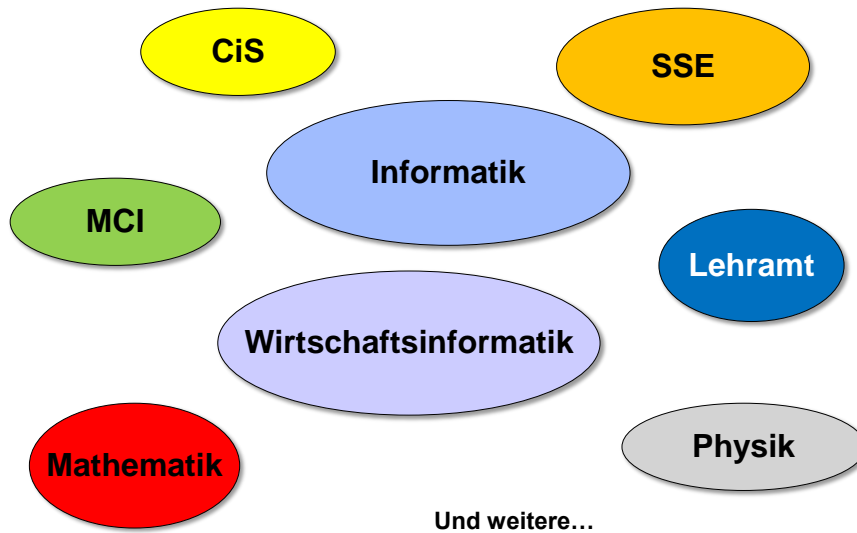


Videoaufzeichnung



- Es findet eine Videoaufzeichnung **aller 14 SE1-Vorlesungen** statt. Hierfür möchten wir euch auf einige Punkte aufmerksam machen:
- Es handelt sich um eine **Aufzeichnung**, nicht um einen Live-Stream!
- Gefilmt werden die **Vortragenden**, nicht das Plenum!
- Nur der **Ton des Vortragenden** wird über ein Funkmikrofon aufgenommen; aber **Hintergrundgeräusche** könnten trotzdem hörbar sein!
- **Fragen aus dem Plenum werden nicht gesondert aufgenommen.** Der Vortragende wiederholt die Fragen sowohl für das Plenum als auch für das Video.
- Die **Folien** werden ebenfalls zeitlich synchronisiert dem Video hinzugefügt.
- Die Vorlesung wird einige Tage später **als Quicktime-Movie im Uninetz veröffentlicht**: Wann und wo wird über den CommSy-Raum bekannt gegeben.
- Verlasst euch nicht darauf, dass ihr innerhalb einer bestimmten Frist alle Vorlesungen zum Download vorfinden werdet!

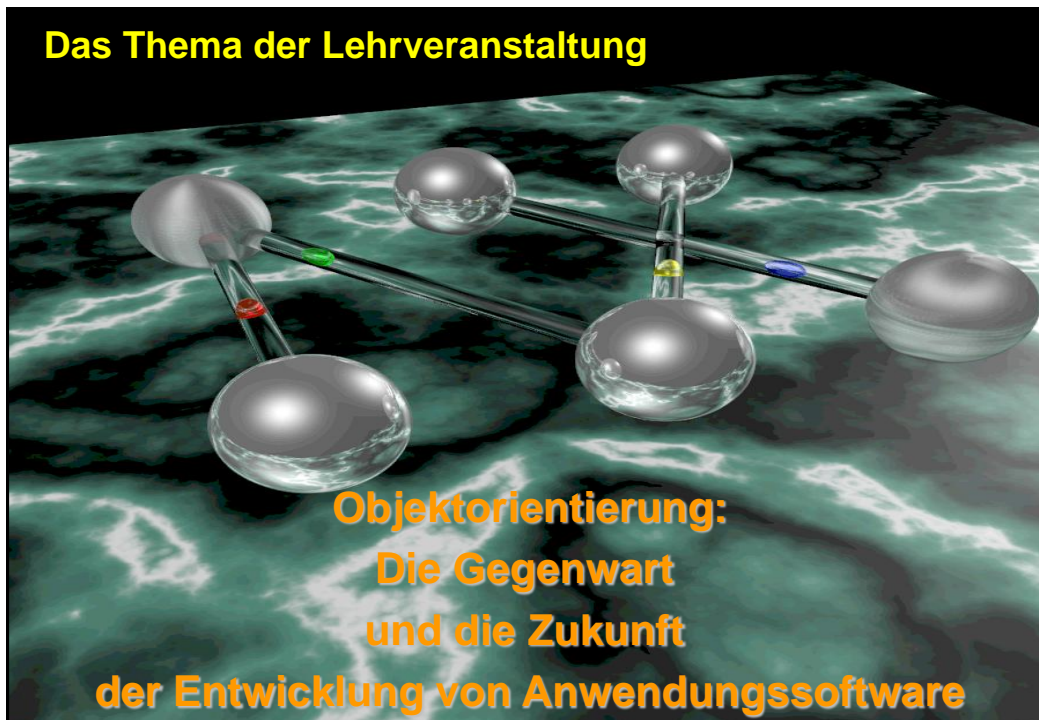
Teilnehmer bei SE1: Viele Studiengänge!



