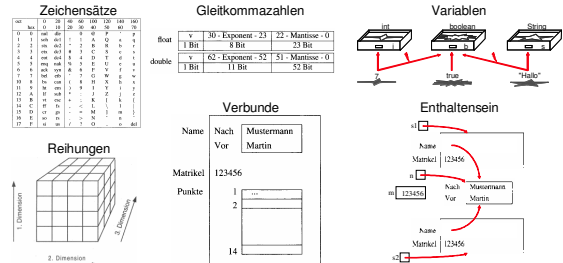


Objektorientierte Analyse und Entwurf

Thomas Röfer

Funktionale Dekomposition \leftrightarrow Objektorientierter Ansatz
 Objekte und Klassen
 Objektbeziehungen
 Beispiel

Rückblick „Datenorganisation und Datenstrukturen“



Funktionale Dekomposition

- ▶ **Schritte**
 - ▶ Abstrakte Spezifikation der Funktion
 - ▶ Hierarchisch absteigende Zerlegung der Funktion in immer elementare Teilfunktionen
 - ▶ Ausprogrammieren der Funktionen
- ▶ **Beispiel**
 - ▶ Verwaltung der PI-1 Studierenden
 - ▶ Einfügen, Ändern, Löschen
 - ▶ z.B. Einfügen: Leeren Datensatz erzeugen, ändern, in Datenbank einfügen
- ▶ **Nachteil**
 - ▶ Datenstrukturen sind nicht Teil des Modellierungsprozesses

Objektorientierter Ansatz der Software-Entwicklung

- ▶ **Analyse**
 - ▶ Suche nach Objekten und Objektbeziehungen in der realen Welt
 - ▶ Wie soll Software genutzt werden?
 - ▶ Welche Funktionalität soll wem zur Verfügung gestellt werden?
 - ▶ Was geschieht *warum* mit Objekten?
- ▶ **Entwurf**
 - ▶ Übertragung in die Welt der Software
 - ▶ Ergänzung und Modifikation aus programmiertechnischen Notwendigkeiten
 - ▶ Wie soll etwas *im Prinzip* geschehen?
 - ▶ Modell der Software-Architektur
- ▶ **Implementierung**
 - ▶ Software-Architektur wird zum lauffähigen Programm konkretisiert
 - ▶ Objektzustände werden durch Datenstrukturen repräsentiert
 - ▶ Objektfunktionalität wird durch Algorithmen realisiert und ausprogrammiert
 - ▶ Festlegung, *wie* alles *im Einzelnen* geschieht

Objekte

- ▶ **Objekt**
 - ▶ Gedankliche oder reale Einheit in der Umwelt oder Software
 - ▶ Ein Objekt hat einen *Zustand* und eine *Funktionalität*
- ▶ **Objekt-Funktionalität**
 - ▶ Nach innen: Veränderung des Objektzustandes
 - ▶ Nach außen: Wirkung auf andere Objekte, zu denen *Objektbeziehungen* bestehen
 - ▶ Wird realisiert durch die *Methoden* eines Objekts
 - ▶ Interaktion mit Objekt geschieht ausschließlich über dessen Methodenschnittstelle
- ▶ **Objekt-Zustand**
 - ▶ *Zustandsvariablen*, auch *Attribute* des Objekts
 - ▶ Attribute sollten nicht nach außen *sichtbar* sein (*Geheimnisprinzip, Kapselung*), sondern nur über Methoden (*Selektoren, set-/get-Methoden*) gesetzt oder gelesen werden können.

Name
Zustand
Funktionalität

Beispiele für Objekte

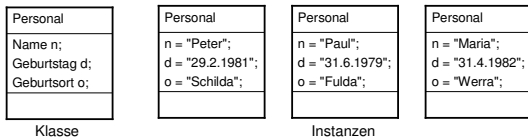
TV2000
EinAus z;
Seriennummer s;
Kanal k;
Lautstärke l;
ein();
aus();
wähle_Kanal();
wähle_Lautstärke();
get_Seriennummer();

Motor
EinAus z;
Vorwärtslauf v;
Rückwärtslauf r;
Drehzahl d;
ein();
aus();
vorwärts();
rückwärts();
set_Drehzahl();
get_Drehzahl();

Robot
EinAus z;
Greifposition p;
ein();
aus();
bewege_Punkt_zu_Punkt();
bewege_Linear();

Objekte und (Objekt-)Klassen

- Objekte \Leftrightarrow Klassen
- Objekte sind Einzelstücke
- Objekte mit gleichen Attributen und Methoden gehören derselben (Objekt-)Klasse an
- Eine Klasse ist der *Typ* eines Objekts
- Objekte sind *Instanzen* einer Klasse



