

Informatik I

Vorlesung am D-BAUG der ETH Zürich

Hermann Lehner, Felix Friedrich
ETH Zürich

HS 2018

1. Einführung

Willkommen zur Vorlesung !

Vorlesungshomepage:

<http://lec.inf.ethz.ch/baug/informatik1>

Das Team

Dozenten

Hermann Lehner
Felix Friedrich

Chef-Assistent

Andrea Lattuada

Assistenten

Vincent Becker	Lukas Burkhalter
Mihai Bace	Irfan Bunjaku
Patrick Gruntz	Max Rossmannek
Josua Schneider	Rafael Wampfler
Temmy Boundedjar	Simon Guldemann
Staal Sander	

Programmieren und Problemlösen

In diesem Kurs “lernen” Sie programmieren in Java

- Die Software Entwicklung ist ein *Handwerk*.
- Vergleich: Erlernen eines Musikinstruments.

Programmieren und Problemlösen

In diesem Kurs “lernen” Sie programmieren in Java

- Die Software Entwicklung ist ein *Handwerk*.
- Vergleich: Erlernen eines Musikinstruments.
- **Das Problem:** Es ist noch keiner vom Zuhören Pianist geworden.

Programmieren und Problemlösen

In diesem Kurs “lernen” Sie programmieren in Java

- Die Software Entwicklung ist ein *Handwerk*.
- Vergleich: Erlernen eines Musikinstruments.
- **Das Problem:** Es ist noch keiner vom Zuhören Pianist geworden.

Deshalb bietet dieser Kurs Ihnen viele Möglichkeiten, zu üben.
Nutzen Sie dies aus!

Programmieren und Problemlösen

In diesem Kurs *lernen* Sie Problemlösen mit ausgewählten Algorithmen und Datenstrukturen.

- Sprach-übergreifendes *Grundlagenwissen*
- Vergleich: Rhythmus-Lehre, Tonleitern, Noten-Lesen.

Programmieren und Problemlösen

In diesem Kurs *lernen* Sie Problemlösen mit ausgewählten Algorithmen und Datenstrukturen.

- Sprach-übergreifendes *Grundlagenwissen*
- Vergleich: Rhythmus-Lehre, Tonleitern, Noten-Lesen.
- **Das Problem:** Ohne Musikinstrument macht dies kein Spass.

Programmieren und Problemlösen

In diesem Kurs *lernen* Sie Problemlösen mit ausgewählten Algorithmen und Datenstrukturen.

- Sprach-übergreifendes *Grundlagenwissen*
- Vergleich: Rhythmus-Lehre, Tonleitern, Noten-Lesen.
- **Das Problem:** Ohne Musikinstrument macht dies kein Spass.

Deshalb kombinieren wir das Problemlösen mit dem Erlernen von Java.

Inhalte der Vorlesung

Programmieren mit Java

Einführung

Anweisungen und Ausdrücke

Zahlendarstellungen

Kontrollfluss

Arrays

Methoden und Rekursion

Typen, Klassen und Objekte

Vererbung und Polymorphie

Algorithmen

Suchen und Sortieren

Ziel der *heutigen* Vorlesung

- Einführung *Computermodell* und Algorithmus
- Allgemeine Informationen zur Vorlesung
- Das *erste Programm* schreiben

1.1 Informatik und Algorithmus

Informatik, der Euklidische Algorithmus

Was ist Informatik?

Was ist Informatik?

- Die Wissenschaft der **systematischen Verarbeitung von Informationen**, . . .

Was ist Informatik?

- Die Wissenschaft der **systematischen Verarbeitung von Informationen**, . . .
- . . . insbesondere der automatischen Verarbeitung mit Hilfe von Digitalrechnern.

(Wikipedia, nach dem “Duden Informatik”)

Informatik \neq EDV-Kenntnisse

EDV-Kenntnisse: *Anwenderwissen*

- Umgang mit dem Computer
- Bedienung von Computerprogrammen (für Texterfassung, Email, Präsentationen, . . .)

Informatik \neq EDV-Kenntnisse

Informatik: *Grundlagenwissen*

- Wie funktioniert ein Computer?
- Wie schreibt man ein Computerprogramm?

