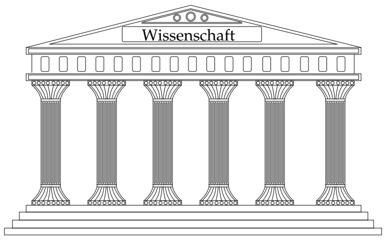
Globale Gemeinschaft der WissenschaftlerInnen



- Alle Menschen, die zu einem bestimmten Thema arbeiten, bilden zusammen eine Gruppe.
- Ihre Arbeitsergebnisse bilden ein Fachgebiet.
- Konferenzen und Zeitschriften artikulieren das Fachgebiet.
- Programmkomitees und Herausgabegremien überwachen die Einhaltung der Standards.

Standards

- Originalität – niemals hinter den Stand der Diskussion zurück!
- Korrektheit
- Einbeziehung verwandter Arbeiten – niemals isoliert arbeiten!
- Relevanz – wem nützt das Ergebnis?

Sie sind diesem Club soeben beigetreten!

Lokale Gemeinschaft der Lehrenden und Lernenden

- Den Stand der Diskussion in einem Gebiet vorantreiben/kennenlernen.
- Die Methoden in dem Gebiet vermitteln/beherrschen.
- Wissenschaftliche Texte verstehen, kontroverse Meinungen verfolgen, zu einer eigenen Meinung gelangen und diese vertreten.
- Selbstständiges Erarbeiten eines Themas.
- Erkennen eines Problems, Entwurf und Umsetzung einer Lösung.
- Vermitteln des angeeigneten Wissens in mündlichem Vortrag und schriftlichen Ausarbeitungen.

- Lehrende: möglichst viele möglichst hoch qualifizierte Nachwuchskräfte
- Lehrende: Ausbildung der Industriepartner und Kollegen von morgen.
- Lernende: Bescheinigung der gewonnenen Fähigkeiten und Fertigkeiten durch erfolgreiche Klausur (Prüfung).
- Lernende: Qualifikation für den Beruf.

Kriterien zur Qualität von InformatikerInnen

1. Anwendern so gut zuzuhören, daß eine umfassende Aufgabenbeschreibung gemeinsam erarbeitet werden kann;

2. übersichtliche Modelle für komplexe Aufgaben zu erstellen;

3. Standardumsetzungen zu kennen und bei der Konkretisierung von Modellen einsetzen zu können;

4. sich anhand der Standardumsetzungen rasch in einer (auch: neuen, noch unbekannten) Programmiersprache wiederzufinden und von der Sprache in Programmbibliotheken bereitgestellte Umsetzungen zu nutzen;

5. die eigene Programmentwicklung klar zu dokumentieren und nach Effektivität und Effizienz zu bewerten.

Motivation der Lernenden

1. Sichere Arbeitsplätze
2. Gutes Gehalt
3. Vielseitige Aufgaben im Beruf
4. Menschen helfen
5. im Team ein gemeinsames Projekt realisieren
6. Flexible Arbeitszeiten, Ortsunabhängigkeit
7. Verstehen, wie die Informationstechnologie funktioniert
8. Programmieren/Formalisieren lernen
9. Abstrakte Beschreibungen von (Alltags-)Prozessen entwerfen und zum Laufen bringen
10. Algorithmen und Programme analysieren