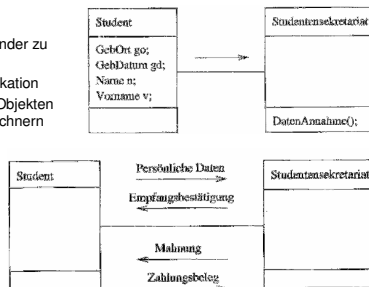
Objektbeziehungen

- Beziehungen zwischen Klassen**
 - Darstellung durch Verbindungslinie
 - Vielfachheit, z.B. 0..2, *, 1..*, wobei * für „beliebig viele“ steht
- Verhaltensbezogene Beziehungen**
 - Informationsfluss oder Nachrichten (*message*)
 - Funktionsaufruf oder Kunde/Lieferant (*client/server*)
- Strukturelle Beziehungen**
 - Einschluss (*has-a*)
 - Subtyp oder Vererbung (*is-a*)



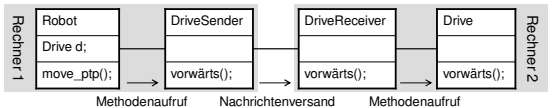
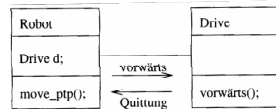
Informationsfluss oder Nachrichten

- Definition**
 - Nachricht wird von Sender zu Empfänger geschickt
 - Asynchrone* Kommunikation
 - Kann auch zwischen Objekten auf verschiedenen Rechnern geschehen
- Modellierung**
 - Interaktionsdiagramm



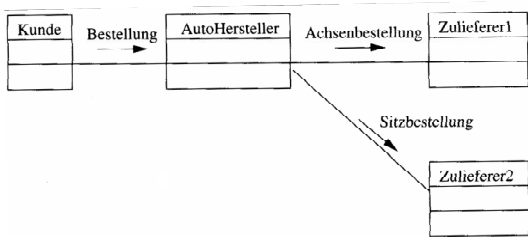
Client/Server-Beziehung

- Definition**
 - Ein Objekt (Client) fordert Dienstleistung von anderem Objekt (Server) an
 - Synchrone* Kommunikation
 - Methodenauf** (Standard in Java)
 - Kann nicht unmittelbar auf verschiedenen Rechnern geschehen
 - Ist aber durch Stellvertreterobjekte möglich (*entfernter Methodenauf*, *Remote Method Invocation*)



Client/Server-Beziehung

- Objekte können sowohl als Client, als auch als Server auftreten



Methodenauf

- Beispiel**
 - class Robot

```

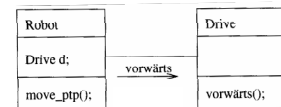
{
  Drive d;
  :
  void move_ptp()
  {
    d.vorwärts();
  }
}

```
 - class Drive

```

{
  void vorwärts()
  {
    :
  }
}

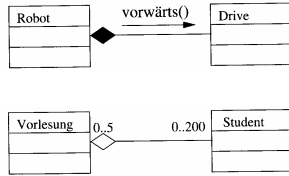
```



- Bitte wählt eure Bezeichner einheitlicher als die Autoren des Buches**
 - nur in einer Sprache (normalerweise Englisch)
 - forward()*;
 - keine Unterstriche (werden nur in Konstanten verwendet)
 - movePTP()*;
 - final int NUM_OF_TESTS = 13;*

Einschlussbeziehungen (has-a)

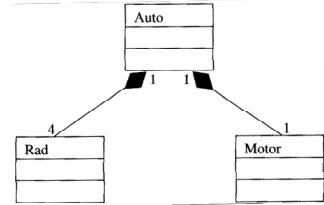
- Definition**
 - Objekte einer Klasse schließen Objekte einer anderen Klasse ein
 - Ein Objekt „hat“ ein anderes Objekt (*has-a*-Beziehung)
- Komposition**
 - Teilobjekt ist fester Bestandteil eines Objektes
 - Beispiel: Ein Roboter hat einen Antrieb
- Aggregation**
 - Ein Objekt ist zu einem anderen zugehörig, aber nicht ausschließlich
 - Beispiel: Eine Vorlesung hat bis zu 200 Studenten, die aber auch an anderen Vorlesungen teilnehmen



Einschlussbeziehungen (has-a)

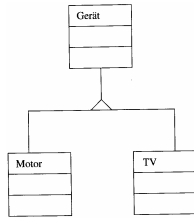
- Beispiel**
 - Ein Auto hat vier Räder und einen Motor
- In Java**

```
class Auto
{
    Rad vr, vl, hr, hl;
    Motor m;
    :
    void starte()
    {
        m.ein();
    }
}
```



Subtyp bzw. Vererbungsbez. (is-a)

- Definition**
 - Eine Klasse besitzt alle Eigenschaften einer anderen Klasse und darüber hinaus noch weitere
 - Der Subtyp (*subtype*) ist (*is-a*) eine spezielle Abart der Obertyps (*supertype*)
 - Der Subtyp ist eine *Spezialisierung* des Obertyps
 - Der Obertyp ist eine *Verallgemeinerung* des Subtyps
- Nutzung der Beziehung**
 - Subtyp *erbt* vom Obertyp, d.h. die gemeinsamen Eigenschaften müssen nur einmal im Obertyp implementiert werden