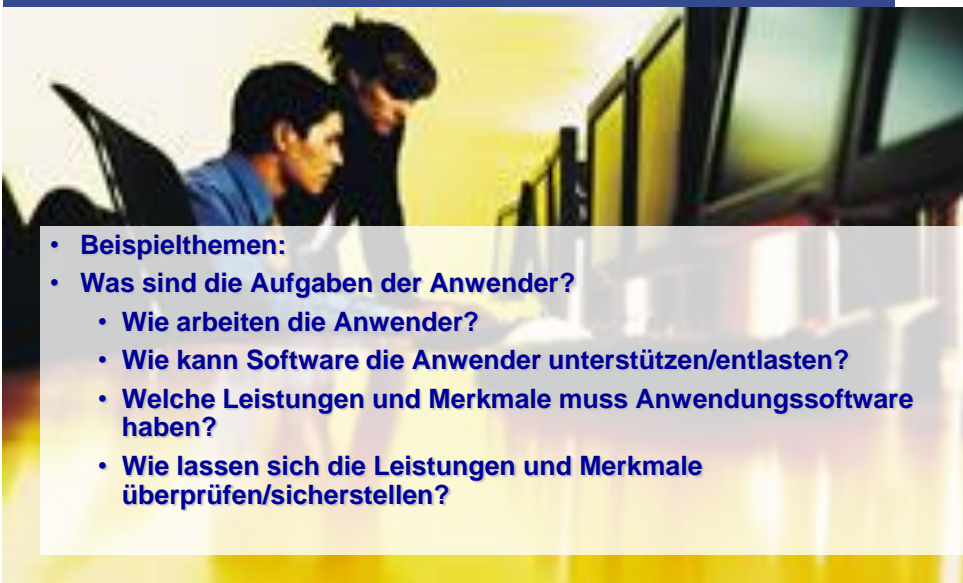
SE1 - Level 1

3

Das Thema der Lehrveranstaltung



Anwendungssoftware

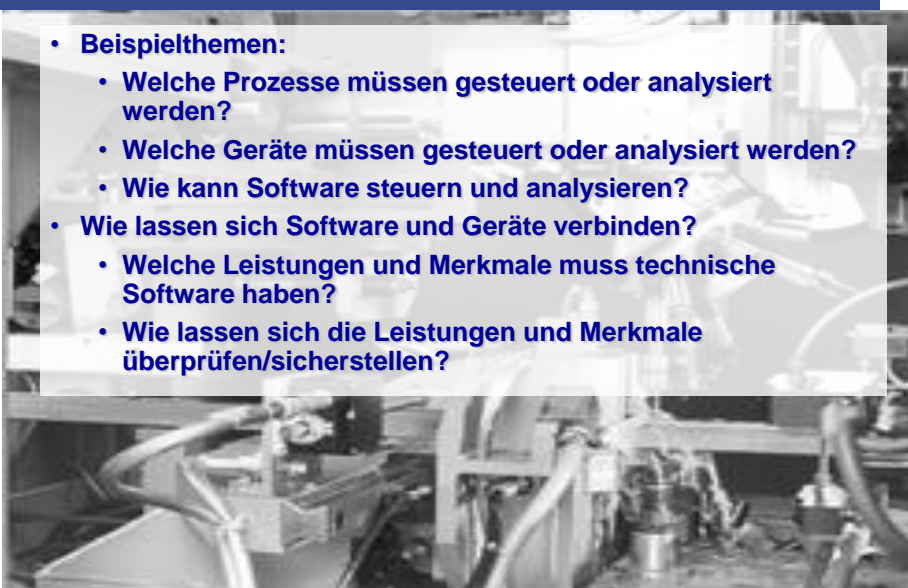


- **Beispielthemen:**
- **Was sind die Aufgaben der Anwender?**
 - Wie arbeiten die Anwender?
 - Wie kann Software die Anwender unterstützen/entlasten?
- **Welche Leistungen und Merkmale muss Anwendungssoftware haben?**
- **Wie lassen sich die Leistungen und Merkmale überprüfen/sicherstellen?**

SE1 – Level 1

5

Technische Software



- **Beispielthemen:**
- **Welche Prozesse müssen gesteuert oder analysiert werden?**
- **Welche Geräte müssen gesteuert oder analysiert werden?**
- **Wie kann Software steuern und analysieren?**
- **Wie lassen sich Software und Geräte verbinden?**
 - Welche Leistungen und Merkmale muss technische Software haben?
 - Wie lassen sich die Leistungen und Merkmale überprüfen/sicherstellen?

SE1 – Level 1

6

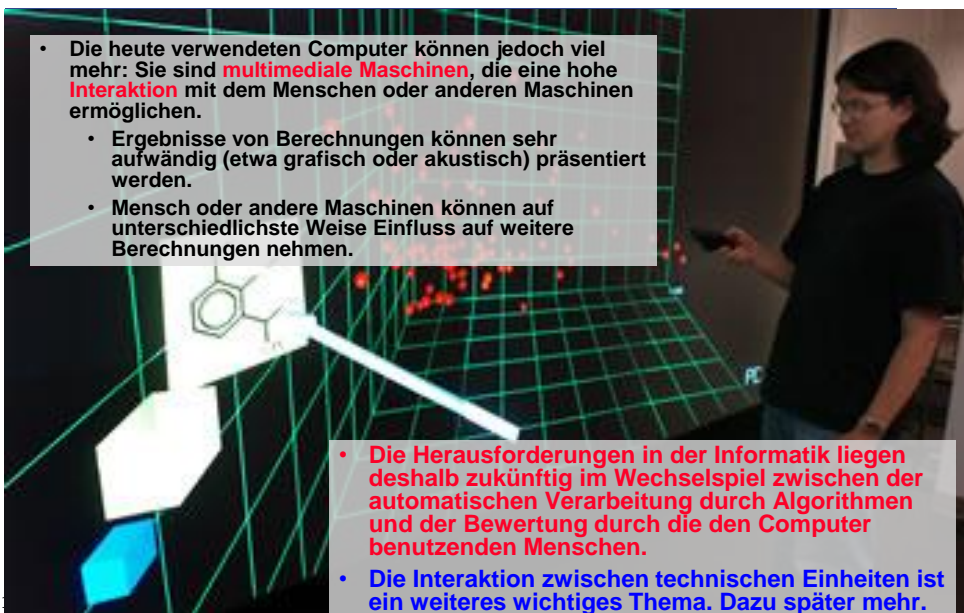
Vom Computer als „Rechner“ ...



SE1 - Level 1

7

... zum Computer als Interaktionsmaschine



SE1

8

Softwareentwicklung I und II (SE1 und SE2)

- **SE1 (2h VL + 2h Ü):**
 - Vorlesung + Übung: **Objektorientierte Programmierung (OOP)**
 - **Objektorientierte und imperative Grundlagen** von OOP
 - Am Beispiel von **Java**
 - Bewusste Beschränkung auf die Konzepte von Java
 - Bewusst nicht der volle Sprachumfang
 - Mit Hilfe von **BlueJ**
- **Ausblick SE2 (2h VL + 2h Ü):**
 - Vorlesung + Übung: **Objektorientierte Programmierung und Modellierung**
 - u.a. mehr zu Java (Vererbung, Fehlerbehandlung, GUI-Programmierung)
 - Entwurf und Modellierung: erste Kontakte mit Entwurfsmustern
 - Größere Beispielsysteme
 - Mit Hilfe von **Eclipse**



