



BEUTH HOCHSCHULE FÜR TECHNIK BERLIN  
University of Applied Sciences

# Software Engineering 1

## Modellierung von Objektstrukturen

### Lernziele

Schnittstellen mit der UML modellieren können

Multiplizitäten und Sichtbarkeiten korrekt notieren können

Begriff der Signatur erklären können

Abstrakte Datentypen und Objektorientierung erklären können

Unterschied zwischen Klassen und Objekten verstehen

Unterschied zwischen Signatur- und Implementierungsvererbung kennen

3

## Warum Objektorientierung?

- Antwort auf die Software-Krise:
  - Software wird exponentiell größer & komplexer
- „Teile und Herrsche“
  - Essentielle Komplexität verstecken
  - Identifizierung von Kernkonzepten („Klassen“)
    - Fassen (Daten + Funktionen) zusammen
  - Klassen verstecken Implementierung
    - Aufruf nur über klare Schnittstellen

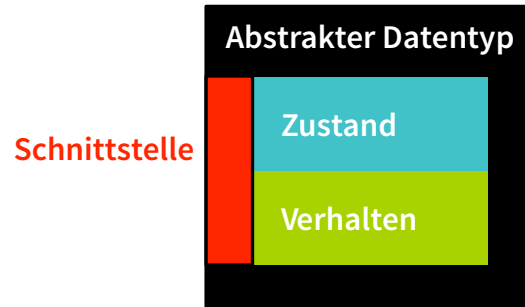
5

## TIOBE Index

### Beliebteste Programmiersprachen Weltweit

4/20	4/19	Trend	Sprache	Anteil
1	1		Java	17%
2	2		C	17%
3	4		Python	9%
4	3		C++	7%
5	6		C#	5%
6	5		Visual Basic	5%
7	7		Javascript	2%
8	9		PHP	2%
9	8		SQL	2%
10	16		R	2%

6



- Verbund von:
  - **Zustand** (strukturierte Daten)
  - **Verhalten** (zulässige Operationen)
- Zugriff nur über **Schnittstelle**
  - Trennt Signatur & Semantik von Implementierung
- Abstrakte Datentypen (ADT) werden instanziiert

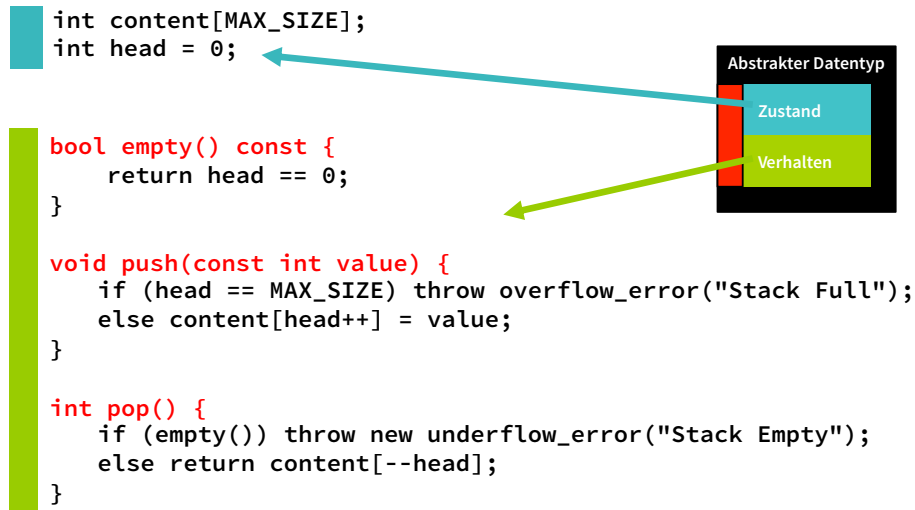
7

## Beispiel für eine Schnittstelle (C++)

```
class MyStack {  
    public:  
        virtual ~MyStack() {}  
        virtual bool empty() const = 0;  
        virtual void push(const int value) = 0;  
        virtual int pop() = 0;  
};
```

