

Informatik II: Modellierung  
Prof. Dr. Martin Glinz

Kapitel 8

# Klassen- und Objektmodelle



Universität Zürich  
Institut für Informatik

---

# Inhalt

---

8.1 Grundkonzepte

8.2 Klassenmodelle

8.3 Objektmodelle



# 8.1 Grundkonzepte

---

## Idee:

- Beschreibung eines Systems durch eine Menge von **Objekten**
- Jedes Objekt beschreibt einen **in sich geschlossenen Teil**
  - der **Daten**
  - der **Funktionalität**
  - des **Verhaltens**eines Systems
- Zusammenfassung **gleichartiger Objekte** zu **Klassen**
- Ähnlichkeiten mit **Datenmodellen**  
**Aber:** In Datenmodellen wird **nur** der **Datenaspekt** modelliert

# Beispiel: Ein Fahrausweis in einem öV-System



## Operationen

- Ausstellen
- Validieren
- Kontrollieren

## Daten

- Art: Einzelfahrt
- Anzahl Zonen: 3
- Validiert: ja
- Gültigkeitsdauer: 2
- Validierungsdatum: 03-05-27
- Validierungszeit: 16:23
- Validierungszone: 4

## Verhalten

- Der *ausgestellte* Fahrausweis muss vor Fahrtantritt *validiert* werden.
- Die Fahrausweise können jederzeit während der Fahrt *kontrolliert* werden

~~Fahrzeug Nr.  
Linie  
Entwerfer Nr.  
Farbband wechseln~~

# Objekte und Klassen

---

**Objekt (object)** – Ein **individuell erkennbares**, von anderen Objekten **eindeutig unterscheidbares Element** der „**Realität**“, das heißt des betrachteten Problem- oder Lösungsbereichs.

Beispiel: Der **konkrete Einzelfahrschein** für drei Zonen, validiert am 27.5.03 um 16:23 Uhr,...

**Klasse (class)** – 1. Eine eindeutig benannte Einheit, welche eine **Menge gleichartiger Objekte** beschreibt. 2. Ein **Typ**, welcher den **Aufbau**, die Bearbeitungsmöglichkeiten und das mögliche **Verhalten** von **Objekten dieses Typs** beschreibt.

Beispiel: Die Klasse **Fahrausweis** mit den Attributen Art, Anzahl Zonen, Validierungsdatum, Validierungszeit,...

# Zwei Sichten: Extension und Intension

---

- **Extensionale** Sicht: Eine Klasse ist eine **Menge** gleichartiger Objekte
- **Intensionale** Sicht: Eine Klasse ist ein **Typ**. Sie beschreibt, wie die Objekte der Klasse aufgebaut sind und wie diese Objekte bearbeitet werden können
- Beide Sichten werden häufig **vermischt**
- **Kompromissicht:**  
Eine Klasse **repräsentiert** eine Menge von Objekten. Sie **beschreibt** den **Aufbau**, die **Bearbeitungsmöglichkeiten** und das mögliche **Verhalten** ihrer **Objekte**.

# Eine Vielfalt von Modellen und Modellvarianten

---

- 1990 – 1998: **Vielfalt** von Modellen und Notationen, die in die Kategorie „Klassen- und Objektmodell“ fallen
- Zum Beispiel: Booch (1994), Coad und Yourdon (1991a, b), Jacobson et al. (1992), Rumbaugh et al. (1991), Wirfs-Brock et al. (1990)
- Seit ca. 1998 **dominiert** ein Industriestandard:  
Die **Unified Modeling Language UML** (Rumbaugh, Jacobson und Booch 1999), OMG (2007)
- In dieser Vorlesung:
  - Allgemeingültige, grundsätzliche **Konzepte**; kompatibel mit UML
  - **Notation**
    - für Klassenmodelle: UML
    - für Klassendefinitionen: angelehnt an UML

