sie in eine Variable *stecken* können und was Sie mit einer Variablen *tun* können. Und schließlich werden wir verstehen, wie das Leben, das sich auf dem Garbage Collectible Heap abspielt, *wirklich* ist.



4 Wie sich Objekte verhalten

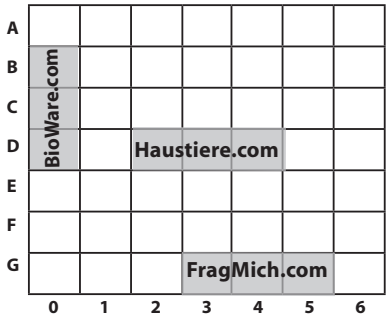
Zustände haben Auswirkungen auf Verhalten, und Verhalten hat Auswirkungen auf Zustände. Wir wissen, dass Objekte **Zustände** und **Verhalten** haben, die von **Instanzvariablen** und **Methoden** repräsentiert werden. Nun sehen wir uns an, in welcher ***Beziehung*** Zustände und Verhalten zueinander ***stehen***. Objekte haben ein ***Verhalten***, das auf ihre ***Zustände*** wirkt. Anders gesagt, ***Methoden nutzen die Werte von Instanzvariablen –*** beispielsweise »wenn Hund weniger wiegt als 7 Kilo, mach Kläffgeräusch« oder »erhöhe Gewicht um 3 Kilo«. ***Ändern wir mal ein paar Zustände***.



5 Methoden mit Superkraft

Jetzt werden wir unseren Methoden etwas mehr Muskelkraft geben. Wir haben uns an Variablen versucht, haben mit ein paar Objekten gespielt und haben etwas Code geschrieben. Jetzt brauchen wir mehr Werkzeuge. Wie ***Operatoren***. Und **Schleifen.** Es könnte auch nützlich sein, **Zufallszahlen zu generieren**. Oder **einen String in ein int umzuwandeln**, ja, das wäre cool. Und warum lernen wir nicht all den Kram, während wir was Echtes *schreiben*, um zu erfahren, wie es ist, wenn man ein Programm von der Pike auf schreiben (und testen) muss? **Vielleicht ein Spiel**, in dem wir Dot-Coms abschießen (wie bei Schiffe versenken).

Wir werden das Spiel »Dot-Coms versenken« entwickeln



6 Die Java-Bibliothek verwenden

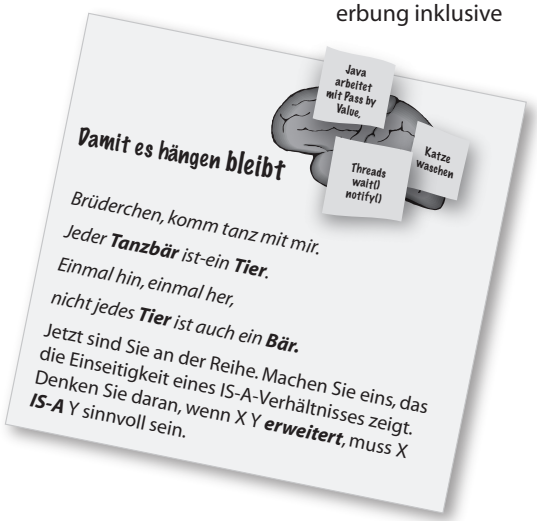
Java wird mit Hunderten vorgefertigter Klassen ausgeliefert. Sie müssen das Rad nicht jedes Mal neu erfinden, wenn Sie wissen, wie Sie das, was Sie brauchen, in der Java-Bibliothek finden, die als die **Java-API** bezeichnet wird. *Schließlich haben Sie etwas Besseres zu tun.* Wenn Sie Code schreiben, können Sie sich ruhig auf die Teile beschränken, die bei Ihrer Anwendung besonders sind. Die Java-Kernbibliothek ist ein gigantischer Haufen von Klassen, die nur darauf warten, dass Sie sie als Bausteine einsetzen.

*»Gut zu wissen, dass es im Package java.util eine ArrayList gibt. Aber wie hätte **ich** das allein rausfinden können?«*

- Julia, 31, Hand-Model

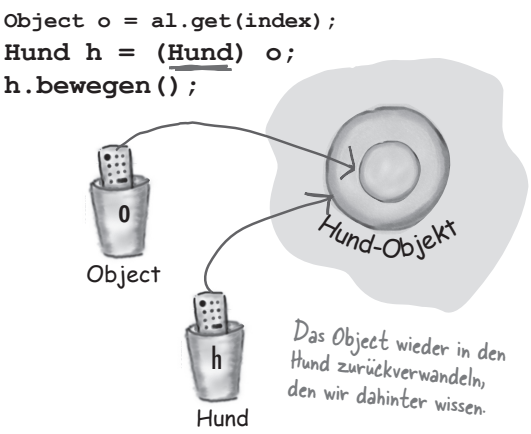
7 Besser leben in Objekthausen

Planen Sie Ihre Programme mit Blick auf die Zukunft Was wäre, wenn Sie Code schreiben könnten, den ein *anderer* **problemlos** erweitern kann? Und was wäre, wenn Ihr Code flexibel genug für diese verfluchten Spezifikationsänderungen in letzter Minute wäre? Wenn Sie auf den Polymorphie-Zug springen, lernen Sie die fünf Schritte zu besserem Klassen-Design, die drei Tricks der Polymorphie und die acht Wege, Code flexibel zu machen. Dazu, wenn Sie sofort zugreifen – eine Bonusstunde mit den vier Tipps zur Nutzung von Vererbung inklusive



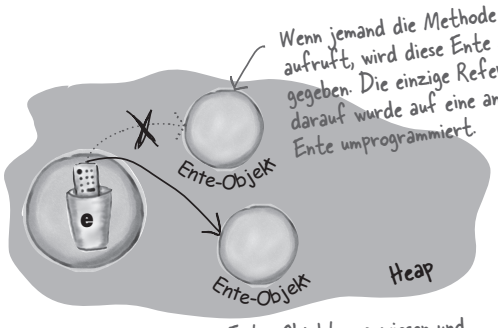
8 Ernsthafte Polymorphie

Vererbung ist nur der Anfang. Um Polymorphie richtig zu nutzen, brauchen wir Schnittstellen. Wir müssen über einfache Vererbung hinausgehen und eine Stufe der Flexibilität erreichen, die man nur erhält, wenn man Schnittstellendefinitionen als Ausgangsbasis für den Entwurf und die Programmierung nimmt. Was eine Java-Schnittstelle ist? Eine 100% abstrakte Klasse. Was eine abstrakte Klasse ist? Das ist eine Klasse, die nicht instantiiert werden kann. Und wozu soll die gut sein? Lesen Sie das Kapitel...



9 Konstruktoren und Garbage Collection

Objekte werden geboren und Objekte sterben. Sie herrschen über das Leben eines Objekts. Sie entscheiden, wie und wann ein Objekt *konstruiert* wird. Sie entscheiden, wann es aufgegeben werden soll. Der **Garbage Collector (gc)** will den Speicher wieder haben. In diesem Kapitel sehen wir uns an, wie Objekte erzeugt werden, wie sie leben, während sie leben, und wie Sie sie effektiv bewahren oder aufgeben. Das bedeutet, dass wir über den Heap, den Stack, Geltungsbereiche, Konstruktoren, Super-Konstruktoren, null-Referenzen und anderes reden werden.



10 Zahlen, bitte!

Alle Instanzen der gleichen Klasse teilen sich ein einziges Exemplar der statischen Variablen.

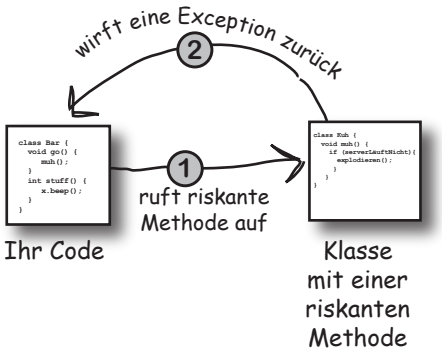
Instanzvariablen: eine pro Instanz

statische Variablen: eine pro Klasse

Rechnen Sie's aus. In der Java-API gibt es eine Menge praktischer Methoden für Absolutbeträge, zum Runden, zur Extremwertbestimmung etc. Aber wie sieht es mit der Formatierung aus? Vielleicht möchten Sie, dass Ihre Zahlen mit exakt zwei Nachkommastellen ausgegeben oder dass große Zahlen mit Trennzeichen unterteilt werden, damit man sie leichter lesen kann. Und wie steht es mit Datumsangaben? Als Erstes sehen wir uns einmal an, was es für eine Variable oder eine Methode bedeutet, statisch zu.