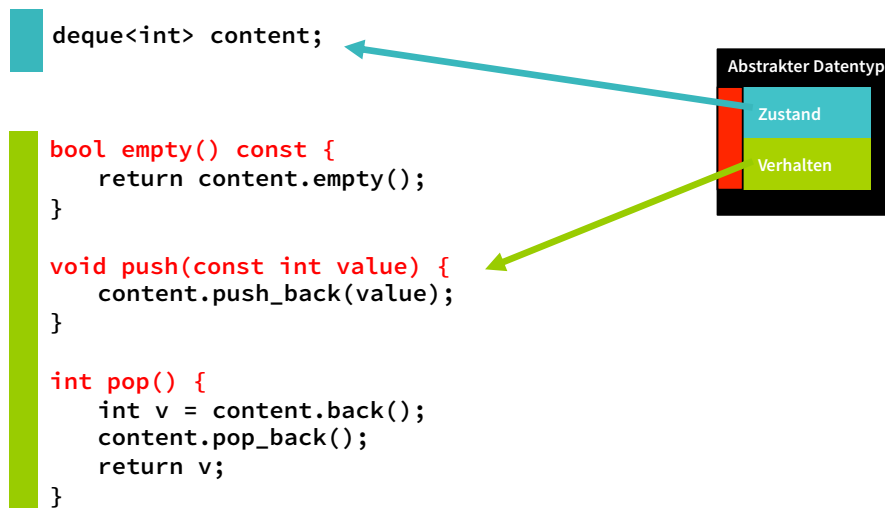
8

## Stack als Feld-Implementierung



9

## Stack als Deque-Implementierung



10

## Eigenschaften von ADT

<b>Universell</b>	<ul style="list-style-type: none"><li>• Kann in anderen Programmen verwendet werden</li></ul>
<b>Präzise Spezifiziert</b>	<ul style="list-style-type: none"><li>• <u>Signatur</u>: Eindeutige, vollständige Schnittstelle zwischen ADT und Anwendung</li></ul>
<b>Einfach</b>	<ul style="list-style-type: none"><li>• Übernimmt Repräsentation und Speicherverwaltung, verbirgt Komplexität</li></ul>
<b>Gekapselt</b>	<ul style="list-style-type: none"><li>• Der Anwender kann in die Implementierung und Datenstruktur nicht eingreifen</li></ul>
<b>Modular</b>	<ul style="list-style-type: none"><li>• Ermöglicht Austausch von Programmteilen</li><li>• Begrenzt den Umfang von Änderungen</li></ul>

11

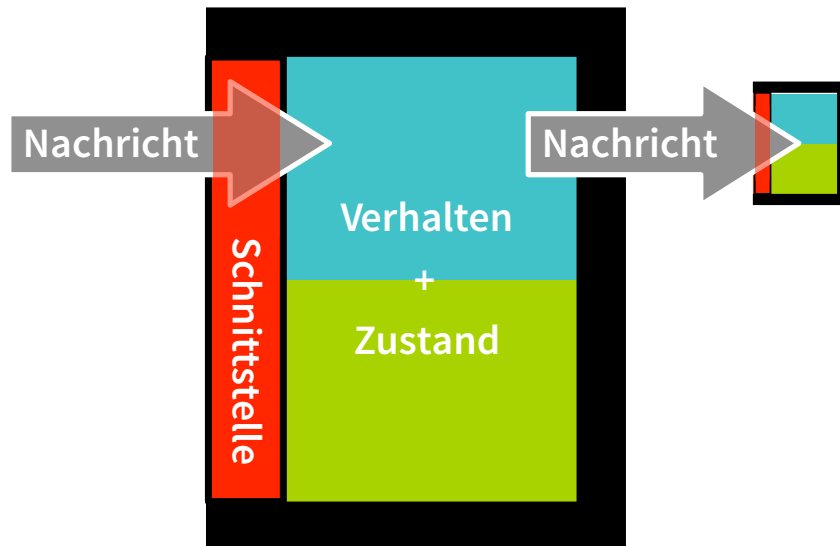
## Objektorientierung

„Systeme bestehen aus kommunizierenden Objekten“

- **Objekte**
  - Klar umrissene Konzepte einer Domäne
  - Senden und empfangen Nachrichten
    - Führt zum Methodenaufruf
- **Klassen**
  - Mengen gleichartiger Objekte
  - Sind abstrakte Datentypen
  - Haben Attribute (Zustand) und Methoden (Verhalten)

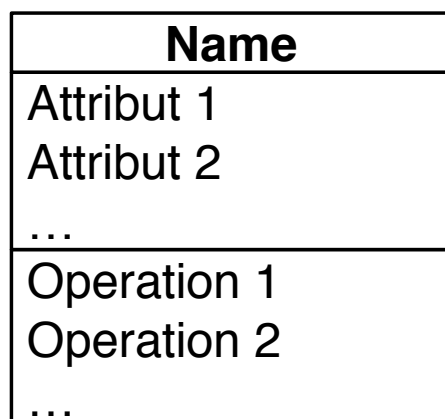
Grundlegende Konzepte aus Simula (1967), Ole-Johan Dahl, Kristen Nygaard