Inhalt dieser Vorlesung

- Systematisches Problemlösen mit Algorithmen und der Programmiersprache Java.
- Also: *nicht nur,*
aber auch Programmierkurs.

Algorithmus: Kernbegriff der Informatik

Algorithmus:

- Handlungsanweisung zur schrittweisen Lösung eines Problems

Algorithmus: Kernbegriff der Informatik

Algorithmus:

- Handlungsanweisung zur schrittweisen Lösung eines Problems
- Ausführung erfordert keine Intelligenz, nur Genauigkeit (sogar Computer können es)

Algorithmus: Kernbegriff der Informatik

Algorithmus:

- Handlungsanweisung zur schrittweisen Lösung eines Problems
- Ausführung erfordert keine Intelligenz, nur Genauigkeit (sogar Computer können es)
- nach *Muhammed al-Chwarizmi*,
Autor eines arabischen
Rechen-Lehrbuchs (um 825)



“Dixit algorizmi...” (lateinische Übersetzung)

Der älteste nichttriviale Algorithmus

Euklidischer Algorithmus (aus Euklids *Elementen*, 3. Jh. v. Chr.)

- Eingabe: ganze Zahlen $a > 0, b > 0$
- Ausgabe: ggT von a und b

Solange $b \neq 0$

Wenn $a > b$ dann

$$a \leftarrow a - b$$

Sonst:

$$b \leftarrow b - a$$

Ergebnis: a .



Der älteste nichttriviale Algorithmus

Euklidischer Algorithmus (aus Euklids *Elementen*, 3. Jh. v. Chr.)

- Eingabe: ganze Zahlen $a > 0, b > 0$
- Ausgabe: ggT von a und b

Solange $b \neq 0$

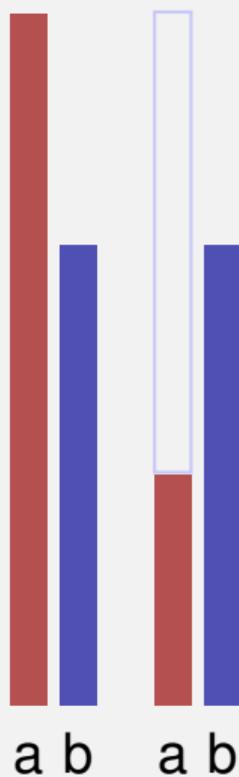
Wenn $a > b$ dann

$$a \leftarrow a - b$$

Sonst:

$$b \leftarrow b - a$$

Ergebnis: a .



Der älteste nichttriviale Algorithmus

Euklidischer Algorithmus (aus Euklids *Elementen*, 3. Jh. v. Chr.)

- Eingabe: ganze Zahlen $a > 0, b > 0$
- Ausgabe: ggT von a und b

Solange $b \neq 0$

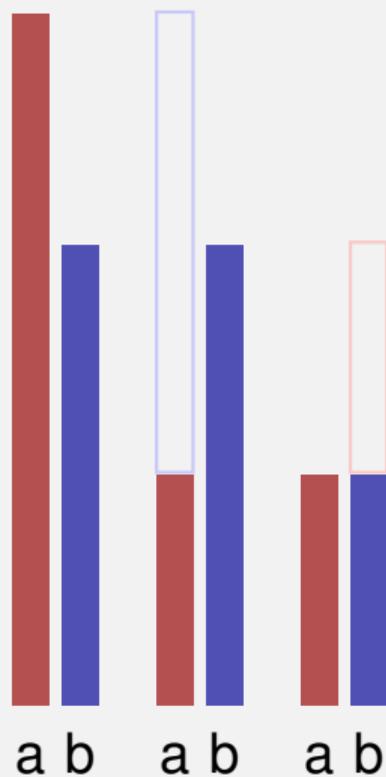
Wenn $a > b$ dann

$$a \leftarrow a - b$$

Sonst:

$$b \leftarrow b - a$$

Ergebnis: a .



Der älteste nichttriviale Algorithmus

Euklidischer Algorithmus (aus Euklids *Elementen*, 3. Jh. v. Chr.)