11 Riskantes Verhalten

Manches passiert einfach. Die Datei ist nicht da. Der Server läuft nicht. Und wenn Sie ein noch so guter Programmierer sind – Sie können nicht *alles* unter Kontrolle haben. Wenn Sie eine riskante Methode schreiben, brauchen Sie Code, der all das Schlimme, das möglicherweise passiert, behandelt. Aber woher *wissen* Sie, wann eine Methode riskant ist? Und wo platzieren Sie den Code, der die ***Ausnahme***situation *behandelt*? In *diesem* Kapitel programmieren wir einen MIDI-Musikplayer, der die riskante Java-Sound-API benutzt – daher sollten wir das besser alles schnell rausfinden!



12 Innen hui, außen GUI

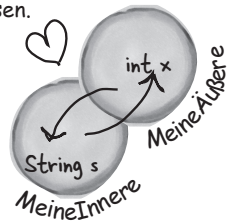
Sehen Sie den Tatsachen ins Auge: Um GUIs (grafische Benutzeroberflächen) kommen Sie nicht herum. Selbst wenn Sie davon ausgehen, dass Sie den Rest Ihres Lebens serverseitigen Code schreiben, müssen Sie früher oder später Tools entwickeln, und dafür hätten Sie dann ganz sicher gern eine grafische Schnittstelle. Wir werden zwei Kapitel mit der Programmierung von GUIs zubringen und dabei grundlegende Features der Sprache Java kennen lernen, z.B. **Event-Handling** und **innere Klassen.** In diesem Kapitel setzen wir einen Button auf den Bildschirm und lassen ihn reagieren, wenn man ihn anklickt. Außerdem zeichnen wir etwas auf den Bildschirm, zeigen ein JPEG-Bild an und versuchen uns sogar an einer kleinen Animation.

```
class MeineÄußereKlasse {

  class MeineInnereKlasse {
    void los() {
    }
  }

}
```

Innere Klasse ist vollständig von äußerer Klasse umschlossen.



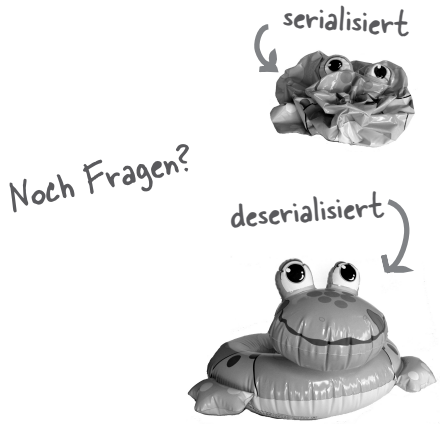
Diese beiden Objekte auf dem Heap haben eine ganz besondere Bindung aneinander. Das innere kann die Variablen des äußeren benutzen (und umgekehrt).

13 Mehr Schwung mit Swing

Swing ist einfach. Wenn es Ihnen nicht so wichtig ist, wo die Dinge auf dem Bildschirm landen. Swing-Code *sieht einfach aus*, aber dann kompilieren Sie das Ganze, führen es aus, sehen es sich an und denken: »Hey, *das* sollte aber nicht *dort* stehen ...« Der Grund dafür, dass der Code so leicht zu *schreiben* ist, ist der **Layoutmanager** – aber genau der macht es auch so *schwer kontrollierbar*. Mit ein bisschen Mühe bringen Sie Layoutmanager jedoch dazu, sich Ihrem Willen zu unterwerfen. In diesem Kapitel bringen wir unsere Swing-Künste in Schwung und lernen etwas über Widgets.

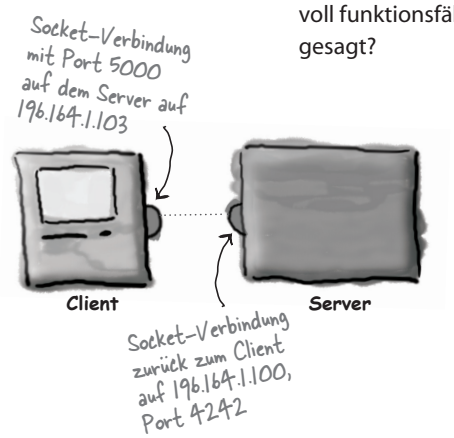
14 Speicherung von Objekten

Objekte lassen sich flach drücken und wieder aufblasen. Objekte haben einen Zustand und ein Verhalten. Das *Verhalten* steckt in der *Klasse,* aber der *Zustand* steckt in jedem einzelnen *Objekt*. Muss Ihr Programm Zustände speichern, *können Sie das natürlich auf die mühsame Weise tun* – Sie befragen jedes Objekt und notieren dann akribisch den Wert jeder Instanzvariablen. Oder **Sie machen es auf die einfache, objektorientierte Weise** – indem Sie lediglich das Objekt selbst gefriertrocknen (serialisieren) und es mit Wasser anrühren (deserialisieren), wenn Sie es zurückhaben wollen.



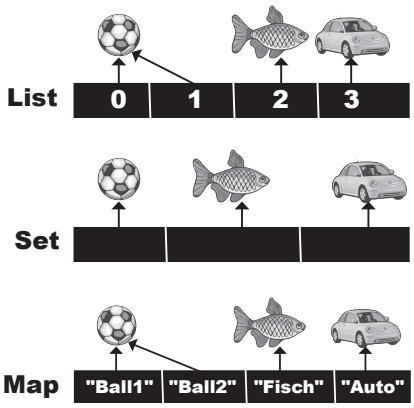
15 Eine Verbindung herstellen

Nehmen Sie Verbindung mit der Außenwelt auf. Das ist nicht schwer. Um all die Lowlevel-Netzwerk-Details kümmern sich Klassen in der java.net-Bibliothek. Einer der großen Vorteile von Java ist, dass das Senden und Empfangen über ein Netz einfache Eingabe/Ausgabe ist, lediglich mit einem etwas anderen Verbindungsstrom am Ende der Kette. In diesem Kapitel machen wir *Client*-Sockets. Wir machen *Server*-Sockets. Wir machen *Clients* und *Server*. Und wir lassen sie miteinander reden. Bevor Sie mit dem Kapitel durch sind, haben Sie einen voll funktionsfähigen und Multithread-tauglichen Chat-Client. Haben wir da gerade *Multithread* gesagt?



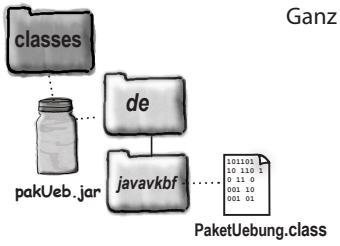
16 Datenstrukturen

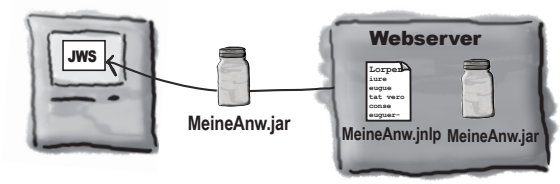
Sortieren ist in Java ein Klacks. Sie haben alle Werkzeuge zur Verfügung, um Daten zu sammeln und damit zu arbeiten, ohne Ihre eigenen Sortieralgorithmen schreiben zu müssen. Das Collections-Framework von Java enthält eine Datenstruktur für praktisch alles, was Sie jemals tun müssen. Sie wollen eine Liste, zu der Sie leicht etwas hinzufügen können? Sie wollen etwas mit Hilfe seines Namens suchen? Oder Sie brauchen eine Liste, die automatisch alles entfernt, was doppelt vorhanden ist? Sie möchten Ihre Kollegen danach sortieren, wie oft sie Ihnen in den Rücken gefallen sind? Es ist alles da ...



