Dem traditionellen Ansatz nach wird ein Entwurf des zu erstellenden Systems auf der Basis des Pflichtenheftes vorgenommen.

Diese Herangehensweise führt zur konventionellen Programmierung, weil keine Klassen oder Objekte im Sinne von OOP in dem Pflichtenheft dargestellt sind. In dem Pflichtenheft fehlt die Verknüpfung zwischen den Daten und Operationen.

Das Pflichtenheft spielt dabei trotzdem eine wichtige Rolle als:

- ⇨ Zusammenfassung aller Wünsche des Auftraggebers
- ⇨ gemeinsames Verständnis der Arbeitsabläufe und der Informationen
- ⇨ Grund für Vertrag

Im Gegenteil - die objektorientierte Analyse gewährleistet einen logischen (und teilweise automatisierten) Übergang zur OOP.

Die **Quellen** für die OOA:

- ⇨ Pflichtenheft
- ⇨ Dokumente des Anwendungsbereiches

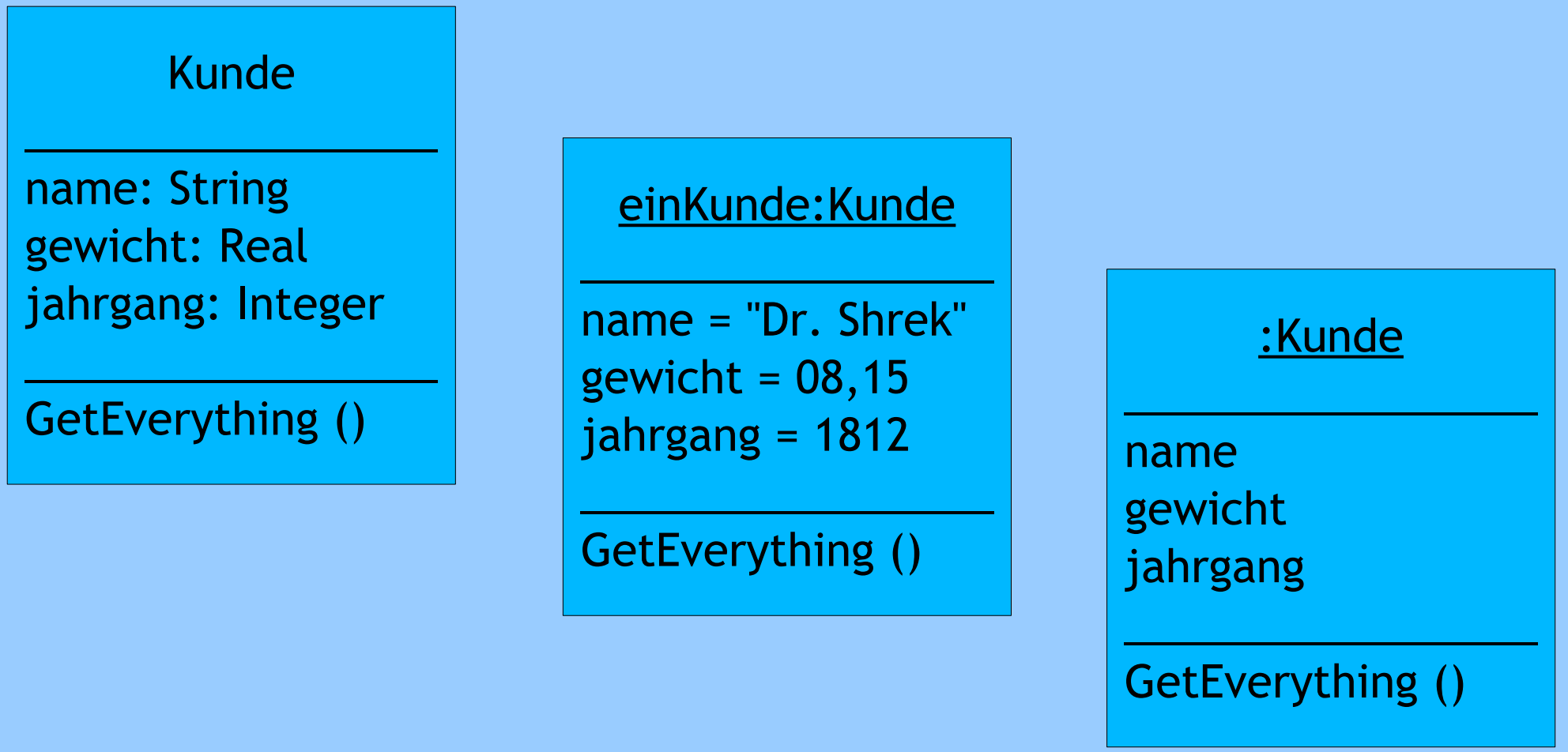
Ziel der OOA ist es, die Objekte, Klassen, Beziehungen, Assoziationen, Attribute, Operationen auf Basis von diesen Quellen zu identifizieren.

Definition:
Eine Klasse ist die möglichst minimale Zusammenfassung der Attribute und der Operationen, die der geschäftlichen Logik nach zusammen gehören.

Definition:
Ein Objekt ist eine Instanz einer Klasse.

Vergleich:

Klasse – Objekt
Variable – Wert
Entitätstyp – Entität
Beziehungstyp - Beziehung
Allgemein - Konkret

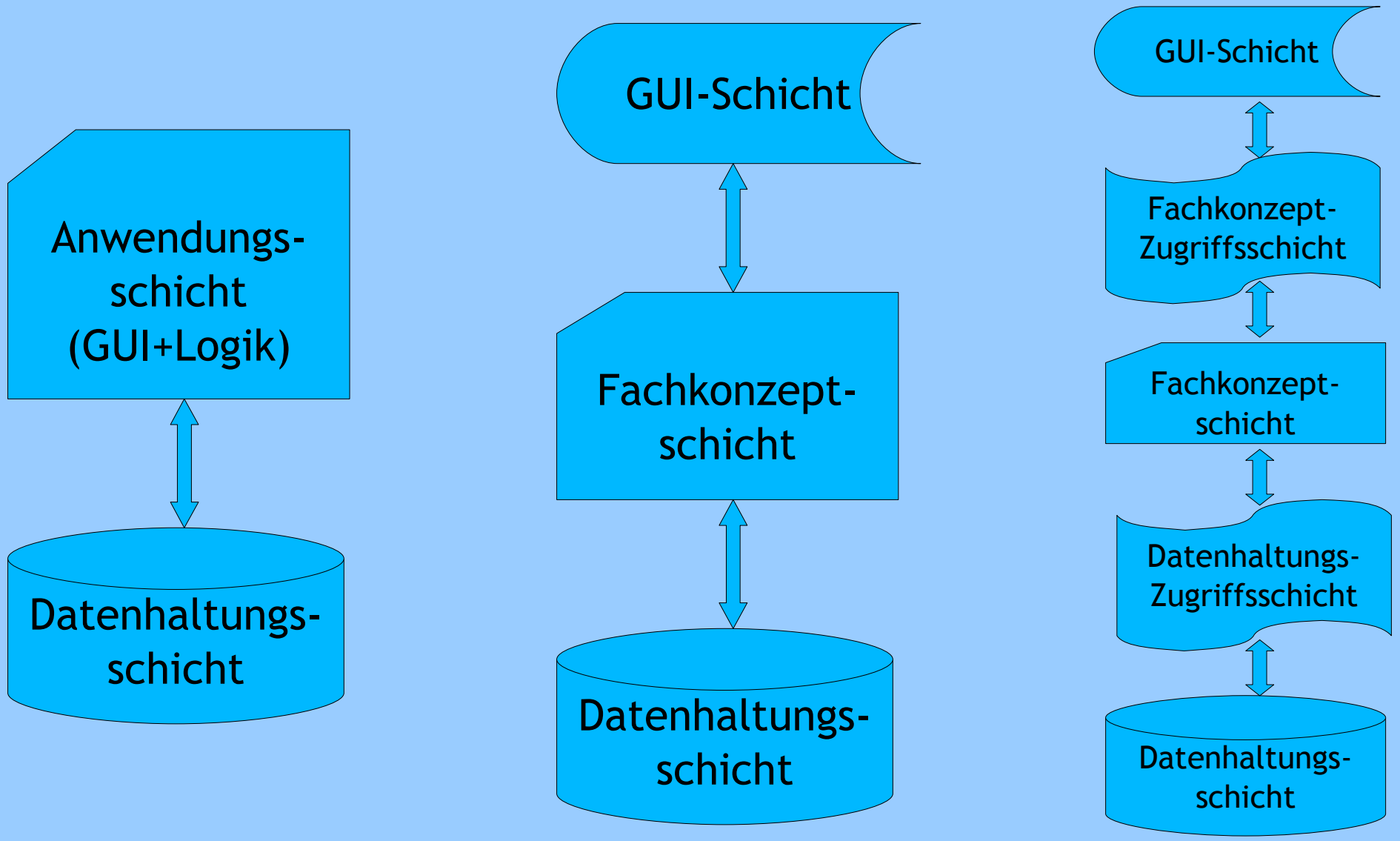

Kunde
name: String
gewicht: Real
jahrgang: Integer
GetEverything ()
einKunde:Kunde
name = "Dr. Shrek"
gewicht = 08,15
jahrgang = 1812
GetEverything ()
:Kunde
name
gewicht
jahrgang
GetEverything ()

Monolithische Anwendungen gelten heute als veraltet. Die Architektur der modernen Softwaresysteme zeichnet sich durch mehrere Schichten aus.

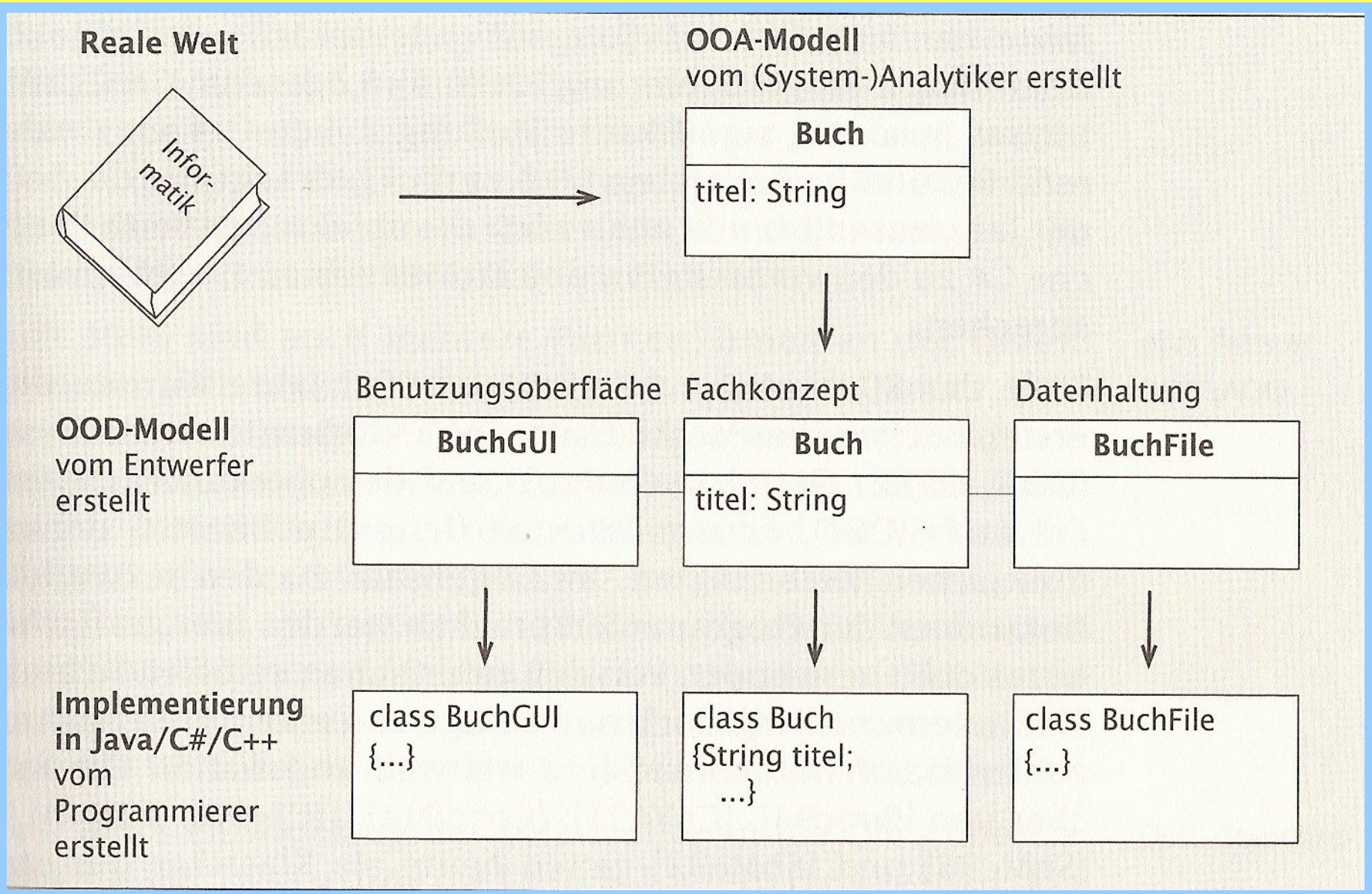
Ziel – Flexibilität. Eine Schicht sollten man austauschen können, ohne die anderen Schichten zu verändern.

Grund – in vielen Systemen sind die anwendungsspezifischen Modulen nach wie vor auf dem hohen Niveau, während die grafische Oberfläche oder die Datenverarbeitungsmethoden den ständig wachsenden Anforderungen nicht mehr entsprechen.

- ⇨ Zwei-Schichten-Architektur
- ⇨ Drei-Schichten-Architektur
- ⇨ Mehr-Schichten-Architektur


Anwendungs-
schicht
(GUI+Logik)
Datenhaltungs-
schicht
GUI-Schicht
Fachkonzept-
schicht
Datenhaltungs-
schicht
GUI-Schicht
Fachkonzept-
Zugriffsschicht
Fachkonzept-
schicht
Datenhaltungs-
Zugriffsschicht
Datenhaltungs-
schicht

OOA, Drei-Schichten-Architektur



Basisbegriffe (1/2)

- „Klasse“ in verschiedener Bedeutung
  - Fachkonzeptklasse
    Eine Abbildung eines Phänomens des betrachteten Weltausschnitts (Miniwelt); Entspricht der Entitätsklasse (Entitätstyp) bei der ER-Modellierung; Es handelt sich um ein Konzept zur *Informationsmodellierung*.
  - Fachklasse
    Eine um Operationen erweiterte Fachkonzeptklasse; Sie stellt einen Plan, eine Vorgabe für ein später zu schaffendes Gebilde, den Programmcode, dar. Es handelt sich um eine Entwurfsklasse, auch „Softwareklasse“ genannt. Weitere Entwurfsklassen sind beispielsweise Dialogklassen und Controllerklassen.
    Eine Fachklasse der OOA ist ein grober Plan, der im Verlaufe des Entwurfs (Schritt nach der OOA) verfeinert wird.
  - Implementierungsklasse
    Eine Klasse, die mit einer objektorientierten Programmiersprache (z. B. Java, C++, C#) realisiert ist; Es handelt sich um eine Ausprägung des Konzepts „Klasse“ dieser objektorientierten Sprache.

