BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

Software Engineering 1

Klassendiagramme I

Agenda Heute

- Grundlagen von Klassendiagrammen
- Übung zur Klassenmodellierung

2

Lernziele

- UML Basiskonzepte für Klassendiagramme verstehen
- Klassendiagramme mit einfachen Assoziationen erstellen können

3

Klassendiagramme I

Grundlagen der Klassendiagramme

4

Klassendiagramme

- Modellierung von Klassen
  - Attribute
  - Operationen (Methoden)
  - Eigenschaften
- Modellierung von Beziehungen
  - Assoziation, Generalisierung, Aggregation und Komposition
- Eine der wichtigsten UML Diagrammarten

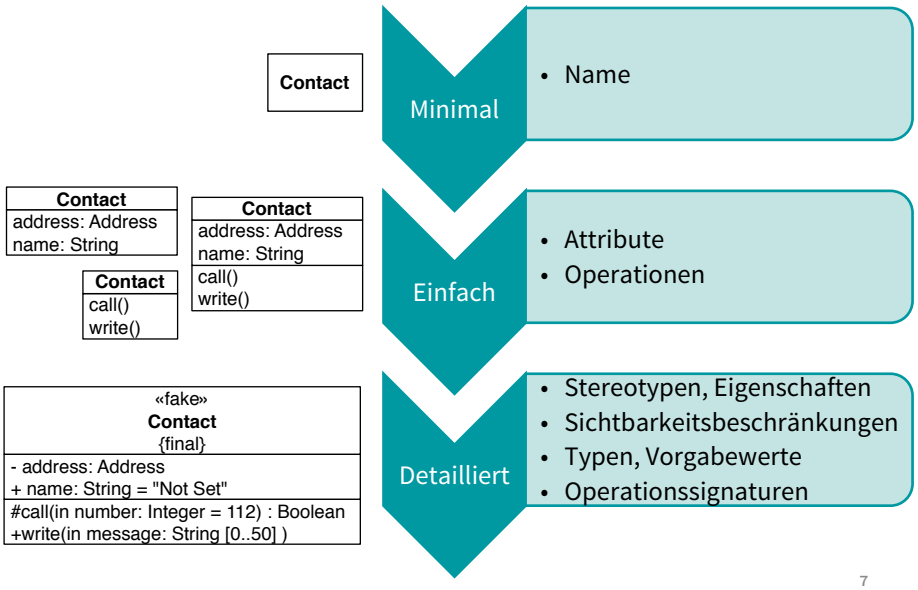
5

Klassen

- Stellen Mengen von gleichartigen Objekten dar
  - Eine Klasse definiert einen Typ
  - Eine Klasse bildet einen Namensraum
- Besitzen strukturelle Merkmale
  - Attribute
- Besitzen Verhaltensmerkmale
  - Operationen (Methoden)

6

Notation von Klassen in UML



7

Einschränkung der Sichtbarkeit

+
- **Öffentlich**
- Sichtbar für alle Ausprägungen

-
- **Privat**
- Nur für Ausprägungen der eigenen Klasse sichtbar

#
- **Geschützt**
- Für Ausprägungen und Spezialisierungen der eigenen Klasse sichtbar

~
- **Paket**
- erlaubt den Zugriff für alle Elemente innerhalb des eigenen Pakets

8

Multiplizitäten

1	• Genau eins, entspricht **1 .. 1**
0 .. 1	• Optional: Entweder eins oder keins
*	• Eine beliebige Anzahl, entspricht **0 .. ***
1 .. *	• Eine beliebige Anzahl, aber mindestens eins
2 .. 3	• Mindestens zwei, höchstens aber drei

9

static

- Attribut oder Operation gehört zum Typ, nicht zur Instanz des Typs
- Notation: Name wird unterstrichen

10

abstract

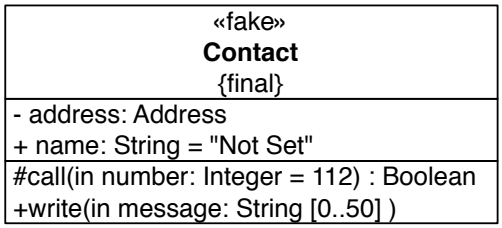
- Klasse oder Operation besitzt keine Implementierung
- Notation: Name wird *kursiv* geschrieben
  - Alternativ `{abstract}` dazuschreiben

Konstruktoren

- Durch «constructor» Stereotyp ausgedrückt
  - Vor den entsprechende Methoden:

```
«constructor» + MeineKlasse()
```

Detaillierte Notation



Attribute

```
[Sichtbarkeit] [/] name [: Typ] [[Multiplizität]]
[= Vorgabewert] [{Eigenschaftswert*}]
```

