Programmiermethodik

Entwurfsmuster

SS 2002

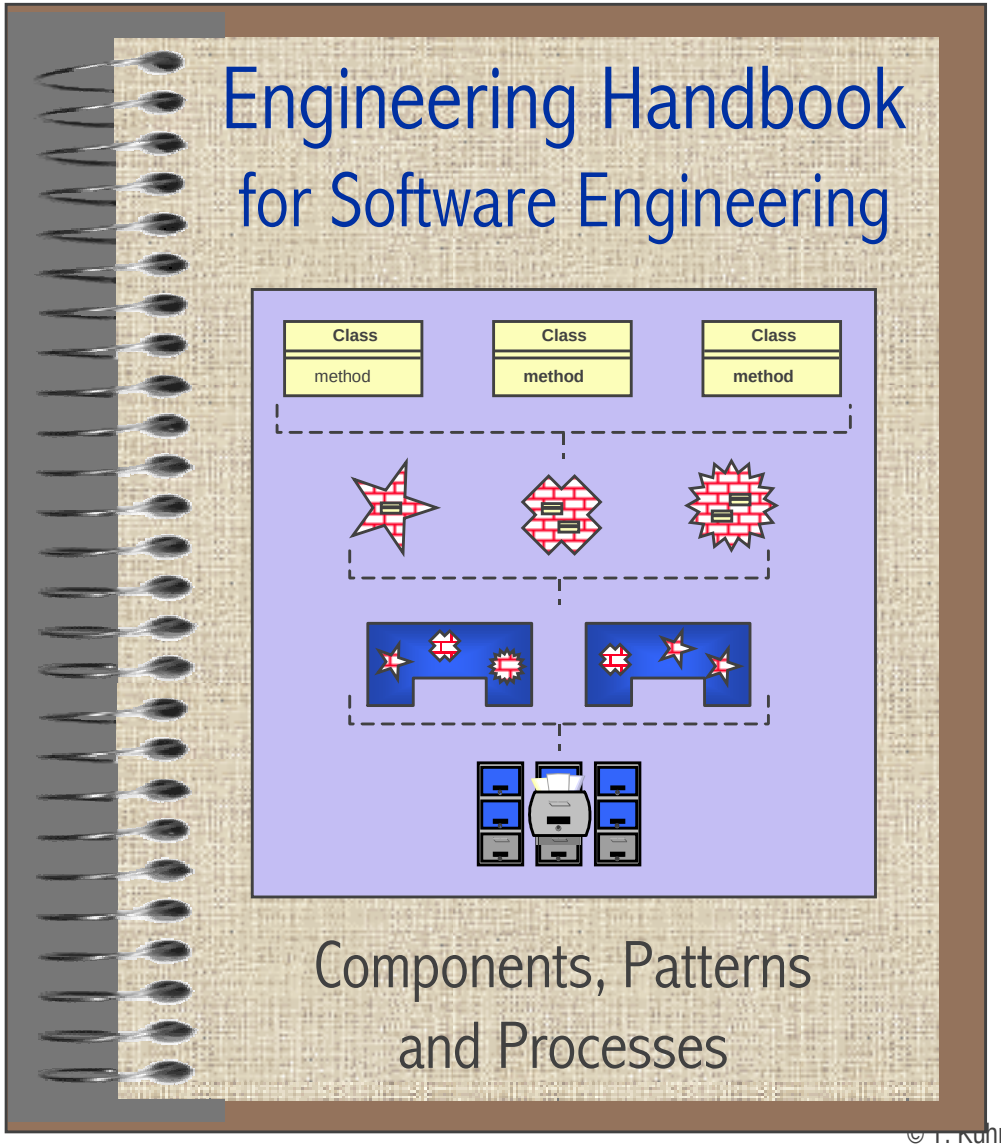
Thomas Kühne

kuehne@informatik.tu-darmstadt.de

http://www.informatik.uni-mannheim.de/informatik/softwaretechnik

Warum Entwurfsmuster?

- Systematischer Softwareentwurf
  - » Dokumentation von Expertenwissen
  - » Verwendung von generischen Lösungen
  - » Erhöhung des Abstraktions-niveaus



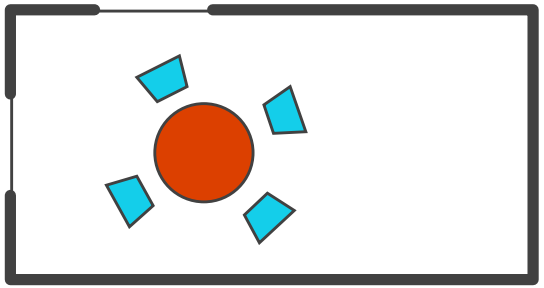
Was ist ein Entwurfsmuster?

Muster erfassen Expertise

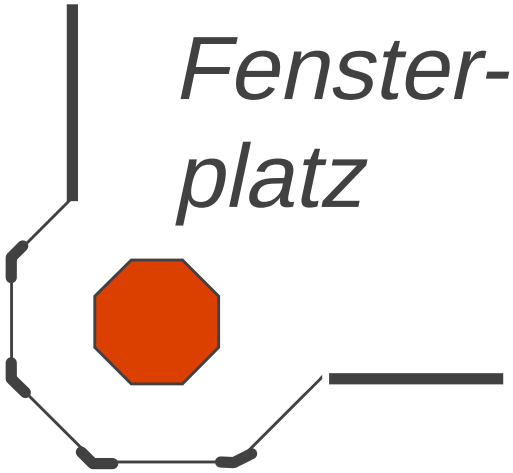
Angeln	*Anekdoten*
Schach	*von den Regeln zum Meister*
Literatur	*älteste “Muster” Referenz*
Agrawirtschaft	*Weisheit vs. Wissenschaft*
Architektur	*Vorbild für Softwaremuster*
Software	

Architekturmuster

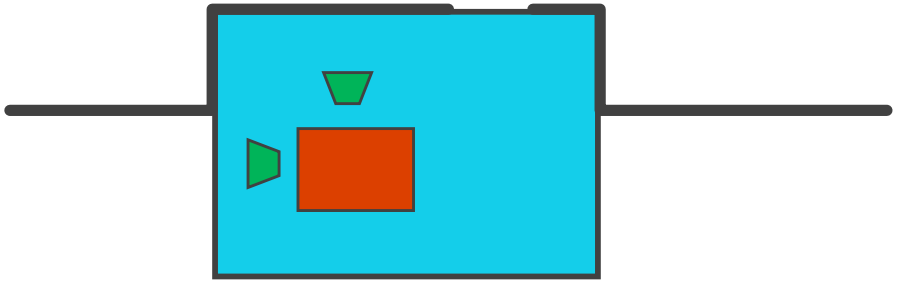
Licht von zwei Seiten



Fenster-platz



Tiefer Balkon



Eine Musterdefinition

*Each pattern describes a **problem** which occurs over and over again in our **environment**, and then describes the core of the **solution** to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice.*

Christopher Alexander

Eine Musterdefinition

Kurze Definition

Ein Entwurfsmuster beschreibt
eine *wiederkehrende Lösung*
zu einem *Problem*
in einem *Kontext.*

Warum Softwaremuster?

