

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

Assoc. Prof. Dr. Mathias Lux
ITEC / AAU



Modalities



- This is the theoretical lecture.
- It has two parts
 - Part one is part of the STEOP, needed to be allowed further studying.
 - Part two is directly connected, please enroll to both.
- Exam Part 1: Oct 24, 2017 / 12-14, HS A
- Exam Part 2: Jan 19, 2018 / 16-17, HS A
 - Part 1, 2nd try is Jan 19, 2018 / after 17:00 in HS A

Schedule



- Thursdays, 14-16, HS C (s.t.)
 - If it's not taking place, there'll be an email and the campus system will be updated.

Practical course & tutorial

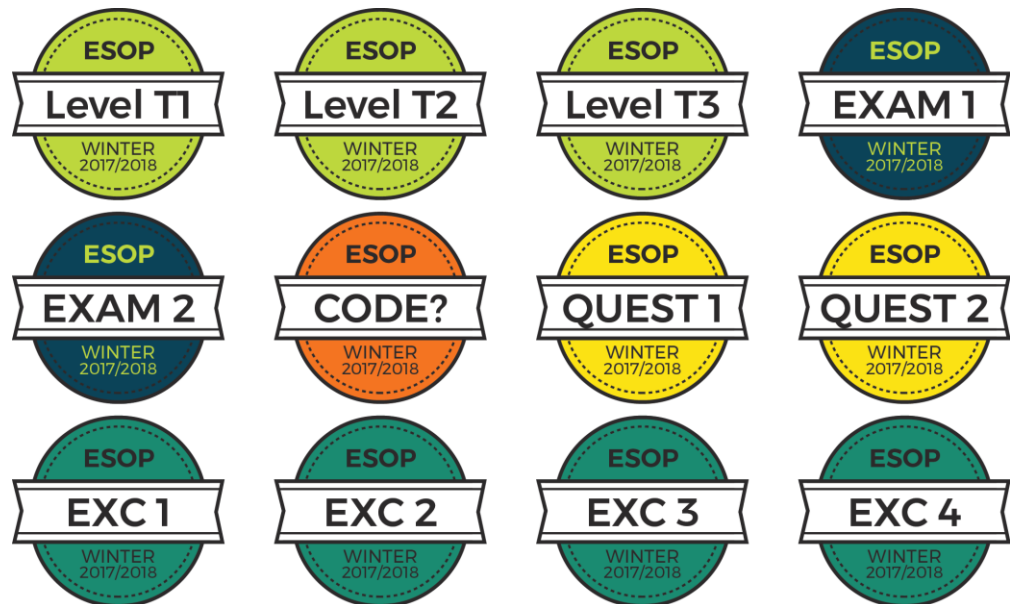


- See online system
 - Bring your computer if you have one.
- The **MORE** course
 - Takes place in a computer lab
 - It's in English and it will revisit the theoretical part too.

Vertical Achievements



- Additional extrinsic motivation
 - up to 12 additional points for the exam (theoretical lecture only)
 - 1 point per badge



Badges



- Level T1-T3: enroll and join in the Tutorium
- EXAM 1-2: take part in the exam preps
- CODE?: send your code to the Code-Review-Team
- QUEST 1-2: ask and answer questions in the lecture
- EXC 1-4: solve optional exercises in Moodle

Readings (German)



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

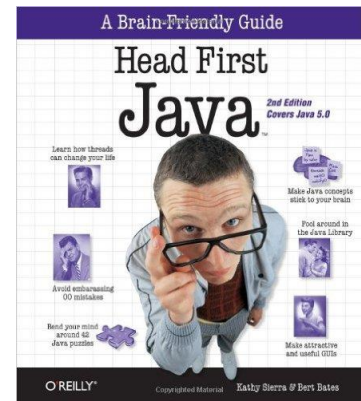


Readings (English)



