ALPEN-ADRIA UNIVERSITÄT
KLAGENFURT | WIEN GRAZ

Einführung in die strukturierte und objektbasierte Programmierung (620.200, »ESOP«)

Assoc. Prof. Dr. Mathias Lux
ITEC / AAU

English slides on
http://www.itec.uni-klu.ac.at/~mlux/index.php?id=courses/esop16

ALPEN-ADRIA UNIVERSITÄT
KLAGENFURT | WIEN GRAZ

MENTORING-PROGRAMM

FACTS

- Für wen? Neue Bachelorstudierende der Angewandten Informatik und neue Informatik-Lehramtsstudierende.
- Freiwillig!
- Ziele:
  - Studierenden in Kleingruppen durch das erste Studienjahr begleiten,
  - einen optimalen Start ins Studium gewährleisten,
  - die Studierenden bei der Planung des Studienverlaufs zu unterstützen
- „Berührungsängste“ sollen abgebaut werden.
- Über Probleme während des Studiums kann gesprochen werden!
- Dadurch sollen die Studienbedingungen verbessert werden.

ALPEN-ADRIA UNIVERSITÄT
KLAGENFURT | WIEN GRAZ

Modalitäten

- Wir sind hier in der Vorlesung
- Gegliedert in zwei Teile:
  - Teil 1 ist Teil der STEOP
  - Teil 2 schließt direct an
- Prüfung Teil 1: siehe Tabelle später
- Prüfung Teil 2: = “ =
  - Teil 1, 2. Termin anschließend

Termine

- Donnerstags, 14-16 Uhr, HS C (c.t.)
  - Eventuelle Ausfälle werden bekannt gegeben

Plan im Detail

Wann		Wo	Thema
Mi, 03.10.2018	15:00 - 17:00	HS 4	Einleitung
Do, 04.10.2018	14:00 - 16:00	HS C	Einfache Programme
Fr, 05.10.2018	15:00 - 17:00	HS C	Verzweigungen und Schleifen
Di, 09.10.2018	12:00 - 14:00	HS 4	Gleitkomma, Arrays und Methoden
Do, 11.10.2018	14:00 - 16:00	HS C	*Buffer*
Do, 18.10.2018	14:00 - 16:00	HS C	*Wiederholung*
Do, 25.10.2018	14:00 - 16:00	HS C	*Tutorium*
Do, 08.11.2018	14:00 - 16:00	HS C	tba.
Di, 13.11.2018	10:00 – 11:00	HS A	Prfg. Teil 2
Di, 13.11.2018	12:30 – 14:00	HS A	Prfg. Teil 1
Do, 15.11.2018	14:00 - 16:00	HS C	Klassen, Objekte, String, Math
Do, 22.11.2018	14:00 - 16:00	HS C	Information Hiding
Do, 29.11.2018	14:00 - 16:00	HS C	Rekursion, Interfaces, Exceptions
Do, 06.12.2018	14:00 - 16:00	HS C	Pakete, Generics
Do, 13.12.2018	14:00 - 16:00	HS C	Innere Klassen, Lambda, Threads
Do, 10.01.2019	14:00 - 16:00	HS C	*Buffer*
Do, 17.01.2019	14:00 - 16:00	HS C	*Wiederholung*
Do, 24.01.2019	14:00 - 16:00	HS C	*Tutorium*
Do, 31.01.2019	15:00 – 17:00	HS A	Prfg. Teil 1+2

Übung

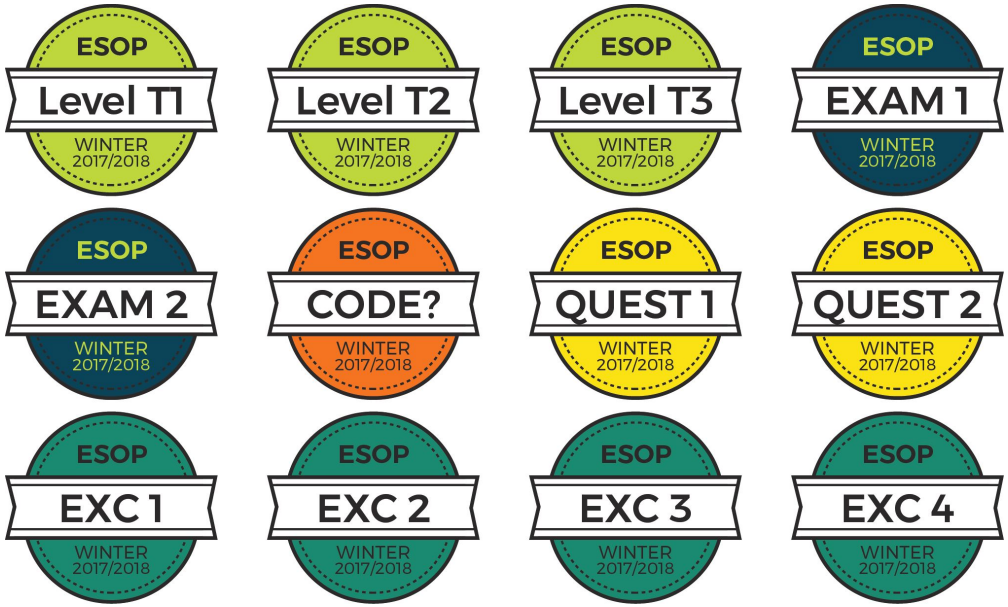
- Übung siehe ZEUS
  - Laptop mitbringen falls vorhanden!