- Der Entwurf von Qualitätssoftware ist schwer
- Anfänger sind von der Vielzahl der Möglichkeiten überwältigt
- Experten nutzen ihre Erfahrung
  - » Viele Entwurfslösungen kehren immer wieder
- Das Verstehen von wiederkehrenden Entwurfslösungen bietet mehrere Vorteile

Warum Softwaremuster?

Vorteile

- Qualität — *Muster fördern Qualitätsaspekte*
- Design Reuse — *Entwurfsbausteine*
- Dokumentation — *Erhöhtes Abstraktionsniveau*
- Kommunikation — *Entwurfsvokabular*
- Lehre — *Weiterreichen von Kulturgut*

Warum Softwaremuster?

Wisdom is often ascribed to those who can tell just the right story at the right moment and who often have a large number of stories to tell.

Robert C. Shank

Was ist ein Softwaremuster?

- erprobte Softwarepraxis
- Literatur (Erfahrungsbeschreibung)
- Baustein (für den Entwurf)
- mögliche Abstraktionsebenen:
  - » Sprachkonstrukt
  - » Idiom
  - » Entwurfsmuster
  - » Architekturmuster (bzgl. Softwarearchitektur)

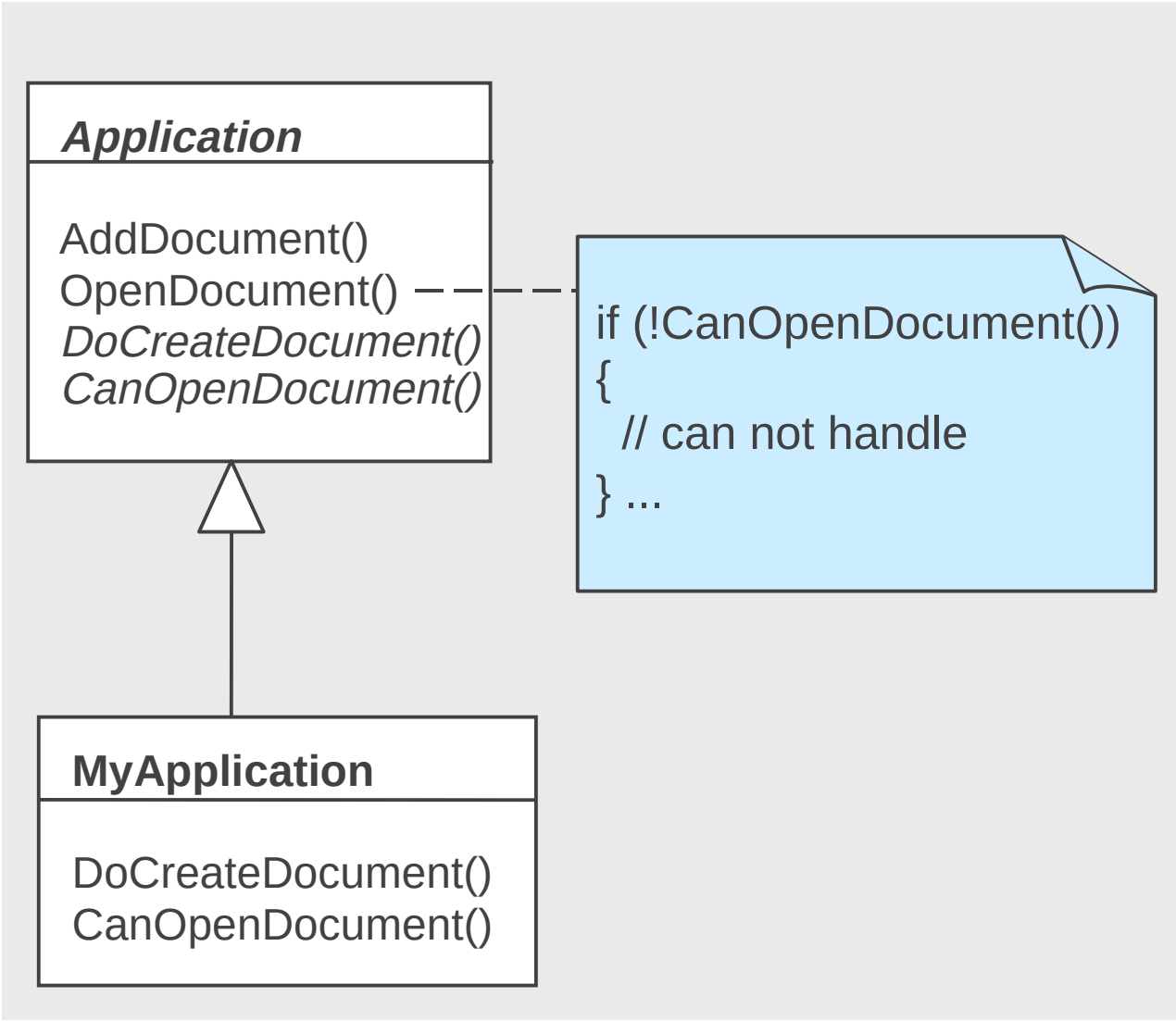
Was ist ein Entwurfsmuster?