- Eingabe: ganze Zahlen $a > 0, b > 0$
- Ausgabe: ggT von a und b

Solange $b \neq 0$

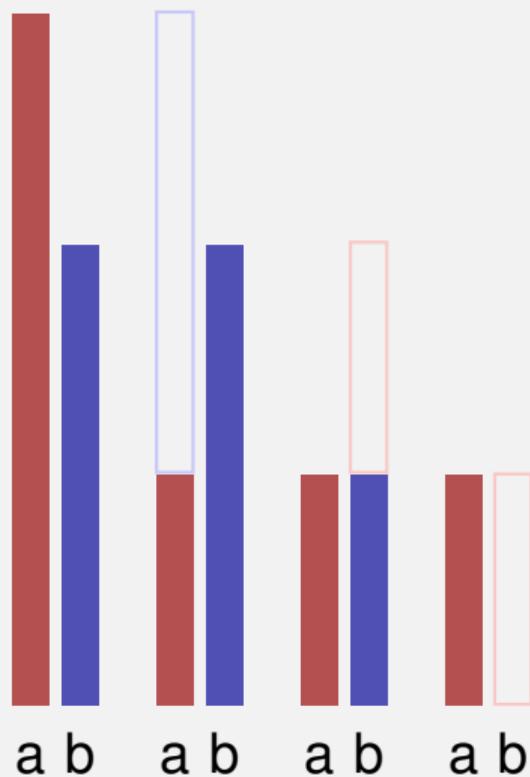
Wenn $a > b$ dann

$$a \leftarrow a - b$$

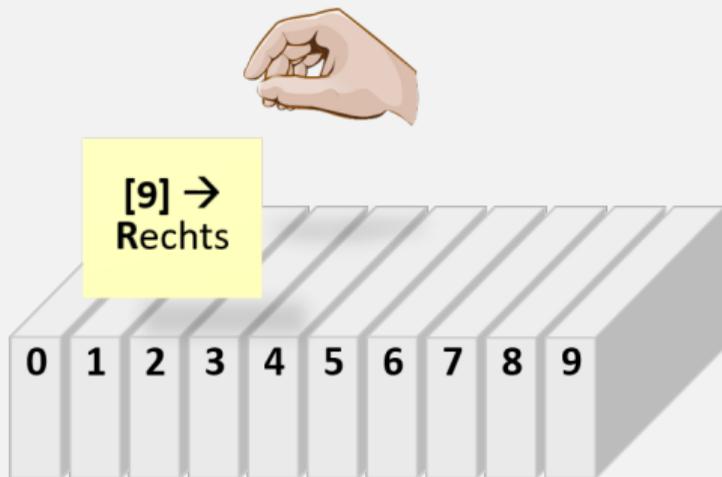
Sonst:

$$b \leftarrow b - a$$

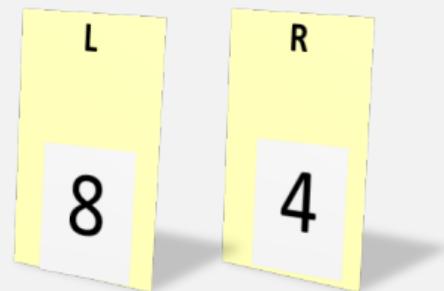
Ergebnis: a .



