

# Der Inhalt (in der Übersicht)

	Einführung	xxiii
1	Willkommen bei den Entwurfsmustern: <i>eine Einführung</i>	1
2	Ihre Objekte auf dem Laufenden halten: <i>das Observer-Muster</i>	37
3	Objekte dekorieren: <i>das Decorator-Muster</i>	79
4	Backen in OO-Qualität: <i>das Factory-Muster</i>	109
5	Ein einzigartiges Objekt: <i>das Singleton-Muster</i>	169
6	Aufrufe inkapseln: <i>das Command-Muster</i>	191
7	Anpassungsfähigkeit beweisen: <i>das Adapter- und das Facade-Muster</i>	235
8	Algorithmen inkapseln: <i>das Template Methode-Muster</i>	275
9	Erfolgreiche Kollektionen: <i>das Iterator- und das Composite-Muster</i>	315
10	Die Zustände in Objekthäusern: <i>das State-Muster</i>	385
11	Den Zugriff auf Objekte kontrollieren: <i>das Proxy-Muster</i>	429
12	Muster von Mustern: <i>zusammengesetzte Muster</i>	499
13	Entwurfsmuster in der realen Welt: <i>besser leben mit Mustern</i>	577
14	Anhang: <i>übriggebliebene Muster</i>	611

# Der Inhalt (jetzt ausführlich)

## Einführung

**Ihr mustergültiges Gehirn.** 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 Entwurfsmuster zu wissen?

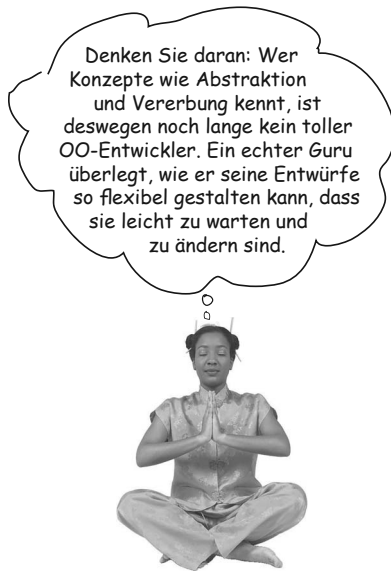
Für wen ist dieses Buch?	xxiv
Wir wissen, was Ihr Gehirn denkt	xxv
Metakognition	xxvii
Machen Sie sich Ihr Hirn untertan	xxix
Die Fachgutachter	xxxii
Danksagungen	xxxiii

# Willkommen bei den Entwurfsmustern

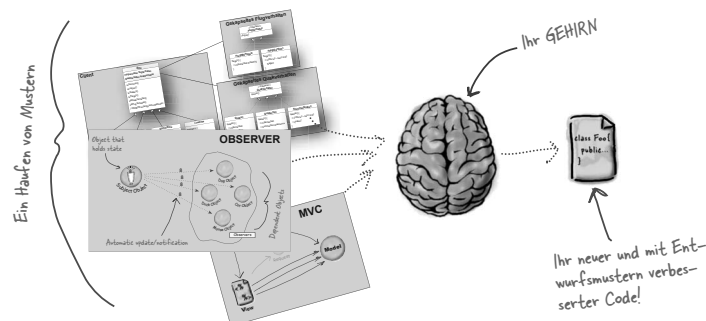
# 1

## Willkommen bei den Entwurfsmustern

**Irgendjemand hat Ihre Probleme bereits gelöst.** In diesem Kapitel lernen Sie, warum (und wie) Sie die Erfahrungen und Lektionen verwerthen können, die andere Entwickler gelernt haben, die in den gleichen Entwurfsschwierigkeiten steckten und den Trip überlebt haben. Dazu werden wir einen Blick auf die Verwendung und die Vorteile von Entwurfsmustern werfen, uns einige grundlegende OO-Entwurfsprinzipien ansehen und ein Beispiel dafür durchgehen, wie ein bestimmtes Muster funktioniert. Am besten arbeiten Sie mit Mustern, indem Sie *Ihr Gehirn mit ihnen aufladen* und dann in Ihren Entwürfen und in bestehenden Anwendungen die *Punkte erkennen*, an denen Sie sie *anwenden können*. An Stelle von *Code*-Wiederverwendung bieten Ihnen Muster *Erfahrungs*-Wiederverwendung.