A methodology tells you how to write down the decisions you have made. A pattern tells you which decisions to make, when and how to make them, and why they are the right

Beck & Johnson

Beispiel: Template Method

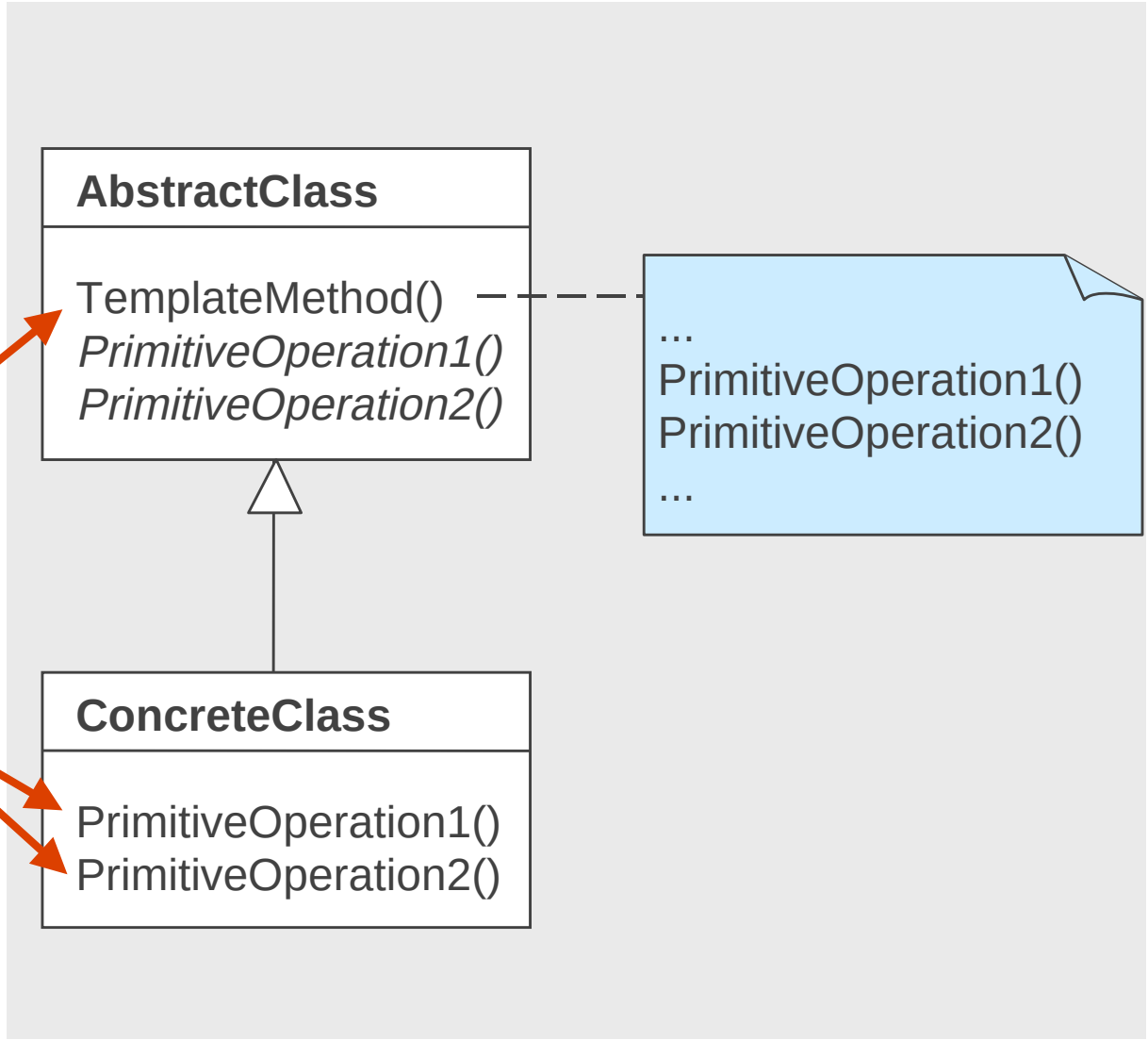
- oft benutzt in Frameworks
- abstrakte Klassen
  - » definieren Schnittstellen
  - » enthalten Code



Beispiel: Template Method

Vorteile

- vermeidet Code-Duplikation
- separiert invariante (Policy) und variante Teile (Mechanisms)
- Kontrolle über Unterklassen-erweiterungen



Entwurfsmusterschablone

- Identifikation
  - » Name
  - » Intent
  - » Also Know As
- Problembeschreibung (Context)
  - » Motivation (Forces)
  - » Applicability

Entwurfsmusterschablone

- Lösung
  - » Structure
  - » Participants
  - » Collaboration
  - » Consequences (Trade offs)
  - » Implementation
  - » Sample Code

Entwurfsmusterschablone

- Referenzen
  - » Known Uses
  - » Related Patterns

Beispiel: Composite

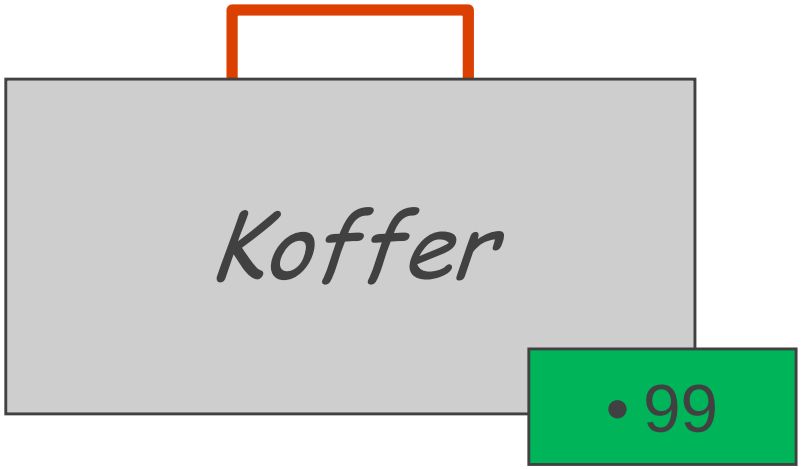
Absicht

Organisiere Objekte in Baumstrukturen, um Ganzes-Teil (whole-part) Hierarchien zu verwirklichen. "Composite" ermöglicht es Klienten einzelne und zusammengesetzte (Behälter-)Objekte gleich zu behandeln.

