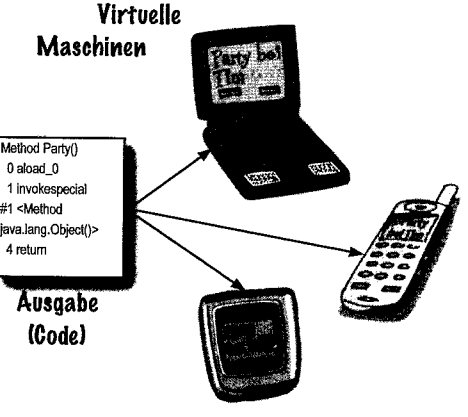
Der Inhalt (jetzt ausführlich)

E 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?

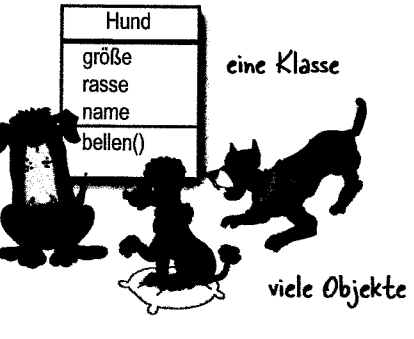
1 Die Oberfläche durchbrechen

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!



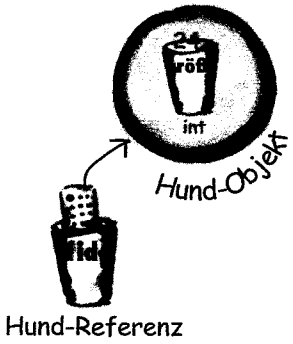
2 Ein Ausflug nach Objekthausen

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

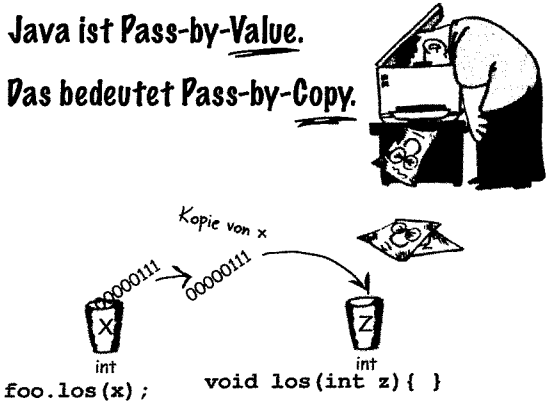
Variablen gibt es in zwei Geschmacksrichtungen: elementare Typen und Referenztypen. Im Leben muss es doch mehr geben als Integer, Strings und Arrays. Was ist, wenn Sie ein HaustierBesitzer-Objekt mit einer Hund-Instanzvariablen haben? Oder ein Auto mit einem Motor? In diesem Kapitel werden wir die Mysterien 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.



---

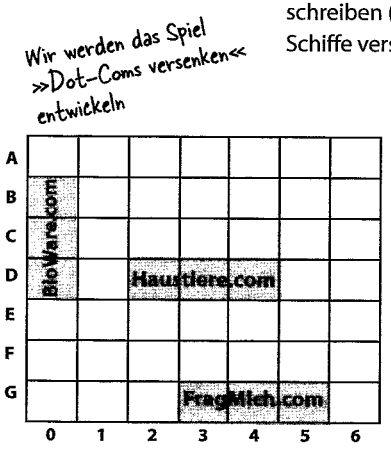
4 Wie sich Objekte verhalten

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).



6 Die Java-Bibliothek verwenden

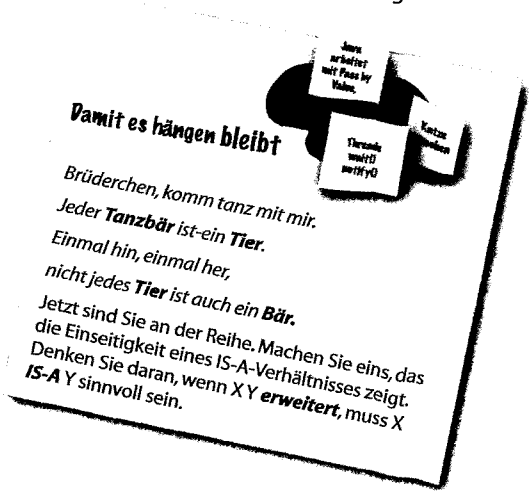
Java wird mit Hunderten vorgefertigter Klassen ausgeliefert. Sie müssen das Rad nicht jedes Mal neu erfinden, wenn Sie wissen, wie Sie das, was Sie brauchen, in der Java-Bibliothek finden, die als die **Java-API** bezeichnet wird. *Schließlich haben Sie etwas Besseres zu tun.* Wenn Sie Code schreiben, 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

