

Softwaretechnik

## Kapitel 5

1. Wieso Software Engineering?
2. Vom Einzelkämpfer zum Großprojekt
- Systematische Softwareentwicklung**
3. Anforderungen und Test: Basis des Projekts
4. Entwurf: Strukturen und nicht-funktionale Eigenschaften
- 5. Entwürfe notieren mit UML: Modelle im SE**
6. Design Patterns: Entwurfserfahrungen nutzen
7. Management: Technik und Projektmanagement

Leibniz Universität Hannover    SWT 2015/16    175

Softwaretechnik

## Inhalt von Kapitel 5

**Systematische Softwareentwicklung**

**5. Entwürfe notieren in UML – Modelle im Software Engineering**

- Historie der UML
- Objektorientierung: Konzepte und Denkweise
- Klassendiagramme
- Sequenz- und Kollaborationsdiagramme
- Information Hiding

Leibniz Universität Hannover    SWT 2015/16    176

Softwaretechnik

## Wozu grafische Notation (2D)?

- Grafische Notation := 2-dimensionale Syntax + Semantik + Pragmatik
- Modelltheorie
  - Irrelevante Details ausblenden
  - Relevante Eigenschaften hervorheben
  - Abundante Eigenschaften günstig gestalten
    - Leicht zu verstehen
    - Leicht zu ändern
    - Menschlicher Kognition entgegenkommen: Patterns, 2D >> 1D (Text)
- Spektrum von Formalitätsgraden
  - Mehr oder weniger viele relevante Eigenschaften
  - Anfangs wenige, Grobstruktur schnell erstellen
  - Schritt für Schritt mehr, Detail ausfeilen, näher zum Code

Kurt Schneider    Leibniz Universität Hannover    SWT 2015/16    177

Softwaretechnik

## Was ist UML?

### Unified Modeling Language

**Sammlung vorwiegend grafischer Sprachen [mehrere!]**

- zur Erstellung von Anforderungs- und Entwurfsmodellen
- aus verschiedenen Perspektiven

**UML-Spezifikation oder UML-Modell =<sup>Def</sup> Sammlung von UML-Diagrammen, die sich inhaltlich ergänzen und überlappen**

- Klassenmodell spezifiziert strukturellen Aufbau des Systems
  - Bei Anforderungen zeigt es folgende relevante Originalaspekte:
    - Gegenstände der Realität, mit denen das System umgehen muss
  - Bei Entwurf und Implementierung zeigt es:
    - Klassen der Lösung

Kurt Schneider    Leibniz Universität Hannover    SWT 2015/16    178

Softwaretechnik

## Klassen und Objekte: Eine Denkweise

### Genauer hingesehen

**Eine Klasse** ist die gemeinsame Beschreibung

- des Verhaltens und
- der Attribute

aller Objekte dieser Klasse

**Klasse beschreibt:**

- Verhaltensmuster
- Attribute pro Objekt
- Bezeichner

**Objekte**

- sind verschiedene Individuen
- haben je eigenen Zustand
- zeigen Verhalten: sie versenden Nachrichten und reagieren auf Nachrichten anderer Objekte

**Jedes Objekt hat**

- Dieselben Bezeichner
- Evtl. verschiedene Zustände
- Evtl. verschiedene Attributwerte

Folglich evtl. verschiedene Situationen im selben Verhaltensmuster

Kurt Schneider    Leibniz Universität Hannover    SWT 2015/16    179

Softwaretechnik

## UML Diagramme

**Use Case Diagram**  
Benutzersicht: Interaktion von Benutzern mit dem System

**Aktivitätsdiagramm**  
Aktivitätssicht: Gemeinsame Bearbeitung einer Aufgabe

**Interaktionssicht**  

**Sequenzdiagramm**  
Zusammenspiel weniger Objekte

**Kommunikationsdiagramm**  
Zusammenspiel vieler Objekte

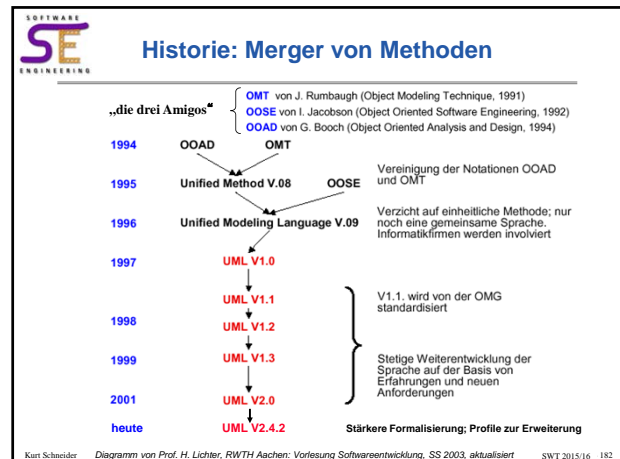
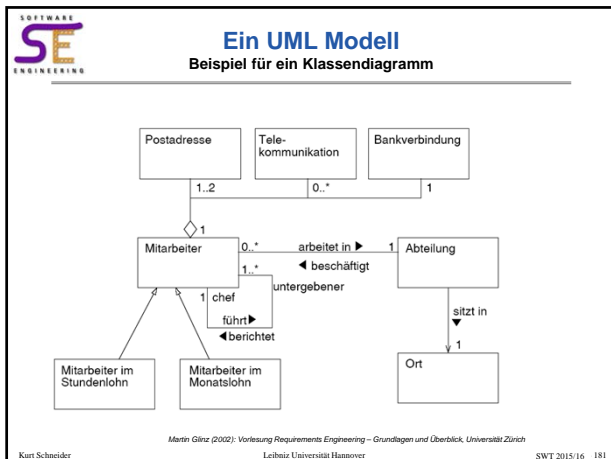
**Zustandsdiagramm**  
Verhaltenssicht: Innere Zustände und Verhalten der Objekte einer Klasse

