

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>	81
4 Backen in OO-Qualität: <i>das Factory-Muster</i>	111
5 Ein einzigartiges Objekt: <i>das Singleton-Muster</i>	171
6 Aufrufe einkapseln: <i>das Command-Muster</i>	193
7 Anpassungsfähigkeit beweisen: <i>das Adapter- und das Facade-Muster</i>	243
8 Algorithmen einkapseln: <i>das Template Methode-Muster</i>	283
9 Erfolgreiche Kollektionen: <i>das Iterator- und das Composite-Muster</i>	323
10 Die Zustände in Objekthäusen: <i>das State-Muster</i>	393
11 Den Zugriff auf Objekte kontrollieren: <i>das Proxy-Muster</i>	437
12 Muster von Mustern: <i>zusammengesetzte Muster</i>	505
13 Entwurfsmuster in der realen Welt: <i>besser leben mit Mustern</i>	583
14 Anhang: <i>übrig gebliebene Muster</i>	617

Der Inhalt (jetzt ausführlich)

Einführung

Ihr mustergütiges 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 untetan	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 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.

Alles begann mit einer simplen SimEnte-Anwendung	2
Aber jetzt müssen die Enten fliegen lernen	3
Aber etwas ging schrecklich schief	4
Eike denkt über Vererbung nach	5
Und wie wäre es mit einem Interface?	6
Was würden Sie tun, wenn Sie an Eikes Stelle wären?	7
Die eine Konstante bei der Software-Entwicklung	8
Das Problem einkreisen	9
Das, was veränderlich ist, von dem trennen, was gleich bleibt	10
Entenverhalten entwerfen	11
Das Entenverhalten implementieren	13
Die Entenverhalten integrieren	15
Integration fortgesetzt ...	16
Verhalten dynamisch setzen	20
Noch mal im Ganzen: Gekapseltes Verhalten	22
HAT-EIN kann IST-EIN überlegen sein	23
Da wir gerade von Entwurfsmustern sprechen ...	24
Im Bistro an der Ecke aufgeschnappt ...	26
Im Büro nebenan aufgeschnappt ...	27
Die Macht eines gemeinsamen Mustervokabulars	28
Wie verwende ich Entwurfsmuster?	29
Werkzeuge für Ihren Design-Werkzeugkasten	32

Das Observer-Muster

Ihre Objekte auf dem Laufenden halten

2

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.

Die Wetterstation-Anwendung im Überblick	39
Gestatten: das Observer-Muster	44
Herausgeber + Abonnenten = Observer-Muster	45
Die Definition des Observer-Musters	51
Die Definition des Observer-Musters: das Klassendiagramm	52
Die Macht der losen Kopplung	53
Die Wetterstation entwerfen	56
Die Wetterstation implementieren	57
Die Wetterstation in Betrieb nehmen	60
Javas eingebautes Observer-Muster verwenden	64
Wie Javas eingebautes Observer-Muster funktioniert	65
Die Wetterstation mit der eingebauten Unterstützung überarbeiten	67
Die dunkle Seite von <code>java.util.Observable</code>	71
Andere Orte im JDK, an denen Sie auf das Observer-Muster stoßen	72
Werkzeuge für Ihren Design-Werkzeugkasten	75

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.

Willkommen bei Sternback-Kaffee	82
Das Offen/Geschlossen-Prinzip	88
Dürfen wir vorstellen: das Decorator-Muster!	90
Ein Getränk mit Dekorierern aufbauen	91
Die Definition des Decorator-Musters	93
Getränke dekorieren	94
Den Sternback-Code schreiben	97
Dekorierer aus der Praxis: Java I/O	102
Die java.io-Klassen dekorieren	103
Einen eigenen I/O-Dekorierer schreiben	104
Werkzeuge für Ihren Design-Werkzeugkasten	107

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.

Die Aspekte identifizieren, die veränderlich sind	114
Die Objekt-Erstellung kapseln	116
Eine einfache Pizzafabrik erstellen	117
Die Definition der einfachen Fabrik	119
Ein Framework für die Pizzeria	122
Die Unterklassen entscheiden lassen	123
Eine Fabrikmethode deklarieren	127
Jetzt ist es endlich Zeit, dem Factory Method-Muster zu begegnen	133
Eine andere Perspektive: parallele Klassenhierarchien	134
Die Definition des Factory Method-Musters	136
Eine sehr abhängige Pizzeria	139
Ein Blick auf Objekt-Abhängigkeiten	140
Das Prinzip der Umkehrung der Abhängigkeiten	141
Das Prinzip anwenden	142
Stellen Sie Ihr Denken auf den Kopf	144
Ein paar Richtlinien, die Ihnen bei der Befolgung des Musters helfen	145
Inzwischen in der Pizzeria	146
Zutatenfamilien	147
Was wir gemacht haben	155
Die Definition des Abstract Factory-Musters	158
Factory Method und Abstract Factory im Vergleich	162
Werkzeuge für Ihren Design-Werkzeugkasten	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	173
Die klassische Implementierung des Singleton-Musters sezieren	175
Die Schokoladenfabrik	177
Definition des Singleton-Musters	179
<small>Kön</small> Hausen , wir haben ein Problem ...	180
Mit Multithreading klarkommen	182
Können wir das Multithreading verbessern?	183
Inzwischen in der Schokoladenfabrik ...	185
Werkzeuge für Ihren	
Design-Werkzeugkasten	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.