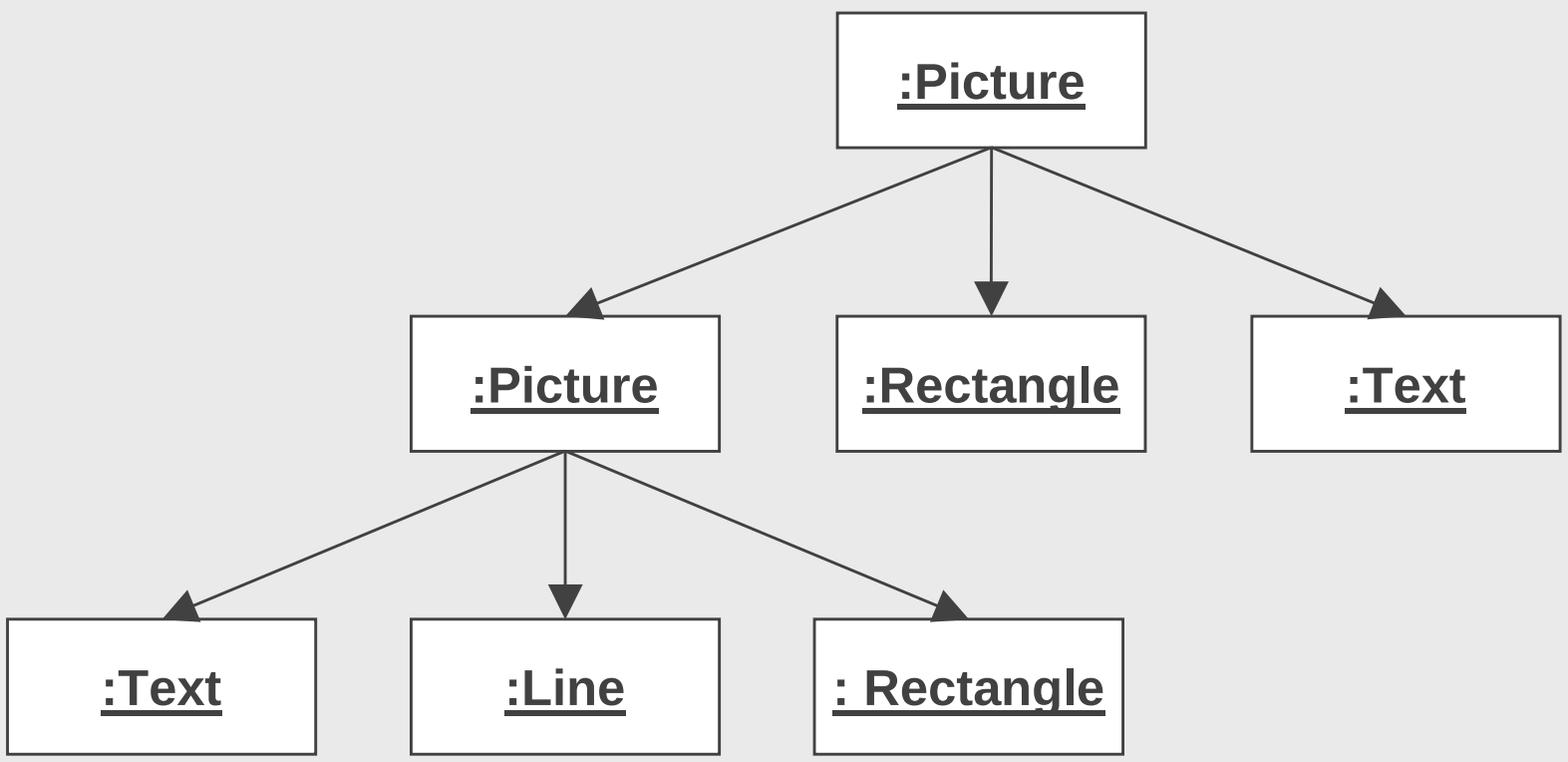
Composite: Motivation

Zeicheneditor:

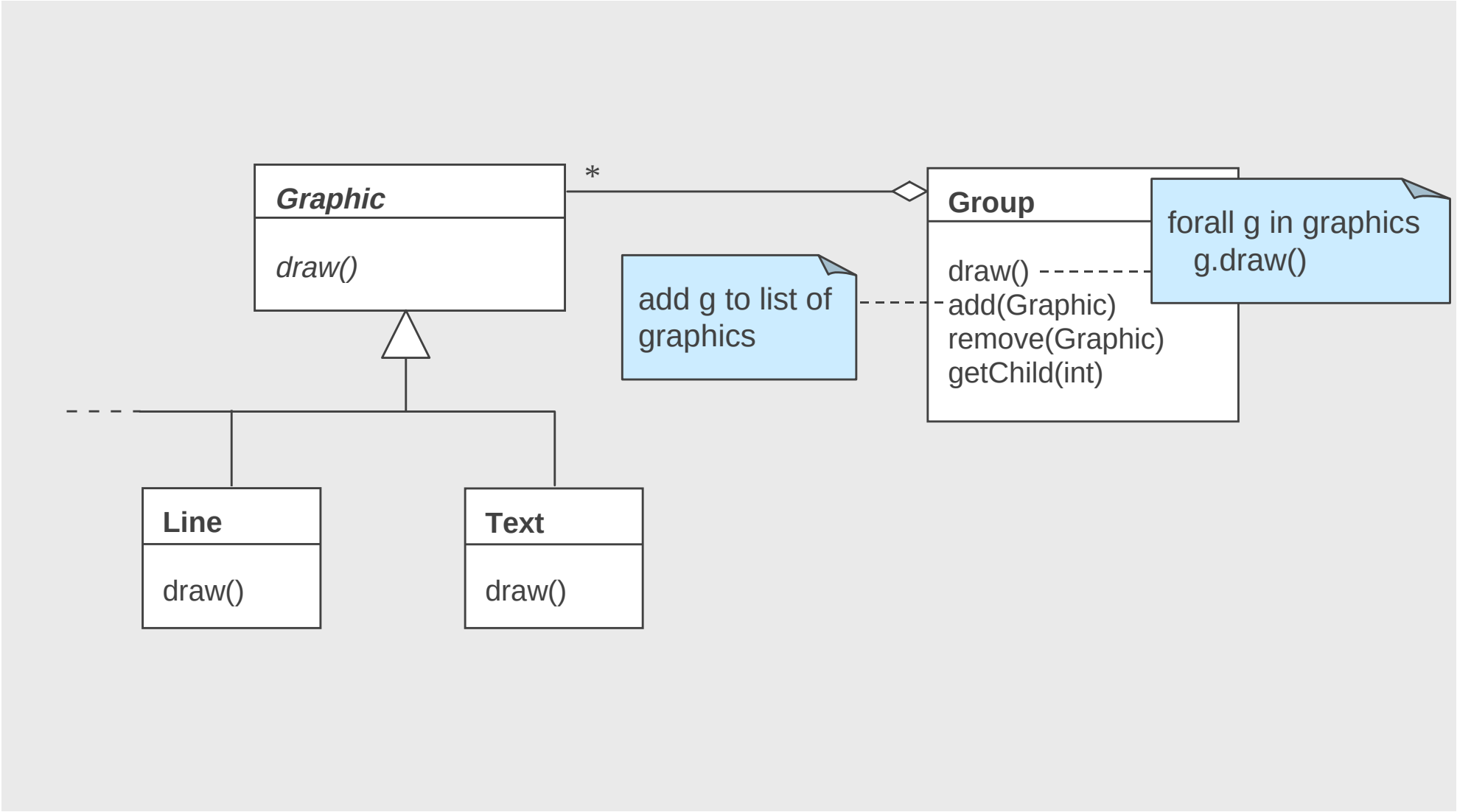
- Bilder enthalten Elemente
- Elemente können gruppiert werden
- Gruppen können andere Gruppen enthalten