Tutorium

- Bitte anmelden! - Egal welches der Tutorien
- Tutoriumstermine
  - Montag, 18-20 Uhr im E.2.42
  - Donnerstag, 18-20 Uhr im E.2.37 (oder 18:30)
- Code Review Team
  - Einsendungen per Email esop.code.review@itec.aau.at
  - Review durch das Tutorenteam
- Prüfungsvorbereitung für PR

Vertikale Achievements

- Zusätzliche extrinsische Motivation
  - Max. 12 zusätzliche Punkte für die VO-Prüfung
  - Pro Badge (Auszeichnung) 1 Punkt



Badges

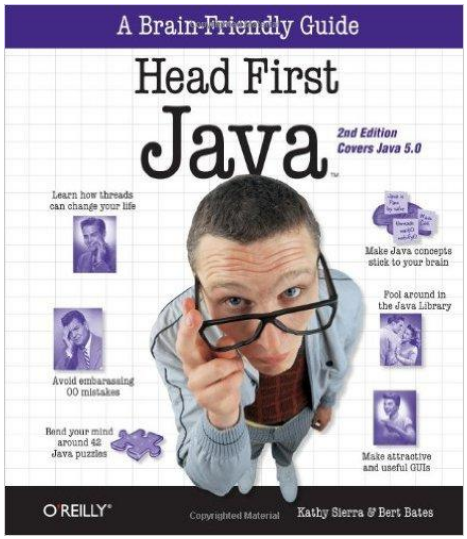
- Level T1-T3: Anmeldung und Teilnahme am Tutorium
- EXAM 1-2: Teilnahme an der Prüfungsvorbereitung
- CODE?: Einsendung an das Code-Review-Team
- QUEST 1-2: Fragen in der VO stellen und beantworten
- EXC 1-4: Optionale Exercises in Moodle lösen

Literatur

Hanspeter Mössenböck, *Sprechen Sie Java? Eine Einführung in das systematische Programmieren*
5. Auflage, dpunkt.verlag, 2014
ISBN 978-3-86490-099-0

Literatur

- Kathy Sierra, Bert Bates (2005) Head First Java (Englisch) Taschenbuch, O'Reilly and Associates; Auflage: 2



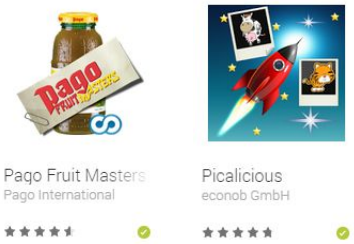
Java Dokumentation

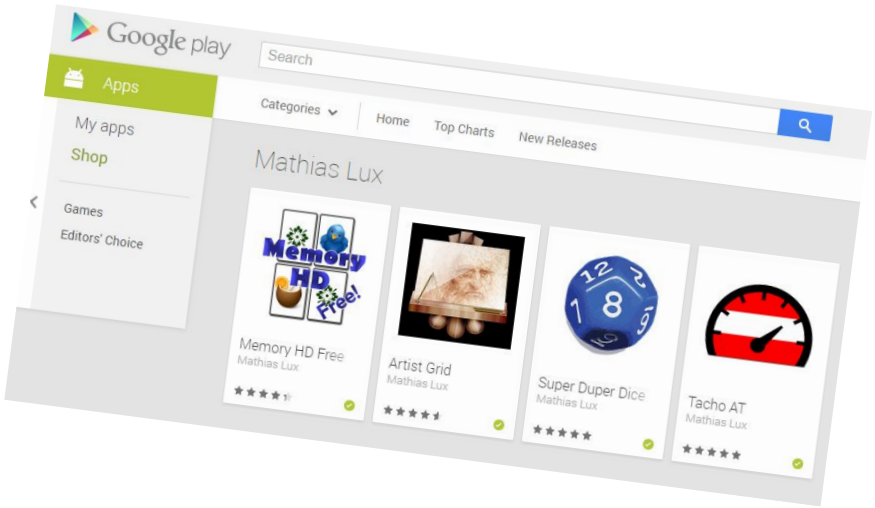
- Java API Doc
  - http://docs.oracle.com/javase/8/docs/api/
- Java Tutorials
  - http://docs.oracle.com/javase/tutorial/

Wie soll ich Java lernen?

1. Spaß am Programmieren entwickeln
2. Java Tutorials & Buch durcharbeiten
3. VO und PR besuchen

Motivation - Warum Lux?





Motivation

- Notwendigkeit in der Forschung
  - Grand Challenge Projekte
- Multimedia, zB. Processing
- Games, Apps, usw.

Was kann ich tun?

- Links auf der Homepage
  - Von visuellen Programmiersprachen bis Games

Problemdefinition

Programme schreiben besteht aus …

- Organisation
- Programmieren
- Testen

Organisation

- Rahmenbedingungen für Programmierung
  - Compiler
  - Entwicklungsumgebung
  - Bibliotheken
  - Dokumentation

Programmieren

- Klare Anweisungen
- In Programmiersprache
- Abbildung von Daten und Prozessen

Testen

- Sicherstellen dass das Programm
  - lauffähig ist,
  - Ergebnisse liefert und
  - terminiert

Was ist Programmieren?

... die Lösung eines Problems so exakt beschreiben, dass es ein Computer lösen kann

Vgl. Kochrezept, Bedienungsanleitung.

Code.org - Hour of Code

- Live Demo ...

Programmieren ist ...

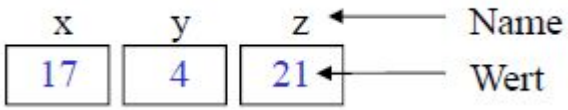
- eine kreative Tätigkeit
- eine Ingenieurstätigkeit
- schwierig, wenn man es gut machen will