12



13

## Notation von Klassen



*Optional*

*Optional*

15

## Beispiel C++

```
class Person {

private:
    string nachname, vorname;
    Geschlecht geschlecht;

public:
    void setNamen(const string vorname, const string nachname);
    string getNachnamen() const;
    string getVornamen() const;
    void setGeschlecht(const Geschlecht geschlecht);
    Geschlecht getGeschlecht() const;
};
```

Person
- nachname : String
- vorname : String
- geschlecht : Geschlecht
+ setNamen(vorname : String, nachname : String)
+ getNachnamen() : String
+ getVornamen() : String
+ setGeschlecht(geschlecht : Geschlecht)
+ getGeschlecht() : Geschlecht

16

## Schnittstellen & Signaturen

### Schnittstellen

- Ermöglichen und regeln Nachrichtenaustausch
- Definieren die formale Signatur einer Klasse

### Klassensignatur

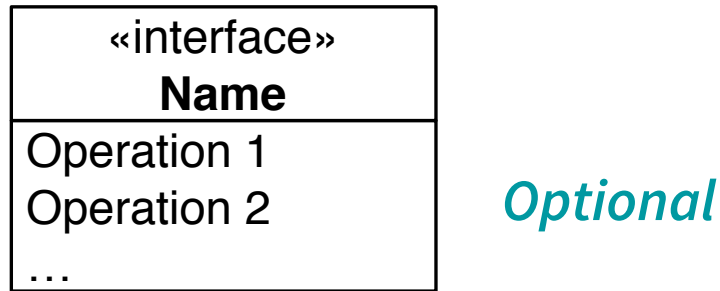
- Besteht aus allen Methodensignaturen einer Klasse

### Methodensignatur

- Sichtbarkeit, Name der Methode, Reihenfolge und Typen der Argumente, Rückgabotyp

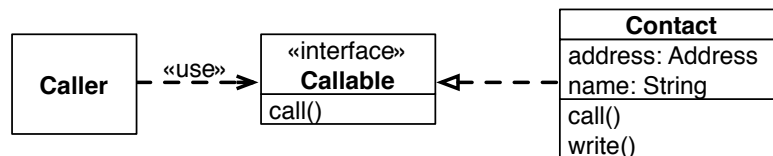
17

## Notation von Schnittstellen



18

## Notation von Schnittstellen



- Verwendung des Schlüsselworts «interface»
  - Nur Operationen definiert, keine Attribute
- Assoziation: Realisierung
  - Gestrichelter Pfeil mit ungefüllter Spitze an der Schnittstellenseite
- Assoziation: Verwendung
  - Gestrichelter Pfeil mit offener Spitze an der Schnittstellenseite
  - «use» Schlüsselwort

19



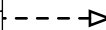
## Beispiel C++

«interface» Identität
+ setNamen(vorname : String, nachname : String)
+ getNachnamen() : String
+ getVornamen() : String
+ setGeschlecht(geschlecht : Geschlecht)
+ getGeschlecht() : Geschlecht

```
class Identitaet {
public:
    virtual ~Identitaet() = 0;
    virtual void setNamen(const string vorname,
                          const string nachname) = 0;
    virtual string getNachnamen() const = 0;
    virtual string getVornamen() const = 0;
    virtual void setGeschlecht(const Geschlecht
                              geschlecht) = 0;
    virtual Geschlecht getGeschlecht() const = 0;
};
```

20

Person
- nachname : String
- vorname : String
- geschlecht : Geschlecht
+ setNamen(vorname : String, nachname : String)
+ getNachnamen() : String
+ getVornamen() : String
+ setGeschlecht(geschlecht : Geschlecht)
+ getGeschlecht() : Geschlecht



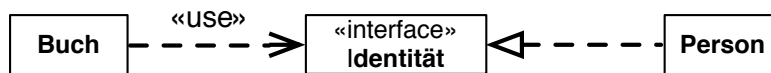
«interface» Identität
--------------------------

## C++ Beispiel

```
class Person : public Identitaet {
Public:
    void setNamen(const string vorname,
                  const string nachname) override;
    string getNachnamen() const override;
    string getVornamen() const override;
    void setGeschlecht(const Geschlecht
                      geschlecht) override;
    Geschlecht getGeschlecht() const override;
};
```

21

## Beispiel C++



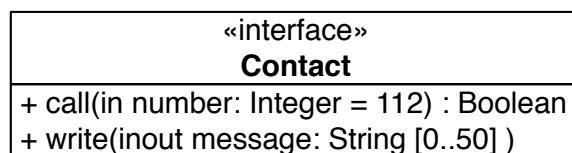
```

class Buch {
private:
    Identitaet *autor;

    ...
}
  
```

22

## Details der Notation



### Operationen





[Sichtbarkeit] **name** ([*Parameter*]) [: Rückgabety] [[Multiplizität]]

