Informatik II

Felix Friedrich

Vorlesung am D-BAUG der ETH Zürich

FS 2016

W i l l k o m m e n

zur Vorlesung Informatik II !

am D-BAUG der ETH Zürich.

Ort und Zeit:

- Montag 12:45– 14:30, HIL E3.
- Pause 13:30 – 13:45, leichte Verschiebung möglich

- **Vorlesungshomepage** http://lec.inf.ethz.ch/baug/informatik2/2016/

Team

Dozent	Felix Friedrich
Chefassistent	David Sidler
Assistenten	Giuseppe Accaputo
	Andrin Kasper
	Marko Pichler
	Tobias Verhulst

Überblick, der universelle Computer (Turing Maschine), Euklidischer Algorithmus, Erstes Java Programm, Java Klassen, Pascal → Java, der Euklidische Algorithmus in Java

1. EINFÜHRUNG

Ziel der Vorlesung

- Hauptziel: **Problemlösen mit dem Computer**

- Hilfsmittel
  - Objektorientierte Programmiersprache (Java)
  - Tools wie Matlab und Datenbanken
- Methodik
  - *Kernthemen:* Objektorientiertes Programmieren, Datenstrukturen, Algorithmen und Komplexität, Datenbanken
  - *Fallstudien:* Interessante Probleme aus der Informatik und angrenzenden Gebieten

Warum Java?

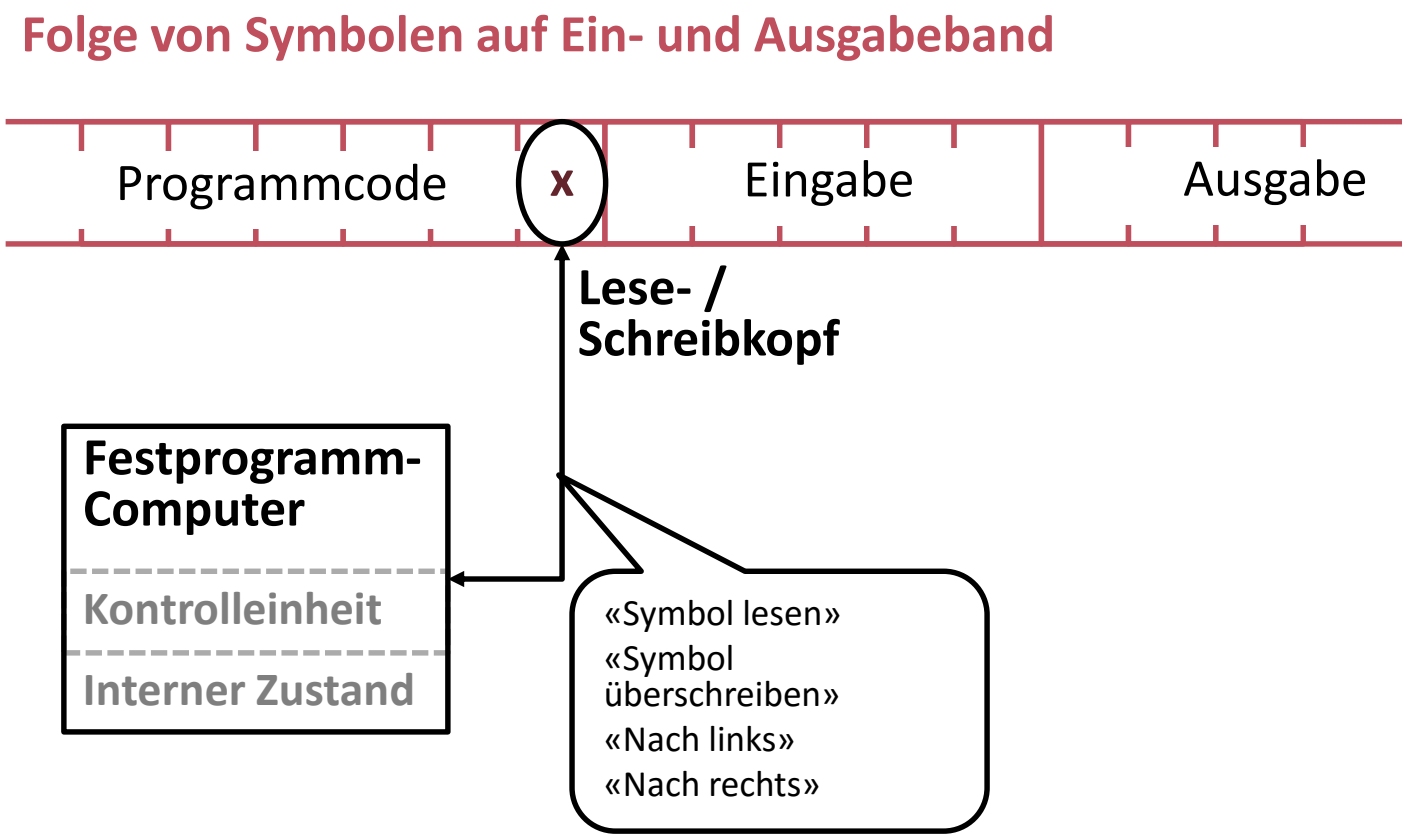
- Sehr weit verbreitete moderne Sprache, funktioniert auf sehr vielen Systemen
  - Portabilität durch Zwischensprache
- Verbietet einige typische Fehler
- Gute Datenbankanbindung