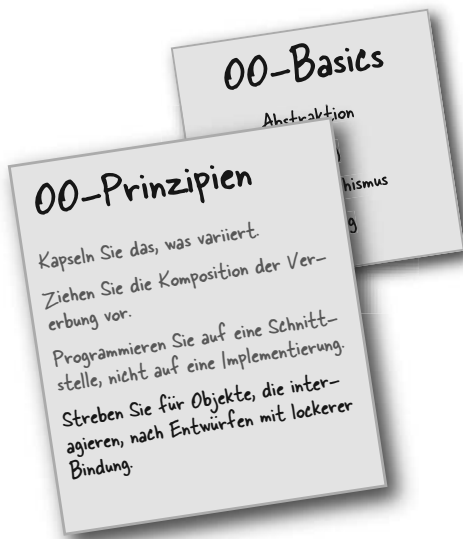
Die SimEnte-Anwendung	2
Eike denkt über Vererbung nach	5
Und wie wäre es mit einem Interface?	6
Die eine Konstante bei der Software-Entwicklung	8
Das, was veränderlich ist, von dem trennen, was gleich bleibt	10
Das Entenverhalten entwerfen	11
Den Enten-Code testen	18
Verhalten dynamisch setzen	20
Noch mal im Ganzen: Gekapseltes Verhalten	22
HAT-EIN kann IST-EIN überlegen sein	23
Das Strategy-Muster	24
Die Macht eines gemeinsamen Mustervokabulars	28
Wie verwende ich Entwurfsmuster?	29
Werkzeuge für Ihren Design-Werkzeugkasten	32
Lösungen zu den Aufgaben	34



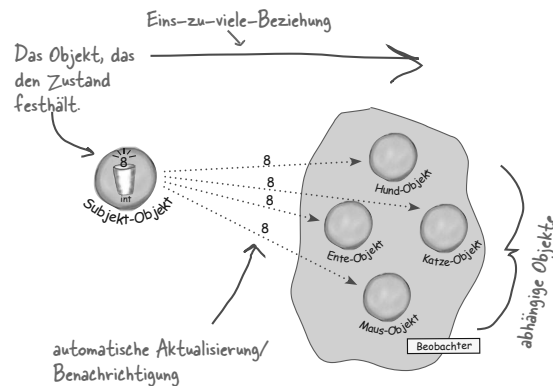
# Das Observer-Muster

## 2 Ihre Objekte auf dem Laufenden halten

**Verpassen Sie nicht, wenn etwas Interessantes passiert!** Wir haben ein Muster, das Ihre Objekte auf dem Laufenden hält, wenn etwas passiert, das Sie interessieren könnte. Objekte können sogar zur Laufzeit entscheiden, ob sie informiert werden möchten. Das Observer-Muster ist eins der Muster, die im JDK am häufigsten verwendet werden. Und es ist unglaublich nützlich. In diesem Kapitel sehen wir uns außerdem Eins-zu-viele-Beziehungen und lockere Bindungen an. Mit dem Observer-Muster werden Sie zum Mittelpunkt der Muster-Party.



Die Wetterstation-Anwendung im Überblick	39
Gestatten: das Observer-Muster	44
Herausgeber + Abonnenten = Observer-Muster	45
Fünf-Minuten-Drama: ein Subjekt unter Beobachtung	48
Die Definition des Observer-Musters	51
Die Macht der lockeren Bindung	53
Die Wetterstation entwerfen	56
Die Wetterstation implementieren	57
Java's eingebautes Observer-Muster verwenden	64
Die dunkle Seite von java.util.Observable	71
Werkzeuge für Ihren Design-Werkzeugkasten	74
Die Lösungen zu den Aufgaben	77

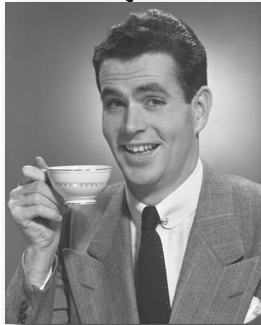


## Das Decorator-Muster

# 3 Objekte dekorieren

**Nennen wir dieses Kapitel einfach »Vererbst du noch oder designst du schon?«.** Wir untersuchen noch einmal einen typischen Fall überstrapazierter Vererbung, und Sie werden lernen, wie Sie Ihre Klassen mit Hilfe einer Form der Objekt-Zusammensetzung erst zur Laufzeit »dekorieren«. Warum? Wenn Ihnen die Techniken des Dekorierens einmal vertraut sind, können Sie Ihren Objekten (oder den Objekten anderer) neue Aufgaben geben, ohne den Code der zu Grunde liegenden Klasse ändern zu müssen.