Operationen

```
[Sichtbarkeit] name ([Parameter]) [: Rückgabetyp]
[[Multiplizität]] [{Eigenschaftswert*}]
```

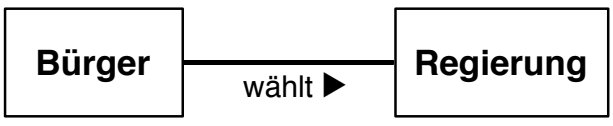
Parameter (Komma-getrennte Liste)

```
[Übergaberichtung] name : Typ [[Multiplizität]]
[= Vorgabewert] [{Eigenschaftswert*}]
```

13

Assoziation

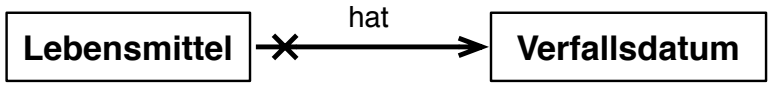
- Assoziation sind Beziehungen zwischen Typen
  - Meist Klassen
- Notation: Durchgezogene Linie
- Assoziationen können Namen haben
  - Es kann eine Leserichtung angegeben werden
- Kommunikation zwischen Objekten findet über Assoziationen statt
  - z.B. Methodenaufrufe



14

Navigierbarkeit

- Es kann die Richtung eingeschränkt werden, in die eine Kommunikation möglich
- Beispiel: Lebensmittel kennt Verfallsdatum, aber nicht umgekehrt



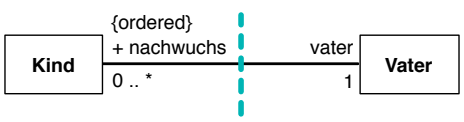
Navigation von Lebensmittel nach Verfallsdatum erlaubt

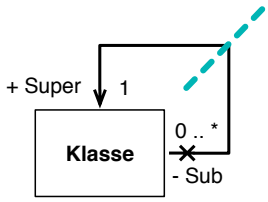
Navigation von Verfallsdatum nach Lebensmittel verboten

15

Assoziationsenden

- Assoziationen besitzen Enden
  - Meist zwei, n-äre Assoziation sind aber auch möglich
    - Notation: Raute
- Assoziationsenden sind annotierbar mit
  - Namen
  - Multiplizitäten
  - Sichtbarkeitsbeschränkungen
  - Speziellen Eigenschaften
    - z.B. Ordnung, Eindeutigkeit
- Beispiele:

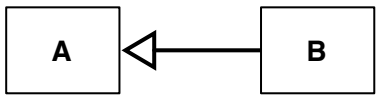




Reflexive Assoziation

16

Generalisierung / Spezialisierung

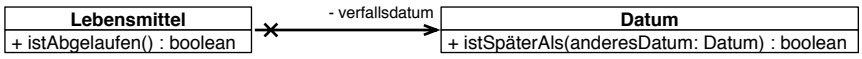


*„**B** ist ein **A** “*

- Binäre Beziehung zwischen zwei UML Typen
  - Ein speziellerer Typ (hier B)
  - Ein generellerer Typ (hier A)
- Notation: Pfeil mit großer, ungefüllter Spitze
  - An der Seite des generelleren Typs
- Der speziellere Typ verfügt dadurch über alle Struktur- und Verhaltensmerkmale des generelleren Typen
  - Bei Klassen sind das die Attribute und Operationen

17

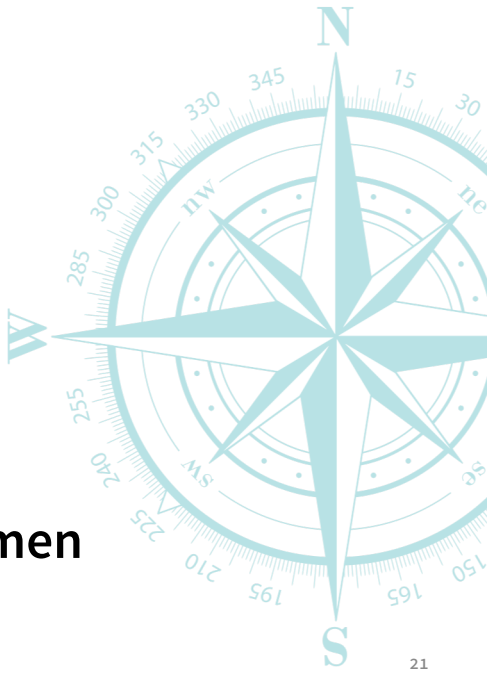
Beispiel zu Assoziationen




```
class Datum {
public:
    bool istSpaeterAls(Datum *anderesDatum) {
        // Datum später als anderesDatum ?
    }
};

class Lebensmittel {
private:
    Datum *verfallsdatum;
public:
    bool istAbgelaufen() {
        Datum jetzt = ...; // Heutiges Datum holen
        return jetzt.istSpaeterAls(verfallsdatum);
    }
};
```