Programm

Programm = Daten + Befehle

Daten

- Menge addressierbarer Speicherzellen



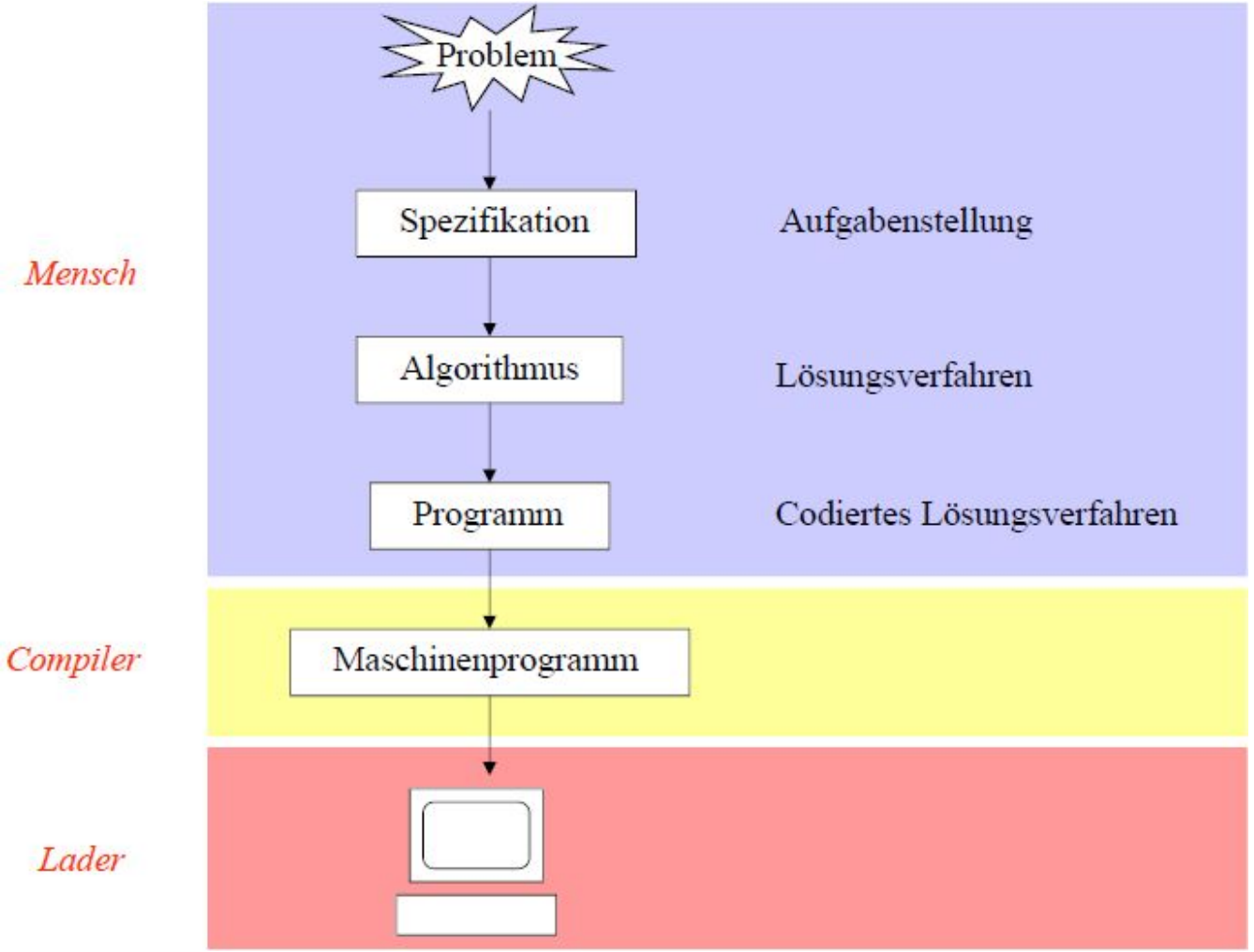
- Daten binär gespeichert (z.B. 17 = 10001)
- Binärspeicherung ist universell
  - (Zahlen, Texte, Bilder, Ton, ...)
- 1 Byte = 8 Bit
- 1 Wort = 4 Byte (typischerweise)

Befehle

- Operationen mit den Speicherzellen

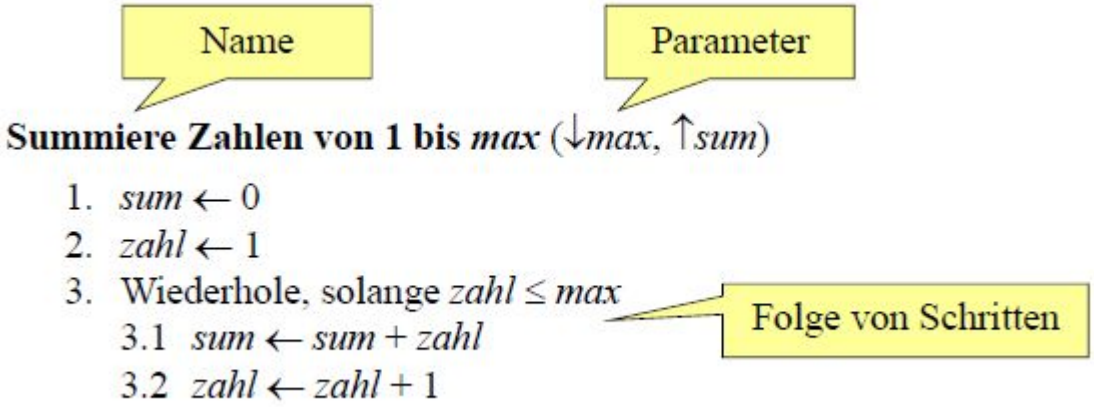
Maschinensprache		*Hochsprache*
ACC ← x	// Lade Zelle x	z = x + y;
ACC ← ACC + y	// Addiere Zelle y	
z ← ACC	// Speichere Ergebnis in Zelle z	