Live Demo: Turing Maschine



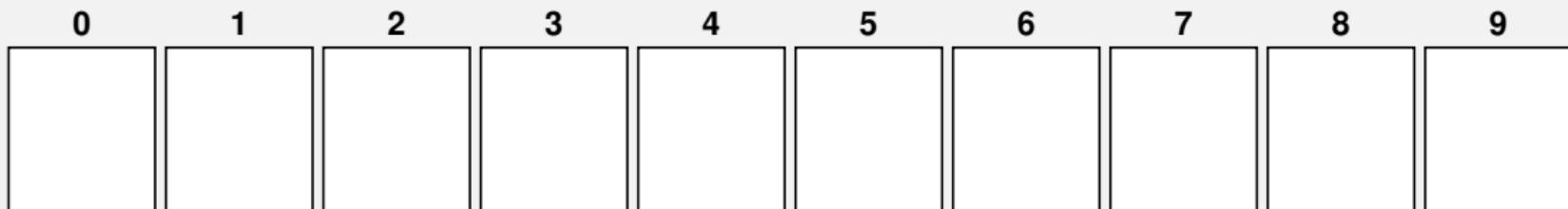
Speicher



Register

Euklid in der Box

Speicher



Links

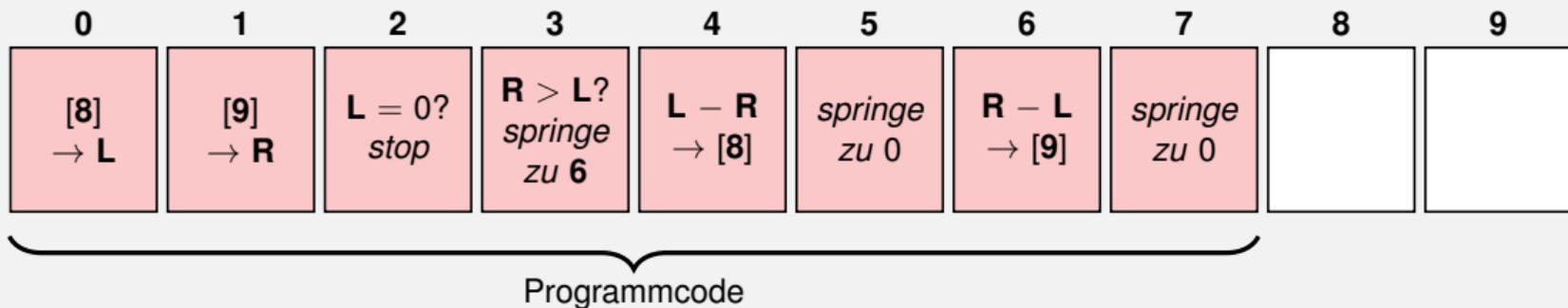
Rechts



Register

Euklid in der Box

Speicher



Links

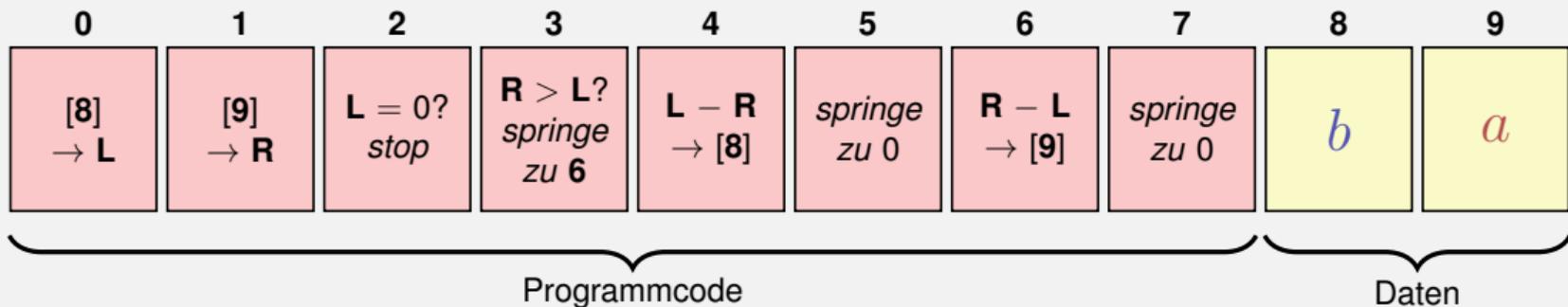
Rechts



Register

Euklid in der Box

Speicher



Links

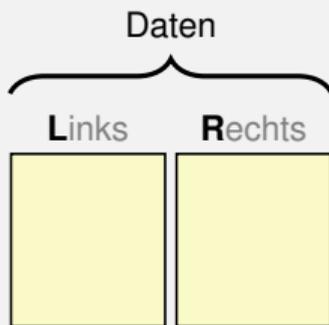