Vorkenntnisse der Lernenden

1. Programmiererfahrung in:
   - Basic und Visual Basic
   - Pascal und Delphi
   - C
   - C++
   - JAVA
   - sonstige

2. Internet-Navigation

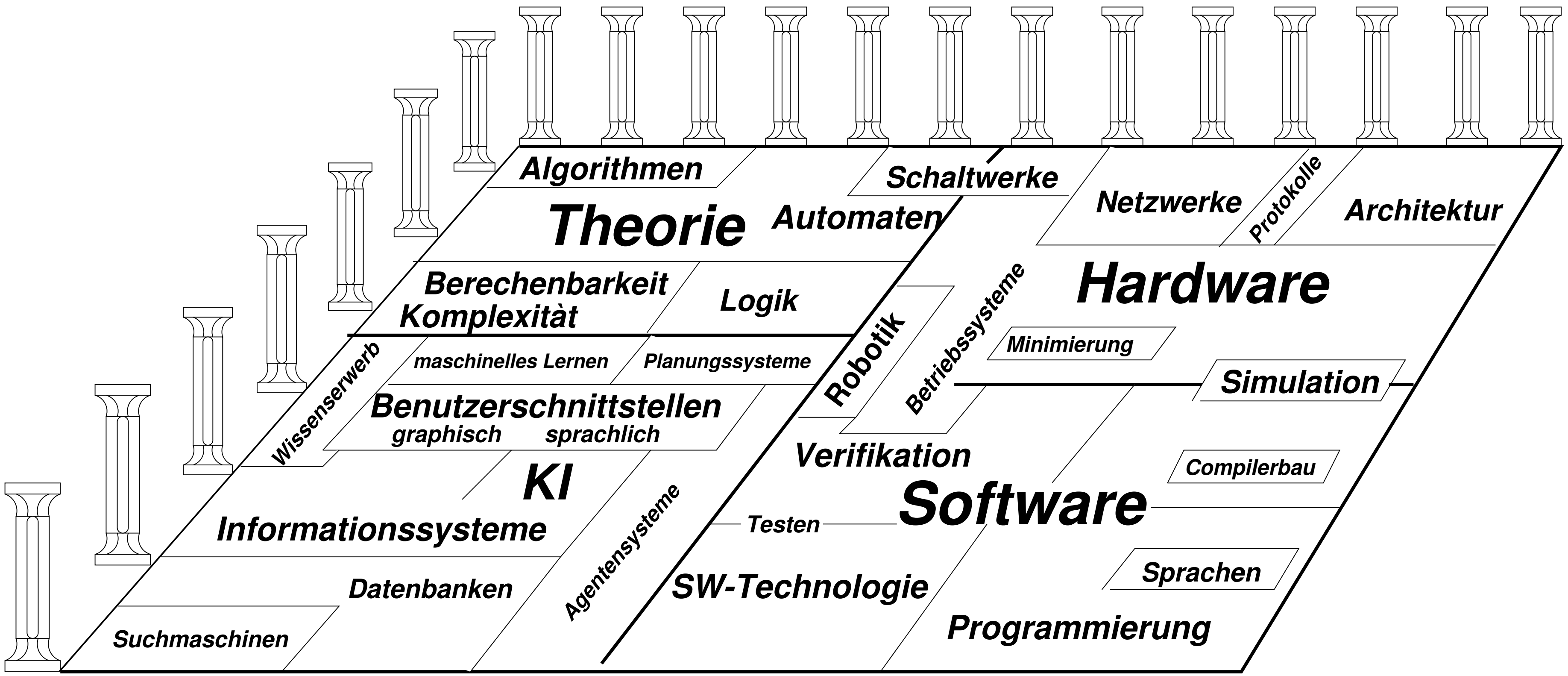
3. Herstellung von WWW-Seiten HTML

4. EMail

5. Logik

6. Ungefähre Vorstellungen eines Rechnersystems (Rechnerarchitektur)

Gebiete der Informatik



Das Informatikstudium in Dortmund

1.Semester: Programmierung I, lineare Algebra I, Elektrotechnische Grundlagen für InformatikerInnen

2.Semester: Rechnerstrukturen, lineare Algebra II, Elektronik für InformatikerInnen

Klausur Informatik I: Programmierung I und Rechnerstrukturen (Voraussetzung für das Software-Praktikum), 3 Stunden

vorlesungsfreie Zeit: Hardware-Praktikum, Programmierkurs

3.Semester: Datenstrukturen, Informatik und Gesellschaft, Analysis I für InformatikerInnen

vorlesungsfreie Zeit: Software-Praktikum

4.Semester: Grundlagen theoretischer Informatik, Wahrscheinlichkeitsrechnung und mathematische Statistik für InformatikerInnen

mündliche Prüfung Informatik II: Datenstrukturen und theoretische Informatik

Vordiplom: Informatik I, Informatik II, Mathematik, Nebenfach und Bescheinigungen über Software- und Hardware-Praktikum sowie Informatik und Gesellschaft.

Hauptstudium: 2 theoretische Vorlesungen, 2 praktische Vorlesungen, Spezialvorlesungen, Seminar und Projektgruppe. 3 Prüfungen bestehen und dann

Diplomarbeit schreiben und **fertig!**

Die Vorlesung Programmierung I - Ziele

Grundsätzlich ist die Informatik jedoch die Wissenschaft der Abstraktion – das richtige Modell für ein Problem zu entwerfen und die angemessene mechanisierbare Technik zu ersinnen, um es zu lösen. (Aho, Ullmann 1996)

Was soll formalisiert werden und wie greift die Formalisierung in den Prozeß ein?

