8
Design Patterns. Events

Jörn Loviscach

Versionsstand: 28. März 2015, 19:13

Die nummerierten Felder sind absichtlich leer, zum Ausfüllen beim Ansehen der Videos:
http://www.j3L7h.de/videos.html

Bitte hier notieren, was beim Bearbeiten unklar geblieben ist

1 Idee der Design Patterns

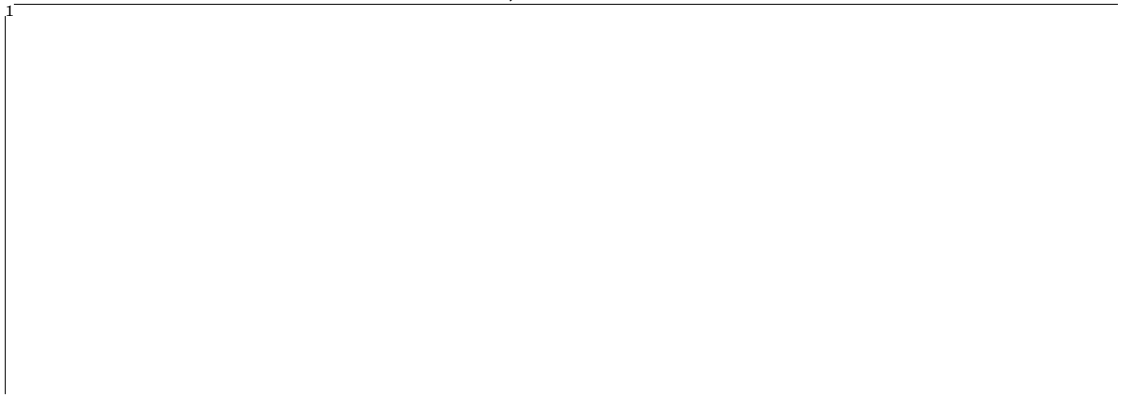
Design Patterns = Entwurfsmuster sind eigentlich eine Idee aus der Architektur (Christopher Alexander, 70er Jahre): bewährte, verallgemeinerbare Lösungen für das Straßennetz einer Stadt ebenso wie für Säulen und Fenster. In den 80/90ern wurde dieser Gedanke auf die Architektur objektorientierter Software übertragen. Die entsprechenden Design Patterns beschreiben keine Algorithmen (z. B. Sortierverfahren), sondern wie Klassen/Objekte zusammenspielen können, um einen bestimmten Effekt zu erreichen. Die Namen der Patterns sind dabei eine Kommunikationshilfe. Je nach Zählung kommt man auf Dutzende von Patterns. Es folgen einige Beispiele.

2 Singleton

Dies ist wohl das einfachste Pattern – und wird von einigen Leuten durchaus kritisch gesehen. Singleton (Einling = kein Zwilling) ist eine Lösung dafür, Klassen zu erzeugen, von denen es maximal eine Instanz geben darf – zum Beispiel Klassen, die Gegenstände modellieren, die es nur einmal gibt (Maus, Systemlaufwerk, . . .).

Die grundsätzliche Idee ist, den Konstruktor privat zu machen, so dass niemand außer der Klasse selbst eine Instanz von ihr erzeugen kann. Statt dessen stellt die Klasse eine statische Methode bereit, mit der man die eine Instanz erhält:

[1]



3 Composite

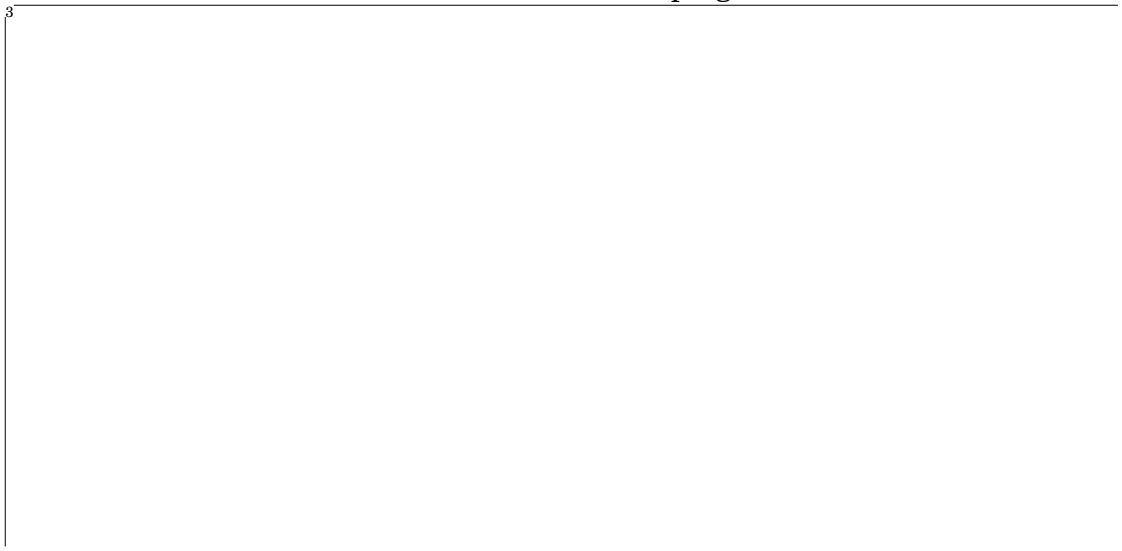
Das Pattern Composite (Verbundstoff) ist eine Lösung dafür, Zusammensetzungen auf dieselbe Art zu behandeln wie Einzelteile. Ein klassisches Beispiel dafür ist die Gruppierungsfunktion in Zeichensoftware: Eine Sammlung von Rechtecken, Kurven, Texten lässt sich zu einer Gruppe machen und dann insgesamt verschieben, drehen, skalieren, duplizieren usw., wie die einzelnen Zeichenobjekte.

