Entwurfsmuster von Kopf bis Fuß

Eric Freeman
Elisabeth Robson

mit
Kathy Sierra
Bert Bates

Deutsche Übersetzung von
Jørgen W. Lang

O'REILLY®

Der Inhalt (im Überblick)

Inhalt (jetzt ausführlich)

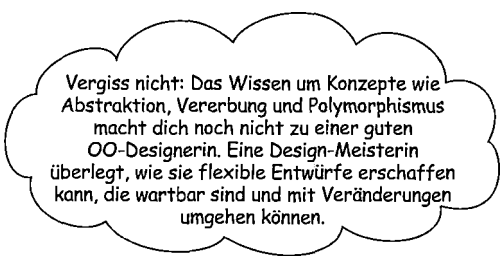
Einführung

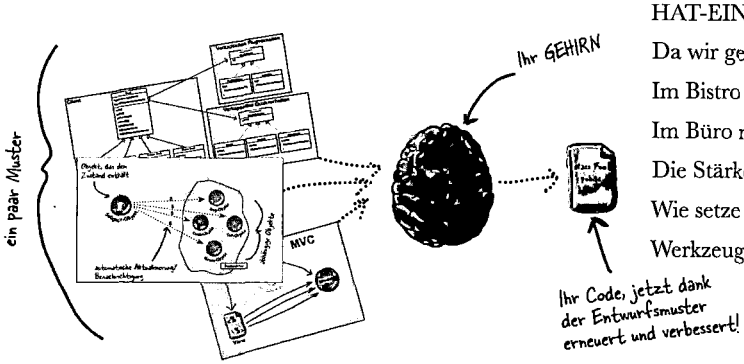
Ihr mustergültiges Gehirn. Sie versuchen, etwas zu lernen, und Ihr Hirn tut sein Bestes, damit das Gelernte nicht hängen bleibt. Es denkt nämlich: »Wir sollten lieber ordentlich Platz für wichtigere Dinge lassen, z. B. für das Wissen, welche Tiere einem gefährlich werden könnten, oder dass es eine ganz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

Einführung in Entwurfsmuster

1 Willkommen bei den Entwurfsmustern

Irgendjemand hat Ihr Problem schon gelöst. In diesem Kapitel lernen Sie, warum (und wie) Sie die Weisheit und die Lehren anderer Entwickler nutzen können, die die gleichen Designprobleme bereits hatten und die Reise überlebt haben. Bevor dieses Kapitel zu Ende ist, kümmern wir uns um die Verwendung und die Vorteile der Entwurfsmuster, sehen uns ein paar grundsätzliche objektorientierte (OO-)Designprinzipien an und gehen mit Ihnen zusammen ein Beispiel für die Funktionsweise von Entwurfsmustern durch. Die beste Möglichkeit, die Muster zu verwenden, ist, sie *in Ihr Gehirn zu laden* und dann die Stellen in Ihren Designs und bestehenden Programmen zu *erkennen*, an denen der Einsatz sinnvoll ist. Im Gegensatz zur Codewiederverwendung können Sie mit Entwurfsmustern die *Erfahrung* anderer Menschen wiederverwenden.

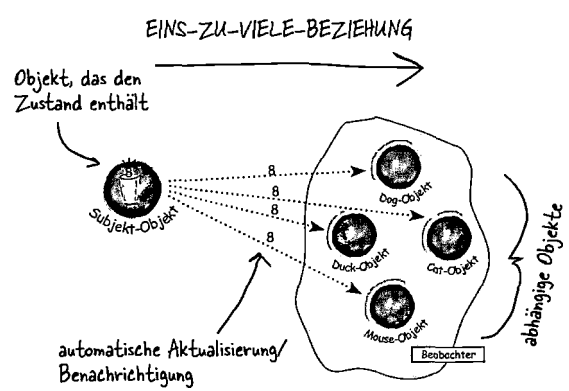




Das Observer-Muster

2 Ihre Objekte auf dem Laufenden halten

Sie wollen interessante Ereignisse doch nicht verpassen, oder? Es gibt ein Muster, das unsere Objekte auf dem Laufenden hält, wenn etwas für sie Wichtiges passiert, und zwar eins der am häufigsten verwendeten und nützlichsten Entwurfsmuster überhaupt: das Observer-Muster. In diesem Kapitel sehen wir uns alle möglichen interessanten Eigenschaften dieses Musters an, wie *Eins-zu-viele-Beziehungen* und *lose Kopplungen*. Mit dem Observer-Muster im Gepäck sind Sie der Star jeder Muster-Party!