## 8.2 Klassenmodelle

---

**Klassenmodell** (class model) – eine Menge zusammengehöriger Klassendefinitionen

- Eine **Klassendefinition** definiert
  - **Attribute** der Objekte der Klasse (lokale Merkmale)
  - **Beziehungen** der Objekte der Klasse zu Objekten anderer Klassen oder der Klasse zu anderen Klassen
  - **Operationen**, die auf Objekten der Klasse oder auf der Klasse selbst möglich sind
- Zusätzlich kann jeder Klasse ein **Verhaltensmodell** zugeordnet sein

# Beziehungen

---

- **Assoziation:** Die Objekte einer Klasse sind Merkmale von Objekten einer anderen Klasse. Gilt in der Regel auch umgekehrt, d.h. Assoziationen sind meistens bidirektional
- **Benutzung:** Die Objekte einer Klasse benutzen Attribute oder Operationen einer anderen Klasse zur Bereitstellung ihrer eigenen Attribute und Operationen
- **Vererbung:** Eine Klasse ist Unterklasse einer anderen Klasse. Sie «erbt» alle Attribute und Operationen dieser Klasse, d.h. alle Objekte der Klasse verfügen über diese, ohne dass sie in der Klasse lokal definiert worden wären
- Zu den Assoziationen werden **Kardinalitäten** modelliert: wieviele Objekte der assoziierten Klasse müssen mindestens / dürfen höchstens mit einem Objekt der eigenen Klasse assoziiert sein.

# Beispiel einer Klassendefinition

---

Eine Klasse aus dem Klassenmodell eines (vereinfachten) Personalinformationssystems:

KLASSE Mitarbeiter im Monatslohn  
UNTERKLASSE von Mitarbeiter  
ATTRIBUTE (Name, Kardinalität, Wertebereich)  
Leistungslohnanteil [1,1]:CHF  
Überzeitsaldo [1,1]:Stunden  
Ferienguthaben [1,1]:Tage  
...  
BEZIEHUNGEN (Name, Kardinalität, mit Klasse)  
eingestuft in [1,1]:Lohnklasse

# Beispiel einer Klassendefinition – 2

---

## OPERATIONEN

Lohn zahlen

Voraussetzung: Mitarbeiter ist aktiv

Ergebniszusicherung: Zahlungsauftrag zugunsten des Mitarbeiters ist erteilt mit Grundlohn aus Lohnklasse und Leistungslohnanteil

## BENUTZT (Klasse.Operation)

Zahlungsauftrag.erteilen

**Hinweis:** Da Mitarbeiter im Monatslohn eine Unterklasse von Mitarbeiter ist, werden alle Attribute (zum Beispiel Name und Vorname), Beziehungen und Operationen der Klasse Mitarbeiter automatisch übernommen, ohne dass sie nochmals definiert werden müssten.

## Aufgabe 8.1

In einem Produktionsplanungssystem sind Lagerartikel charakterisiert durch: die Artikelnummer, den Namen, die vorrätige Anzahl, die insgesamt reservierte Anzahl und den Preis. Ferner gibt es Funktionen, welche die aktuell verfügbare Anzahl liefern bzw. eine bestimmte Menge für einen Produktionsschritt reservieren.

Stellen Sie diese Information in Form einer Klassendefinition dar.

# Klassendiagramme

---

- **Klassendiagramme** beschreiben die Klassen und ihre Zusammenhänge (analog zu Entity-Relationship-Diagrammen).
- Es gibt eine Vielzahl von verschiedenen Notationen; wesentliche Gemeinsamkeit:
  - **Rechtecke** für Klassen
  - **Linien** für Assoziationen
- Heute **dominiert UML** (Unified Modeling Language) als **Notation**