**Klassendiagramm**  
Statische Sicht: Beteiligte Klassen und Strukturen

**Paketdiagramm** Gliederungssicht: Aufteilung in Pakete und Substrukturen

**Verteilungsdiagramm** Physische Sicht: Physische Systemstruktur und Verteilung

Kurt Schneider    Leibniz Universität Hannover    SWT 2015/16    180



### UML-Diagramme bearbeiten

- Mein Tipp:
  - Malen Sie erst von Hand
  - Danach holen Sie sich ein Werkzeug, zum Beispiel:
    - StarUML (open source, kostenlos)
      - Download: <http://sourceforge.net/projects/staruml/> (letzter Zugriff: 6.12.2014)
      - StarUML ist nicht hypermodern, aber leicht zu handhaben
      - Viele andere Werkzeuge sind für unsere Zwecke weniger geeignet
    - Auch gut: DIA Diagramm Editor
      - open source: <http://dia-installer.de/> (letzter Zugriff: 6.12.2014)
- Wozu Sie StarUML oder DIA verwenden sollten
  - Nach einer Handskizze: Diagramme sauber abzeichnen
  - Speichern, Datei im Team verteilen und verwenden
  - Ändern und aktualisieren
- Generieren, C# usw.?
  - Es geht noch viel mehr mit DIA, StarUML und ähnlichen Werkzeugen
  - Das brauchen Sie in SWT noch nicht
- Welche UML-Variante gilt für die Klausur? Siehe: UML-Poster V2 von SE

Kurt Schneider Leibniz Universität Hannover SWT 2015/16 183

### Klassen

#### Basisvarianten

**Bedeutung**

**Klassenname**

Keine Attribute def.

Operationen auch nicht

**Attributnamen**

**Operationsnamen**

**Verwendungshinweise**

Kann Stück für Stück vervollständigt werden

Müssen nicht Java-konform sein

() ist Konvention für „Operation“, nicht nur in Java

Dies sind die Minimalangaben

Oft zuerst die Operationen festlegen; Attribute zweitrangig

Kurt Schneider Hannover SWT 2015/16 184

### Klassen

#### fortgeschrittene Angaben

**Bedeutung**

**Klassenname**

**Attribute mit Typen**

**Sichtbarkeit**

+ public

- private

~ package

# protected

**Operationen**

**Sichtbarkeit**

**Parameter**

mit und ohne Typ

**Verwendungshinweise**

Kann Stück für Stück vervollständigt werden

Man kann Sichtbarkeit auch zunächst „offen lassen“

Ebenso: Parametertypen

Viele Tools dulden das nicht: alle oder keine Sichtbarkeiten

Möglichst wenige „Getter und Setter“

Nicht für alle Attribute

Evtl. im Modell ausblenden

Hintergrundfarbe weiß

Farbe, Schatten, 3D vermeiden!

Kurt Schneider Leibniz Universität Hannover SWT 2015/16 185

### Klassen

#### und entsprechender Code

**UML-Modell**

**Java**

```

class Konto {
    public int kontoNr;
    private EURO kontostand;
    private EURO kreditRahmen;
    public SecretPINType verschueseltePIN;

    public int erzeugeKontoNr() { }
    public SecretPINType verschueseltePIN(int pin) { }

    public EURO getKontoStand() { }
    public void einzahlen(EURO betrag) { }
    public EURO abheben(EURO betrag) { }

    public void ueberweisen(EURO betrag, Konto zielkonto) { }
    public boolean istPINKorrekt(Object pin){}

    public void setInhaber(String name) { }
    public String getInhaber() { }
    public void setKreditRahmen(EURO kreditObergrenze) { }
    public EURO getKreditRahmen() { }
}
    
```