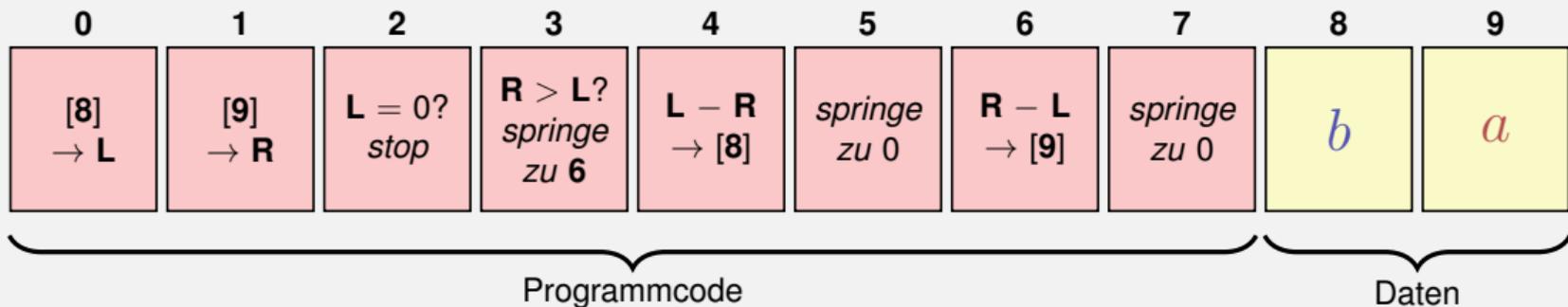
Rechts



Register

Euklid in der Box

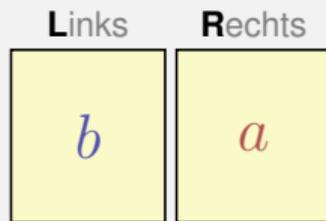
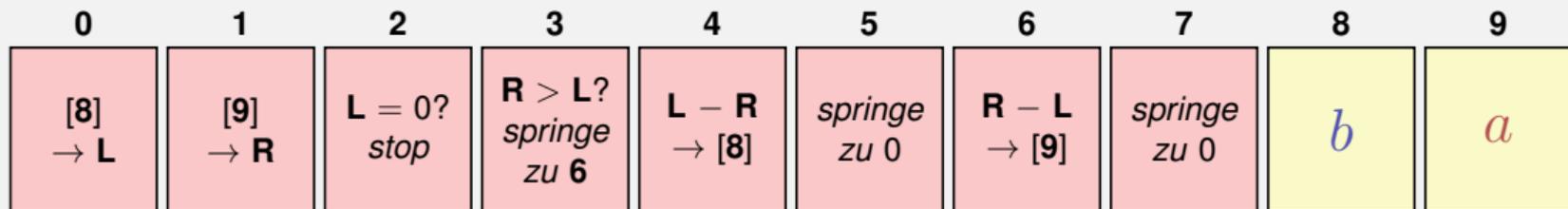
Speicher



Register

Euklid in der Box

Speicher



Register

Solange $b \neq 0$

Wenn $a > b$ dann

$$a \leftarrow a - b$$

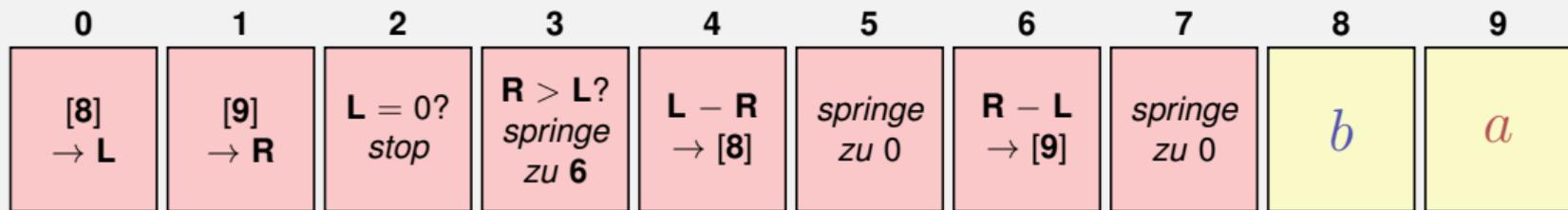
Sonst:

$$b \leftarrow b - a$$

Ergebnis: a .