Das Decorator-Muster

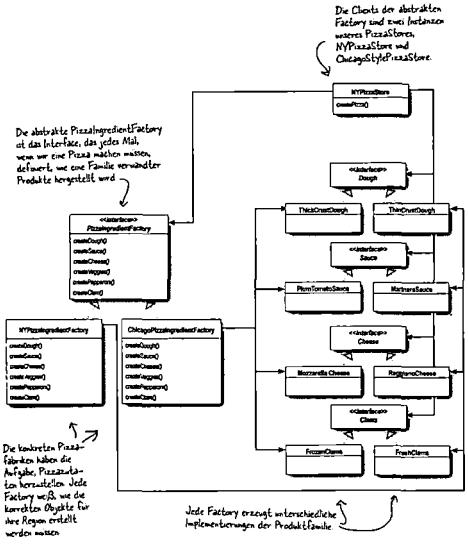
3 Objekte dekorieren

Nennen wir dieses Kapitel einfach »Vererbst du noch, oder designst du schon?«. Hier werfen wir einen weiteren Blick auf das typische Überstrapazieren von Vererbung. Sie werden lernen, wie Sie Ihre Klassen zur Laufzeit mit einer Form der Objektkomposition dekorieren können. Warum? Sobald Sie die Dekoration beherrschen, können Sie Ihren Objekten (oder denen anderer Leute) neue Verantwortung geben, *ohne hierfür den Code der zugrunde liegenden Klassen ändern zu müssen.*

Das Factory-Muster

4 In der OO-Bäckerei ...

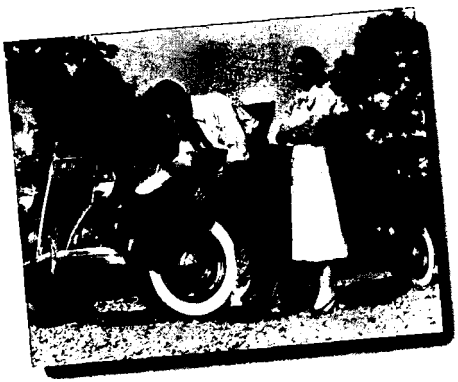
Machen Sie sich bereit, ein paar lose gekoppelte OO-Entwürfe zu backen. Zur Erstellung von Objekten gehört mehr, als einfach den **new**-Operator einzusetzen. Sie werden lernen, dass Instanziierung nicht in der Öffentlichkeit durchgeführt werden sollte und oft zu *Kopplungsproblemen* führen kann. Und *das* wollen wir nun wirklich nicht, oder? Finden Sie heraus, wie das Factory-Muster Sie vor peinlichen Abhängigkeiten bewahren können.

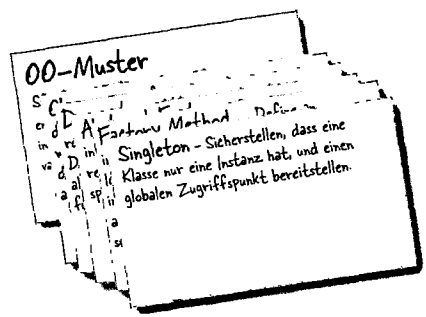


Das Singleton-Muster

5 Einmalige Objekte

Unser nächster Halt ist das Singleton-Muster, unsere Fahrkarte für die Erstellung einmaliger Objekte, von denen es immer nur eine Instanz gibt. Es wird Sie freuen, zu erfahren, dass das Singleton-Muster, bezogen auf sein Klassendiagramm, das einfachste Muster von allen ist. Tatsächlich enthält es nur eine einzige Klasse! Aber machen Sie es sich nicht zu bequem. Trotz des einfachen Klassendesigns müssen wir für seine Implementierung einige tiefgehende objektorientierte Überlegungen anstellen. Also, setzen Sie Ihre Denkmütze auf, und los geht's.