**C++**

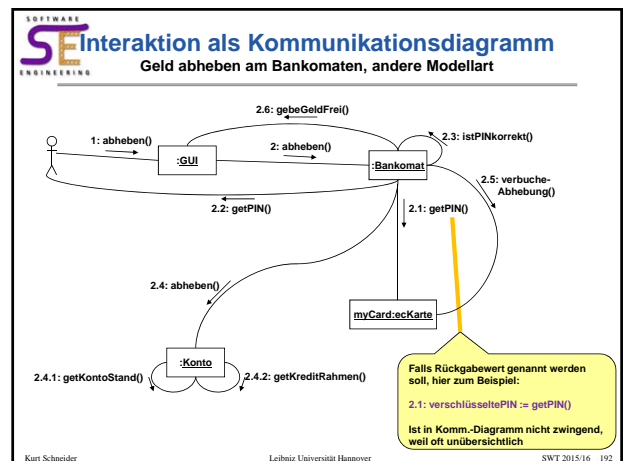
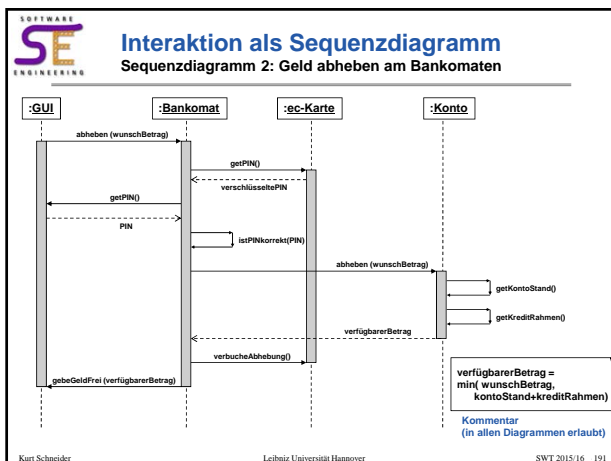
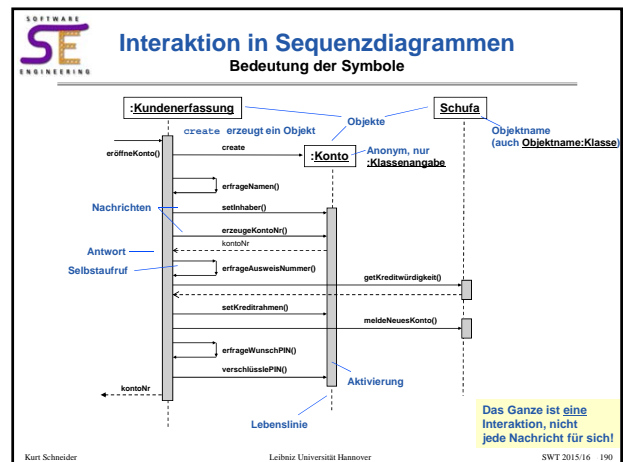
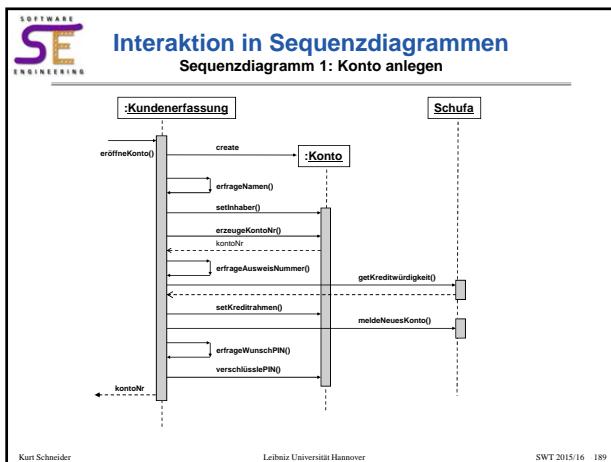
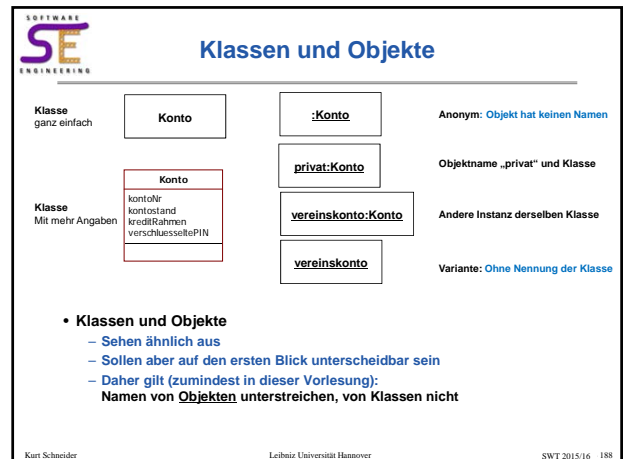
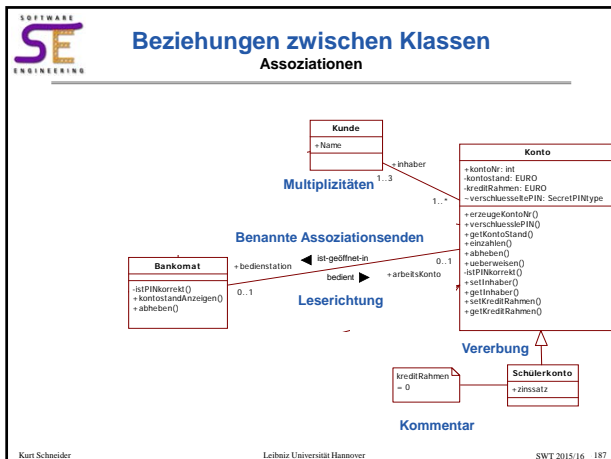
```

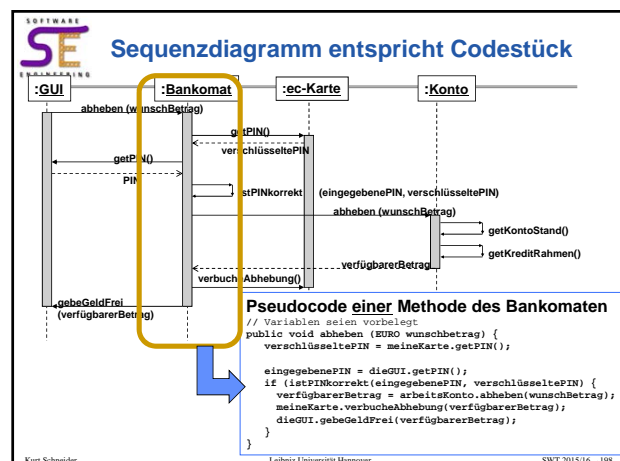
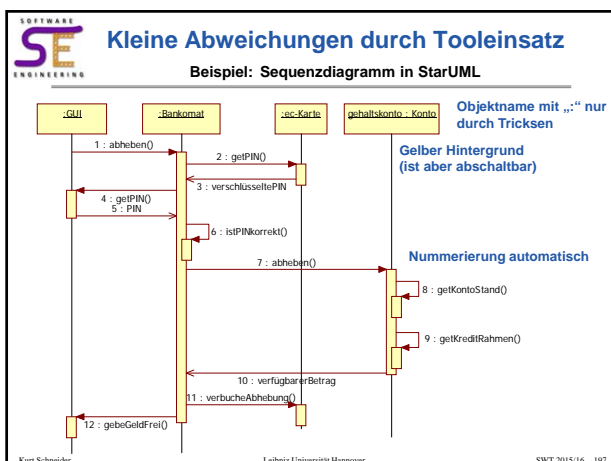
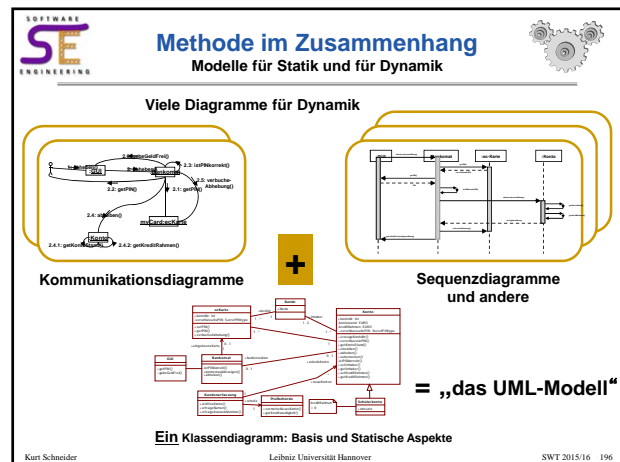
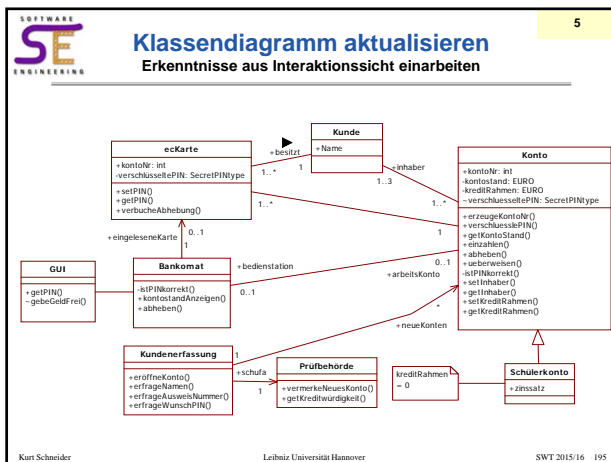
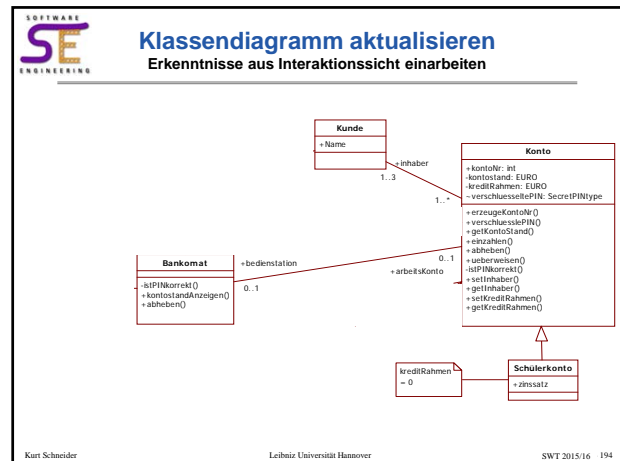