Euklid in der Box

Speicher



Solange $b \neq 0$

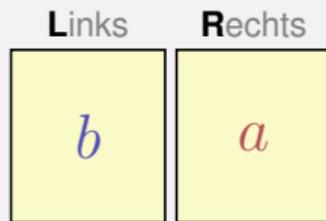
Wenn $a > b$ dann

$$a \leftarrow a - b$$

Sonst:

$$b \leftarrow b - a$$

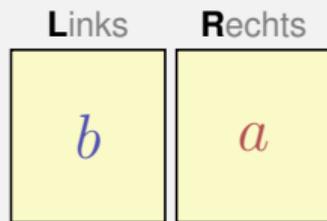
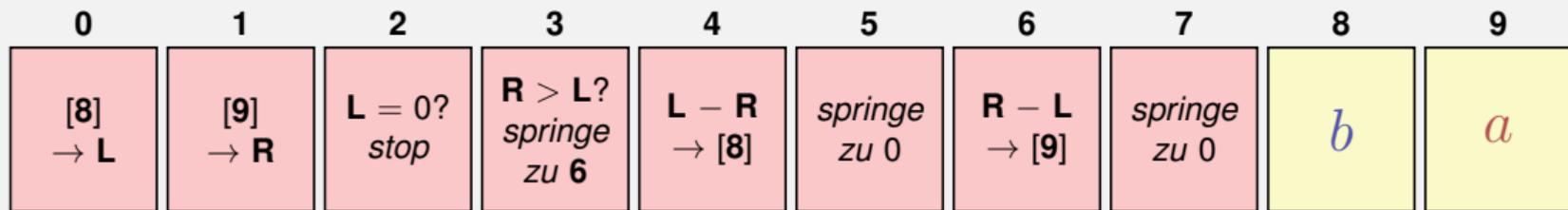
Ergebnis: a .



Register

Euklid in der Box

Speicher



Register

Solange $b \neq 0$

Wenn $a > b$ dann

$$a \leftarrow a - b$$

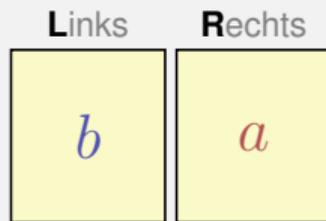
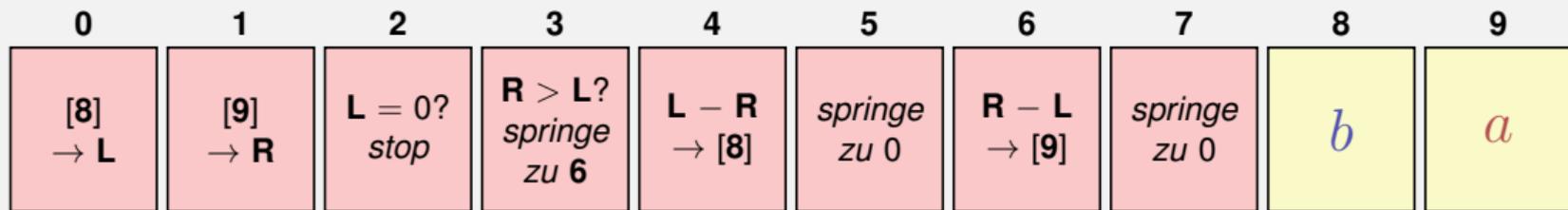
Sonst:

$$b \leftarrow b - a$$

Ergebnis: a .

Euklid in der Box

Speicher



Register

Solange $b \neq 0$

Wenn $a > b$ dann

$$a \leftarrow a - b$$

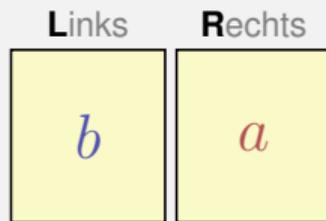
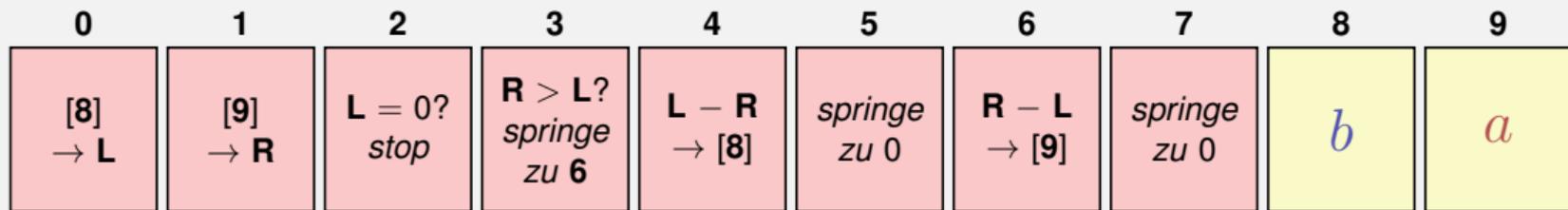
Sonst:

$$b \leftarrow b - a$$

Ergebnis: a .