17 Veröffentlichen Sie Ihren Code

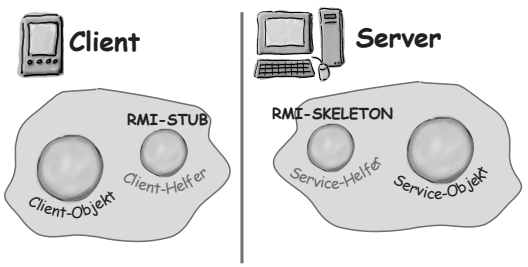
Es ist Zeit loszulassen. Sie haben Ihren Code geschrieben. Sie haben ihn getestet. Sie haben ihn verbessert. Sie haben allen Bekannten erzählt, es wäre schön, wenn Sie nie wieder eine einzige Zeile Code sehen würden. Und doch haben Sie zu guter Letzt ein Kunstwerk geschaffen. Tatsächlich, das Ding läuft! Aber was nun? In den beiden letzten Kapiteln des Buchs befassen wir uns mit dem Organisieren, Packen und Deployment des Codes. Wir sehen uns lokale und entfernte Deployment-Möglichkeiten an, inklusive ausführbarer JARs, RMI und Servlets. Ganz ruhig – einige der coolsten Dinge, die Java bietet, sind einfacher, als Sie denken.





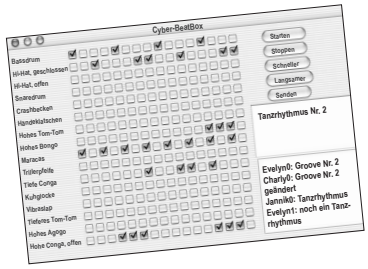
18 Verteilte Anwendungen

Entfernung muss kein Problem sein. Sicher, es *ist* einfacher, wenn alle Teile Ihrer Anwendung an einem Ort sind, auf einem Heap, mit einer JVM, die über alles herrscht. Aber das ist nicht immer möglich. Und auch nicht immer wünschenswert. Was ist, wenn Ihr Programm leistungsintensive Berechnungen durchführen soll? Was ist, wenn Ihr Programm Daten aus einer sicheren Datenbank braucht? In diesem Kapitel lernen wir, wie man die erstaunlich einfache Java-Technik namens Remote Method Invocation (RMI) einsetzt. Wir werfen auch einen kurzen Blick auf Servlets, Enterprise Java Beans (EJB) und Jini.



A Anhang A

Das CodeKüchen-Finale! Der vollständige Code für unser BeatBox-Client-Programm – Ihre Chance, ein Rockstar zu werden!

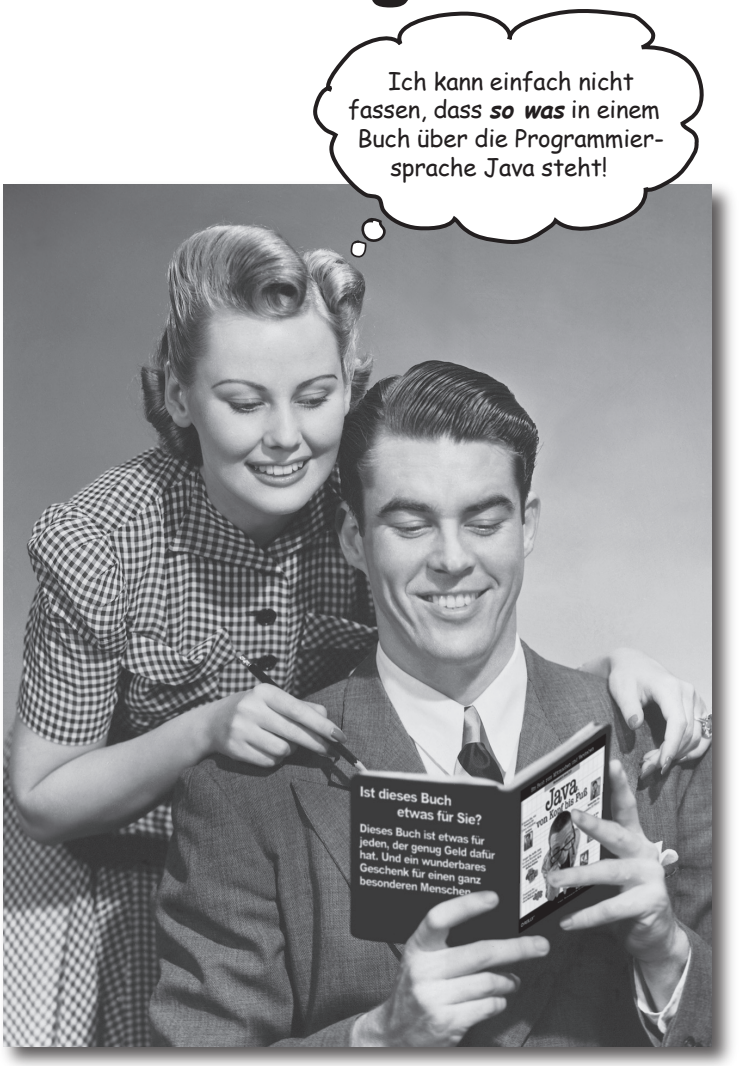


B Anhang B

Die Top Ten der Themen, die es nicht ins Buch geschafft haben. Noch können wir Sie nicht ganz gehen lassen. Ein paar Sachen haben wir noch, aber dann sind Sie fertig. Diesmal meinen wir es ernst.

I Index

Einführung



In diesem Abschnitt beantworten wir die brennende Frage:
»Und? Warum STEHT so was in einem Buch über die Programmiersprache Java?«

Für wen ist dieses Buch?

Wenn Sie ***alle*** folgenden Fragen mit »Ja« beantworten können ...

1. **Haben Sie schon Programmiererfahrung?**
2. **Wollen Sie Java lernen?**
3. **Ziehen Sie eine anregende Unterhaltung beim Abendessen einer trockenen, langweiligen Vorlesung vor?**

... dann ist dieses Buch etwas für Sie.

Dies ist KEIN Nachschlagewerk. Java von Kopf bis Fuß ist ein Buch zum *Lernen*, keine Enzyklopädie mit Java-Fakten.

Wer sollte eher die Finger von diesem Buch lassen?

Wenn Sie ***eine*** der folgenden Fragen mit »Ja« beantworten können ...

1. **Beschränken sich Ihre Programmierkenntnisse auf HTML? Haben Sie keine Erfahrung mit Skriptsprachen?**

   (Wenn Sie schon irgendetwas mit Schleifen oder Wenn-dann-Logik gemacht haben, werden Sie mit diesem Buch prima zurechtkommen, aber HTML-Tags allein reichen vielleicht nicht.)

2. **Sind Sie ein Top-C++-Entwickler, der ein Buch zum *Nachschlagen* sucht?**

3. **Haben Sie Angst, etwas Neues auszuprobieren? Ist Ihnen eine Zahnwurzelbehandlung lieber, als Streifen kombiniert mit Karos zu tragen? Glauben Sie, dass ein Technikfachbuch nicht seriös sein kann, wenn im Abschnitt über Speicherverwaltung das Bild einer Ente vorkommt?**

... dann ist dieses Buch ***nicht*** das Richtige für Sie.

[Anmerkung der Marketing-Abteilung: Wer hat den Abschnitt darüber gestrichen, dass dieses Buch etwas für jeden Besitzer einer gültigen Kreditkarte ist? Und was ist mit dieser »Legen-Sie-Java-auf-den-Gabentisch«-Werbung? ... – Fred]

Wir wissen, was Sie gerade denken.

»Kann das wirklich ein seriöses Programmierlehrbuch sein?«

»Was ist mit all den Abbildungen?«

»Kann ich das auf diese Art wirklich lernen?«

»Riecht's hier nicht nach Pizza?«

Und wir wissen, was Ihr *Gehirn* gerade denkt.

Ihr Gehirn lechzt nach Neuem. Es ist ständig dabei, Ihre Umgebung abzusuchen, und es *wartet* auf etwas Ungewöhnliches. So ist es nun einmal gebaut, und es hilft Ihnen zu überleben.

Heutzutage ist es weniger wahrscheinlich, dass Sie von einem Tiger verputzt werden. Aber Ihr Gehirn hält immer noch Ausschau. Man weiß ja nie.

Also, was macht Ihr Gehirn mit all den gewöhnlichen, normalen Routinesachen, denen Sie begegnen? Es tut alles in seiner Macht stehende, damit es dadurch nicht bei seiner *eigentlichen* Arbeit gestört wird: Dinge zu erfassen, die wirklich *wichtig* sind. Es gibt sich nicht damit ab, die langweiligen Sachen zu speichern, sondern lässt diese gar nicht erst durch den »Dies-ist-offensichtlich-nicht-wichtig«-Filter.

Woher *weiß* Ihr Gehirn denn, was wichtig ist? Nehmen Sie an, Sie machen einen Tagesausflug und ein Tiger springt vor Ihnen aus dem Gebüsch: Was passiert dabei in Ihrem Kopf und Ihrem Körper?