Basisbegriffe (2/2)

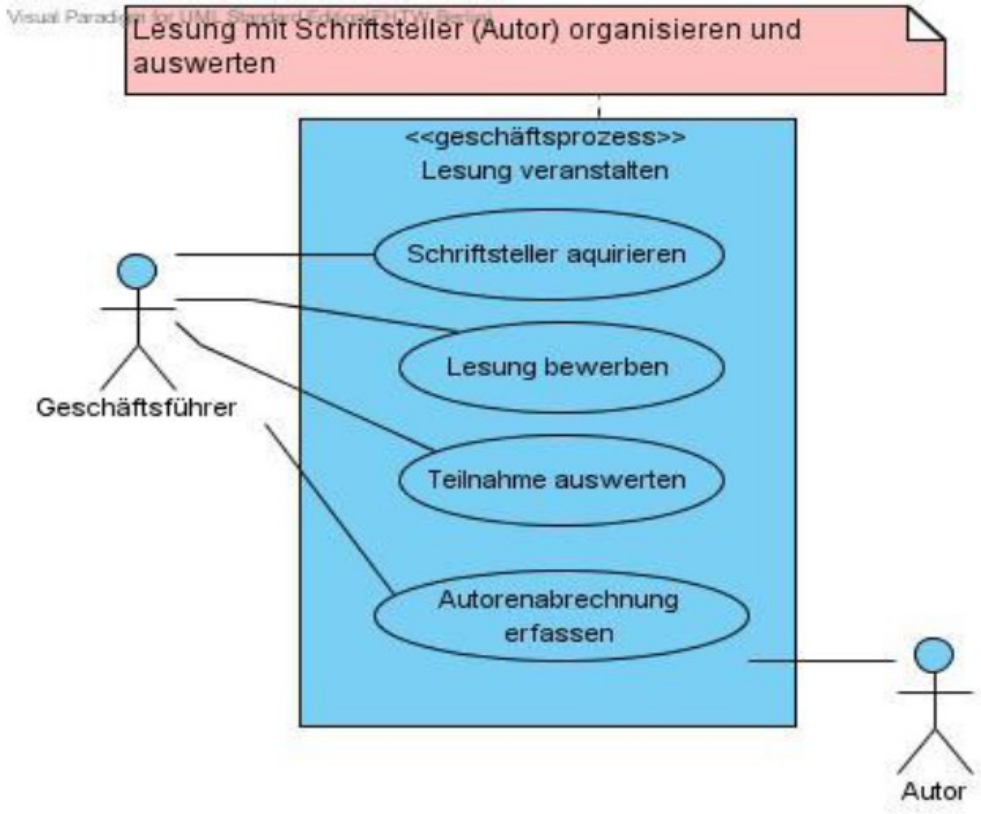
- Analysemuster
  Ein *Analysemuster* besteht aus Fachkonzeptklassen, die in bestimmten Beziehungen zueinander stehen. Es *modelliert eine* bestimmte, häufig vorkommende *Informationsstruktur*.

- GRASP-Muster (**G**eneral **R**esponsibility **A**ssignment **S**oftware **P**atterns)
  Ein GRASP-Muster stellt ein aus der Erfahrung gewonnenes Prinzip dar, nach dem den Fachkonzeptklassen diejenigen Operationen zugeordnet werden, die zur Umsetzung der SAF-Operationen notwendig sind.

- SAF-Operation
  Eine SAF-Operation ist ein vom System auszuführender *Schritt eines System-Anwendungsfalls* (SAF). Sie wird in der Regel durch mehrere Programmoperationen realisiert.

Fallbeispiel: Anwendungsfall „Autorenabrechnung erfassen“ (1/2)

- GAF „Autorenabrechnung erfassen“ als Teil des GP „Lesung veranstalten“



Fallbeispiel: Anwendungsfall „Autorenabrechnung erfassen“ (2/2)

- Es gibt 2 Realisierungen (SAF) zum GAF „Autorenabrechnung erfassen“: 1. Der Autor erfasst selbst bzw. 2. der Geschäftsführer erfasst auf der Basis vorliegender Angaben des Autors. Die Unterschiede sind geringfügig. Deshalb soll nur der SAF „ Autorenabrechnung erfassen durch Autor selbst“ für die Beispiele benutzt werden, im weiteren kurz als „ Autorenabrechnung erfassen“ bezeichnet.
- Ablaufschritte des SAF „Autorenabrechnung erfassen“

1. Aufruf der Funktion „Autorenabrechnung erfassen“ durch Akteur;
2. Das System generiert eine Liste von Lesungen, zu denen noch keine Lesungstermine abgerechnet wurden.
3. Auswahl einer Lesung;
   Der Akteur wählt eine Lesung aus einer Liste aus, der Titel des Buches, aus dem gelesen wurde, wird angezeigt.
4. Eingabe von Datum, Beginn- und Ende-Uhrzeit des Lesungstermins sowie der Reisekosten (gefahrene Kilometer bei Autoreise oder Fahrkartenpreis bei Bahnreise); Absenden (Speichern) der Daten;
5. Das System validiert die Daten. Nach erfolgreicher Validierung werden berechnet: die einzelnen Abrechnungspositionsbeträge sowie der Abrechnungs-Gesamtbetrag.
6. Abrechnung anzeigen
   Die Eingabedaten sowie die berechneten Daten werden angezeigt.

Die OOA läuft in folgenden Schritten ab:

- ⇨ **Statisches Modell:** Modellieren von Klassen und deren Attributen.
- ⇨ **Statisches Modell:** Modellieren der Beziehungen zwischen den Klassen, die Assoziationen, Aggregationen, Kompositionen, Generalisierungen/Spezialisierungen/Vererbungen.
- ⇨ **Dynamisches Modell:** Modellieren von Operationen der Klassen.
- ⇨ **Dynamisches Modell:** Modellieren von Botschaften zwischen den Klassen.

Dafür werden die UML-Diagramme wie Klassendiagramme, System-Sequenzdiagramme, Kommunikationsdiagramme, Zustandsautomaten, semiformale Beschreibungen verwendet.

Obwohl die Beziehungen zwischen den Klassen theoretisch ganz willkürlich sein können, kann man nach einer sorgfältigen Betrachtung der Klassendiagramme feststellen, dass einige bestimmte Muster häufig vorkommen. Diese Muster (**Analysemuster**) spielen ganz wichtige Rolle, weil sie die Modellierung erheblich erleichtern und standardmäßig in ein C++/C#/Java-Code umgesetzt werden können.

Methodische Hinweise bei der OOA-Modellierung:

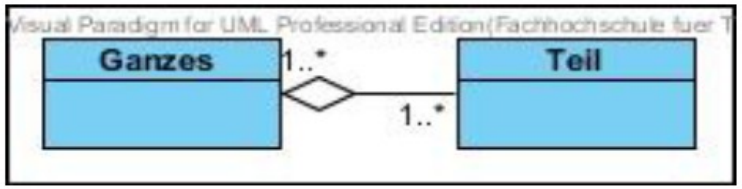
- ⇨ Die Analysemuster erlernen und erkennen.
- ⇨ Die Analysemuster versuchen, konsequent zu verwenden.

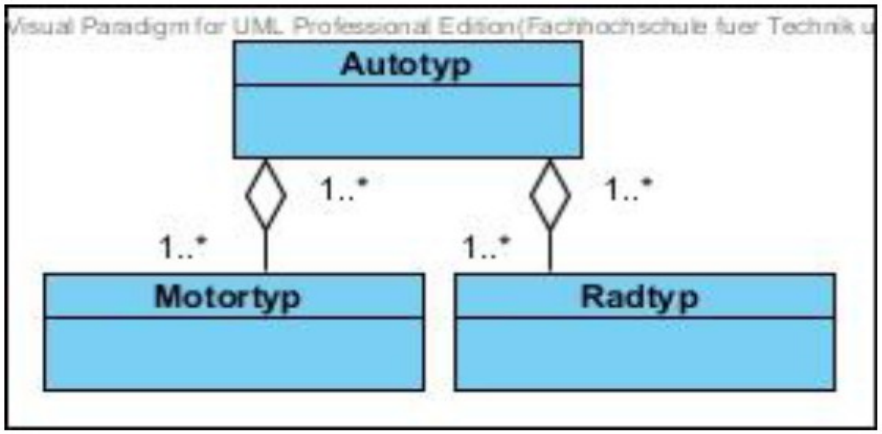
Grundlagen für Analysemuster sind die Assoziationen, Aggregationen, Kompositionen, Generalisierungen und deren Kardinalitäten (Multiplizitäten).

Schwache Aggregation

- Eine sogenannte „Schwache Aggregation “ ist wie folgt charakterisiert:
  - Die Existenz eines Teile-Objekts ergibt auch ohne ein zugehöriges Ganzes-Objekt einen Sinn.
  - Ein Teile-Objekt ist einem oder mehreren Ganzes-Objekten zugeordnet.

- Allgemeine Darstellung in UML Beispiel

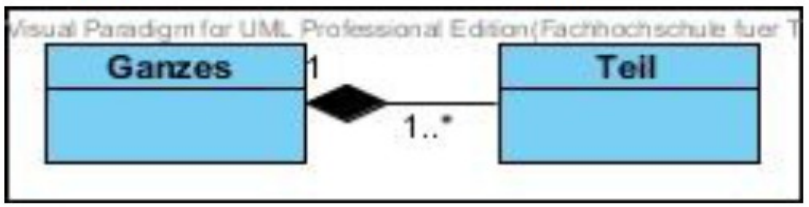


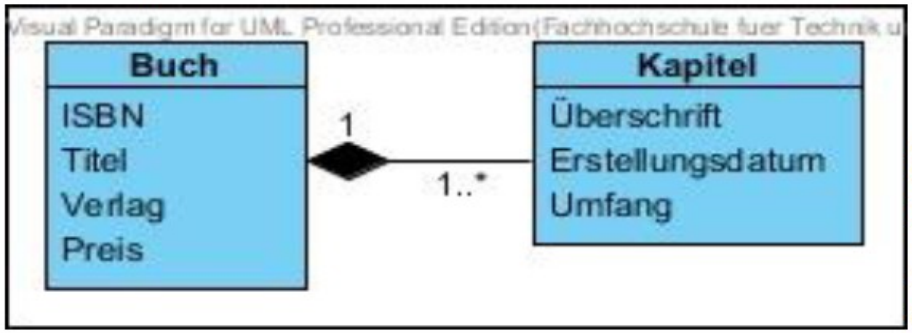


Starke Aggregation (Komposition)

- Eine sogenannte „Starke Aggregation “, auch „Komposition“ genannt, ist wie folgt charakterisiert:
  - Die Existenz eines Teile-Objekts ergibt ohne ein zugehöriges Ganzes-Objekt *keinen* Sinn.
  - Ein Teile-Objekt ist genau einem Ganzes-Objekt zugeordnet.

- Allgemeine Darstellung in UML Beispiel:



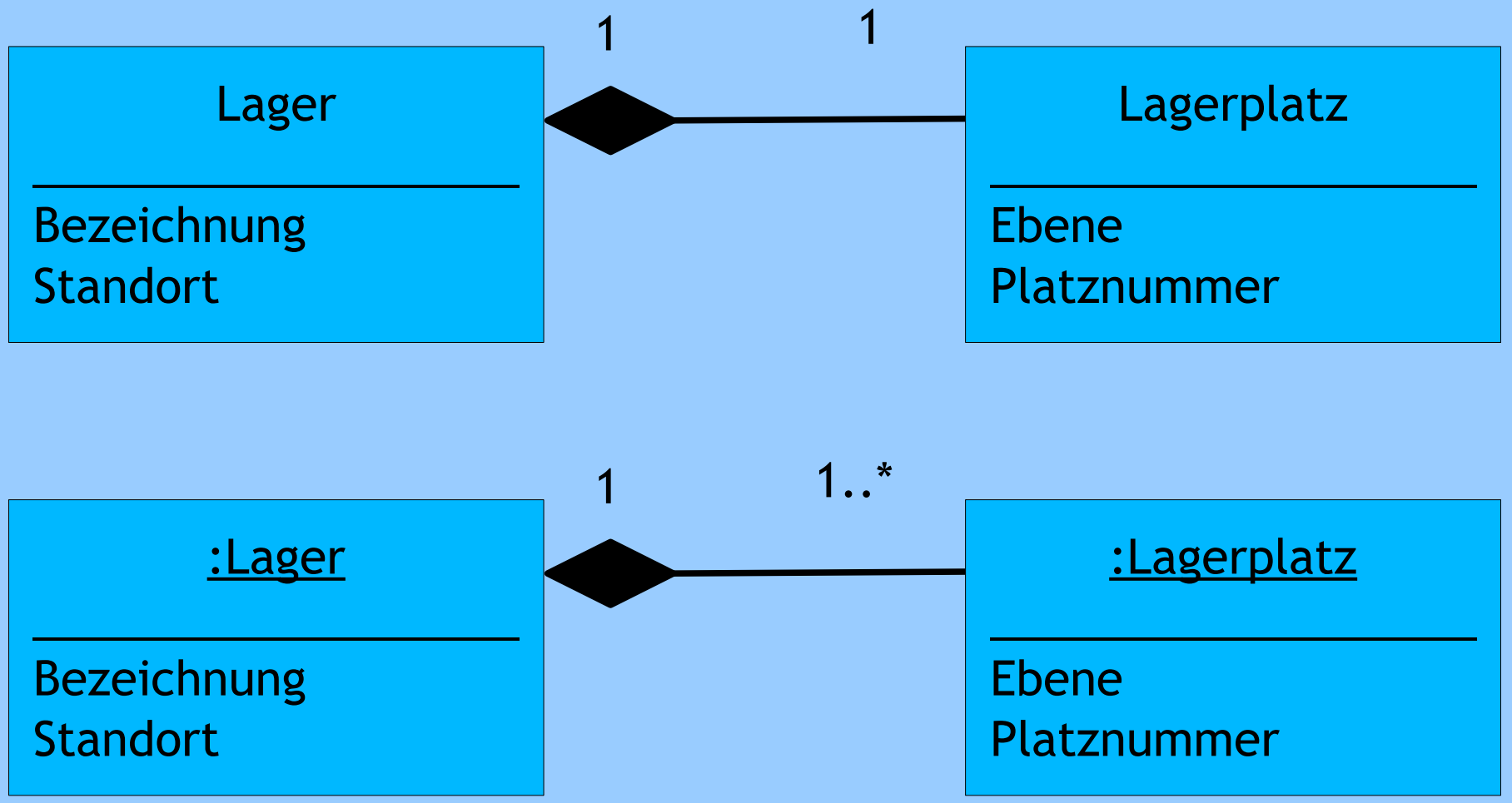


Überlegen Sie sich 2 weitere Beispiele!

Ein Lager und seine Lagerplätze lassen sich als Liste modellieren. Ein Lagerplatz kann ohne Lager nicht existieren. Ein Lagerplatz kann dem anderen Lager nicht zugeordnet werden.

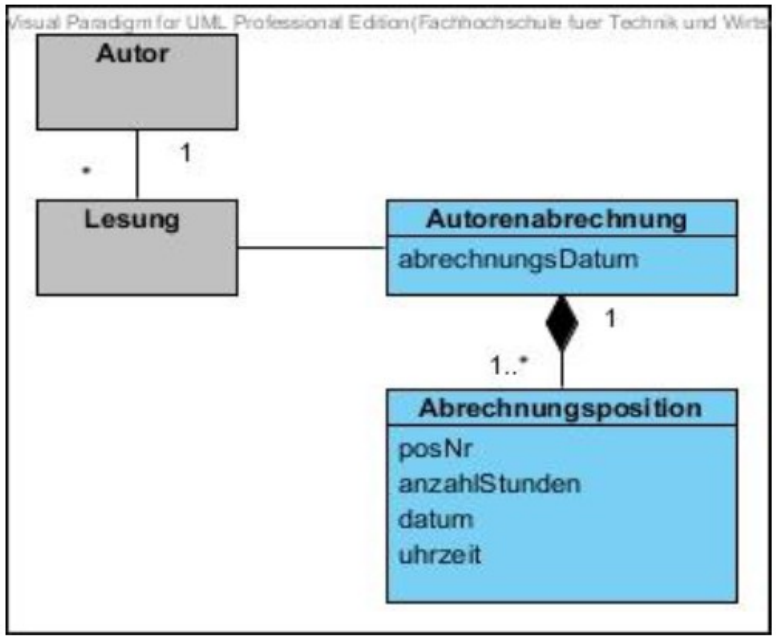
Eigenschaften:

- ⇨ Es liegt eine Komposition vor.
- ⇨ Das Ganze besteht aus gleichartigen Teilen,
  d.h. die Aggregat-Klasse besteht aus nur einer Teil-Klasse.
- ⇨ Das Ganze ist nicht gleichartig mit den Teilen.
- ⇨ Das Aggregat-Objekt besteht mindestens aus einem Teil-Objekt.
- ⇨ Die Attribute der Aggregat-Klasse gelten auch für die Teil-Klasse.