Subtyp bzw. Vererbungsbez. (is-a)

- Beispiel**
 - Kaffeemaschine Cafe2000 hat eine Funktion zur Befüllung von Kaffee und Wasser von bis zu 12 Tassen
 - Kaffeemaschine Cafe2000LT hat zusätzlich eine Timerfunktion
- In Java**

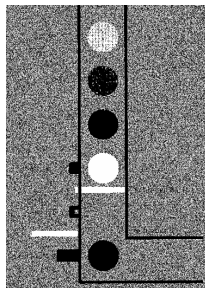
```
class Cafe2000
{
    int tasse;
    befüllung(int);
}

class Cafe2000LT extends Cafe2000
{
    Time t;
    void timer(Time time) {t = time;}
}
```



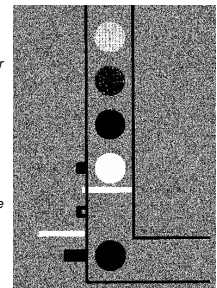
Objektorientierte Analyse und Entwurf

- Situationsbeschreibung**
 - Die Vereinzelungseinheit besteht aus einer senkrechten Röhre, zwei Schiebern A und B, zwei Sensoren C und D, sowie einem Druckluftventil V.
 - In der Röhre können Bälle gespeichert werden, die von oben zugeführt werden und die auf Anforderung unten einzeln aus der Röhre geblasen werden sollen.
 - Die Schieber können die Stellungen „geöffnet“ oder „geschlossen“ einnehmen.
 - Ist ein Schieber geschlossen, so ist die Röhre geschlossen.
 - Die Sensoren melden, ob sich an der entsprechenden Stelle ein Ball befindet; falls ja, sind sie aktiviert, falls nein, sind sie deaktiviert.
 - Am unteren Ende der Röhre befindet sich ein Druckluftventil, das geöffnet oder geschlossen werden kann und das nicht ständig geöffnet bleiben sollte.



Objektorientierte Analyse und Entwurf

- Nutzungsszenarien**
 - Standardoperation**
 - Die Bälle werden von oben zugeführt und von der Röhre gespeichert.
 - Auf Anforderung wird genau ein Ball freigegeben und aus dem Ausgabeschacht geblasen.
 - Wartung/Fehlerbeseitigung**
 - Der Wartungstechniker aktiviert einen Wartungszyklus, in dem alle Operationen der beteiligten Geräte einmal ausgeführt werden.
 - Stellt er eine Fehlfunktion fest, so lässt er sich die Seriennummer des betreffenden Gerätes ausgeben und tauscht es aus.



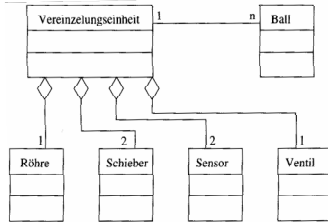
Objektorientierte Analyse und Entwurf

Mögliche Objekte

- › Vereinzelungseinheit, Schieber, Sensor, Ventil, Ball (?), Röhre (?)

Beziehungen

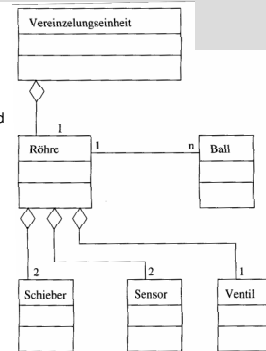
- › Die Vereinzelungseinheit enthält Schieber, Sensoren, ein Ventil und eine Röhre.
- › Bälle sind kein integraler Bestandteil, aber es besteht in jedem Fall eine (eventuell temporäre) Beziehung.



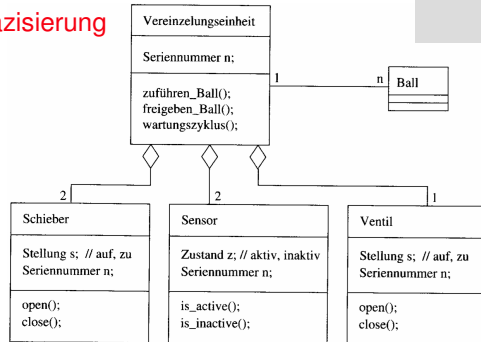
Alternative

Aber:

- › Gewöhnliche Röhren enthalten normalerweise keine Schieber und Sensoren.
- › Daher sollte man eher von einer Vereinzelungsröhre sprechen.
- › Damit gibt es aber keinen Unterschied zwischen Vereinzelungsröhre und -einheit, denn die Röhre hat keine von der Einheit getrennt Funktionalität.



Präzisierung



Mit Vererbung

