

Entwurfsmuster Zustand

Julian Fietkau

Universität Hamburg

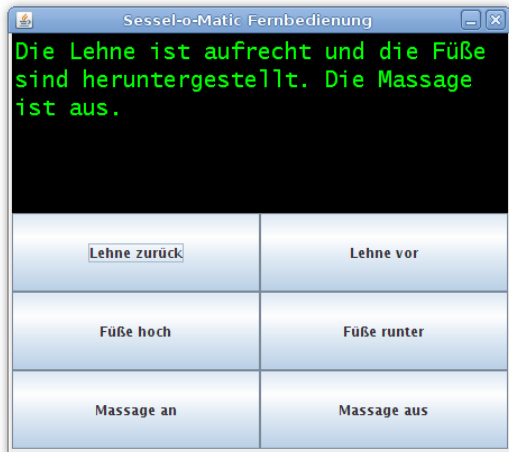
27. April 2010

basierend auf einem Teachlet von:
Janina Nemeč, Julian Fietkau

Enterprise Sessel Solutions braucht euch!



der Sessel-o-Matic



der Sessel-o-Matic

🔍 Beobachtungen:

- Die Lehne kann zwischen zwei Zuständen umgeschaltet werden: aufrecht und zurückgestellt
- Die Massage hat ebenfalls zwei mögliche Zustände: ein- und ausgeschaltet
- Diese Zustände sind nicht unabhängig voneinander: Nur, wenn die Lehne zurückgestellt ist, kann die Massage aktiviert werden, und nur, wenn die Massage ausgeschaltet ist, kann die Lehne hochgestellt werden
- Die Zustände von Lehne und Massage werden über je zwei Buttons manipuliert
- Die zwei Buttons für die Fußstütze sind noch ohne Funktion

Was das System bisher kann



Da muss noch mehr gehen

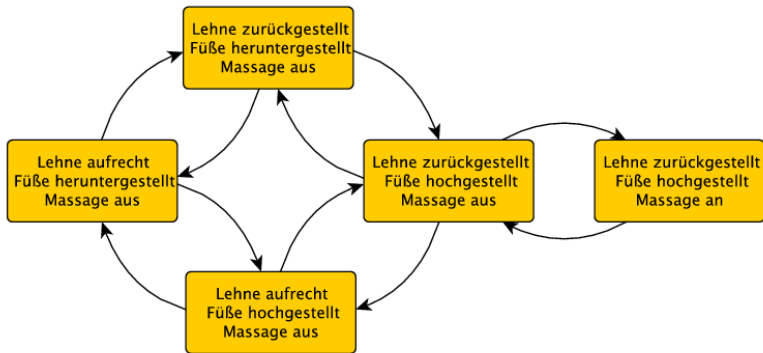
- Hochstellbare Füße (unabhängig von der Lehne)
- Massage nur aktivierbar, wenn Lehne zurück und Füße hochgestellt

Da muss noch mehr gehen

🔍 Beobachtungen:

- Wir erinnern uns: Bereits im Ausgangssystem war die Massage nur bei zurückgestellter Lehne aktivierbar
- Nun sollen Lehne und Fußstütze unabhängig voneinander umgeschaltet werden können
- Die Massage soll jedoch nur bei zurückgestellter Lehne und hochgestellten Füßen aktivierbar sein
- Wie würde ein aktualisiertes Zustandsdiagramm aussehen?

Da muss noch mehr gehen



Blick auf die „Innereien“!

Machen wir uns die Finger schmutzig...

Blick auf die „Innereien“!

🔍 Beobachtungen:

- Die Fernbedienung kennt ihren Sessel, der Sessel kennt seine Fernbedienung nur als ein Display für Nachrichten
- Die Fernbedienung ruft beim Drücken eines Buttons einfach eine Methode am Sessel auf
- Der Sessel speichert seinen eigenen Zustand in zwei Zustandsfeldern
- Es existiert ein bisher unbenutzter Enumerationstyp für den Zustand der Füße
- Alle benötigten Strings sind in der Klasse Nachrichten abgelegt

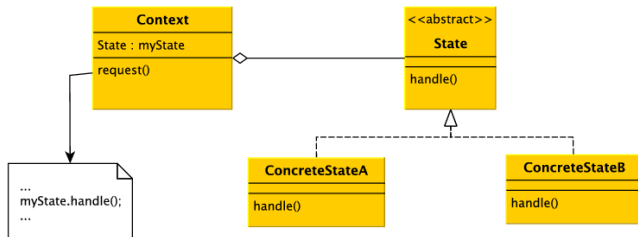
Was ist das Problem?

- if/switch/case-Salat wird schnell unübersichtlich
- Zustände und Zustandsübergänge werden getrennt voneinander definiert
- Der (Gesamt-)Zustand des Sessels ist nicht vom Sessel entkoppelt

Wie lösen wir das?

Gibt es eine pragmatische, objektorientierte Lösung?

Zustand objektorientiert nach GoF



- Kapselung der Zustände und ihres Verhaltens in eigenen Klassen
- vermeidet if/switch/case-Salat
- Dank Koppelung von Zustand und Verhalten können neue Zustände leicht hinzugefügt werden

Jetzt wird gebaut!

Es wird ernst: Die Umsetzung.

Jetzt wird gebaut!

🔍 Beobachtungen:

- Erstellung neuer Zustände ist im fertigen System eine leichte Übung
- Der Sessel und sein Zustand wurden entkoppelt: Der Sessel hält nun ein Exemplar eines (möglichen) Zustands
- Jeder Zustand ist zusammen mit seinem Verhalten und seinen Übergängen gekapselt
- Der Code ist insgesamt klarer strukturiert
- An der Fernbedienung oder der Schnittstelle des Sessels waren keinerlei Änderungen notwendig

Der Auftrag wurde erfüllt!



Noch mal Gedanken machen

Was hätte noch anders gemacht werden können? Gibt es mögliche Varianten der Implementierung?

Füße hochlegen!

Jetzt geht es ja. ;)

Weiterführende Literatur:

Modellierung zustandsorientierter Systeme in Java: Das Zustandsmuster, Varianten und Alternativen

Julian Fietkau, Janina Nemec

[http://www.julian-fietkau.de/
modellierung_zustandsorientierter_systeme_in_java](http://www.julian-fietkau.de/modellierung_zustandsorientierter_systeme_in_java)

Folien-Download und Feedback-Möglichkeit sowie (nach
Aufbereitungszeit) das Video:

http://www.julian-fietkau.de/entwurfsmuster_zustand_2010