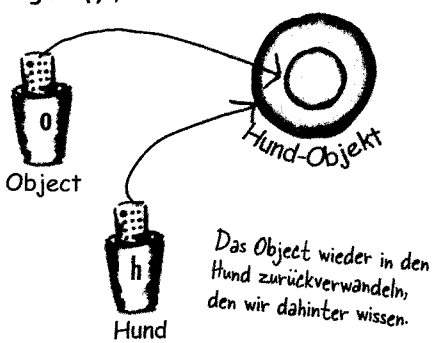
Planen Sie Ihre Programme mit Blick auf die Zukunft Was wäre, wenn Sie Code schreiben könnten, den ein *anderer* **problemlos** erweitern kann? Und was wäre, wenn Ihr Code flexibel genug für diese verfluchten Spezifikationsänderungen in letzter Minute wäre? Wenn Sie auf den Polymorphie-Zug springen, 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

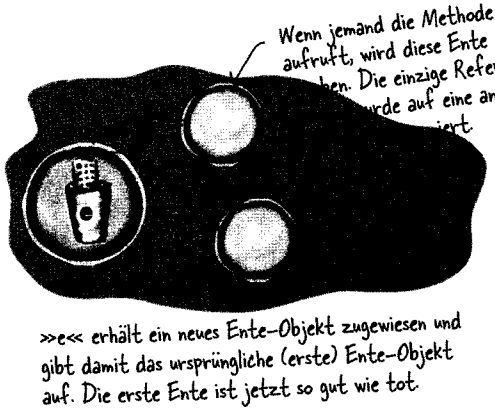
Vererbung ist nur der Anfang. Um Polymorphie richtig zu nutzen, brauchen wir Schnittstellen. Wir müssen über einfache Vererbung hinausgehen und eine Stufe der Flexibilität erreichen, die man nur erhält, wenn man Schnittstellendefinitionen als Ausgangsbasis für den Entwurf und die Programmierung nimmt. 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...

```
Object o = al.get(index);
Hund h = (Hund) o;
h.bewegen();
```



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.



>>e<< erhält ein neues Ente-Objekt zugewiesen und gibt damit das ursprüngliche (erste) Ente-Objekt auf. Die erste Ente ist jetzt so gut wie tot.

10 Zahlen, bitte!

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

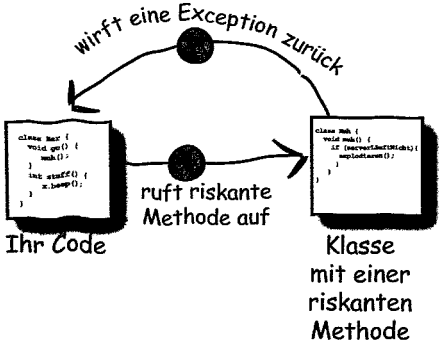
Rechnen Sie's aus. In der Java-API gibt es eine Menge praktischer Methoden für Absolutbeträge, zum Runden, zur Extremwertbestimmung etc. Aber wie sieht es mit der Formatierung aus? Vielleicht möchten Sie, dass Ihre Zahlen mit exakt zwei Nachkommastellen ausgegeben oder dass große Zahlen mit Trennzeichen 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.

Instanzvariablen: eine pro Instanz

statische Variablen: eine pro Klasse

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

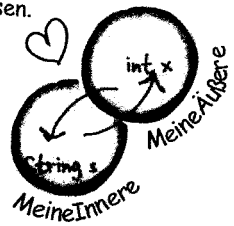
Sehen Sie den Tatsachen ins Auge: Um GUIs (grafische Benutzeroberflächen) kommen Sie nicht herum. Selbst wenn Sie davon ausgehen, dass Sie den Rest Ihres Lebens serverseitigen Code schreiben, müssen Sie früher oder später Tools entwickeln, und dafür hätten Sie dann ganz sicher gern eine grafische Schnittstelle. 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() {
    }
  }