Früher dachte ich immer, echte Männer bilden grundsätzlich für alles Unterklassen. Bis ich gelernt habe, welche Macht man in den Händen hält, wenn man zur Laufzeit und nicht zur Kompilierzeit erweitert. Und sehen Sie mich heute mal an!



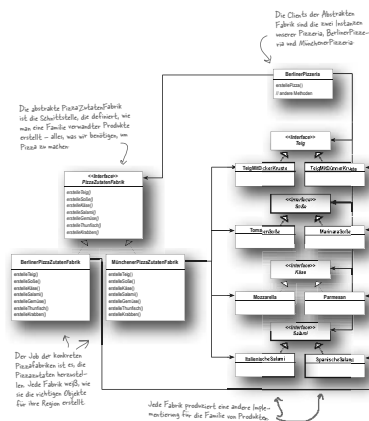
Willkommen bei Sternback-Kaffee	80
Das Offen/Geschlossen-Prinzip	86
Dürfen wir vorstellen: das Decorator-Muster!	88
Ein Getränk mit Dekorierern aufbauen	89
Die Definition des Decorator-Musters	91
Getränke dekorieren	93
Den Sternback-Code schreiben	95
Dekorierer aus der Praxis: Java I/O	100
Einen eigenen I/O-Dekorierer schreiben	102
Werkzeuge für Ihren Design-Werkzeugkasten	105
Lösungen zu den Übungen	106

# Das Factory-Muster

## 4

### Backen in OO-Qualität

Machen Sie sich bereit, ein paar locker gebundene OO-Entwürfe zu backen. Das Erstellen von Objekten hat mehr zu bieten als die simple Verwendung des new-Operators. Sie werden lernen, dass Instanziierung eine Aktivität ist, die nicht immer in der Öffentlichkeit verübt werden sollte und oft zu Bindungsproblemen führen kann. Und das wollen Sie doch nicht, oder? Lernen Sie, wie Sie das Factory-Muster vor lästigen Abhängigkeiten retten kann.



Die Aspekte identifizieren, die veränderlich sind	112
Die Objekt-Erstellung kapseln	114
Eine einfache Pizzafabrik erstellen	115
Die Definition der einfachen Fabrik	117
Ein Framework für die Pizzeria	120
Die Unterklassen entscheiden lassen	121
Eröffnen wir also eine Pizzeria	123
Eine Fabrikmethode deklarieren	125
Zeit, das Factory Method-Muster zu treffen	131
Parallele Klassenhierarchien	132
Die Definition des Factory Method-Musters	134
Eine sehr abhängige Pizzeria	137
Ein Blick auf Objekt-Abhängigkeiten	138
Das Prinzip der Umkehrung der Abhängigkeiten	139
Inzwischen in der Pizzeria	144
Zutatenfamilien	145
Die Zutatenfabriken aufbauen	146
Was wir gemacht haben	153
Die Definition des Abstract Factory-Musters	156
Factory Method und Abstract Factory im Vergleich	160
Werkzeuge für Ihren Design-Werkzeugkasten	162
Lösungen zu den Übungen	164

## Das Singleton-Muster

# 5

### Ein einzigartiges Objekt

Unser nächster Halt ist das Singleton-Muster, unsere Fahrkarte zur Erstellung einzigartiger Objekte von Klassen, von denen es nur eine einzige Instanz geben kann. Vielleicht freut es Sie zu erfahren, dass das Singleton-Muster in Bezug auf das Klassendiagramm das einfachste aller Muster ist. Das Diagramm enthält tatsächlich nur eine einzige Klasse! Aber machen Sie es sich nicht zu bequem. Trotz der Einfachheit in Bezug auf das Klassendiagramm werden wir auf eine Reihe Buckel und Schlaglöcher in seiner Implementierung stoßen. Sie schnallen sich also besser an.



