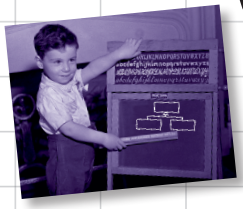
Ein Buch zum Mitmachen und Verstehen

Objektorientierte Analyse & Design

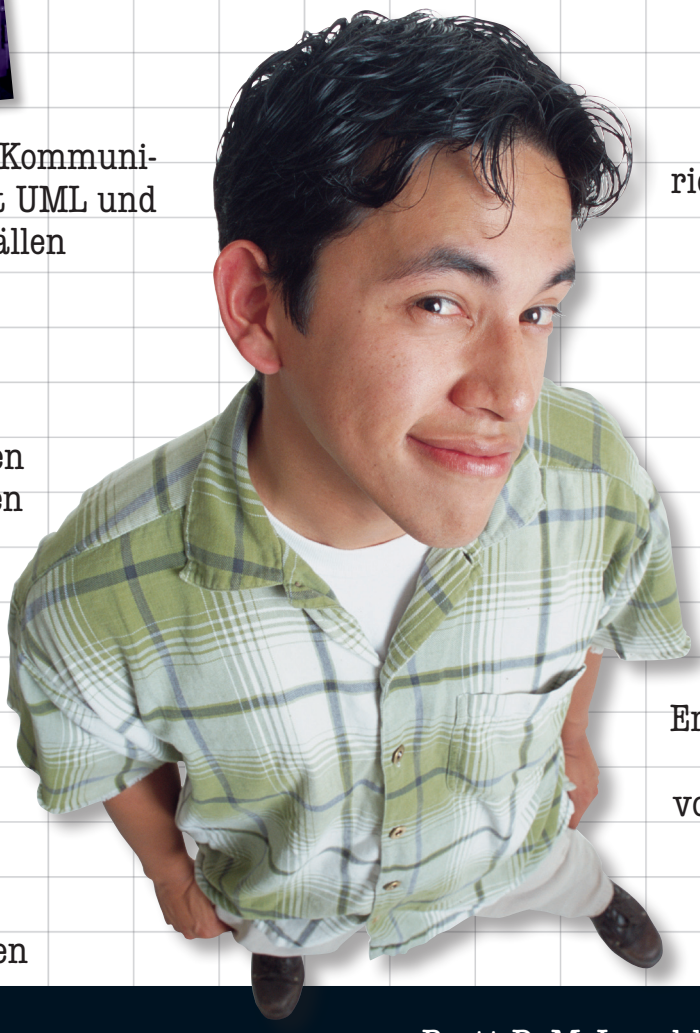
von Kopf bis Fuß



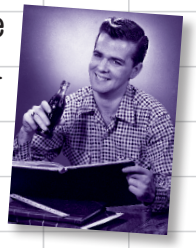
Verbessern Sie Ihre Kommunikationsfähigkeit mit UML und Anwendungsfällen

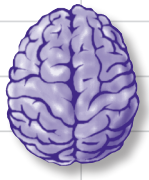
Trainieren Sie Ihren Kopf mit Dutzenden von OO-Übungen

Vermeiden Sie unzufriedene Kunden



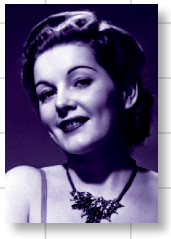
Machen Sie aus Ihren Anforderungen und Entwürfen richtige Software





Laden Sie sich die entscheidenden OO-Entwurfsprinzipien direkt ins Hirn

Erfahren Sie, wie Tosca mit Hilfe von Abstraktion, Aggregation und Delegation zu strahlen begann



O'REILLY®

Brett D. McLaughlin, Gary Pollice & David West

Deutsche Übersetzung von 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 2006 unter dem Titel
Head First Object-Oriented Analysis & Design 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
Lektorat: Christine Haite, Köln
Fachgutachten: Karsten Loesing, Bamberg & Anke Werner, Dortmund
Korrektorat: Sibylle Feldmann, Düsseldorf
Satz: Conrad Neumann, München
Umschlaggestaltung: Mike Kohnke & Edie Freedman, Boston
Produktion: Andrea Miß, Köln
Belichtung, Druck und buchbinderische Verarbeitung: Media-Print, Paderborn

ISBN-10 3-89721-495-4
ISBN-13 978-3-89721-495-8

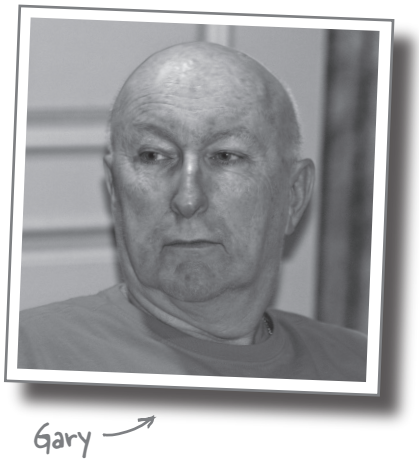
Dieses Buch ist auf 100% chlorfrei gebleichtem Papier gedruckt.

All den brillanten Menschen, denen die unterschiedlichen Möglichkeiten eingefallen sind, Anforderungen zu sammeln, Software zu analysieren und Code zu entwerfen ...

... danke, dass ihr etwas ausgedacht habt, das gut genug ist, um gute Software zu entwickeln, und trotzdem so schwer, dass wir dieses Buch brauchten, um es zu erklären.

Brett McLaughlin ist ein Gitarrenspieler, der immer noch mit der Einsicht kämpft, dass man seine Rechnungen nicht zahlen kann, wenn man akustischen Fingerstyle-Blues und -Jazz spielt. Aber vor nicht allzu langer Zeit hat er zu seiner Freude entdeckt, dass das Schreiben von Büchern, die anderen Menschen helfen, bessere Programmierer zu werden, dazu beiträgt, Rechnungen zu bezahlen. Darüber ist er sehr glücklich, genau wie seine Frau Leigh und seine Jungs Dean und Robbie.