Kostenlose Hardware! Sehen wir uns mal diese Fernsteuerung an	195
Werfen wir einen Blick auf die Klassen der Hersteller	196
Inzwischen im Restaurant	199
Rollen und Verantwortlichkeiten im Restaurant Objekthausen	201
Vom Restaurant zum Command-Muster	203
Unser erstes Befehl-Objekt	205
Die Definition des Command-Musters	208
Den Fernsteuerungsplätzen Befehle zuweisen	211
Die Fernbedienung implementieren	212
Die Befehle implementieren	213
Die Fernsteuerung in Gang setzen	214
Zeit, diese Dokumentation zu schreiben	217
Einen Status verwenden, um Rückgängig zu implementieren	222
Jede Fernsteuerung braucht einen Party-Modus!	226
Einen Makro-Befehl verwenden	227
Das Command-Muster erfordert eine Menge an Klassen	230
Die Fernsteuerung mit Lambda-Ausdrücken vereinfachen	231
Weitere Verwendungen des Command-Musters: Warteschlangen für Befehle	237
Weitere Verwendungen des Command-Musters: Anfragen protokollieren	238
Werkzeuge für Ihren Design-Werkzeugkasten	239

7

Die Adapter- und Façade-Muster

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.

Adapter, wo wir nur hinschauen	244
Objektorientierte Adapter	245
Wenn es quakt wie eine Ente und watschelt wie eine Ente, muss könnte es eine Ente ein Truthahn sein, der mit einem Ente-Adapter eingepackt ist.	246
Den Adapter testen	248
Das Adapter-Muster erklärt	249
Der Client verwendet die Schnittstelle folgendermaßen:	249
Die Definition des Adapter-Musters	251
Objekt- und Klassen-Adapter	252
Adapter aus dem wirklichen Leben	256
Einen Enumerator an einen Iterator anpassen	257
Gemütliches Heimkino	263
Beleuchtung, Kamera, Fassade	266
Die Heimkino-Fassade aufbauen	269
Die Definition des Façade-Musters	272
Das Prinzip der Verschwiegenheit	273
Wie man sich KEINE Freunde macht	274
Das Façade-Muster und das Prinzip der Verschwiegenheit	277
Werkzeuge für Ihren Design-Werkzeugkasten	278

8

Das Template Method-Muster

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.

Zeit für noch etwas Koffein	284
Ein paar Kaffee- und Tee-Klassen (in Java) zusammenröhren	285
Dürfte ich vielleicht Ihren Kaffee, Tee abstrahieren?	288
zubereitungsRezept() abstrahieren	290
Was also haben wir gemacht?	293
Dürfen wir vorstellen: das Template Method-Muster!	294
Was hat uns das Template Method-Muster gebracht?	296
Die Definition des Template Method-Musters	297
Haken wir uns bei einer Template-Methode ein ...	300
Den Hook verwenden	301
Das Hollywood-Prinzip und das Template Method-Muster	305
Template-Methoden im wirklichen Leben	307
Mit dem Template Method-Muster sortieren	308
Wir haben ein paar Enten, die sortiert werden müssen	309
Enten mit Enten vergleichen	310
Sortieren wir also ein paar Enten	311
Der Aufbau einer Enten-Sortiermaschine	312
Swinging mit Frames	314
Werkzeuge für Ihren Design-Werkzeugkasten	319