SE1 - Level 1

9

Inhalt SE1 laut Modulbeschreibung

- | | |
|---|---|
| ◆ Objekte, Klassen, Operationen | ◆ Umgang mit APIs, Schnittstellen, Interfaces |
| ◆ Datenfelder, Methoden, Konstruktoren, Parameterübergabe | ◆ Sammlungen benutzen: |
| ◆ Strukturierte Programmierung (Sequenz, Auswahl, Wiederholung) | ◆ Listen, Mengen, Abbildungen |
| ◆ Variablen und Typen, Attribut Zuweisungen | ◆ Operatoren, Gleichheit und Identität |
| ◆ Unterscheidung Syntax, Pragmatik bei Programmierung | ◆ Arrays implementieren: |
| ◆ Syntax in EBNF | ◆ Arrays, verkettete Strukturen |
| ◆ Basistypen und Referenztypen | ◆ Bäume, Hashing |
| ◆ Grundelemente der Modellierung mit UML | ◆ Komplexität verschiedener Implementierungen |
| ◆ Modulbegriff, Klasse als Modul, Klasse als Typ | ◆ Fehlersuche/Debugging, Fehlerbehandlung |
| ◆ Funktionale Dekomposition, Rekursion | ◆ Testen: Modultests, Regressionstests |




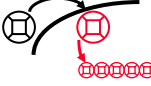
**Grundkonzepte der
imperativen und
objektorientierten
Programmierung**

SE1 - Level 1

10

Level 1: Einfache Klasse, einfache Objekte

Inhaltliche Gliederung von SE1

Stufe	Titel	Themen u.a.	Woche
1	 „Simple Klasse, simple Objekte“	Klasse, Objekt, Methode, Parameter, Feld, Variable, Zuweisung, Basistyp	1 - 4
2	 „Objekte benutzen Objekte“	Klasse als Typ, Referenz, Schleife, Rekursion	5 - 7
3	 „Schnittstellen mit Interfaces“	Black-Box-Test, Testklasse, Interface, Sammlungen benutzen	8 - 9
4	 „Hinter den Kulissen von Sammlungen“	Arrays, Sammlungen implementieren: Array-Liste, verkettete Liste, Hashing; Sortieren; Stack; Graphen	10 - 14

SE1 - Level 1

11

Struktur des SE1-Moduls

**Beginn der
Vorlesung? 14 Uhr?
14:15 Uhr?**

Die Vorlesung

Eine Einführung (heute)

13 Vorlesungen zur objektorientierten Programmierung und ihren imperativen Grundlagen

Die Übungen

1 etablierende Sitzung heute hier (Regeln, etc.)

13 Labortermine

Tutorium

Termin wird noch bekannt gegeben



SE1 - Level 1

12

Inhaltliche Aufteilung im Modul

Die Vorlesung

- befasst sich mit Konzepten,
- systematisiert Erfahrungen,
- stellt Einschätzungen der Dozenten zur Diskussion,
- ist **kein** Programmierkurs,
- ist **keine Vorbereitung** der Übungen.

Der Übungsbetrieb

- vermittelt praktische Erfahrungen,
- anhand von kleinen und größeren Konstruktionsaufgaben,
- die in Kleingruppen erarbeitet und diskutiert werden,
- gibt unmittelbares Feedback über den Lernfortschritt.

Über die Inhalte hinaus geht es um...

- den Umgang mit Fremd-Sprachen
- den Umgang mit Notationen, Fachbegriffen
- Texte schreiben
- Präsentieren lernen
- Die Menschen hinter der Informatik
- und...



Level 1: Einfache Klasse, einfache Objekte



Das SE1 Übungskonzept

- **Unsere Ziele: Ihr sollt**
 - aktiv den Umgang mit der Programmiersprache Java lernen,
 - Konzepte verstehen und erklären können.
- **Dabei steht im Mittelpunkt:**
 - **PROGRAMMIEREN**
 - hier: objektorientierte Konstruktion als Konzeptarbeit gepaart mit „handwerklicher Tätigkeit“,
 - aber auch: Vorahnung von professioneller objektorientierter Programmierung als Teamarbeit.

„Wer sich für die kleinen Dinge zu groß fühlt,
ist für die großen Dinge meist zu klein.“

Laotse

Das Konzept in der Umsetzung

- **2 h (3 LP) klassische Übung** sind realisiert als
 - **3,0 h betreute Gruppenarbeit im Labor** (Räume des RZ, Haus D)
 - Programmieren und Lernen **in Paaren am Rechner**
- 1 Betreuer betreut pro Labor 7 bis 8 Studierende (Vorgabe vom Fachbereich)
 - Also: **max. 4 Paare pro Betreuer**
- **Vorteile:**
 - Weniger Zeitbedarf trotz intensiverer Beschäftigung mit dem Stoff
 - Schnelle Hilfe bei Fragen
 - Gruppenarbeit bedeutet gemeinsames Erarbeiten
 - Unmittelbare Betreuung direkt am Problem

Einleitende Texte der Aufgabenblätter

- **Jedes Aufgabenblatt hat, neben den Aufgaben, einen einleitenden Text.**
Dieser Text setzt die wichtigsten Begriffe und Konzepte der jeweiligen Übung in einen Zusammenhang.
- **Jeder Teilnehmer sollte diesen Text vor Beginn der Übung gelesen haben!**
Die Blätter sind üblicherweise einige Tage vor der jeweiligen Durchführung digital im Web verfügbar.
- **Wer den einleitenden Text ohne Probleme versteht, wird in der Übung gut zurecht kommen.**
- **Wer den Text nicht vollständig versteht: Keine Panik!**
Das Durchführen der Aufgaben wird mehr Klarheit bringen. Anschließend den Text noch einmal lesen. Bei Fragen: fragen!

Zeitaufwand für das Modul SE1

Das gesamte Modul hat einen Umfang von **6 Leistungspunkten** (LP).
Jeder LP soll ca. **30 Zeitstunden** innerhalb des Semesters entsprechen.
Für SE1 darf der **gesamte Zeitaufwand** also bei **180 Stunden** liegen.

Wir verteilen diese Stunden nicht lediglich auf die Vorlesungswochen (14), sondern **auf 22,5 Wochen** eines gesamten Semesters (26 Wochen).

Rechnerisch ergibt sich somit ein Zeitaufwand von **min. 8 Stunden pro Woche**.

Bei 2 Vorlesungsstunden und 3 Stunden betreuter Gruppenarbeit im Labor bleiben **mindestens 3 Stunden für Vor- und Nachbereitung von Vorlesung und Übung!**

Das ist aus unserer Sicht die **Untergrenze**:

- Auch wer Talent und/oder Vorkenntnisse hat, sollte diese Zeit auf jeden Fall investieren.
- Wer keine Vorkenntnisse hat und nicht gern programmiert, braucht vermutlich mehr Zeit!