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

- This book covers object oriented programming, so there is a gap in the first part. For this I recommend [Introduction to Programming Using Java, Seventh Edition](#) by David J. Eck. This book is an extensive introduction to programming based on Java. Read over chapters 1, 2, and 3 to get the necessary background knowledge on variables



Java Documentation



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

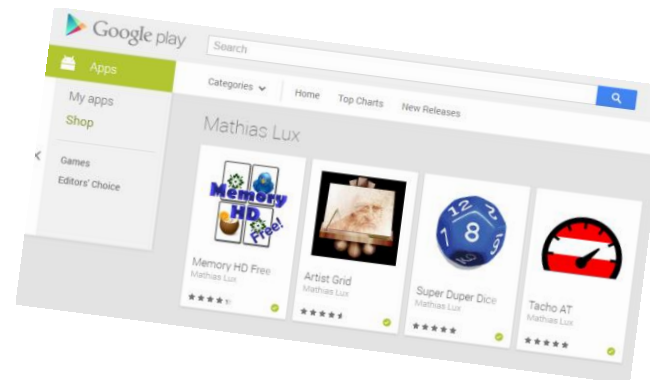
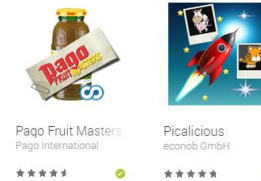


How should I learn Java?



1. Learn to have fun programming. It makes it easier.
2. Invest time in the Java Tutorials and the readings.
3. Go to the course.

Motivation - Why Lux?



Motivation



- It's necessary for research & development
 - Grand Challenge projects, prototypes
- Projects for multimedia production, ie. Processing
- Games, apps, etc.

What is “programming”?



... describing the solution of a problem in such an exact way, that a computer can solve the problem.

Cp. recipes, manuals, etc.

Quelle für die folgenden Folien: Grundlagen der Programmierung, Prof. Dr. Hanspeter Mössenböck

Programming is



- a creative process
- an engineering skill
- a complex task if you want to do it right.

What is a program

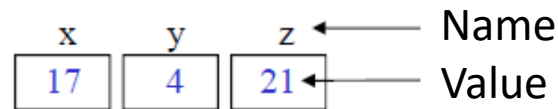


program = data + commands

Data



- Set of address-able memory cells



- Data is stored in binary format, eg. $17 = 10001$
- Binary format is universal
 - numbers, text, image, audio, ...
- 1 Byte = 8 Bit
- 1 Wort = 4 Byte (typically)

