

UML: Unified Modeling Language

Hannes Federrath / Christian Wolff

Textgrundlage: Vorlesungsskript Federrath, mit Ergänzungen

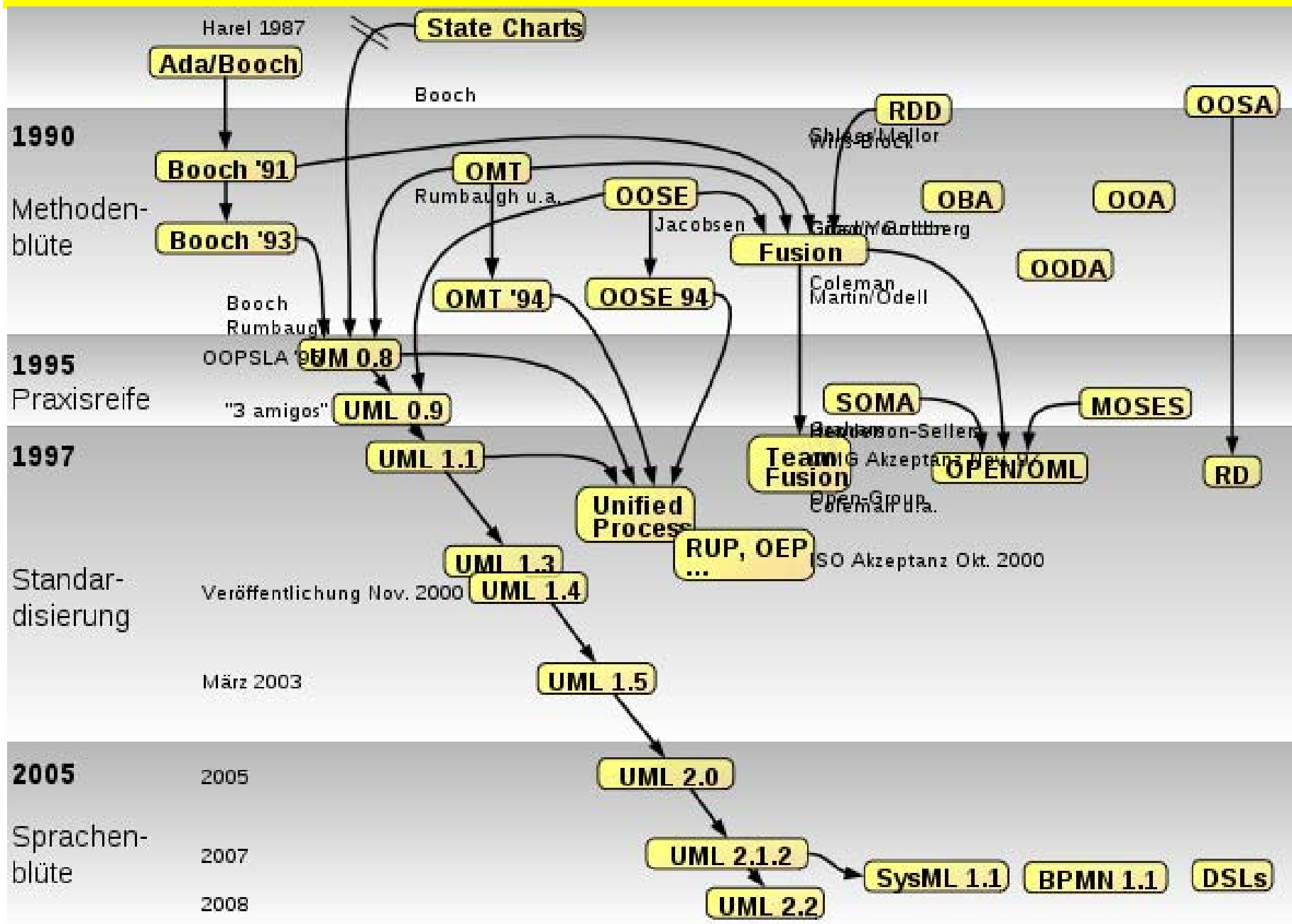
UML: Unified Modeling Language

- ⌘ **Graphische Sprache** zur
Visualisierung, Spezifikation, Konstruktion und Dokumentation
eines softwareintensiven Systems
- ⌘ **Entwicklung**
 - ⊠ ab 1994 entwickelt, 1995 erste Version 0.8
 - ⊠ Zusammenführung verschiedener objektorientierter Methoden
 - ⊕ Beispiele: Booch-Methode, OOSE (Object-Oriented Software Engineering) von Jacobsons, OMT (Object Modeling Technique) von Rumbaugh, ...
 - ⊠ 1996 Gründung des UML-Konsortiums (DEC, HP, IBM, Microsoft, Oracle, Rational, ...) und Vorstellen der Version 1.0
 - ⊠ ab 1997 Standardisierung durch die OMG (Object Management Group)
 - ⊠ 2005: UML 2.0
 - ⊠ zahlreiche „Abspaltungen“ für Spezialbereiche, z. B. SysML, BPMN
- ⌘ **Leseempfehlung**
 - ⊠ Booch, Rumbaugh, Jacobson: Das UML-Benutzerhandbuch. Addison-Wesley-Longman, Bonn 1999.

Historie der objektorientierten Methoden und Notationen

(Quelle: Axel Scheithauer, oose.de, online: <http://upload.wikimedia.org/wikipedia/commons/0/0b/OO-historie.svg>,

Zugriff 01 / 09)



UML als Modellierungshilfsmittel

⌘ UML 2: 13 Typen von Diagrammen

⌘ grundlegende Aufteilung:

- ⊗ Darstellung der **Struktur** (Klasse, Objekt, Paket, Komponente, Verteilung)
- ⊗ Darstellung des **Verhaltens** (einer Anwendung)
 - ⊕ Einsatzszenarien (use cases)
 - ⊕ Zustände des Systems
 - ⊕ Aktivitäten
 - ⊕ Interaktion (Sequenz, Kommunikation)
 - ⊕ Zusammenarbeit / Kollaboration
 - ⊕ Zeitverhalten (timing)

⌘ UML als visuelle Sprache

- ⊗ visuelle graphische/diagrammatische Darstellungsmittel
- ⊗ **wenige** graphische Primitive für den Diagrammaufbau
- ⊗ eindeutige Definition von Syntax und Semantik der Diagramme
- ⊗ Beispielprimitive
 - ⊕ einfache geometrische Formen (Kreis, Ellipse, Rechteck)
 - ⊕ Linien und Pfeile
 - ⊕ Linienmuster
 - ⊕ Texte / Label

Sprachelemente der UML

⌘ Dinge

- ⊗ Strukturelemente
 - ⊕ Klassen
 - ⊕ Schnittstellen
 - ⊕ Kollaborationen
 - ⊕ Anwendungsfälle
 - ⊕ Aktive Klassen
 - ⊕ Komponenten
 - ⊕ Knoten
- ⊗ Verhaltensweisen
 - ⊕ Nachrichten
 - ⊕ Zustände
- ⊗ Gruppierungen
 - ⊕ Pakete
- ⊗ Kommentare
 - ⊕ Notizen

⌘ Beziehungen

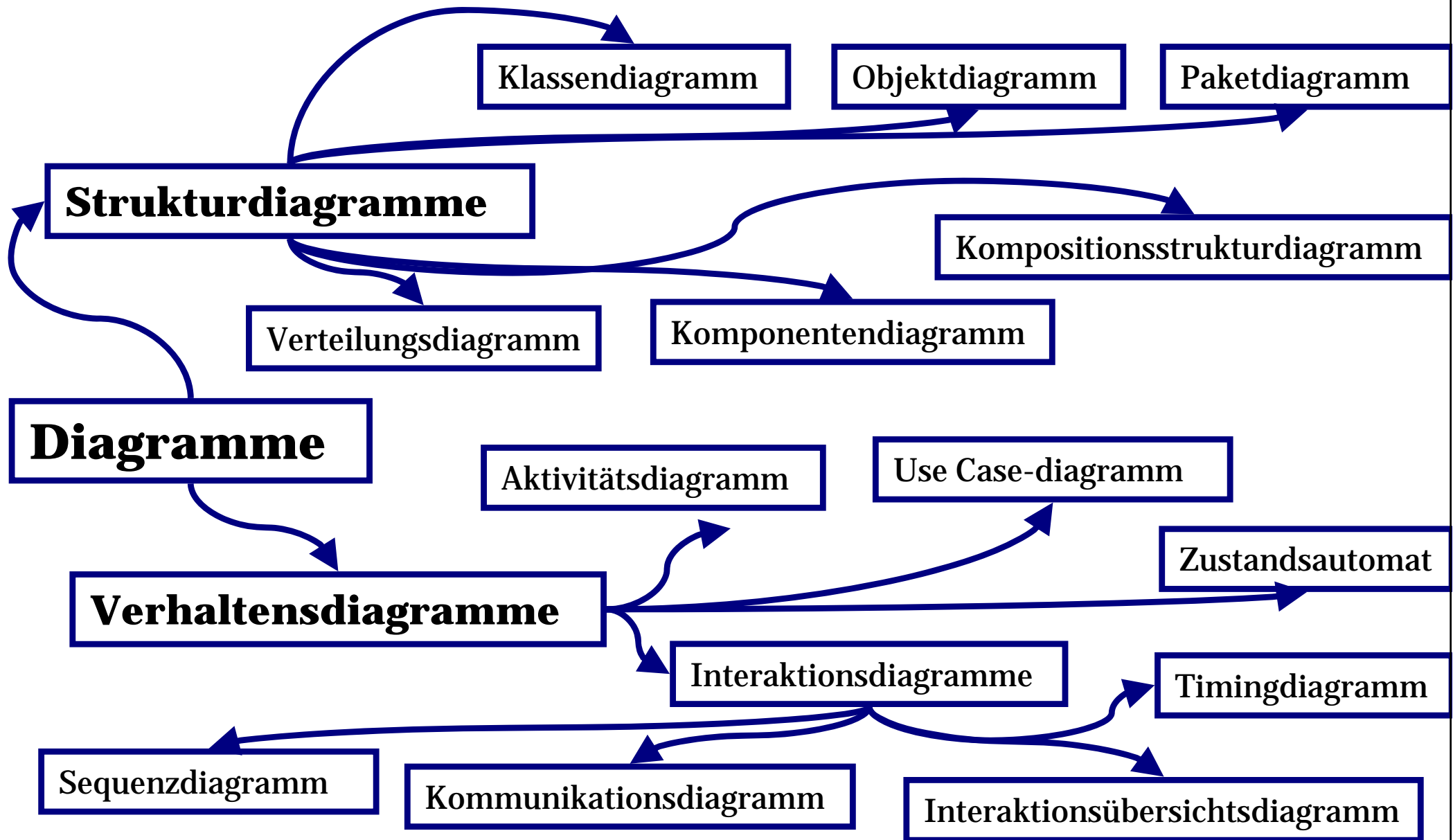
- ⊗ Abhängigkeiten
- ⊗ Assoziationen
- ⊗ Generalisierungen
- ⊗ Realisierungen

