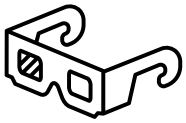
Was ist ein Entwurfsmuster?

Arbeitsauftrag



1. **Lesen** Sie den Einführungstext zu Entwurfsmustern
2. **Bearbeiten** Sie anschließend die **Aufgaben zum MVC-Pattern**. **Informieren** Sie sich dazu zunächst über das Pattern.

Christoper Alexander schreibt: „Jedes Muster beschreibt ein in unserer Umwelt beständig wiederkehrendes Problem und erläutert den Kern der Lösung für dieses Problem, so das Sie diese Lösung beliebig oft anwenden können, ohne sie jemals ein zweites Mal gleich auszuführen." Obwohl Alexander über Muster in Gebäuden und Städten spricht, trifft seine Definition auch für objektorientierte Entwurfsmuster zu. Zwar beschreiben wir unsere Lösungen mit Hilfe von Objekten und Schnittstellen statt mit Wänden und Türen, im Prinzip aber sind beide Arten von Mustern als Problemlösungen für bestimmte Situationen zu verstehen.

Allgemein betrachtet besitzt ein Muster vier grundlegende Elemente:

1. Der **Mustername** ist ein Stichwort, das wir benutzen, um ein Entwurfsproblem und seine Losungen und Auswirkungen mit ein oder zwei Worten zu benennen. Durch die Benennung von Mustern erweitern wir unser Entwurfsvokabular, so dass wir lernen auf einer höheren Abstraktionsebene zu entwerfen. Das neue Vokabular hilft uns, besser mit Kollegen über Muster und Entwürfe zu sprechen und sie zu dokumentieren. Es erleichtert uns, über Entwürfe nachzudenken und ihre Vor- und Nachteile anderen zu vermitteln. Gute Namen zu finden war eine der schwierigsten Aufgaben beim Entwickeln unseres Katalogs.

2. Der **Problemabschnitt** beschreibt, wann das Muster anzuwenden ist, welches Problem adressiert wird und was sein Kontext ist. Es beschreibt mögliche spezifische Entwurfsprobleme, beispielsweise wie Algorithmen als Objekte zu repräsentieren sind. Es kann auch Klassen- oder Objektstrukturen beschreiben, die symptomatisch für einen unflexiblen Entwurf sind. Mitunter wird eine solche Problembeschreibung eine Liste von Bedingungen aufführen, die erfüllt sein müssen, wenn die Anwendung des Musters sinnvoll sein soll.

3. Der **Lösungsabschnitt** beschreibt die Elemente, aus denen der Entwurf besteht, sowie ihre Beziehungen, Zuständigkeiten und Interaktionen. Die Lösung beschreibt weder einen bestimmten Entwurf noch eine konkrete Implementierung, sondern vielmehr eine Schablone, die in vielen verschiedenen Situationen angewendet werden kann. Ein Muster bietet eine abstrakte Beschreibung eines Entwurfsproblems und zeigt, wie eine allgemeine Anordnung von Elementen es löst. In unserem Fall stellen Klassen und Objekte die verwendeten Elemente dar.

4. Der **Konsequenzenabschnitt** beschreibt die Konsequenzen der Musteranwendung durch die Auflistung der Vor- und Nachteile des resultierenden Entwurfs. Obwohl die Konsequenzen oftmals unausgesprochen bleiben, wenn wir Entwurfsentscheidungen beschreiben, sind sie doch von zentraler Bedeutung für die Bewertung von Entwurfsalternativen und für das Verständnis der Vor- und Nachteile der Musteranwendung.

Die Konsequenzen der Musteranwendung für den Entwurf betreffen oft den Speicherplatzverbrauch und die Ausführungszeit. Sie können ebenfalls Sprach- und Implementierungsaspekte betreffen. Da der Wiederverwendbarkeit In objektorientierten Systemen oft eine wichtige Bedeutung zukommt, umfasst der Konsequenzabschnitt eines Musters auch seinen Einfluss auf die Flexibilität, Erweiterbarkeit und Portabilität des Systems. Die explizite Aufführung dieser Konsequenzen erleichtert es Ihnen, sie zu verstehen und auszuwerten.