Ziel der heutigen Vorlesung

- Einführung: Computermodell und Algorithmus
- Prozedurales Programmieren mit Java, Pascal → Java

Der universelle Computer

- Eine geniale Idee: Universelle Turing Maschine (Alan Turing, 1936)



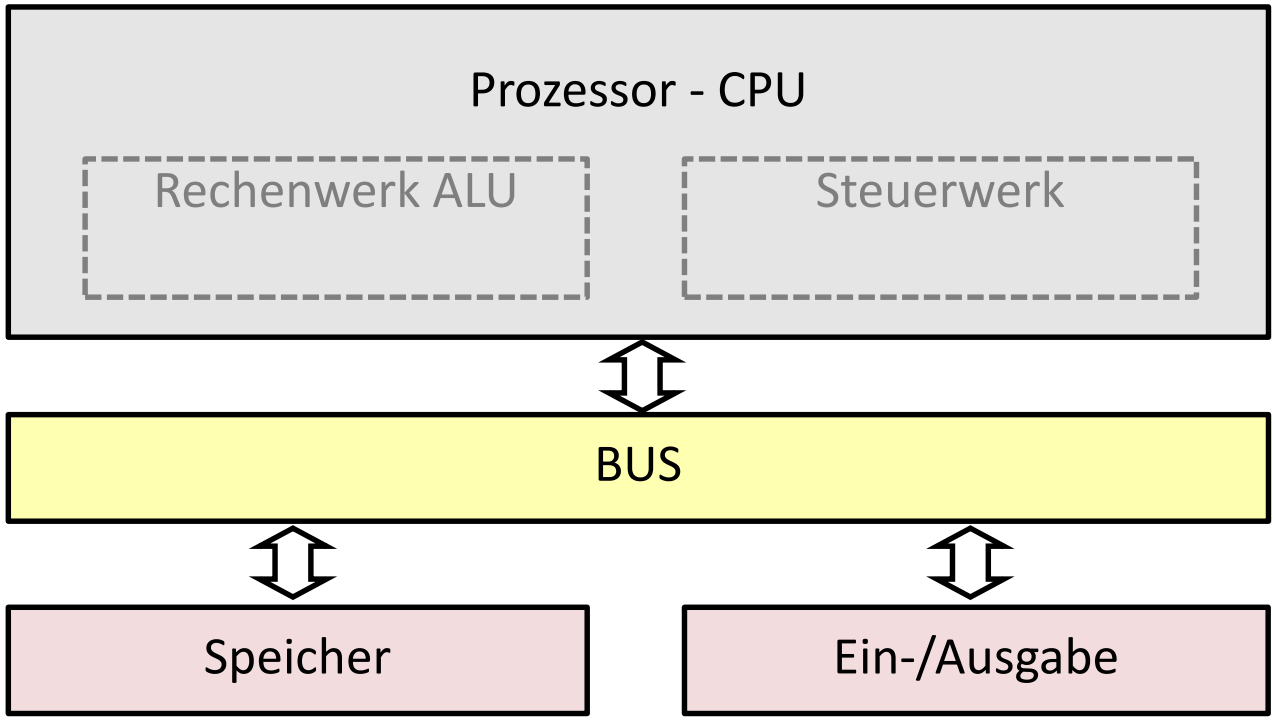
Alan Turing

http://en.wikipedia.org/wiki/Alan_Turing

Realisierungen

- Z1 – Konrad Zuse (1938)
  ENIAC – Von Neumann (1945)

Von Neumann Architektur



/http://www.hs.uni-hamburg.de/DE/GNT/hh/biogr/zuse.htm

Konrad Zuse

http://commons.wikimedia.org/wiki/File:John_von_Neumann.jpg

John von Neumann

Universelles Rechenmodell

- Zutaten der *Von Neumann Architektur*
- Hauptspeicher (RAM) für Programme *und* Daten
- Prozessor (CPU) zur Verarbeitung der Programme und Daten
- I/O Komponenten zur Kommunikation mit der Aussenwelt

Ältester nichttrivialer Algorithmus

Euklidischer Algorithmus (3. Jh v. Chr.) Grösster gemeinsamer Teiler

Annahme: a>0, b>0.