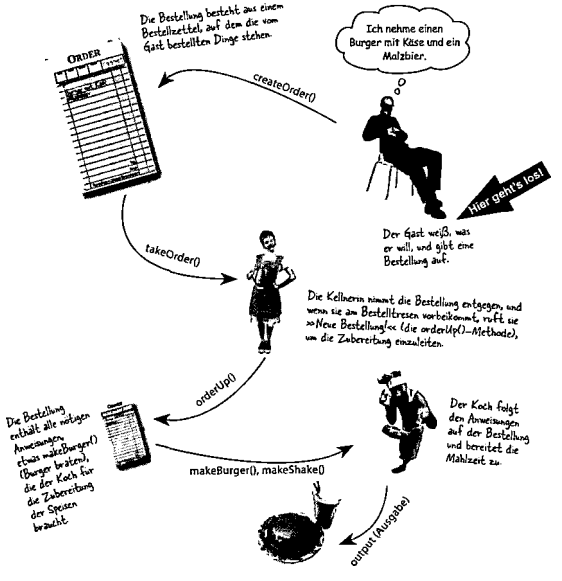


Das Command-Muster

6 Aufrufe verkapseln

In diesem Kapitel heben wir die Verkapselung auf ein ganz neues Niveau: Wir werden Methodenaufrufe verkapseln.

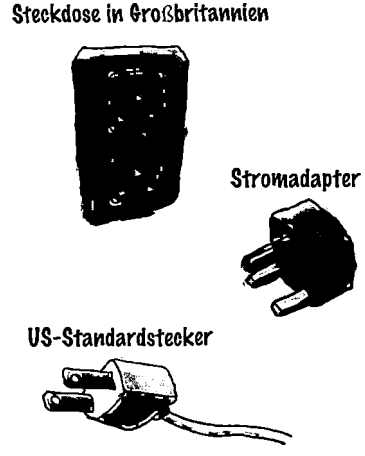
Ja, richtig gehört: Methodenaufrufe. Damit können wir Teile von Berechnungen »einfrieren«, wodurch sich das aufrufende Objekt nicht um die Details der Berechnung kümmern muss. Es nutzt einfach die eingefrorene Methode für die Erfüllung seiner Aufgabe. Mit diesen verkapselten Methodenaufrufen sind aber noch ganz andere schlaue Dinge möglich. Wir können sie beispielsweise zur Protokollierung nutzen oder sie wiederverwenden, um eine »Rückgängig«-Funktionalität zu implementieren.

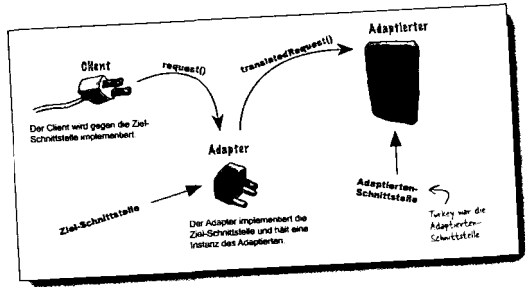


Die Adapter- und Facade-Muster

7 Anpassungsfähigkeit beweisen

In diesem Kapitel versuchen wir so unmögliche Dinge wie die Quadratur des Kreises. Klingt ausgeschlossen? Aber nicht mit Entwurfsmustern. Erinnern Sie sich noch an das Decorator-Muster? Wir haben Objekte verpackt, um sie mit Verantwortlichkeiten zu versehen. Diesmal **verpacken wir Objekte**, damit ihre Schnittstellen wie etwas aussehen, das sie nicht sind. So können wir Designs, die bestimmte Schnittstellen erwarten, an Klassen anpassen, die eine andere Schnittstelle implementieren. Und das ist noch nicht alles. Wenn wir schon dabei sind, sehen wir uns gleich noch ein anderes Muster an, das Objekte verpackt, um ihre Schnittstelle zu vereinfachen.