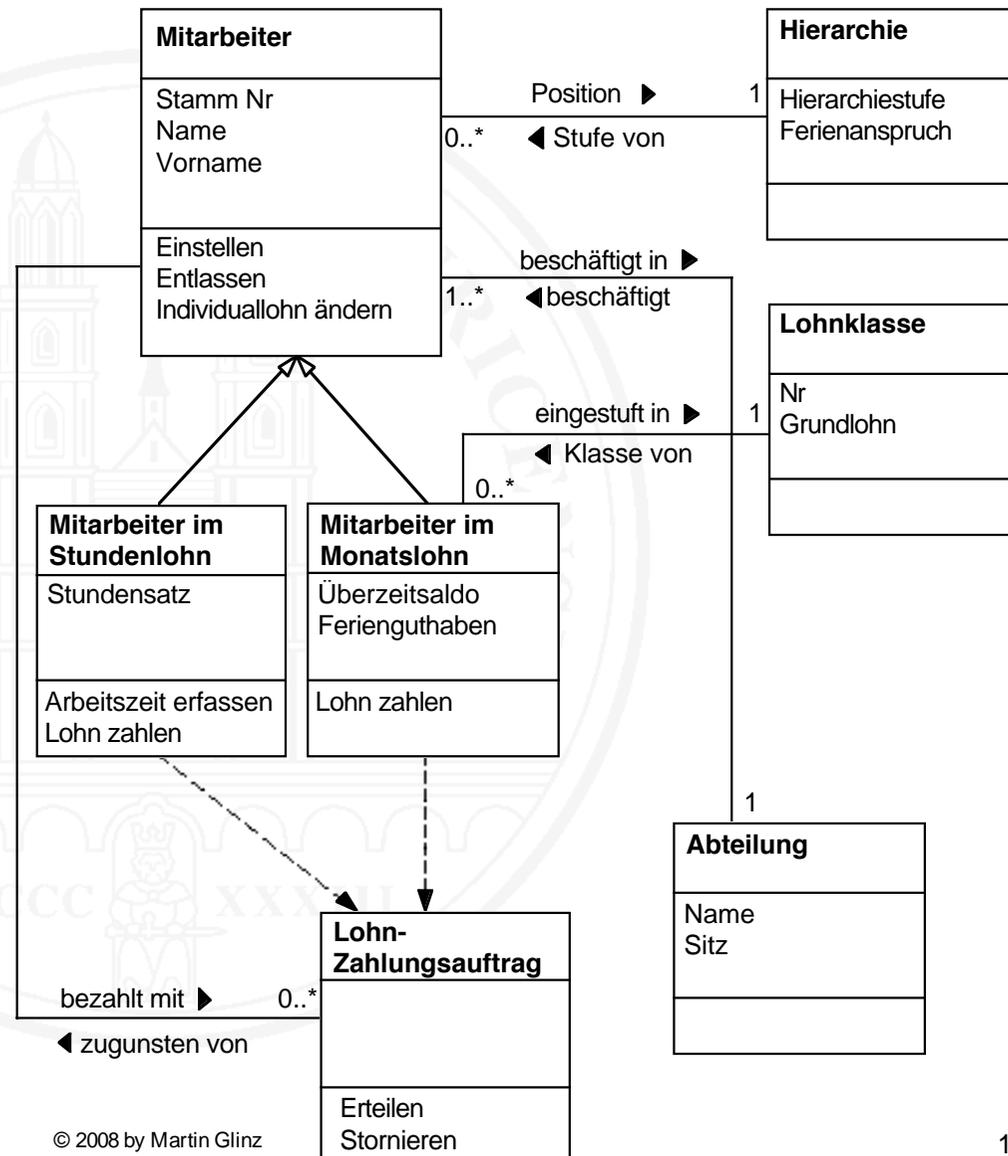
# Klassendiagramme in UML

---

- **Rechtecke** für Klassen
- Unterteilung der Klassensymbole durch zwei **Linien**: Oben der Klassenname, in der Mitte die wichtigsten Attribute, unten die wichtigsten Operationen
- **Linien** für Assoziationen
- **Gestrichelte Pfeile** für Benutzung
- **Linien mit leeren Pfeilspitzen** für Vererbung
- **Pfeilspitzen** geben die Beziehungsrichtung an
- **Kardinalitäten**: Die erste Zahl (notiert bei der Klasse, auf welche die Pfeilspitze zeigt) ist die Mindestzahl, die zweite Zahl die Höchstzahl der verknüpften Objekte. Ist nur eine Zahl angegeben, so sind Mindest- und Höchstzahl gleich.