- Aufgabenbeschreibung
- Modell, Architektur

Wie kann ich formalisieren?

- Umsetzungen finden
  was: Datenmodell
  wie: Datenstruktur
  warum: Eigenschaften der Umsetzung
- Umsetzung realisieren

Warum ist das eine angemessene Formalisierung? Welche Eigenschaften hat sie?

- Analysieren, Diskutieren, Experimentieren

Die Vorlesung Programmierung I - Vorgehen

1. Objektorientierte Modellierung
2. JAVA – erste Schritte
3. Umsetzungen für Standardmodelle:
   - abstrakte Datentypen und ihre Realisierung in JAVA
   - Induktionsbeweis und Performanztest
4. Ereignisbehandlung und graphische Oberflächen
5. Nebenläufige Programmierung
6. Verteilte Programmierung

Programmierung – Paradigmen

- Imperative Programmierung (Basic, Pascal)
- Funktionale Programmierung (LISP, ML)
- Objekt-orientierte Programmierung (JAVA, C++)
- Logische Programmierung (Prolog)

Programmierung im Großen

Programmierung im Großen Die Konzentration auf das Zusammenwirken von Teilen nennt man Programmierung im Großen, weil hier von der inneren Gestalt der Teile abstrahiert wird. Betrachtet wird das Außenverhalten von Teilen. Der englische Terminus ist *programming in the large*.

Bei der Programmierung im Großen werden Arbeitsabläufe betrachtet, eine Architektur für die Software entworfen, die Mengen von Aufgaben zu Modulen anordnet und die Beziehungen zwischen den Modulen angibt.

Programmierung im Kleinen

Programmierung im Kleinen Die Ausformulierung von Teilen nennt man Programmierung im Kleinen, weil auf die Einzelheiten eines Teils geachtet wird. Betrachtet wird die interne Realisierung eines Teils. Der englische Terminus ist *programming in the small*.

Beurteilungskriterien von Programmen

Arbeitsablauf: Wird der Arbeitsablauf durch das Programm besser und für die Menschen angenehmer? Welche Ziele der an dem Arbeitsablauf beteiligten Menschen werden in welchen Anteilen erfüllt, welche nicht? → *Informatik und Gesellschaft*.

Wartbarkeit: Große Programme müssen gewartet werden. Zum einen, weil die Welt sich ändert, in der die Programme eingesetzt werden und die sie in Teilen abbilden. Zum anderen, weil man immer einen Fehler macht, etwas übersieht, wenn man ein Programm entwickelt. → *Softwaretechnologie*, *Künstliche Intelligenz*

Wiederverwendbarkeit: nicht immer von vorn anfangen, sondern Teile veralteter Programme in den neuen Programmen verwenden!

Effektivität: Sind alle Fälle, die vorkommen können, abgedeckt?(*Vollständigkeit*) Berechnen wir immer das richtige Ergebnis? (*Korrektheit*) → *Programmverifikation*.

Effizienz: Innerhalb der *Komplexitätstheorie* wird die Schwierigkeit von Problemen untersucht: wie lange wird der Rechner uns im schlimmsten Falle auf ein Ergebnis warten lassen? Zusätzlich empirische Überprüfung durch systematische Experimente: wie lang braucht das Programm unter bestimmten, systematisch variierten Umständen, bis es ein Ergebnis liefert?

Objektorientierte Modellierung – Objekte

Wir fassen eine Aufgabe oder ein System als eine Menge miteinander kooperierender Einheiten (Objekte) auf. Ein Objekt ist ein Ding mit einer Tätigkeit: der Fußball, der rollt, meine Lampe, die leuchtet.

> Zum Beispiel sind historische Ereignisse, materielle Objekte, Menschen und deren Schatten nach meinem wie nach den gängigsten Arten philosophischen Sprachgebrauchs sämtlich Einzeldinge; Eigenschaften, Zahlen und Gattungen dagegen nicht... (Strawson, 1959)

Objekte sind voneinander verschieden, auch wenn ihre Beschreibung es nicht deutlich macht. Zahlen sind deshalb keine Einzeldinge, weil sie stets mit sich selbst gleich sind: es gibt nur eine 1, egal wo, wofür und wie oft wir sie verwenden.

Objektorientierte Modellierung – Attribute

Objekte haben *Eigenschaften*, die wir angeben können: Utas Ball ist blau, meine Schreibtischlampe ist weiß.