9

Die Iterator- und Composite-Muster

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

Nachricht des Tages: Restaurant Objekthausen und Pfannkuchenhaus Objekthausen fusionieren	324
Sehen wir uns die Speisen an	325
Können wir die Iteration kapseln?	332
Darf ich vorstellen: das Iterator-Muster	334
Der RestaurantSpeisekarte einen Iterator hinzufügen	335
Was wir gemacht haben	339
Mit java.util.Iterator sauber machen	342
Die Definition des Iterator-Musters	345
Eine einzige Verantwortlichkeit	348
Werfen wir einen Blick auf die Speisekarte des Cafés	351
Was wir gemacht haben?	355
Iteratoren und Collections	357
Ist die Kellnerin bereit für den Ansturm der Gäste?	359
Die Definition des Composite-Musters	364
Mit dem Composite-Muster Speisekarten entwerfen	367
Die SpeisekartenKomponente implementieren	368
Die Komposita-Speisekarte implementieren	370
Ein Rückblick auf Iterator	376
Der KompositumIterator	377
Der Null-Iterator	380
Die Magie von Iteratoren und Komposita zusammen	382
Werkzeuge für Ihren Design-Werkzeugkasten	388

10

Das State-Muster

Die Zustände in Objekthäusen

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 ...«

Java bringt die Kugel ins Rollen	394
Einführungskurs »Zustandsautomaten«	396
Den Code schreiben	398
Das musste ja kommen ... eine Änderungsanfrage!	402
ZUSTÄNDE wie bei Hempels unterm Sofa ...	404
Der neue Entwurf	406
Definition des Zustands-Interface und der Zustandsklassen	407
Implementierung unserer Zustandsklassen	409
Umbau des Kaugummiautomaten	410
Die Implementierung weiterer Zustände	412
Sehen wir uns mal an, was wir bis jetzt gemacht haben ...	415
Die Definition des State-Musters	418
Unser 1-von-10-Kaugummispiel ist noch nicht fertig	421
Demo für den Hauptgeschäftsführer von Kaukugel & Co. KG	423
Stimmt alles?	425
Kamingespräche	426
Das hätten wir beinahe vergessen!	428
Werkzeuge für Ihren Design-Werkzeugkasten	431

11

Das Proxy-Muster

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.

Der Überwachungscode	439
Die Rolle des »Remote-Proxy«	442
Einführungskurs »Remote-Methoden«	445
Zurück zu unserem Remote-Proxy für den Kaugummiautomaten	457
Die Definition des Proxy-Musters	467
Der virtuelle Proxy	469
Entwurf des virtuellen Proxy für das CD-Cover	471
Was haben wir gemacht?	477
Die Erstellung eines Schutz-Proxy mit dem Proxy aus der Java-API	481
Kurzdrama: Objektschutz	485
Erzeugung eines dynamischen Proxy	486
Der Proxy-Zoo	494
Werkzeuge für Ihren	
Design-Werkzeugkasten	497
Der Code für den CD-Cover-Viewer	501

Zusammengesetzte Muster

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	506
Ein Wiedersehen mit den Enten	507
Einen Adapter hinzufügen	510
Einen Decorator hinzufügen	512
Eine Fabrik hinzufügen	514
Jetzt noch das Composite-Muster und ein Iterator	519
Zum Schluss noch ein Observer	522
Was wir gemacht haben ...	529
Aus der VogelEntenperspektive: das Klassendiagramm	530
Der König der zusammengesetzten Muster	532
Wir stellen vor: Model-View-Controller	535
MVC, durch die Musterbrille betrachtet	538
Mit MVC den Takt angeben ...	540
Erstellung der Einzelteile	543
Implementierung des Views	546
Nun zum Controller	548
Alles zusammensetzen	550
Strategy intensiv	551
Anpassung des Models	552
MVC und das Web	555
Model 2: DJ am Handy	557
Model 2 im Test ...	561
Zum Ausprobieren:	562
Muster und Model 2	563
Werkzeuge für Ihren Design-Werkzeugkasten	566

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.

bjekthausener	584
Leitfaden für ein besseres Leben mit Mustern	584
Definition eines Entwurfsmusters	585
Die Entwurfsmusterdefinition näher betrachtet	587
Möge die Macht mit Ihnen sein!	588
So, Sie möchten also selbst Entwurfsmuster schreiben?	593
Ordnung in Entwurfsmuster bringen	595
In Mustern denken	600
Ihr Denken wird mustergültig	603
Vergessen Sie nicht die Macht des gemeinsamen Vokabulars	605
Eine Fahrt durch Objekthausen mit der Gang of Four	607
Ihre Reise hat gerade erst begonnen ...	608
Der Musterzoo	610
Mit Antimustern gegen die Schlechtigkeit	612
Werkzeuge für Ihren	
Design-Werkzeugkasten	614
Abschied von Objekthausen ...	615
Schön, dass Sie hier waren!	615

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.

Bridge-Muster	618
Builder-Muster	620
Chain of Responsibility-Muster	622
Flyweight-Muster	624
Interpreter-Muster	626
Mediator-Muster	628
Memento-Muster	630
Prototype-Muster	632
Visitor-Muster	634

