Der Inhalt (in der Übersicht)

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?

1 Willkommen bei den Entwurfsmustern

Irgendjemand hat Ihre Probleme bereits gelöst. In diesem Kapitel lernen Sie, warum (und wie) Sie die Erfahrungen und Lektionen verwerten 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*. Anstelle von *Code*-Wiederverwendung bieten Ihnen Muster *Erfahrungs*-Wiederverwendung.

Das Observer-Muster

2 Ihre Objekte auf dem Laufenden halten

Verpassen Sie es 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.

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 mithilfe 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 zugrunde liegenden Klasse ändern zu müssen.

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 Instantiierung 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.

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 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 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. Da wir gerade dabei sind, werden wir uns noch ein weiteres Muster ansehen, das Objekte umhüllt, um ihre Schnittstelle zu vereinfachen.

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.

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 Objekt 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 beeindruckend Datenstrukturen überspringen können. Und wenn Ihnen das immer noch nicht ausreicht, werden Sie außerdem ein oder zwei Dinge über Objektverantwortlichkeit lernen.

Das State-Muster

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 ...«

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.

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.

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.

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.