Bevor Brett ins Von Kopf bis Fuß-Land gewandert ist, hat er für Nextel Communications und Allegiance Telecom Java Enterprise-Anwendungen aufgebaut. Als ihm das zu banal wurde, hat Brett sich Application-Servern zugewandt und an den Interna der Lutris Enhydra Servlet-Engine und EJB-Containers gearbeitet. Auf diesem Weg wurde Brett süchtig nach Open Source-Software und war an der Gründung mehrerer cooler Programmierwerkzeuge wie Jakarta Turbine und JDOM beteiligt. Schreiben Sie ihm an *brett@oreilly.com*.



Gary Pollice bezeichnet sich selbst als Griesgram (das ist ein mürrischer, übellauniger, in der Regel alter Mann), der mehr als 35 Jahre in der Industrie verbracht hat, um herauszufinden, was er werden will, wenn er erwachsen geworden ist. Auch wenn er noch nicht erwachsen geworden ist, hat er 2003 den Sprung in die heiligen Hallen der Akademie gemacht, wo er die Köpfe der nächsten Generation von Softwareentwicklern mit radikalen Ideen wie diesen verdirbt: »Entwickeln Sie Software für Ihren Kunden. Lernen Sie, als Teil eines Teams zu arbeiten. Design und Codequalität und Eleganz und Richtigkeit zählen. Es ist okay, wenn man ein Nerd ist, solange man ein guter Nerd ist.«

Gary ist Professor of Practice (was bedeutet, dass er einen richtigen Job hatte, bevor er Professor wurde) am Worcester Polytechnic Institute. Er lebt in Zentral-Massachusetts mit seiner Frau Vikki und ihren beiden Hunden Aloysius und Ignatius. Sie können seine WPI-Homepage unter **`http://web.cs.wpi.edu/~gpollice/`** besuchen. Sie können ihm gern Anmerkungen zukommen lassen oder sich über das Buch beschweren – oder es loben.

Dave West würde sich selbst als Sheik-Geek bezeichnen. Unglücklicherweise würde ihn sonst keiner so beschreiben. Sie würden sagen, dass er ein professioneller Engländer sei, der am liebsten mit der Leidenschaft und Energie eines Laienpredigers über bewährte Verfahren zur Softwareentwicklung spricht. Vor Kurzem hat Dave zu Ivar Jacobson Consulting gewechselt, wo er für Nord-, Süd- und Mittelamerika verantwortlich ist und seine Leidenschaften kombinieren kann, über Softwareentwicklung zu reden, über Rugby und Football zu diskutieren und zu behaupten, dass Kricket aufregender sei als Baseball.

Bevor er für Ivar Jacobson Consulting die Amerikas übernommen hat, arbeitete Dave eine Reihe von Jahren bei Rational Software (jetzt ein Teil von IBM). Bei Rational und IBM nahm Dave viele Positionen ein, unter anderem als Product Manager für RUP, wo er die Konzepte von Prozess-Plugins und -Agilität in RUP eingeführte. Sie können ihn unter *dwest@ivarjacobson.com* kontaktieren.

Über den Übersetzer dieses Buchs (der bei aller Berühmtheit nicht auf der Straße erkannt werden will)

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.

Zur deutschen Übersetzung

Das leidige Thema der deutschen Umlaute im Code haben wir natürlich auch gründlich diskutiert. Der besseren Lesbarkeit wegen 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.

Weitere verwandte Bücher von O'Reilly

Die Kunst des IT-Projektmanagements

UML 2.0 in a Nutshell

Practical Development Environments

Process Improvement Essentials

Prefactoring

Ajax Design Patterns

Learning UML

Applied Software Project Management

Unit Test Frameworks

Weitere Bücher in O'Reillys *Von Kopf bis Fuß*-Reihe

Entwurfsmuster von Kopf bis Fuß

Java von Kopf bis Fuß

Ajax von Kopf bis Fuß

HTML mit CSS & XHTML von Kopf bis Fuß

OOA&D von Kopf bis Fuß

Head First Servlets and JSP

Head First EJB

Head First PMP (2007)

Head First Algebra (2007)

Head First Software Development (2007)

Der Inhalt (in der Übersicht)

Der Inhalt (jetzt ausführlich)

Einführung

Ihr Gehirn und OOA&D. 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 objektorientierte Analyse und Design zu wissen?