Von Schafen und Ziegen...

**“There are 10 kinds of people in the world...
those who understand binary, and those who don't.”**



- **Programming Sheep** and **Non-Programming Goat:**

- <http://www.codinghorror.com/blog/archives/000635.html>



Übrigens: Es gibt für alles immer zwei Möglichkeiten...

Programmieranfänger oder alter Hase?

- **Du bist Programmieranfänger?**

- Der Anspruch von SE1 ist, Dich an das Programmieren heran zu führen.
- Es gibt **zwei Möglichkeiten**:
 - Minimaler Aufwand durch „Ranhängen“ an Stärkere.
 - **Aktive Teilnahme an den Übungen; jede Gelegenheit nutzen, praktische Erfahrungen mit der Programmierung zu sammeln.**

- **Du hast schon Erfahrung mit Programmieren gesammelt? Du hattest als Leistungskurs Informatik in der Schule? Du programmierst schon für Geld?**

- Dann ist vieles in SE1 möglicherweise sehr einfach für Dich.
- Es gibt **zwei Möglichkeiten**:
 - Minimaler Aufwand, gute Note abstauben.
 - **Interessierte Teilnahme an den Übungen; Erfahrung weitergeben an Teilnehmer ohne Vorerfahrung.**

Wo und wann gibt's Unterlagen, etc.

Folienskripte (vier Teile, jeweils etwas im Voraus):

Sind als PDF-Dateien im CommSy verfügbar.
Außerdem werden gedruckte Fassungen in den Übungen verteilt.

Vorlesungsfolien:

Stehen nach jeder Vorlesung als aktualisierte PDF-Datei im CommSy zur Verfügung (gilt auch für heute).

Manuskript eines Buchentwurfs zu SE1 (Level 1):

Steht als PDF-Datei im CommSy zur Verfügung.

Aufgabenblätter:

Sind nach ihrer Erstellung (meist Montag vor der jeweiligen Übungswoche) als PDF-Dateien im CommSy verfügbar.

Stehen außerdem gedruckt in den Übungen zur Verfügung.

Level 1: Einfache Klasse, einfache Objekte

Das CommSy – unser Kommunikationsraum!

<https://www.mincommsy.uni-hamburg.de>

1. **Account** beim CommSy beantragen (hier könnt ihr auch euer Stine-Login benutzen)
2. Mit der (neuen) Kennung **einloggen**
3. Dem **Raum** durch Klicken auf die Tür **beitreten**



SE1 - Level 1

Scheinkriterien (Auszug)

Ausgangslage:

- Das Semester hat 14 Wochen, in denen **13 Präsenztermine zu je 3 Zeitstunden stattfinden**.
- Die SE1-Übungswoche beginnt stets am **Donnerstag** und endet am **Mittwoch**. **Wichtig für die Fristen!**
- Pro Aufgabenblatt gibt es **meistens 4** (mindestens 3) **Aufgaben**.

Individueller Beitrag:

- **Alle Aufgabenblätter** müssen bearbeitet werden.
- Persönliche Anwesenheit an **mindestens 11 Übungsgruppenterminen**.
- **Reguläre Abnahme: Jedes Aufgabenblatt soll in seiner Ausgabewoche in den Laborzeiten gelöst und von den Übungsgruppenleitenden abgenommen werden.**
Ausnahme: Spätestens in der Woche nach der Ausgabewoche eines Aufgabenblattes müssen alle Abnahmen für dieses Blatt abgeschlossen sein.
- Für eine Abnahme müssen die **Lösungen vorgeführt und erklärt** werden.
- **Alle Aufgaben auf einem Aufgabenblatt müssen bearbeitet werden.** Die jeweils letzten Aufgabe muss nicht gegen gezeichnet werden.

Weiter gilt: Die Aufgaben werden **in Paaren** bearbeitet.

SE1 - Level 1

24

Wir empfehlen das Pflegen eines Glossars

• Problem:

- Es gibt viele neue Begriffe gerade in den ersten Wochen.
- Alles baut aufeinander auf.
- Wer die Begriffe nicht beherrscht, wird große Schwierigkeiten mit dem nachfolgenden Stoff haben.

• Lösung: Glossar

- Alle für euch wichtigen Begriffe schreibt ihr regelmäßig – am Besten handschriftlich – in einer Art Lexikon mit Kurzbeschreibung.
- Bei den Abnahmen können wir auf euer Glossar zurückgreifen und euch besser helfen.
- Habt ihr bei der Abnahme Schwierigkeiten mit den Begriffen, werden die Betreuer das Führen eines Glossars verlangen.

Der Wochenplan WS 2010/11

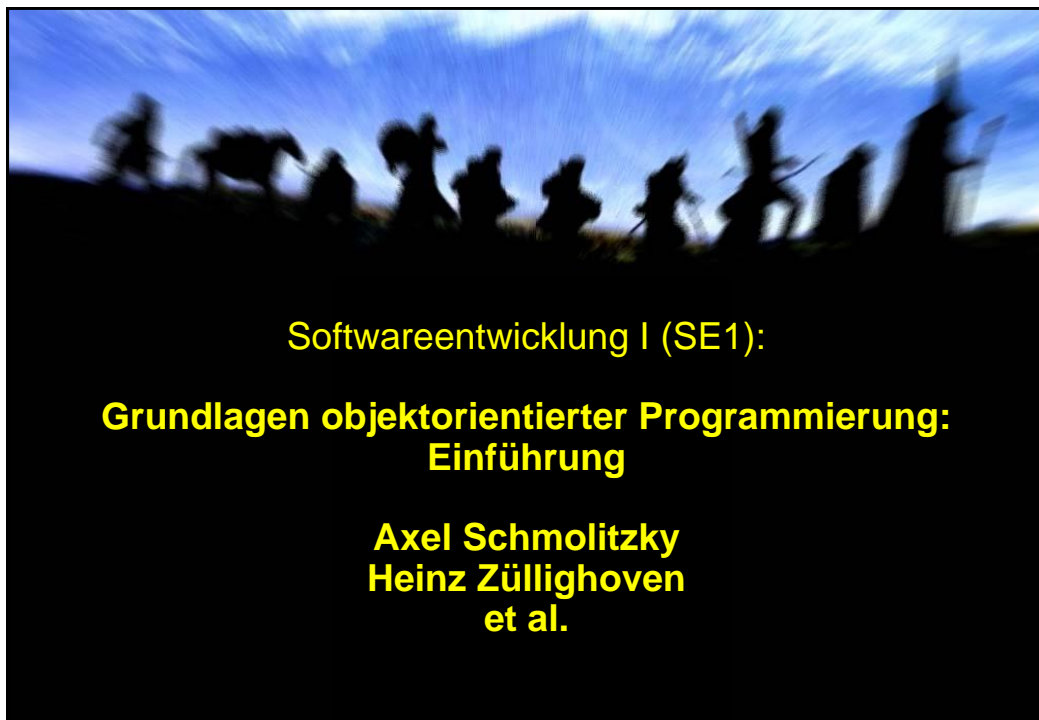
Übungsräume: Campus Stellingen, Haus D
D-010, D-017, D-018 (D-114, D-118)