# Beispiel eines Klassendiagramms

Klassendiagramm eines  
Personalinformations-  
systems in UML-Notation  
(stark vereinfacht)



## Aufgabe 8.2

Die folgenden Klassen aus dem Modell eines Produktionsplanungssystem seien gegeben: Lagerartikel (vgl. Aufgabe 8.1), Artikel, Sonderartikel, Lager, Lieferant. Lagerartikel und Sonderartikel sind Unterklassen von Artikel.

Zu diesen Klassen sind folgende Attribute bekannt:

- Artikel: Artikelnummer, Name
- Lager: Name, Ort
- Lieferant: Nummer, Name

Ferner seien folgende Operationen und Beziehungen bekannt:

- Lagerartikel sind in genau einem Lager eingelagert
- Zu jedem Lagerartikel gibt es mindestens einen Lieferanten
- Für Sonderartikel gibt es eine Beschaffungsoperation.

Modellieren Sie diese Informationen als Klassendiagramm in UML-Notation.

# Verhaltensbeschreibung

---

- Verhalten kann in den heutigen Klassenmodellen **nicht integriert** beschrieben werden
  - Aktueller Forschungsgegenstand, u.a. in Zürich
- Stattdessen:
  - **Lokale** Verhaltensmodellierung durch **separates Verhaltensmodell** für jede Klasse
  - Mittel: In der Regel **Statecharts**
- **Globales** Verhalten (über Klassen hinweg)
  - In vielen Modellen (einschl. UML) **nicht modelliert**
  - **Möglich** über ein **neben dem Klassenmodell** stehendes globales Verhaltensmodell
  - aber: **Bruch** des Prinzips der Objektorientierung

## 8.3 Objektmodelle

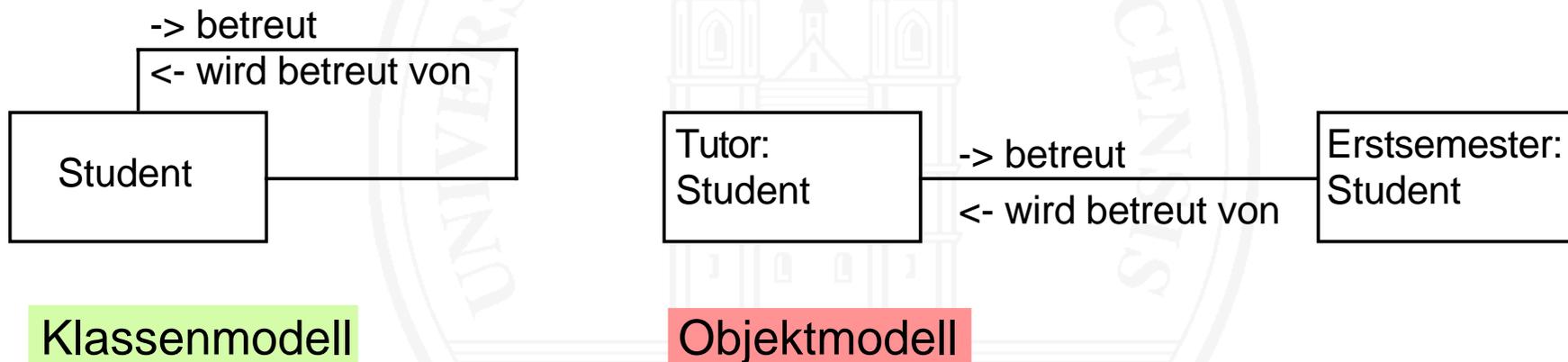
---

- **Objektmodelle** können als **Alternative** oder als **Ergänzung** zu Klassenmodellen verwendet werden.
- **Keine konkreten Objekte** („der Tutor Peter Meier“) sondern **abstrakte Objekte**, die als Repräsentant für konkrete Objekte stehen („ein Tutor“).
- Hintergrund: **Klassenmodelle versagen**, wenn
  - der **konkrete Verwendungskontext** eines Objekts modelliert werden soll
  - **verschiedene Objekte der gleichen Klasse** zu modellieren sind
  - ein **Modell hierarchisch** in Komponenten **zerlegt** werden soll

[Joos et al. 1997, Glinz, Berner, Joos 2002]