Sauber entworfene Anwendungen rocken

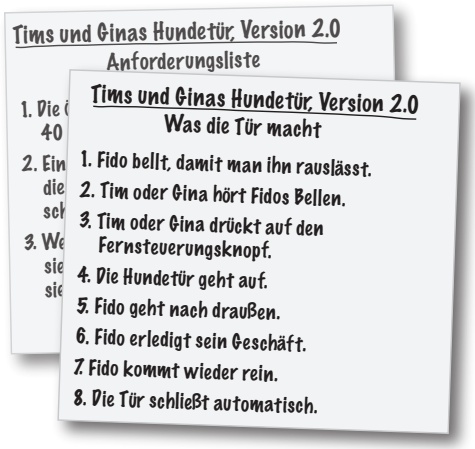
1 Hier fängt gute Software an

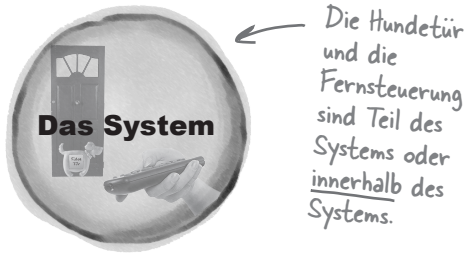
Wie schreibt man den nun *wirklich* gute Software? Es ist immer schwierig, den **Anfangspunkt zu finden**. Macht die Anwendung tatsächlich das, **was sie tun soll**? Und was ist mit Dingen wie doppeltem Code – das kann doch nicht gut sein, oder? In der Regel ist es ziemlich schwer, herauszufinden, **woran man als Erstes arbeiten soll**, und dabei gleichzeitig darauf zu achten, dass bei diesem Vorgang nichts anderes ruiniert wird. Aber Kopfzerbrechen ist unnötig. Wenn Sie mit diesem Kapitel durch sind, **werden Sie wissen, wie man gute Software schreibt**, und sind auf dem richtigen Weg, Ihre Art, Anwendungen zu entwickeln, dauerhaft zu verbessern. Und schließlich werden Sie verstehen, warum **OOA&D** ein Akronym ist, das man wirklich kennen sollte.

Anforderungen sammeln

2 Gebt Ihnen, was sie wollen

Jeder will zufriedene Kunden. Sie wissen bereits, der erste Schritt beim Schreiben guter Software ist sicherzustellen, dass sie macht, was der Kunde will. Aber wie findet man heraus, **was der Kunde wirklich will**? Und wie prüft man, ob der Kunde überhaupt *weiß*, was er wirklich will? Das ist der Punkt, an dem **gute Anforderungen** ins Spiel kommen, und in diesem Kapitel werden Sie lernen, wie Sie **Ihre Kunden zufriedenstellen**, indem Sie darauf achten, dass das, was Sie ausliefern, genau das ist, wonach der Kunde gefragt hat. Haben Sie die Arbeit abgeschlossen, sind all Ihre Projekte mit einer »Zufriedenheitsgarantie« ausgestattet. Und Sie sind auf dem besten Weg dahin, gute Software zu schreiben. Jedes Mal!



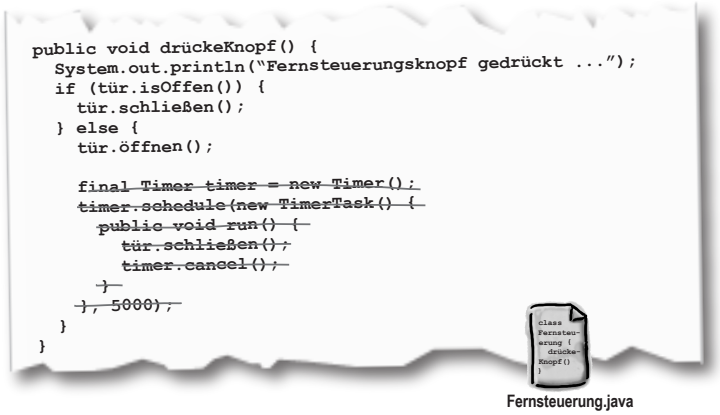


Anforderungen ändern sich

3 Ich liebe dich, du bist perfekt ... jetzt ändere dich endlich

Sie meinen, Sie haben genau das, was der Kunde wollte?

Nicht so schnell ... Sie haben also mit Ihrem Kunden gesprochen, Anforderungen gesammelt, Ihre Anwendungsfälle geschrieben und eine Killer-Anwendung abgeliefert. Zeit für ein nettes kühles Bier, stimmt's? Stimmt ... bis sich Ihr Kunde überlegt, dass er eigentlich etwas **anderes will, als er Ihnen gesagt hat**. Was Sie gemacht haben, ist wunderbar, glauben Sie es mir, aber es ist **nicht mehr wirklich gut genug**. Im wahren Leben **ändern sich Anforderungen permanent**, und Sie müssen mit diesen Änderungen Schritt halten, um Ihre Kunden zufriedenzustellen.



Analyse

4 Ihre Software ins richtige Leben führen

Es ist Zeit, sich an richtige Anwendungen zu wagen.

Ihre Anwendung muss nicht nur auf Ihrer genau abgestimmten und perfekt eingerichteten Entwicklungsmaschine funktionieren. Ihre Anwendungen müssen funktionieren, wenn **echte Menschen sie verwenden**. In diesem Kapitel geht es darum, wie man sicherstellt, dass Software im **echten Leben** funktioniert. Sie werden lernen, wie eine **Textanalyse** den Anwendungsfall, an dem Sie gearbeitet haben, in Klassen und Methoden verwandelt, bei denen Sie sich darauf verlassen können, dass sie tun, was Ihre Kunden wollen. Und wenn Sie fertig sind, können auch Sie sagen: »Ich hab's geschaft! Meine Software ist **bereit fürs echte Leben**!«

Die richtige Welt

Gutes Design = flexible Software

5 (Teil 1)

Nichts bleibt jemals gleich

Veränderung ist unausweichlich. Egal ob Ihnen Ihre Software so gefällt, wie sie jetzt ist, morgen wird sie sich wahrscheinlich **ändern**. Und je schwerer Sie es machen, dass sich Ihre Software ändern kann, umso schwerer wird es, auf die **sich ändernden Anforderungen Ihres Kunden** zu reagieren. In diesem Kapitel werden wir einen alten Freund besuchen und versuchen, ein bestehendes Softwareprojekt zu ändern. Dabei werden wir uns ansehen, wie sich **kleine Änderungen in große Probleme verwandeln können**. Und wir werden tatsächlich ein Problem aufdecken, für dessen Lösung wir ein ZWEITEILIGES Kapitel brauchen!

5 (Intermezzo)

Gutes Design = flexible Software

5 (Teil 2)

Geben Sie Ihrer Software ein 30-Minuten-Workout

Haben Sie sich jemals gewünscht, Sie wären flexibler?