Neuronen feuern. Gefühle werden angekurbelt. *Chemische Substanzen durchfluten Sie.*

Und so weiß Ihr Gehirn:

Dies muss wichtig sein! Vergiss es nicht!

Aber nun stellen Sie sich vor, Sie sind zu Hause oder in einer Bibliothek. In einer sicheren, warmen, tigerfreien Zone. Sie lernen. Bereiten sich auf eine Prüfung vor. Oder Sie versuchen, irgendein schwieriges Thema zu lernen, von dem Ihr Chef glaubt, Sie bräuchten dafür eine Woche oder höchstens zehn Tage.

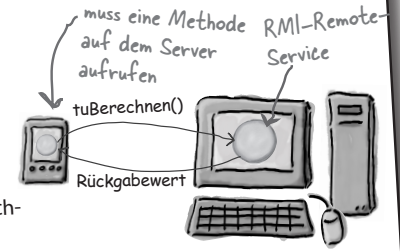
Da ist nur ein Problem: Ihr Gehirn versucht Ihnen einen großen Gefallen zu tun. Es versucht dafür zu sorgen, dass diese *offensichtlich* unwichtigen Inhalte nicht knappe Ressourcen verstopfen. Ressourcen, die besser dafür verwendet würden, die wirklich *wichtigen* Dinge zu speichern. Wie Tiger. Wie die Gefahren des Feuers. Oder dass Sie nie wieder in Shorts snowboarden sollten.

Und es gibt keine einfache Möglichkeit, Ihrem Gehirn zu sagen: »Hey, Gehirn, vielen Dank, aber egal, wie langweilig dieses Buch auch ist und wie klein der Ausschlag auf meiner emotionalen Richterskala gerade ist, ich *will* wirklich, dass du diesen Kram behältst.«

Wir stellen uns unseren Leser als einen aktiv Lernenden vor.

Also, was ist nötig, damit Sie etwas lernen? Erst einmal müssen Sie es aufnehmen und dann dafür sorgen, dass Sie es nicht wieder vergessen. Es geht nicht darum, Fakten in Ihren Kopf zu schieben. Nach den neuesten Forschungsergebnissen der Kognitionswissenschaft, der Neurobiologie und der Lernpsychologie gehört zum Lernen viel mehr als nur Text auf einer Seite. Wir wissen, was Ihr Gehirn anmacht.

Einige der Lernprinzipien dieser Buchreihe:



Wir setzen Bilder ein. An Bilder kann man sich viel besser erinnern als an Worte allein und lernt so viel effektiver (bis zu 89% Verbesserung bei Abrufbarkeits- und Lerntransferstudien). Außerdem werden die Dinge dadurch verständlicher. **Wir setzen Text in oder neben die Grafiken,** auf die sie sich beziehen, anstatt darunter oder auf eine andere Seite. Die Leser werden auf den Bildinhalt bezogene Probleme dann mit *doppelt* so hoher Wahrscheinlichkeit lösen können.

Wir verwenden einen gesprächsorientierten Stil mit persönlicher Ansprache. Nach neueren Untersuchungen haben Studenten nach dem Lernen bei Tests bis zu 40% besser abgeschnitten, wenn der Inhalt den Leser direkt in der ersten Person und im lockeren Stil angesprochen hat statt in einem formalen Ton. Wir halten keinen Vortrag, sondern erzählen Ihnen Geschichten. Wir benutzen eine zwanglose Sprache. Wir versuchen, uns selbst nicht zu ernst zu nehmen. Denn: Würden *Sie* einer anregenden Unterhaltung beim Abendessen mehr Aufmerksamkeit schenken oder einem Vortrag?

Es ist wirklich ätzend, eine abstrakte Methode zu sein, so ganz ohne Rumpf ...

Kein Methodenrumpf! Am Schluss steht ein Semikolon.

Wir wollen Sie dazu bringen, intensiver nachzudenken. Mit anderen Worten: Falls Sie nicht aktiv Ihre Neuronen strapazieren, passiert in Ihrem Gehirn nicht viel. Ein Leser muss motiviert, begeistert und neugierig sein und angeregt werden, Probleme zu lösen, Schlüsse zu ziehen und sich neues Wissen anzueignen. Und dafür brauchen Sie Herausforderungen, Übungen, zum Nachdenken anregende Fragen und Tätigkeiten, die beide Seiten des Gehirns und mehrere Sinne einbeziehen.

Ergibt es einen Sinn zu sagen: Wanne IST-EIN Badezimmer? Badezimmer IST-EINE Wanne? Oder ist das eine HAT-EINE-Beziehung?

Wir versuchen, Ihre Aufmerksamkeit zu erlangen – und sie zu behalten. Wir alle haben schon Erfahrungen dieser Art gemacht: »Ich will das wirklich lernen, aber ich kann einfach nicht über Seite 1 hinaus wach bleiben.« Ihr Gehirn passt auf, wenn Dinge ungewöhnlich, interessant, merkwürdig, auffällig, unerwartet sind. Ein neues, schwieriges, technisches Thema zu lernen muss nicht langweilig sein. Wenn es das nicht ist, lernt Ihr Gehirn viel schneller.

Wir sprechen Gefühle an. Wir wissen, dass Ihre Fähigkeit, sich an etwas zu erinnern, wesentlich von dessen emotionalem Gehalt abhängt. Sie erinnern sich an das, was Sie *bewegt*. Sie erinnern sich, wenn Sie etwas *fühlen*. Nein, wir erzählen keine herzzerreißenden Geschichten über einen Jungen und seinen Hund. Was wir erzählen, ruft Überraschungs-, Neugier-, Spaß- und Was-soll-das?-Emotionen hervor und dieses Hochgefühl, das Sie beim Lösen eines Puzzles empfinden oder wenn Sie etwas lernen, was alle anderen schwierig finden. Oder wenn Sie merken, dass Sie etwas können, das dieser »Ich-bin-ein-besserer-Techniker-als-du«-Typ aus der Technikabteilung *nicht kann*.

Metakognition: Nachdenken übers Denken

Wenn Sie wirklich lernen möchten, und zwar schneller und nachhaltiger, dann schenken Sie Ihrer Aufmerksamkeit Aufmerksamkeit. Denken Sie darüber nach, wie Sie denken. Lernen Sie, wie Sie lernen.

Die Meisten von uns haben in ihrer Jugend keine Kurse in Metakognition oder Lerntheorie gehabt. Es wurde von uns *erwartet,* dass wir lernen, aber nur selten wurde uns auch *beigebracht,* wie man lernt.

Wir nehmen aber an, dass Sie wirklich Java lernen möchten, wenn Sie dieses Buch in den Händen halten. Und wahrscheinlich möchten Sie nicht viel Zeit aufwenden.

Wenn Sie so viel wie möglich von diesem Buch profitieren wollen oder von irgendeinem anderen Buch oder einer anderen Lernerfahrung, übernehmen Sie Verantwortung für Ihr Gehirn. Ihr Gehirn im Zusammenhang mit *diesem* Lernstoff.

Der Trick besteht darin, Ihr Gehirn dazu zu bringen, neuen Lernstoff als etwas wirklich Wichtiges anzusehen. Als entscheidend für Ihr Wohlbefinden. So wichtig wie ein Tiger. Andernfalls stecken Sie in einem dauernden Kampf, in dem Ihr Gehirn sein Bestes gibt, um die neuen Inhalte davon abzuhalten, hängen zu bleiben.

Wie bringen Sie also Ihr Gehirn dazu, Java so zu behandeln, als wäre es ein hungriger Tiger?

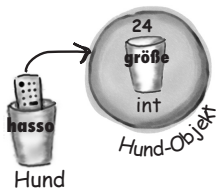
Da gibt es den langsamen, ermüdenden Weg oder den schnelleren, effektiveren Weg. Der langsame Weg führt über bloße Wiederholung. Natürlich ist Ihnen klar, dass Sie lernen und sich sogar an die langweiligsten Themen erinnern *können,* wenn Sie sich die gleiche Sache immer wieder einhämmern. Wenn Sie nur oft genug wiederholen, sagt Ihr Gehirn: „Er hat zwar nicht das *Gefühl,* dass das wichtig ist, aber er sieht sich dieselbe Sache *immer und immer wieder* an – dann muss sie wohl wichtig sein."