Solange $b \neq 0$

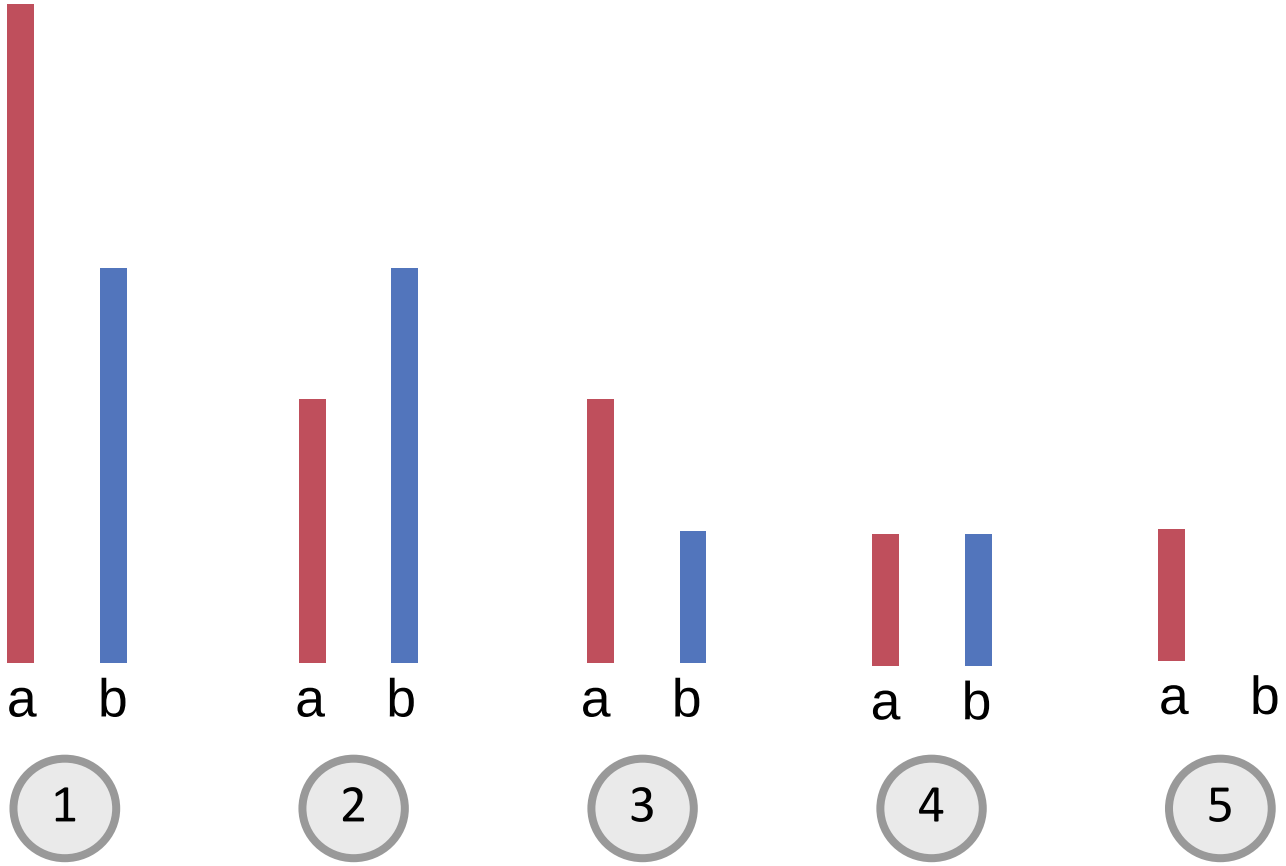
Wenn a $>$ b

dann $a \leftarrow a - b$

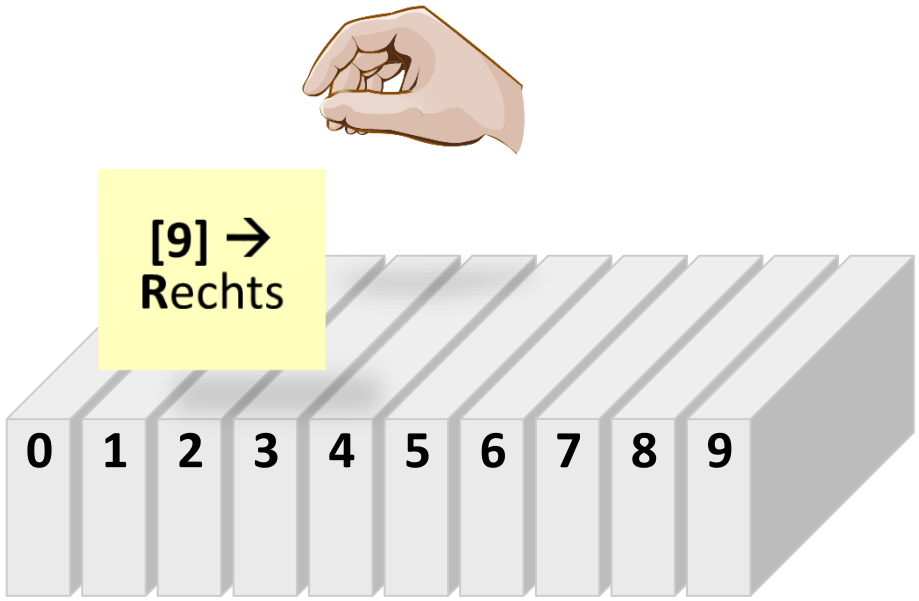
Sonst

$b \leftarrow b - a$

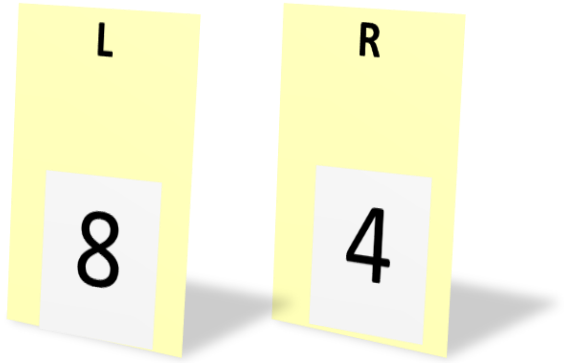
Ergebnis = a.



Live Demo: Turing Maschine



Speicher



Register

Euklid in der Box

Speicher

0	1	2	3	4	5	6	7	8	9
[8] → **L**inks	**[9] →** **R**echts	**L**=0? stop	**R**>**L**? nach 6	**L**-**R** **→[8]**	nach 0	**R**-**L** **→[9]**	nach 0	**(b)**	**(a)**

Register

Links	Rechts

Solange $b \neq 0$

Wenn a > b

dann $a \leftarrow a - b$

Sonst

$b \leftarrow b - a$

Geschwindigkeit

- In der mittleren Zeit, die der Schall von mir zu
- Ihnen unterwegs ist...