	Montag	Dienstag	Mittwoch	Donnerstag	Freitag
09 – 12	Labor 1 (MoVo) Kapazität 70 Frei: 15 frei!	Labor 2 (DiVo) Kapazität 84 Frei: 13 frei!	Labor 4 Kapazität 60 Frei: 0 VOLL!	Labor 5 Kapazität 42 Frei: 31 frei!	Labor 7 Kapazität 63 Frei: 29 frei!
12 – 14					
14/15 – 17/18		Labor 3 (DiNa) Kapazität 49 Frei: 11 frei!	(14-16) Vorlesung Hörsaal A (Chemie) (16:30-18:00) Tutorium	Labor 6 Kapazität 90 Frei: 0 VOLL!	

An- und Ummeldungen über Stine bis zum 30.10.2011, 13:00 Uhr.

Wo bekomme ich Hilfe?

- **Anmeldung zur Übung:**
 - **Online-Anmeldung** ist über die zentrale Anmeldung der Uni verfügbar:
<https://www.stine.uni-hamburg.de/>
- **Organisatorisches zu den Übungen und Wechselwünsche:**
 - Christian Späh (D-214, spaeh@informatik.uni-hamburg.de).
- **Inhaltliches zu Vorlesung und Übungen:**
 - Axel Schmolitzky (D-206, schmolit@informatik.uni-hamburg.de)
oder bei den jeweiligen ÜbungsgruppenleiterInnen.



Softwareentwicklung I (SE1):

**Grundlagen objektorientierter Programmierung:
Einführung**

**Axel Schmolitzky
Heinz Züllighoven
et al.**

Level 1: Einfache Klasse, einfache Objekte

Grundlegende Lehrbücher

David J. Barnes, Michael Kölling: **Java lernen mit BlueJ – Eine Einführung in die objektorientierte Programmierung**, 4. Aufl., Pearson Studium, 2009. (deutsche Übersetzung von: **Objects First with Java - A Practical Introduction using BlueJ**, 4. Aufl., Pearson Education, 2009.)
[Der aktuelle „Objects First“ Ansatz mit BlueJ. Teilweise Grundlage für die Übungen. Gut geeignet zum Selbststudium.]



Cornelia Heinisch, Frank Müller-Hofmann, Joachim Goll: **Java als erste Programmiersprache**, 6. überarb. u. erw. Aufl., Teubner, Stuttgart, 2010.
[Eine gute konventionelle Einführung in Java.]

Khalid Mughal, Torill Hamre, Rolf W. Rasmussen: **Java Actually: A First Course in Programming**. Thomson, 2007.
[Engelsprachige Einführung in das Programmieren mit Java, die – ähnlich wie wir – Vererbung bewusst ausklammert.]

SE1 – Level 1

29

Mehr zu Java

Reinhard Schiedermeier, Klaus Köhler: **Das Java-Praktikum**, 2. überarb. u. erw. Aufl., dpunkt Verlag, 2011.
[Eine sehr nützliche Sammlung von Aufgaben zu Java.]

Ken Arnold, James Gosling, David Holmes: **The Java Programming Language**, Fourth Edition, Addison-Wesley, 2005.
[Der Java-Klassiker. Knapp und ohne didaktischen Anspruch. Eher zum Einlesen für erfahrene Programmierer.]

David Flanagan: **Java in a Nutshell**, 5. Aufl., O'Reilly Media, 2005.
[Der Java-Nachschlage-Klassiker. Kurz und knapp (auf 1224 Seiten) durch die wesentlichen Java-Bestandteile und -Packages.]

Joshua Bloch: **Effective Java Programming Language Guide**, 2. Aufl., Addison-Wesley Longman, 2008.
[Die Fallstricke von Java ausführlich und sehr kompetent. Eher für Fortgeschrittene.]

James Gosling, Bill Joy, Guy Steele: **The Java Language Specification**, Third Edition, Addison-Wesley, Juli 2005.
[Die offizielle Sprachdefinition. Für die, die es genau wissen wollen.]

SE1 – Level 1

30

Weitere Grundlagenwerke

Peter Rechenberg, Gustav Pomberger, **Informatik-Handbuch**,
Hanser-Verlag, 4., aktualis. u. erw. Aufl., 2006.
[Handbuch der wesentlichen Gebiete der Informatik.]

Duden Informatik, Dudenverlag, Ausgabe 2006.
[Grundbegriffe kurz und grundlegend definiert.]

Grady Booch, James Rumbaugh, Ivar Jacobson, **The Unified
Modeling Language Reference Manual**, Addison-Wesley, 2004.
[Die Referenz von den „Erfindern“ der UML.]

Englischsprachig und weiterführend

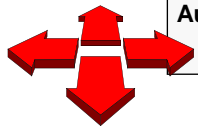
Robert W. Sebesta, **Concepts of Programming Languages**,
Addison-Wesley Educational Publishers, 9. Auflage, 2009.
[Gute und verständliche Einführung in die Definition von
Programmiersprachen.]

Bertrand Meyer: **Object-Oriented Software Construction**, Second
Edition, Prentice Hall, 1997.
[Der Klassiker unter den Programmierbüchern zur
Objektorientierung (am Beispiel der Sprache Eiffel). Viele
allgemeingültige und wertvolle Hinweise. Engagiert und bissig.]

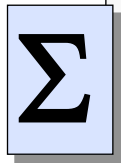
Heinz Züllighoven et al.: **Object-Oriented Construction Handbook**.
dpunkt-Verlag, 2004.
[Unser Diskussionsbeitrag. Für Fortgeschrittene (und die, die es
werden wollen :-)]

Level 1: Einfache Klasse, einfache Objekte

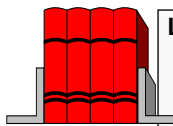
Symbole und Zeichen

**Ausblick:**

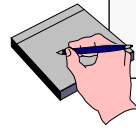
sagt, was kommt.