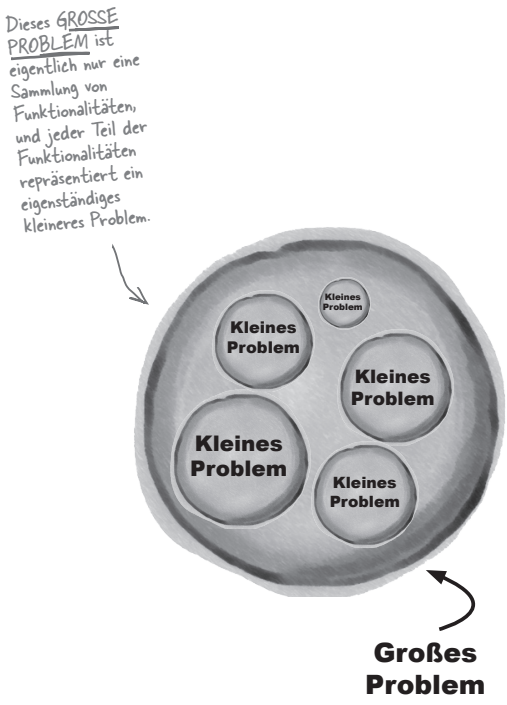
Wenn Sie beim Ändern Ihrer Anwendung auf Probleme stoßen, bedeutet das wahrscheinlich, dass Ihre Software **flexibler und resistenter sein muss**. Um Ihre Anwendung etwas zu stärken, werden Sie etwas analysieren, eine Menge designen und lernen, wie OO-Prinzipien helfen können, **Ihre Anwendung aufzulockern**. Und im großen Finale werden Sie erfahren, wie **eine höhere Kohäsion Ihren Bindungen helfen kann**. Klingt interessant? Blättern Sie um, damit wir uns wieder der Reparatur dieser unflexiblen Anwendung widmen können.

Die richtig großen Probleme lösen

6 »Mein Name ist Art Vandelay ... Ich bin Architekt«

Es ist Zeit, etwas RICHTIG GROSSES aufzubauen. Sind Sie bereit?

Sie haben Massen von Werkzeugen in Ihrem OOA&D-Werkzeugkasten. Aber wie setzen Sie diese Werkzeuge ein, wenn Sie etwas **richtig Großes** aufbauen müssen? Na, vielleicht haben Sie es ja noch nicht erkannt, aber **Sie haben bereits alles, was Sie brauchen**, um große Probleme anzugehen. Wir werden ein paar neue Werkzeuge kennenlernen wie **Bereichsanalyse** und **Anwendungfallsdiagramme**, aber selbst diese neuen Werkzeuge basieren auf Dingen, die Sie bereits kennen – beispielsweise dass man auf den Kunden hören soll und verstehen muss, was man aufbaut, bevor man beginnt, Code zu schreiben. Machen Sie sich bereit ... es ist Zeit, Architekt zu spielen.



Architektur

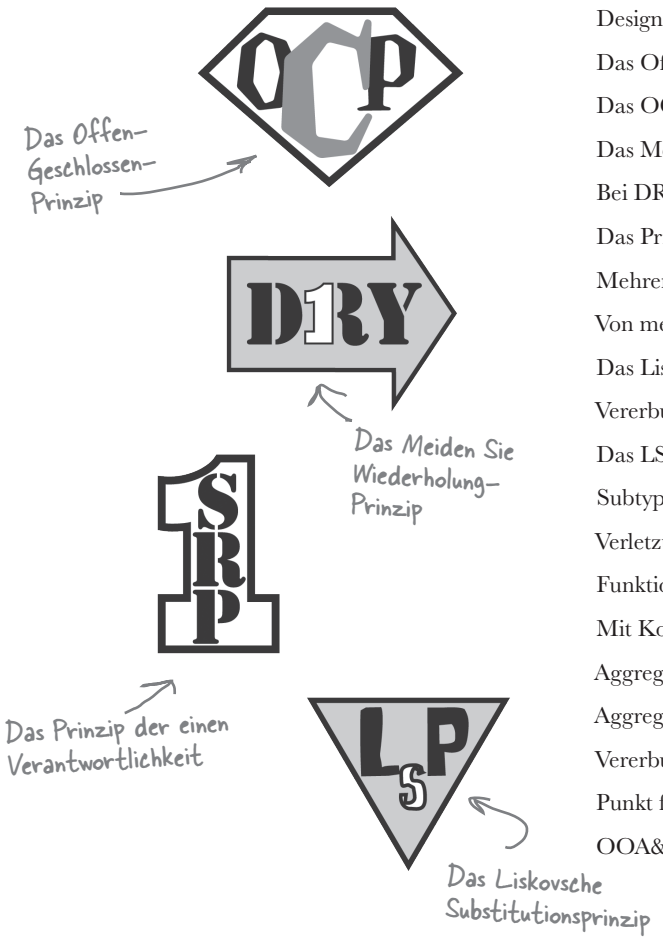
7 Ordnung ins Chaos bringen

Irgendwo müssen Sie anfangen, aber Sie wählen besser das *richtige* Irgendwo! Sie wissen, dass Sie Ihre Anwendung in viele kleine Probleme aufbrechen müssen. Aber das heißt bloß, dass Sie **VIELE** kleine Probleme haben. In diesem Kapitel werden wir Ihnen helfen, herauszufinden, **wo Sie anfangen sollten**, und sorgen dafür, dass Sie keine Zeit damit verschwenden, an den falschen Dingen zu arbeiten. Es ist Zeit, all diese **kleinen Teile** zu nehmen, die auf Ihrem Arbeitsplatz herumliegen, und herauszufinden, wie Sie sie in eine **wohlgeordnete, sauber entworfene Anwendung** verwandeln. Unterwegs werden Sie noch etwas zu den allmächtigen **drei Fs der Architektur** lernen und erfahren, warum **Risiko** mehr als ein cooles Kriegsspiel aus den 80er-Jahren ist.