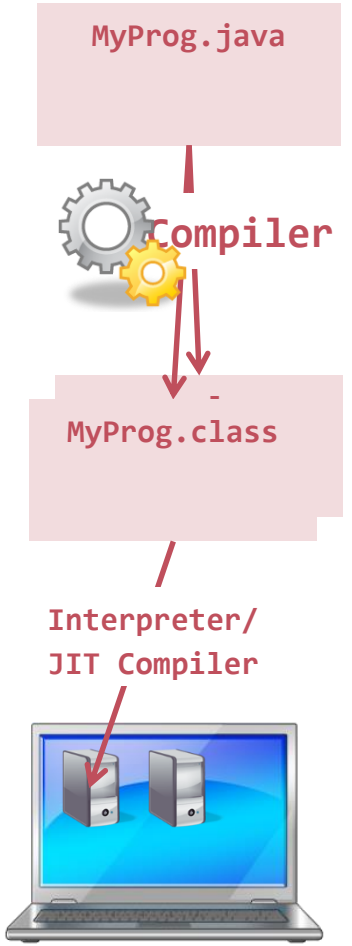
30 m

≙ mehr als 100.000.000 Instruktionen

- arbeitet ein heutiger Desktop-PC
- mehr als 100 Millionen Instruktionen ab.

Java

- Basiert auf einer *virtuellen Maschine*,
  (mit Von-Neumann Architektur)
  - Programmcode wird in Zwischencode übersetzt,
  - Zwischencode läuft in "simulierter" Rechnerumgebung.
    Interpretation des Zwischencodes durch einen sog. Interpreter
  - Optimierung: JIT (Just-In-Time) Kompilation von häufig benutztem Code.
    virtuelle Maschine → physikalische Maschine



- Folgerung und erklärtes Ziel von Java
  - Portabilität (Write Once Run Anywhere)

ORGANISATORISCHES

Übungen

- Übungszeiten / Orte

Donnerstag	**12:45 – 14:30**	Giuseppe Accaputo	**HIT F 31.1**
		Marko Pichler	**HIT F 12**
		Andrin Kasper	**HIT F 13**
	14:45– 16:30	Tobias Verhulst	**HCI D4**

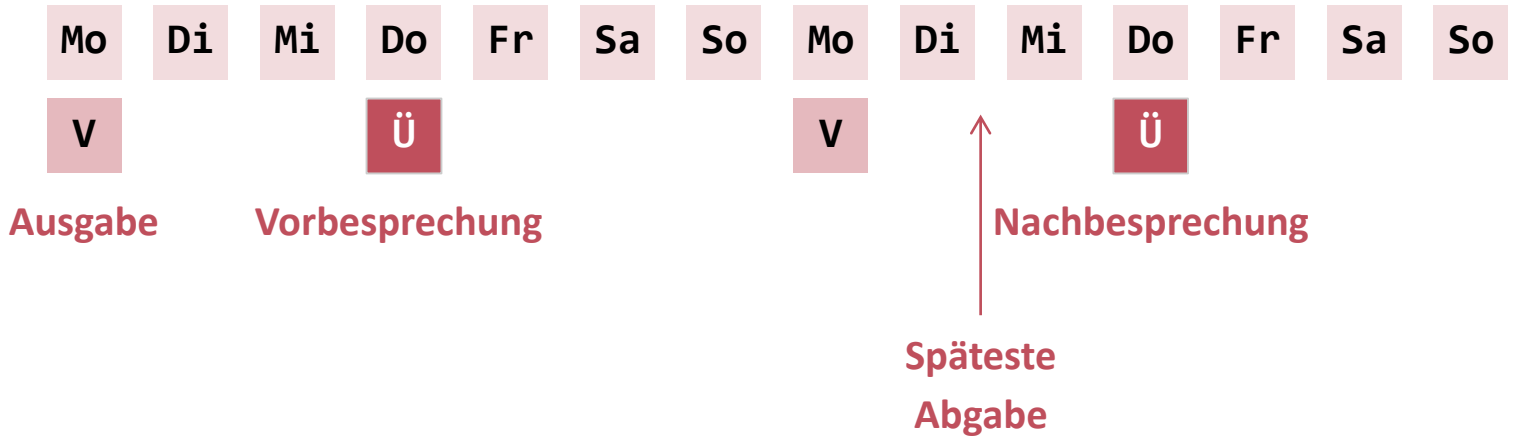
- Gruppeneinteilung selbstständig via Webseite