Commands



- Operations on memory cells

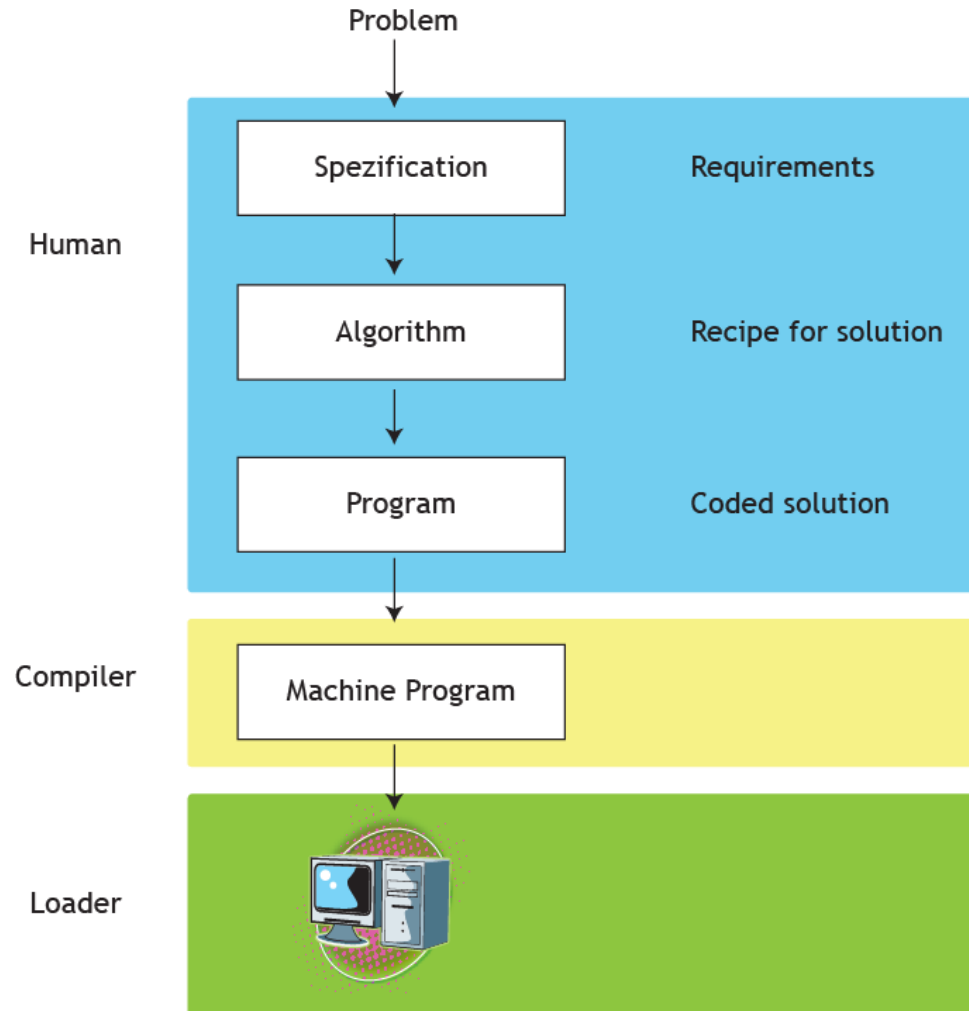
Machine language

Programming Language

$ACC \leftarrow x$	//	load memory cell x
$ACC \leftarrow ACC + y$	//	add memory cell y
$z \leftarrow ACC$	//	store result in memory cell z

$z = x + y;$

How to create a program?



Algorithm



- Precise, step by step solution to a problem

name

parameters

Sum up numbers from 1 to *max* (in:*max*, out:*sum*)

1. *sum* \leftarrow 0

2. *number* \leftarrow 1

3. Iterate as long as *number* smaller or equal *max*

1. *sum* \leftarrow *sum* + *number*

2. *number* \leftarrow *number* + 1

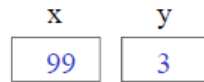
steps

- program = specification of an algorithm in a programming language

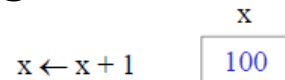
Variables



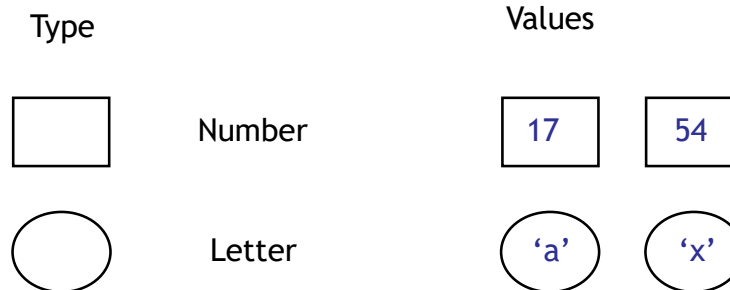
- Variables are named container for values.



- Values can change



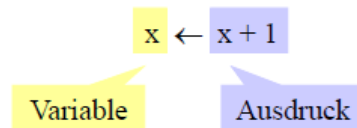
- Variables have a data type
 - which is the set/range of values allowed for a variable.