Der schnellere Weg besteht darin, ***alles zu tun, was die Gehirnaktivität erhöht***, vor allem verschiedene Arten von Gehirnaktivität. Eine wichtige Rolle dabei spielen die auf der vorhergehenden Seite erwähnten Dinge – alles Dinge, die nachweislich helfen, dass Ihr Gehirn *für* Sie arbeitet. So hat sich z.B. in Untersuchungen gezeigt: Wenn Wörter *in* den Abbildungen stehen, die sie beschreiben (und nicht irgendwo anders auf der Seite, z.B. in einer Bildunterschrift oder im Text), versucht Ihr Gehirn herauszufinden, wie die Wörter und das Bild zusammenhängen, und dadurch feuern mehr Neuronen. Und je mehr Neuronen feuern, umso größer ist die Chance, dass Ihr Gehirn mitbekommt: Bei dieser Sache lohnt es sich aufzupassen und vielleicht auch, sich daran zu erinnern.

Ein lockerer Sprachstil hilft, denn Menschen tendieren zu höherer Aufmerksamkeit, wenn ihnen bewusst ist, dass sie ein Gespräch führen – man erwartet dann ja von ihnen, dass sie dem Gespräch folgen und sich beteiligen. Das Erstaunliche daran ist: Es ist Ihrem Gehirn ziemlich egal, dass die „Unterhaltung" zwischen Ihnen und einem Buch stattfindet! Wenn der Schreibstil dagegen formal und trocken ist, hat Ihr Gehirn den gleichen Eindruck wie bei einem Vortrag, bei dem Sie in einem Raum voll passiver Zuhörer sitzen. Nicht nötig, wach zu bleiben.

Aber Abbildungen und ein lockerer Sprachstil sind erst der Anfang.

Das haben WIR getan:



Wir haben ***Bilder*** verwendet, weil Ihr Gehirn auf visuelle Eindrücke eingestellt ist, nicht auf Text. Soweit es Ihr Gehirn betrifft, sagt ein Bild *wirklich* mehr als 1.024 Worte. Und dort, wo Text und Abbildungen zusammenwirken, haben wir den Text *in* die Bilder eingebettet, denn Ihr Gehirn arbeitet besser, wenn der Text *innerhalb* der Sache steht, auf die er sich bezieht, und nicht in einer Bildunterschrift oder irgendwo vergraben im Text.

Wir haben ***Wiederholung*** eingesetzt, d.h. dasselbe auf *unterschiedliche* Art und mit verschiedenen Medientypen ausgedrückt, damit Sie es über *mehrere Sinne* aufnehmen. Das erhöht die Chance, dass die Inhalte an mehr als nur einer Stelle in Ihrem Gehirn verankert werden.

Wir haben Konzepte und Bilder in ***unerwarter*** Weise eingesetzt, weil Ihr Gehirn auf Neuigkeiten programmiert ist. Und wir haben Bilder und Ideen mit zumindest *etwas* ***emotionalem*** *Charakter* verwendet, weil Ihr Gehirn darauf eingestellt ist, auf die Biochemie von Gefühlen zu achten. An alles, was ein *Gefühl* in Ihnen auslöst, können Sie sich mit höherer Wahrscheinlichkeit erinnern, selbst wenn dieses Gefühl nicht mehr ist als ein bisschen ***Belustigung, Überraschung*** oder ***Interesse.***

Wir haben einen ***umgangssprachlichen Stil*** mit direkter Anrede benutzt, denn Ihr Gehirn ist von Natur aus aufmerksamer, wenn es Sie in einer Unterhaltung wähnt als wenn es davon ausgeht, dass Sie passiv einer Präsentation zuhören – sogar dann, wenn Sie *lesen.*

Wir haben mehr als 50 ***Übungen*** für Sie vorgesehen, denn Ihr Gehirn lernt und behält von Natur aus besser, wenn Sie Dinge ***tun,*** als wenn Sie nur darüber *lesen.* Und wir haben die Übungen zwar anspruchsvoll, aber doch lösbar gemacht, den so ist es den meisten *Lesern* am liebsten.

Wir haben ***mehrere unterschiedliche Lernstile*** eingesetzt, denn vielleicht bevorzugen *Sie* ein Schritt-für-Schritt-Vorgehen, während jemand anders erst einmal den groben Zusammenhang verstehen und ein Dritter einfach nur ein Codebeispiel sehen möchte. Aber ganz abgesehen von den jeweiligen Lernvorlieben profitiert *jeder* davon, wenn er die gleichen Inhalte in unterschiedlicher Form präsentiert bekommt.

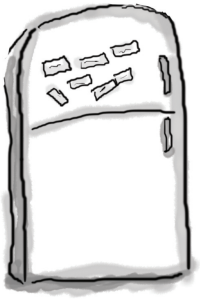
Wir liefern Inhalte für ***beide Seiten Ihres Gehirns,*** denn je mehr von Ihrem Gehirn Sie einsetzen, umso wahrscheinlicher werden Sie lernen und behalten und umso länger bleiben Sie konzentriert. Wenn Sie mit einer Seite des Gehirns arbeiten, bedeutet das häufig, dass sich die andere Seite des Gehirns ausruhen kann; so können Sie über einen längeren Zeitraum produktiver lernen.

Und wir haben ***Geschichten*** und Übungen aufgenommen, die ***mehr als einen Blickwinkel repräsentieren,*** denn Ihr Gehirn lernt von Natur aus intensiver, wenn es gezwungen ist, selbst zu analysieren und zu beurteilen.

Wir haben ***Herausforderungen*** eingefügt, in Form von Übungen und indem wir ***Fragen*** stellen, auf die es nicht immer eine eindeutige Antwort gibt, denn Ihr Gehirn ist darauf eingestellt, zu lernen und sich zu erinnern, wenn es an etwas *arbeiten* muss (so wie Sie ja auch Ihren *Körper* nicht in Form bekommen können, indem Sie andere auf dem Sportplatz *beobachten).* Aber wir haben unser Bestes getan, um dafür zu sorgen, dass Sie – wenn Sie schon hart arbeiten – an den *richtigen* Dingen arbeiten. Dass Sie ***nicht einen einzigen Dendriten darauf verschwenden,*** ein schwer verständliches Beispiel zu verarbeiten oder einen schwierigen, mit Fachbegriffen gespickten oder extrem gedrängten Text zu analysieren.

Wir haben ***Menschen*** eingesetzt. In Geschichten, Beispielen, Bildern usw. – denn *Sie sind* ein Mensch. Und Ihr Gehirn schenkt *Menschen* mehr Aufmerksamkeit als *Dingen.*

Wir haben einen ***80/20***-Ansatz benutzt. Wir gehen davon aus, dass dies nicht Ihr einziges Buch sein wird, wenn Sie einen Doktor in Java machen wollen. Deshalb besprechen wir nicht *alles.* Nur das, was Sie wirklich *verwenden* werden.



Schneiden Sie dies aus und heften Sie es an Ihren Kühlschrank.

Und das können SIE tun, um sich Ihr Gehirn untertan zu machen

So, wir haben unseren Teil der Arbeit geleistet. Der Rest liegt bei Ihnen. Diese Tipps sind ein Anfang; hören Sie auf Ihr Gehirn und finden Sie heraus, was bei Ihnen funktioniert und was nicht. Probieren Sie neue Wege aus.

1. **Immer langsam. Je mehr Sie verstehen, umso weniger müssen Sie auswendig lernen.**

   Lesen Sie nicht nur. Halten Sie inne und denken Sie nach. Wenn das Buch Sie etwas fragt, springen Sie nicht einfach zur Antwort. Stellen Sie sich vor, dass Sie das wirklich jemand *fragt.* Je gründlicher Sie Ihr Gehirn zum Nachdenken zwingen, umso größer ist die Chance, dass Sie lernen und behalten.

2. **Bearbeiten Sie die Übungen. Machen Sie selbst Notizen.**

   Wir haben sie entworfen, aber wenn wir sie auch für Sie lösen würden, wäre dass, als ob jemand anderes Ihr Training für Sie absolviert. Und sehen Sie sich die Übungen *nicht einfach nur an.* **Benutzen Sie einen Bleistift.** Es deutet vieles darauf hin, dass körperliche Aktivität *beim* Lernen den Lernerfolg erhöhen kann.

3. **Lesen Sie die Abschnitte »Es gibt keine Dummen Fragen«.**

   Und zwar alle. Das sind keine Zusatzanmerkungen – sie gehören zum Kerninhalt! Aus den Fragen lernen Sie manchmal noch mehr als aus den Antworten.