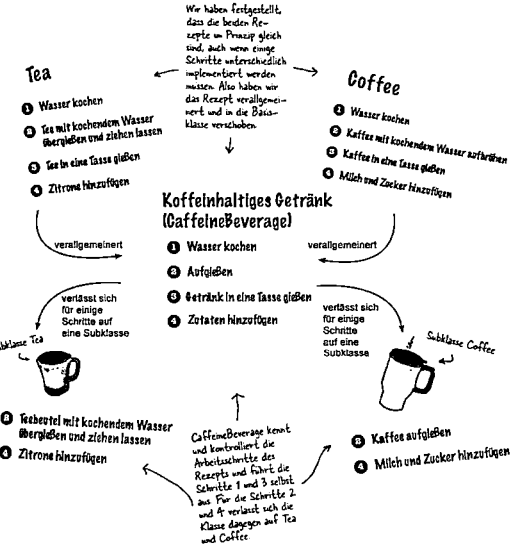




Das Template Method-Muster

8 Algorithmen verkapseln

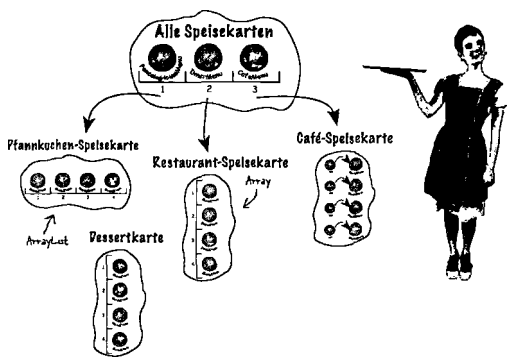
Wir sind auf dem Verkapselungstrip. Was haben wir schon alles verkapselt? Objekterstellung, Methodenaufrufe, komplexe Schnittstellen, Enten, Pizzas – was kommt wohl als Nächstes? In diesem Kapitel gehen wir der Verkapselung von Algorithmenteilen auf den Grund, damit Subklassen sich bei Bedarf jederzeit direkt in eine Berechnung einklinken können. Außerdem lernen wir ein Entwurfsprinzip kennen, das von Hollywood inspiriert ist. Na dann mal los ...



Die Iterator- und Composite-Muster

9 Erfolgreiche Collections

Es gibt viele Möglichkeiten, Objekte in einer Collection zu speichern. Zum Beispiel in einem Array, einem Stack, einer Liste oder einer HashMap – Sie haben die Wahl. Dabei hat jede Form ihre Vor- und Nachteile. Irgendwann werden Ihre Clients allerdings über diese Objekte iterieren wollen. Und wollen Sie ihnen dann Ihre Implementierung offenbaren? Hoffentlich nicht! Das wäre einfach nicht professionell. Aber keine Sorge. Ihre Karriere ist nicht gefährdet. In diesem Kapitel werden Sie sehen, wie Ihre Clients über Ihre Objekte iterieren können, ohne zu wissen, wie sie gespeichert sind. Außerdem lernen Sie, wie man Super Collections von Objekten erstellt, die mit einem einzigen Satz einige eindrucksvolle Datenstrukturen überspringen können. Und als wäre das noch nicht genug, werden Sie auch noch das eine oder andere über Objektverantwortlichkeit lernen.

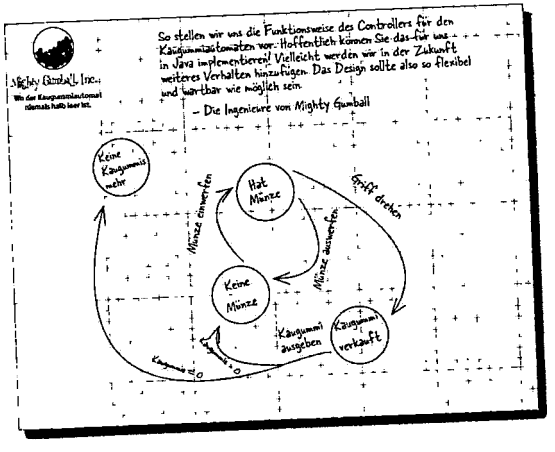


Das State-Muster

10 Der (Zu-)Stand der Dinge

Eine kaum bekannte Tatsache: Die Strategy- und das State-Muster sind Zwillinge, die bei der Geburt getrennt wurden.