Das kleine Singleton	171
Die klassische Implementierung des Singleton-Musters sezieren	173
Bekenntnisse eines Singleton	174
Die Schokoladenfabrik	175
Definition des Singleton-Musters	177
<sup>Köln</sup> <del>Houston</del> , wir haben ein Problem ...	178
Spielen Sie JVM	179
Mit Multithreading klarkommen	180
Fragen und Antworten zum Singleton	184
Werkzeuge für Ihren Design-Werkzeugkasten	186
Lösungen zu den Übungen	188

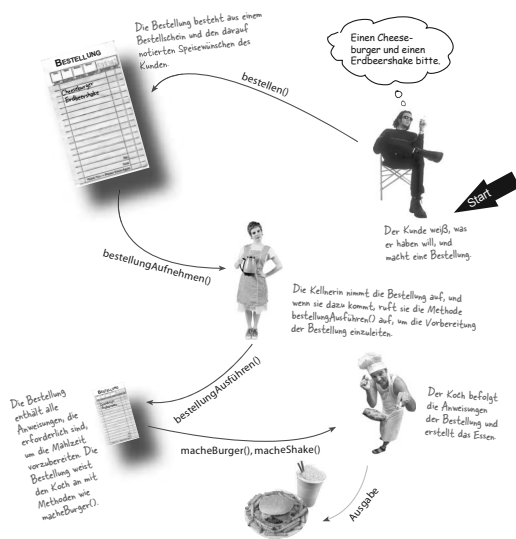


# Das Command-Muster

## 6

### Aufrufe einkapseln

In diesem Kapitel heben wir die Kapselung noch einmal auf ein ganz neues Niveau: Wir werden **Methodenaufrufe einkapseln**. Ja, wirklich. Indem wir den Methodenaufruf kapseln, können wir Teile von Berechnungen einfrieren, damit das Objekt, das die Berechnung aufruft, sich nicht darum kümmern muss, wie diese Dinge gemacht werden. Es verwendet einfach unsere eingefrorene Methode, um sie ausführen zu lassen. Mit diesen eingekapselten Methodenaufrufen können wir außerdem einige unverschämt geschickte Dinge tun, sie beispielsweise speichern, um sie zu protokollieren, oder wiederverwenden, um unserem Code eine Rückgängig-Funktionalität zu spendieren.



Die Fernsteuerung	193
Werfen wir einen Blick auf die Klassen der Hersteller	194
Inzwischen im Restaurant	197
Sehen wir uns das Zusammenspiel etwas gründlicher an	198
Rollen und Verantwortlichkeiten im Restaurant Objekthausen	199
Vom Restaurant zum Command-Muster	201
Unser erstes Befehl-Objekt	203
Die Definition des Command-Musters	206
Den Fernsteuerungsplätzen Befehle zuweisen	209
Die Fernbedienung implementieren	210
Die Fernsteuerung in Gang setzen	212
Zeit, diese Dokumentation zu schreiben	215
Einen Status verwenden, um Rückgängig zu implementieren	220
Jede Fernsteuerung braucht einen Party-Modus!	224
Einen Makro-Befehl verwenden	225
Weitere Verwendungen: Warteschlangen für Befehle	228
Weitere Verwendungen: Anfragen protokollieren	229
Werkzeuge für Ihren Design-Werkzeugkasten	230
Lösungen zu den Aufgaben	232

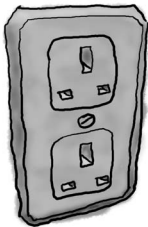
# Die Adapter- und Facade-Muster

## 7

### Anpassungsfähigkeit beweisen

In diesem Kapitel werden wir uns an unmöglichen Dingen versuchen – einen rechteckigen Pflock in ein rundes Loch zu stecken beispielsweise. Klingt unmöglich? Nicht, wenn man Design-Patterns hat. Erinnern Sie sich an das Decorator-Muster? Wir haben Objekte umhüllt, um ihnen neue Verantwortlichkeiten zu geben. Jetzt werden wir einige Objekte mit einem anderen Ziel einpacken: um ihren Schnittstellen den Anschein zu verleihen, dass sie wie etwas aussehen, das sie nicht sind. Warum sollten wir das tun? Wir haben damit die Möglichkeit, ein Design, das eine bestimmte Schnittstelle erwartet, an eine Klasse anzupassen, die eine andere Schnittstelle implementiert. Und das ist nicht alles. Während wir dabei sind, werden wir uns noch ein weiteres Muster ansehen, das Objekte umhüllt, um ihre Schnittstelle zu vereinfachen.

