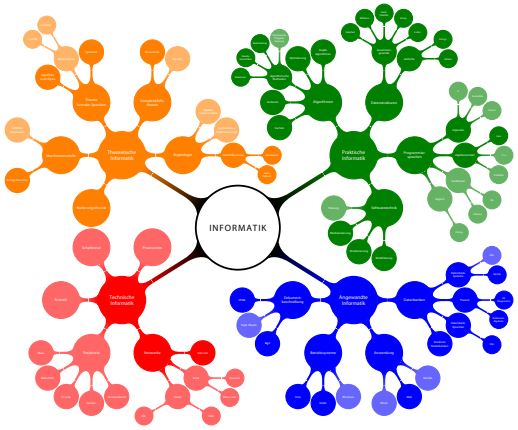
UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR
THEORETISCHE INFORMATIK

Kapitel 24

Objektorientierte Modellierung

Das Abbild der Wirklichkeit im Rechner

Vorlesung Einführung in die Informatik 1 vom 28. Januar 2014 von Till Tantau



Lernziele von Kapitel 24

1. Modellierung von Daten mittels Klassen verstehen
2. Klassen zur Modellierung von Fallbeispielen erstellen können

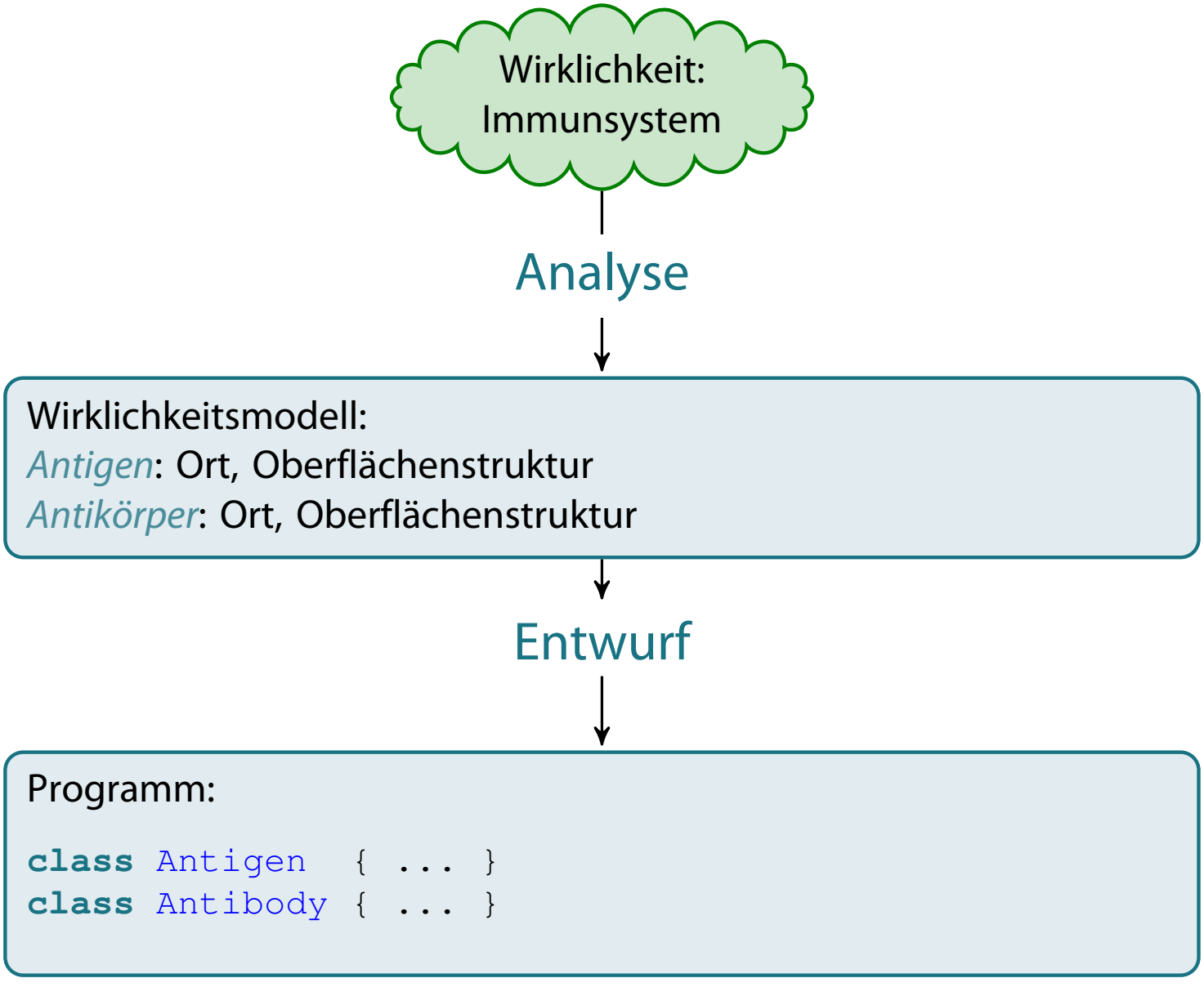
Gliederung von Kapitel 24

Arten der Modellierung