Design-Prinzipien

8 Originalität wird überschätzt

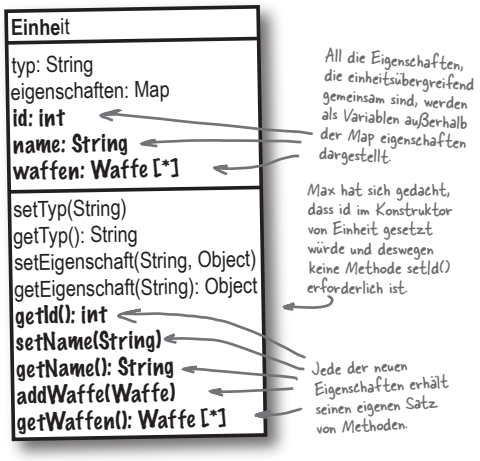
Nachahmung ist die aufrichtigste Form, keine Dummheiten zu machen. Es gibt nichts, was so befriedigend ist wie die Entdeckung einer vollständig neuen und originellen Lösung für ein Problem, das einen seit Tagen quält – bis man entdeckt, dass ein anderer schon **das gleiche Problem gelöst hat**, lange vor einem, und seine Sache dabei noch besser gemacht hat als man selbst! In diesem Kapitel werden wir uns einige **Design-Prinzipien** anschauen, die andere über die Jahre gefunden haben, und wie sie Sie zu einem besseren Programmierer machen können. Legen Sie Ihren Anspruch ab, »es auf meine Weise zu machen«. In diesem Kapitel geht es darum, **den klügeren, schnelleren Weg zu nehmen.**



Iteration und Tests

9 Die Software ist immer noch für den Kunden

Es ist Zeit, dem Kunden zu zeigen, wie viele Gedanken Sie sich wirklich machen. Nörgelnde Chefs? Besorgte Kunden? Teilhaber, die ständig fragen: »Wird es rechtzeitig fertig?« Der ganze sauber entworfene Code wird Ihre Kunden nicht zufriedenstellen. Sie müssen **Ihnen etwas zeigen, das funktionieriert**. Und da Sie jetzt einen stabilen Werkzeugkasten für die OO-Programmierung haben, ist es Zeit, dass Sie lernen, wie Sie **dem Kunden beweisen können**, dass Ihre Software funktioniert. In diesem Kapitel werden wir zwei Wege kennenlernen, **tiefer** in die Funktionalität Ihrer Software **einzutauchen** und dem Kunden das angenehme Gefühl zu vermitteln, das ihn sagen lässt: ***Ja. Du hast definitiv den richtigen Entwickler für diesen Job!***



Der OOA&D-Lebenszyklus

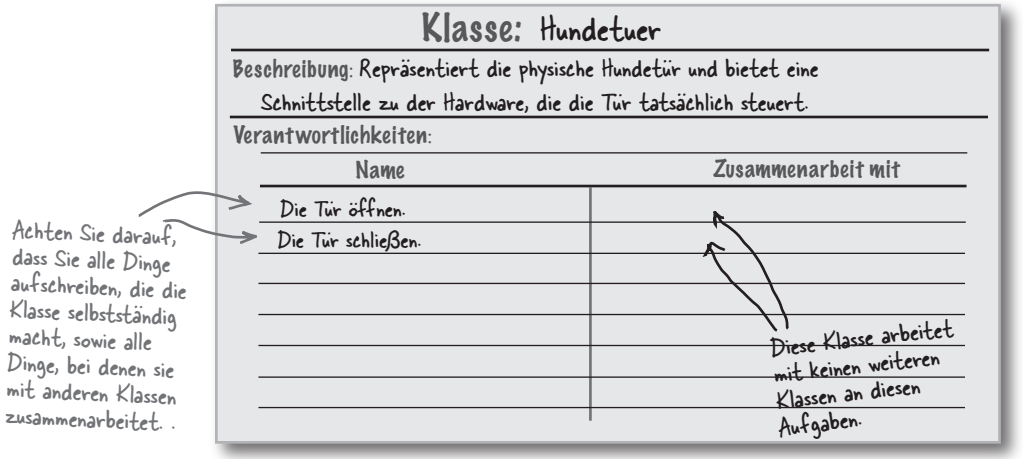
10 Alles zusammenfügen

Sind wir schon angekommen? Wir haben eine Menge an den einzelnen Möglichkeiten gearbeitet, Ihre Software zu verbessern, aber jetzt ist es Zeit, **alles zusammenzufügen**. Das ist doch das, worauf Sie gewartet haben: Wir werden **alles** nehmen, was Sie gelernt haben, und Ihnen zeigen, wie das alles Teil eines **einzigen Prozesses** ist, den Sie immer wieder einsetzen können, um **gute Software zu schreiben**.