1
1
Lager
Bezeichnung
Standort
Lagerplatz
Ebene
Platznummer
1
1..*
:Lager
Bezeichnung
Standort
:Lagerplatz
Ebene
Platznummer

Analysemuster „Liste“

- Das Analysemuster „Liste“ sollte verwendet werden, wenn eine starke Aggregation (Komposition) vorliegt, für die gilt:
  - Ein Aggregat-Objekt besteht aus *gleichartigen* Teile-Objekten.
  - Aggregat-Objekt und Teile-Objekte sind *nicht gleichartig*.
- Beispiel:

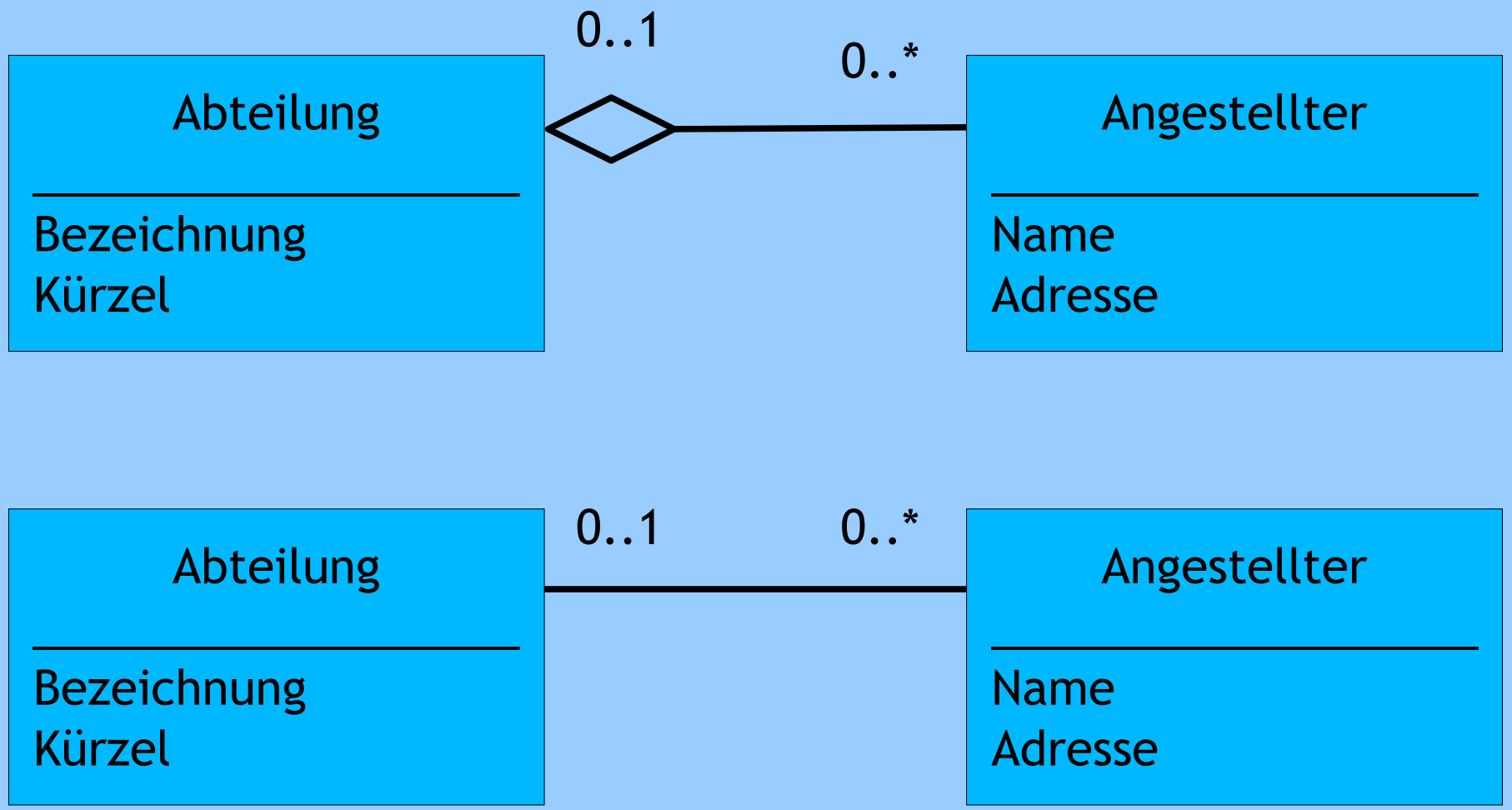


Überlegen Sie sich 2 weitere Beispiele!

Mehrere Angestellte gehören zu einer Abteilung. Die Abteilung kann kurzfristig ohne Angestellte existieren.

Eigenschaften:

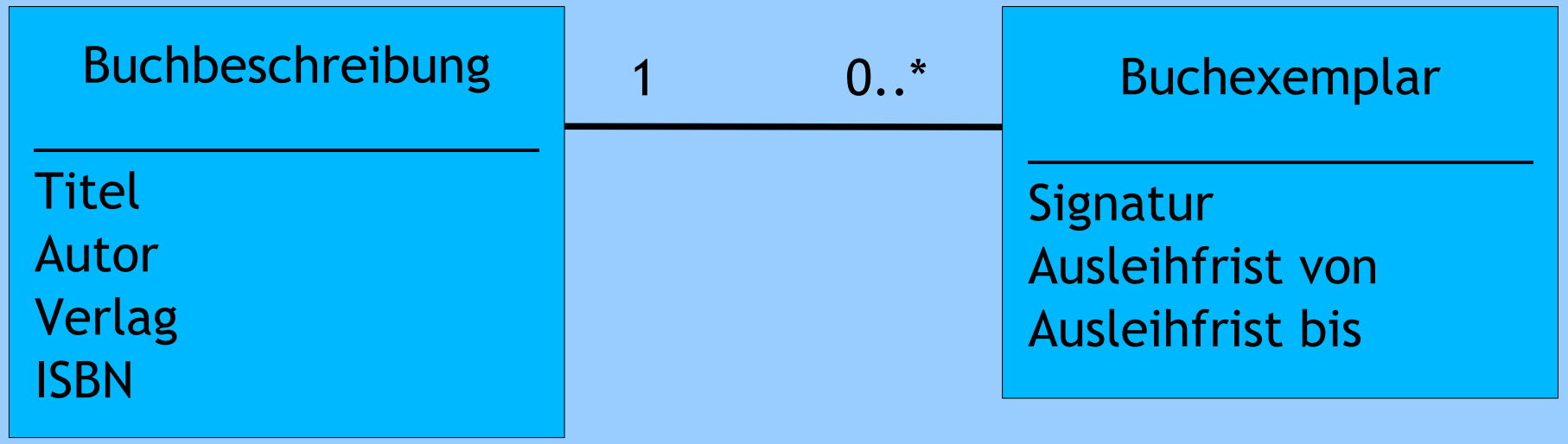
- ⇨ Es liegt eine einfache Assoziation oder eine schwache Aggregation vor.
- ⇨ Das Ganze ist nicht gleichartig mit den Teilen.
- ⇨ Das Ganze kann ohne Teil-Objekte existieren.
- ⇨ Die Attribute des Ganzen gelten auch für die Teil-Klasse.


0..1
0..*
Abteilung
Bezeichnung
Kürzel
Angestellter
Name
Adresse
0..1
0..*
Abteilung
Bezeichnung
Kürzel
Angestellter
Name
Adresse

In einer Bibliothek gibt es normalerweise mehrere Bücher mit demselben Titel von demselben Autor und mit derselben ISBN, die sich durch Signatur, Ausleihfrist, Ausleihsperre unterscheiden. Würden diese Daten durch eine einzige Klasse modelliert, dann muss man viele Informationen mehrfach speichern.

Eigenschaften:

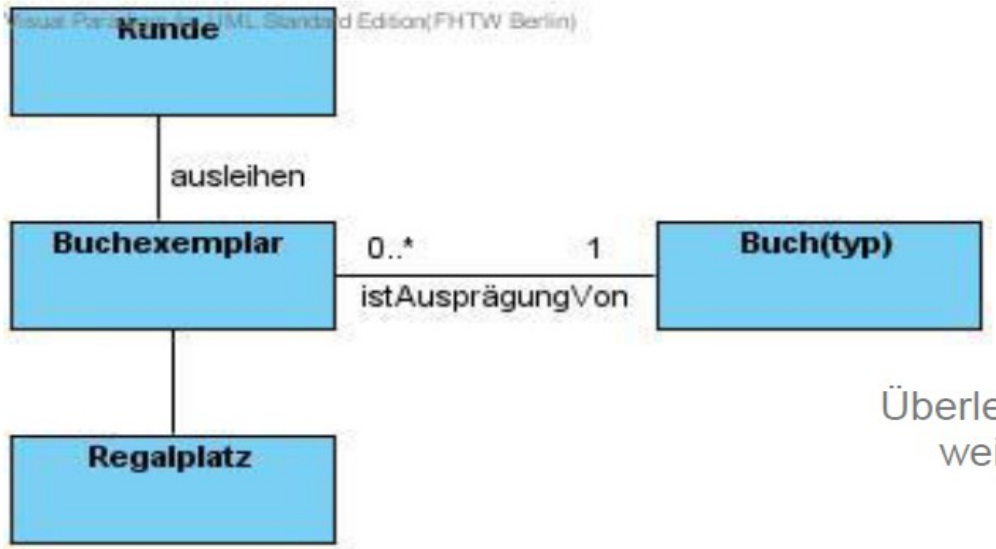
- ⇨ Es liegt eine einfache Assoziation vor.
- ⇨ Die Typ-Klasse kann ohne Exemplar-Klasse existieren, umgekehrt nicht.
- ⇨ Auf die Typ-Klasse kann man verzichten, wenn man ihre Attribute in die Exemplar-Klasse verlagert und die Redundanz in Kauf nimmt.


Buchbeschreibung
Titel
Autor
Verlag
ISBN
1
0..*
Buchexemplar
Signatur
Ausleihfrist von
Ausleihfrist bis

Analysemuster „Exemplar-Typ“

- Eine Exemplar-Typ-Beziehung zwischen zwei Entitätsklassen (Fachkonzeptklassen) liegt vor, wenn
  - redundante Attribute einer Entitätsklasse in eine separate Klasse ausgelagert werden können,
  - die Multiplizität auf der Seite der Typklasse **1** und auf der Seite der Exemplarklasse **0..*** ist; Eine Typ-Entität kann also zumindest zeitweise ohne korrespondierende Exemplar-Entität(en) existieren.

- Beispiel:

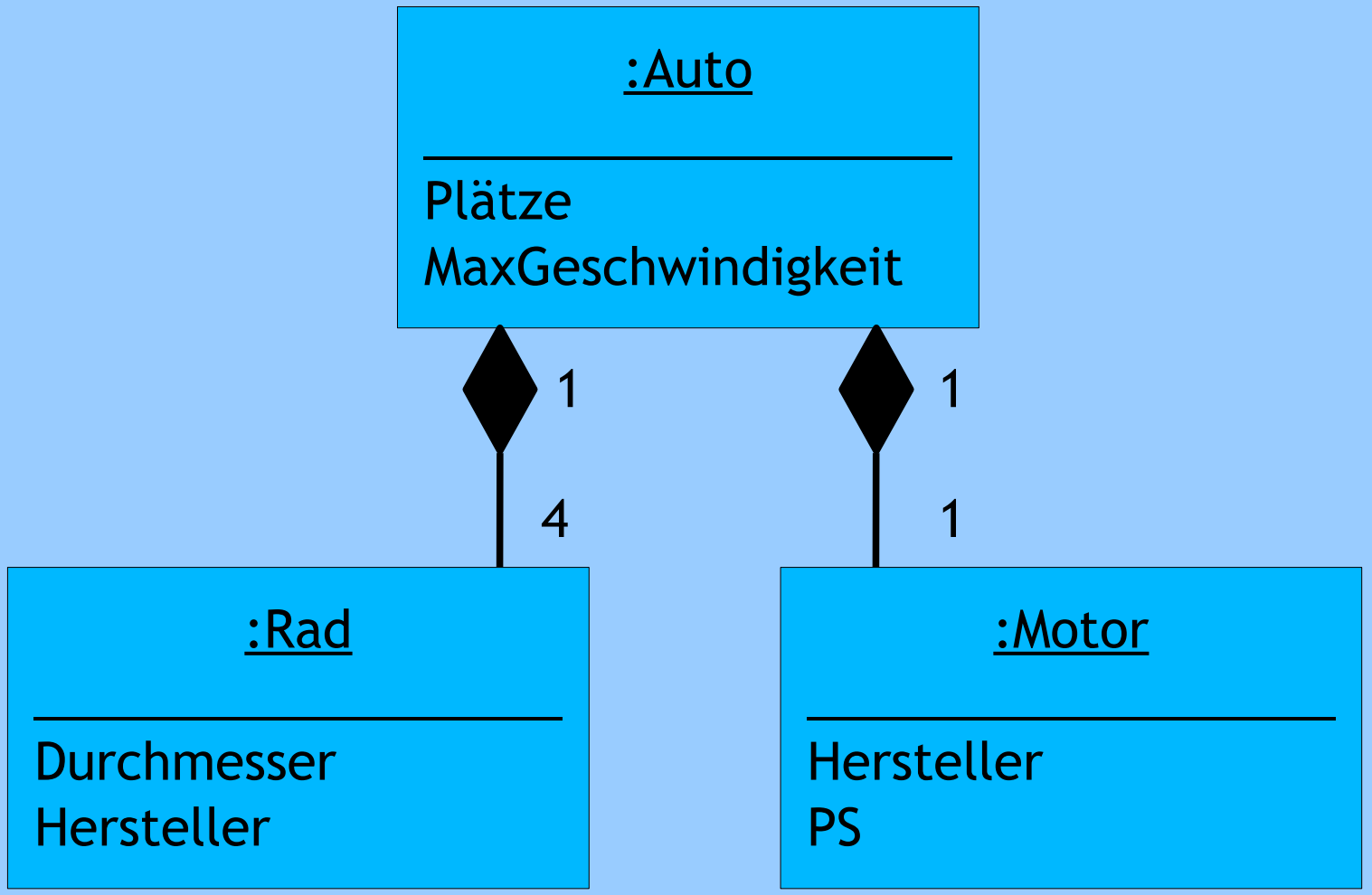


Überlegen Sie sich 2 weitere Beispiele!

Ein Auto besteht aus einem Motor und vier Rädern. Wird das Auto zerstört, dann gehen auch dessen Bestandteile verloren. Der Motor kann in ein anderes Auto eingebaut werden.

Eigenschaften:

- ⇨ Es liegt eine schwache oder eine starke Aggregation vor.
- ⇨ Das Aggregat-Objekt besteht aus unterschiedlichen Teil-Objekten.
- ⇨ Ein Teil-Objekt kann einem anderen Aggregat-Objekt zugeordnet werden.