}
```

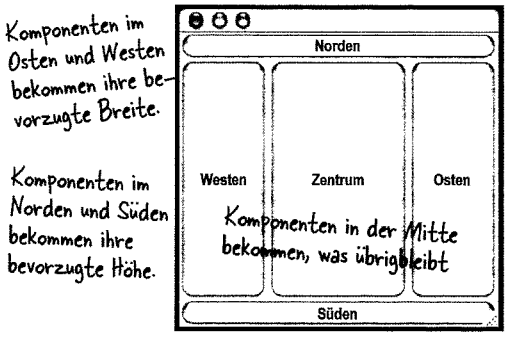
Innere Klasse ist vollständig von äußerer Klasse umschlossen.



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

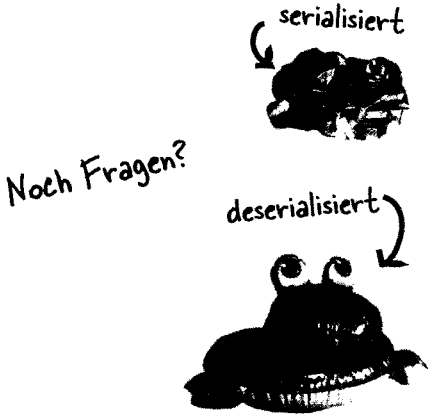
13 Mehr Schwung mit Swing

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.



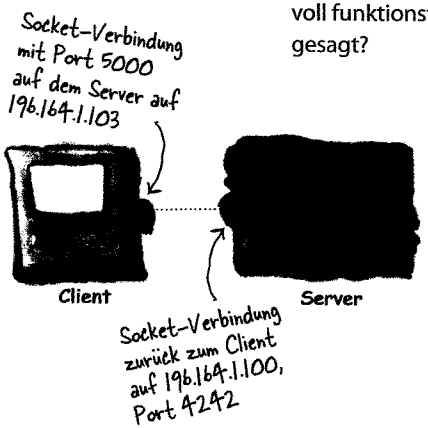
14 Speicherung von Objekten

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.



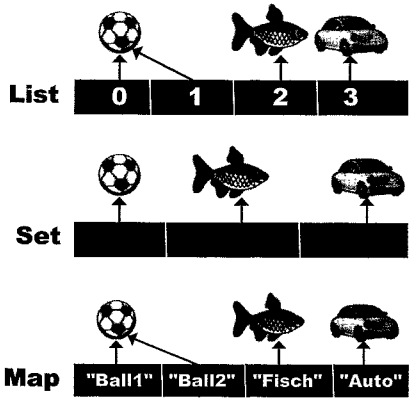
15 Eine Verbindung herstellen

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?



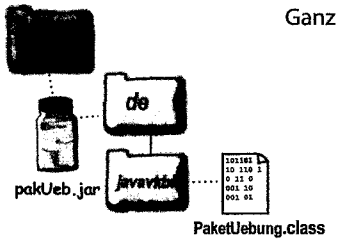
16 Datenstrukturen

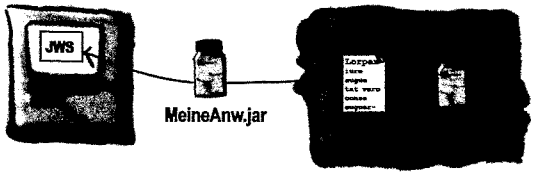
Sortieren ist in Java ein Klacks. Sie haben alle Werkzeuge zur Verfügung, um Daten zu sammeln und damit zu arbeiten, ohne Ihre eigenen Sortieralgorithmen schreiben zu müssen. Das Collections-Framework von Java enthält eine Datenstruktur für praktisch alles, was Sie jemals tun müssen. Sie wollen eine 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

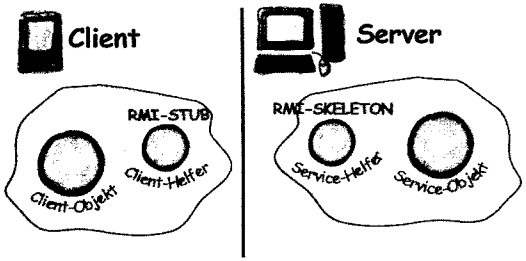
Es ist Zeit loszulassen. Sie haben Ihren Code geschrieben. Sie haben ihn getestet. Sie haben ihn verbessert. Sie haben allen Bekannten erzählt, es wäre schön, wenn Sie nie wieder eine einzige Zeile Code sehen würden. Und doch haben Sie zu guter Letzt ein Kunstwerk geschaffen. Tatsächlich, das Ding läuft! 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.





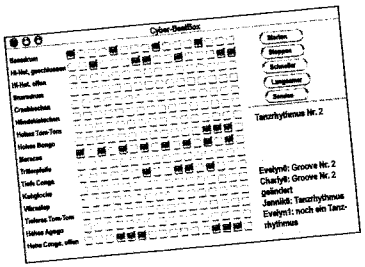
18 Verteilte Anwendungen

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