### Parameter (Mehrere Parameter werden durch Kommata getrennt)

[Übergaberichtung] **name** : Typ [[Multiplizität]] [= Vorgabewert]

23

## Einschränkung der Sichtbarkeit

	<ul style="list-style-type: none"><li>• <b><u>Öffentlich</u></b></li><li>• Sichtbar für alle Ausprägungen</li></ul>
	<ul style="list-style-type: none"><li>• <b><u>Privat</u></b></li><li>• Nur für Ausprägungen der eigenen Klasse sichtbar</li></ul>
	<ul style="list-style-type: none"><li>• <b><u>Geschützt</u></b></li><li>• Für Ausprägungen und Spezialisierungen der eigenen Klasse sichtbar</li></ul>
	<ul style="list-style-type: none"><li>• <b><u>Paket</u></b></li><li>• erlaubt den Zugriff für alle Elemente innerhalb des eigenen Pakets</li></ul>

24

## Multiplizitäten in der UML

- Zusammenhängende Bereiche aus den nichtnegativen ganzen Zahlen
- Geben Menge von Elementen an
- Durch Bereichsgrenzen beschrieben
  - **von .. bis**
  - Beide Grenzen sind inklusive
- Es ist möglich die untere Grenze wegzulassen
  - Verzicht auf Bereichssymbol „..“
  - Die untere Grenze entspricht dann der oberen

25

## Beispiele für Multiplizitäten

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

26

## Generalisierung / Spezialisierung

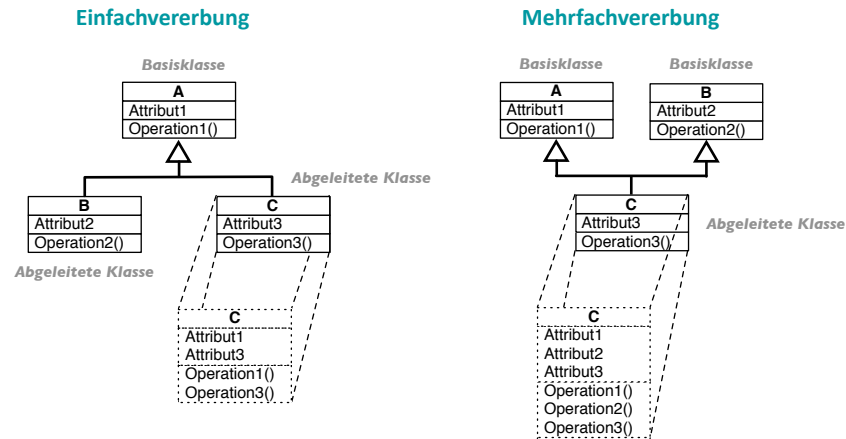


- Binäre Beziehung zwischen zwei UML Typen
  - Ein speziellerer Typ (hier B)
  - Ein generellerer Typ (hier A)
- Notation: Pfeil mit großer, ungefüllter Spitze
  - An der Seite des generelleren Typs
- Der speziellere Typ verfügt dadurch über alle Struktur- und Verhaltensmerkmale des generelleren Typen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

32

# Vererbung

Abgeleitete Klassen übernehmen  
Attribute und Methoden der Basisklassen

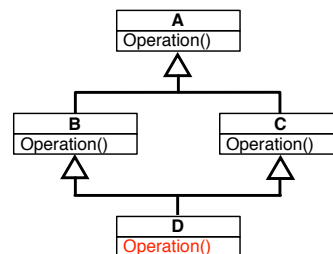


33

## Prinzipien der Vererbung

- Schnittstellenvererbung
  - Nur Signaturen werden übernommen
- Implementierungsvererbung
  - Signaturen und Implementierung werden übernommen
- Zeitpunkt der Vererbung
  - Übersetzungszeit
  - Laufzeit (Cloning)

**Das Diamanten Problem**  
Bei mehrfacher  
Implementierungsvererbung



34

## Links & Literatur



H. Balzert, „Lehrbuch der Objektmodellierung“, 2. Auflage, Spektrum Akademischer Verlag, 2005.  
ISBN: 978-3827402851

36

## Zusammenfassung

- Stellenwert von OO
- Abstrakte Datentypen (ADT)
  - Zustand, Verhalten, Schnittstellen
- OO: Systeme kommunizierender Objekte
  - Klasse und Objekt, Nachrichtenaustausch
- UML Notation von Klassen
  - Schnittstellen, Signaturen, Sichtbarkeiten, Multiplizitäten
- Vererbung
  - Schnittstellen- und Implementierungsvererbung
  - Diamantenproblem der Mehrfachvererbung

38