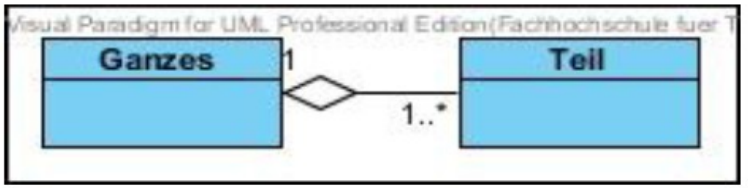

:Auto
Plätze
MaxGeschwindigkeit
1
4
1
1
:Rad
Durchmesser
Hersteller
:Motor
Hersteller
PS

Analysemuster „Baugruppe“

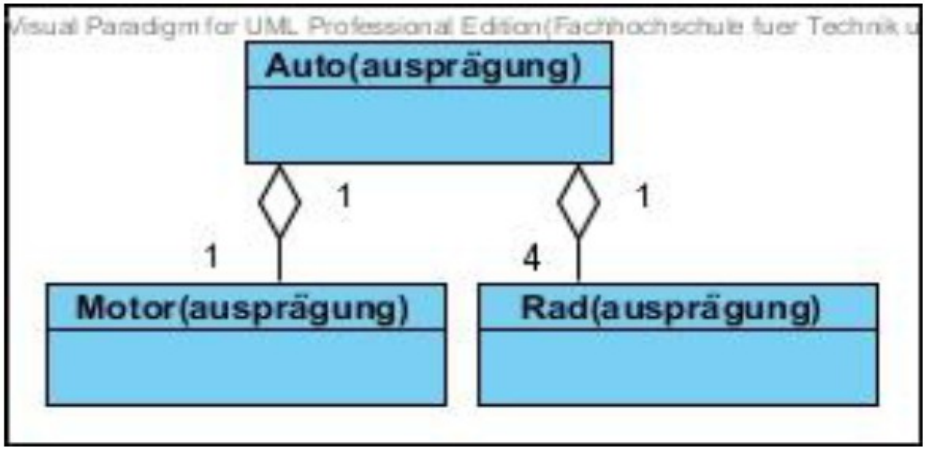
- Das Analysemuster „Baugruppe“ sollte verwendet werden, wenn eine schwache Aggregation vorliegt, für die gilt:

  Ein Teile-Objekt ist (zu einem Zeitpunkt) *genau einem* Aggregat-Objekt zugeordnet.

- Allgemeine Darstellung in UML:



- Beispiel:



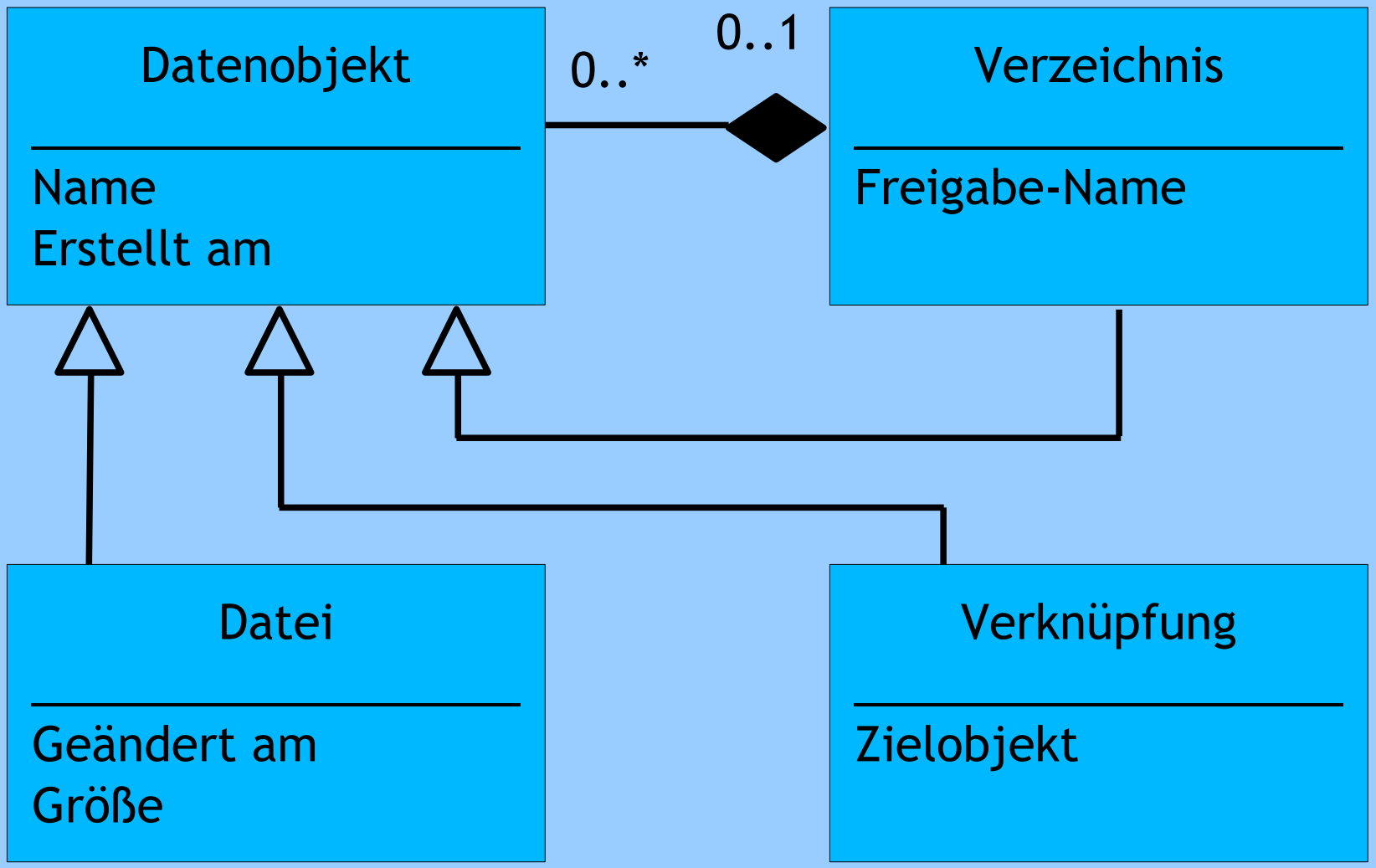
Überlegen Sie sich 2 weitere Beispiele!

Ein Container kann die weiteren Container sowie die anderen Blatt-Objekte beinhalten.

ein Verzeichnis enthält die Unterverzeichnisse, Dateien und Verknüpfungen.

Eigenschaften:

- ⇨ Es liegt eine starke Aggregation (Komposition) vor.
- ⇨ Es liegt eine Generalisation vor.
- ⇨ Das Aggregat-Objekt und seine Teil-Objekte werden zusammen und einzeln behandelt.
- ⇨ Die Teil-Objekte gehören zu unterschiedlichen Klassen.

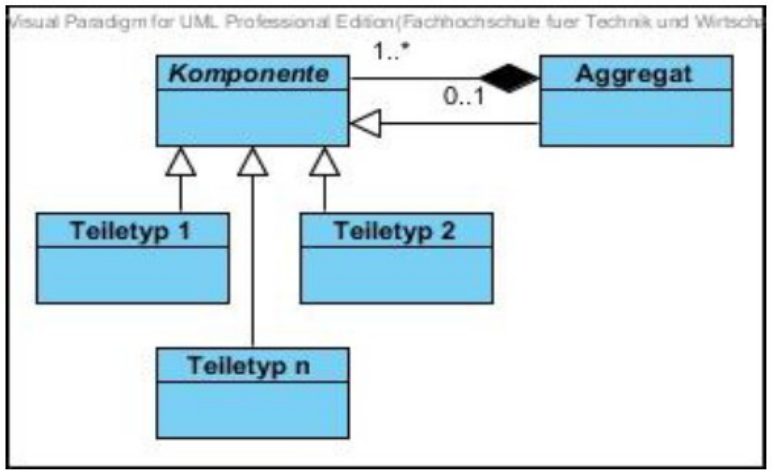

Datenobjekt
Name
Erstellt am
0..*
0..1
Verzeichnis
Freigabe-Name
Datei
Geändert am
Größe
Verknüpfung
Zielobjekt

Analysemuster „Heterogene Stückliste“

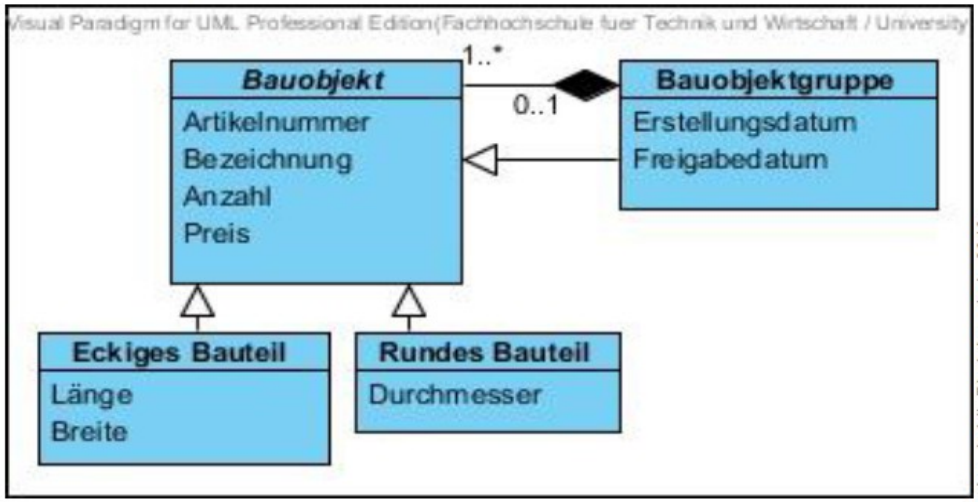
- Das Analysemuster „Heterogene Stückliste“ sollte verwendet werden, wenn eine starke Aggregation (Komposition) vorliegt, für die gilt:

  Sowohl Aggregat-Objekt(e) und Teile-Objekte als auch Teile-Objekte untereinander sind *nicht gleichartig*, d.h., sie werden zumindest partiell durch *unterschiedliche* Eigenschaften beschrieben.

- Allgemeine Darstellung



Beispiel

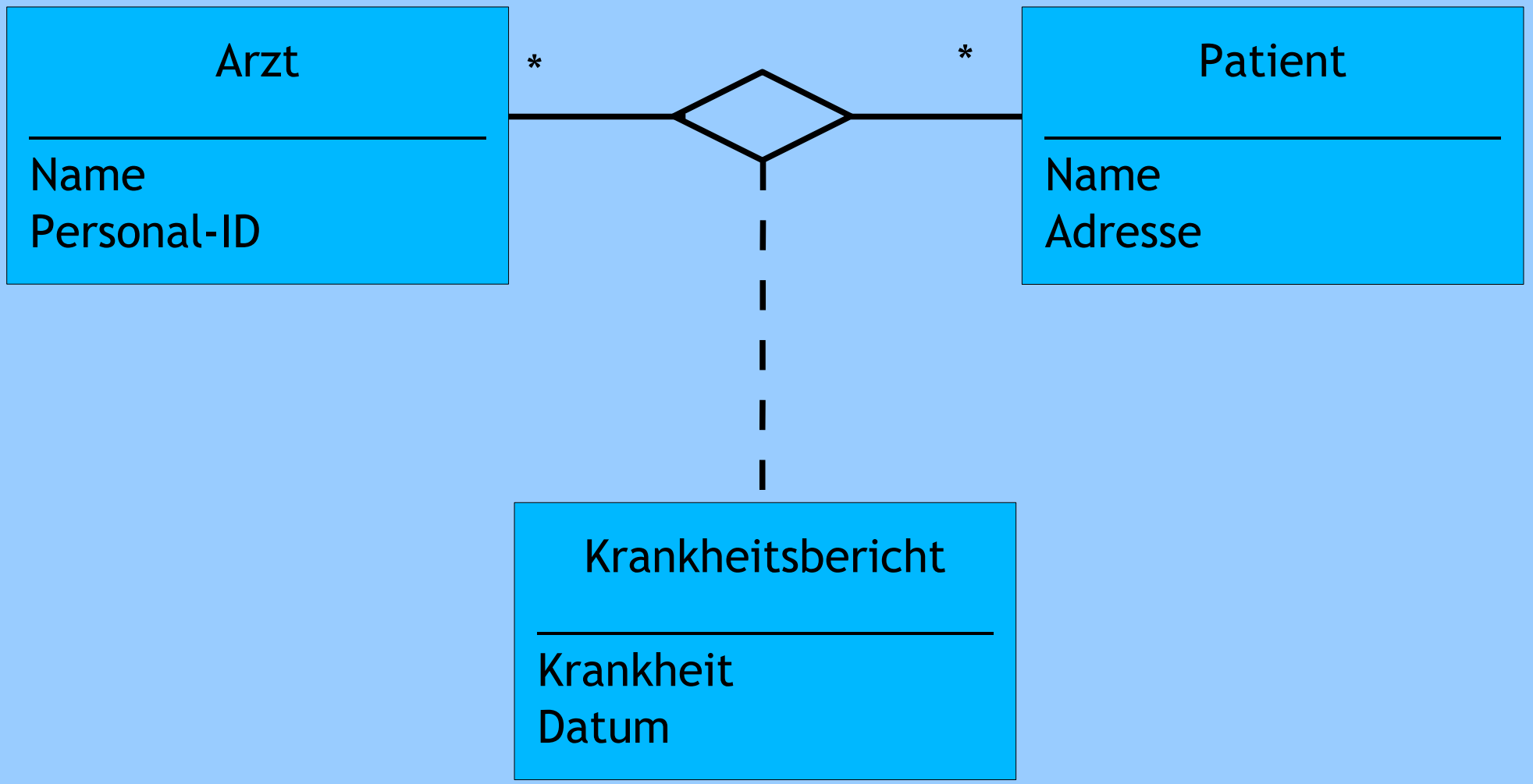


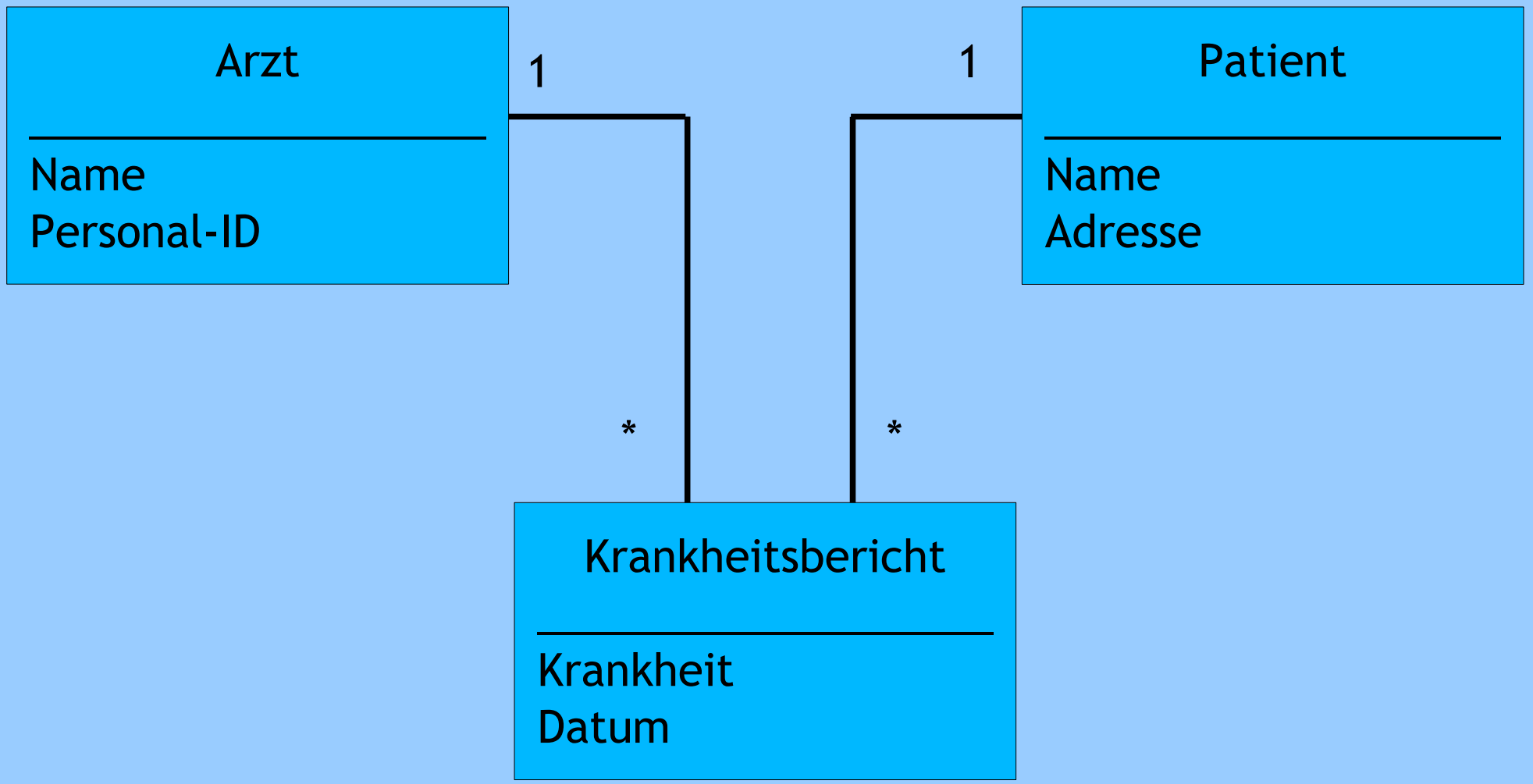
Die Koordinator-Klasse ersetzt eine n-äre Beziehung.