⌘ Diagramme

- ⊗ Klassendiagramm
- ⊗ Objektdiagramm
- ⊗ Anwendungsfalldiagramm
- ⊗ Sequenzdiagramm
- ⊗ Kollaborationsdiagramm
- ⊗ Zustandsdiagramm
- ⊗ Aktivitätsdiagramm
- ⊗ Komponentendiagramm
- ⊗ Einsatzdiagramm
- ⊗ ...

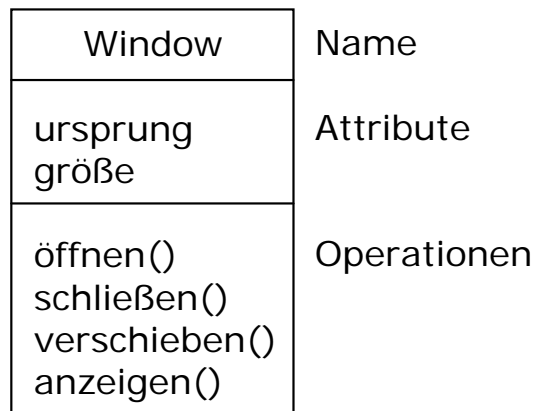
Diagrammtypen in UML

Strukturdiagramme und Verhaltensdiagramme



⌘ Strukturelemente (1/3)

☒ Klassen



Details sind ebenfalls modellierbar
(siehe später).

☒ Schnittstellen



Schnittstellen werden tlw. auch als sog.
stereotypisierte Klassen dargestellt (siehe
später).

⌘ Strukturelemente (2/3)

☒ Kollaborationen



definiert eine Interaktion, besteht aus einer Gruppe von Rollen und anderen Elementen, die zusammenwirken, um kooperatives Verhalten hervorzubringen

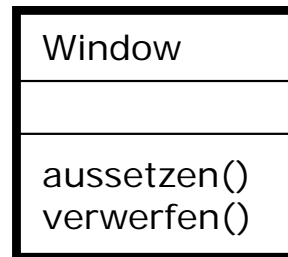
☒ Anwendungsfälle



Menge von aufeinanderfolgenden Aktionen, wird durch eine Kollaboration realisiert

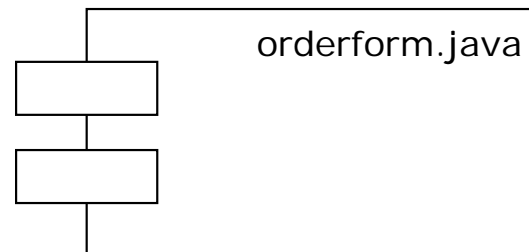
⌘ Strukturelemente (3/3)

☒ Aktive Klassen



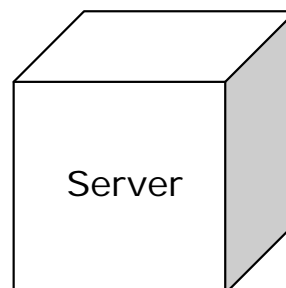
setzen Steuerungsaktivitäten in Gang, besitzen einen oder mehrere Prozesse oder Threads, Verhalten ist nebenläufig zu anderen Klassen

☒ Komponenten



ein physischer, austauschbarer Teil eines Systems, der sich einer Menge von Schnittstellen anpasst und ihrer Realisierung dient

☒ Knoten



ein zur Laufzeit vorhandenes physisches Element, das eine Rechnerressource (mit Speicher und ggf. Rechenkapazität) repräsentiert

⌘ Verhaltensweisen

☒ Nachrichten

anzeigen →

☒ Zustände

Warten

⌘ Gruppierungen

☒ Pakete

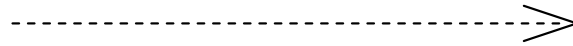
Geschäftsregeln

⌘ Kommentare

☒ Notizen

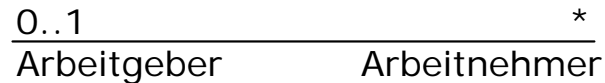
Gebe Kopie
Deiner selbst
zurück

⌘ Abhängigkeiten

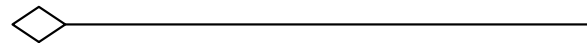


semantische Beziehung zwischen zwei Gegenständen, die Änderung des unabhängigen G. beeinflusst den abhängigen G.

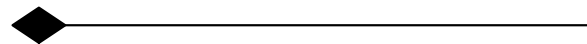
⌘ Assoziationen



Strukturbeziehung, die eine Menge von Objektbeziehungen beschreibt

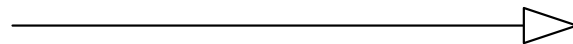


Ganzes-/Teil-Beziehung



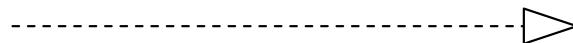
Ganzes-/Teil-Beziehung mit starker Eigentümerschaft des Ganzen

⌘ Generalisierungen



beschreibt eine Beziehung zwischen einem Allgemeinen und Speziellen

⌘ Realisierungen



semantische Beziehung zwischen Klassifizierungen, in der die eine Klassifizierung einen Vertrag spezifiziert, für dessen Ausführung die andere garantiert, z.B. zwischen Schnittstelle und ausführender Klasse

⌘ Klassendiagramm

- ⊠ statische Beziehungen zwischen Klassen, Schnittstellen, Kollaborationen

⌘ Objektdiagramm

- ⊠ statische Beziehungen zwischen Objekten

⌘ Anwendungsfalldiagramm

- ⊠ statische Beziehungen zwischen Akteuren (spezielle Art von Klasse)

⌘ Sequenzdiagramm

- ⊠ Interaktionsdiagramm, dynamische Sicht auf zeitliche Abfolge von Nachrichten zwischen Objekten

⌘ Kollaborationsdiagramm

- ⊠ Interaktionsdiagramm, dynamische Sicht auf strukturelle Organisation von Objekten und deren Nachrichtenaustausch

⌘ Zustandsdiagramm

- ⊠ beschreibt einen Automaten, bestehend aus Zuständen, Zustandsübergängen, Ereignissen und Aktivitäten, dynamische Sicht

⌘ Aktivitätsdiagramm

- ⊠ spezielles Zustandsdiagramm, zeigt den Fluss von einer Aktivität zur anderen innerhalb des Systems (Kontrollfluss)

⌘ Komponentendiagramm

- ⊠ zeigt die Organisation und die Abhängigkeiten einer Menge von Komponenten

⌘ Einsatzdiagramm

- ⊠ zeigt Konfiguration der im Prozess befindlichen Knoten und ihre existierenden Komponenten