- Bei der *Modellierung* (im IT-Kontext) will man *einen Ausschnitt der Wirklichkeit* abbilden.
- Dazu *analysiert* man, was wichtig ist, und erstellt daraus ein *Modell*.
- Es gibt viele Arten von Modellen, wie zum Beispiel:
  - *Datenmodell (Wer? Was?)*
  - Aktivitätsmodell (Wer? Wann?)
  - Sequenzmodell (Zuerst? Danach?)
  - Anwendungsfallmodell (Was wäre wenn?)
- Die Modelle sind Startpunkte für die Entwicklung der Software.
- Während der Entwicklung schaut man immer wieder, ob die Software den Modellen (noch) entspricht.

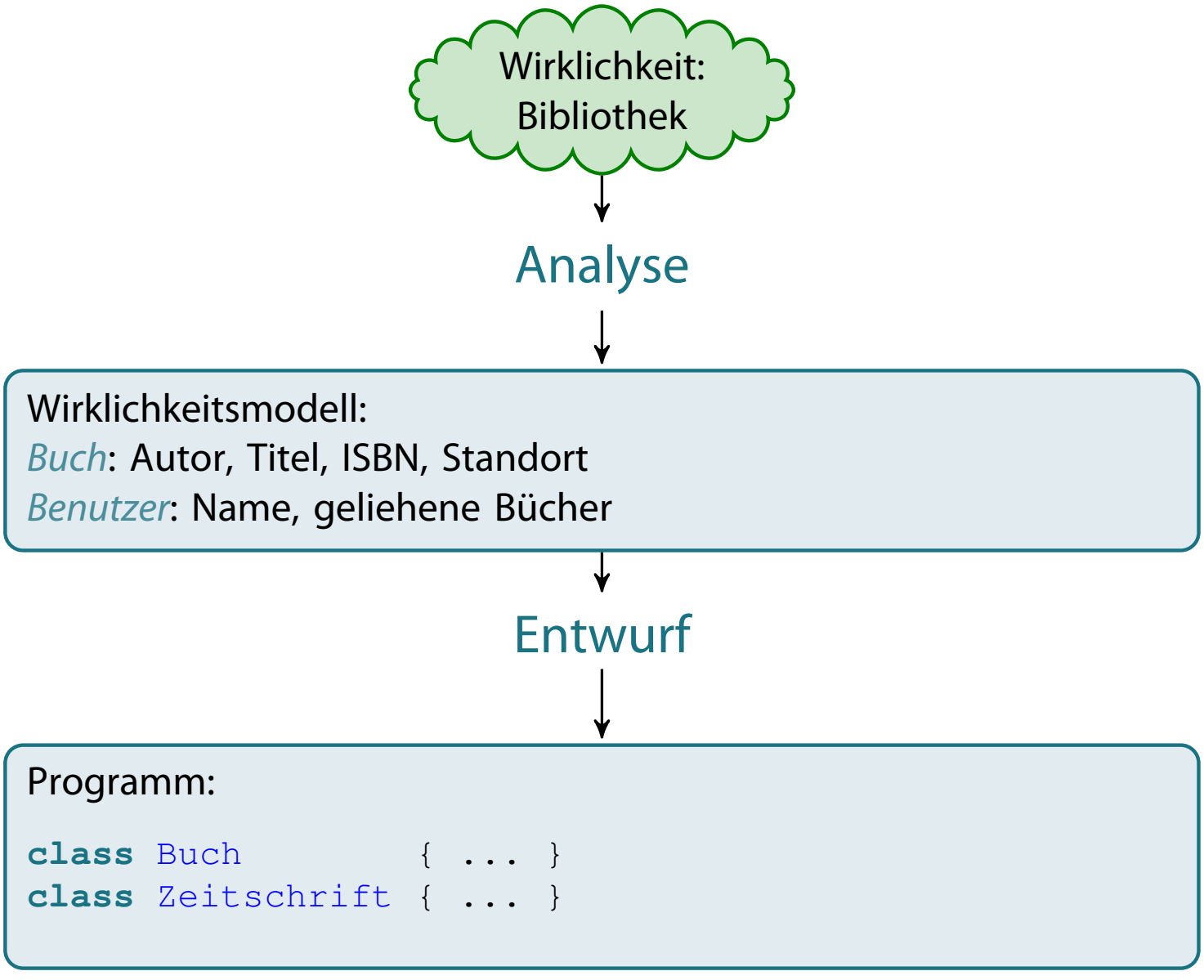
Ein Modell bildet ein Problem ab.