4. **Lesen Sie nicht alles am gleichen Ort.**

   Stehen Sie auf, recken Sie sich, gehen Sie herum, wechseln Sie den Stuhl, gehen Sie in einen anderen Raum. Das trägt dazu bei, dass Ihr Gehirn etwas *fühlt,* und verhindert, dass Ihr Lernerfolg an einen bestimmten Ort gebunden ist.

5. **Lesen Sie dies als Letztes vor dem Schlafengehen. Oder lesen Sie danach zumindest nichts *Anspruchsvolles* mehr.**

   Ein Teil des Lernprozesses (vor allem die Übertragung in das Langzeitgedächtnis) findet erst statt, *nachdem* Sie das Buch zur Seite gelegt haben. Ihr Gehirn braucht Zeit für sich, um weitere Verarbeitung zu leisten. Wenn Sie in dieser Zeit etwas Neues aufnehmen, geht ein Teil dessen, was Sie gerade gelernt haben, verloren.

6. **Trinken Sie Wasser. Viel.**

   Ihr Gehirn arbeitet am besten in einem schönen Flüssigkeitsbad. Austrocknung (zu der es schon kommen kann, bevor Sie überhaupt Durst verspüren) beeinträchtigt die kognitive Funktion.

7. **Reden Sie drüber. Laut.**

   Sprechen aktiviert einen anderen Teil des Gehirns. Wenn Sie etwas verstehen wollen oder Ihre Chancen verbessern wollen, sich später daran zu erinnern, sagen Sie es laut. Noch besser: Versuchen Sie es jemand anderem laut zu erklären. Sie lernen dann schneller und haben vielleicht Ideen, auf die Sie beim bloßen Lesen nie gekommen wären.

8. **Hören Sie auf Ihr Gehirn.**

   Achten Sie darauf, Ihr Gehirn nicht zu überladen. Wenn Sie merken, dass Sie etwas nur noch überfliegen oder dass Sie das gerade erst Gelesene vergessen haben, ist es Zeit für eine Pause. Ab einem bestimmten Punkt lernen Sie nicht mehr schneller, indem Sie mehr hineinzustopfen versuchen; das kann sogar den Lernprozess stören.

9. **Aber bitte mit *Gefühl*!**

   Ihr Gehirn muss wissen, dass es *um etwas Wichtiges geht.* Lassen Sie sich in die Geschichten hineinziehen. Erfinden Sie eigene Bildunterschriften für die Fotos. Über einen schlechten Scherz zu stöhnen ist *immer noch* besser, als gar nichts zu fühlen.

10. **Tippen Sie den Code ab und führen Sie ihn aus.**

    Tippen Sie die Codebeispiele ab und führen Sie sie aus. Dann können Sie herumexperimentieren und den Code verändern und verbessern (oder ihn kaputtmachen – das ist manchmal der beste Weg, um herauszufinden, was wirklich passiert). Bei langen Beispielen oder dem »Fertiggericht«-Code können Sie die Quellcode-Dateien von der Webseite zum Buch im Online-Katalog von O'Reilly herunterladen.

Was Sie für dieses Buch brauchen:

Was Sie *nicht* brauchen, ist ein zusätzliches Entwicklungswerkzeug wie z.B. eine integrierte Entwicklungsumgebung (Integrated Development Environment, abgekürzt IDE). Wir empfehlen Ihnen nachdrücklich, *ausschließlich* einen einfachen Texteditor zu verwenden, solange Sie dieses Buch noch nicht durchhaben (und vor allem, solange Sie Kapitel 16 noch nicht durchhaben). Eine IDE kann Ihnen einige der Details ersparen, die wirklich wichtig sind; daher ist es viel besser, wenn Sie das Ganze zunächst auf der Kommandozeile lernen. Erst wenn Sie das, was da passiert, wirklich verstehen, sollten Sie auf ein Tool umsteigen, das einen Teil des Prozesses automatisiert.

INSTALLATION VON JAVA

- Sie benötigen ein **Java 2 Standard Edition SDK** (Software Development Kit) der Version **1.5** oder höher, falls Sie es nicht bereits haben. Für Linux, Windows und Solaris erhalten sie es kostenlos von java.sun.com (der Website von Sun für Java-Entwickler). Normalerweise braucht man nicht mehr als zwei Klicks, um von der Hauptseite auf die J2SE-Download-Seite zu gelangen. Holen Sie sich die neueste verfügbare *Nicht-Beta*-Version. Das SKD enthält alles, was Sie brauchen, um Java zu kompilieren und auszuführen.
  Wenn Sie mit Mac OS X 10.4 arbeiten, ist das Java SDK bereits installiert. Es ist Teil von OS X, daher brauchen Sie *nichts weiter* zu tun. Verwenden Sie eine ältere Version von OS X, haben Sie eine frühere Java-Version, die bei 95% des Codes in diesem Buch funktionieren wird.
  Hinweis: Dieses Buch basiert auf Java 1.5, aber aus merkwürdig unklaren Marketing-Gründen hat Sun dieses kurz vor der Veröffentlichung in Java 5 umbenannt, während als Versionsnummer für das SDK »1.5« beibehalten wurde. Wenn Sie also Java 1.5 lesen oder Java 5 oder Java 5.0 oder auch »Tiger« (der ursprüngliche Codename von Version 5) – *es bedeutet alles das Gleiche.* Java 3.0 und 4.0 hat es nie gegeben, sondern es gab einen Sprung von Version 1.4 zu 5.0, aber an manchen Stellen heißt es eben immer noch 1.5 statt 5. Fragen Sie nicht, warum. (Oh, und damit es noch lustiger wird: Sowohl Java 5 als auch Mac OS X 10.4 hatten als Codenamen »Tiger«! Und weil OS X 10.4 die Version von Mac OS ist, die man für die Ausführung von Java 5 benötigt, hört man die Leute manchmal auch von »Tiger auf Tiger« reden. Das bedeutet einfach: Java 5 auf OS X 10.4.)
- *Nicht* im SDK enthalten ist die **API-Dokumentation** – und die brauchen Sie! Gehen Sie noch einmal auf java.sun.com und holen Sie sich dort die J2SE-API-Dokumentation. Sie können auch online auf die API-Docs zugreifen, ohne sie herunterzuladen, aber das ist mühsam. Glauben Sie uns, es ist den Download wert.
- Sie brauchen einen **Texteditor.** Geeignet sind praktisch alle Texteditoren (vi, emacs, pico), auch die GUI-Editoren, die bei den meisten Betriebssystemen dabei sind. Notepad, Wordpad, Textedit usw. sind allesamt verwendbar, solange Sie sicherstellen, dass am Ende Ihres Quellcode-Dateinamens nicht ».txt« angehängt wird.
- Nach dem Herunterladen und Entpacken/Extrahieren/was auch immer (das ist versions- und betriebssystemabhängig) müssen Sie einen Eintrag zu Ihrer **PATH**-Umgebungsvariablen hinzufügen, der auf das Verzeichnis /bin innerhalb des Haupt-Java-Verzeichnisses verweist. Wenn beispielsweise das J2SDK ein Verzeichnis auf Ihrer Festplatte anlegt, das »j2sdk1.5.0« heißt, sehen Sie in diesem Verzeichnis nach; Sie finden dort das Verzeichnis bin, in dem die ausführbaren Java-Binärdateien (die Werkzeuge) liegen. Für dieses bin -Verzeichnis brauchen Sie einen PATH-Eintrag; dadurch weiß Ihr Terminal, wo es den Compiler *javac* findet, wenn Sie auf der Kommandozeile
  ```
  %javac
  ```
  eingeben.
  Hinweis: Wenn Sie Probleme bei der Installation haben, empfehlen wir Ihnen, das Forum für Java-Anfänger auf javaranch.com aufzusuchen! Aber das sollten Sie auf jeden Fall tun – egal ob Sie Probleme haben oder nicht.

Hinweis: Ein großer Teil des Codes aus diesem Buch kann von www.oreilly.de heruntergeladen werden.

Was Sie sonst noch wissen sollten:

Dies ist ein Lernerlebnis, kein Nachschlagewerk. Wir haben bewusst alles herausgestrichen, was an irgendeiner Stelle des Buchs hinderlich für den *Lernprozess* sein könnte. Und wenn Sie das Buch das erste Mal durcharbeiten, müssen Sie am Anfang anfangen, denn das Buch macht bestimmte Annahmen darüber, was Sie schon gesehen und gelernt haben.

Wir verwenden einfache, UML-ähnliche Diagramme.