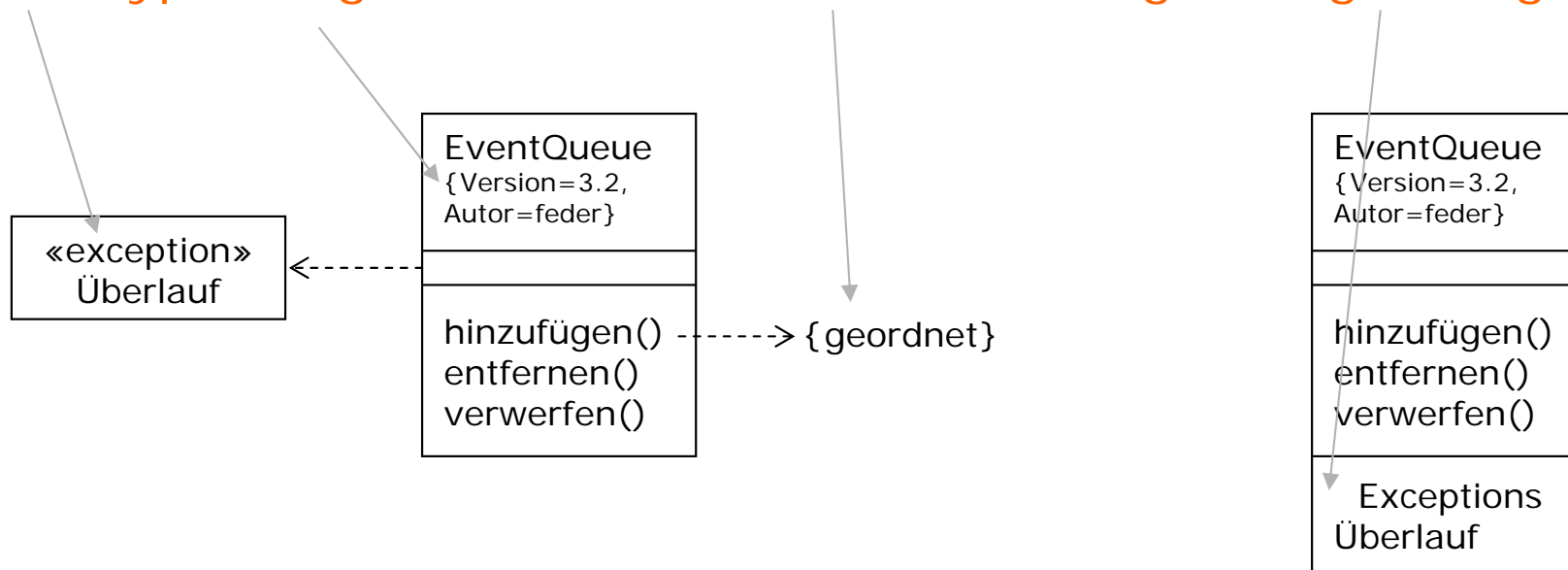
⌘ weitere Diagrammformen sind möglich

Details sind ebenfalls modellierbar

⌘ Unterscheidung Klasse — Objekt

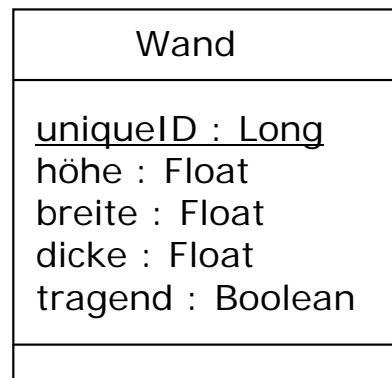


⌘ Stereotypen, Eigenschaftswerte, Einschränkungen, Ergänzungen



Details sind ebenfalls modellierbar

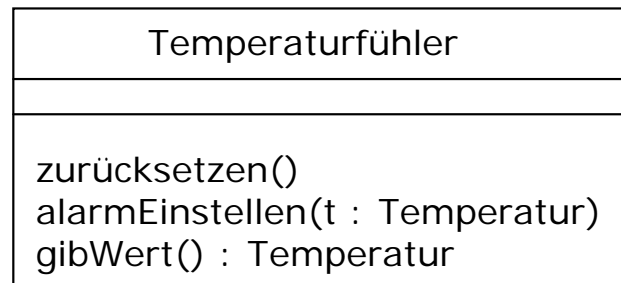
- ⌘ Attribute und Operationen können durch Typangaben ergänzt werden



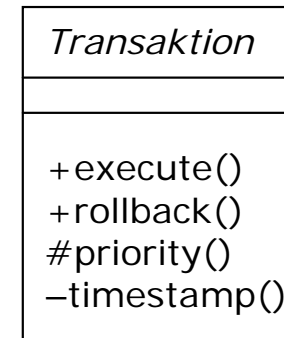
unterstrichen: Klassenvariable bzw. Klassenmethode

<Name> ":" <Typ>

- ⌘ Angabe der Signaturen bei Operationen



- ⌘ Abstrakte Klassen, Sichtbarkeit



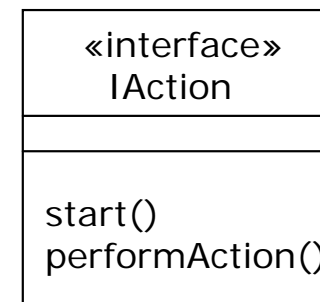
kursiv: abstrakte KI.

+: public

#: protected

–: private

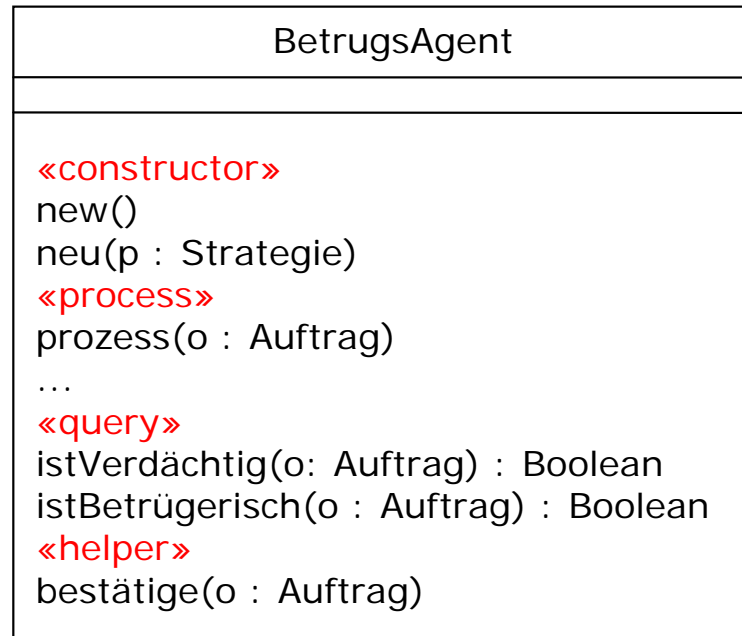
- ⌘ Interfaces



Stereotyp

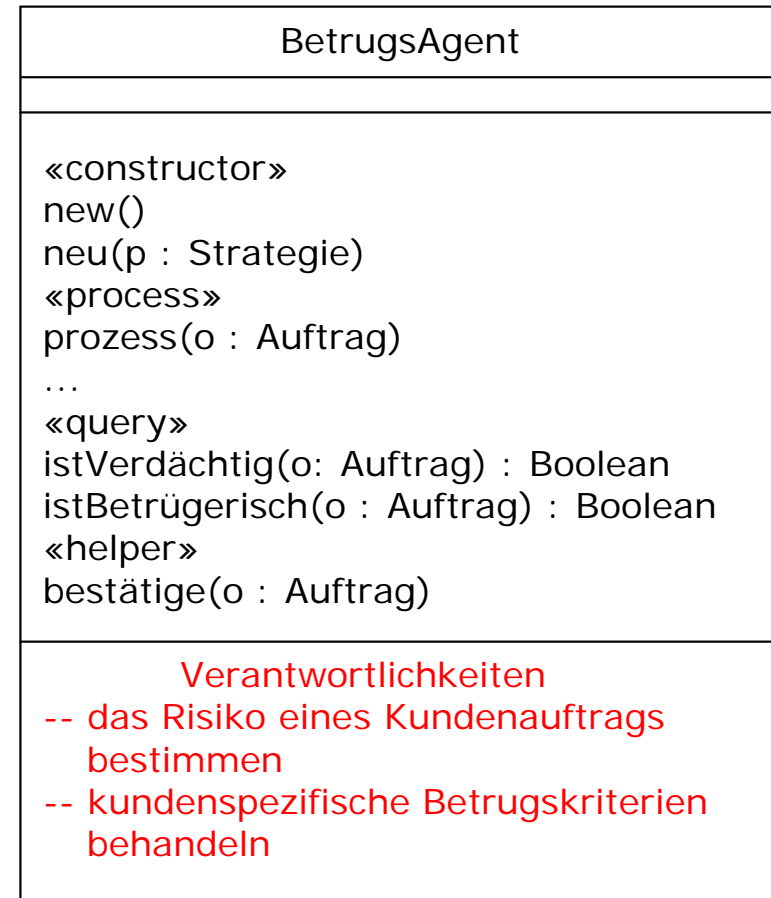
Details sind ebenfalls modellierbar

⌘ Attribute und Operationen gruppieren



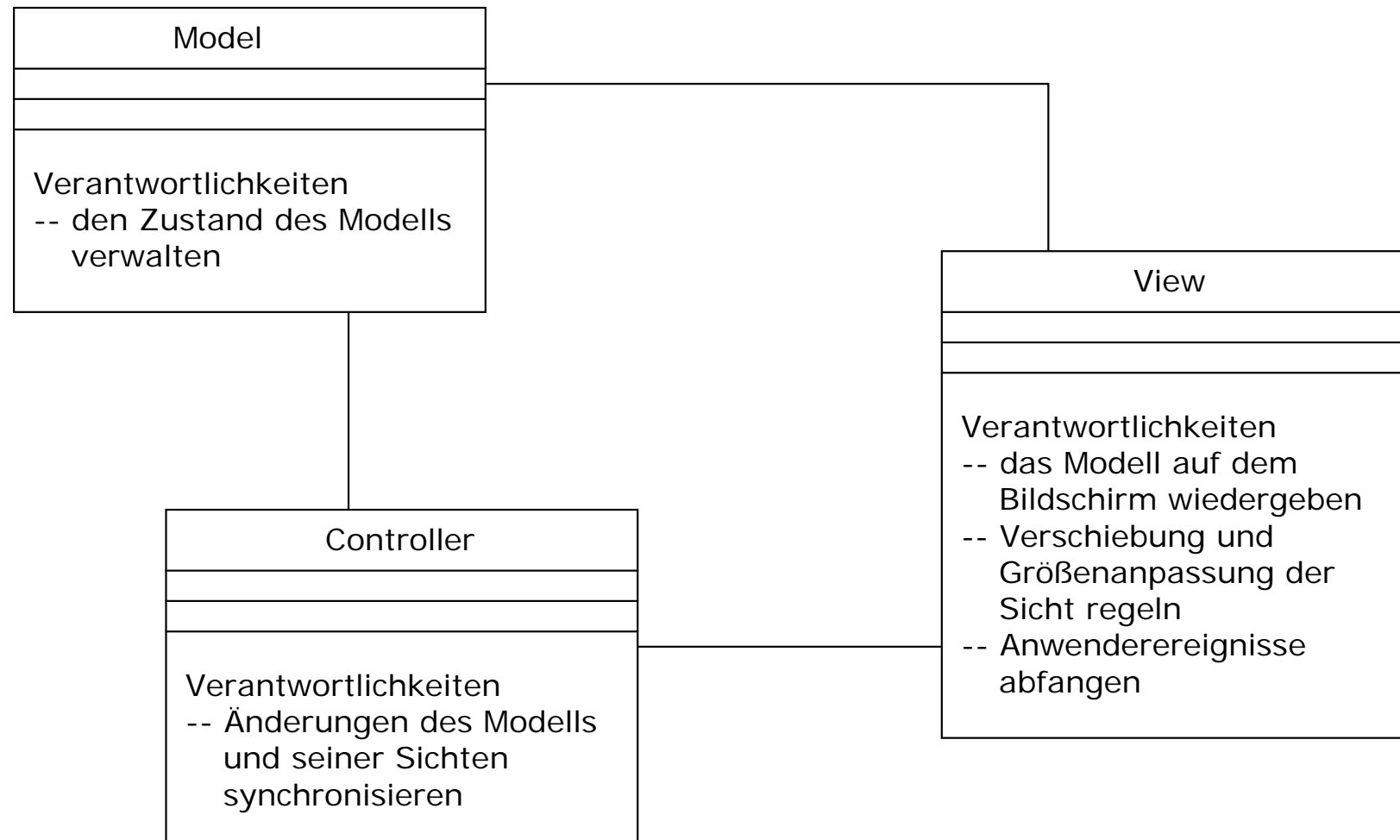
Zur Gruppierung werden Stereotype verwendet.

