Memento Pattern

Memento Pattern

Allgemein

- Memento [lat.]: Erinnerung
- Klassifizierung: objektbasiertes Verhaltensmuster
- Alias: Token, Snapshot

Entwurfsmuster		Aufgabe		
		Erzeugungsmuster	Strukturmuster	Verhaltensmuster
Gültigkeits-bereich	klassenbasiert	Fabrikmethode	Adapter (klassenbasiert)	Interpreter Schablonenmethode
	objektbasiert	Abstrakte Fabrik Erbauer Prototyp Singleton	Adapter (objektbasiert) Brücke Dekorierer Fassade Fliegengewicht Kompositum Proxy	Befehl Beobachter Besucher Iterator Memento Strategie Vermittler Zustand Zuständigkeitskette

Memento Pattern

Zweck	Anwendung
• Erfassung und Auslagerung des internen Zustandes eines Objektes, ohne seine Kapselung zu verletzen, so dass das Objekt später in diesen Zustand zurückversetzt werden kann.	• Undo/Redo-Operationen in allen Bereichen • Transaktionsprotokollierung, z.B. bei Datenbanken • Zustandsinformationen einer Anwendung zum Schutz vor Hardwareausfallen erfassen • Ermittlung eines Objekteszustandes welches keine direkte Schnittstelle dafür zur Verfügung stellt

Memento Pattern

Teilnehmer

Memento (Memento)

- speichert den internen Zustand des Originator-Objektes.
- schützt gegen Zugriffe von anderen Objekten als dem Urheber.
- Mementi haben eigentlich zwei Schnittstellen. Caretaker sieht eine schmale Schnittstelle. Der Urheber hingegen sieht eine breite Schnittstelle. Im Idealfall hat nur der Urheber, der das Memento erzeugt hat, Zugriff auf den internen Zustand des Memento.

Caretaker (Aufbewahrer)

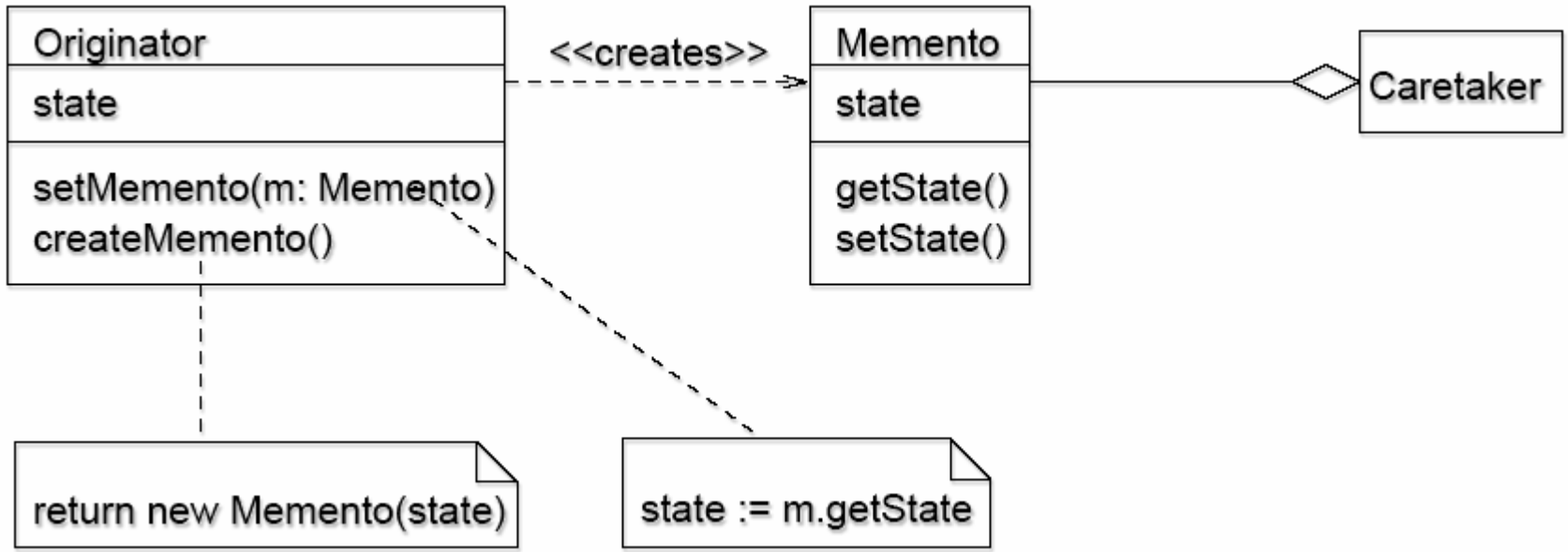
- ist für Aufbewahrung und Verwaltung des Memento zuständig
- inspiziert oder verändert niemals den Zustand des Memento

Originator (Urheber)

- erzeugt ein Memento seines gegenwärtigen internen Zustandes
- benutzt das Memento, um seinen internen Zustand wiederherzustellen