Statements

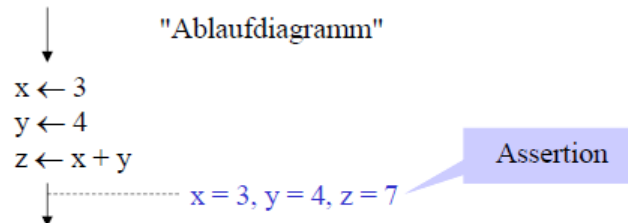


- Assignment



1. compute value
2. assign result to variable

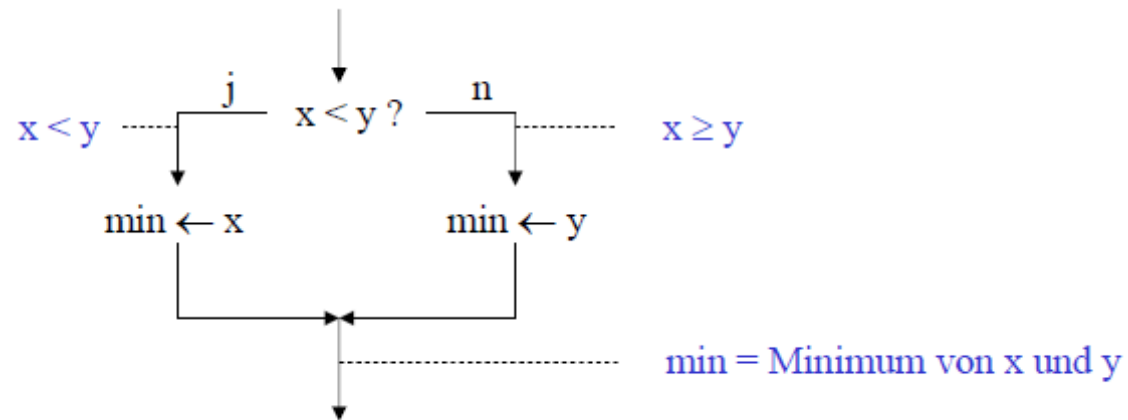
- Sequence of statements



Statements



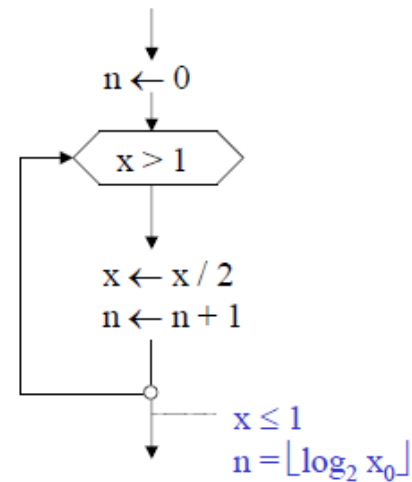
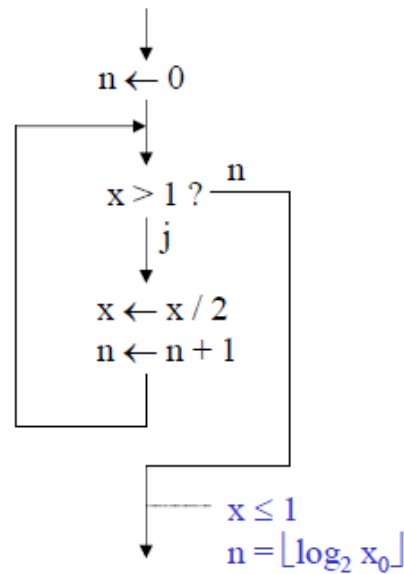
- Condition / Choice



Statements



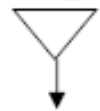
- Iterations, Loops



Example: swap values



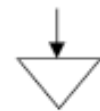
Swap ($\uparrow x, \uparrow y$)