# Klassenmodelle vs. Objektmodelle

- **Klassenmodelle** können **verschiedene Objekte** der **gleichen Klasse** nicht modellieren
- ⇒ **Objektmodelle** sind in solchen Situationen **überlegen**



- Ganz auf Klassenmodelle verzichten?
- ⇒ Geht nicht, weil Vererbungszusammenhänge auf Klassen, nicht auf Objekten definiert sind

# Objektmodelle in UML

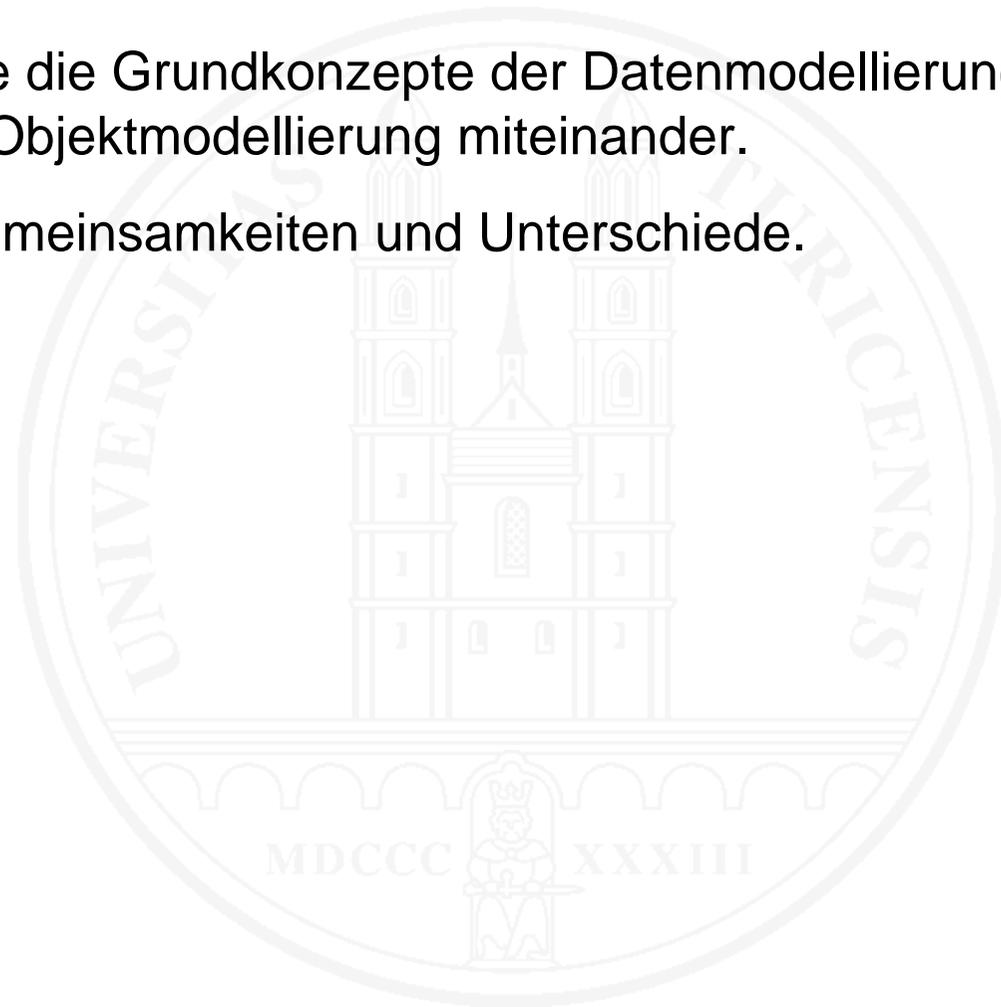
---

- UML 1 verwendet Objektmodelle nur zur **Ergänzung** eines Klassenmodells (zur **auszugsweisen Modellierung** des **Arbeitskontextes** der Objekte einer Klasse)
- UML 2 enthält neu die Möglichkeit **hierarchischer Objektmodelle**: die so genannten **Strukturklassen** in **Kompositionsstrukturdiagrammen** sind faktisch Hierarchien abstrakter Objekte

## Aufgabe 8.3

Vergleichen Sie die Grundkonzepte der Datenmodellierung und der Klassen- bzw. Objektmodellierung miteinander.