⌘ Verantwortlichkeiten



Verwendung von Notizen ist alternativ möglich, um Verantwortlichkeiten darzustellen

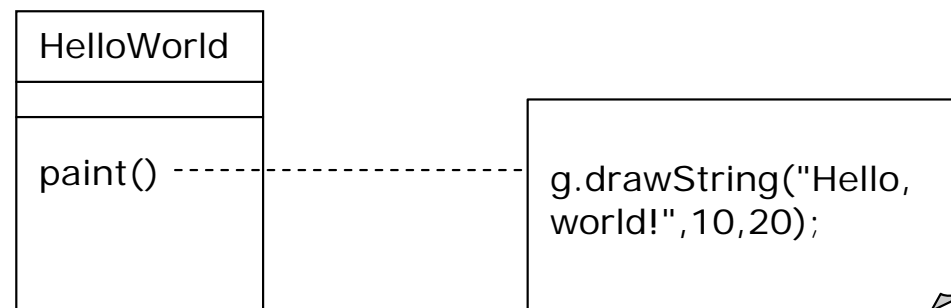
Beispiel: Verantwortlichkeiten beim MVC-Konzept



⌘ HelloWorld- Applet

```
import java.awt.Graphics;  
import java.applet.Applet;  
public class HelloWorld extends Applet {  
    public void paint(Graphics g) {  
        g.drawString("Hello, world!",10,20);  
    }  
}
```

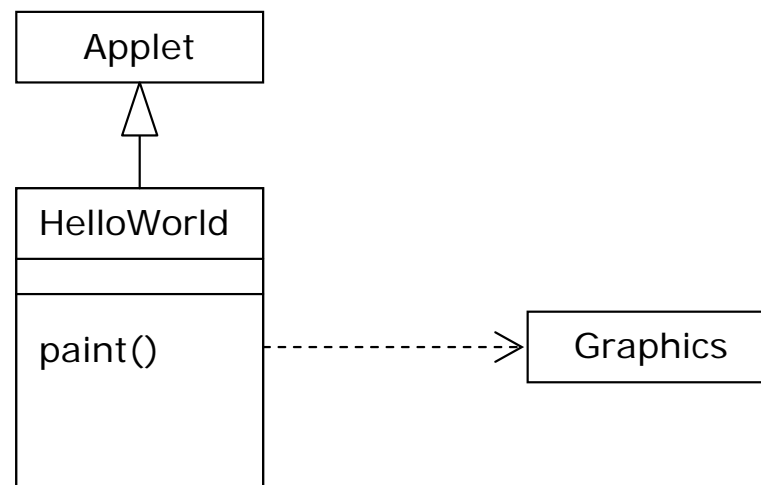
⌘ Abstraktionen



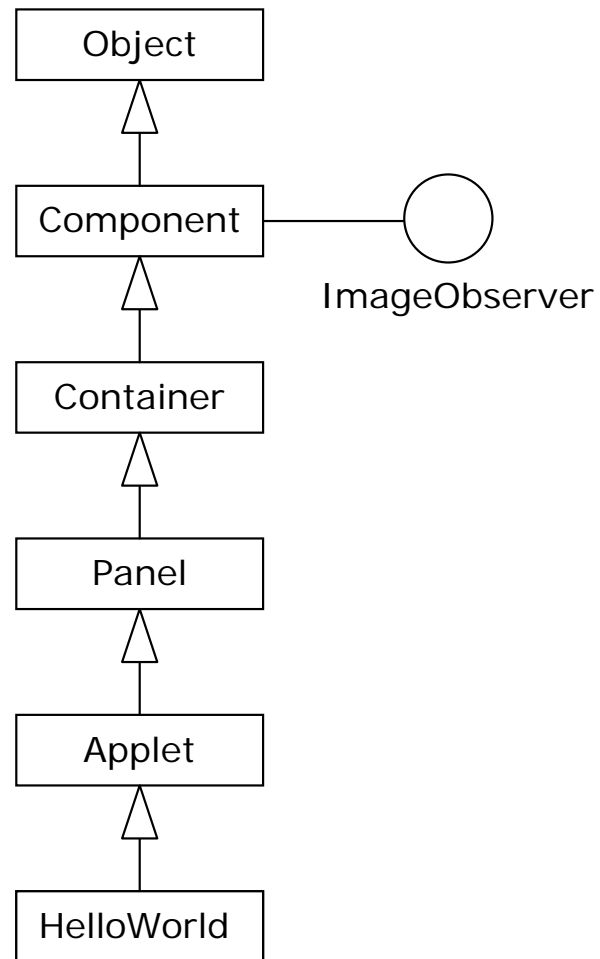
⌘ HelloWorld- Applet

```
import java.awt.Graphics;  
import java.applet.Applet;  
public class HelloWorld extends Applet {  
    public void paint(Graphics g) {  
        g.drawString("Hello, world!",10,20);  
    }  
}
```