Wenn wir *reines* UML verwendet hätten, würden Sie zwar etwas sehen, das wie Java *aussieht,* aber mit einer Syntax, die schlicht und einfach *falsch* ist. Deshalb haben wir eine vereinfachte UML-Version benutzt, die nicht im Widerspruch zur Java-Syntax steht. Falls Sie UML noch nicht kennen, müssen Sie sich hier nicht damit abplagen, Java *und* UML gleichzeitig zu lernen.

Wir verwenden ein einfacheres, modifiziertes Pseudo-UML.

Damit Sie es einfacher haben, schieben wir das Organisieren und Verpacken des Codes auf bis zum Ende des Buchs.

In diesem Buch können Sie sich in aller Ruhe dem Lernen von Java widmen, ohne mit bestimmten Organisations- oder Verwaltungsdetails der Entwicklung von Java-Programmen belastet zu werden. Später – in der »echten Welt« – müssen Sie diese Details kennen und damit arbeiten, und wir werden sie daher eingehend behandeln. Aber wir schieben sie bis zum Ende des Buchs auf (Kapitel 17). So können Sie sich ganz entspannt an Java gewöhnen.

Die Spitzen-Sie-Ihren-Bleistift-Aufgaben sollten Sie ALLE versuchen zu lösen.

Die Übungen am Kapitelende sind notwendig; die Rätsel sind optional. Die Antworten für beides stehen jeweils am Ende des Kapitels.

Eins müssen Sie über die »Puzzles« und »Rätsel« wissen – *es sind wirklich Rätsel*. So wie Knobeleien, Denksportaufgaben, Kreuzworträtsel usw. Die *Übungen* sind dazu gedacht, das Gelernte einzuüben, und sollten daher auf keinen Fall ausgelassen werden. Das gilt nicht für die Rätsel; manche davon stellen sogar eine ziemliche Herausforderung als Rätsel dar. Sie sind etwas für *Rätselfreunde* – und ob Sie ein Rätselfreund sind oder nicht, wissen Sie sicher selbst. Wenn Sie es nicht genau wissen, schlagen wir Ihnen vor, ein paar davon auszuprobieren, aber lassen Sie sich auf keinen Fall entmutigen, wenn Sie ein Rätsel oder Puzzle *nicht* lösen können oder einfach nicht genug Zeit dafür aufwenden können.

Aufgaben, die mit dem Trainingslogo (dem Laufschuh) markiert sind, sind Pflicht! Überspringen Sie sie nicht, wenn Sie es ernst meinen mit dem Java-Lernen.

Zu den »Spitzen-Sie-Ihren-Bleistift«-Übungen gibt es keine Lösungen.

Zumindest sind diese nicht im Buch abgedruckt. Für manche gibt es keine eindeutige Antwort, und bei anderen sollen *Sie* im weiteren Verlauf Ihrer Lernerfahrung selbst entscheiden, ob *Ihre* Antworten richtig waren. (Einige Antwort*vorschläge* von uns finden Sie – in englischer Sprache auf wickedlysmart.com.)

Wenn Sie das Puzzle-Logo sehen, geht es um etwas Freiwilliges, und wenn Sie keinen Spaß an Knobeleien oder Kreuzworträtseln haben, werden Ihnen auch diese nicht gefallen.

Die Codebeispiele sind so kurz wie möglich.

Es ist frustrierend, sich durch 200 Zeilen Code zu wühlen, um die beiden Zeilen zu finden, die man zum Verständnis benötigt. Bei den meisten Beispielen in diesem Buch wird vom Kontext so wenig abgedruckt wie möglich – so wird der Teil, den Sie gerade lernen wollen, klar und einfach. Erwarten Sie beim Code nicht unbedingt, dass er stabil läuft oder vollständig ist. Das bleibt als Aufgabe für *Sie*, sobald Sie mit dem Buch durch sind. Die Beispiele im Buch wurden speziell zum Zweck des *Lernens* geschrieben und sind nicht immer voll funktionsfähig.

Fachgutachter

»Der Dank gebührt allen, aber Fehler liegen in der alleinigen Verantwortung des Autors ...« Glaubt das eigentlich irgend jemand? Sehen Sie die beiden Menschen auf dieser Seite? Falls Sie fachliche Fehler finden, dürfte es *deren* Schuld sein. :)

Jess arbeitet bei Hewlett-Packard im Self-Healing Services Team. Sie machte ihren Bachelor-Abschluss in Technischer Informatik an der Villanova University, hat SCJP 1.4- und SCWCD-Zertifizierungen und steht kurz vor ihrem Master in Technischer Informatik an der Drexel University (puh!).

Wenn Jess nicht arbeitet, studiert oder in ihrem MINI Cooper S unterwegs ist, kämpft sie vielleicht gerade mit ihrer Katze um das Garn, mit dem sie ihr neuestes Strick- oder Häkelprojekt fertig stellen möchte (will jemand einen Hut?). Ursprünglich stammt sie aus Salt Lake City, Utah (nein, sie ist keine Mormonin ... ja, auch *Sie* wollten das fragen ...) und lebt zurzeit mit ihrem Mann, Mendra, und zwei Katzen, Chai und Sake, in der Nähe von Philadelphia.

Begegnen können Sie ihr beim Moderieren von Fachforen auf *javaranch.com*.

Valentin machte seinen Master-Abschluss in Informatik an der Schweizer École Polytechnique Fédérale de Lausanne (EPFL). Er hat als Entwickler bei SRI International (Menlo Park, Kalifornien) und als leitender Entwickler im Software Engineering Laboratory der EPFL gearbeitet.

Valentin ist Mitbegründer und Technischer Direktor von Condris Technologies, einer Firma, die sich auf die Entwicklung von Softwarearchitekturlösungen spezialisiert hat.

Zu seinen Forschungs- und Entwicklungsinteressen gehören aspektorientierte Programmierung, Entwurfs- und Architekturmuster, Webdienste und Softwarearchitektur. Valentin kümmert sich um seine Frau, um die Gartenarbeit, liest, treibt Sport – und neben all dem moderiert er auch noch die SCBCD- und SCDJWS-Foren auf javaranch.com. Er hat SCJP-, SCJD-, SCBCD-, SCWCD- und SCDJWS-Zertifikate. Zudem hatte er auch die Gelegenheit, als Koautor für den Whizlabs SCBCD Exam Simulator zu fungieren.

(Es hat uns einen Schock versetzt, ihn mit *Krawatte* zu sehen.)

Dank schulden wir

~~Schuld haben~~ außerdem:

O'Reilly:

Unser besonderer Dank geht an **Mike Loukides** bei O'Reilly, der das Risiko dieses Buchs eingegangen ist und uns geholfen hat, aus dem »Head First«-Konzept ein Buch (bzw. eine Buch*reihe)* zu entwickeln. Wenn diese zweite Auflage in den Druck geht, gibt es bereits fünf Bücher aus der Reihe »Head First«, und Mike hat uns auf dem ganzen Weg begleitet. Wir danken auch **Tim O'Reilly** für seine Bereitschaft, sich auf etwas *völlig* Neues und Andersartiges einzulassen. Und der klugen **Kyle Hart** dafür, dass sie herausgefunden hat, wie »Head First« in diese Welt passt, und für die Markteinführung der Reihe. Und schließlich **Edie Friedman** für das »Kopf-betonende« Cover der Reihe.

Unseren tapferen Beta-Testern und Review-Team-Mitgliedern:

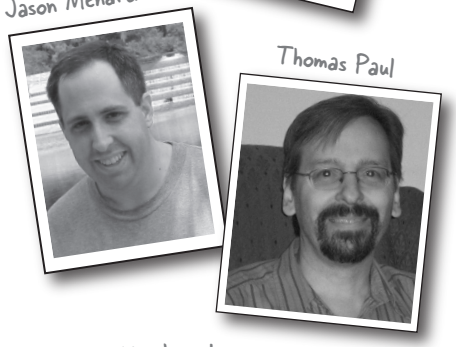
Ganz besonders danken wir dem Leiter unseres Javaranch-Review-Teams, **Johannes de Jong.** Du bist jetzt zum fünften Mal bei einem unserer »Head First«-Bücher dabei, und wir freuen uns mächtig, dass du trotzdem noch mit uns sprichst. **Jeff Cumps** ist zum dritten Mal dabei und findet unerbittlich die Stellen, an denen wir deutlicher oder exakter sein müssen.