Anhang A: Was übrig bleibt

Die Top Ten der Themen, die wir nicht behandelt haben

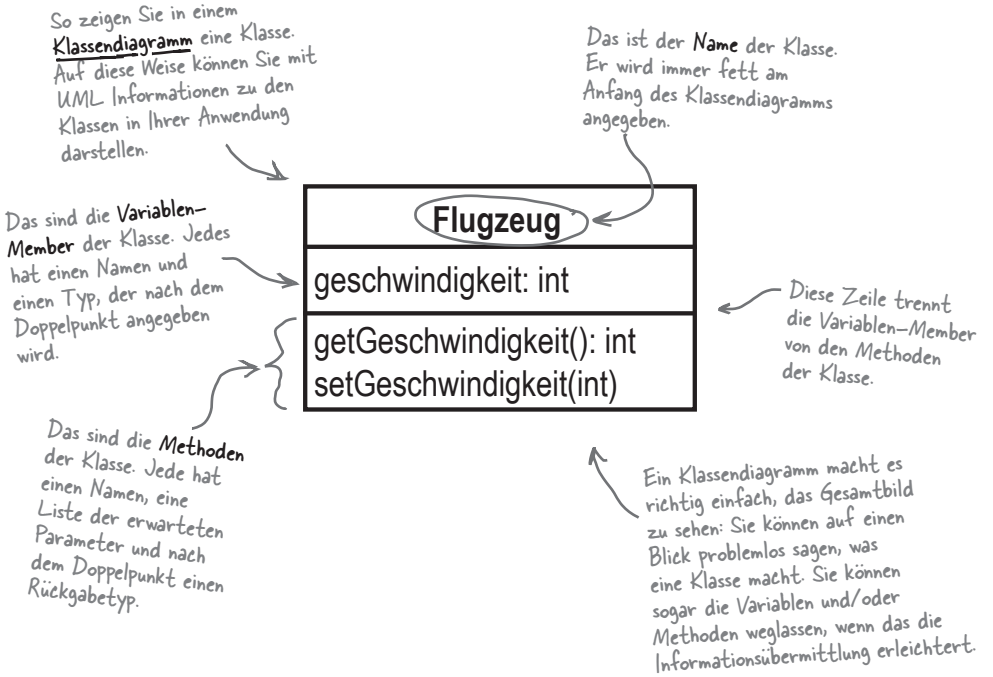
Glauben Sie es oder nicht: Es kommt noch mehr. Ja. Auch mit über 500 Seiten im Rücken gibt es immer noch Dinge, die wir nicht reinstopfen konnten. Selbst wenn diese abschließenden Top-Ten-Themen nicht mehr als eine Erwähnung verdienen, wollten wir Sie nicht ohne ein paar zusätzliche Informationen über jedes von ihnen aus Objekthausen entlassen. Kopf hoch. Schließlich haben Sie dann etwas mehr, über das Sie während der Werbeunterbrechungen bei OO-KATASTROPHE! reden können ... und wer steht nicht gelegentlich auf eine nette OOA&D-Plauderei?



Anhang B: Anleitung für Objekthausen

B Die OO-Sprache sprechen

Sind Sie bereit für für eine Reise in ein fremdes Land? Objekthausen ist eine seltsame Stadt, deren Umgangs- und Verkehrsregeln Ihnen beim ersten Aufenthalt vielleicht merkwürdig vorkommen könnten. Sie fühlen sich ausgeschlossen, weil das **Erben** scheinbar nur bei den Anderen funktioniert? Die häufige Erwähnung von **Polymorphie** lässt bei Ihnen ernsthafte Zweifel an den Moralvorstellung der Einheimischen aufsteigen? Dann sollten Sie zur **Akklimatisierung** zunächst vielleicht diesen Anhang lesen, um sich mit den Sitten und der Sprache von Objekthausen **vertraut zu machen**. Sie müssen sich keine Sorgen machen. Es wird nicht lange dauern, und bevor Sie es merken, werden Sie die **OO-Sprache sprechen**, als würden Sie schon seit Jahren in den sauber entworfenen Gebieten von Objekthausen leben.



Wie man dieses Buch benutzt

Einführung

In diesem Abschnitt beantworten wir die brennende Frage:
»Und? Warum STEHT so was in einem OOA&D-Buch?«

Für wen ist dieses Buch?

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

1. Können Sie Java? (Sie müssen kein Guru sein.)

Wahrscheinlich kommen Sie auch klar, wenn Sie stattdessen C# beherrschen.

2. Möchten Sie objektorientierte Analyse und Design **lernen, verstehen, behalten und auf echte Projekte *anwenden*** und so dazu kommen, bessere Software zu schreiben?

3. Ziehen Sie **anregende Partyunterhaltungen** trockenen, öden, akademischen Vorlesungen vor?

... dann ist dieses Buch etwas für Sie.

Wer sollte eher die Finger von diesem Buch lassen?

Wenn Sie ***eine*** dieser Fragen mit »Ja« beantworten müssen ...

1. Sind Sie ein **völliger Java-Neuling**? (Sie müssen keine fortgeschrittenen Kenntnisse haben, und selbst, wenn Sie kein Java können, aber C# beherrschen, werden Sie fast alle Codebeispiele verstehen. Auch ein reiner C++-Hintergrund könnte ausreichend sein.)

2. Sie sind ein Top-OO-Designer/-Entwickler und suchen nach einem ***Nachschlage*werk?**

3. Sie haben **Angst, etwas Neues auszuprobieren**? Sie unterziehen sich lieber einer Wurzelbehandlung, als in einer Streifen-/Karo-Kombination auf die Straße zu gehen? Sie sind überzeugt, dass ein Fachbuch nicht seriös sein kann, wenn Programmierkonzepte vermenschlicht werden?

... dann ist dieses Buch nicht das richtige für Sie.

[Anmerkung aus dem Marketing: Dieses Buch ist etwas für jeden, der eine Kreditkarte besitzt!]

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*?«

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.

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!

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

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.

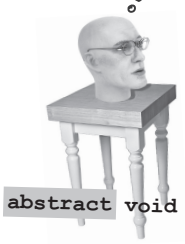
Einige der Lernprinzipien dieser Buchreihe:

All das wird durch ein einziges Verbindung-Objekt dargestellt.

Bilder einsetzen. 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. **Text in oder neben die Grafiken setzen,** 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.