**Zwischensumme:**

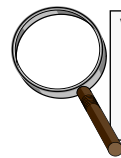
hebt für einzelne Themen hervor, was uns wichtig ist.

**Literaturliste:**

Grundlage für den nachfolgenden Teil mit Bewertung.

**Zentraler Begriff:**

wird hier eingeführt oder erläutert.

**Vertiefung:**

hier werden Hintergründe etwas genauer beleuchtet.

© <Autor>

Literaturreferenz:

auf Bücher aus der Literaturliste.

SE1 - Level 1

33

Symbole und Zeichen (II)

**Merker:**

kommentiert, stellt Bezüge her und hebt etwas hervor.

**Imperativer Begriff:**

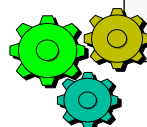
hier wird ein Begriff aus der imperativen Programmierung verwendet, der sich von der OO-Terminologie unterscheidet.

**Java-Beispiel:**

erläutert ein Konzept oder Konstrukt am Beispiel der Sprache Java.

**Negativbeispiel:**

warnt vor häufigen, aber bedenklichen Konstruktionen.

**UML-Notation:**Diagramm in der Notation der Unified Modeling Language.**Softwaretechnik:**

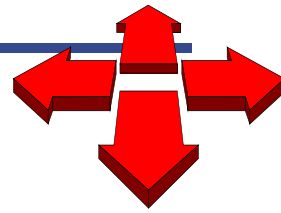
diskutiert Erfahrungen und Prinzipien.

SE1 - Level 1

34

Die objektorientierte Sichtweise

- Objektorientierte Modellierung und Programmierung
- Objekte: Klienten und Dienstleister
- Andere Paradigmen: Funktionale und Logische Programmierung

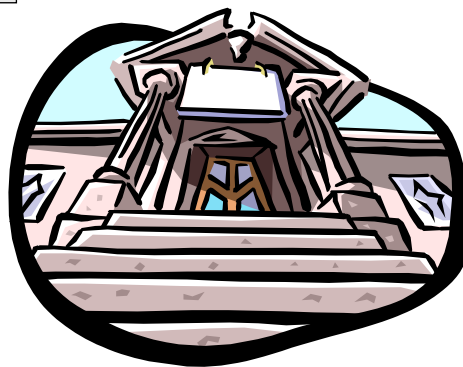


SE1 - Level 1

35

Ein Szenario

Ein Bankkunde geht zu seiner Bank

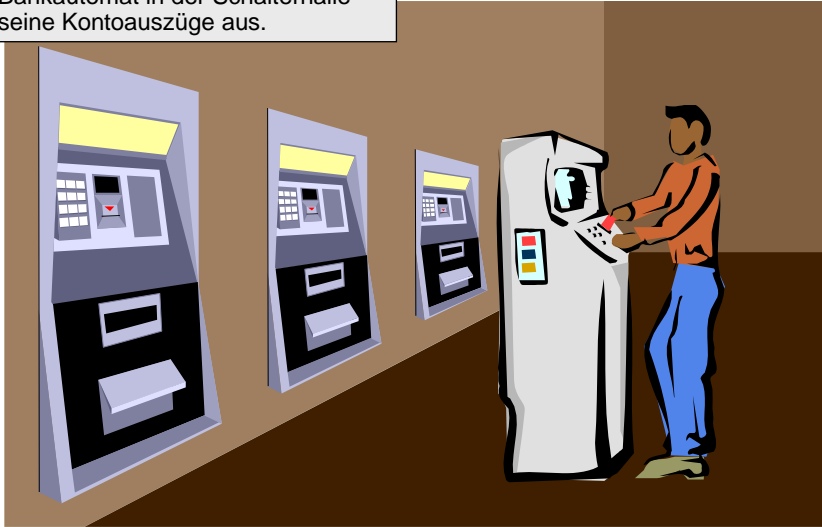


SE1 - Level 1

36

Ein Szenario (II)

Der Bankkunde drückt sich am Bankautomat in der Schalterhalle seine Kontoauszüge aus.



SE1 - Level 1

37

Ein Szenario (III)

Dann lässt sich der Bankkunde mit seiner ec-Karte 300 EUR am Bankautomat von seinem Girokonto auszahlen.



SE1 - Level 1

38

Das Szenario: Kunde erledigt Bankgeschäfte



Personen erledigen Aufgaben, um ein Ziel zu erreichen.

Aufgaben werden durch verschiedene Tätigkeiten / Handlungen erledigt.

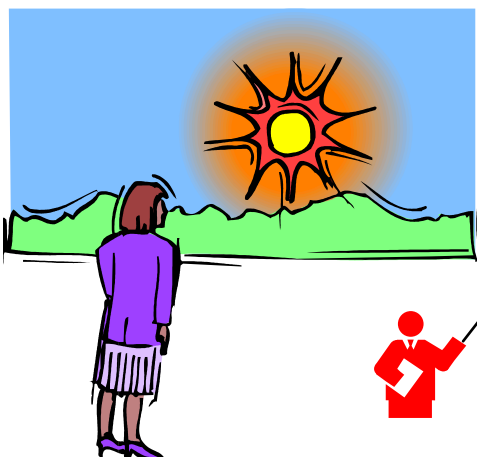
Handlungen sind charakterisiert durch die Art und Weise, wie die Personen mit Gegenständen umgehen.

Ein Bankkunde geht zu seiner Bank.

Der Bankkunde drückt sich am Bankautomat in der Schalterhalle seine Kontoauszüge aus.

Dann lässt sich der Bankkunde mit seiner ec-Karte 300 EUR am Bankautomat von seinem Girokonto auszahlen.

Objektorientierung ist eine „Sicht der Welt“



Oft spricht man von einem **Paradigma**. Damit ist eine Sichtweise gemeint, die uns hilft, einen Sachverhalt zu interpretieren und zu verstehen.

Paradigma [gr.-lat.] das; -s, ...men (auch: -ta): ...
4. Denkmuster, das das wissenschaftliche Weltbild, die Weltsicht einer Zeit prägt.

