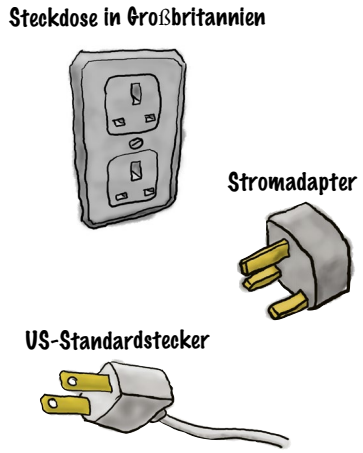
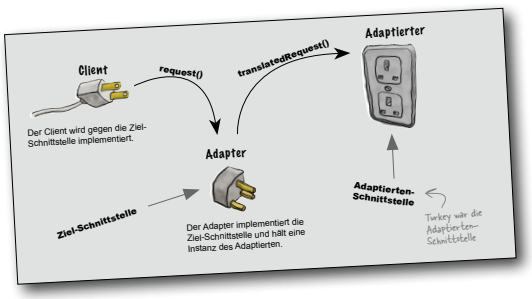
Die Adapter- und Facade-Muster

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.

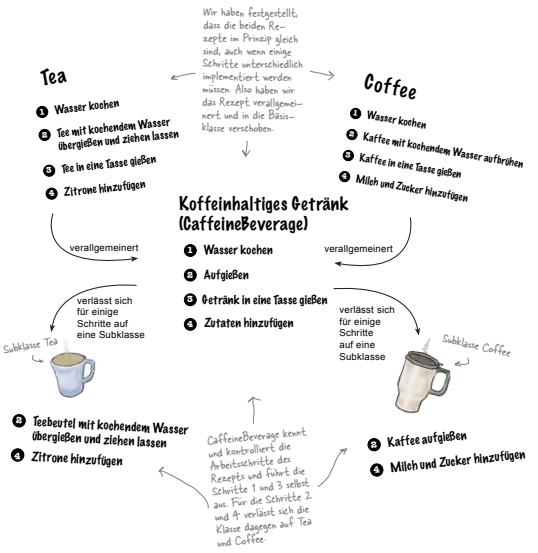




Das Template Method-Muster

8 Algorithmen verkapseln

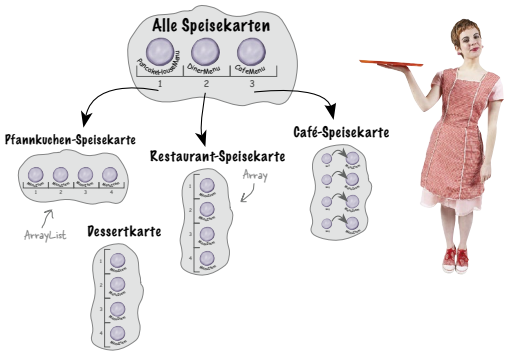
Wir sind auf dem Verkapselungstrip. Was haben wir schon alles verkapselt? Objekterstellung, Methodenaufrufe, komplexe Schnittstellen, Enten, Pizzas – was kommt wohl als Nächstes? In diesem Kapitel gehen wir der Verkapselung von Algorithmenteilen auf den Grund, damit Subklassen sich bei Bedarf jederzeit 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.

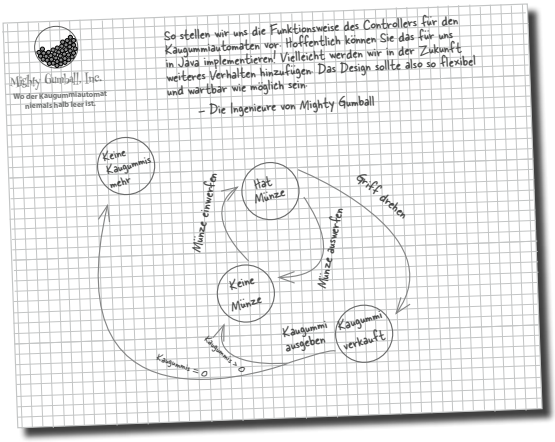


Das State-Muster

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.