Ein Arzt behandelt viele Patienten, und ein Patient wird in seinem Leben von vielen Ärzten therapiert. Das ist eine binäre N:M-Beziehung, die in mehreren Krankheitsberichten dargestellt wird.

Eigenschaften:

- ⇨ Es liegt eine n-äre N:M-Assoziation vor (n >= 2).
- ⇨ Es liegt eine Generalisation vor.
- ⇨ Das Koordinator-Objekt besitzt eine Assoziation zu genau einem Objekt jeder Klasse.
- ⇨ Das Koordinator-Objekt besitzt besitzt kaum Attribute.

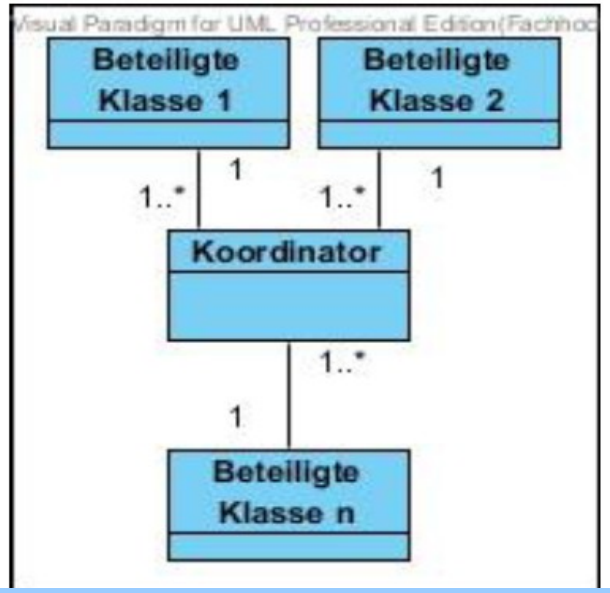

Arzt
Name
Personal-ID
*
*
Patient
Name
Adresse
Krankheitsbericht
Krankheit
Datum


Arzt
Name
Personal-ID
1
1
Patient
Name
Adresse
*
*
Krankheitsbericht
Krankheit
Datum

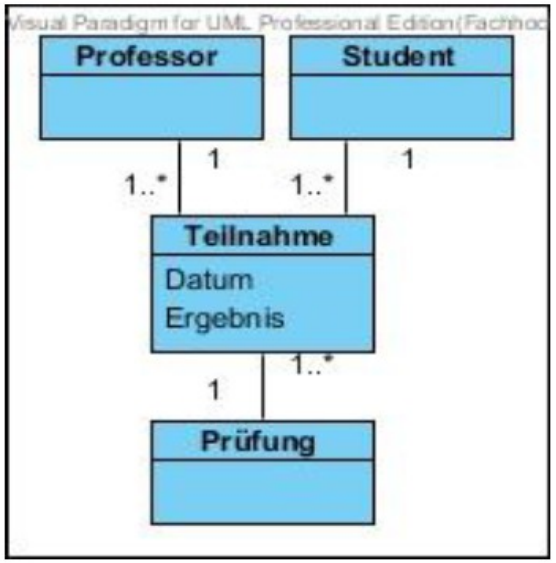
Analysemuster „Koordinator“

- Das Analysemuster „Koordinator“ sollte in zwei Fällen verwendet werden:
  - 1) wenn sich aus der Informationsanalyse die Modellierung eines *n-ären* Beziehungstyps ergibt (n > 2),
  - 2) wenn attribuierte binäre Beziehungstypen vorliegen.

- Allgemeine Darstellung



Beispiel



Überlegen Sie sich 2 weitere Beispiele!

Zwischen zwei Klassen existieren unterschiedliche Arten von Beziehungen. Versucht man diese Beziehungen durch neue Klassen zu ersetzen, dann muss man viele Informationen redundant speichern.

Eigenschaften:

- ⇨ Es liegen einfache Assoziationen vor
- ⇨ Ein Objekt kann in Bezug auf die Objekte der anderen Klasse unterschiedliche Rollen spielen.
- ⇨ Die Objekte, die unterschiedliche Rollen spielen, besitzen gleiche Attribute und möglicherweise Operationen unabhängig von der Rolle.

Analysemuster „Rollen I“ (auch: „Rollen“)

- Das Analysemuster „Rollen I“ sollte verwendet werden, wenn :
  - ein Objekt der realen Welt zu verschieden Zeitpunkten bzw. gleichzeitig unterschiedliche Zustände bzw. Rollen besitzen kann UND
  - wenn es in **keiner** Rolle durch rollenspezifische Eigenschaften charakterisiert ist.

- Beispiel: Ein Teilnehmer an einer wissenschaftlichen Veranstaltung kann in Bezug auf die Vorträge sowohl Vortragender als auch Zuhörer sein.

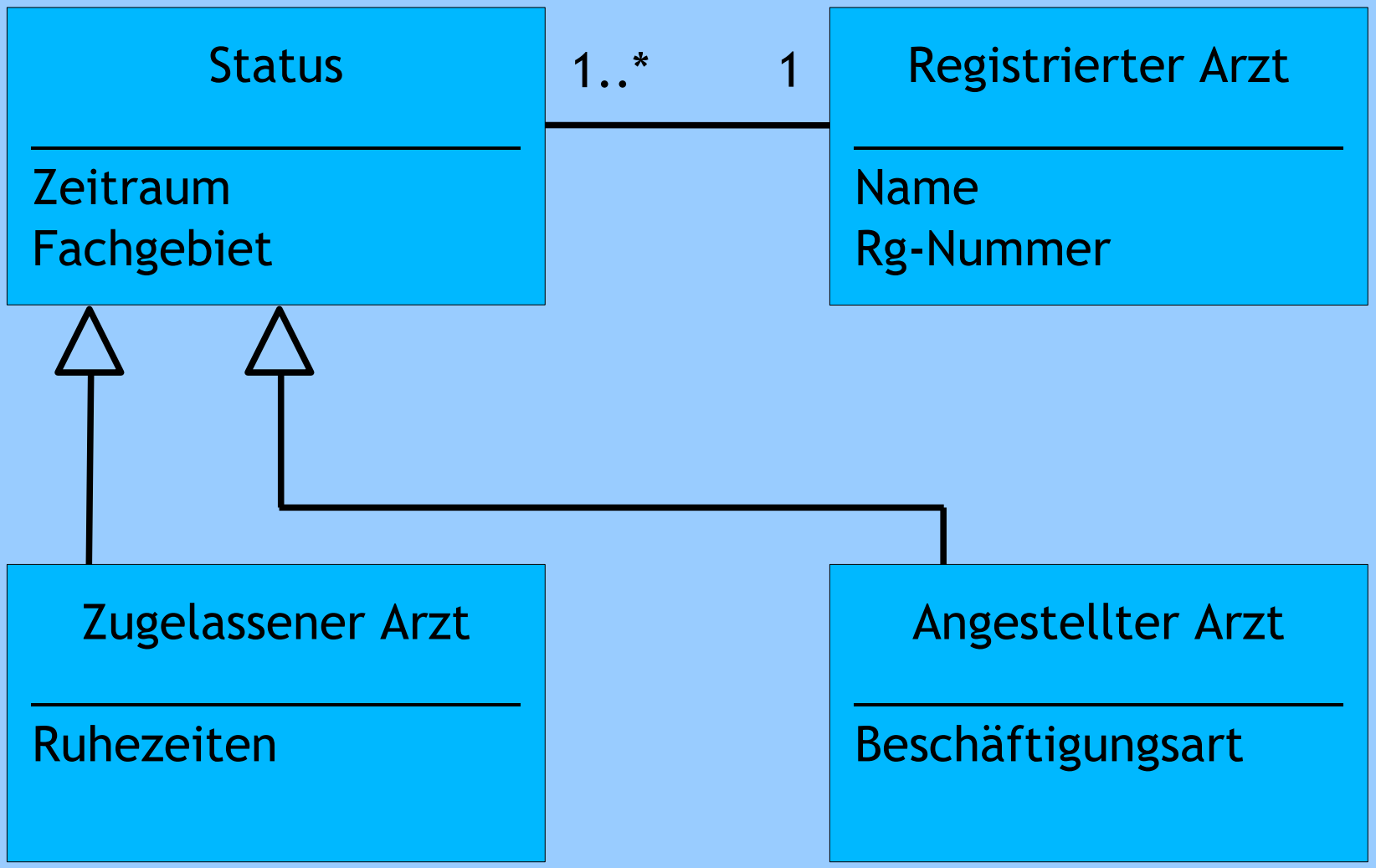
Überlegen Sie sich 2 weitere Beispiele!

Diskutieren Sie die Multiplizität „1..*“ für den Beziehungstyp (die Assoziation) „Zuhörer“ auf der Seite der Fachkonzeptklasse „Teilnehmer“!

Im Laufe der Zeit wechselt ein Objekt seine Rollen. Die Informationen über diese Rollen sind unterschiedlich und müssen über den entsprechenden Zeitraum festgehalten werden.

Eigenschaften:

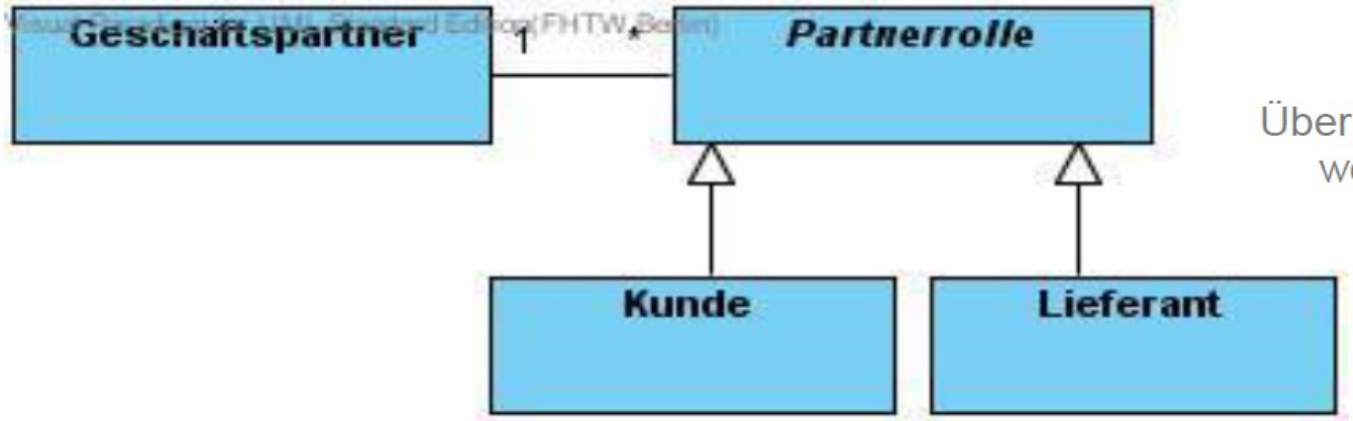
- ⇨ Es liegen einfache Assoziationen vor.
- ⇨ Zwischen der Haupt-Klasse der Rollen-Klasse existieren 1:N-Beziehungen.
- ⇨ Die unterschiedlichen Rollen werden durch die Generalisierung modelliert
- ⇨ Die Objekte, die unterschiedliche Rollen spielen, besitzen Attribute und Operationen, die abhängig von der Rolle sind.


Status
Zeitraum
Fachgebiet
1..*
1
Registrierter Arzt
Name
Rg-Nummer
Zugelassener Arzt
Ruhezeiten
Angestellter Arzt
Beschäftigungsart

Analysemuster „Rollen II“ (auch: „Zustand“ oder „Wechselnde Rollen“)

- Das Analysemuster „Rollen II“ sollte verwendet werden, wenn :
  - ein Objekt der realen Welt zu verschieden Zeitpunkten bzw. gleichzeitig unterschiedliche Zustände bzw. Rollen besitzen kann UND
  - wenn es in **mindestens einer** Rolle durch rollenspezifische Eigenschaften charakterisiert ist.

- Beispiel: Ein Geschäftspartner kann sowohl Kunde als auch Lieferant sein. Kunde und Lieferant werden durch zumindest teilweise unterschiedliche Eigenschaften charakterisiert.

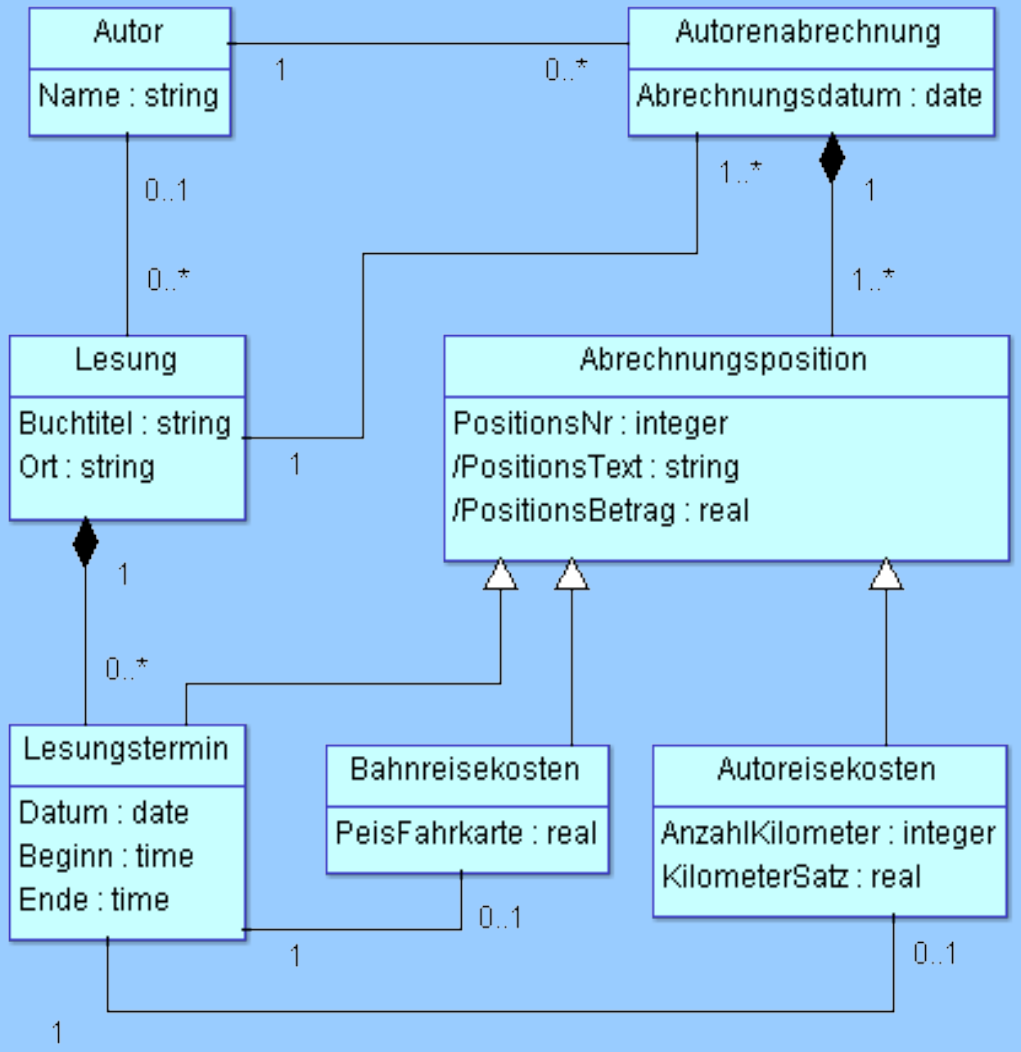


Überlegen Sie sich 2 weitere Beispiele!