$h \leftarrow x$

$x \leftarrow y$

$y \leftarrow h$



proof of concept

x	y	h
3	2	3
2	3	

Example: swap values



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

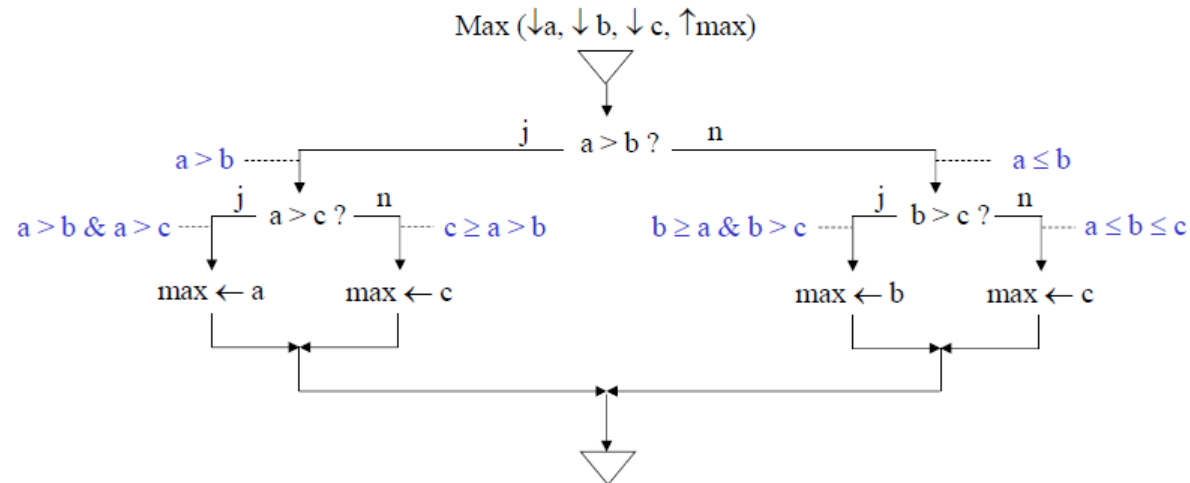
```
println(x);  
println(y);
```

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

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

- Source Code for Processing
- Processing is „like Java“
- int ... data type
- ; ... ends a statement
- println() ... function for printing text on screen.

Example: maximum of three numbers



Example: maximum of three numbers



```
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;
    }
}

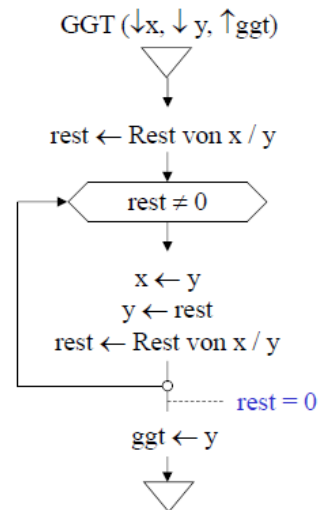
println(max);
```

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

Example: Euclidean algorithm



- Greatest common divisor (ggt) of two numbers.



proof of concept

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

Why does this work?

(ggt divides x) & (ggt divides y)

$\rightarrow x = i \cdot \text{ggt}, y = j \cdot \text{ggt}, (x-y) = (i-j) \cdot \text{ggt}$

\rightarrow ggt divides $(x-y)$

\rightarrow ggt divides $(x - q \cdot y)$

\rightarrow ggt divides rest of x/y

$\rightarrow \text{ggt}(x, y) = \text{ggt}(y, \text{rest})$

Example: Euclidean algorithm



```
int x = 21;
```

```
int y = 14;
```

```
int rest = x % y;
```

```
while (rest != 0) {
```

```
    x = y;
```

```
    y = rest;
```

```
    rest = x % y;
```