Wir unterscheiden, *daß* ein Objekt eine Eigenschaft hat – z.B. eine Farbe – davon, *welche Ausprägung* der Eigenschaft es hat – z.B. blau.

Objekte können etwas tun oder auf Tätigkeiten reagieren: Utas Ball rollt, wenn sie ihn tritt, meine Lampe beleuchtet den Schreibtisch, wenn ich sie einschalte.

Wir betrachten den Tritt gegen den Ball als eine *Botschaft* an den Ball. Es wird ihm mitgeteilt, mit welcher Kraft, an welcher Stelle er getreten wird. Der Ball hat eine *Methode*, wie er auf eine Botschaft reagiert: er rollt in eine bestimmte Richtung eine bestimmte Strecke.

Objektorientierte Modellierung – Klassen

Objekte sind Exemplare (Beispiele, Instanzen) einer Klasse. Utas blauer Ball *ist ein* Ball, meine Schreibtischlampe *ist eine* Lampe.

Alle Objekte einer Klasse haben die Eigenschaften und Methoden dieser Klasse. *Daß* ein Ball eine Farbe hat wird durch die Klasse festgelegt.

Wenn ein neues Objekt einer Klasse erzeugt wird (Instanziierung), dann bekommt es alle Eigenschaften und eventuell einige Ausprägungen der Eigenschaften.

Skript für die Vorlesung

Das Skript gibt die wichtigsten Definitionen in einheitlicher Terminologie wieder; zeigt durch den Index und die Abschnitte *Was wissen Sie jetzt?*, was Studierende wissen müssen, um erfolgreich weiterstudieren zu können.

Das Skript bekommen Sie:

- Im Skriptenverkauf: Emil-Figge-Straße 50, Erdgeschoß (gegenüber der Cafeteria)
- Im Skriptendruck der IRB: im WWW-Bereich der IRB unter *Mitschriften* einfach in der Tabelle die gewünschte Mitschrift anklicken
- Auf der CD: wer einen Postscript-fähigen Drucker hat, druckt einfach das Skript aus.

Bücher, die ich empfehle

Flanagan, David (1998): JAVA in a Nutshell, deutsche Ausgabe für JAVA 1.1, O'Reilly

Gosling, James, **Joy**, Bill, **Steele**,Guy (1997): JAVA – Die Sprach-spezifikation, deutsche Ausgabe, Addison-Wesley

Bishop, Judy (1998): JAVA Gently – Programming Principles Explained, 2nd edition, Addison-Wesley

Campione, Mary, **Walrath**, Kathy(1998): The JAVA Tutorial, 2nd edition, Addison-Wesley

Goos, Gerhard (1996): Vorlesungen über Informatik Band 2 – Objektorientiertes Programmieren und Algorithmen, Springer

Dißmann, Stefan, **Doberkat**, Ernst-Erich (demnächst): Einführung in die objektorientierte Programmierung mit JAVA

Bibliotheken

Bücher zur Programmierung, JAVA, objektorientierten Modellierung finden Sie

- in der Bereichsbibliothek: Pavillon 6, Campus Süd
- in der Universitätsbibliothek: Campus Nord
- in Buchhandlungen:

  UniBuch in der Mensa, Campus Nord

  Schaten, bei der S-Bahn, Campus Nord

  Krüger, Westenhellweg, Innenstadt

Übungen

Übungen sind zwingend erforderlich, um sich den Stoff anzueignen!

Tutorium für ausländische Studierende

girls and boys learn better with girls

Erstsemester-CD

enthält JAVA für Linux, MacIntosh, Windows, nützliche Programme, Informationen von Studierenden für Studierende. Ich habe für Sie vorbereitet:

1. Programme, die Standardumsetzungen darstellen
2. Programme, die die Verwendung solcher Umsetzungen illustrieren
3. ganz winzige Programme, die auch im Skript vorkommen
4. ein etwas umfangreicheres Beispiel

CD

- Erst einmal nur die allgemeinen Informationen lesen und
- das Skript besorgen.
- Wenn Sie in der nächste Woche daran gehen wollen, Ihren eigenen Rechner fit für JAVA zu machen, beachten Sie stets:

  README heißen Texte, die Sie lesen müssen, **bevor** Sie etwas anderes tun.
- Alle Unterlagen sind an der Universität über Ihren Rechnerarbeitsplatz im Pizza-Pool erreichbar – Sie müssen keinen eigenen Rechner mit CD- Laufwerk besitzen!!!