Durch die Anwendung der Analysemuster bekommt man die UML-Klassendiagramme, die Fachkonzeptklassen enthalten (Fachkonzeptklassendiagramme).

Sie beschreiben die statischen Aspekte der OOA.

OOA, Fallbeispiel - Fachkonzeptklassen



Fachkonzeptklassen zum SAF "Autorenabrechnung erfassen"

Nachdem das statische Modell vollendet ist, fängt man mit dem dynamischen Modell an. Ziel des dynamischen Modells ist es, die Operationen den Klassen zuzuordnen.

Hier arbeitet man mit schon lange bekannten Regeln. Diese Regeln heißen auch **Entwurfsmuster**.

Die Entwurfsmuster beschreiben allgemein, welche Klassen und Objekte wofür zuständig sein sollten.

GRASP-Muster

- Kopplung und Bindung sind nicht unabhängig voneinander.
- Schlechte, d. h. *hohe Kopplung* hat oft schlechte, d. h. *geringe Bindung* zur Folge.
- Craig Larman hat grundsätzliche Prinzipien für den Klassenentwurf in Verbindung mit der Zuordnung von Verantwortung herausgearbeitet, die er in Mustern umsetzte: GRASP.
- GRASP: **G**eneral **R**esponsibility **A**ssignment **S**oftware **P**atterns

Larman, C: Applying UML and Patterns. An Introduction to Object-Oriented Analysis and Design. 2. Aufl., Upper Saddle River 2002

Kopplung

- Die Kopplung drückt aus, in welchem Umfang eine Klasse „Wissen“ über eine andere Klasse braucht, auf andere Klassen angewiesen bzw. mit wie vielen anderen Klassen verbunden ist.

- Eine hohe Kopplung einer Klasse sollte vermieden werden, weil
  - bei Änderungen mehrere betroffene Klassen geändert werden müssen;
  - bei isolierter Betrachtung solche Klassen schwer verständlich sind;
  - die Wiederverwendung einer Klasse eingeschränkt ist, da das Vorhandensein der verbundenen Klassen notwendig ist.

Bindung (funktionale Bindung)

- Die Bindung drückt aus, wie eng zusammengehörig Operationen in einer Klasse sind.
- Eine Klasse mit geringer Bindung ist für viele nicht zusammengehörige Aufgaben verantwortlich.
- Eine geringe Bindung bringt folgende Nachteile mit sich:
  - Die Klasse ist schwer verständlich.
  - Die Klasse ist nur begrenzt wiederverwendbar.
  - Die Klasse lässt sich schwer warten bzw. pflegen.
  - Die Klasse ist unter Umständen von ständigen Änderungen betroffen.

Bei dem objektorientierten Ansatz muss ein Objekt alle Methoden (Operationen) besitzen, die irgendwelche Aktionen mit ihm (genau gesagt mit seinen Attributen) durchführen.

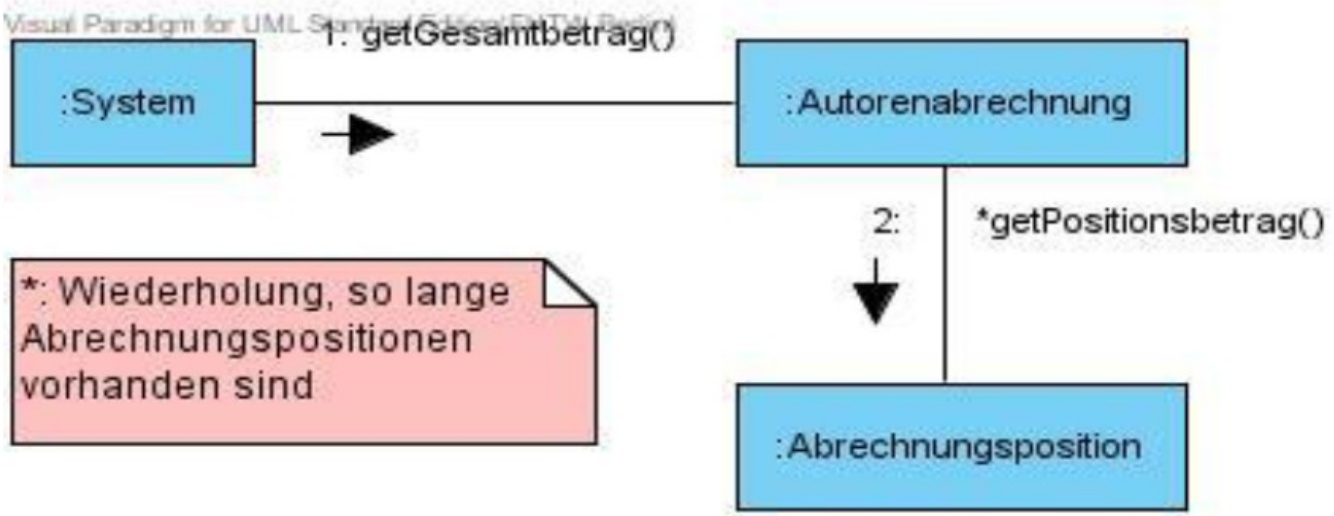
Deswegen stellt sich die Frage, welche Objekte oder welche Klassen die Verantwortlichkeit über die Operationen übernehmen sollen.

Das Experten-Muster besagt, dass eine Operation der Klasse zugeordnet werden muss, die dafür notwendige Attribute besitzt.

Experten-Muster (1/3)

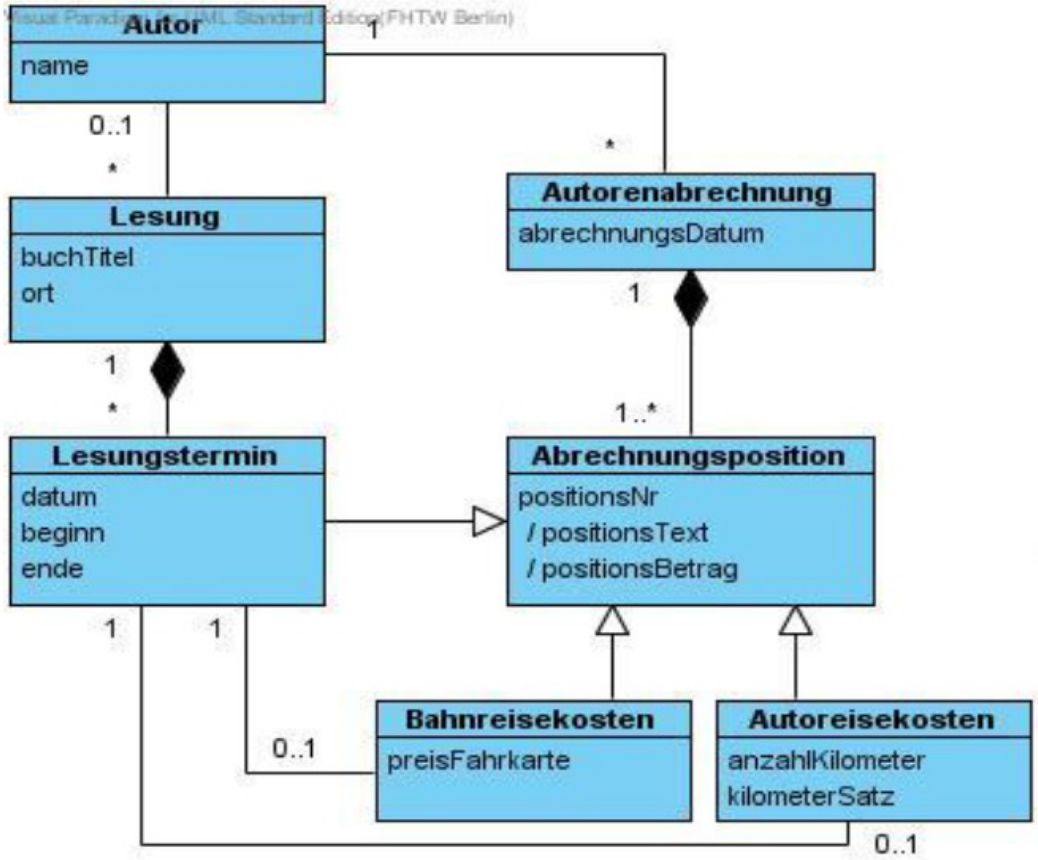
- Eine Operation ist der Klasse zuzuordnen, die über die Attribute verfügt, welche zum Ausführen der Operation notwendig sind.

- Beispiel: Für die Aktion 6 „Abrechnung anzeigen“ des SAF „Autorenabrechnung erfassen“ ist es notwendig die Beträge der einzelnen Abrechnungspositionen sowie den Gesamtbetrag zu berechnen. Mit Hilfe eines **Kommunikationsdiagramms** lassen sich die Zusammenhänge übersichtlich darstellen.



Experten-Muster (2/3)

Fach*konzept*klassen für den SAF „Autorenabrechnung erfassen“

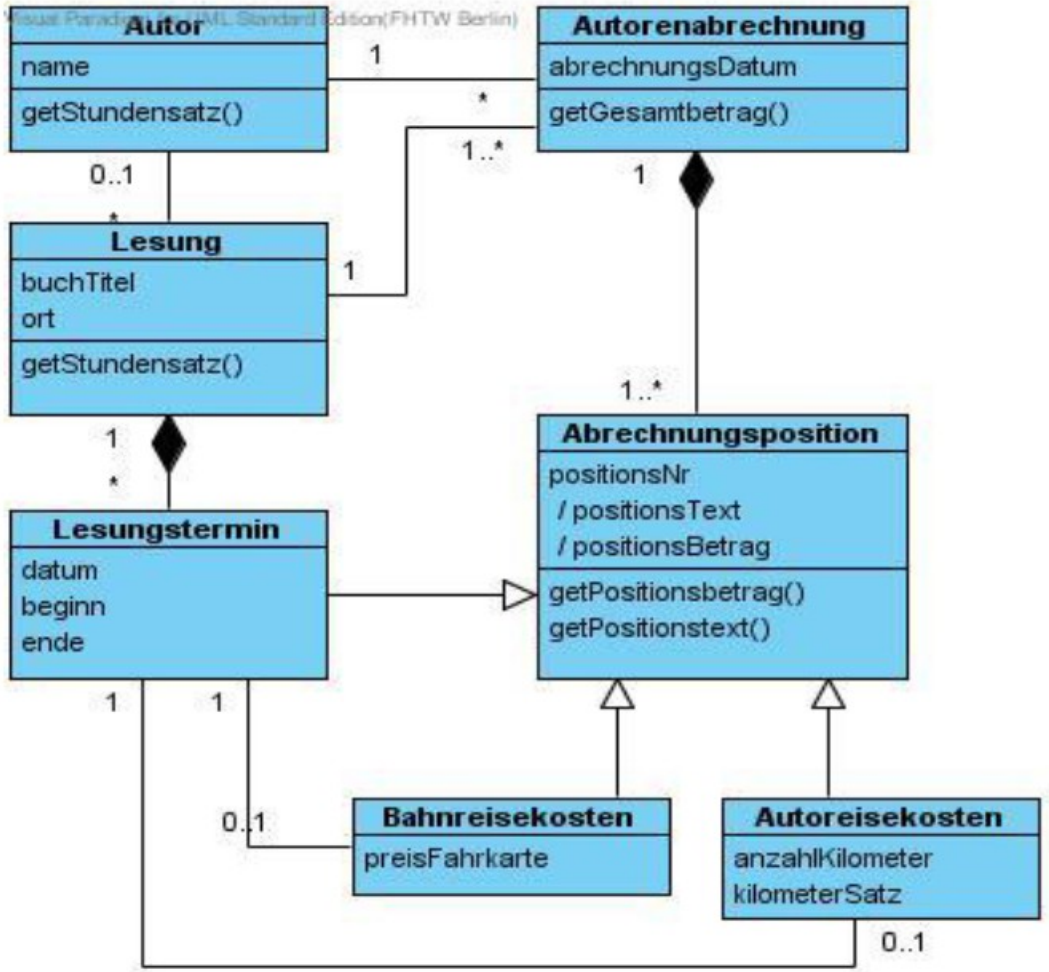


In diesem Modell fehlt eine Assoziation. Welche?

Welches Analysemuster könnte bei diesem Modell angewendet werden?
Welche Verbesserungen ergäben sich damit?

Experten-Muster (3/3)

Fachklassen für den SAF „Autorenabrechnung erfassen“



Zuordnung der Operationen zu den „Informationsexperten“

Welche Nachteile bringt diese Modellierung der Operationen *getPositionsbetrag()* und *getPositionstext()* mit sich?

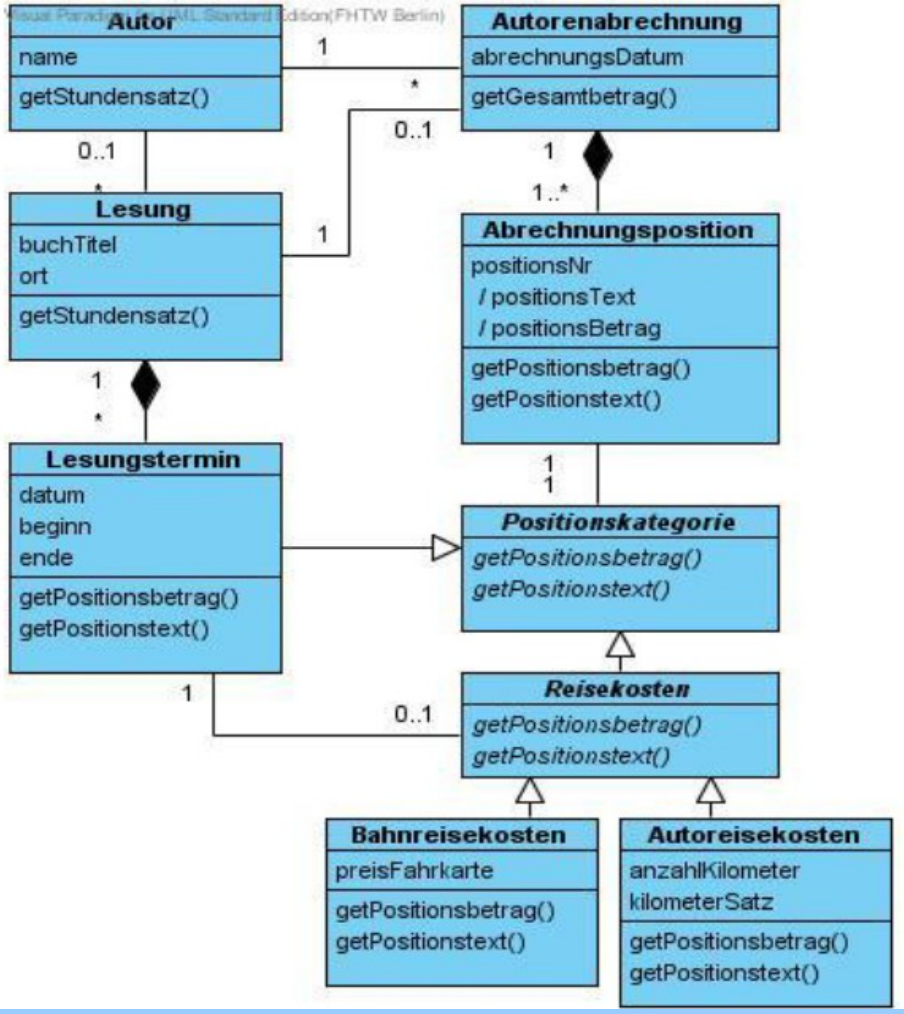
Das Polymorphismus-Muster verlangt, dass eine Operation, die zu mehreren Klassen zugeordnet sein kann, ihr Verhalten abhängig von der Klasse implementiert.