Die Lösung dafür besteht darin, eine abstrakte Mutterklasse `Component` zu haben, die auf zwei Arten realisiert wird: `Leaf` ist das normale Einzelelement; `Composite` ist ein Verbund von `Component`s, also Einzelelementen oder anderen `Composite`s. Es ergibt sich eine Baumstruktur mit den Instanzen von `Leaf` als Blättern; daher der Name `Leaf`:

[2]

So sieht das ansatzweise in Code für ein Zeichenprogramm aus:

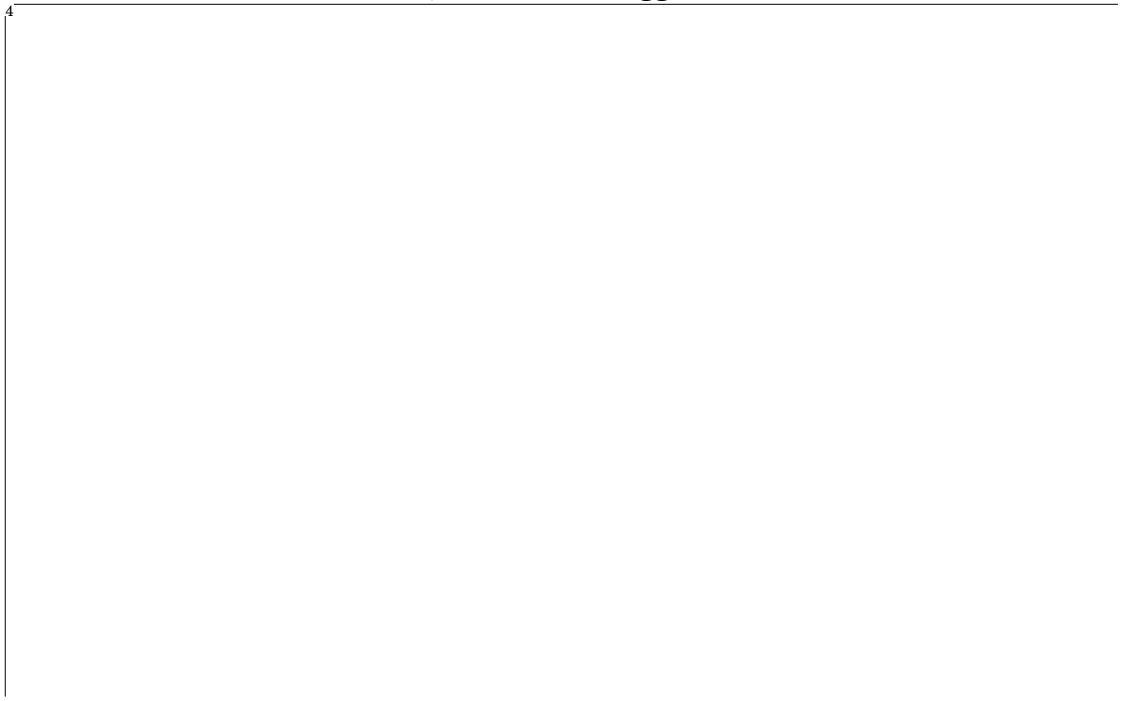
[3]



4 State

Das Pattern State (Zustand) erlaubt, endliche Automaten mit Objekten auszudrücken. Jeder Zustand des Automaten wird eine Instanz einer eigenen Unterklasse von `State`. Eine Instanz der anderen Klasse `StateContext` merkt sich den aktuellen Zustand. Auf „Ereignisse“ hin wird passend von einem `State` zum anderen umgeschaltet; dazu ergänzt man zu jedem Ereignis eine entsprechende Methode in `StateContext`, in `State` und ggf. in den Kindklassen von `State`:

[4]



5 Model–View–Controller

Insbesondere in Datenbankanwendungen (Buchungssysteme, Web-Shops, ...) verliert man leicht die Übersicht angesichts eines Gestrüpps an Funktionen, die Daten pflegen, und Funktionen, die zur Anzeige dienen. Die klassische Lösung dafür ist, die Funktionalität auf drei relativ eigenständige Klassen zu verteilen: Model (Datenmodell), View (Präsentation) und Controller (Steuerung):

[5]

6 Observer, Publish & Subscribe, Events

Einige Patterns sind so wichtig, dass sie inzwischen von Programmiersprachen direkt unterstützt werden – insbesondere das Observer-Pattern, auch Publish & Subscribe genannt:

[6]

In Java und C# gibt es „Events“ (Ereignisse), um das Abonnieren zu vereinfachen. Wir kennen das schon aus Windows Presentation Foundation:

[7]

Die Ereignisbehandlungsroutinen in C# erhalten typischerweise ein `object`, in dem der Auslöser übergeben wird, und obendrein eine Sammlung von

Zusatzangaben, die je nach Ereignis verschieden sind, eine Ableitung von `EventArgs`.

Ereignisse kann man auch selbst definieren. Das ist dann ein weiterer Typ von Elementen einer Klasse – nach Attributen, Methoden und Properties. Hinter den Kulissen macht der Compiler allerdings aus Ereignissen wieder nur eine kunstvolle Zusammenstellung von Attributen und Methoden.

[8]