  E-Mail mit Link nur nach Belegung dieser Vorlesung in myStudies

  Genau eine E-Mail mit personalisiertem Link ➔Link behalten !

Übungen Ablauf

- Übungsblattausgabe zur Vorlesung.
- Vorbesprechung der Übung am Donnerstag der selben Woche
- Abgabe der Übung spätestens eine Woche später (Dienstag 23:59).
  Abgabe via Autograder oder per email an den Assistenten.
- Nachbesprechung der Übung am Donnerstag darauf. Korrektur der Abgaben innerhalb einer Woche nach Nachbesprechung.



Zu den Übungen

- Seit HS 2013 für Prüfungszulassung kein Testat mehr erforderlich.
- Bearbeitung der wöchentlichen Übungsserien ist freiwillig, wird aber **dringend** empfohlen!

Autograder

- In den Übungen wird ein Tool ("DOM Judge") zur automatischen Bewertung Ihrer Lösung eingesetzt.
- Vorgehen:
  - Erstellen eines Programmes mit Hauptklasse "Main" gemäss Problemstellung
  - Einreichen des Programmes beim judge (webseite).
  - Link auf dem Übungsblatt und Vorlesungshomepage
- Judge prüft, ob Ihr Code bei Vorliegen einer bestimmten Eingabe die richtige Ausgabe (innerhalb einer vorgegebenen Maximalzeit) erzeugt.

Autograder

submissions | scoreboard | problem overview | logout

New Submission

Problem: IB1501: Hello Name
Language: Java
Timelimit: 10 sec for testcase "Name"

Solution:

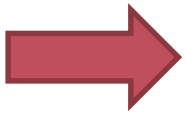
You may either cut&paste your source code or upload a file containing it.

Code:

```
class Main
{
public static void main (String[] args)
{
System.out.println("Hello you");
}
}
```

File: Browse... No file selected.

Submit solution



submissions | scoreboard | problem overview | logout

Submissions of Felix Oliver Friedrich

time	problem	lang	status
11.02. 12:17	IB1500	JAVA	PENDING
11.02. 12:14	IB1501	JAVA	WRONG-ANSWER
11.02. 10:07	IB1500	JAVA	CORRECT
11.02. 10:03	IB1500	JAVA	WRONG-ANSWER
02.02. 14:57	IB1501	JAVA	CORRECT
02.02. 14:53	IB1501	JAVA	CORRECT
02.02. 14:40	IB1501	JAVA	RUN-ERROR

Autograder

Resultate

- Richtige Ausgabe
- Falsche Ausgabe
- Übersetzungsproblem (Compilation failed)
- Laufzeitproblem (Exception u.ä.)
- Timeout

Sie können jederzeit neu einreichen bis alles funktioniert

Unser Angebot !

- Eine spezielle Programmierübung. Punkte dieser Übungen werden als *Bonus* für die Note der Basisprüfung mitgenommen.
- Termin: Vorr. in der 10. Woche. Zwei Wochen Bearbeitungszeit.

- Maximal erreichbarer Bonus: 1/4 einer Note.
- Wir behalten uns mündliche Prüfgespräche für die geleisteten Übungen vor.

Tipps

- Üben Sie!
- Wenn Sie zu zweit an Übungen arbeiten, stellen Sie Gleichverteilung des aktiven Parts sicher.
- Lassen Sie sich nicht frustrieren. Schlafen Sie eine Nacht, wenn Sie nicht vorwärtskommen.
- Holen Sie sich Hilfe, wenn Sie nicht vorwärtskommen.
- Üben Sie!