Vielleicht denken Sie, dass beide ein ähnliches Leben führen. In Wirklichkeit hat Strategy jedoch ein unglaublich erfolgreiches Unternehmen rund um austauschbare Algorithmen aufgebaut, während State den vermutlich edleren Weg gewählt hat. Es hilft anderen Objekten, ihr Verhalten zu kontrollieren, indem es ihren inneren Zustand verändert. So unterschiedlich die Wege der beiden auch scheinen – hinter den Kulissen ist ihr Design fast identisch. Wie das sein kann und worum es beim State-Muster wirklich geht, werden wir herausfinden. Am Ende des Kapitels sehen wir uns dann an, welche Beziehung beide Muster tatsächlich zueinander haben.

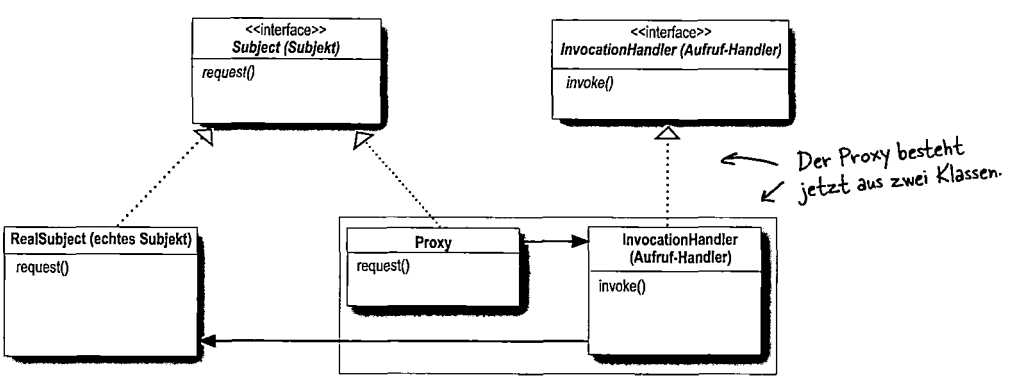


Das Proxy-Muster

11 Objektzugriff kontrollieren

Haben Sie schon einmal »guter Bulle, böser Bulle« gespielt?

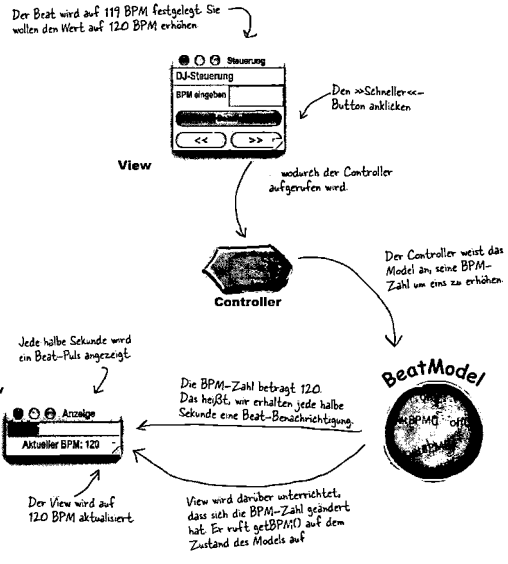
Sie sind der gute Polizist und stellen alle Ihre Dienste auf eine nette und freundliche Weise bereit. Wenn Sie aber nicht wollen, dass jeder ungefragt Ihre Dienste nutzt, übernimmt der böse Polizist die Zugangskontrolle für Sie. Denn genau das tun Proxies (»Stellvertreter«): Sie kontrollieren und verwalten den Zugriff. Wie Sie sehen werden, gibt es viele Möglichkeiten, Proxies als Vertreter für andere Objekte zu nutzen. Proxies sind dafür bekannt, dass sie für die von ihnen vertretenen Objekte komplette Methodenaufrufe über das Internet abwickeln. Außerdem nehmen sie bekanntermaßen den Platz einiger ziemlich fauler Objekte ein.