Corey McGlone, du bist Spitze. Und wir finden, du gibst auf javaranch.com die besten Erklärungen. Wahrscheinlich hast du gemerkt, dass wir eine oder zwei davon geklaut haben. **Jason Menard** hat uns mehr als einmal – und nicht nur bei Kleinigkeiten – in fachlicher Hinsicht gerettet, und **Thomas Paul** hat uns wie immer fachliches Feedback geliefert und verzwickte Java-Probleme gefunden, die wir anderen übersehen hatten. **Jane Griscti,** Java-Programmiererin mit Brief und Siegel (die zudem das eine oder andere übers Schreiben weiß), war uns – gemeinsam mit dem altgedienten Javaranch-er **Barry Gaunt** – eine große Hilfe bei der Neuauflage.

Marilyn de Queiroz hat uns bei *beiden* Auflagen des Buchs großartig unterstützt. Bei der ersten Auflage waren uns außerdem **Chris Jones, John Nyquist, James Cubeta, Terri Cubeta und Ira Becker** eine riesige Hilfe.

Besonderer Dank geht an einige der Head-First-Leute, die uns von Anfang an hilfreich zur Seite gestanden haben: **Angelo Celeste, Mikalai Zaikin** und **Thomas Duff** (*twduff.com*). Und danke an unseren wunderbaren Agenten, **David Rogelberg** von StudioB (aber mal ernsthaft, was ist mit den *Filmrechten*?).

Ein paar der Java-Experten aus unserem Review-Team ...



Gerade als Sie dachten, es kämen keine weiteren Danksagungen* ...

Noch mehr Java-Experten, die bei der ersten Auflage geholfen haben (in pseudozufälliger Reihenfolge):

Emiko Hori, Michael Taupitz, Mike Gallihugh, Manish Hatwalne, James Chegwidden, Shweta Mathnur, Mohamed Mazahim, John Paverd, Joseph Bih, Skulrat Patanavanich, Sunil Palicha, Suddhasatwa Ghosh, Ramki Srinivasan, Alfred Raouf, Angelo Celeste, Mikalai Zaikin, John Zoetebier, Jim Pleger, Barry Gaunt und Mark Dielen.

Das Puzzle-Team der ersten Auflage:

Dirk Schreckmann, Mary »Java-Kreuzworträtsel-Champion« Leners, Rodney J. Woodruff, Gavin Bong und Jason Menard. Javaranch hat wirklich Glück, dass ihr alle dort aushelft.

Weitere Mitverschwörer, denen wir danken möchten, sind:

Paul Wheaton, der Ober-Cowboy von javaranch.com, der Tausende von Java-Anfängern unterstützt hat.

Solveig Haugland, J2EE-Meisterin und Autorin von »Dating Design Patterns«.

Die Autoren **Dori Smith** und **Tom Negrino** (backupbrain.com), die uns durch die Welt der technischen Fachbücher gelotst haben.

Unsere Head-First-Komplizen, **Eric Freeman** und **Beth Freeman** (die Autoren von »Head First Design Patterns« – in deutscher Übersetzung: »Entwurfsmuster von Kopf bis Fuß«), die uns Bawls™ zu trinken gaben und uns dadurch den nötigen Koffeinschub verpassten, damit wir das hier pünktlich fertig stellen konnten.

Sherry Dorris – für die wirklich wichtigen Dinge.

Die mutigen »Early Adopters« dieser Reihe:

Joe Litton, Ross P. Goldberg, Dominic Da Silva, honestpuck, Danny Bromberg, Stephen Lepp, Elton Hughes, Eric Christensen, Vulinh Nguyen, Mark Rau, Abdulhaf, Nathan Oliphant, Michael Bradly, Alex Darrow, Michael Fischer, Sarah Nottingham, Tim Allen, Bob Thomas und Mike Bibby (der Erste).

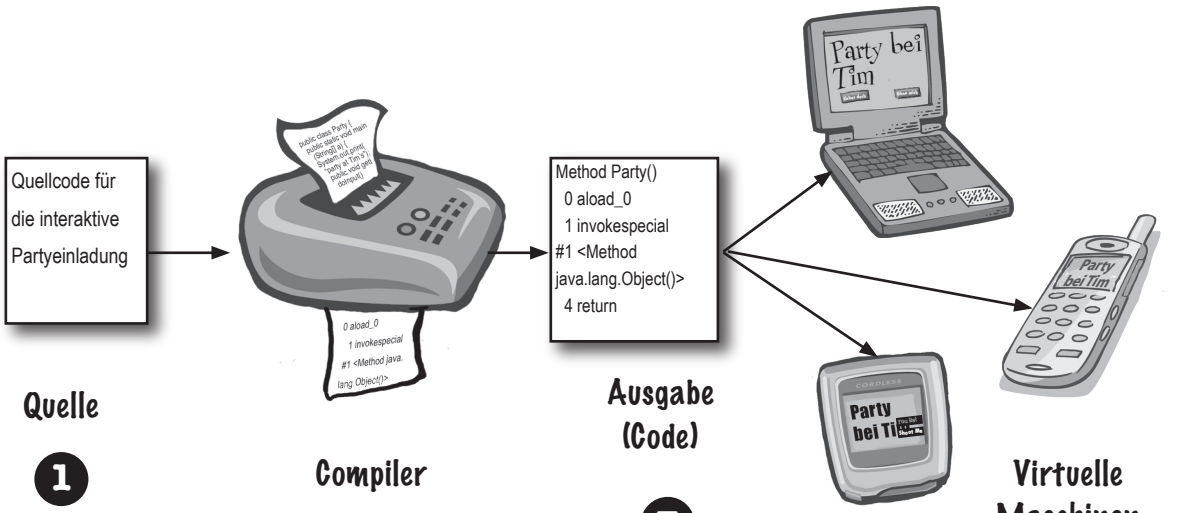
* Die große Zahl der Danksagungen hat folgenden Grund: Wir testen die Theorie, dass jeder, der in einem Buch bei den Danksagungen erwähnt wird, mindestens ein Exemplar davon kauft, wahrscheinlich sogar mehrere, wegen der Verwandtschaft und so. Wenn Sie bei den Danksagungen in unserem nächsten Buch erwähnt werden möchten und eine große Familie haben, schreiben Sie uns!

Die Oberfläche durchbrechen

Java führt Sie an neue Orte. Seit seinem bescheidenen Schritt an die Öffentlichkeit mit der (kümmerlichen) Version 1.02 hat Java mit seiner freundlichen Syntax, seinen objektorientierten Features, der Speicherverwaltung und natürlich mit dem Versprechen der Portabilität Programmierer verführt. Die Verlockung des **Write-once/Run-anywhere** ist einfach zu groß. Die hingebungsvolle Gefolgschaft vermehrte sich explosionsartig, während Programmierer gegen Bugs, Beschränkungen und den Umstand kämpften, dass es schneckenlangsam war. Aber das ist Jahrhunderte her. Wenn Sie gerade erst mit Java beginnen, **haben Sie Glück**. Einige von uns mussten sich zehn Kilometer durch den Schnee kämpfen (barfuß), in jede Richtung bergauf, um selbst das trivialste Applet ans Laufen zu bringen. Aber *Sie*, *Sie* können sich von dem **geschmeidigeren, schnelleren, viel, viel mächtigeren** Java von heute tragen lassen.

Wie Java funktioniert

Das Ziel ist es, eine Anwendung zu schreiben (in diesem Fall eine interaktive Partyeinladung), die auf jedem beliebigen, von Ihren Freunden verwendeten Gerät funktionieren soll.



Quelle

1

Erstellen Sie ein Quelldokument. Verwenden Sie ein etabliertes Protokoll (in diesem Fall die Sprache Java).

Compiler

2

Lassen Sie Ihr Dokument durch einen Quellcode-Compiler laufen. Der Compiler prüft, ob Fehler vorhanden sind, und lässt Sie erst kompilieren, wenn er sicher ist, dass alles richtig läuft.

Ausgabe (Code)

3

Der Compiler erstellt ein neues Dokument, das in Java-***Bytecode*** kodiert ist. Jedes Gerät, das Java ausführen kann, kann diese Datei interpretieren bzw. in etwas übersetzen, das es ausführen kann. Der kompilierte Bytecode ist plattformunabhängig.

Virtuelle Maschinen

4

Ihre Freunde haben keine physische Java-Maschine, aber alle haben eine ***virtuelle*** (in Software implementierte) Java-Maschine, die innerhalb ihrer elektronischen Spielzeuge läuft. Die virtuelle Maschine liest den Bytecode und führt ihn aus.