Programmerstellung



Algorithmus

- Schrittweises, präzises Verfahren zur Lösung eines Problems

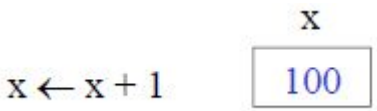


- Programm = Beschreibung eines Algorithmus in einer Programmiersprache

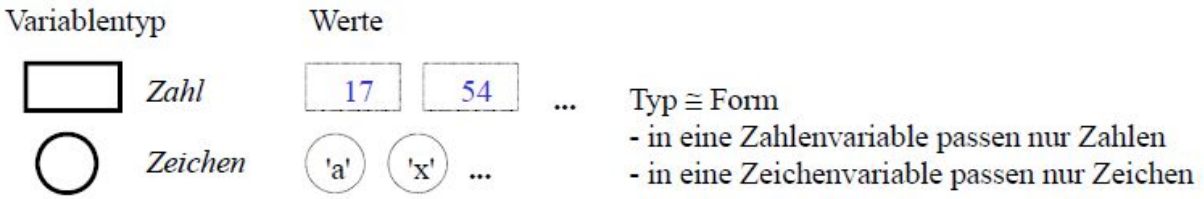
Variablen

- Sind benannte Behälter für Werte

- Können ihren Wert ändern

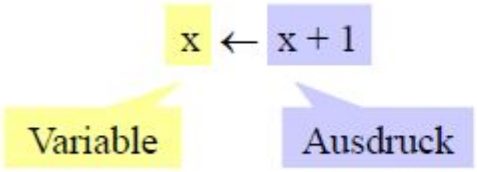


- Haben einen Datentyp
  - definiert Menge erlaubter Werte



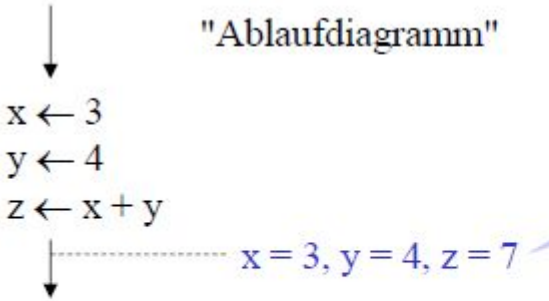
Anweisungen

- Wertzuweisung



1. werte Ausdruck aus
2. weise seinen Wert der Variablen zu

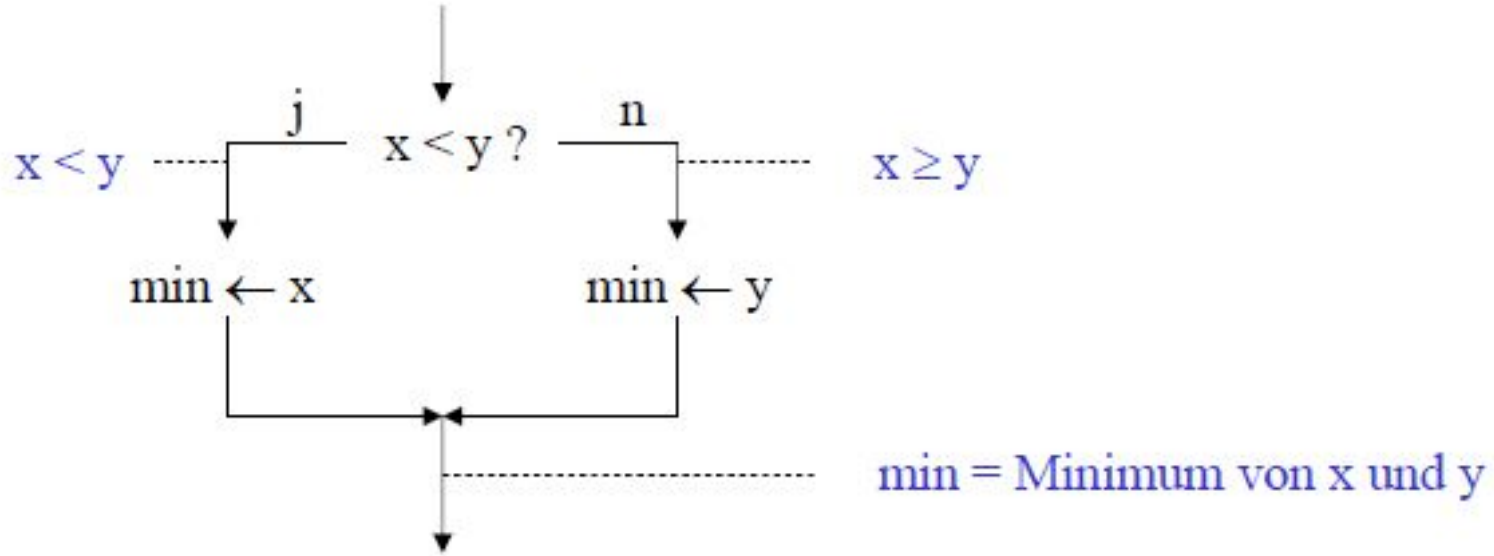
- Anweisungsfolge (Sequenz)



Assertion (Zusicherung)
Aussage über den Zustand des Algorithmus an einer bestimmten Stelle

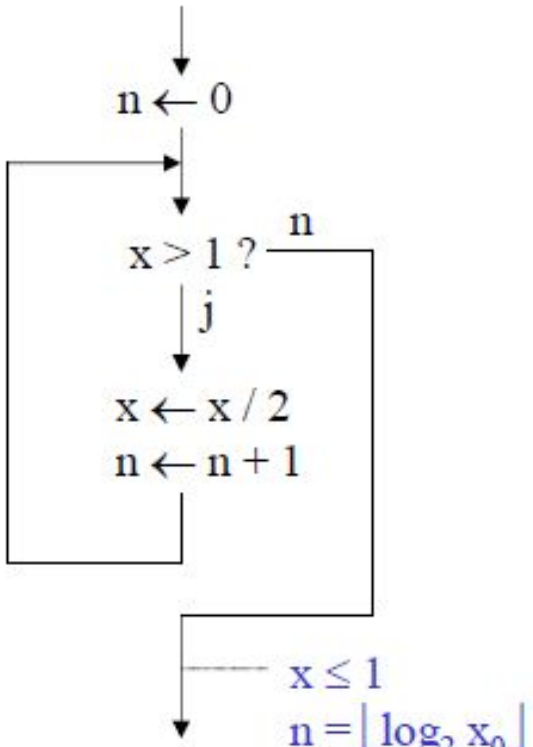
Anweisungen

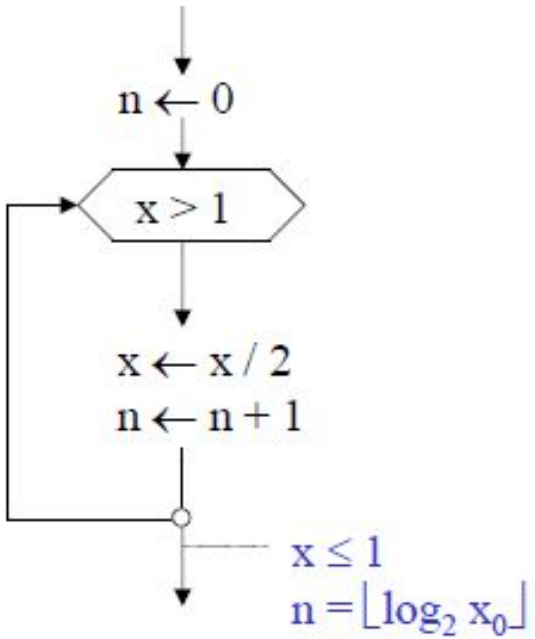
- Auswahl
  - auch Verzweigung, Abfrage, Selektion