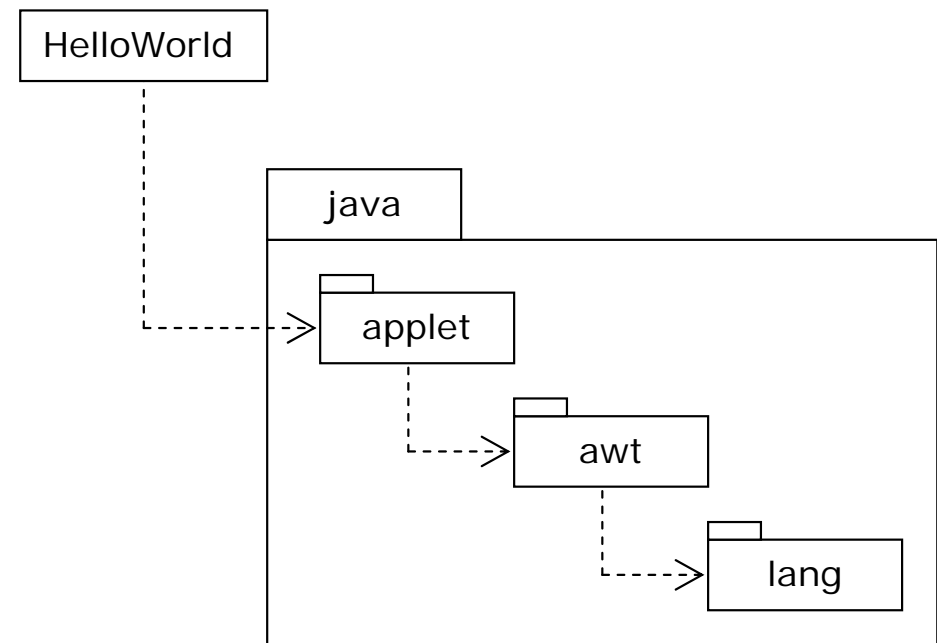
⌘ Unmittelbare Nachbarn



⌘ Vererbungshierarchie



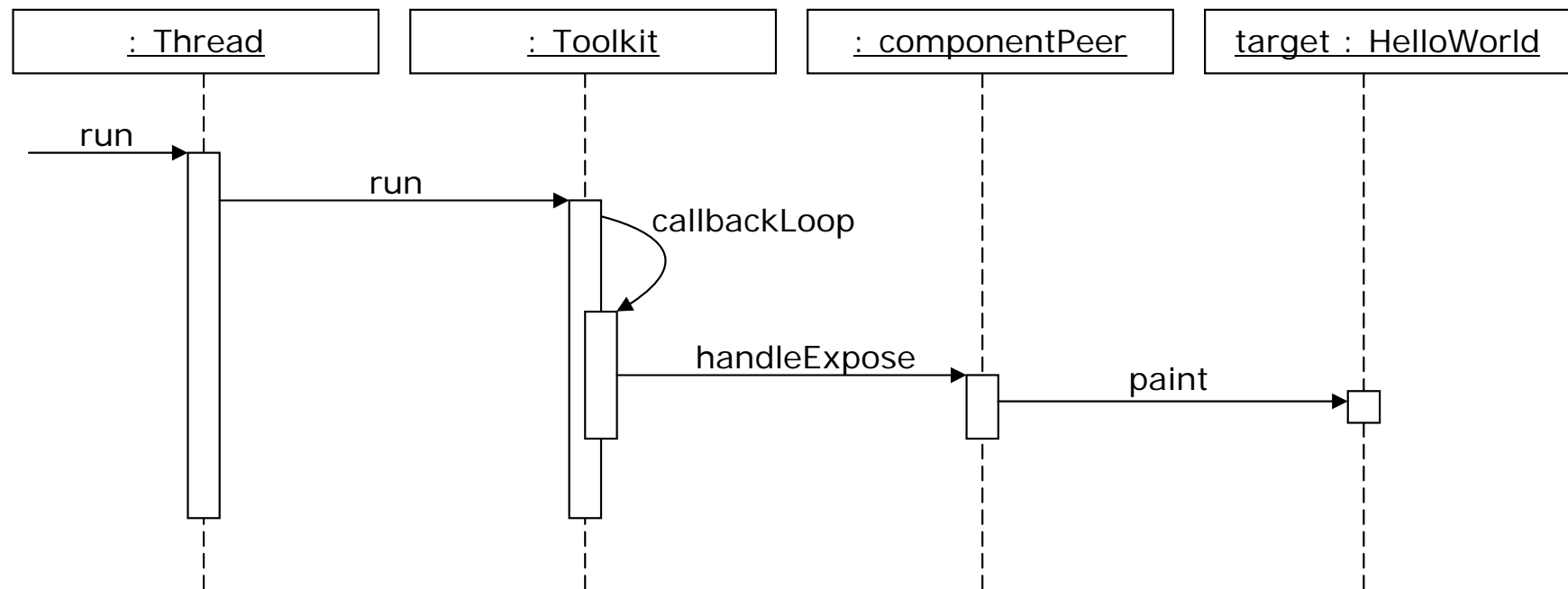
⌘ Paketstruktur



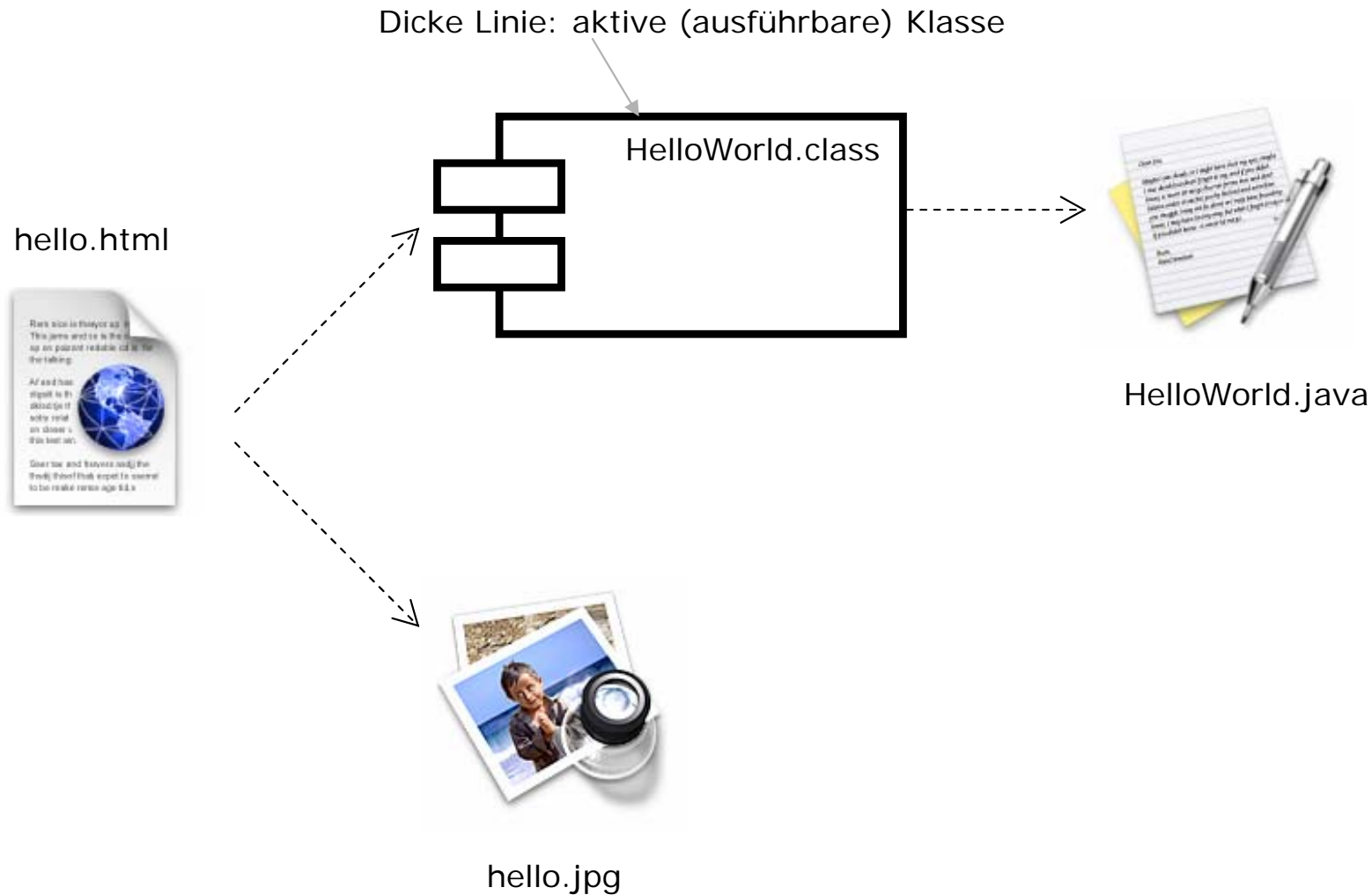
Paint-Mechanismus als Sequenzdiagramm

⌘ Applet wird in einem Thread von der JVM gestartet

Anonyme Objekte
(haben keinen
eindeutigen
Namen)



Komponenten von Hello World

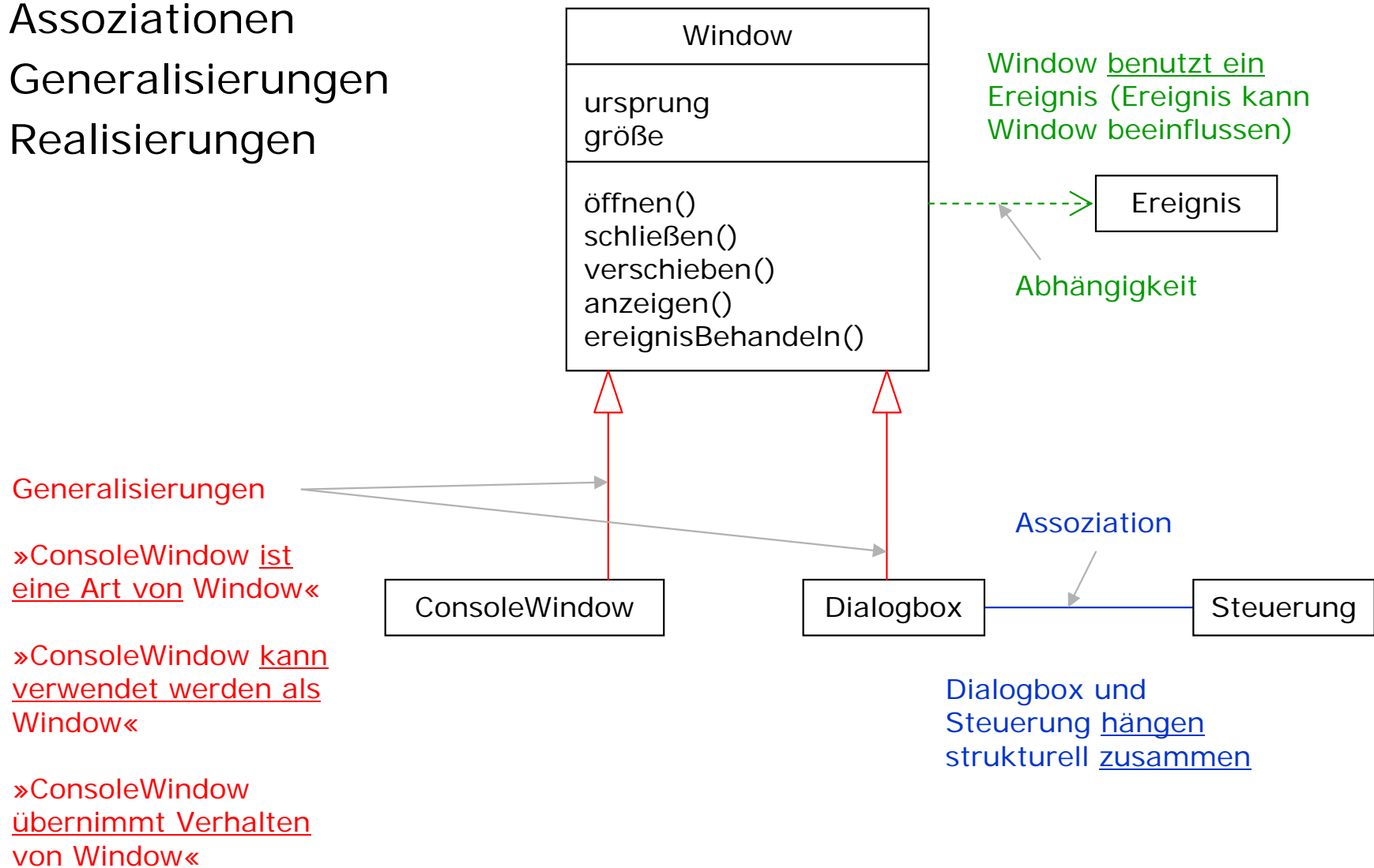


Ende des HelloWorld-Beispiels.