Die Bestimmung dessen, was ein Muster ist und was nicht, hangt von der jeweiligen Perspektive ab. Was aus einer Perspektive als Muster erscheint, stellt aus einer anderen Perspektive betrachtet einen primitiven Baustein dar. In diesem Buch haben wir uns auf eine bestimmte Abstraktionsebene konzentriert. Entwurfsmuster befassen sich nicht mit Entwürfen wie verkettete Listen oder Hash-Tabellen, die als einzelne Klassen programmiert und wiederverwendet werden können. Sie stellen auch keine komplexen für einen Anwendungsbereich spezifische Entwürfe dar, die eine ganze Anwendung oder ein Subsystem realisieren. Die Entwurfsmuster In diesem Buch sind Beschreibungen zusammenarbeitender Objekte und Klassen, die maßgeschneidert sind, um ein allgemeines Entwurfsproblem in einem bestimmten Kontext zu lösen.

Ein Entwurfsmuster benennt, abstrahiert und Identifiziert die relevanten Aspekte einer allgemeinen Entwurfsstruktur. Diese Aspekte beschreiben, warum das Muster für die Entwicklung eines wiederverwendbaren objektorientierten Entwurfs nützlich ist. Das Entwurfsmuster Identifiziert die teilnehmenden Klassen und Objekte, die Rollen, die sie spielen, die Interaktionen zwischen den Rollen und die Ihnen zugeteilten Aufgaben. Jedes Entwurfsmuster konzentriert sich auf ein bestimmtes objektorientiertes Entwurfsproblem. Es beschreibt, wann es einsetzbar ist, ob es angesichts einschränkender Randbedingungen eingesetzt werden kann, und welche Konsequenzen sein Einsatz hat. Da wir schließlich unsere Entwurfsmuster implementieren müssen, veranschaulichen wir mögliche Implementierungen mittels Beispielcode In C++ und (mitunter) Smalltalk.

Die bekannten Implementierungen der Entwurfsmuster basieren auf handelsüblichen objektorientierten Programmiersprachen wie Smalltalk und C++, obwohl sie im Prinzip allgemeine objektorientierte Entwürfe darstellen. Wir haben eher prozedurale Sprachen wie Pascal, C oder Ada, oder eher dynamische objektorientierte Sprachen wie CLOS, Dylan oder Self haben, nicht zum Zuge kommen lassen. Wir haben Smalltalk und C++ aus pragmatischen Gründen gewählt: Unsere tagtägliche Erfahrung machen wir In diesen Sprachen, sie sind bekannt, und ihr Bekanntheitsgrad steigt weiter.

Die Wahl einer Programmiersprache ist wichtig, weil sie den eigenen Blickwinkel beeinflusst. Unsere Muster basieren auf Smalltalk- oder C++-Sprachmitteln. Diese Wahl legt fest, was leicht und was nicht so leicht implementiert werden kann. Wären wir von prozeduralen Sprachen ausgegangen, so hatten wir vielleicht Muster wie Vererbung, Kapselung und Polymorphie aufgenommen. Auf ähnliche Weise werden manche unserer Muster von den weniger weit verbreiteten objektorientierten Sprachen direkt unterstützt. So verfugt CLOS beispielsweise über Multimethoden, was den Wunsch nach einem Muster wie dem Besuchermuster (Seite 301) verringert. Die zahlreichen Unterschiede zwischen Smalltalk und C++ haben zur Folge, dass man manche Muster leichter in der einen als der anderen Sprache umsetzen kann (siehe zum Beispiel das Iteratormuster).

Quelle: Gamma, Erich, Helm, Richard, Johnson, Ralph & John Vlissides (1996): Entwurfsmuster – Elemente wiederverwendbarer objektorientierter Software, aus dem Amerikan. von Dirk Riehle, 1. Auflage, Bonn: Addiso-Wesley-Longmann, S. 3ff.

Entwurfsmuster in Smalltalk – MVC

Das MVC-Paradigma wird durch 3 Objekte umgesetzt. Das Model-Objekt stellt das Anwendungsobjekt dar, das View-Objekt seine Bildschirmpräsentation und das Controller-Objekt bestimmt die Möglichkeiten, mit denen die Benutzungsschnittstelle auf Benutzereingaben reagieren kann. In den Zeiten vor MVC tendierten Entwürfe von Benutzungsschnittstellen dazu, diese Objekte in einem einzigen Objekt zusammenzuführen. Das MVC-Pattern entkoppelte sie, die Flexibilität und Wiederverwendbarkeit zu erhöhen.