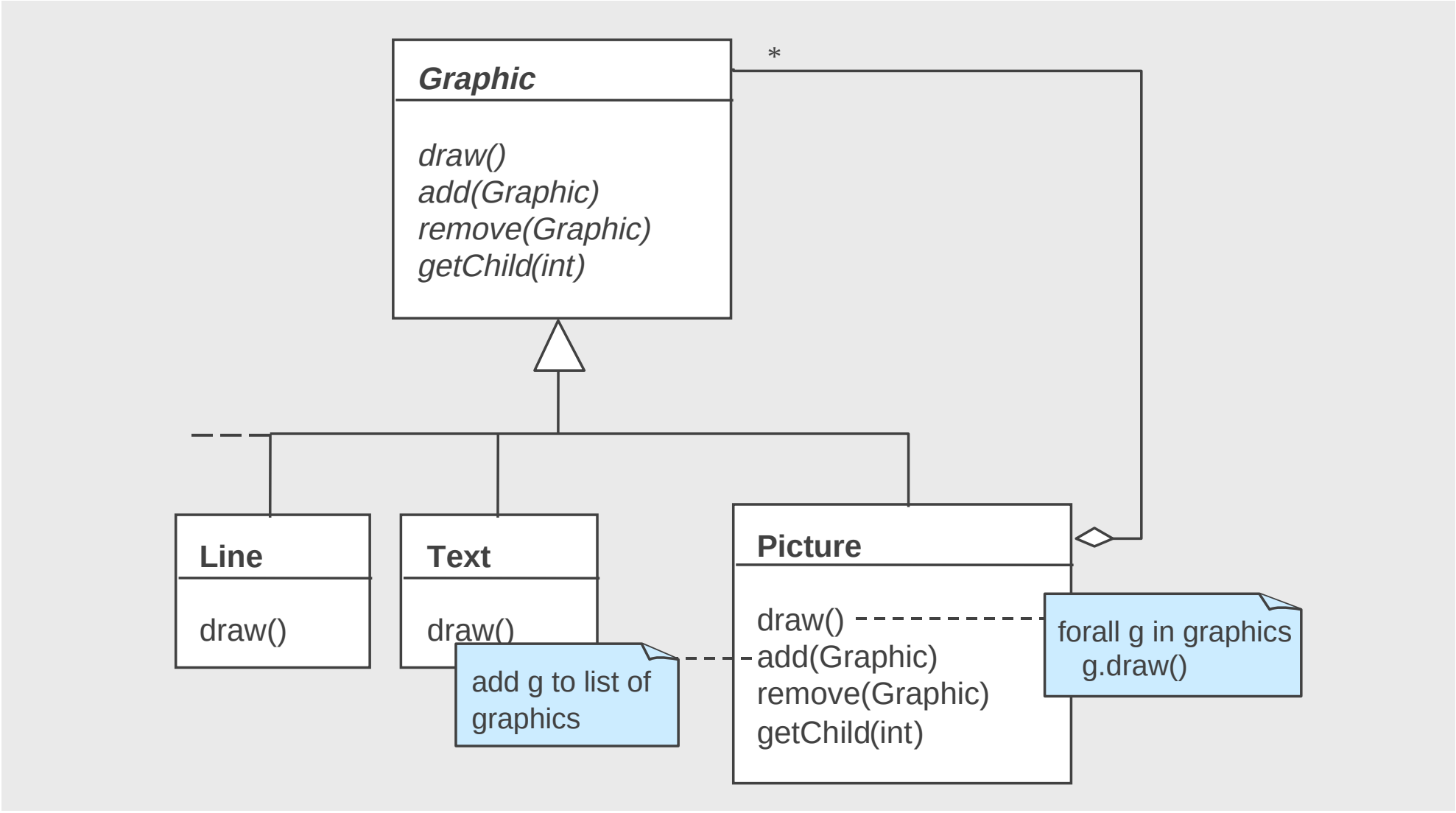
Composite: Motivation



Composite: Motivation



Composite: Motivation