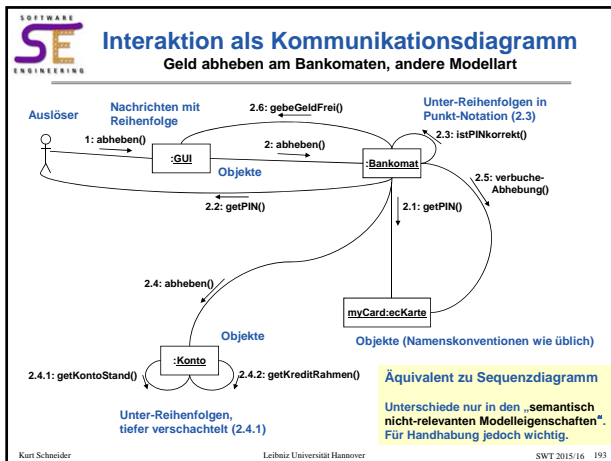
#define _KONTO_H
#include "Kunde.h"

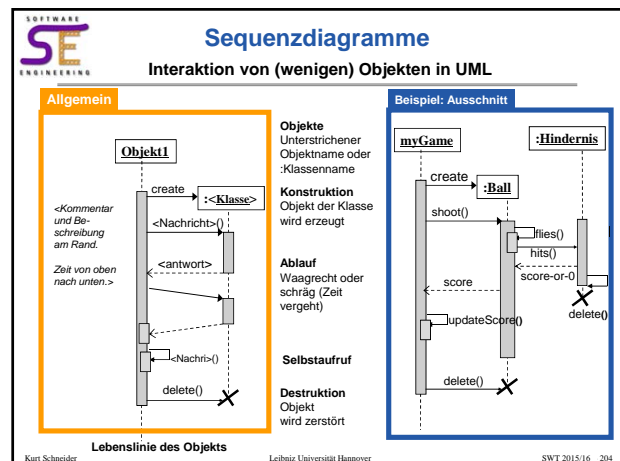
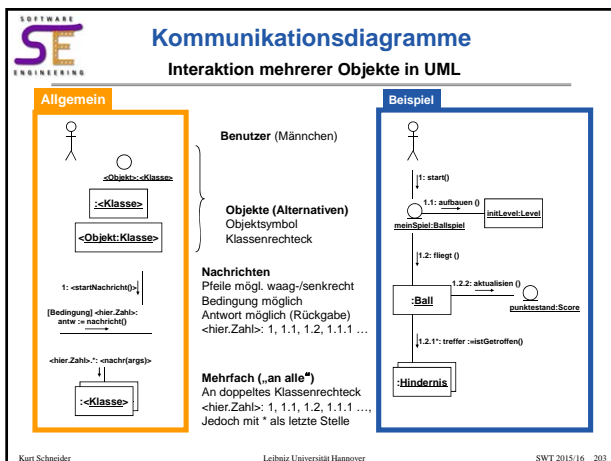
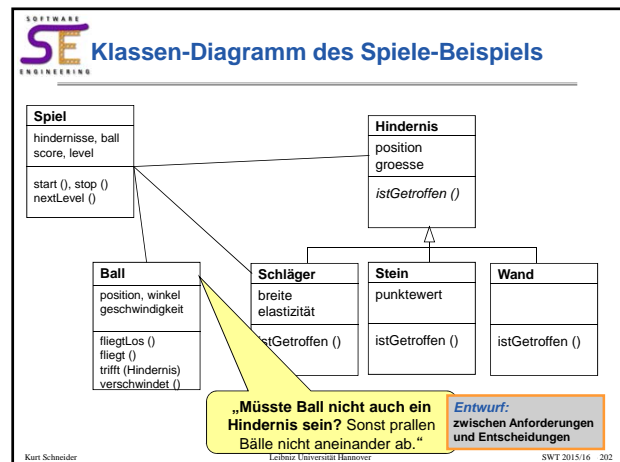
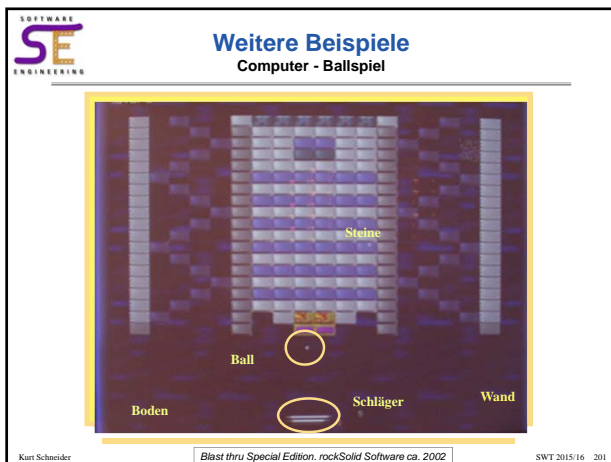
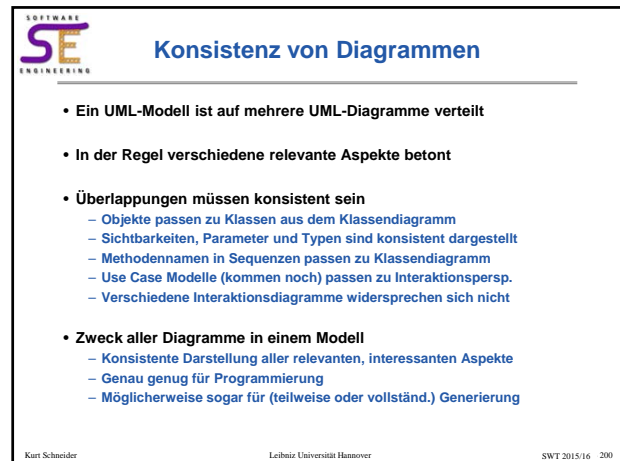
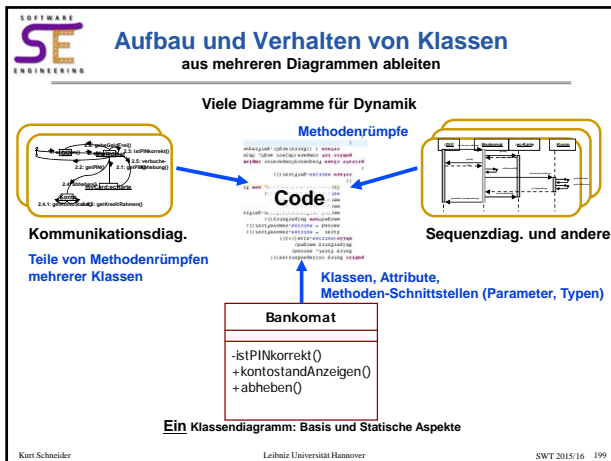
class Konto {
public:
    int kontoNr;
    int erzeugeKontoNr();
    SecretPINType verschueseltePIN(int pin);
    EURO getKontoStand();