Anweisungen

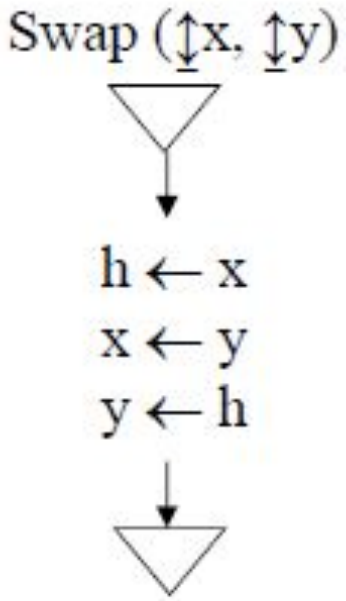
- Wiederholung
  - Auch: Schleife, Iteration





Alternative Darstellung

Beispiel: Vertauschen zweier Variablen



Schreibtischtest

x	y	h
~~3~~	~~2~~	3
2	3	

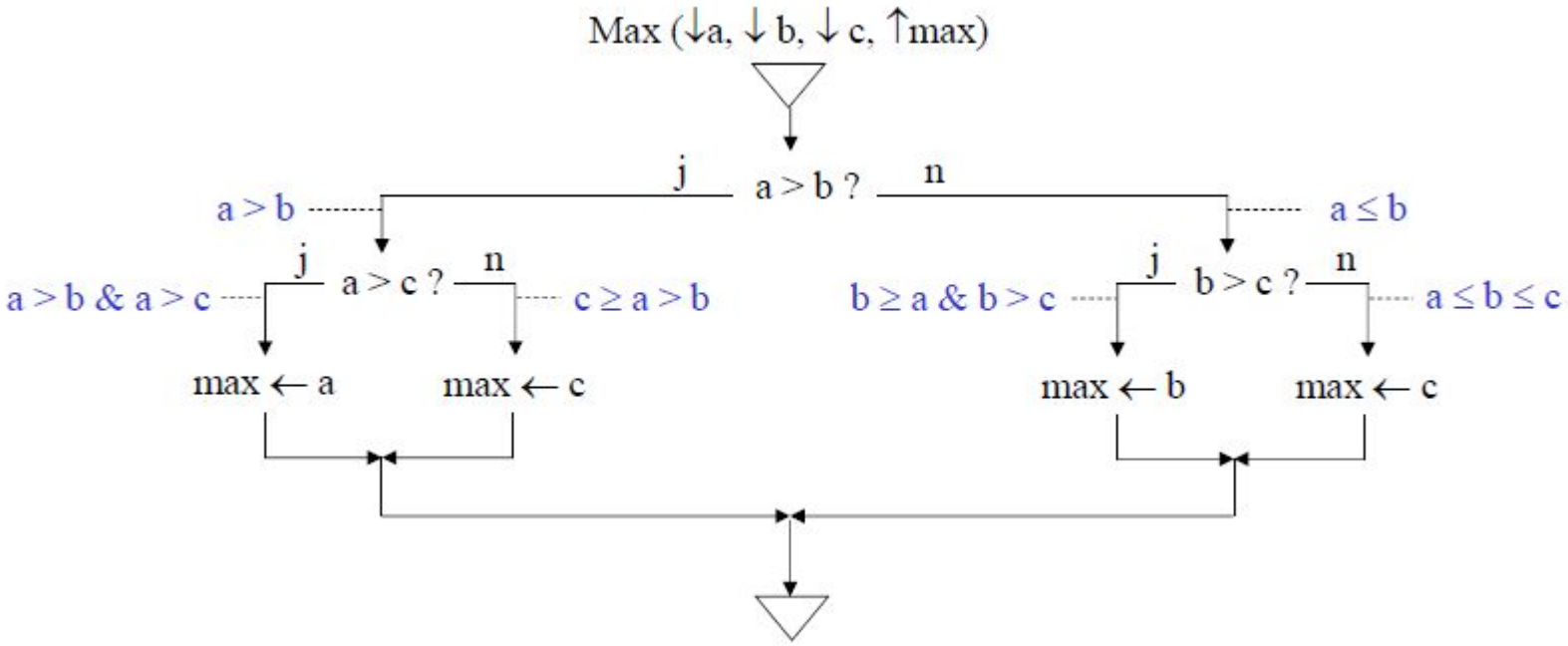
Beispiel: Vertauschen zweier Variablen

```
int x = 10;
int y = -5;
int h;

println(x);
println(y);

h = x;
x = y;
y = h;

println(x);
println(y);
```

- Source Code für Processing
- Processing ist „wie Java“

- int … Datentyp
- ; … beendet Anweisung
- println() … Funktion zur Ausgabe

Beispiel: Maximum dreier Zahlen bestimmen



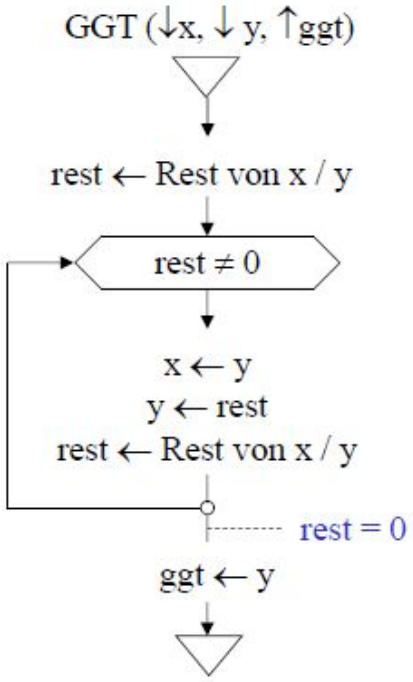
Beispiel: Maximum dreier Zahlen bestimmen

```
int a = 11;
int b = 12;
int c = 13;
int max;

if (a<b) {
  if (b<c) {
    max = c;
  } else {
    max = b;
  }
} else {
  if (a<c) {
    max = c;
  } else {
    max = a;
  }
}

println(max);
```