Computerarbeitsplätze

- In den Computerräumen des D BAUG ist die nötige Software (Java + Eclipse) installiert, genauso auch im Hauptgebäude im Zentrum.
- Falls Sie auf Ihrem eigenen Rechner arbeiten möchten: Installationsanweisung für Java / Eclipse auf dem ersten Übungsblatt

Bringen Sie Ihren Computer doch bitte zur ersten Übungsstunde mit.

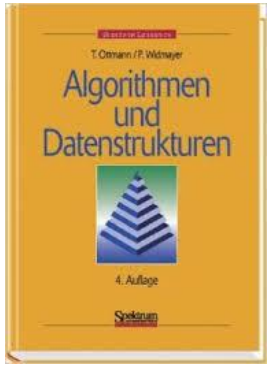
Literatur

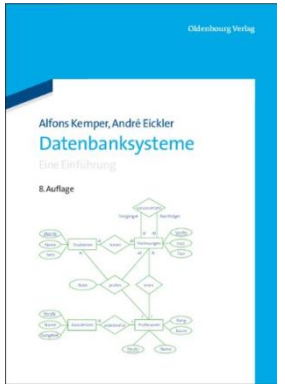
- Robert Sedgewick, Kevin Wayne, **Einführung in die Programmierung mit Java**. Pearson, 2011

  Thomas Ottmann, Peter Widmayer, **Algorithmen und Datenstrukturen**, Springer pdf derzeit auch als download bei Springer

- Kemper, Eickler: **Datenbanksysteme: Eine Einführung**. Oldenbourg Verlag, 9. Auflage, 2013.

- Weitere online-Referenzen auf der Vorlesungshomepage.





Relevantes für die Prüfung

Prüfungsstoff für die Endprüfung schliesst ein

- Vorlesungsinhalt und
- Übungsinhalte.

Prüfung (120 min) ist schriftlich.

Zugelassene Hilfsmittel: keine

Programmierkenntnisse sind Minimalvoraussetzung zum Lösen der Prüfung.

In Ihrem und unserem Interesse ...

wichtig !

Bitte melden sie *frühzeitig*, wenn Sie Probleme sehen, wenn Ihnen

- die Vorlesung zu schnell, zu schwierig, zu .... ist
- die Übungen nicht machbar sind ...
- Sie sich nicht gut betreut fühlen ...

Kurz: wenn Ihnen irgendetwas auf dem Herzen liegt

JAVA

Erstes Pascal Programm

```
PROGRAM Hello;
BEGIN
   Write('Hello World.');
END.
```

- Aufruf: Hello

Erstes Java Programm

```
public class Hello {

  public static void main(String[] args)  {
    System.out.println("Hello World.");
  }
}
```

Aufruf: `java Hello`

class: ein Programm

Methode: Eine (benannte) Folge von Anweisungen

Anweisung: ein Kommando, welches auszuführen ist

Ein Funktionsaufruf einer Methode (Prozedur) einer anderen Klasse

Java Klassen

Java-Programm besteht aus mindestens einer *Klasse*.

Ein Java-Programm hat eine Klasse mit **`main`**-Funktion (Methode). Diese spielt die Rolle des Programmrumpfes bei Pascal

```
public class Test{

	// potentiell weiterer Code und Daten

	public static void main(String[] args) {
		...
	}
}
```

Prozedurale Programmierung mit Java

Pascal → Java

Deklarationen, Ausdrücke und Anweisungen: separate Tabelle

Programm	→ *Klasse* mit `public static void main`	
Prozedur	→ `static` *Methode* in dieser Klasse	
Globale Variablen	→ `static` Variablen in dieser Klasse	
Records	→ Klassen	(Achtung: Referenzsemantik*)
Arrays	→ Arrays	(Achtung: Referenzsemantik*)
Var-Parameter	→ *es gibt in Java kein Pass-by-Reference!*	

*wird noch erklärt

Ein Java-Programm entsteht

Programm schreiben [z.B. im eclipse editor]

- Quellcode

Programm compilieren [z.B. in der eclipse *IDE*]

- Quellcode wird zu Bytecode übersetzt
- Passiert in der eclipse IDE permanent im Hintergrund