Nennen Sie Gemeinsamkeiten und Unterschiede.



# Methodik der Klassen- und Objektmodellierung

---

- Erweiterung der von der **Datenmodellierung** bekannten Verfahren:
- **Objektanalyse**
  - Liefert Klassen, Attribute, Assoziationen
  - Zusätzlich: Auswertung der Handlungen
    - Operationen, Verhalten
  - Zusätzlich: Untersuchung von Gemeinsamkeiten und Spezialfällen
    - Vererbungszusammenhänge
- **Ereignis-Reaktions-Analyse**
  - Zusätzlich: Auswertung der geforderten Reaktionen
    - Operationen, Verhalten
  - Zusätzlich: Untersuchung von Gemeinsamkeiten und Spezialfällen
    - Vererbungszusammenhänge
- **Verhaltensmodelle**: Analog zum im Kapitel 7 beschriebenen Vorgehen

## Aufgabe 8.4

Zu entwickeln sei ein Informationssystem für Pauschalreisen. Folgender Auszug aus der Problemstellung ist gegeben:

Das System kennt das Angebot an Pauschalreisen und die zugehörigen Anbieter. Zu jeder Reise im Angebot sind Hotelname, Kategorie, Ort, Reisedaten und Preis bekannt.

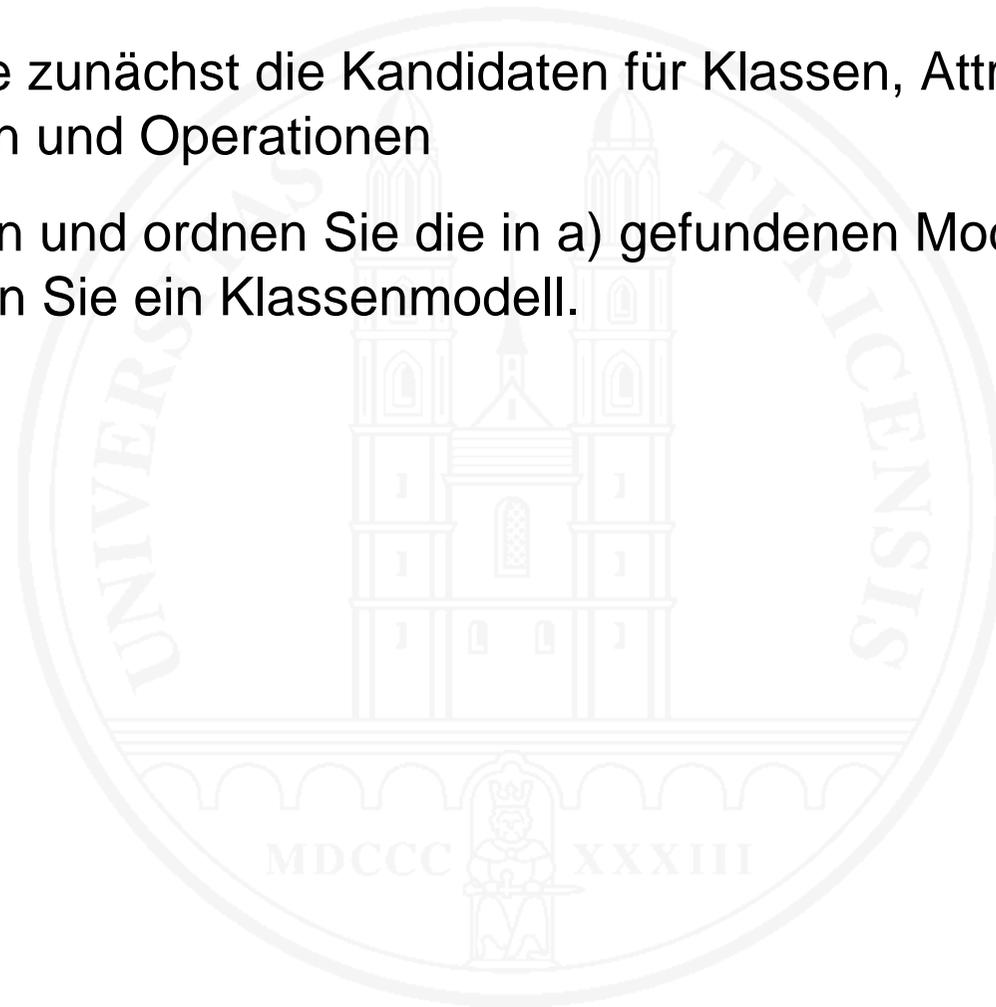
Ein Kunde kann das Angebot durchsehen und bei Interesse eine Reise provisorisch oder fest buchen. Zu jeder Buchung müssen Datum, Preis der Reise sowie Name, Vorname, Geburtsdatum und Adresse des Kunden vermerkt werden. Der Kunde kann eine Buchung stornieren.