Euklid in der Box

Speicher



Register

Solange $b \neq 0$

Wenn $a > b$ dann

$$a \leftarrow a - b$$

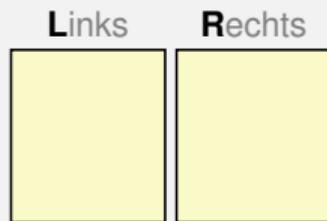
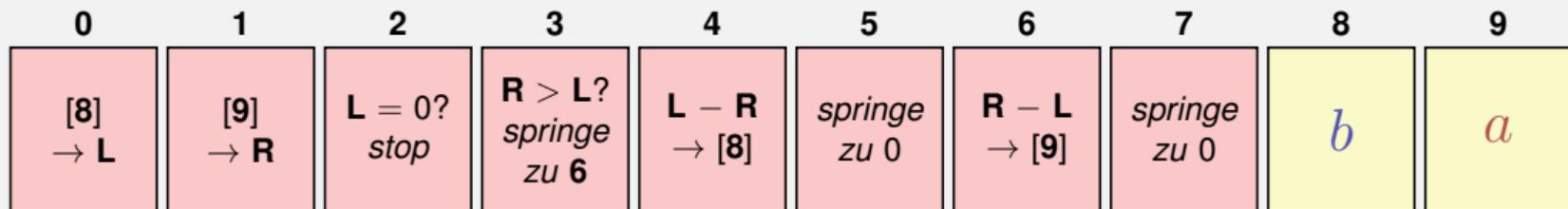
Sonst:

$$b \leftarrow b - a$$

Ergebnis: a .

Euklid in der Box

Speicher



Register

Solange $b \neq 0$

Wenn $a > b$ dann

$$a \leftarrow a - b$$

Sonst:

$$b \leftarrow b - a$$

Ergebnis: a .

1.3 Computermodell

Turing Maschine, Von Neumann Architektur

Computer – Konzept

Eine geniale Idee: Universelle Turingmaschine (Alan Turing, 1936)

Folge von Symbolen auf Ein- und Ausgabeband



Lese- /
Schreibkopf



«Symbol lesen»
«Symbol überschreiben»
«Nach links»
«Nach rechts»

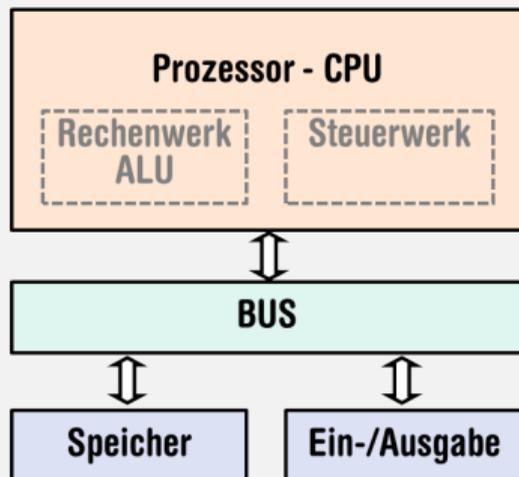


Alan Turing

Computer – Umsetzung

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

Von Neumann Architektur



Konrad Zuse



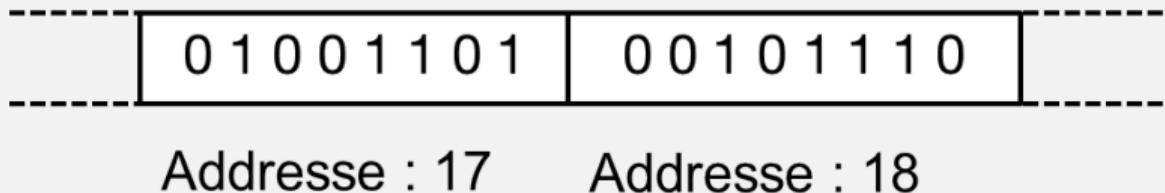
John von Neumann

Speicher für Daten *und* Programm

- Folge von Bits aus $\{0, 1\}$.
- Programmzustand: Werte aller Bits.
- Zusammenfassung von Bits zu Speicherzellen (oft: 8 Bits = 1 Byte).