Ziel – Vermeidung von Fallunterscheidungen.

In C++-artigen Programmiersprachen gibt es dafür den Mechanismus der virtuellen Methoden.

Polymorphismus-Muster

Fachklassen für den SAF „Autorenabrechnung erfassen“



Zuordnung der Operationen unter Berücksichtigung der Polymorphie

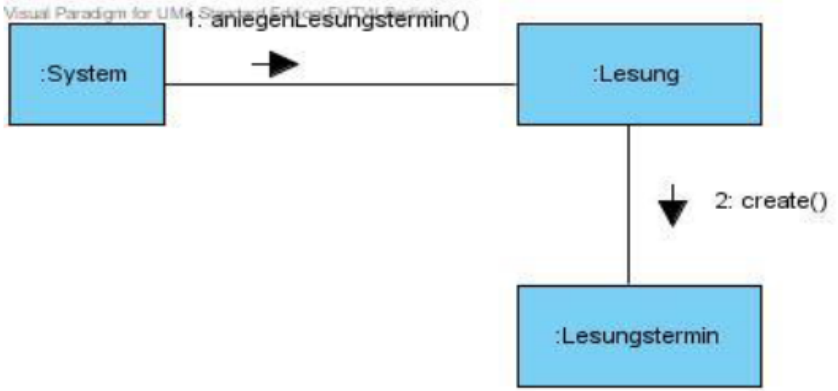
Wie wird man die abstrakte Klasse „*Positionskategorie*“ in Java implementieren?

Das Erzeuger-Muster verschreibt, welche Objekte können/dürfen die Objekte einer anderen Klasse erzeugen.

Erzeuger-Muster

- Einer Klasse A wird die Verantwortlichkeit für das Erzeugen eines Objektes der Klasse B zugewiesen, wenn eine oder mehrere der folgenden Aussagen zutreffen:
  - Ein A-Objekt aggregiert B-Objekte.
  - Ein A-Objekt enthält B-Objekte.
  - Ein A-Objekt erfasst B-Objekte.
  - Ein A-Objekt verfügt über die Initialisierungswerte, die zum Anlegen eines B-Objektes notwendig sind.

- Beispiel: Lesungstermin-Objekt erzeugen (Kommunikationsdiagramm)



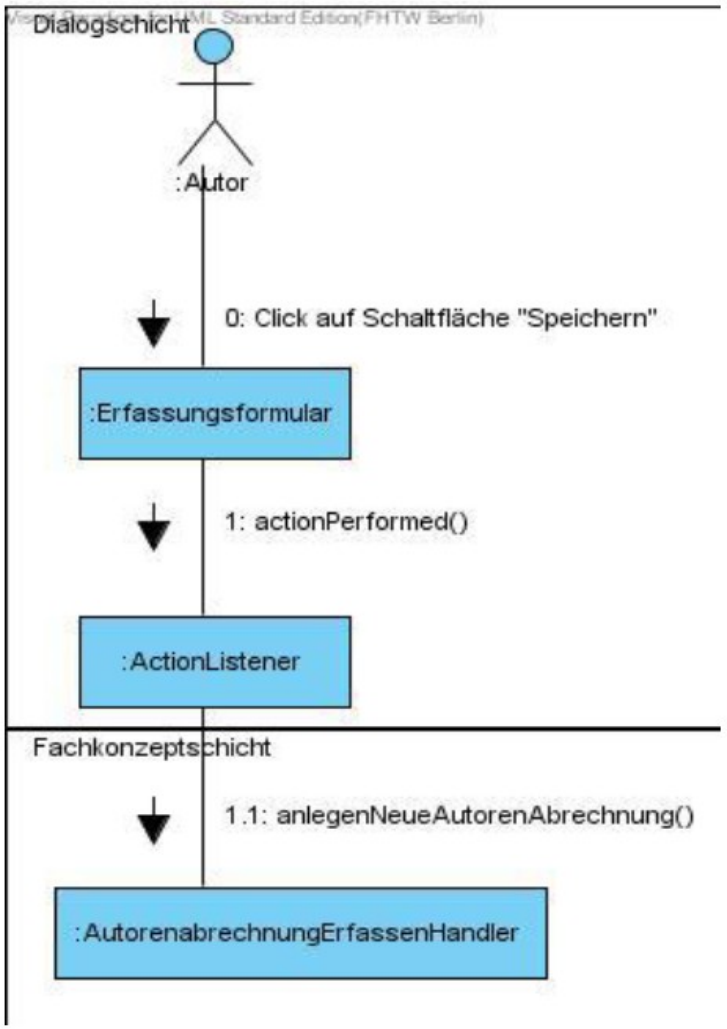
Erstellen Sie das Kommunikationsdiagramm für ein weiteres Erzeugermuster aus dem Fachklassenmodell „Autorenabrechnung erfassen“!

Laut dem Controller-Muster muss eine Controller-Klasse die Informationen zusammenfassen, welche Operationen die für die Klassen bestimmten Systemereignisse verarbeiten.

Also, die Controller-Klasse beinhaltet die Logik, die die Klassen unter einander verbindet.

Controller-Muster

- Die Fachkonzeptschicht (domain layer) sollte von der Dialogschicht (user interface layer) durch ein oder mehrere Controller-Klassen getrennt werden.
- Wenn man die Anwendungsfälle als Basis für den Entwurf nimmt, dann ist es günstig, für jeden Anwendungsfall einen separaten Controller zu entwerfen.
- Beispiel: SAF „Autorenabrechnung erfassen“

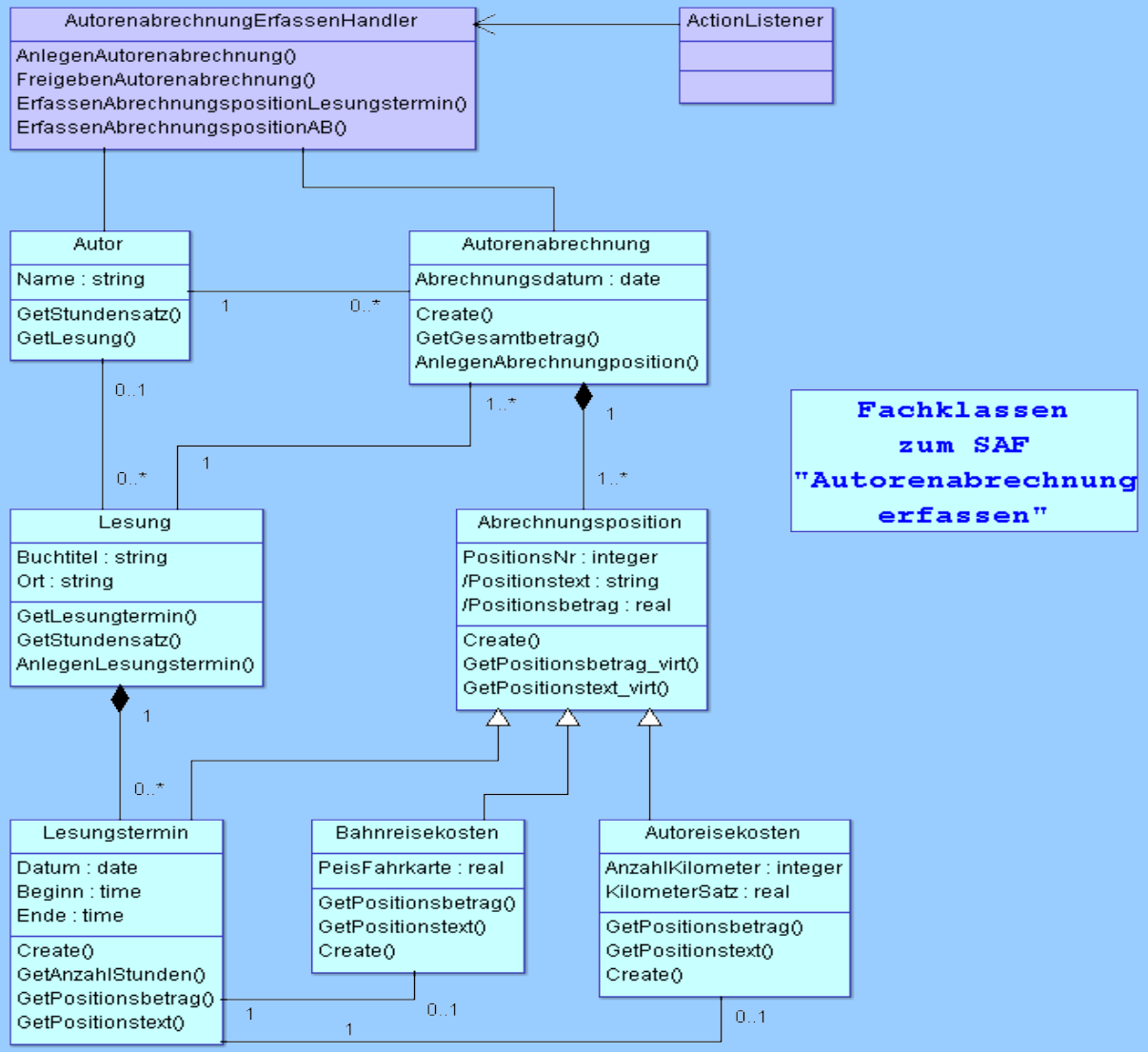


Laut dem Erfindung-Muster muss eine Operation, die zu keiner Klasse zugeordnet werden kann, in einer speziellen Klasse erfasst werden.

Solche Operationen kommen nicht aus dem Anwendungsbereich, sondern stellen ein allgemeines Wissen dar.

Dieses Muster sollte nur selten verwendet werden.

OOA, Fallbeispiel - Fachklassen



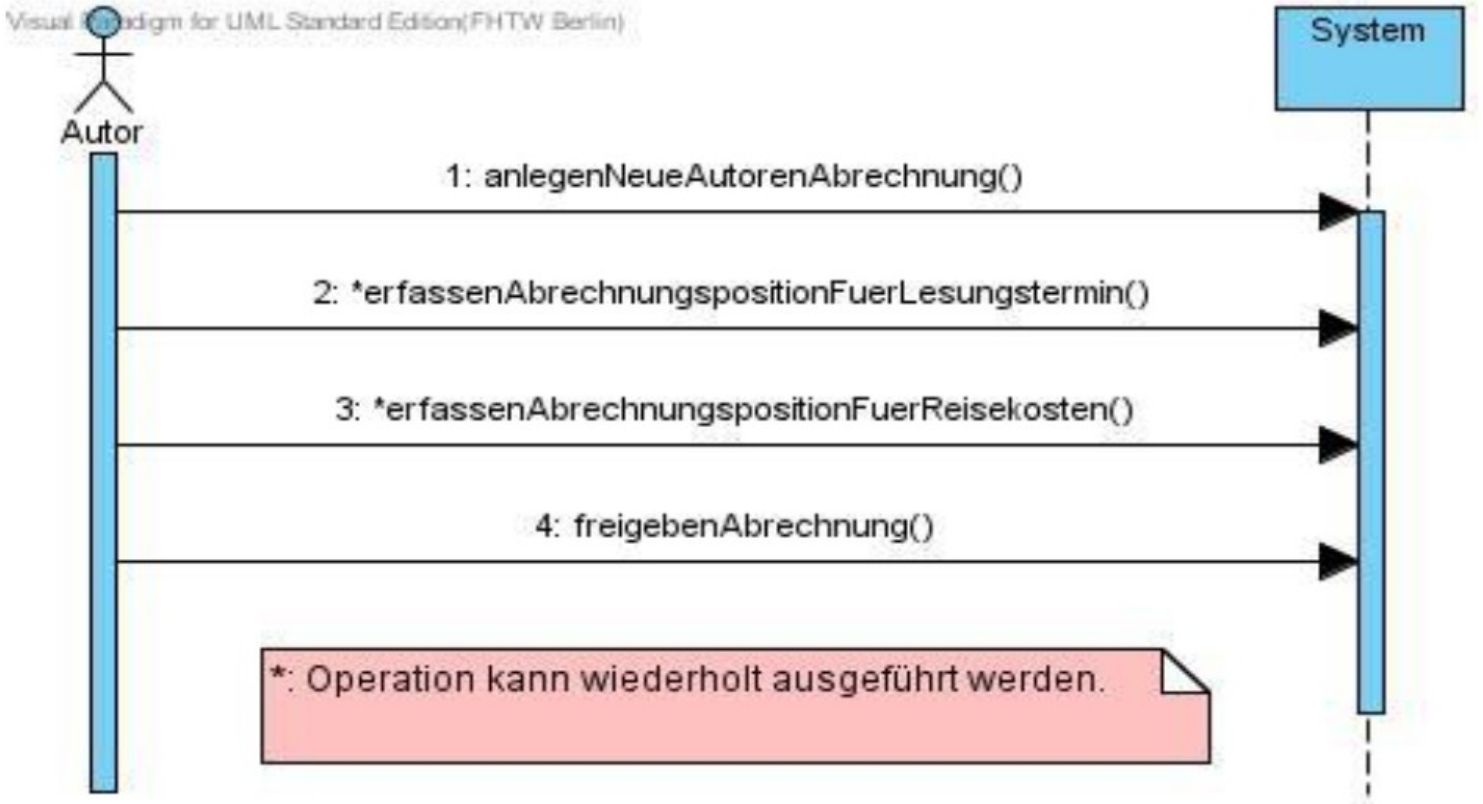
Sind die SAF-Operationen den Klassen zugeordnet, dann bleibt noch der letzte Schritt, um das dynamische Modell der OOA zu vollenden. Das ist die Beschreibung der SAF-Operationen.

Die SAF-Operationen können auf unterschiedliche Weise dokumentiert werden:

- ⇨ Die Zusammenfassung aller Operationen eines SAF kann man in einem **System-Sequenzdiagramm** darstellen.
- ⇨ Für die Beschreibung der einzelnen Operationen verwendet man **Schablonen** und **Kommunikationsdiagramme**.

Beschreibung von SAF-Operationen

- Beispiel: Operationen des SAF „Autorenabrechnung erfassen“
  Zeitlicher Ablauf dargestellt mit einem System-Sequenzdiagramm



Fehlt in diesem Diagramm eventuell eine SAF-Operation?