    void einzahlen(EURO betrag);
    void ueberweisen(EURO betrag, Konto zielkonto);
    void setInhaber(String name);
    String getInhaber();
    void setKreditRahmen(EURO kreditObergrenze);
    EURO getKreditRahmen();

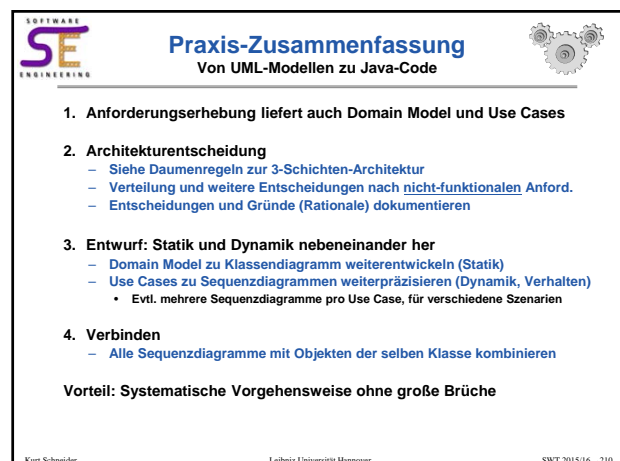
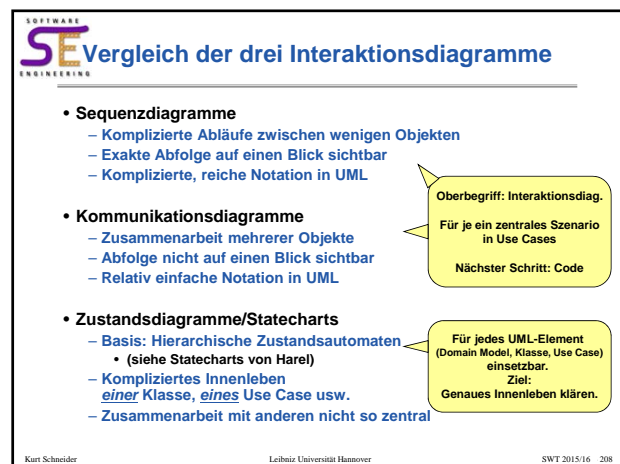
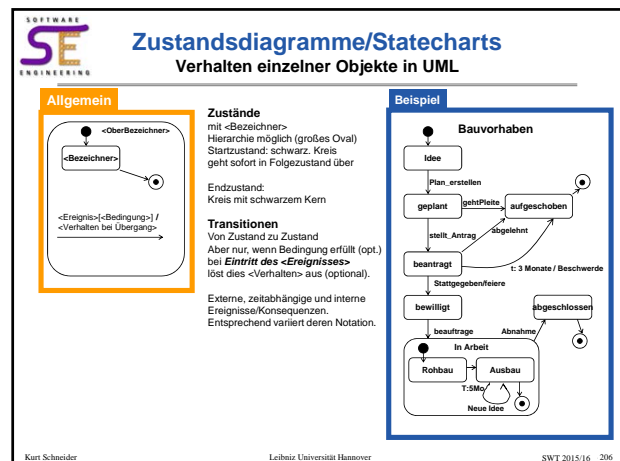
private:
    EURO kontostand;
    EURO kreditRahmen;
    boolean istPINKorrekt(int pin);
};
    
```