Beispiel: Das Immunsystem



Ein Modell bildet ein Problem ab.

Beispiel: Eine Bibliothek



Es gibt viele Verfahren, Modelle zu erstellen.

- Das Teilgebiet der *Softwaretechnik* beschäftigt sich (unter Anderem) damit, welche Verfahren besonders geeignet sind, Analysen und Modellierung durchzuführen.
- Genau wie es viele Programmiersprachen gibt, gibt es viele Verfahren zur Analyse und Modellbildung;
- und genau wie Programmiersprachen Vor- und Nachteile haben und einige mehr en vogue sind als andere, haben Analyse- und Modellierungsmethoden Vor- und Nachteile und sind en vogue oder auch nicht.
- Wir werden im Folgenden eine einfache Form der weit verbreiteten *objektorientierten Analyse und Modellierung* anschauen.
- (Das trifft sich, da Java eine objektorientierte Sprache ist.)

Hauptsatz der Softwareentwicklung.

Merke

Erst nachdenken, was eigentlich gebraucht wird; programmiert wird später.

- Dies widerspricht der Art, wie man im Allgemeinen Programmieren erlernt: Man bekommt klar beschriebene Aufgaben, die dann in Code umgesetzt werden müssen, wodurch man einen Aspekt der Programmierung erlernt.
- »In der Wirklichkeit« gibt es aber keine klaren Aufgaben sondern diffuse Wünsche und Unbehaglichkeiten.
- Softwaretechniker beklagen zu Recht, dass viel zu viel »wild rumprogrammiert« wird.

Modellierung bei größeren Projekten.

1. Man teilt das Problem *top-down* in Teilbereich ein (siehe das Kapitel »Modularisierung im Großen«).
2. Für jedes Paket führt man nun *bottom-up* eine Analyse durch: Man findet alle relevanten Begriffe und Beziehungen und ordnet sie.
3. Dann führt man einen *Entwurf* durch: Geeignete Klassen, Objekte, Attribute und Methoden werden identifiziert, die tatsächlich in der Software vorkommen sollen.

Wozu macht man eine Analyse?

- Ziel der Analyse ist es herauszufinden, was alles zum zu modellierenden Ausschnitt der Realität gehört. Dies schließt alles ein, was für die spätere Anwendung relevant sein könnte.
- *Insbesondere kann die Analyse Dinge einschließen, die später gar nicht im Programm vorkommen werden.*
  Beispiel: Man möchte Software für einen Geldautomaten entwickeln. Dann wird in der Analyse sicherlich der Mensch vorkommen, der Geld haben will – in der späteren Software gibt es aber wahrscheinlich keine `class Human`.

Ein einfaches Vorgehen bei der statischen objektorientierten Analyse.

Man kann (und sollte) viele *Aspekte* eines Systems analysieren wie Verhalten, Datenaufbau, Anwendungsfälle und vieles mehr. Wir beschränken uns auf das *statische Datenmodell*, welches man so findet:

1. Man erstellt eine Liste aller *Begriffe*, die für das Gesamtsystem wichtig erscheinen.
2. Dann wird »aufgeräumt« und »vereinheitlicht«: gleichartige Begriffe werden zusammengefasst (Clustering) und Standardbegriffe festgelegt.
3. Man erstellt eine Liste aller *Beziehungen* zwischen den Begriffen wie:
   - »ist ein Teil von« (»part-of«),
   - »ist ein Spezialfall von« (»is-a«),
   - »hängt ab von«,
   - . . .

Die Wirklichkeit: Eine Bibliothek

- Es soll eine Software zur Verwaltung des Bestandes einer Institutsbibliothek erstellt werden.
- Sie sind bei *Molecular Sheep* angestellt und sollen diese Software entwerfen.
- Zunächst soll nur die Buchverwaltung durch Software unterstützt werden (und nicht die Ausleihe, Bestellung, etc.).

Schritt 1: Die Begriffsliste.

Der erste Schritt der Analyse war die Erstellung einer Liste interessanter Begriffe wie:

- Buch,
- Zeitschrift,
- Raum,
- Regal,
- Signatur,
- Buchtitel,
- Autor,
- Artikeltitel, . . .