Beispiel: Beziehungen

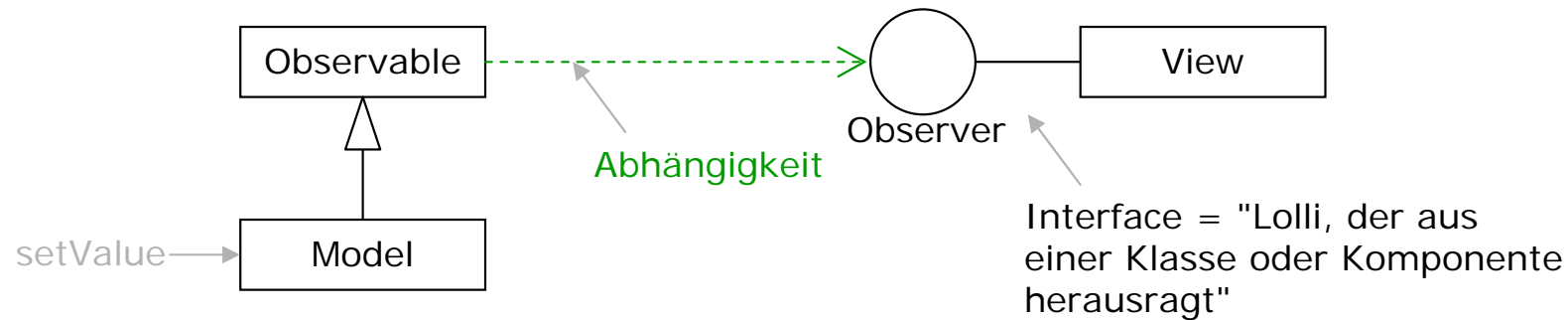
⌘ Beziehungen:

- ☒ Abhängigkeiten
- ☒ Assoziationen
- ☒ Generalisierungen
- ☒ Realisierungen

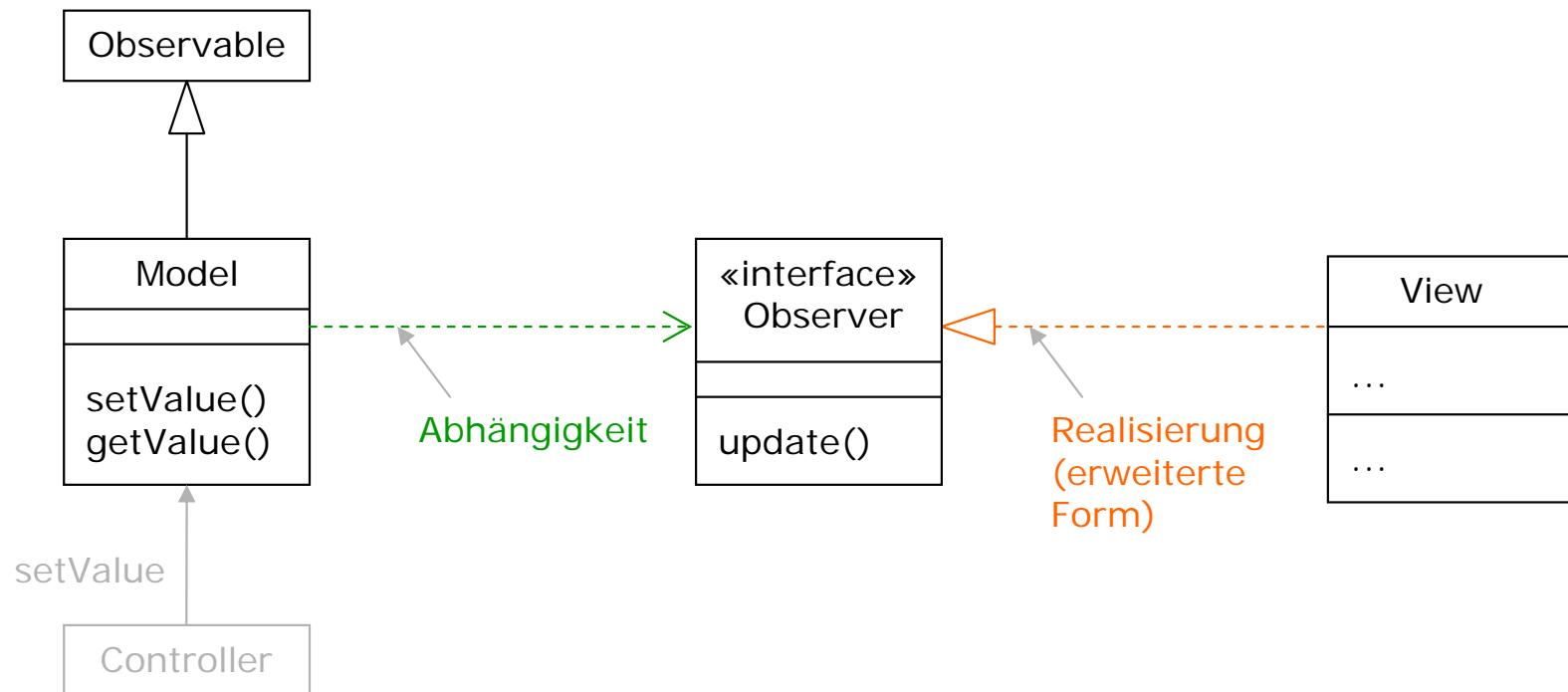


Beispiel: Beziehungen

⌘ Observer-Konzept

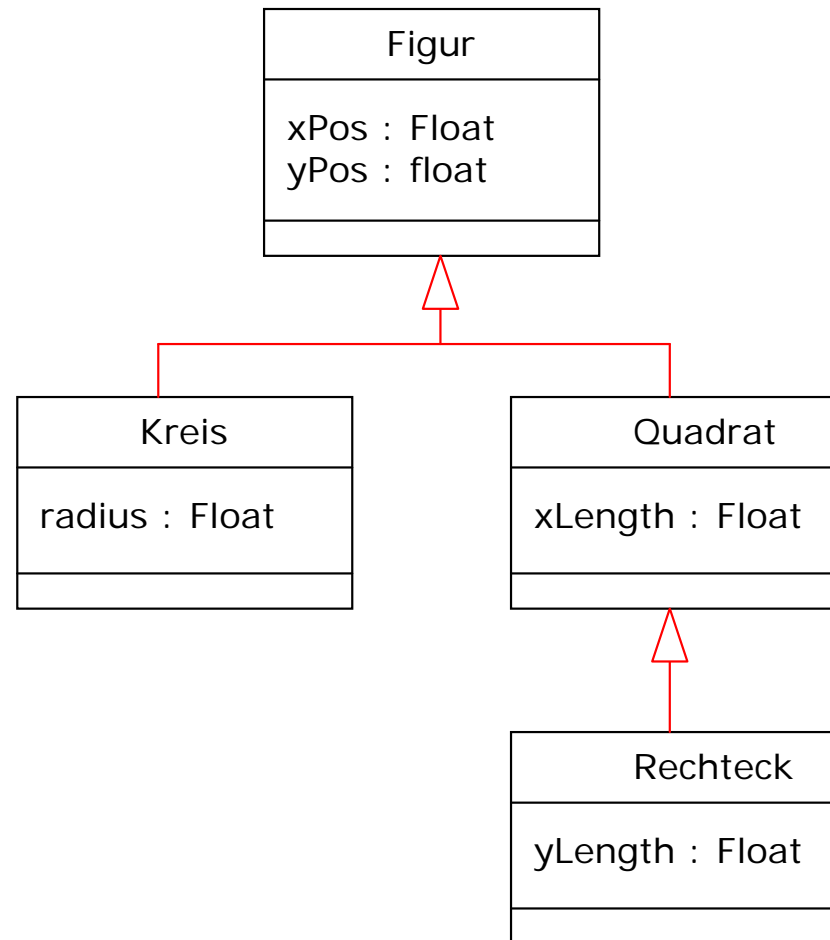


⌘ Erweiterte Form mit Interface als stereotypisierte Klasse

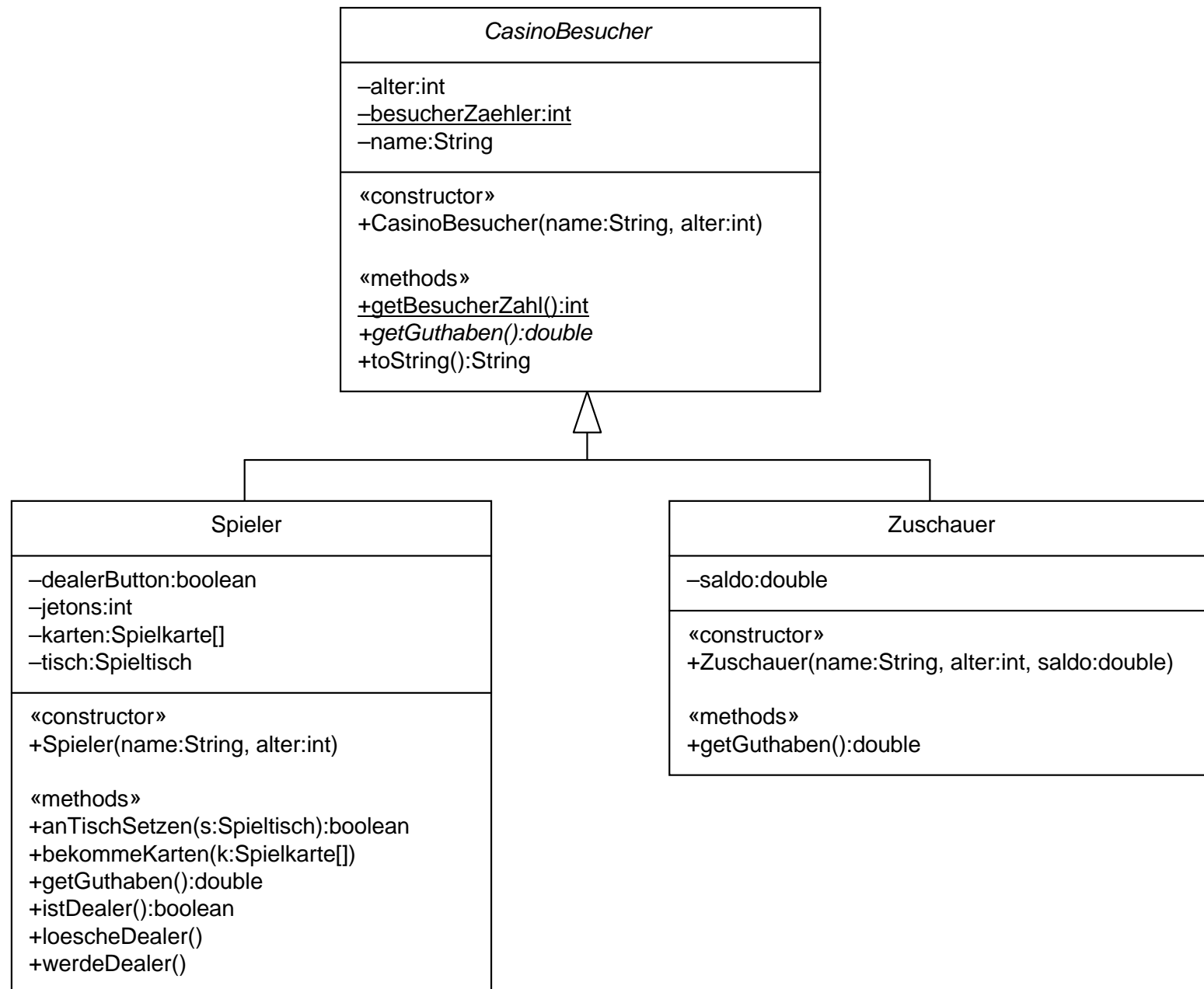


Beispiel: Einfachvererbung

⌘ Anwendung der Generalisierungsbeziehung

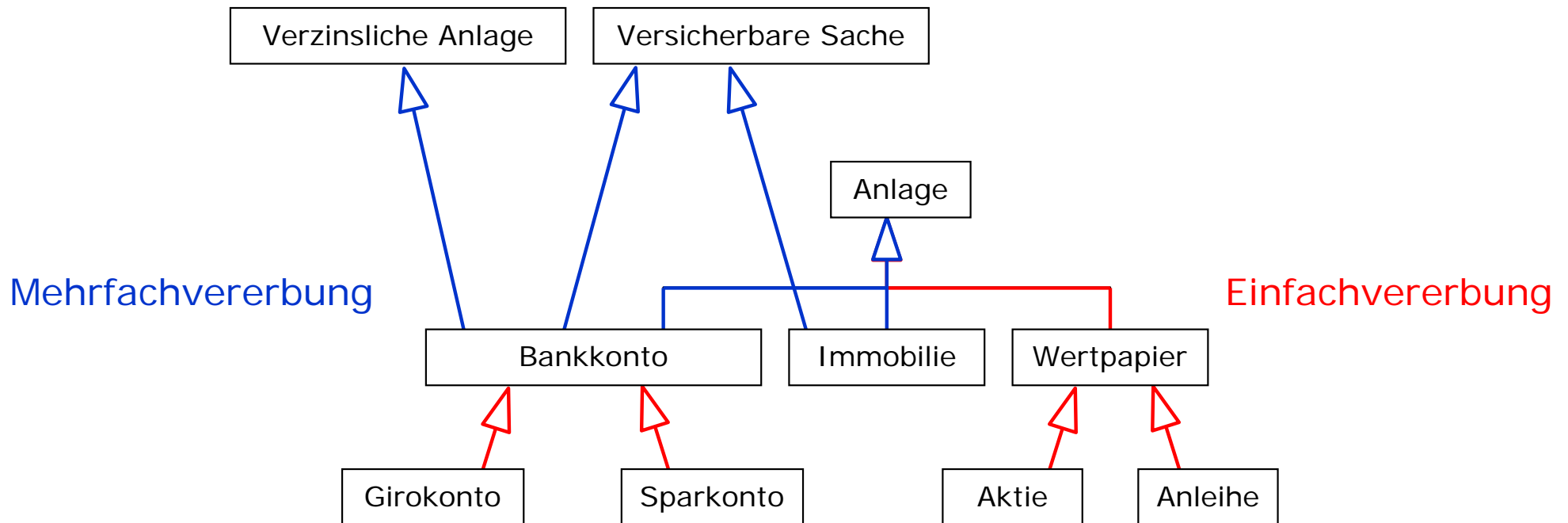


Klausuraufgabe OOP Wintersemester 2006/2007



Einfach- und Mehrfachvererbung

⌘ Anwendung der Generalisierungsbeziehung

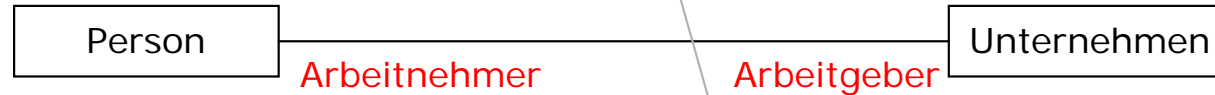


Assoziationen sind universell

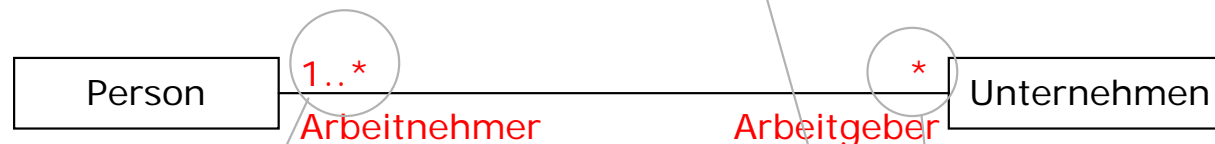
⌘ können Namen und Namensichtung tragen



⌘ können Rollenangaben tragen



⌘ können Angaben zur Multiplizität tragen

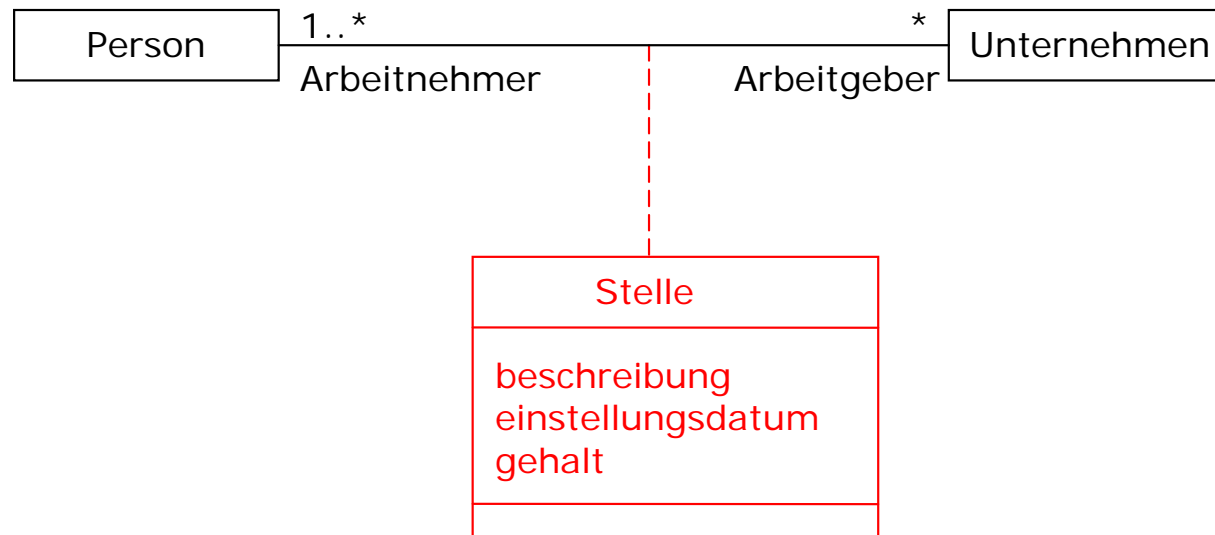


Ein Unternehmen hat 1..* Arbeitnehmer.

Eine Person arbeitet für * Unternehmen.