Level 1: Einfache Klasse, einfache Objekte

Modellieren und Programmieren

• Modellieren:

- Ein Abbild von etwas machen, um damit etwas zu zeigen, prüfen, auszuprobieren.
- Beim Modellieren verwenden wir oft vorgegebene „Modell-elemente“, Regeln und eine bestimmte Herangehensweise.
- **Objektorientierte Modellierung** ist ein Beispiel für eine solche Modellierungsart.



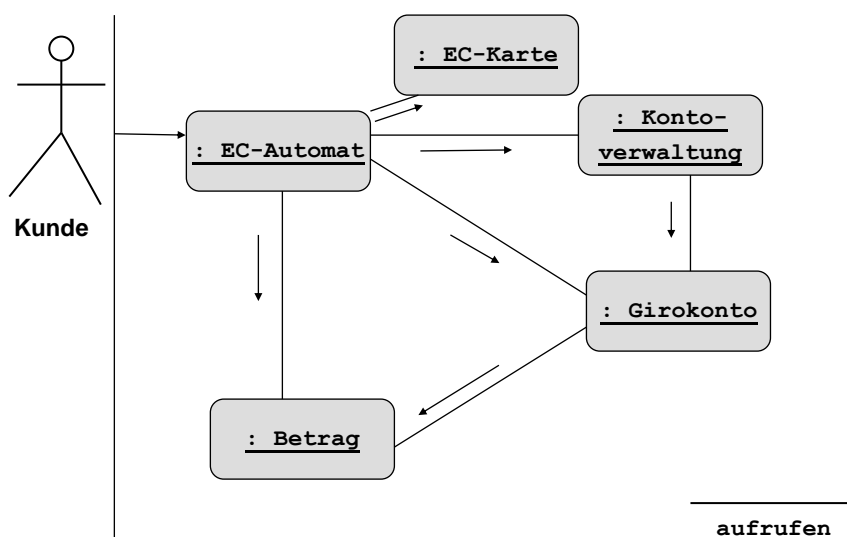
• Programmieren:

- Ein Programm für einen Rechner in einer Programmiersprache schreiben, testen, weiterentwickeln.
- Beim Programmieren verwenden wir die Elemente der gegebenen Programmiersprache, Regeln der Programmkonstruktion und bestimmte Herangehensweisen.
- **Objektorientierte Programmierung** in Java ist eine solche Art der Programmierung.

SE1 - Level 1

41

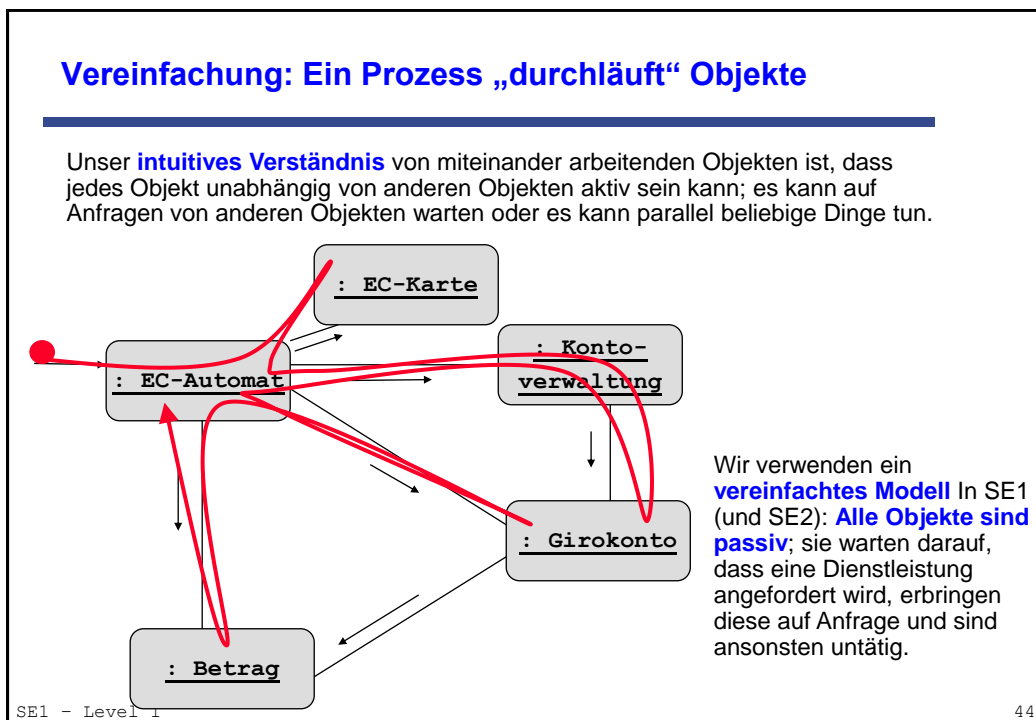
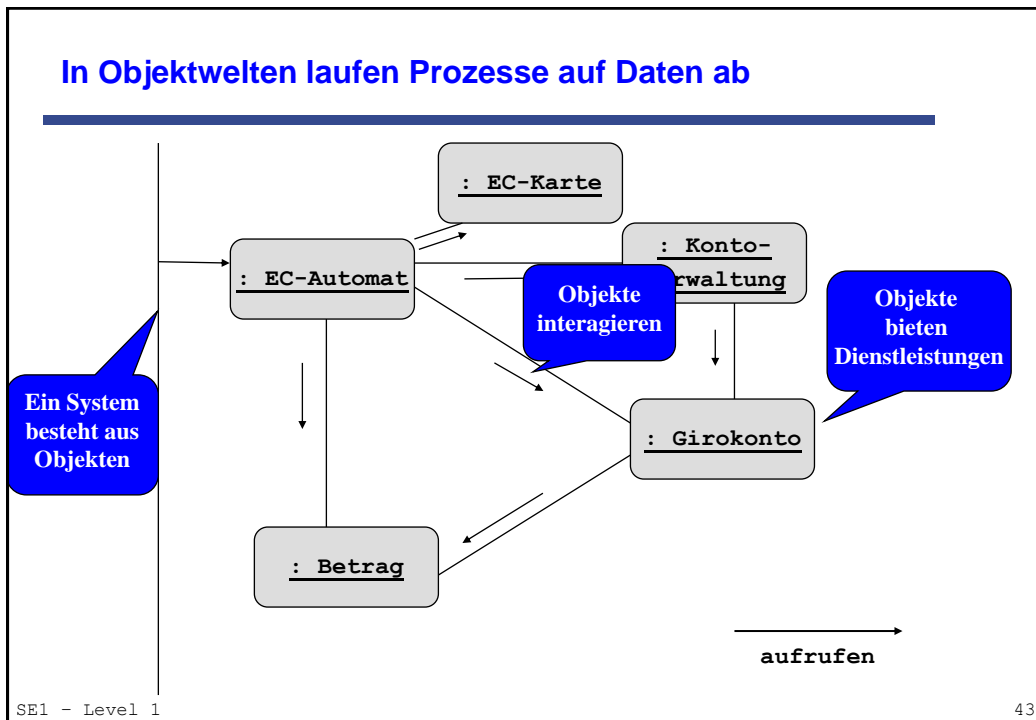
Akteure interagieren mit einem System von Objekten