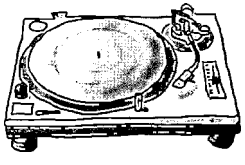


Zusammengesetzte Muster

12 Muster von Mustern

Wer hätte gedacht, dass Muster auch zusammenarbeiten können? Sie haben bereits die erbitterten Kamingespräche miterlebt (und dabei haben wir Ihnen die Seiten mit dem »Pattern Death Match« noch gar nicht gezeigt, die wir auf Druck des Verlegers wieder entfernen mussten). Wer hätte gedacht, dass Muster eigentlich sogar recht gut miteinander auskommen können? Ob Sie's glauben oder nicht – einige der mächtigsten OO-Entwürfe verwenden Kombinationen mehrerer Muster. Machen Sie sich bereit für die nächste Stufe Ihrer Entwurfsmuster-Fähigkeiten: zusammengesetzte Muster (»Compound-Muster«)

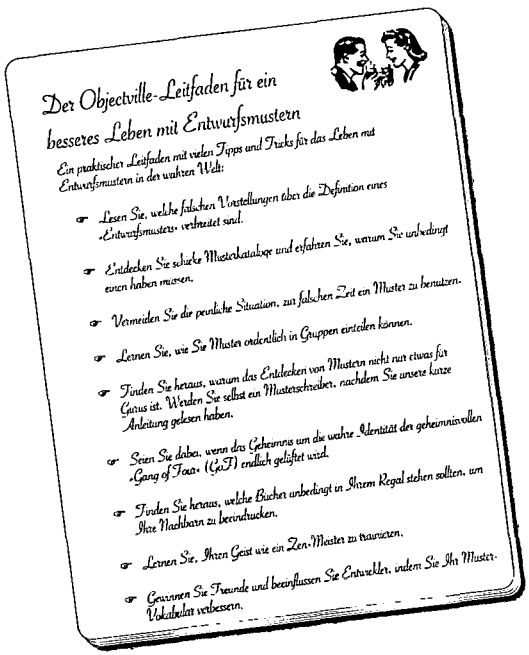


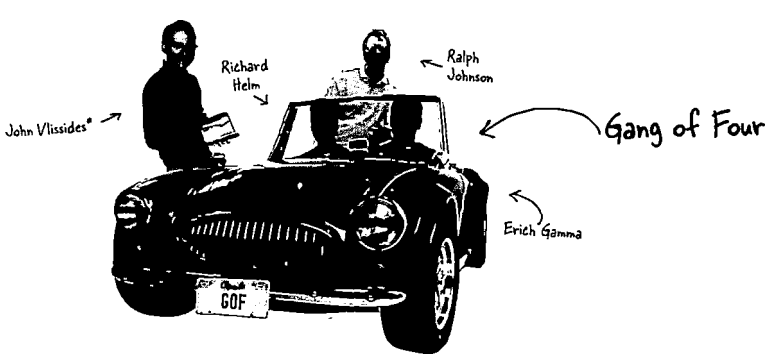


Schöner leben mit Mustern

13 Muster in der wahren Welt

Und damit sind Sie bereit für eine strahlende neue Welt voller Entwurfsmuster. Aber bevor Sie all die neuen Chancen nutzen, müssen wir uns noch um ein paar Details kümmern, die Ihnen draußen in der wahren Welt begegnen können, wo die Dinge etwas komplexer sind als hier in Objectville. Auf den folgenden Seiten haben wir für Sie einen kleinen Reiseführer (oder Leitfaden) vorbereitet, der Sie beim Übergang begleiten wird ...





14 Anhang: Übrig gebliebene Muster

Nicht jeder kann der Beliebteste sein. Seit der Erstveröffentlichung von *Design Patterns: Elements of Reusable Object-Oriented Software* hat sich viel verändert. Entwickler haben diese Muster tausendfach verwendet. Die in diesem Anhang vorgestellten Muster sind Vollmitglieder der offiziellen GoF-Musterfamilie, werden aber nicht so oft genutzt wie die bisher gezeigten. Trotzdem sind sie auf ihre eigene Art großartig, und wenn die Situation es erfordert, können Sie sie einsetzen, ohne sich dafür zu schämen. In diesem Anhang wollen wir Ihnen einen Überblick darüber geben, worum es bei diesen Mustern geht.