Assoziationsklassen

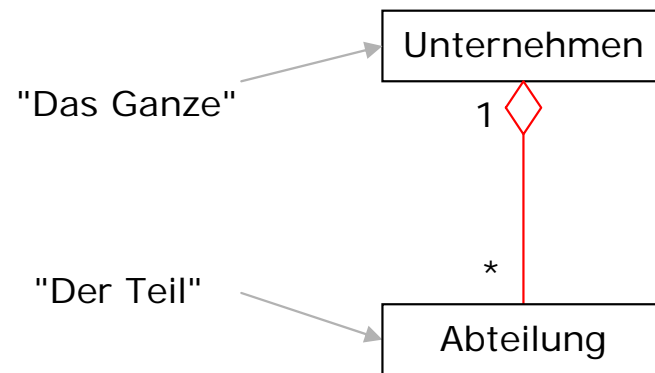
⌘ Assoziationen können über Assoziationsklassen modelliert werden



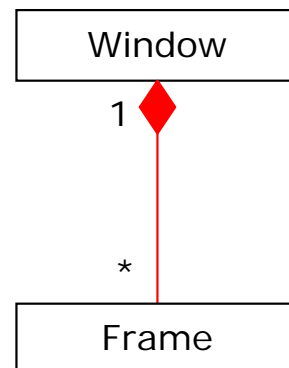
Spezielle Assoziationen

⌘ Normale Assoziationen: Beziehungen zwischen Gleichberechtigten

⌘ **Aggregation**: Modellierung einer Ganzes/Teil-Beziehung



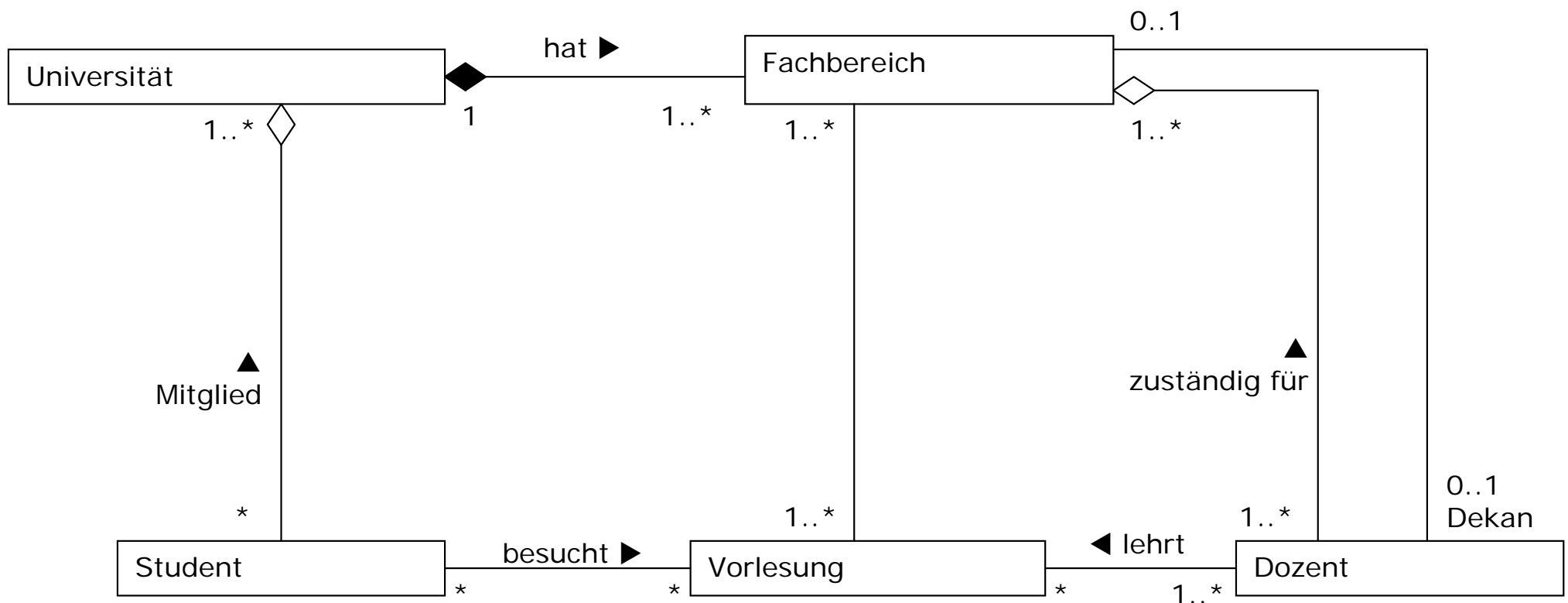
⌘ **Komposition** (spezielle Form der Aggregation): Ausgeprägte Eigentümerschaft des "Ganzen"



Das Frame kann zu einer bestimmten Zeit nur Teil eines Windows sein. Beim Schließen eines Windows muss es dafür sorgen, dass seine Frames zerstört werden. Ein Frame kann ohne ein Window nicht existieren

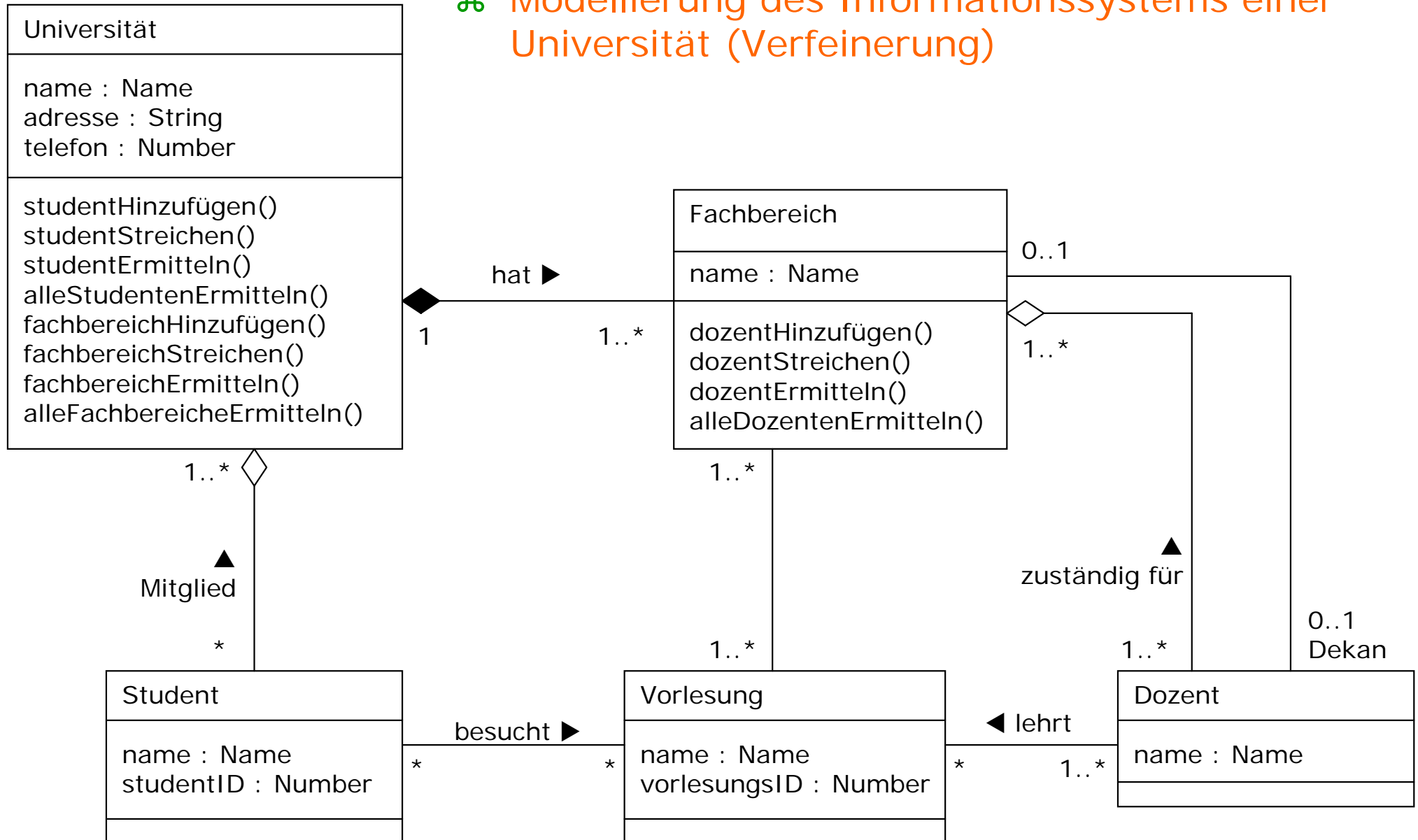
Beispiel: Assoziationen

⌘ Modellierung des Informationssystems einer Universität



Beispiel: Assoziationen

⌘ Modellierung des Informationssystems einer Universität (Verfeinerung)



Sprachelemente der UML

⌘ Dinge

- ⊗ Strukturelemente
 - ⊕ Klassen
 - ⊕ Schnittstellen
 - ⊕ Kollaborationen
 - ⊕ Anwendungsfälle
 - ⊕ Aktive Klassen
 - ⊕ Komponenten
 - ⊕ Knoten
- ⊗ Verhaltensweisen
 - ⊕ Nachrichten
 - ⊕ Zustände
- ⊗ Gruppierungen
 - ⊕ Pakete
- ⊗ Kommentare
 - ⊕ Notizen

⌘ Beziehungen

- ⊗ Abhängigkeiten
- ⊗ Assoziationen
- ⊗ Generalisierungen
- ⊗ Realisierungen

⌘ Diagramme

- ⊗ Klassendiagramm
- ⊗ Objektdiagramm
- ⊗ Anwendungsfalldiagramm
- ⊗ Sequenzdiagramm
- ⊗ Kollaborationsdiagramm
- ⊗ Zustandsdiagramm
- ⊗ Aktivitätsdiagramm
- ⊗ Komponentendiagramm
- ⊗ Einsatzdiagramm

UML: Modellieren einer Systemarchitektur

Vokabular
Funktionalität

Systemzusammensetzung
Konfigurationsmanagement