- Source Code für Processing
- if (test) {..}
- else {..}

Beispiel: Euklidscher Algorithmus

- Berechnet den größten gemeinsamen Teiler zweier Zahlen x und y



Schreibtischtest

x	y	rest
~~28~~	~~20~~	8
~~20~~	~~8~~	4
8	4	0

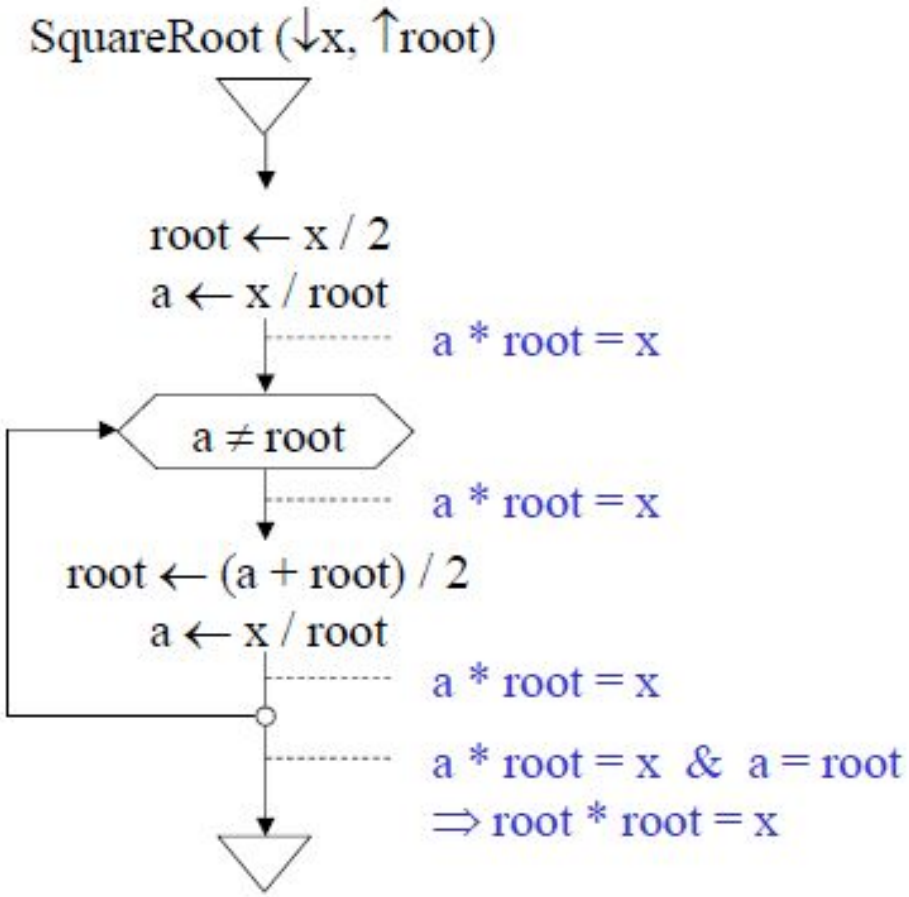
Warum funktioniert dieser Algorithmus?
(ggt teilt x) & (ggt teilt y)
⇒ x = i*ggt, y = j*ggt, (x-y) = (i-j)*ggt
⇒ ggt teilt (x - y)
⇒ ggt teilt (x - q*y)
⇒ ggt teilt rest
⇒ GGT(x, y) = GGT(y, rest)

Beispiel: Euklidscher Algorithmus

```
int x = 21;
int y = 14;

int rest = x % y;

while (rest != 0) {
   x = y;
   y = rest;
   rest = x % y;
}

println(y);
```

- Source Code für Processing
- While (test) {..}
- % … modulo

Beispiel: Quadratwurzel



Schreibtischtest

x	root	a
10	~~5~~	~~2~~
	~~3.5~~	~~2.85714~~
	~~3.17857~~	~~3.14607~~
	~~3.16232~~	~~3.16223~~
	3.16228	3.16228

Beispiel: Quadratwurzel

```
float x = 10;

float root = x / 2;
float a = x / root;

while (a != root) {
  root = (a + root) / 2;
  a = x / root;
}

println(root);
```

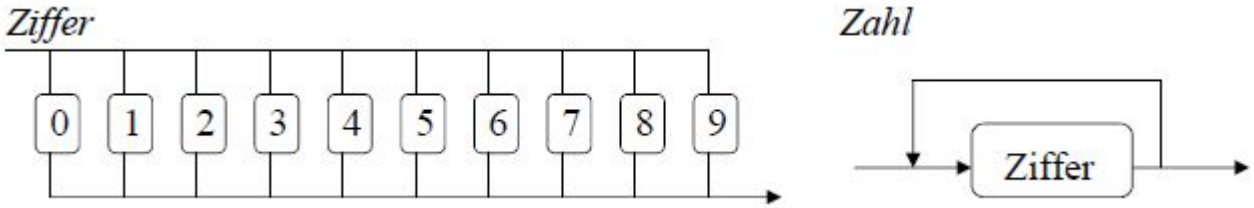
- Source Code für Processing
- float … Datentyp
- / … Division
- Tipp: float nicht auf Gleichheit prüfen!
  - |a-root| < 0,00001

Beschreibung von Programmiersprachen

- Syntax
  - Regeln, nach denen Sätze gebaut werden dürfen
  - z.B.: Zuweisung = Variable „<-" Ausdruck
- Semantik
  - Bedeutung der Sätze
  - z.B.: werte Ausdruck aus und weise ihn der Variablen zu

Beschreibung von Programmiersprachen

- Grammatik
  - Menge von Syntaxregeln
  - z.B. Grammatik der ganzen Zahlen
    - Ziffer = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9".
    - Zahl = Ziffer {Ziffer}.



EBNF (Erweiterte Backus-Naur-Form)

Beispiele

- *Grammatik der Gleitkommazahlen*
  - Zahl = Ziffer {Ziffer}.
  - Gleitkommazahl = Zahl "." Zahl ["E" ["+" | "-"] Zahl].
- *Grammatik der If-Anweisung*
  - IfAnweisung = "if" "(" Ausdruck ")" Anweisung ["else" Anweisung].