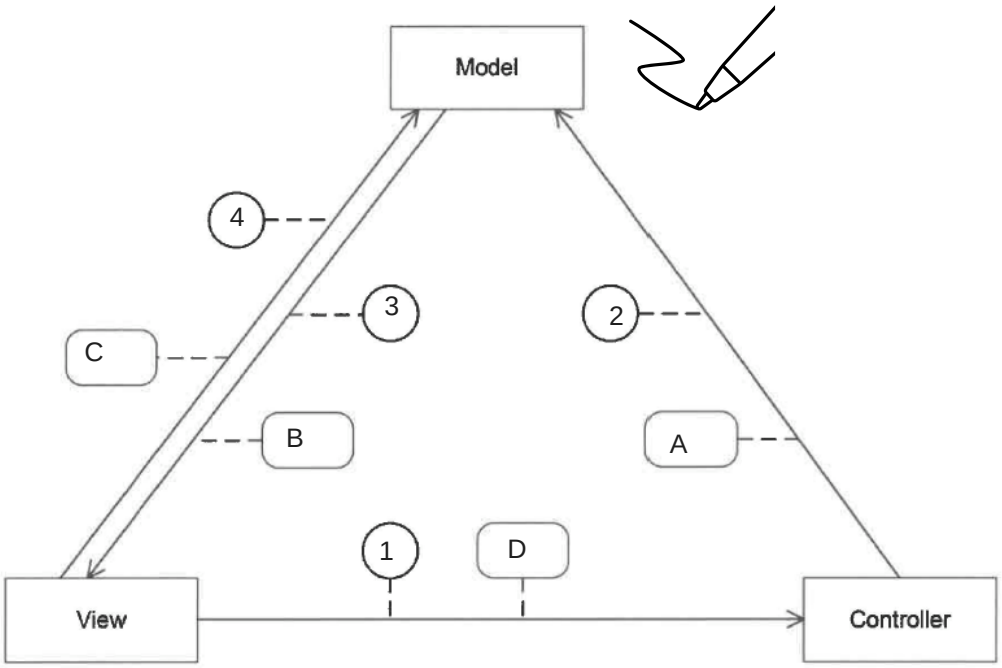
MVC ist eher ein architektonisches Muster, aber nicht für eine komplette Anwendung. MVC bezieht sich hauptsächlich auf die Benutzeroberfläche / Interaktionsschicht einer Anwendung. Sie benötigen immer noch eine Geschäftslogikschicht, vielleicht eine Dienstschicht und eine Datenzugriffsschicht.

MVC-Entwurfskomponenten

- Das Modell enthält nur die reinen Anwendungsdaten, es enthält keine Logik, die beschreibt, wie die Daten einem Benutzer präsentiert werden sollen.
- Die Ansicht stellt dem Benutzer die Daten des Modells dar. Der View weiß, wie er auf die Daten des Models zugreifen kann, aber er weiß nicht, was diese Daten bedeuten oder wie der Benutzer sie manipulieren kann.
- Der Controller steht zwischen der Ansicht und dem Modell. Er hört auf Ereignisse, die von der Ansicht (oder einer anderen externen Quelle) ausgelöst werden, und führt die entsprechende Reaktion auf diese Ereignisse aus. In den meisten Fällen besteht die Reaktion darin, eine Methode des Modells aufzurufen. Da der View und das Model über einen Benachrichtigungsmechanismus verbunden sind, wird das Ergebnis dieser Aktion automatisch in der View übernommen.

Aufgabe

Vervollständigen Sie die schematische Darstellung des MVC-Patterns. **Beschriften** Sie dabei die **Pfeile mit Aktivitäten** (in der Tabelle) und geben Sie die **Reihenfolge** an (im Schema die leeren Kreise mit 1 bis 4 beschriften).



A	Controller fordert Model zu Zustandsänderung auf
B	Model informiert View über Zustandsänderung
C	View fordert die geänderten Daten vom Model zur Ansicht für den Benutzer an.
D	View informiert Controller über Benutzereingabe

Lösungshilfe **https://t1p.de/xhzw5**	

Beispielcode in Java

Git-Link **https://t1p.de/5pi03**	