Verwenden Sie 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. Halten Sie keinen Vortrag, sondern erzählen Sie Geschichten. Benutzen Sie eine zwanglose Sprache. Nehmen Sie sich selbst nicht zu ernst. 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 Körper...



```
abstract void wandern();
```

Kein Methoden-Body! Am Ende steht ein Semikolon.

Bringen Sie den Lernenden dazu, 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.

Gute Software und das *jedes Mal*. Ich kann mir kaum **vorstellen**, wie das wäre!

Ziehen Sie die Aufmerksamkeit des Lesers auf sich – und behalten Sie sie. 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.

Sprechen Sie 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, was 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 etwas über Entwurfsmuster lernen möchten, wenn Sie dieses Buch in den Händen halten. Und wahrscheinlich möchten Sie nicht viel Zeit aufwenden. Und Sie wollen sich an das *erinnern*, was Sie lesen, und es anwenden können. Und deshalb müssen Sie es *verstehen*. 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, Entwurfsmuster für so wichtig zu halten wie einen Tiger?

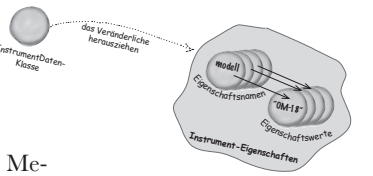
Da gibt es den langsamen, ermüdenden Weg oder den schnelleren, effektiveren Weg. Der langsame Weg geht über bloße Wiederholung. Natürlich ist Ihnen klar, dass Sie lernen und sich sogar an die langweiligsten Themen erinnern *können*, wenn Sie sich die gleiche Sache immer wieder einhämmern. Wenn Sie nur 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 in einem Raum passive 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 ***Redundanz*** 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 ***unerwarteter*** 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.***

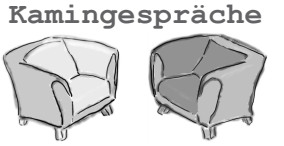
Wir haben einen ***umgangssprachlichen Stil*** mit direkter Anrede benutzt, denn Ihr Gehirn ist von Natur aus aufmerksamer, wenn es Sie in einer Unterhaltung wähnt als wenn es davon ausgeht, dass Sie passiv einer Präsentation zuhören – sogar dann, wenn Sie *lesen.*

Wir haben mehr als 40 ***Aktivitäten*** 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, denn 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 Code-Beispiel sehen möchte. Aber ganz abgesehen von den jeweiligen Lernvorlieben profitiert *jeder* davon, wenn er die gleichen Inhalte in unterschiedlicher Form präsentiert bekommt.

Punkt für Punkt

Wir liefern Inhalte für ***beide Seiten Ihres Gehirns,*** denn je mehr Sie von Ihrem Gehirn 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.

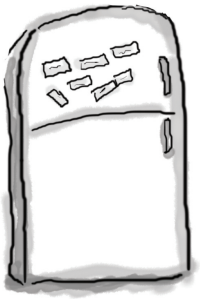


Und wir haben ***Geschichten*** und Übungen aufgenommen, die ***mehr als einen Blickwinkel repräsentieren,*** denn Ihr Gehirn lernt von Natur aus intensiver, wenn es gezwungen ist, selbst zu analysieren und zu beurteilen.

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. Überlegen Sie: Ihren *Körper* bekommen Sie ja auch nicht in Form, wenn Sie nur die Leute 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 übermäßig 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 Software-Design machen wollen. Deshalb besprechen wir nicht *alles.* Nur das, was Sie wirklich *brauchen* 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!*** Überspringen Sie sie nicht.

4. **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.

5. **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.

6. **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.

7. **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.

8. **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.

9. ***Entwerfen* Sie etwas!**
   Wenden Sie das hier Gelernte auf einen Entwurf an, an dem Sie gerade arbeiten, oder gestalten Sie ein älteres Projekt damit um. Tun Sie *irgendetwas*, um neben den Übungen in diesem Buch weitere Erfahrungen zu sammeln. Sie brauchen dazu nur einen Bleistift und ein zu lösendes Problem ... ein Problem, das von der einen oder anderen Technik profitieren würde, die wir hier besprechen.

Lies mich

Dies ist ein Lehrbuch, keine Referenz. Wir haben mit Absicht alles weggelassen, was Ihnen dabei in die Quere kommen könnte, das zu lernen, was auch immer wir an einem bestimmten Punkt gerade behandeln. Und wenn Sie das Buch das erste Mal durchhaben, müssen Sie wieder am Anfang beginnen, weil das Buch Annahmen darüber macht, was Sie bereits gesehen und gelernt haben.

Wir gehen davon aus, dass Sie mit Java vertraut sind.

Es würde ein vollständiges Buch brauchen, um Ihnen Java beizubringen (und das hat es in der Tat auch gebraucht: *Java von Kopf bis Fuß*). Wir haben uns entschlossen, uns in diesem Buch auf Analyse und Design zu konzentrieren. Die Kapitel wurden also in der Annahme geschrieben, dass Sie mit den Grundlagen von Java vertraut sind. Wenn fortgeschrittene oder komplexe Konzepte auftauchen, werden diese allerdings so erläutert, als wären sie Ihnen vollkommen unbekannt.

Wenn Sie ein absoluter Java-Neuling sind oder sich diesem Buch mit einem C#- oder C++-Hintergrund nähern, empfehlen wir Ihnen dringend, zum Ende des Buchs zu gehen und Anhang B zu lesen, bevor Sie weitermachen. Dieser Anhang enthält einführenden Stoff, der Ihnen helfen wird, dieses Buch auf dem richtigen Fuß zu erwischen.

Java 5 nutzen wir nur, wenn wir es müssen.