Ausführen [z.B. in der IDE oder an der Kommandozeile]

- Bytecode wird interpretiert und teilweise zu Maschinencode übersetzt und ausgeführt

Der Euklidische Algorithmus in Java

```
public class Euclidean {

   public static void main(String[] args){
      int a = 24;
      int b = 20;
      while (b != 0) {
         if (a>b)
            a = a - b;
         else
            b = b - a;
      }

      System.out.println("ggt(24,20)= " + a);
   }
}
```

Variablen müssen vor Gebrauch deklariert werden. Aber: keine separate Deklarations-sequenz wie bei Pascal nötig.

Blöcke werden durch geschweifte Klammern gekennzeichnet

Bedingungen stehen bei if und while immer in runden Klammern.

Verhalten eines Programmes

Zur Compilationszeit:

- vom Compiler akzeptiertes Programm – syntaktisch korrekt*
- Compiler Fehler – manchmal nicht einfach zu verstehen

Zur Laufzeit:

- korrektes Resultat
- inkorrektes Resultat
- Programmabsturz
- Programm terminiert nicht

*Compiler prüfen auch einen Teil der Semantik

Der Euklidische Algorithmus in Java

```
public class Euclidean {

   public static void main(String[] args){
      int a = 24;
      int b = 20;
      while (b != 0) {
         if (a>b)
            a = a - b;
         else
            b = b - a;
      }
      System.out.println("ggt(24,20)=" + a);
   }
}
```

Ist das ein gutes Programm?

Der Euklidische Algorithmus in Java

```
public class Euclidean {

   public static void main(String[] args){
      int a = 24;
      int b = 20;

      while (b != 0) {
         int h = a % b;
         a = b;
         b = h;
      }

      System.out.println("ggt(24,20)=" + a);
   }
}
```

Ok, schon etwas besser.

Vergleich mit Pascal

```java
public class Euclidean
{

public static void main(String[] args)
   {
      int a = 24;
      int b = 20;
      while (b != 0) {
         int h = a % b;
         a = b;
         b = h;
      }
      System.out.println("ggt(24,20)= " + a);
   }
}
```

```
program Euklid;

var a, b, rest: integer;

begin
   a := 24;
   b := 20;
   while b <> 0 do begin
      Rest:=a mod b;
      a:=b;
      b:=rest;
   end;

   writeln('ggt(24,20)=',a:5);

end.
```

Euklid in einer Funktion (Methode)

```
public class Euclidean {

   // PRE: a, b >= 0
   // POST: gibt GGT(a,b) zurück
   static int ggt(int a, int b){
      while (b != 0) {
         int h = a % b;
         a = b;
         b = h;
      }
      return a;
   }

   public static void main(String[] args){
      System.out.println("ggt(24,20)= " + ggt(24,20));
   }
}
```

Noch besser

Ein- und Ausgabe

Ausgabe via `System.out` mit

```
System.out.print(...);
System.out.println(...);
```

Eingabe via `System.in` mit Scanner

```
Scanner input = new Scanner(System.in);
        String s = input.next();
        int i = input.nextInt();
```

benötigt `import java.util.Scanner;`

Euklidischer Algorithmus mit Eingabe

```
import java.util.Scanner;

public class Euclidean {
   static int ggt(int a, int b){
      while (b != 0) {
         int h = a % b;
         a = b;
         b = h;
      }
      return a;
   }

   public static void main(String[] args){
      Scanner input = new Scanner(System.in);
      int a = input.nextInt();
      int b = input.nextInt();
      System.out.println("ggt(" + a + "," + b + ")= " + ggt(a,b));
   }
}
```

Zeichenketten (Strings)

Strings sind *Objekte* in Java.

Zuweisung eines *Stringliterals*:

```
String hello = "Hallo Leute";
```

Stringlänge:

```
int len = hello.length();
```

int length():
Funktionalität des Typs `String`, hier bezogen auf die Instanz `hello`.

Elementzugriff

```
char c = hello.charAt(5);
```

Verkettung

```
String helloLong = hello + ". Alles wird gut.";
```