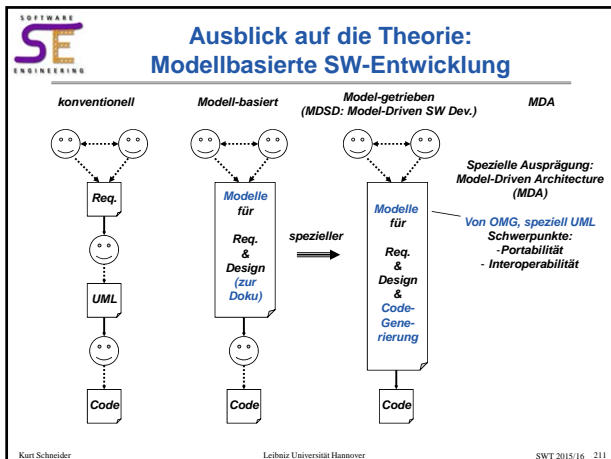
Kurt Schneider Leibniz Universität Hannover SWT 2015/16 186











## Softwaretechnik

### Inhalt von Kapitel 5

#### Systematische Softwareentwicklung

#### 5. Entwürfe notieren in UML – Modelle im Software Engineering

- Historie der UML
- Objektorientierung: Konzepte und Denkweise
- Klassendiagramme
- Sequenz- und Kollaborationsdiagramme
- Information Hiding

Kurt Schneider | Leibniz Universität Hannover | SWT 2015/16 | 212

### Sie arbeiten im Team

Stellen Sie sich vor ...

- Sie haben den Huffman-Code programmiert.
- Nun soll ihn das ganze Team benutzen.
- Was geben Sie den Kollegen, damit sie das tun können?

Kurt Schneider | Leibniz Universität Hannover | SWT 2015/16 | 213

### Was müssen die anderen wissen?

Beispiel Huffman-Code-Programm

The diagram shows a team of four people. One person is holding a codebook. The codebook contains the following information:

Charakter	Binärcode
A	000010
B	000011
C	000000
D	0001
E	0100
F	0001

Kurt Schneider | Leibniz Universität Hannover | SWT 2015/16 | 214

### Perspektivenwechsel

Was möchten Sie in dieser Situation wissen (müssen)?

„Oh, es gibt jetzt einen effizienteren Algorithmus! Verwenden Sie doch den! Wir haben ein Modul gekauft. Es ist aber sehr kompliziert; 12 Seiten Mathe – wenn Sie das nicht verstanden haben, können Sie das Programm kaum richtig einsetzen. Aber wir haben auch die ganze Entwicklerdoku hier – drei Ordner! Super, oder?“

**Wollen Sie das?**

Oder eher das:

```

BetterCode
+addChar(char, int)
+printCodeBook()
    
```

Kurt Schneider | Leibniz Universität Hannover | SWT 2015/16 | 215

### Also: Was müssen die anderen wissen?

Beispiel Huffman-Code-Programm

The diagram shows a team of four people. One person is holding a codebook. The codebook contains the following information:

Charakter	Binärcode
A	000010
B	000011
C	000000
D	0001
E	0100
F	0001

**Das meiste verstecken wir! Zum Glück.**

**Method Summary**

```

void addChar(char newChar, int frequency)
    Include a new character.

void printCodeBook()
    Derive Huffman code and print codebook for it (to standard out)
    
```

Kurt Schneider | Leibniz Universität Hannover | SWT 2015/16 | 216



## Entwurfssprinzip „Information Hiding“

- Von David Parnas, 1972
- Absolute Grundlage des SE!
  - Taucht immer wieder auf
  - Von Implementierung bis Doku.
- Idee: Jede Einheit gibt nur das Nötigste preis, um sie zu nutzen
  - Das ist die Schnittstelle mit Dokumentation
  - Wie sie funktioniert, ist dagegen versteckt (hidden)
    - Ändert sich etwas Internes, merkt man es außen nicht
    - Da vor ihm/ihre versteckt, braucht Nutzer keine Selbstdisziplin
    - „Jede Einheit versteckt (min.) eine Entwurfsentscheidung“ Parnas