Java 5.0 führt in die Programmiersprache Java viele neue Features ein – von Generics über parametrisierte und enumerierte Typen bis zu `foreach`-Schleifenkonstrukten. Da viele professionelle Java-Entwickler gerade erst zu Java 5 wechseln, wollten wir vermeiden, dass Sie an einer neuen Syntax hängen bleiben, während Sie versuchen, OOA&D zu lernen. In den meisten Fällen bleiben wir bei einer Vor-Java 5-Syntax. Die einzige Ausnahme bildet Kapitel 1, in dem wir einen enumerierten Typ brauchen – aber wir haben Enums in diesem Abschnitt etwas ausführlicher erklärt.

Wenn Sie ein Java 5-Neuling sind, sollten Sie keine Probleme mit den Codebeispielen haben. Sind Sie mit Java 5 bereits vertraut, erhalten Sie ein paar Compiler-Warnungen auf Grund ungeprüfter oder unsicherer Operationen, da wir keine typisierten Collections verwenden. Aber Sie sollten dazu in der Lage sein, den Code problemlos selbstständig für Java 5 zu aktualisieren.

Die Aktivitäten sind NICHT optional.

Die Übungen und Aktivitäten sind keine Zusätze. Sie sind Teil des Kerninhalts des Buchs. Einige von ihnen sollen Ihr Gedächtnis stützen, andere dienen dem Verständnis, und einige helfen Ihnen, das Gelernte anzuwenden. ***Überspringen Sie die Übungen nicht.*** Die Kreuzworträtsel sind die einzigen Dinge, die Sie nicht tun *müssen*. Aber sie geben Ihrem Gehirn eine Möglichkeit, über die Wörter und Begriffe nachzudenken, die Sie in unterschiedlichen Lernkontexten gelernt haben.

Die Wiederholungen sind beabsichtigt und wichtig.

Ein entscheidendes Merkmal eines Von Kopf bis Fuß-Buchs ist, dass wir wollen, dass Sie die Sache *wirklich* verstehen. Und wir wollen, dass Sie die Dinge behalten, wenn Sie mit dem Buch fertig sind. Behalten und Wiedererinnern gehören nicht zu den Zielen der meisten Referenzbücher. Aber in diesem Buch geht es um das *Lernen,* und deswegen werden Sie die gleichen Konzepte manchmal mehrfach auftauchen sehen.

Die Beispiele sind so schlank wie möglich.

Unsere Leser sagen uns, dass es einfach frustrierend sei, durch 200 Zeilen eines Beispiels zu irren, um nach den zwei Zeilen zu suchen, um deren Verständnis es geht. Die meisten Beispiele in diesem Buch werden mit so wenig Inhalt wie möglich gezeigt, damit der Teil, den es zu lernen gilt, klar und einfach ist. Erwarten Sie nicht, dass alle Beispiele robust oder überhaupt vollständig sind – sie wurden ganz gezielt für Lehrzwecke geschrieben und sind nicht immer vollständig funktional.

In einigen Fällen haben wir nicht alle erforderlichen Importanweisungen eingeschlossen, aber wir gehen davon aus, dass Sie, wenn Sie Java-Programmierer sind, beispielsweise wissen, dass sich **`ArrayList`** in **`java.util`** befindet. Wenn die Importe nicht Teil der normalen J2SE-Kern-API sind, erwähnen wir das. Außerdem haben wir den gesamten Quellcode ins Web gestellt, damit Sie ihn herunterladen können. Sie finden ihn auf der Website zu diesem Buch:

`http://www.oreilly.de/catalog/hfobjectsger/`

Damit Sie sich auf die Lehrseite des Codes konzentrieren können, haben wir unsere Klassen nicht in Packages gesteckt (anders gesagt, Sie sind alle im Java-Default-Package). Das empfehlen wir Ihnen nicht fürs richtige Leben. Und wenn Sie die Codebeispiele für dieses Buch herunterladen, werden Sie feststellen, dass die Klassen in Packages *stecken*.

Zu den Kopfnuss-Übungen gibt es keine Lösungen.

Für manche dieser Übungen gibt es keine richtige Lösung, und bei anderen gehört es zum Lernprozess der Kopfnuss-Aktivitäten, dass Sie selbst überlegen, ob und wann Ihre Lösungen richtig sind. Bei einigen Kopfnuss-Übungen finden Sie Hinweise, die Sie in die richtige Richtung lenken.

Die Fachgutachter

Fachgutachter:

Gewaltigen Dank an unser wunderbares Fachgutachter-Trio. Diese Jungs haben Fehler gefunden, die wir übersehen haben, uns gesagt, wo wir zu schnell weitergegangen sind (und wo zu langsam), und es uns sogar wissen lassen, wenn unsere Witze einfach nur nervig waren. Mehrfach haben sie in wenigen Stunden ganze Kapitel umgekrempelt ... wir sind uns nicht ganz klar, ob das heißt, dass sie wirklich eine Hilfe sind oder einfach etwas Abstand von der Softwareentwicklung brauchen. **Hannibal** insbesondere hat uns die Woche versüßt, als er uns wissen ließ, dass der große OOA&D-Pfeil in Kapitel 10 »Heiß!« ist. Danke Jungs, ohne eure harte Arbeit wäre dieses Buch nicht annähernd so solide.

Kathy Sierra und Bert Bates:

Wir sind immer noch erstaunt, welch tiefes Wissen **Bert Bates** über Kliffe und **Kathy Sierra** über Hundetüren hat. Falls das komisch auf Sie wirkt – alles, was Sie über fast egal was wissen, wird auf den Kopf gestellt, wenn Sie diese beiden treffen, und trotzdem haben wir alle von ihrer Hilfe sehr profitiert.

Bert und Kathy haben auf den letzten Drücker eine Menge Begutachtungen durchgeführt, und wir danken ihnen für ihre Arbeit. Ihre Hilfe und Anleitung bleibt das Herz der Von Kopf bis Fuß-Bücher.