Metazeichen	Bedeutung	Beispiel	beschreibt
=	trennt Regelseiten	A = x y z .	
.	schließt Regel ab		
|	trennt Alternativen	x | y	x, y
( )	klammert Alternativen	(x | y) z	xz, yz
[ ]	wahlweises Vorkommen	[x] y	xy, y
{ }	0..n-maliges Vorkommen	{x} y	y, xy, xxy, xxxy, ...

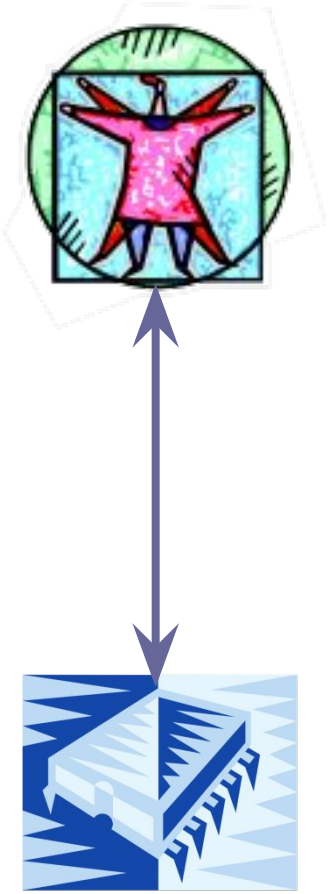
Programmiersprachen

- Maschinell (durch ein Programm) übersetzbare formale Sprachen
  - Ein Programm ist ein »Text« in einer formalen Sprache
- Viele verschiedene formale Sprachen
  - Java, Python, C, C++, Objective C, Pascal, Modula, Perl, Basic, C#, JavaScript, Dart, Erlang, LUA uvm.

Programmiersprachen

- Compiler: Programmtext wird
  - von einem Übersetzungsprogramm
  - in Maschinensprache übersetzt
  - Bsp. C, C++
- Interpreter:
  - Programmtext wird unmittelbar, schrittweise ausgeführt
  - Bsp. Python, Ruby

Algorithmennotation



Grafische Notationen	Verbale Notationen
Höhere Programmiersprachen (wie Java)	
Assemblersprachen	
Maschinencode (binär)	
Hardware (elektrische Signale)	

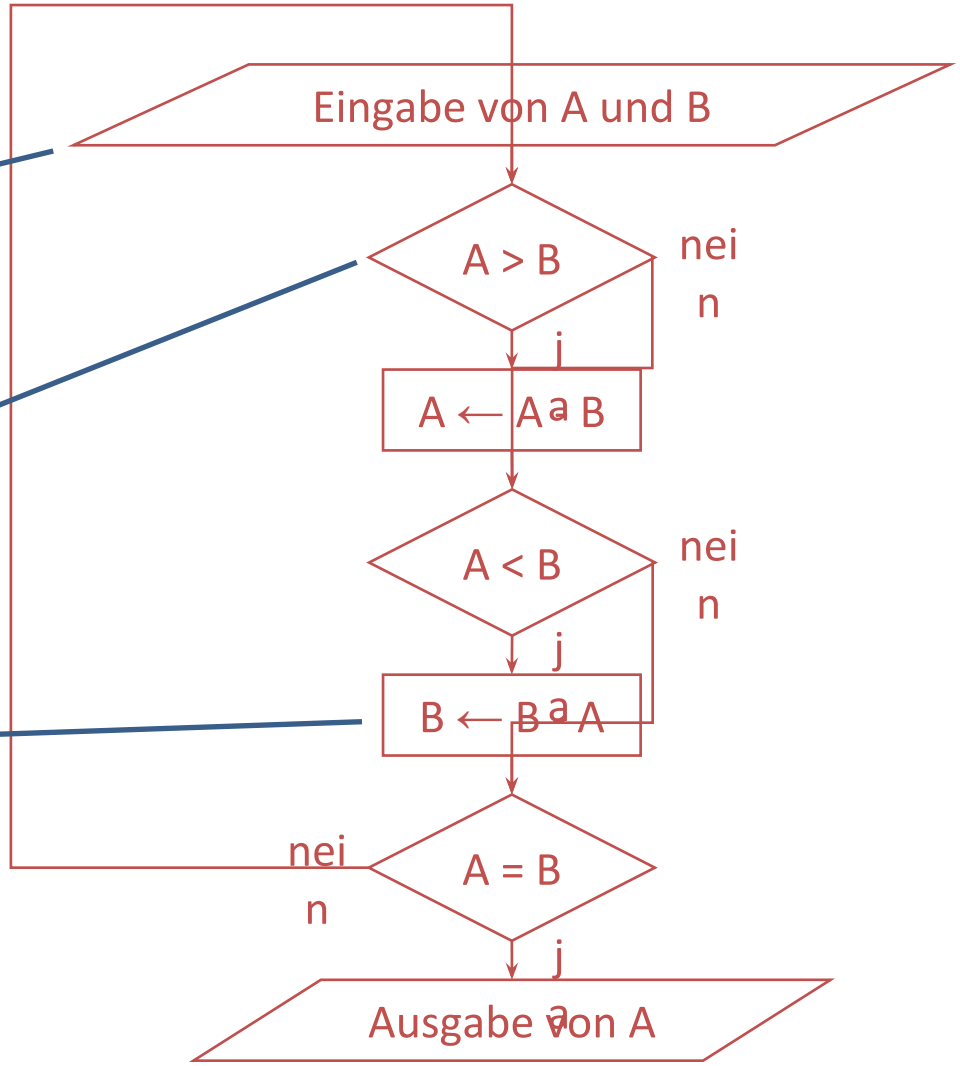
Verbale Darstellung

- Beschreibung in natürliche Sprache

```
Euklidischer Algorithmus ggT(A, B)
    0. Eingabe von A und B
    1. Wenn A größer B, dann subtrahiere B
    von A und weise das Ergebnis A zu
    2. Wenn A kleiner B, dann subtrahiere A
    von B und weise das Ergebnis B zu
    3. Wenn A ungleich B, weiter bei Schritt 1.
    4. Das Ergebnis ist A (oder B)
```