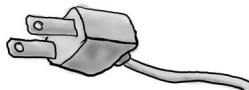
Steckdose



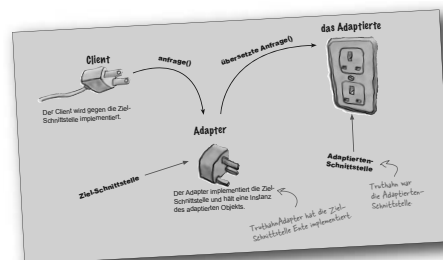
Adapter



Stecker



Adapter, wo wir nur hinschauen	236
Objektorientierte Adapter	237
Das Adapter-Muster erklärt	241
Die Definition des Adapter-Musters	243
Objekt- und Klassen-Adapter	244
Kamingespräche: Objekt- und Klassen-Adapter	247
Adapter aus dem wirklichen Leben	248
Einen Enumerator an einen Iterator anpassen	249
Kamingespräche: Decorator- und Adapter-Muster	252
Gemütliches Heimkino	255
Beleuchtung, Kamera, Fassade	258
Die Heimkino-Fassade aufbauen	261
Die Definition des Facade-Musters	264
Das Prinzip der Verschwiegenheit	265
Werkzeuge für Ihren Design-Werkzeugkasten	270
Lösungen zu den Aufgaben	272



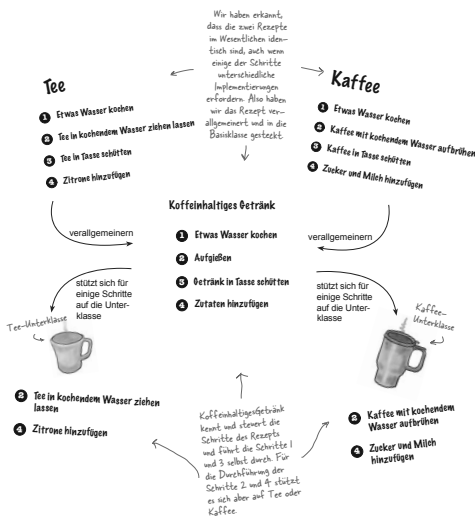
# Das *Template Method*-Muster

## 8

### Algorithmen einkapseln

**Wir sind auf dem totalen Kapselungstrip. Wir haben die Objekt-Erstellung eingekapselt, Methodenaufrufe, komplexe Schnittstellen, Enten, Pizzas ... was könnte als Nächstes kommen?** Wir werden dazu übergehen, Teile von

Algorithmen zu kapseln, damit Unterklassen sich jederzeit in eine Berechnung einkapseln können, wenn sie das möchten. Außerdem werden wir einiges über ein Entwurfsprinzip lernen, das von Hollywood inspiriert ist.



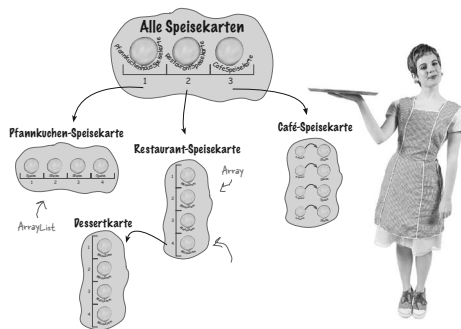
Ein paar Kaffee- und Tee-Klassen (in Java) zusammenrühren	277
Dürfte ich vielleicht Ihren Kaffee, Tee abstrahieren?	280
Den Entwurf weiterentwickeln	281
zubereitungsRezept() abstrahieren	282
Was also haben wir gemacht?	285
Dürfen wir vorstellen: das Template Method-Muster!	286
Kochen wir doch mal etwas Tee ...	287
Was hat uns das Template Method-Muster gebracht?	288
Die Definition des Template Method-Musters	289
Haken wir uns bei einer Template-Methode ein ...	292
Den Hook verwenden	293
Führen wir den Testlauf aus	294
Das Hollywood-Prinzip	296
Das Hollywood-Prinzip und das Template Method-Muster	297
Template-Methoden im wirklichen Leben	299
Mit dem Template Method-Muster sortieren	300
Wir haben ein paar Enten, die sortiert werden müssen	301
Enten mit Enten vergleichen	302
Der Aufbau einer Enten-Sortiermaschine	304
Swinging mit Frames	306
Kamingespräche: Template Method und Strategy	308
Werkzeuge für Ihren Design-Werkzeugkasten	311
Lösungen zu den Aufgaben	312

## Die Iterator- und Composite-Muster

# 9

### Erfolgreiche Kollektionen

Es gibt viele Möglichkeiten, Objekte in eine Sammlung zu packen. Stecken Sie sie in ein Array-, Stack-, -List- oder Hashtable-Objekt. Sie haben die freie Auswahl. Und jede hat ihre Vor- und Nachteile. Aber irgendwann wird Ihr Client über diese Objekte iterieren wollen. Werden Sie ihm Ihre Implementierung zeigen, wenn er das tut? Wir hoffen ganz entschieden, dass Sie das nicht tun werden! Es wäre einfach nicht professionell. Sie müssen Ihre Karriere nicht riskieren. Sie werden sehen, wie Sie es Clients ermöglichen, über Ihre Objekte zu iterieren, ohne dass er je sieht, wie Sie Ihre Objekte speichern. Sie werden auch lernen, wie Sie Super Collections von Objekten pflegen, die mit einem einzigen Satz einige beeindruckende Datenstrukturen überspringen können. Und wenn Ihnen das immer noch nicht ausreicht, werden Sie außerdem ein oder zwei Dinge über Objektverantwortlichkeit lernen.