Jede Buchung/Stornierung wird der Buchhaltung mitgeteilt, welche sich um Inkasso/Rückerstattungen kümmert.

Modellieren Sie die genannten Anforderungen in einem Klassenmodell. Lassen Sie Details der Buchhaltung beim Modellieren weg.

## Aufgabe 8.4 (Fortsetzung)

- a) Ermitteln Sie zunächst die Kandidaten für Klassen, Attribute, Beziehungen und Operationen
- b) Klassifizieren und ordnen Sie die in a) gefundenen Modellelemente und zeichnen Sie ein Klassenmodell.



# Literatur

---

Booch, G. (1994). *Object Oriented Analysis and Design with Applications*. Second Edition. Redwood City, Ca.: Benjamin/Cummings.

Coad, P., E. Yourdon (1991a). *Object-Oriented Analysis*. 2nd edition. Englewood Cliffs, N.J.: Prentice Hall. [auf Deutsch: Objektorientierte Analyse, 1994 im gleichen Verlag]

Coad, P., E. Yourdon (1991b). *Object-Oriented Design*. Englewood Cliffs, N.J.: Prentice Hall. [auf Deutsch: Objektorientiertes Design, 1994 im gleichen Verlag]

Glinz, M., S. Berner, S. Joos (2002). Object-Oriented Modeling With ADORA. *Information Systems* **27**, 6. 425-444.

Jacobson, I., M. Christerson, P. Jonsson, G. Övergaard (1992). *Object-Oriented Software Engineering: A Use Case Driven Approach*. Amsterdam; Reading, Mass. [u.a.]: Addison-Wesley.

Joos, S., S. Berner, M. Arnold, M. Glinz (1997). Hierarchische Zerlegung in objektorientierten Spezifikationsmodellen. *Softwaretechnik-Trends*, **17**, 1 (Feb. 1997). 29-37.

Meyer, B. (1998). *Object Oriented Software Construction*. 2nd ed. Englewood Cliffs, N.J.: Prentice Hall.

Oestereich, B. (2004). *Objektorientierte Softwareentwicklung: Analyse und Design mit der UML 2.0*. 6. Auflage. München: Oldenbourg Verlag.

OMG (2007). *Unified Modeling Language: Superstructure*, version 2.1.1. OMG document formal/2007-02-05

Rumbaugh, J., M. Blaha, W. Premerlani, F. Eddy, W. Lorensen (1991). *Object-Oriented Modeling and Design*. Englewood Cliffs, N.J.: Prentice Hall.

[auf Deutsch: *Objektorientiertes Modellieren und Entwerfen*, München: Hanser, 1993]

# Literatur – 2

---

Rumbaugh, J., I. Jacobson, G. Booch (1999). *The Unified Modeling Language Reference Manual*. Reading, Mass., etc.: Addison-Wesley.

Wirfs-Brock, R., B. Wilkerson, L. Wiener (1990). *Designing Object-Oriented Software*. Englewood Cliffs, N.J.: Prentice Hall.

[auf Deutsch: *Objektorientiertes Software Design*, München: Hanser, 1993]