Flussdiagramm

- Ein- und Ausgabewerte
- Verzweigung / Selektion
- Zuweisung



Flussdiagramm

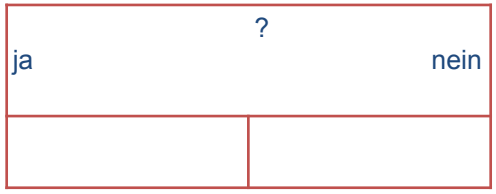
Nachteile eines Flussdiagramms

- Oft unstrukturiert, keine formalen Vorgaben
- Für fremde Leser nicht verständlich, nicht teamfähig
- Nicht wartbar, nicht erweiterbar

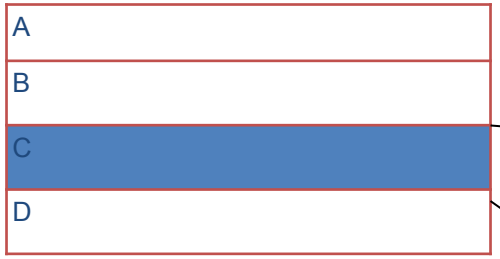
Nassi-Shneiderman-Diagramm

- Einschränkung der Darstellungsmöglichkeiten, führt zu strukturierteren Graphen
- Sequenz

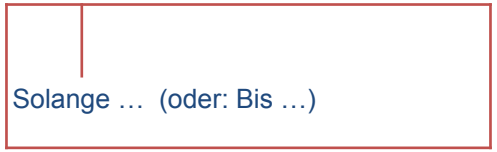
- Fallunterscheidung

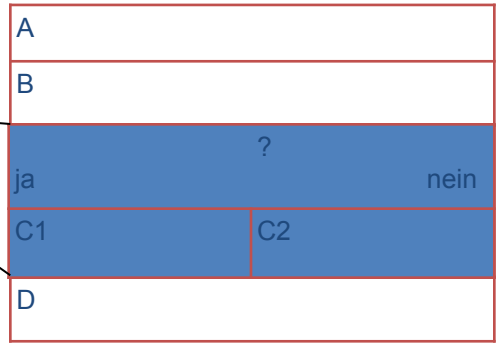


- + Schachtelung!

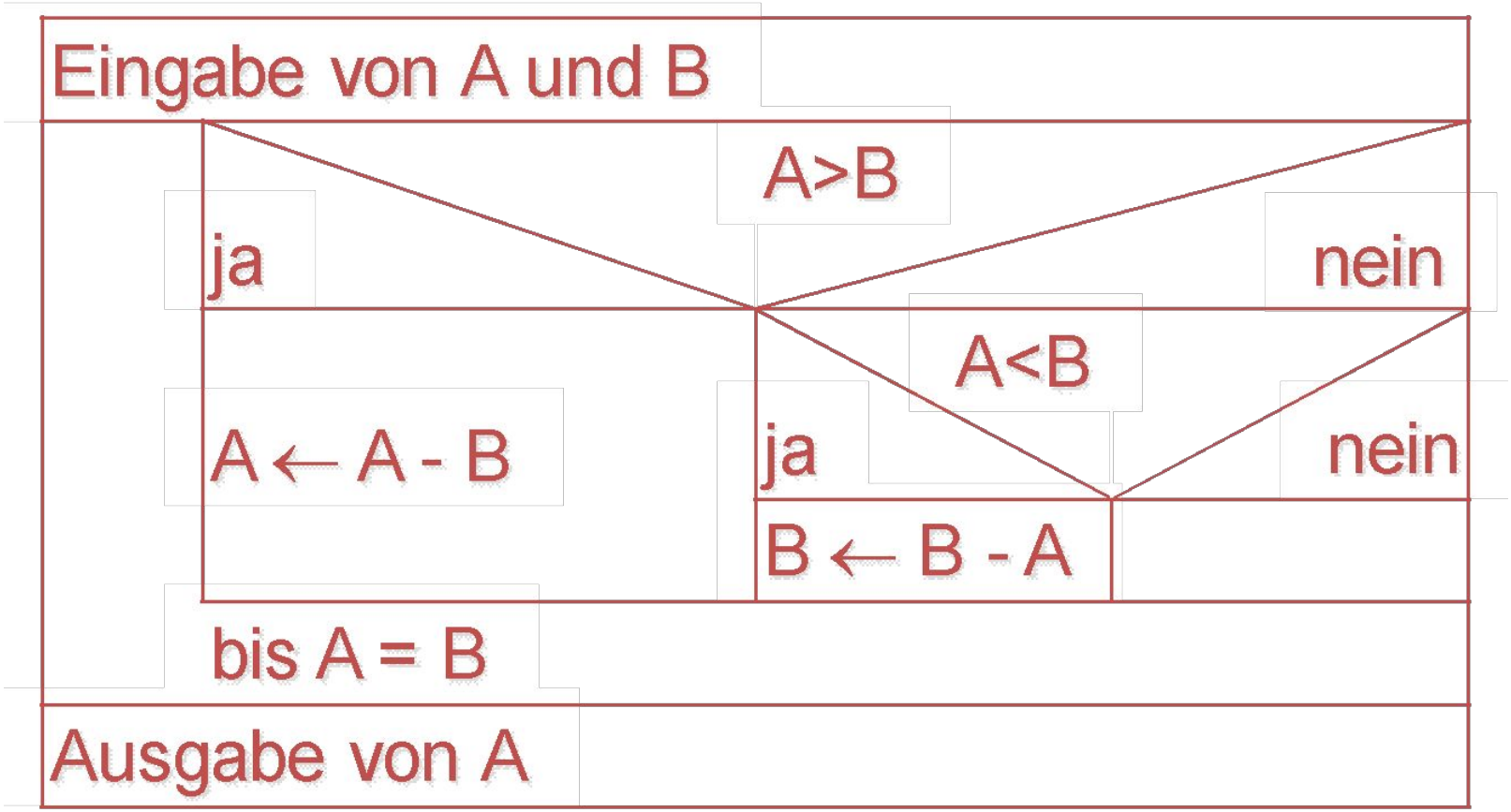


Wiederholung / Schleife





Nassi-Shneiderman-Diagramm: Euklidischer Algorithmus



Pseudocode

- Semiformale Sprachen
- Beispiel:

```
WHILE    A ungleich B
        IF A > B
        THEN subtrahiere B von A
        ELSE
                subtrahiere A von B
        ENDIF
ENDWHILE
ggT := A
```