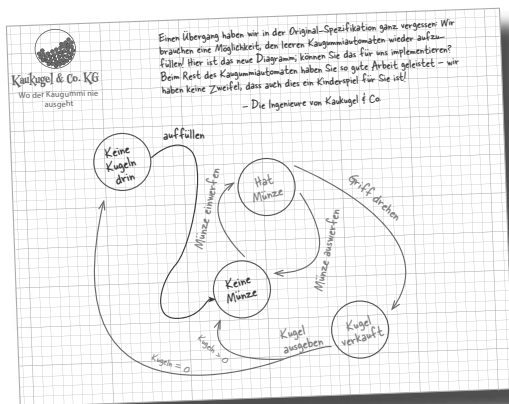


Restaurant Objekthausen und Pfannkuchenhaus Objekthausen fusionieren	316
Jupps und Wims Speisekarten-Implementierungen	318
Können wir die Iteration kapseln?	323
Darf ich vorstellen: das Iterator-Muster	325
Der RestaurantSpeisekarte einen Iterator hinzufügen	326
Blick auf den Entwurf	331
Mit java.util.Iterator sauber machen	333
Was bringt uns das?	335
Die Definition des Iterator-Musters	336
Eine einzige Verantwortlichkeit	339
Iteratoren und Collections	348
Iteratoren und Collections in Java 5	349
Und gerade als wir dachten, alles wäre in trockenen Tüchern ...	353
Die Definition des Composite-Musters	356
Mit dem Composite-Muster Speisekarten entwerfen	359
Die Komposita-Speisekarte implementieren	362
Ein Rückblick auf Iterator	368
Der Null-Iterator	372
Die Magie von Iteratoren und Komposita zusammen	374
Werkzeuge für Ihren Design-Werkzeugkasten	380
Lösungen zu den Aufgaben	381

# 10

## Die Zustände in Objekthausen

**Eine kaum bekannte Tatsache ist: Das Strategy- und das State-Muster sind Zwillinge, die bei der Geburt getrennt wurden.** Wie Sie schon wissen, hat das Strategy-Muster später ein supererfolgreiches Geschäft mit austauschbaren Algorithmen aufgebaut. Das State-Pattern hingegen hat einen anderen – vielleicht edelmütigeren – Weg eingeschlagen: Es hilft Objekten, ihr Verhalten mittels Veränderung ihres internen Zustands zu kontrollieren. Oft hört man es zu seiner Objekt-Klientel sagen: »Sprecht mir nach: Ich bin gut genug, ich bin klug genug, verdammt noch mal ...«



Wie implementieren wir einen Zustand?	387
Einführungskurs »Zustandsautomaten«	388
Der Code für einen ersten Zustandsautomaten	390
Das musste ja kommen ... eine Änderungsanfrage!	394
ZUSTÄNDE wie bei Hempels unterm Sofa ...	396
Definition des Zustands-Interface und der Zustandsklassen	399
Implementierung unserer Zustandsklassen	401
Umbau des Kaugummiautomaten	402
Die Definition des State-Musters	410
State und Strategy	411
Gesundheitscheck: Stimmt alles	417
Das hätten wir beinahe vergessen!	420
Werkzeuge für Ihren Design-Werkzeugkasten	423
Lösungen zu den Übungen	424



## Das Proxy-Muster

# 11

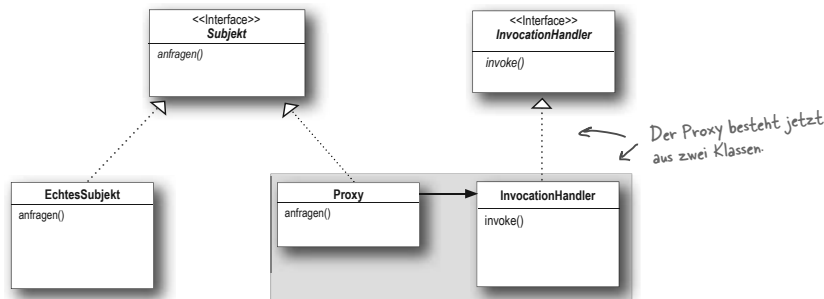
### Den Zugriff auf Objekte kontrollieren