Memento Pattern

Strukturdiagramm



Memento Pattern

Beispiel: Dokument

```
// Dokument.java (Originator)

import java.util.ArrayList;
import java.util.Iterator;

public class Dokument {
     ArrayList _zeilen = new ArrayList();

     public void beschreiben(int index, String zeile){
             _zeilen.add(index, zeile);
     }
     public void entferne(String zeile){
             _zeilen.remove(zeile);
     }
     public Memento erzeugeMemento(){
             return new Memento(_zeilen.toArray());
     }
     public void setzeMemento(Memento memento){
             _zeilen.clear();
             Object[] elemente = memento.holeElemente();
             for (int i = 0; i < elemente.length; i++) {
                     _zeilen.add(elemente[i]);
             }
     }
     public String toString() {
             String dokument = "";
             Iterator iter = _zeilen.iterator();
             while(iter.hasNext())
                     dokument += iter.next()+ "\n";
             return dokument;
     }
}
```

```
// Memento.java (Memento)

public class Memento {

     private Object[] _elemente;

     public Memento(Object[]
     elemente) {
             _elemente = elemente;
     }

     public Object[] holeElemente() {
             return _elemente;
     }
}

// alternativ:
// Memento als innere Klasse
// von Dokument, um eine schmale
// Schnittstelle nach außen
// sicherzustellen.
```

Memento Pattern

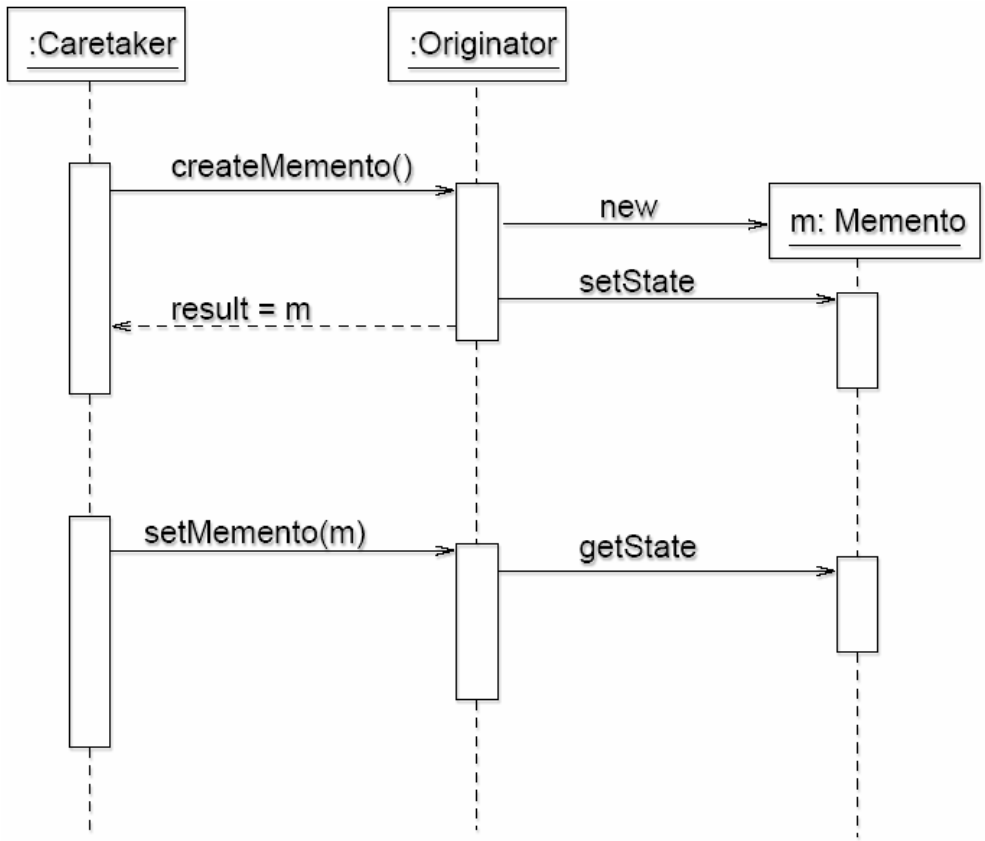
Beispiel: Dokument

```
// Aufbewahrer.java (Caretaker)

import Memento;

public class Aufbewahrer {
     public static void main(String[] args){

               // erstellt neues Dokument
               Dokument dokument = new Dokument();

               // Dokument beschreiben und anzeigen
               dokument.beschreiben(0,"-------");
               dokument.beschreiben(1,"|     |");
               dokument.beschreiben(2,"| alt |");
               dokument.beschreiben(3,"|     |");
               dokument.beschreiben(4,"-------");
               System.out.println("Das Ausgangsdokument:\n"+dokument);

               // merke den Zustand des Dokuments
               Memento memento = dokument.erzeugeMemento();

               // Dokument verändern und anzeigen
               dokument.entferne("| alt |");
               dokument.beschreiben(2,"| neu |");
               System.out.println("Das veränderte Dokument:\n"+dokument);

               // altes Dokument wiederherstellen und anzeigen
               dokument.setzeMemento(memento);
               System.out.println("Das wiederhergestellte
                              Dokument:\n"+dokument);
     }
}
```

```
Console:

Das Ausgangsdokument:
-------
|     |
| alt |
|     |
-------

Das veränderte Dokument:
-------
|     |
| neu |
|     |
-------

Das wiederhergestellte Dokument:
-------
|     |
| alt |
|     |
-------
```