Zur Übung

Ergänzen Sie die Liste. Die Ergebnisse werden dann gesammelt.

Schritt 2: Vereinheitlichung.

Der zweite Schritt ist, Begriffe zu vereinheitlichen. Heraus kommt eine Art Glossar:

Titel	Dieser Begriff schließt alle Arten von Titeln wie Buchtitel, Artikeltitel oder Zeitschriftentitel ein.
Autor	Dieser Begriff bezeichnet speziell nur die Personen, die selber einen Text geschrieben haben. Er bezeichnet nicht Herausgeber, dies ist ein Extrabegriff.
Herausgeber	Siehe Autor. Eine Person kann sowohl Autor wie Herausgeber eines Buches sein.
Buch	. . .

Zur Übung

Führen Sie das Clustering für die Liste aus der vorherigen Aufgabe durch. Die Ergebnisse werden dann gesammelt.

Schritt 3: Beziehungen finden.

Der nächste Schritt ist, Beziehungen zwischen den jetzt ermittelten Begriffen zu finden.

- Ein Buch *hat* Autoren und/oder Herausgeber.
- Eine Zeitschrift *hat* einen Herausgeber.
- Ein Buch *steht* in einem Regal.
- Ein Regal *steht* in einem Raum.
- Ein Konferenzband *besteht aus* Artikeln.

Zur Übung

Finden Sie möglichst viele Beziehungen, in denen ein *Autor* involviert ist.

Der Entwurf bildet einen Teil der Wirklichkeit ab.

- Ziel des *Entwurfs* ist es, die Wirklichkeit in Software abzubilden.
- Bei einer objektorientierten Sprache ist dies *besonders einfach*, da man viele Begriffe (wie »Buch«) direkt auf Klassen der Programmiersprache abbilden kann.
- Es wird aber *nicht* zu jedem Begriff eine Klasse eingeführt; einige Begriffe werden sogar *gar nicht* in der Software widergespiegelt.

Ein einfaches Vorgehen bei dem statischen objektorientierten Entwurf.

Wieder behandeln wir nur die statischen Datenaspekte. Wir erstellen also nur ein *statisches Datenmodell*.

1. Man teilt die in der Analyse gefundenen Begriffe in *Klassen, Objekte und Attribute* ein. Dies gilt nur für die Begriffe, die tatsächlich in Software abgebildet werden sollen.
2. Man legt so genannte *Vererbungsbeziehungen* zwischen den Klassen fest. (Lassen wir in dieser Vorlesung weg.)
3. Man legt geeignete Methoden für die Klassen fest. (Dazu benötigt man eigentlich eine *Verhaltensanalyse*, die wir aber nicht gemacht haben.)

Schritt 1: Identifikation der Klassen, Objekte, Attribute.

Man geht die Begriffe durch und entscheidet, ob sie Klassen sein sollen oder Attribute.
Aus den Begriffen »Autor«, »Name«, »Buch« und »ISBN« könnte werden:

```
class Autor {
  String vorname;
  String nachname;
}

class Buch {
  Autor[] autoren; // Liste der Autoren
  String  titel;
  String  isbn;
}
```

Zur Übung

Wie sollten die Begriffe »Herausgeber«, »Regal« und »Gebäude« modelliert werden?

Schritt 3: Bestimmung der Methoden.

- Die Objekte der entworfenen Klassen haben ein *erwünschtes Verhalten*, das durch *Methoden* modelliert wird.
- (Die gewünschte Verhalten findet man durch eine *Verhaltensanalyse*, die wir aber nicht gemacht haben.)
- Die Methoden werden *noch nicht implementiert*.

Was können Bücher?

```
class Buch {
  // ...

  void   regalwechsel (Regal wohin) {...}
  void   ausmusterung ()            {...}
  int    seitenzahl   ()            {...}
}
```

Zur Übung

Nennen Sie sinnvolle Methoden für eine Klasse `Regal`.

Zusammenfassung

Ablauf einer (statischen) OO-Analyse

1. Teile das Thema top-down in sinnvolle Teile auf.
2. Erstelle eine Liste aller Begriffe, die für das Thema irgendwie wichtig sein könnten.
3. Vereinheitliche die Begriffe.
4. Bestimme die Beziehungen der Begriffe untereinander.

Ablauf eines OO-Entwurfs

1. Entscheide, welche Begriffe aus der Analyse überhaupt in Software abgebildet werden sollen.
2. Lege Klassen und Attribute fest, die die Begriffe repräsentieren.
3. Lege fest, welche Methoden die Klassen haben.