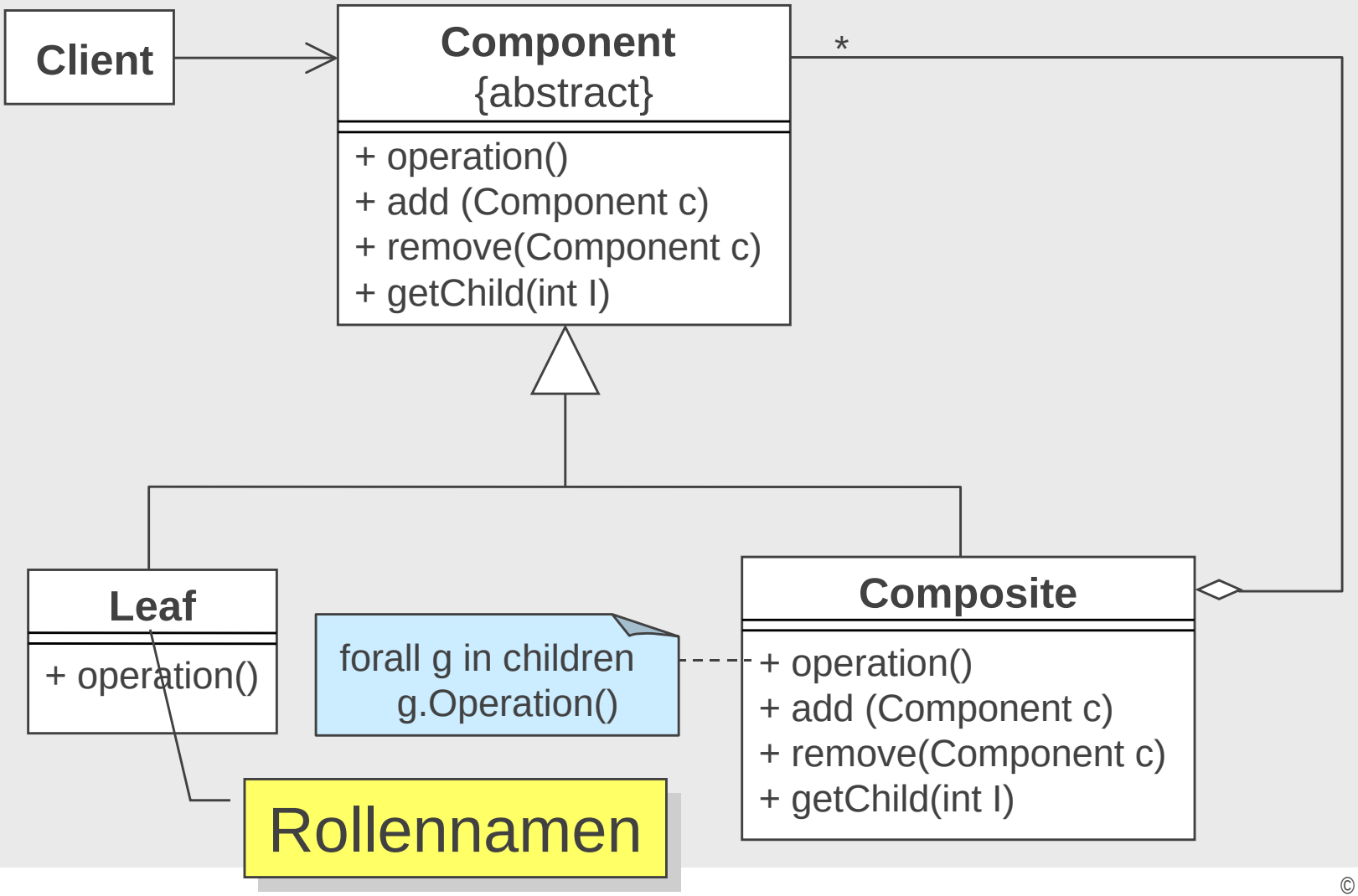


Composite: Anwendbarkeit

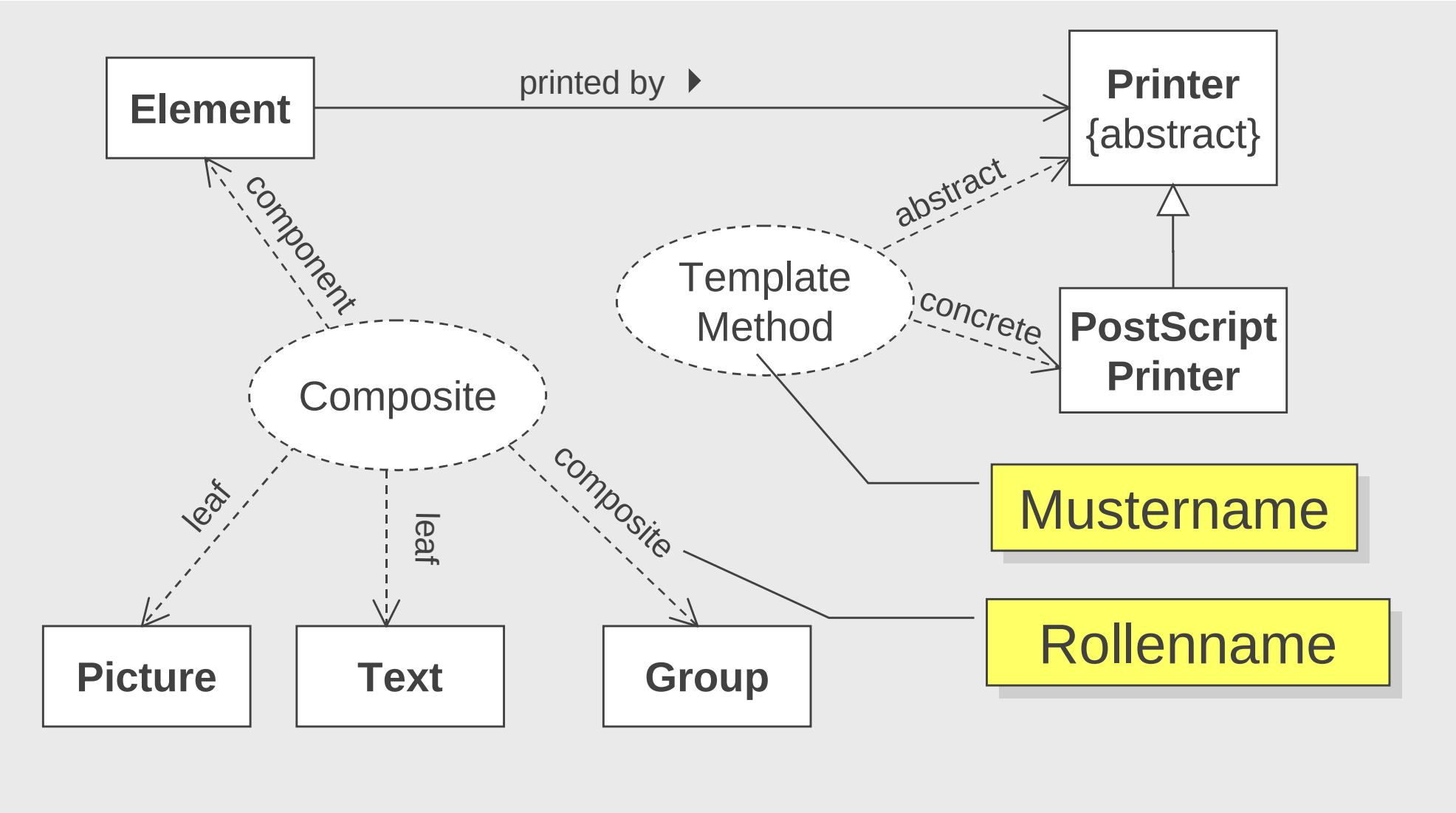
Benutze "Composite" falls:

- Ganzes-Teil Hierarchien repräsentiert werden sollen
- Klienten in der Lage sein sollen, den Unterschied zwischen Elementen und Behältern zu ignorieren

Composite: Struktur



Dokumentation von Mustern



Composite: Teilnehmer

- **Component**
  - » deklariert Schnittstelle / Default-Verhalten
- **Leaf**
  - » atomare Elemente / primitives Verhalten
- **Composite**
  - » speichert Kinder / Komponentenverhalten
- **Client**
  - » benutzt die "Component" Schnittstelle

Composite: Kollaborationen

- Kunden interagieren mit Objekten durch die "Component"-Schittstelle
- Blätter reagieren direkt
- Behälter leiten Anfragen and ihre Kinder weiter und fügen möglicherweise "Vorher/Nacher-Operationen" hinzu

Composite: Konsequenzen

- primitive Objekte können rekursiv zusammengesetzt werden
- Klienten können Behälter und Elemente gleich behandeln
- es ist leicht neue Komponenten hinzuzufügen
- Entwurf kann zu allgemein werden (Verschwimmen von Unterschieden)

Composite: Implementierung

- explizite Elt-Verweise
- Teilen von Komponenten
- Component-Schnittstelle mit Behälterfunktionalität?
  - » ja, ansonsten wird doch wieder ein Unterschied zwischen Elementen und Behältern deutlich
  - » nein, Elemente können diese Versprechen nicht sinnvoll halten

Composite: Beispiel-Code

```
abstract class Equipment {
  String          name() { return name; }
  abstract int    price();

  abstract void add(Equipment eq);
  abstract void remove(Equipment eq);

  private String name;
}
```

Composite: Beispiel-Code

```
class FloppyDisk extends Equipment {
  int price()
  {
    return 50;
  }

  ...
}
```

Composite: Beispiel-Code

```
class CompositeEquipment extends Equipment {
  int price()
  { ...
    for (i=0; i<equipment.length; i++)
      total += equipment[i].price();
  }

  void add(Equipment eq)    {...};
  void remove(Equipment eq) {...};

  private Equipment[] equipment;
}
```

Composite: Beispiel-Code

```
Chassis.add(new FloppyDisk("3.5in Floppy"));

Bus bus = new Bus("PCI Bus");
bus.add(new Card("ATM Network card"));

chassis.add(bus);
cabinet.add(chassis);

System.out.println("Total price: " +
                   cabinet.Price()
                  );
```

Composite: Benutzungen

- "View"-Klasse von Model/View/Controller
- Anwendungs-Frameworks & Toolkits
  - » ET++, 1988
  - » Graphics, 1988
  - » Glyphs, 1990
  - » InterViews, 1992
- Finanzportfolios

Composite: "Related Patterns"

- Iterator
  - » Enummerieren der Kinder
- Visitor
  - » besuchen von heterogenen Strukturen
- Chain of Responsibility
- Flyweight
- Decorator

Musterexpertise

Bewußtseinsstufen:

- Unschuld
- Kenntniss einiger Tricks
- kompetente Trickanwendung
- Anwendbarkeit & Konsequenzen bekannt
- weites Wissen über Muster & ihre Interaktion
- fähig eigenes Wissen literarisch zu fassen (→ Musterbeschreibung)

“Muster” mal wörtlich...

Aspekte von Entwurfsmustern:

- Mustererkennung *Entdecken von Wiederauftreten*
- “Mustergültig” *Nachahmen bewährten Entwurfs*
- Strickmuster *Anordnung exisitierender Klassen*
- Schnittmuster *Konstruktionsplan*
- Ordnungsmuster *Teilnehmer formen eine Struktur*
- Verhaltensmuster *Teilnehmer “spielen” Rollen*

Werden Muster erfunden?

Entwurfsmuster werden *gefunden*

- “Aggressive disregard for originality”
- “Rule of three”

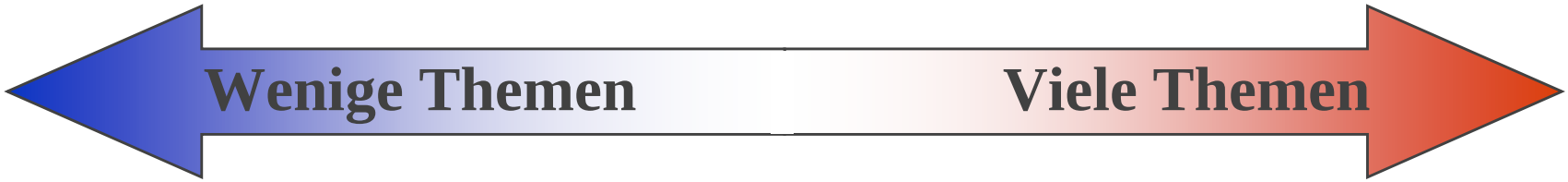
> *Once is an event, twice is an incident, thrice it's a pattern.*
> Jerry Weinberg

Entwurfsmuster: "Forces"

Kontext Komposition eines Liedes

Problem Die Aufmerksamkeit des Zuhörers soll gehalten werden ohne ihn zu überfordern

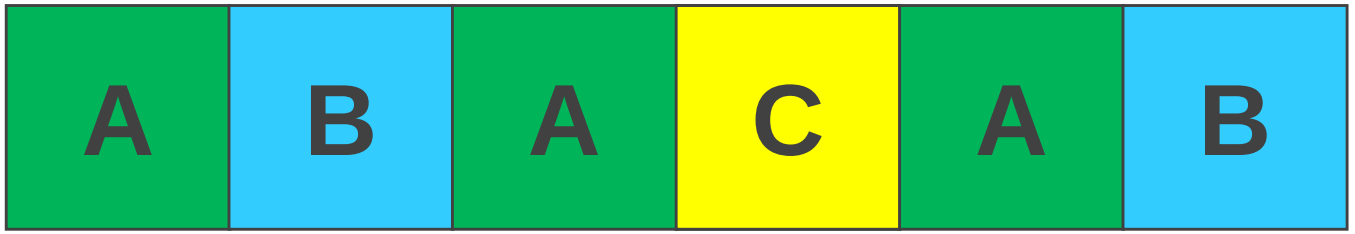
Kräfte Die Zahl der Themen ist kritisch



Zuhörer gelangweilt *Zuhörer überfordert*

Entwurfsmuster: "Forces"