18

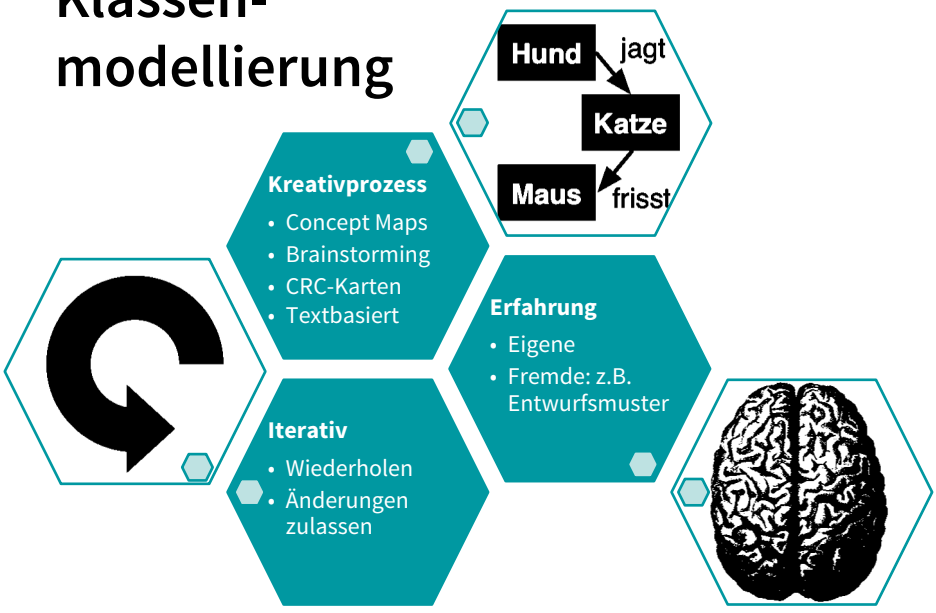


Klassendiagramme I

Erstellung von Klassendiagrammen

21

Klassen-modellierung



22

Verb/Substantiv Methode

Sie brauchen zwei verschiedenfarbige Stifte

- Anforderungen / Anwendungsfälle lesen
  - Verben mit **einer Farbe** unterstreichen
  - Substantive mit **der anderen** unterstreichen
- Substantive in zwei Gruppen teilen
  - Kandidaten für **Klassen**
  - Kandidaten für **Attribute**
- Die Verben sind Ihre **Methoden**kandidaten

23

Use Case #5: CD wiedergeben

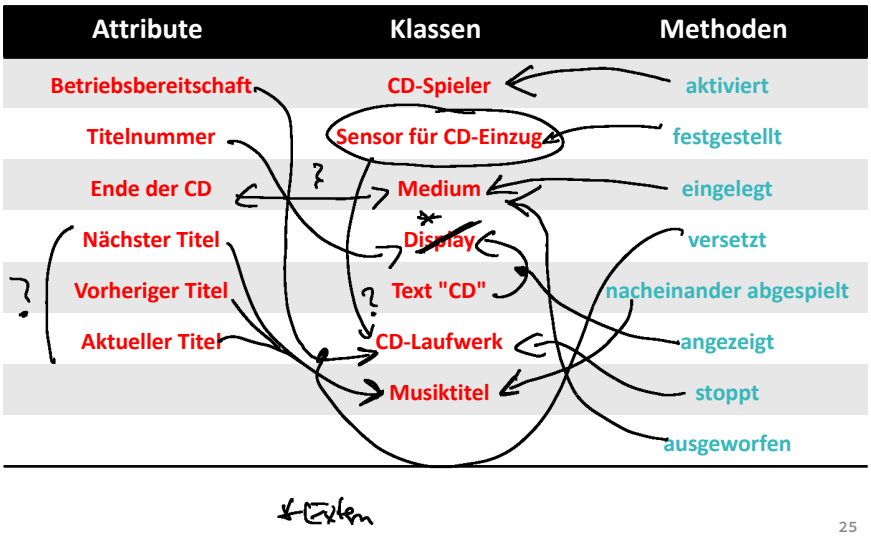
Ziel: Eine **Musik-CD wiedergeben**
Umfang: **CD-Spieler** im **Autoradio**
Übergeordneter Anwendungsfall: **Wiedergabe**
Vorbedingung: **Autoradio** ist **angeschaltet**
Nachbedingung: Der **Inhalt** der **CD** wird **wiedergegeben** (**Erfolg**), es erfolgt keine **Wiedergabe** (**Fehler**)
Primärer Akteur: **Benutzer**
Anstoßereignisse: **Benutzer drückt Taste „CD“** oder eine **CD** wird **eingelegt**