Kurt Schneider      Leibniz Universität Hannover      SWT 2015/16    217

## Beispiel/Szenario Vereinfachtes Telefonbuch

Beispiel Interfaces

- A schreibt Customer Management System (CMS)
  - Will darin ein vereinfachtes Telefonbuch nutzen
  - Beauftragt B
- B macht sich Anforderungen klar
  - (Name, Telefonnummer)-Einträge verwalten
  - Suche nach richtigem Eintrag unterstützen
  - Löschen, Initialisieren
- Alles Kompliziertere lassen wir in diesem Beispiel weg!
  - Namen seien eindeutig, null werde abgefangen usw.
- Ziel: Sinn von Information Hiding mit Interfaces zeigen

Kurt Schneider      Leibniz Universität Hannover      SWT 2015/16    218

## Erster Versuch: B verwendet Array

Beispiel Interfaces

Einfach  
Schnell fertig

```

PhoneTable
PhoneTable(int size)
setEntry(int id, String name, String number)
getNumber(String name): String
getID(String name): int
delete(id)
  
```

CMS verwendet PhoneTable

```

public void organize() {
    PhoneTable ph = new PhoneTable(7);
    ...
    ph.setEntry(3, "May", "089/4567");
    nr = ph.getNumber("Köhler");
    ...
    ph.delete(ph.getID("Huber"));
    ...
}
  
```

Als simples Array implementiert

1	Müller	0511/97865
2	Huber	0241/12-4711
3		
4	Schneider	762-19666
5	Schmidt	001-(303) 492
6		
7	Köhler	030/762-123

Kurt Schneider      Leibniz Universität Hannover      SWT 2015/16    219

## Zweiter Versuch: B versucht verkettete Liste

Beispiel Interfaces

Flexibel

```

PhoneList
first(): Node
search(String): Node
insert(Node)
append(Node)
delete(Node)
  
```

CMS verwendet PhoneList

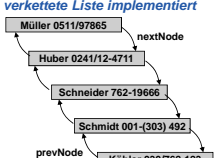
```

import theNodePackage.Node;
public void organize() {
    PhoneList ph = new PhoneList();
    ...
    // insert or append
    ph.insert(new Node("May", "089/4567"));
    nr=ph.search("Köhler").get_Number();
    ...
    ph.delete(ph.search("Huber"));
    ...
}
  
```

Als verkettete Liste implementiert

```

Node
get_Name(): String
set_Name(String)
get_Number(): String
set_Number(String)
get_nextNode(): Node
set_nextNode(Node)
get_prevNode(): Node
set_prevNode(Node)
  
```



Kurt Schneider      Leibniz Universität Hannover      SWT 2015/16    220

## C überredet A zu sortiertem Baum

Beispiel Interfaces

Schnelle Suche  
Flexibel

```

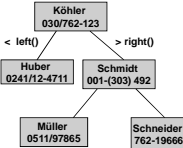
PhoneTree
first(): TreeNode
search(String): TreeNode
add(TreeNode)
delete(TreeNode)
  
```

CMS verwendet PhoneList

```

import theTreePackage.TreeNode;
public void organize(){
    PhoneTree ph = new PhoneTree();
    ...
    ph.add(new TreeNode("May", "089/4567"));
    nr=ph.search("Köhler").get_Number();
    ...
    ph.delete(ph.search("Huber"));
    ...
}
  
```

Als speziell sortierter Baum implementiert



Kurt Schneider      Leibniz Universität Hannover      SWT 2015/16    221

## Information Hiding: Interfaces!

Beispiel Interfaces

```

«interface»
IPhoneBook
add(String name, String number)
get_Number(String name): String
delete(String name)
  