Lösung Wiederholung mit Variation

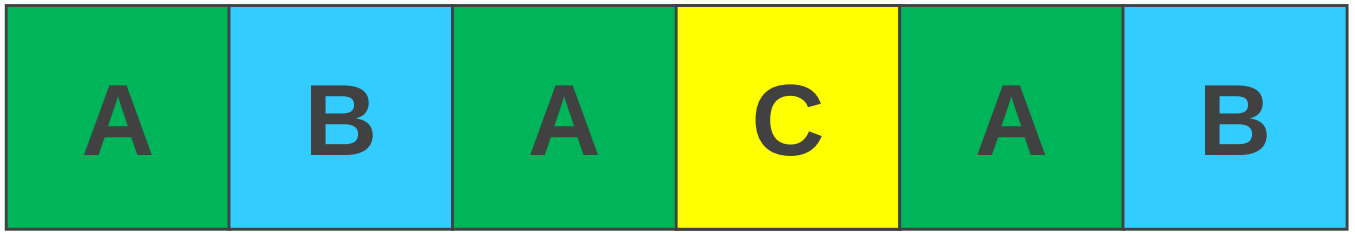


Konsequenzen

- Zugänglichkeit *Leichtes Lernen durch Wiederholung*
- Aufmerksamkeit *“Aufwecken” durch Variation*
- Mißbrauch *Abnutzungseffekt möglich*

Entwurfsmustersprachen

Lösung Wiederholung mit Variation



Konsequenzen

- Zugänglichkeit
- Aufmerksamkeit
- Mißbrauch

Ergebnis eines Musters setzt den Kontext für das nächste

Lösung als neuer Kontext

Kontext Komposition eines Liedes nach ABACAB kann langweilig wirken

Problem Das neue Thema soll Änderung bringen, soll sich aber einfügen

Langeweile *Überforderung*

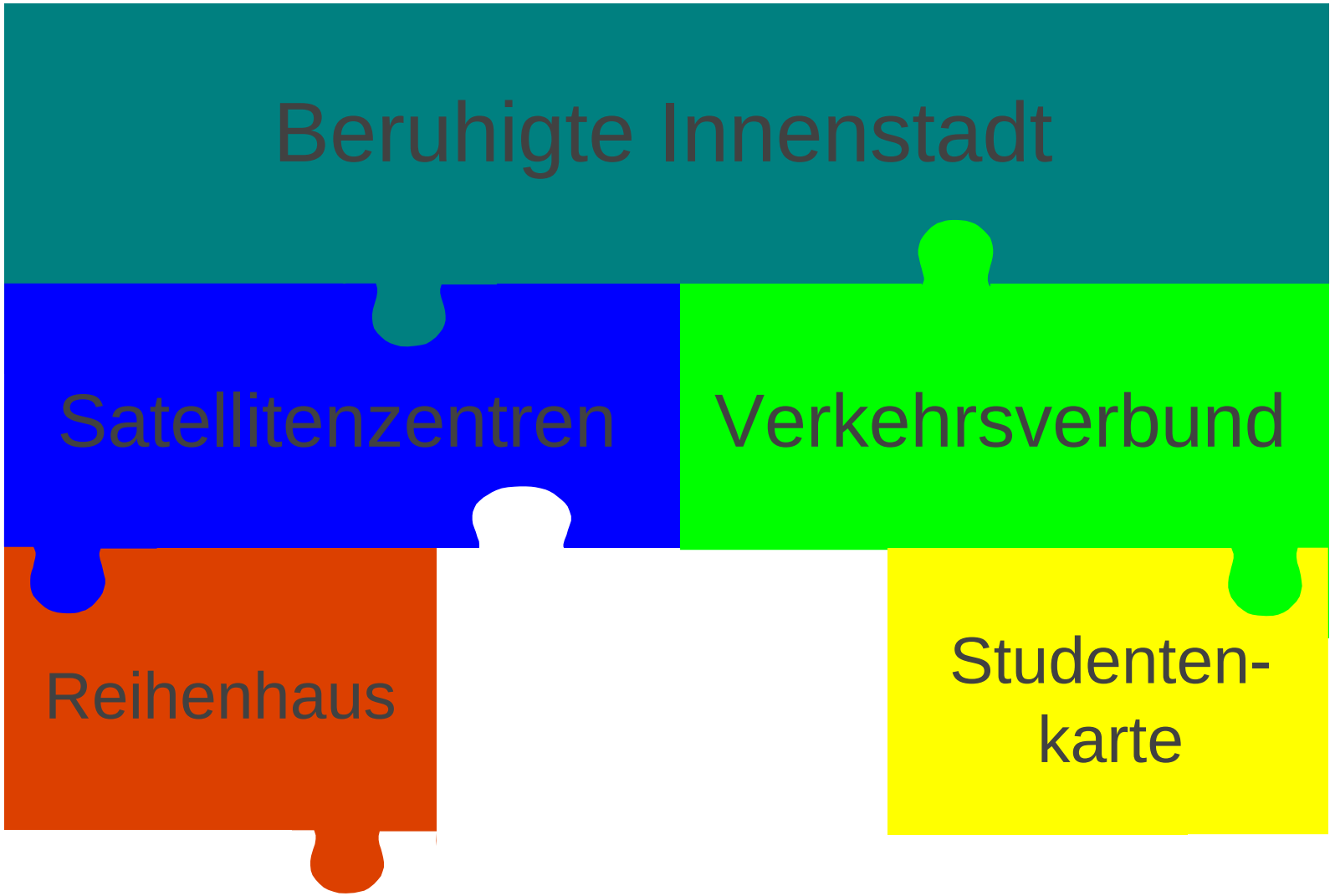
Lösung Wiederholung eine Note höher

Mustersprachen

Beruhigte Innenstadt

Mustersprachen

Mustersprachen



Literatur

- Design Patterns, E. Gamma et al.,
- Pattern-Oriented Software Architecture, Buschmann et. al.
- Pattern Languages of Program Design, PLoP & EuroPLoP conferences (4 vols.)
- Patterns Home Page & Mailing List
  http://hillside.net/patterns
  patterns-discussion@cs.uiuc.edu

Der Wert des Entwurfs

If you reuse code,
You'll save a load,
but if you reuse design,
Your future will shine.

Ralph E. Johnson.