Erfolgsszenario:

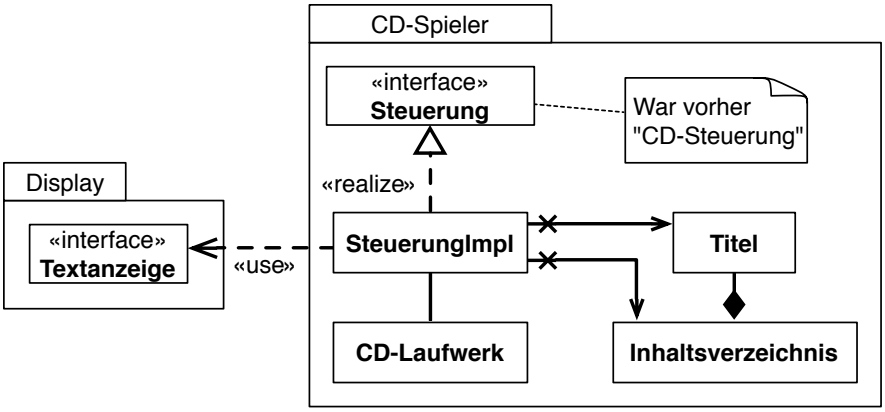
1. Der **CD-Spieler** wird **aktiviert**.
2. Anhand des **Sensors für den CD-Einzug** wird **festgestellt**, dass ein **Medium eingelegt** ist.
3. Auf dem **Display** wird der **Text „CD“ angezeigt**.
4. Das **CD-Laufwerk** wird in **Betriebsbereitschaft versetzt**.
5. Die **Musiktitel** werden **nacheinander abgespielt** und auf dem **Display** werden die jeweiligen **Titelnummern angezeigt**.
6. Am **Ende der CD stoppt** das **CD-Laufwerk** und es wird der **Text „CD“** auf dem **Display angezeigt**.

24

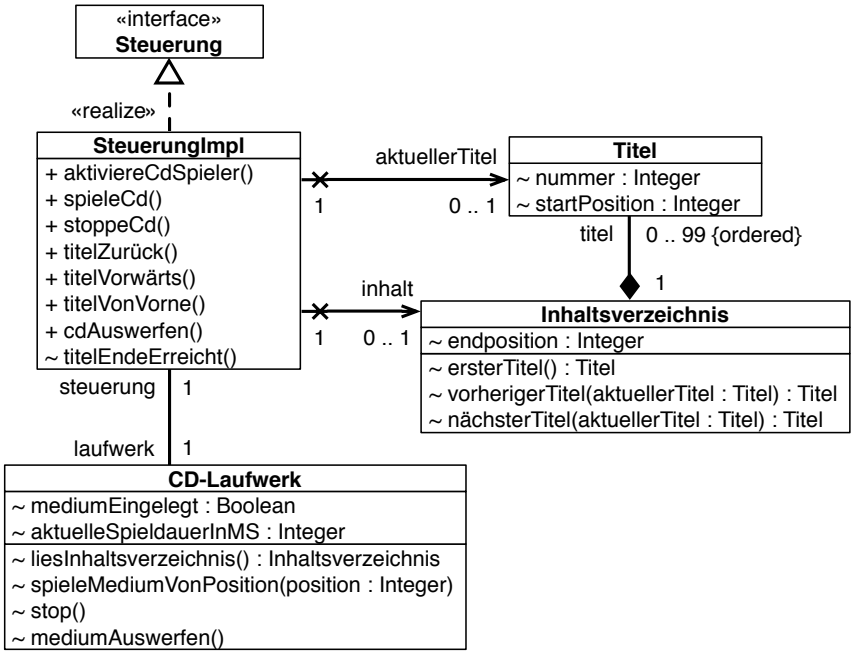
Identifizierte Klassenkandidaten



25



26



27

Klassendiagramme I

Ergebnissicherung

29

Zusammenfassung

- Klassendiagramme in der UML
- Notation von Klassen im Detail
  - Sichtbarkeit, Multiplizitäten, static, abstract
- Assoziationen
  - Navigierbarkeit, Enden, Gen/Spec
- Erstellung von UML Klassendiagrammen
  - Verb/Substantiv Methode