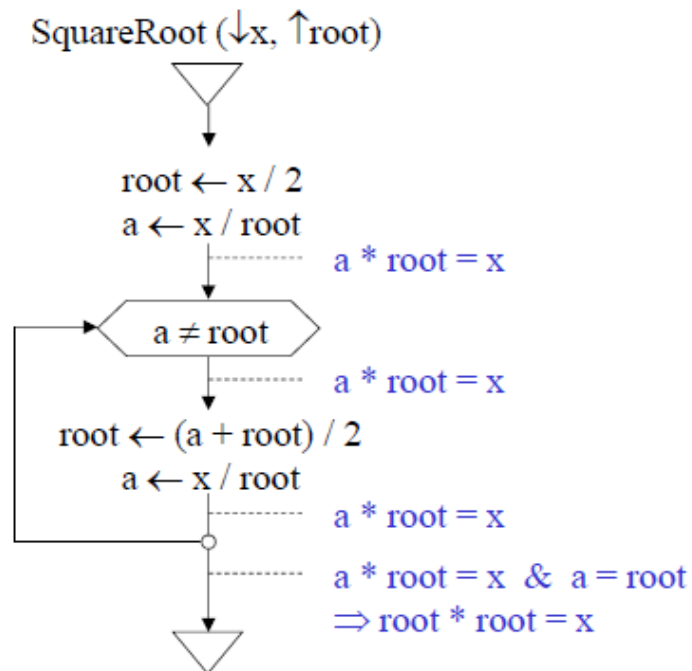
```
}
```

```
println(y);
```

- Source Code for Processing

- While (test) {..}
- % ... modulo

Example: square root



proof of concept

x	root	a
10	8	2
	3.5	-2.85714
	3.17857	3.14607
	3.16232	3.16223
	3.16228	3.16228

Example: square root



```
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 for Processing
- float ... data type
- / ... Division
- Hint: Don't test float on equality!
 - $|a - \text{root}| < 0,00001$

Specification of programming languages

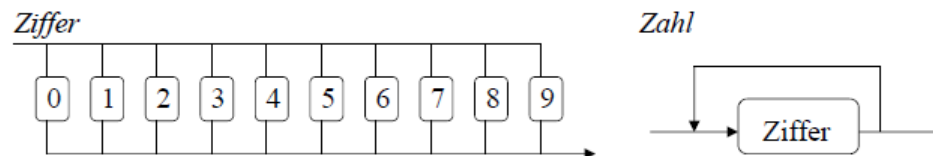


- Syntax
 - rules to build sentences
 - e.g. assignment = variable <- statement
- Semantics
 - Actual meaning of sentences
 - e.g.: compute statement and assign result to variable.

Specification of programming languages



- Grammar
 - Set of syntax rules
 - eg. grammar for discrete positive numbers.
 - Ziffer = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9".
 - Zahl = Ziffer {Ziffer}.



EBNF (Extended Backus-Naur-Form)



Examples

- *Grammar for floating point values*
 - number = numeral {numeral}.
 - float = number "." number ["E" ["+" | "-"] number].