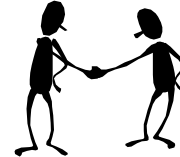
SE1 - Level 1

42

Level 1: Einfache Klasse, einfache Objekte



Dienstleister und Klienten



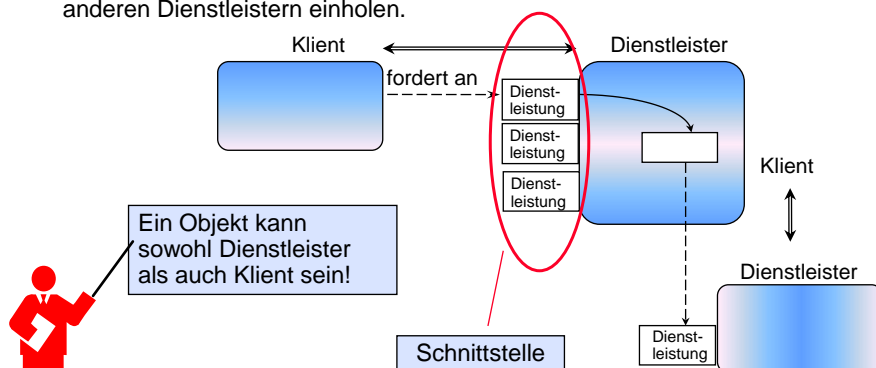
- Das Objekt, das bei einer bestimmten (Teil-)Aufgabe einen Dienst leistet, ist der **Dienstleister**.
- Das Objekt, das eine konkrete Dienstleistung eines anderen Objektes in Anspruch nimmt, wird als **Klient** bezeichnet.

SE1 - Level 1

45

Dienstleistungen an der Schnittstelle

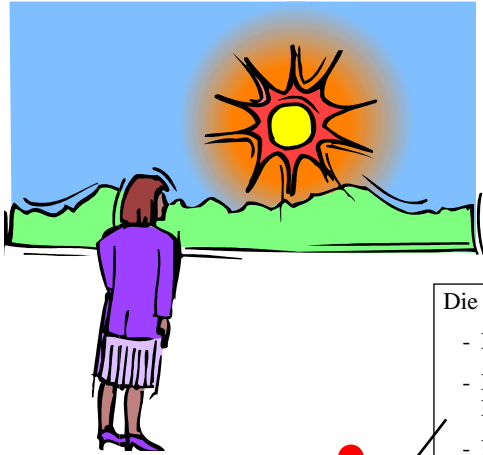
- Objekte bieten **Dienstleistungen** als **Methoden** an ihrer **Schnittstelle** an.
- Diese Dienstleistungen werden von anderen Objekten, den Klienten, benutzt. Dazu fordert der Klient eine Dienstleistung des Anbieters an.
- Der Anbieter kann selbst wieder Teile seiner Dienstleistung von anderen Dienstleistern einholen.



SE1 - Level 1

46

Es gibt andere informatische Sichten der Welt: z.B. die funktionale Sicht



Die funktionale Sicht

- Ein Programm besteht aus Funktionen.
- Eine Funktion berechnet einen Ergebniswert anhand von Eingabewerten.
- Funktionen können kombiniert werden.
- Ein (Wert-) Ausdruck kann immer durch einen wertgleichen anderen Ausdruck substituiert werden.

SE1 - Level 1

47

Beispiel für funktionale Programmierung

```
(define (max a b)
  (if (> a b) a b))
```

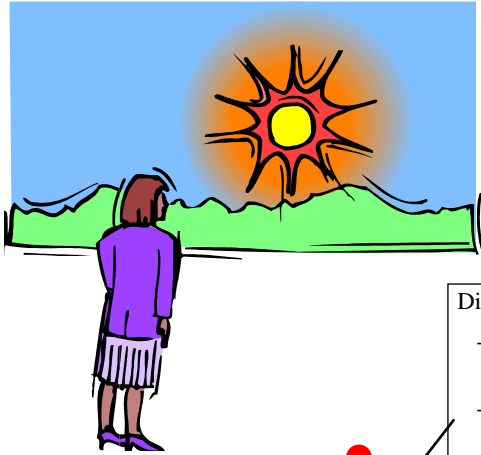
- Eine Funktion „max“ wird definiert.
- Sie nimmt die Werte **a** und **b**.
- Sie gibt den größeren der beiden Werte zurück.

Mehr dazu:
wählbar in SE3.

SE1 - Level 1

48

Es gibt andere informatische Sichten der Welt: z.B. die logische Sicht



Die logische Sicht

- Ein Programm besteht aus logischen Regeln und Fakten.
- Logische Anfragen können evaluiert werden. Dabei wird anhand der Regeln und Fakten abgeleitet, ob eine logische Anfrage wahr ist.

SE1 - Level 1

49

Beispiel für logische Programmierung

max (A, A, A) .

max (A, B, A) :- A > B .

max (A, B, B) :- A < B .



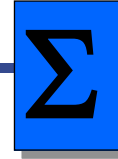
- Ein dreistelliges Prädikat „**max**“ wird definiert.
- 1. Regel: Wenn zwei Zahlen gleich sind, ist auch ihr Maximum gleich.
- 2. Regel: Wenn **A** größer **B**, dann ist ihr Maximum gleich **A**.
- 3. Regel: Wenn **A** kleiner **B**, dann ist ihr Maximum gleich **B**.

Mehr dazu:
wählbar in SE3.

SE1 - Level 1

50

Zusammenfassung



- Softwaresysteme können als eine **Menge interagierender Objekte** modelliert und programmiert werden.
- Objekte bieten **Dienstleistungen** an.
- Die für Klienten abrufbaren Dienstleistungen bilden die **Schnittstelle** eines Objektes.
- Es gibt auch andere Sichtweisen der Programmierung: die **funktionale** und die **logische** Sichtweise.