#### Haben Sie schon mal »good cop – bad cop« gespielt?

Sie sind der gute Polizist und helfen den Menschen nett und freundlich. Aber Sie möchten einfach nicht *jedem* zu Diensten sein, und deshalb haben Sie den bösen Polizisten, der den *Zugang zu Ihnen kontrolliert*. Genau das tun Proxys: Sie kontrollieren und steuern den Zugang zu etwas anderem. Wie Sie sehen werden, können Proxys sich auf ganz unterschiedliche Art und Weise vor ihre zugehörigen Objekte stellen. Proxys haben schon komplette Methodenaufrufe über das Internet für ihre Objekte durchgeführt; manchmal sind sie aber auch nur geduldige Stellvertreter für ziemlich faule Objekte.



Kaugummiautomaten überwachen	430
Die Rolle des »Remote-Proxy«	434
Einführungskurs »Remote-Methoden«	437
Ein Remote-Proxy für den Kaugummiautomaten	450
Hinter den Kulissen	458
Die Definition des Proxy-Musters	460
Der virtuelle Proxy	462
Entwurf des virtuellen Proxy für das CD-Cover	464
Was haben wir gemacht?	470
Der Proxy aus der Java-API	474
Kurzdrama: Objektschutz	478
Erzeugung eines dynamischen Proxy	479
Der Proxy-Zoo	488
Werkzeuge für Ihren Design-Werkzeugkasten	491
Lösungen zu den Aufgaben	492



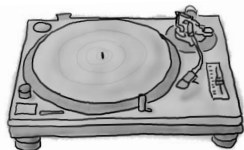
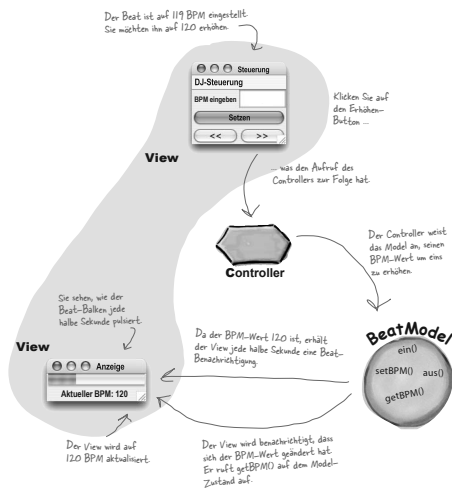
# Zusammengesetzte Muster

# 12

## Muster von Mustern

### Wer hätte je gedacht, dass Entwurfsmuster

**zusammenarbeiten könnten?** Sie sind ja schon Zeuge der erbitterten Auseinandersetzungen am Kamin geworden (und dabei haben Sie noch nicht mal die Seiten mit den Kämpfen auf Leben und Tod gesehen, die wir auf Druck des Verlags wieder herausnehmen mussten). Mal ehrlich, hätten Sie geglaubt, dass Muster gut miteinander auskommen können? Also, ob Sie es glauben oder nicht: Einige der leistungsfähigsten OO-Designs setzen mehrere Muster gemeinsam ein. Machen Sie sich also bereit für Ihren nächsten Muster-Qualifikationslevel, denn jetzt stehen zusammengesetzte Muster auf dem Plan.



Mustergültige Zusammenarbeit	500
Ein Wiedersehen mit den Enten	501
Einen Adapter hinzufügen	504
Einen Decorator hinzufügen	506
Eine Fabrik hinzufügen	508
Jetzt noch das Composite-Muster und ein Iterator	513
Zum Schluss noch ein Observer	516
Was wir insgesamt gemacht haben ...	523
Aus der VogelEntenperspektive: das Klassendiagramm	524
Das Model-View-Controller-Lied	526
Entwurfsmuster und MVC	528
MVC, durch die Musterbrille betrachtet	532
Mit MVC den Takt angeben ...	534
Das Model	537
Der View	539
Der Controller	542
Strategy intensiv	545
Anpassung des Modells	546
Jetzt sind wir bereit für einen HerzController	547
Das war's! Jetzt brauchen wir noch Testcode ...	547
MVC und das Web	549
Muster und Model 2	557
Werkzeuge für Ihren Design-Werkzeugkasten	560
Lösungen zu den Übungen	561