- *Grammar for If-statements*
 - IfStatement = "if" "(" Statement ")" Statement ["else" Statement].

Usage	Notation
definition	=
concatenation	,
termination	;
termination	. ^[1]
alternation	
option	[...]
repetition	{ ... }
grouping	(...)
terminal string	" ... "
terminal string	' ... '
comment	(* ... *)
special sequence	? ... ?
exception	-

src. Wikipedia

Programming Languages



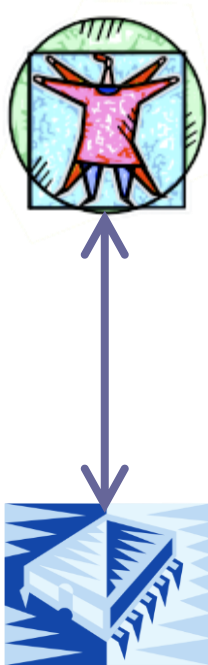
- Formal languages that can be translated to machine language with a program.
 - A program is a „text“ written in a formal language
- There are a lot of different languages
 - Java, Python, C, C++, Objective C, Pascal, Modula, Perl, Basic, C#, JavaScript, Dart, Erlang, LUA uvm.

Programming Languages



- **Compiler:** program is translated
 - by a program
 - to machine code
 - Eg. C, C++
- **Interpreter:**
 - program is executed step by step by another program
 - Eg. Python, Ruby, JavaScript, Perl, LUA

Specification of Algorithms



Graphical or verbal notation

Higher programming languages (like Java)

Assembly languages

Machine code

Hardware, electric signals