Spezifikation von SAF-Operationen

- Beispiel: Operation „anlegenNeueAutorenabrechnung“ des SAF „Autorenabrechnung erfassen“

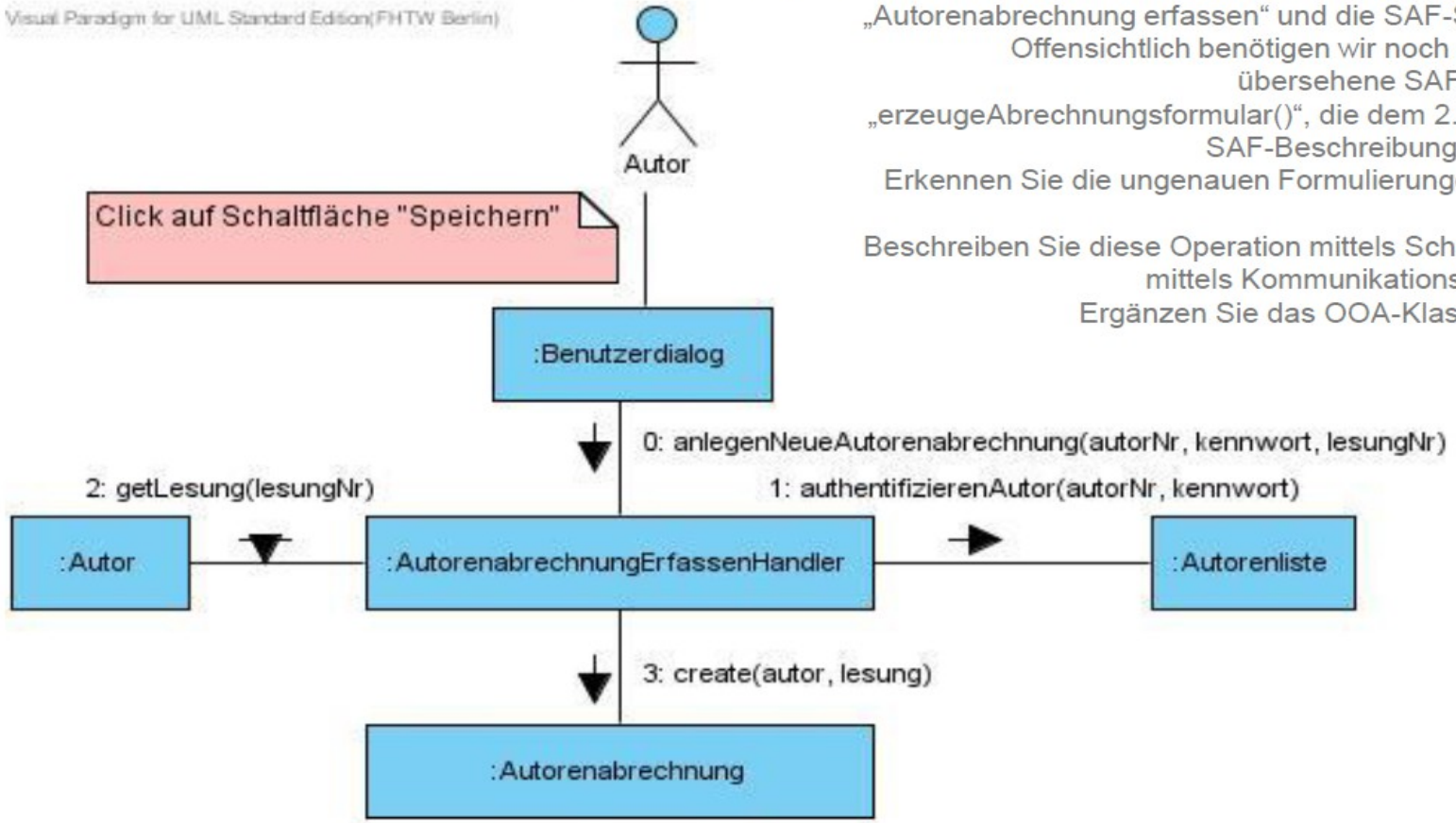
  Beschreibung mittels Schablone

Name der Operation	anlegenNeueAutorenabrechnung(autorenNummer, kennwort, lesungsnummer);
Verwendungsnachweis	SAF „Autorenabrechnung erfassen;
Vorbedingungen	Autoren und Lesungen sowie die Verbindung zwischen beiden sind im System vorhanden;
Nachbedingungen	1. Ein neues Objekt vom Typ „Autorenabrechnung“ wurde erzeugt. 2. Das neue Objekt vom Typ „Autorenabrechnung“ wurde mit dem zugehörigen Objekt vom Typ „Autor“ verknüpft. 3. Das neue Objekt vom Typ „Autorenabrechnung“ wurde mit dem zugehörigen Objekt vom Typ „Lesung“ verknüpft.

Realisierung von SAF-Operationen (1/6)

- Beispiel 1: Operation „anlegenNeueAutorenabrechnung“ des SAF „Autorenabrechnung erfassen“
  Beschreibung mittels Kommunikationsdiagramm

Wo kommt der Übergabeparameter „lesungNr“ her?
Schauen Sie sich das Dialogfenster „Autorenabrechnung erfassen“ und die SAF-Schritte an!
Offensichtlich benötigen wir noch eine bisher übersehene SAF-Operation „erzeugeAbrechnungsformular()“, die dem 2. Schritt der SAF-Beschreibung entspricht.
Erkennen Sie die ungenauen Formulierungen im SAF-Ablauf?
Beschreiben Sie diese Operation mittels Schablone und mittels Kommunikationsdiagramm!
Ergänzen Sie das OOA-Klassenmodell!



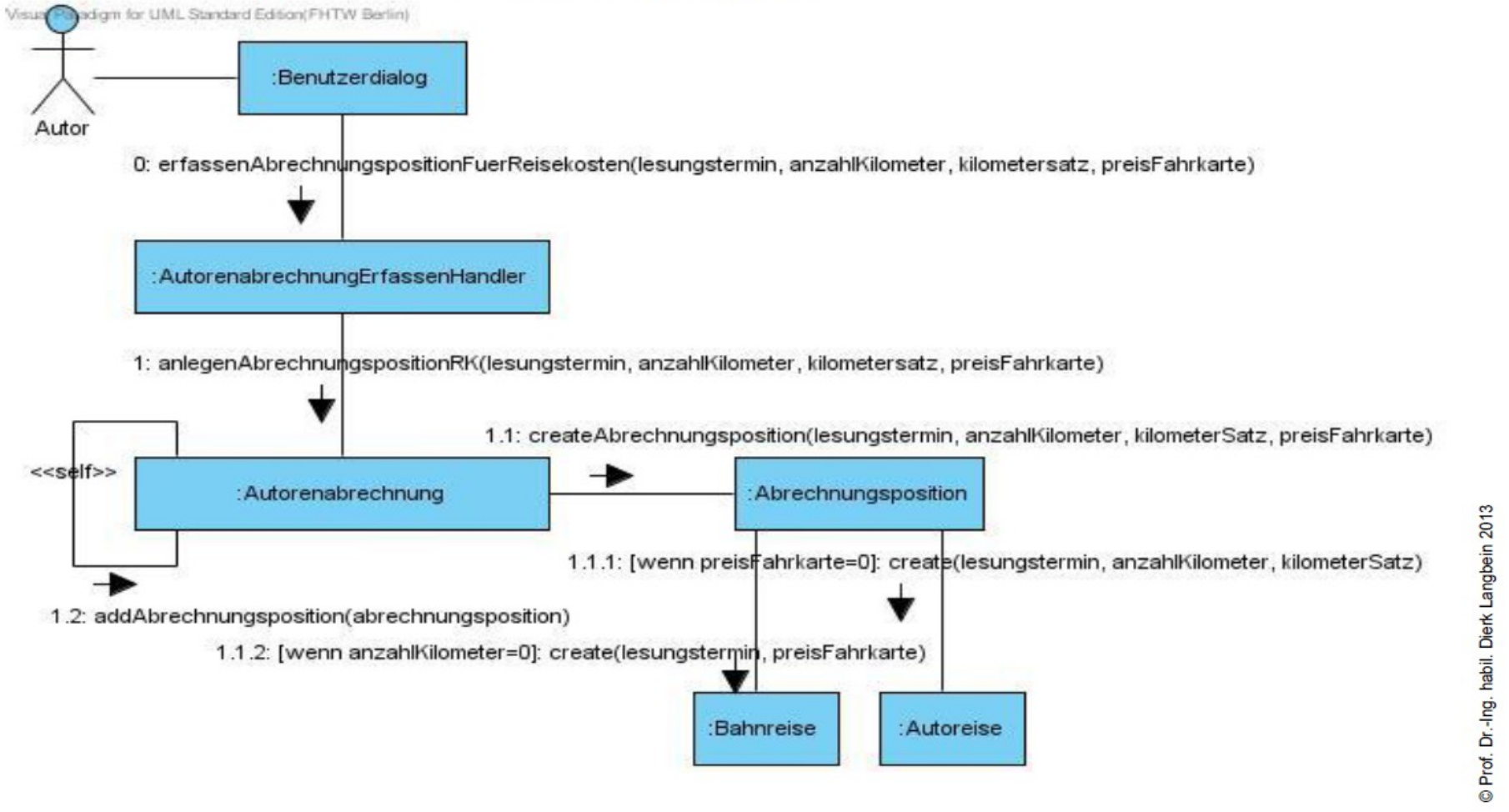
Realisierung von SAF-Operationen (2/6)

- Beispiel 2: SAF-Operation „erfassenAbrechnungspositionFuerReisekosten“
  Beschreibung mittels Schablone

Name der Operation	erfassenAbrechnungspositionFuerReisekosten(lesungstermin, anzahlKilometer, kilometerSatz, preisFahrkarte)
Verwendungsnachweis	SAF „Autorenabrechnung erfassen;
Vorbedingungen	Objekt der Klasse „Autorenabrechnung“ sowie Objekte vom Typ „Lesungstermin“, deren Datum in der Vergangenheit liegt und die als „nicht abgerechnet“ gekennzeichnet sind, sind im System vorhanden;
Nachbedingungen	1. Ein neues Objekt vom Typ „Abrechnungsposition“ wurde erzeugt. 2. Ein neues Objekt vom Typ „Bahnreisekosten“ bzw. vom Typ „Autoreisekosten“ wurde mit dem zugehörigen Objekt vom Typ „Lesungstermin“ verbunden. 3. Das neue Objekt vom Typ „Bahnreisekosten“ bzw. vom Typ „Autoreisekosten“ wurde mit dem zugehörigen Objekt der Klasse „Autorenabrechnung“ und mit dem Objekt vom Typ „Lesungstermin“ verbunden.

Realisierung von SAF-Operationen (3/6)

- Beispiel 2: SAF-Operation „erfassenAbrechnungspositionFuerReisekosten“
  Beschreibung mittels Kommunikationsdiagramm



Realisierung von SAF-Operationen (4/6)

- Beispiel 3: SAF-Operation „erfassenAbrechnungspositionFuerLesungstermin“
  Beschreibung mittels Schablone

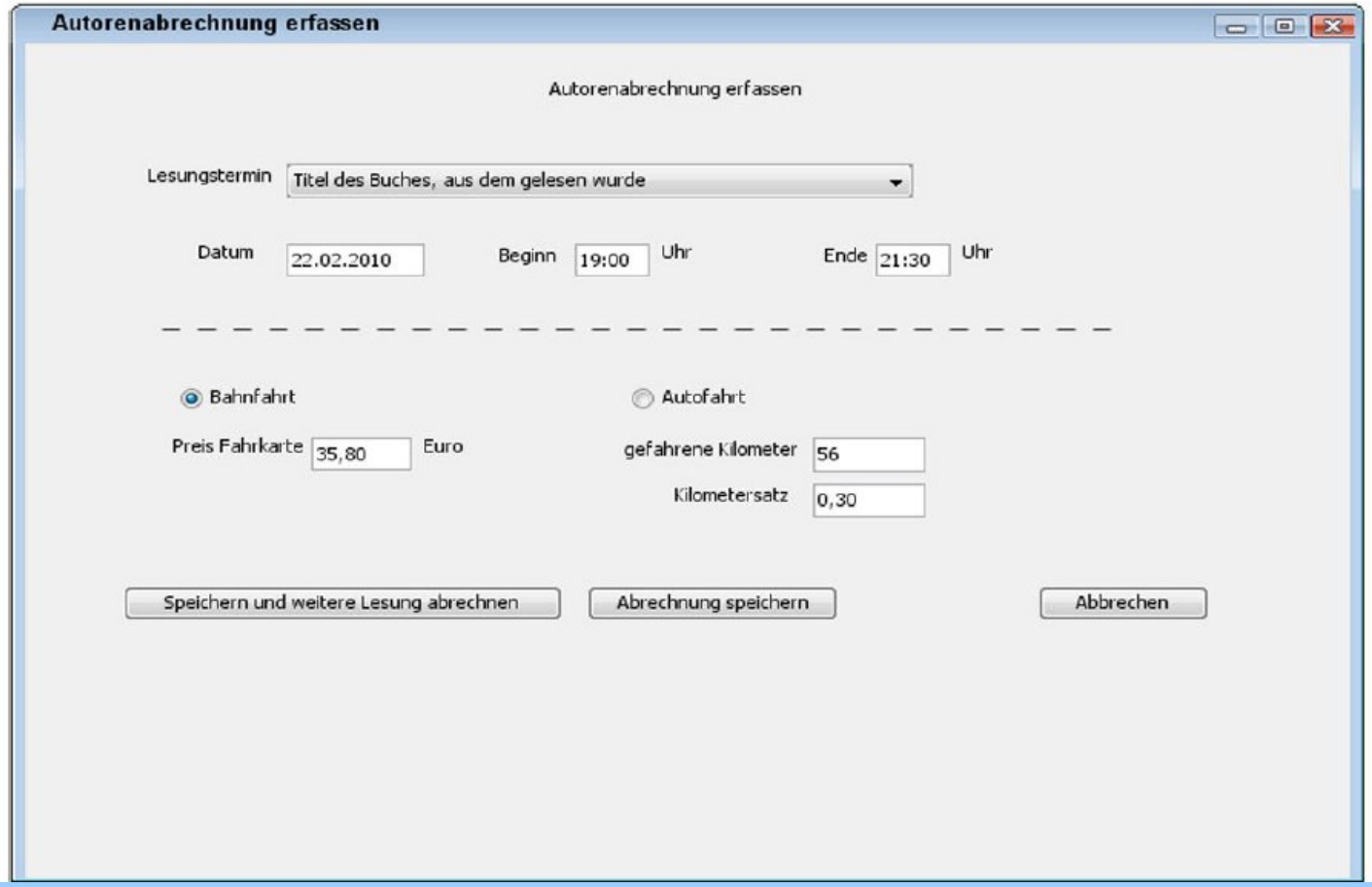
Name der Operation	erfassenAbrechnungspositionFuerLesungstermin(datum, beginnZeit, endeZeit)
Verwendungsnachweis	SAF „Autorenabrechnung erfassen;
Vorbedingungen	Objekt der Klasse „Autorenabrechnung“ sowie Objekte vom Typ „Lesungstermin“, deren Datum in der Vergangenheit liegt und die als „nicht abgerechnet“ gekennzeichnet sind, sind im System vorhanden;
Nachbedingungen	1. Ein neues Objekt vom Typ „Abrechnungsposition“ wurde erzeugt. 2. Das neue Objekt vom Typ „Abrechnungsposition“ wurde mit dem zugehörigen Objekt vom Typ „Lesungstermin“ verbunden. 3. Das neue Objekt vom Typ „Abrechnungsposition“ wurde mit dem zugehörigen Objekt der Klasse „Autorenabrechnung“ verbunden.

Erstellen Sie das Kommunikationsdiagramm für die SAF-Operation „erfassenAbrechnungspositionFuerLesungstermin()“!

Realisierung von SAF-Operationen (5/6)

- Beispiel: Operationen des SAF „Autorenabrechnung erfassen“

Entwurf für das Dialogfenster



Der Bezeichner (das Label) eines der Eingabeelemente ist nicht korrekt. Welcher?

Hinweis: Vergleiche mit dem Ablauf des SAF!

Realisierung von SAF-Operationen (6/6)

OOA-Klassenmodell mit Controller-Muster und Verwaltungsklassen für den SAF „Autorenabrechnung erfassen“

Warum wird oben das Wort „Klassenmodell“ verwendet? Weiter vorn war die Rede von „Fachklassen“ und noch weiter vorn von „Fachkonzeptklassen“. Erstellen Sie ein Sequenzdiagramm für die Operation „getGesamtbetrag()“!