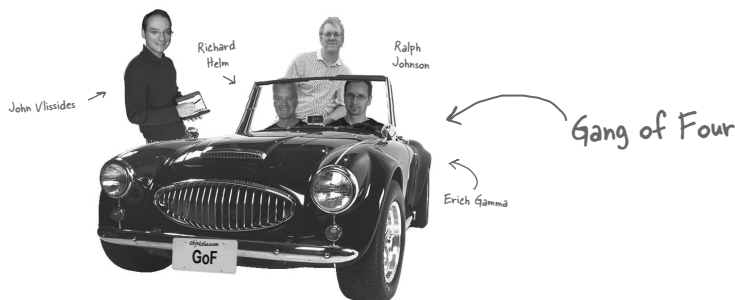
## Besser leben mit Mustern

# 13 Entwurfsmuster in der realen Welt

**Aaah, jetzt sind Sie bereit für eine strahlende neue Welt voller Entwurfsmuster!** Aber bevor Sie all die tollen Chancen nutzen, die sich Ihnen jetzt bieten, müssen wir noch ein paar Einzelheiten besprechen, die Sie in der realen Welt beachten müssen – ja, ein bisschen komplizierter als hier in Objekthausen wird es schon! Schauen Sie mal auf die nächste Seite: Dort haben wir einen schönen Leitfaden, der Ihnen die Eingewöhnung erleichtern wird.

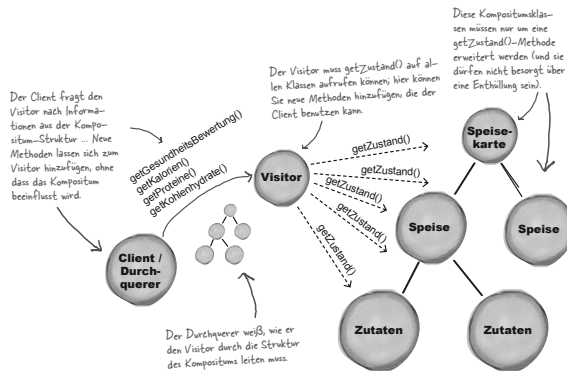


Der Objekthausener Muster-Leitfaden	578
Definition eines Entwurfsmusters	579
Die Entwurfsmusterdefinition näher betrachtet	581
Möge die Macht mit Ihnen sein!	582
Musterkataloge	583
Wie Muster auf die Welt kommen	586
So, Sie möchten also selbst Entwurfsmuster schreiben?	587
Ordnung in Entwurfsmuster bringen	589
In Mustern denken	594
Ihr Denken wird mustergültig	597
Vergessen Sie nicht die Macht des gemeinsamen Vokabulars	599
Die fünf besten Wege zu einem gemeinsamen Vokabular	600
Eine Fahrt durch Objekthausen mit der Gang of Four	601
Ihre Reise hat gerade erst begonnen ...	602
Der Musterzoo	604
Mit Antimustern gegen die Schlechtigkeit	606
Werkzeuge für Ihren Design-Werkzeugkasten	608
Abschied von Objekthausen ...	609



# 14 Anhang: Übrig gebliebene Muster

**Nicht jeder kann eine Berühmtheit sein.** In den letzten zehn Jahren hat sich eine Menge geändert. Seit die 1. Auflage von *Entwurfsmuster: Elemente wiederverwendbarer objektorientierter Software* erschienen ist, haben Entwickler diese Muster tausende von Malen angewendet. Die Muster, die in diesem Anhang zusammengefasst sind, sind vollwertige, ausgewiesene, offizielle GoF-Muster, sie werden nur nicht so oft verwendet wie die Muster, mit denen wir uns bis jetzt beschäftigt haben. Dennoch werden diese Muster mit vollem Recht als großartige Muster betrachtet, und wenn Sie in einer Situation sind, die danach verlangt, können Sie sie mit erhobenem Haupt anwenden. In diesem Anhang möchten wir Ihnen eine ungefähre Vorstellung davon vermitteln, worum es bei diesen Mustern geht.



Das Bridge-Muster	612
Das Builder-Muster	614
Die Chain of Responsibility	616
Das Flyweight-Muster	618
Das Interpreter-Muster	620
Das Mediator-Muster	622
Das Memento-Muster	624
Das Prototype-Muster	626
Das Visitor-Muster	628