Verbale Notation



- Description in natural language

Euclidian Algorithm $\text{ggT}(A, B)$

0. Input of A and B

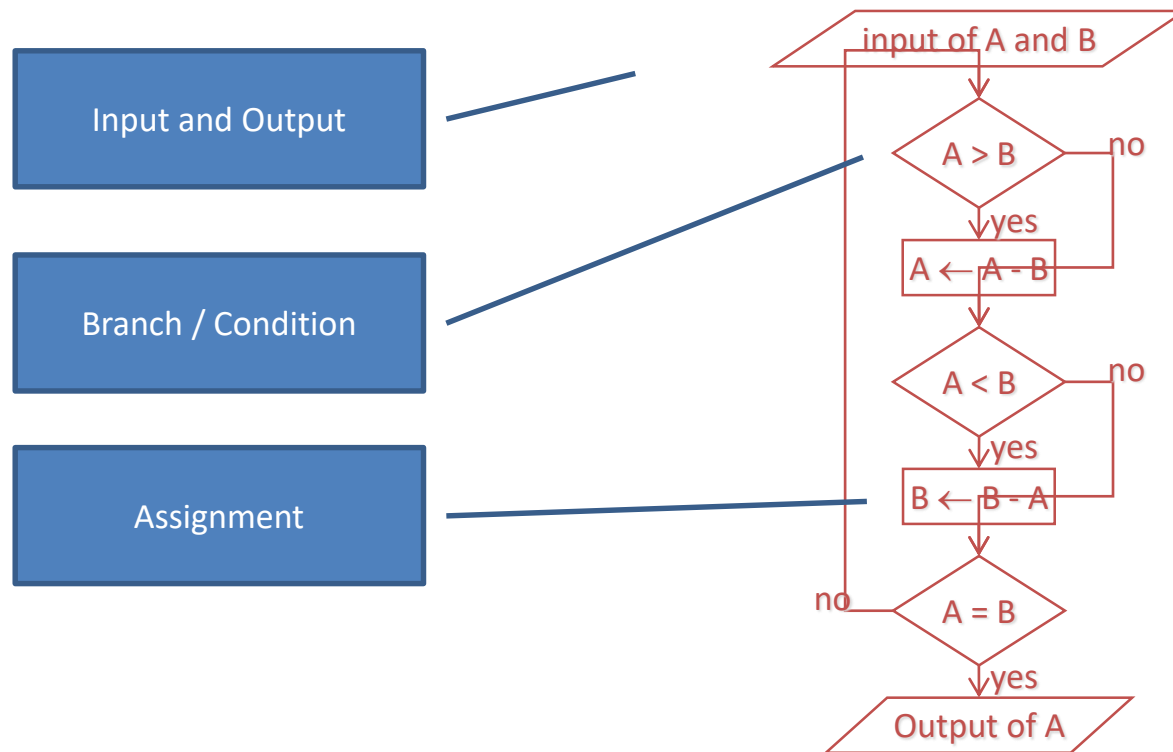
1. If A larger than B, then subtract B from A and assign the result to A.

2. If A smaller than B then subtract A from B and assign the result to B.

3. If A is not equal B then go to step 1

4. The result is A (or B)

Flowchart



Flowchart



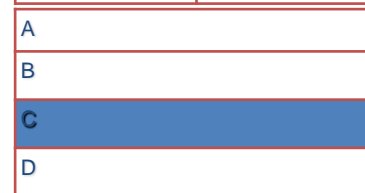
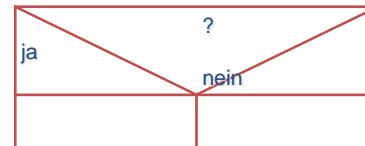
Contra

- Often unstructured, no formal framework.
- Not good for working in teams, hard to read for others
- Hard to update and revise.

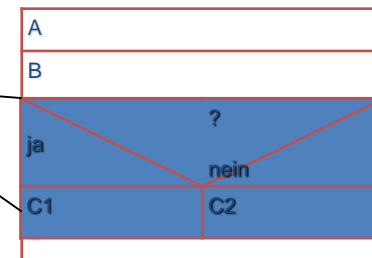
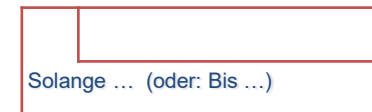
Nassi-Shneiderman-Chart



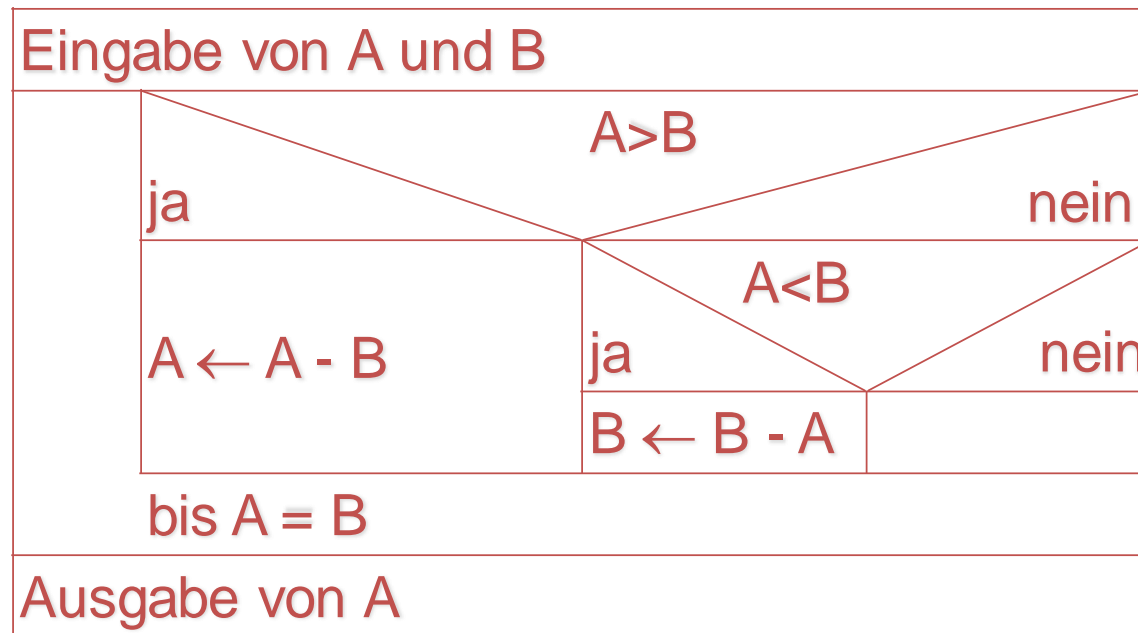
- More structured due to stronger restrictions.
- Sequence
- Branch / Condition
- + nesting!



Iteration/ Loop



Nassi-Shneiderman-Chart: Euclidian Algorithm



Pseudocode



- Semi-formal languages
- Examples:

```
WHILE  A not equal B
  IF A > B
    THEN subtract B from A
  ELSE
    subtract A from B
  ENDIF
ENDWHILE
ggT := A
```