```

Java Interface

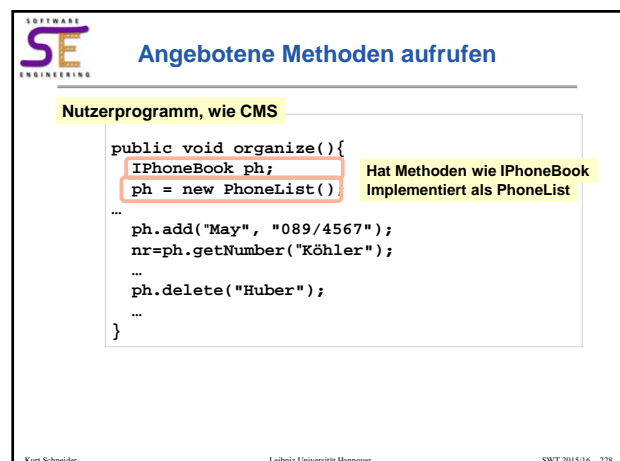
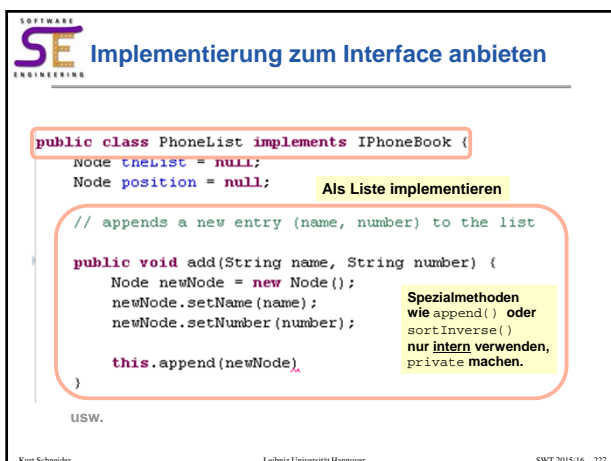
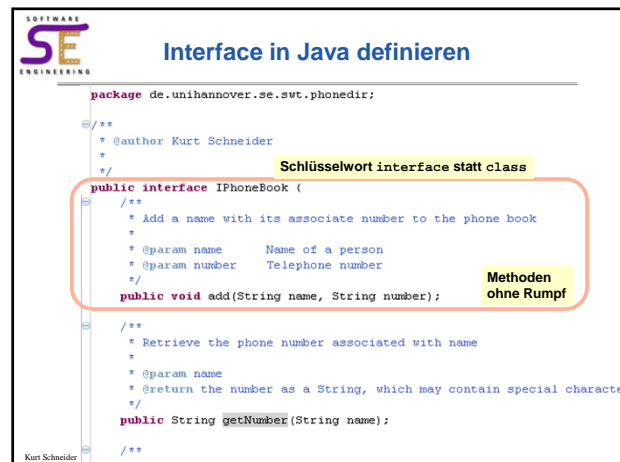
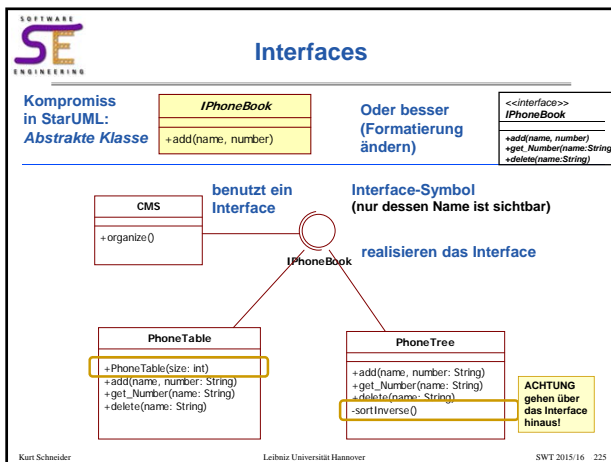
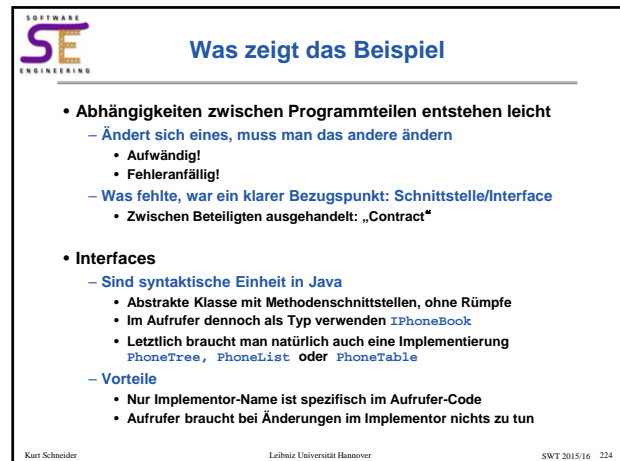
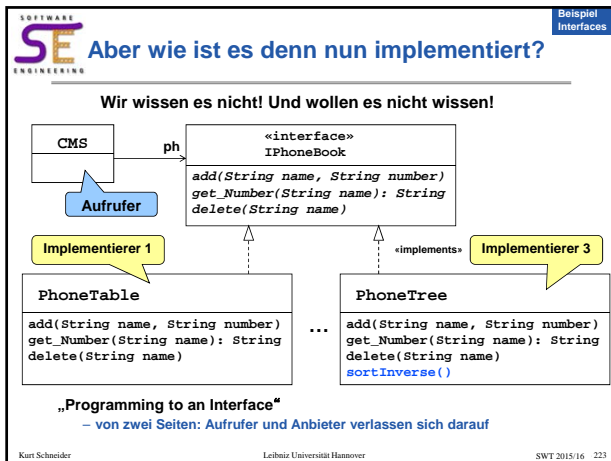
```


public void organize(){
    IPhoneBook ph;
    ph = new PhoneTree();
    ...
    ph.add("May", "089/4567");
    nr=ph.get_Number("Köhler");
    ...
    ph.delete("Huber");
    ...
}
  
```

oder: PhoneList()  
oder: PhoneTable(7)  
Das ist der einzige Unterschied!

Kurt Schneider      Leibniz Universität Hannover      SWT 2015/16    222







SOFTWARE


ENGINEERING

## Information Hiding

Zusammenfassung

- Trennung von Schnittstelle und Implementierung
  - Schnittstelle muss ich kennen
  - Implementierung lieber nicht (information hiding)
- In Java: Spezielle Klassenkonstruktion
  - Ein Interface kann keine Instanzen haben
    - Kann ja nichts „tun“, spezifiziert nur
    - Etwas allgemeineres Konzept: „Abstrakte Klasse“
  - „Implementor“ erbt alle Methoden der Schnittstelle
    - Überschreibt sie, d.h.: implementiert, was da passiert
    - Kann zusätzliche Methoden haben; die nutzt Aufrufer lieber nicht
    - Nutzer bezieht sich nur auf das Interface
  - Vorteil von „Programming to an Interface“
    - Beide Seiten können sich unabhängig ändern
    - Probleme werden zerlegt


Kurt SchneiderLeibniz Universität HannoverSWT 2015/16 229



SOFTWARE

ENGINEERING

## Beispiel aus dem Alltag: Info. Hiding



Volkswagen-Website, 14.11.04

Smart-Website, 14.11.04

Kurt SchneiderLeibniz Universität HannoverSWT 2015/16 230