Memento Pattern

Sequenzdiagramm



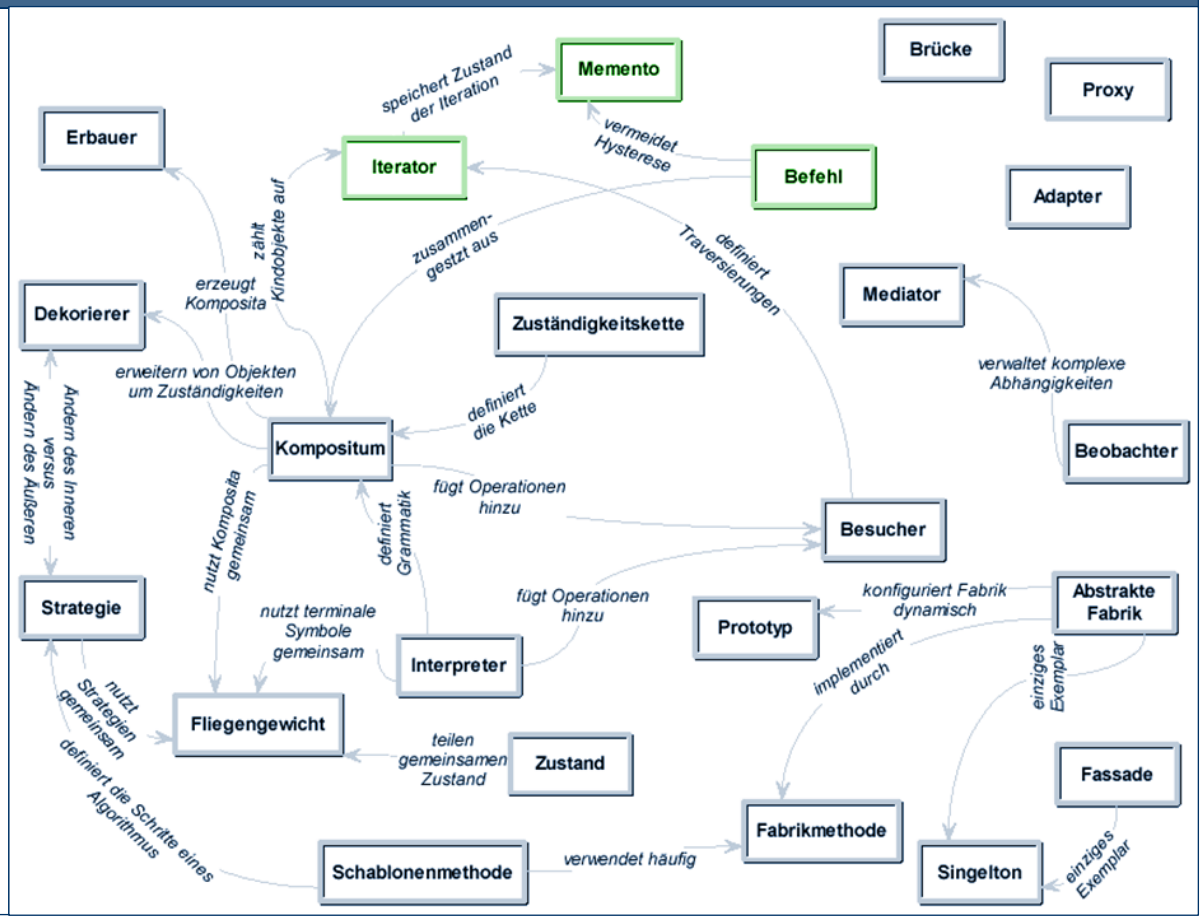
Memento Pattern

Pro & Contra	Performance Checkliste
+ Wahrung der Kapselungsgrenzen + Vereinfachung des Originators - Verwendung von Mementi kann teuer sein - Definition schmaler und breiter Schnittstellen - Versteckte Kosten beim Aufbewahren von Mementi	Vor der Verwendung von Memento: • Laufzeitkostenanalyse durchführen • Häufigkeit der Erzeugung und ggf. Einsparpotential durch inkrementelle Erzeugung ermitteln

Memento Pattern

Verwandte Patterns

- **Iterator**
  Memento speichert den Zustand der Iteratoren
- **Command**
  Memento vermeidet Abhängigkeit eines Zustandes von früheren Zuständen



Memento Pattern

Quellen

Internet

- Fraunhofer Institut für Datenverarbeitung:
  http://genesis.iitb.fhg.de/servlet/is/3710/
- University of Calgary - Ann Dang & Kristen Anderson
  http://sern.ucalgary.ca/courses/SENG/443/W02/assignments/Memento/

Literatur

- The Design Patterns Java Companion – James W. Cooper
  Addison Wesley, 1998