Speicher für Daten *und* Programm

- Jede Speicherzelle hat eine Adresse.
- Random Access: Zugriffszeit auf Speicherzelle (nahezu) unabhängig von ihrer Adresse.



Rechengeschwindigkeit

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

¹Uniprozessor Computer bei 1GHz

Rechengeschwindigkeit

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



30 m

arbeitet ein heutiger Desktop-PC mehr als 100

¹Uniprozessor Computer bei 1GHz

Rechengeschwindigkeit

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



30 m $\hat{=}$ mehr als 100.000.000 Instruktionen

arbeitet ein heutiger Desktop-PC mehr als 100 Millionen Instruktionen ab.¹

¹Uniprozessor Computer bei 1GHz

Programmieren

- Mit Hilfe einer *Programmiersprache* wird dem Computer eine Folge von Befehlen erteilt, damit er genau das macht, was wir wollen.
- Die Folge von Befehlen ist das *(Computer)-Programm*.



The Harvard Computers, Menschliche Berufsrechner, ca.1890

Warum Programmieren?

- Da hätte ich ja gleich Informatik studieren können ...

Warum Programmieren?

- Da hätte ich ja gleich Informatik studieren können ...
- Es gibt doch schon für alles Programme ...

Warum Programmieren?

- Da hätte ich ja gleich Informatik studieren können ...
- Es gibt doch schon für alles Programme ...
- Programmieren interessiert mich nicht ...

Warum Programmieren?

- Da hätte ich ja gleich Informatik studieren können ...
- Es gibt doch schon für alles Programme ...
- Programmieren interessiert mich nicht ...
- Weil Informatik hier leider ein Pflichtfach ist ...

Warum Programmieren?

- Da hätte ich ja gleich Informatik studieren können ...
- Es gibt doch schon für alles Programme ...
- Programmieren interessiert mich nicht ...
- Weil Informatik hier leider ein Pflichtfach ist ...
- ...

*Mathematik war früher die Lingua franca der
Naturwissenschaften an allen Hochschulen. Und heute ist
dies die Informatik.*

Lino Guzzella, Präsident der ETH Zürich, NZZ Online, 1.9.2017

Darum Programmieren!

- Jedes Verständnis moderner Technologie erfordert Wissen über die grundlegende Funktionsweise eines Computers.
- Programmieren (mit dem Werkzeug Computer) wird zu einer Kulturtechnik wie Lesen und Schreiben (mit den Werkzeugen Papier und Bleistift)

Darum Programmieren!

- Jedes Verständnis moderner Technologie erfordert Wissen über die grundlegende Funktionsweise eines Computers.
- Programmieren (mit dem Werkzeug Computer) wird zu einer Kulturtechnik wie Lesen und Schreiben (mit den Werkzeugen Papier und Bleistift)
- Die meisten qualifizierten Jobs benötigen zumindest elementare Programmierkenntnisse.

Darum Programmieren!

- Jedes Verständnis moderner Technologie erfordert Wissen über die grundlegende Funktionsweise eines Computers.
- Programmieren (mit dem Werkzeug Computer) wird zu einer Kulturtechnik wie Lesen und Schreiben (mit den Werkzeugen Papier und Bleistift)
- Die meisten qualifizierten Jobs benötigen zumindest elementare Programmierkenntnisse.
- Programmieren macht Spass!

Dieser Kurs ist für Sie

- Sie erlernen das *Fundament* – die Grundlagen der Informatik und des Programmierens – bei uns auf einem anspruchsvollen Niveau.
- Sie müssen die erlernten *Prinzipien* später in einem anderen Kontext – unter anderem für andere Programmiersprachen (C++, Python ,Matlab ,R) – *anwenden können*.
- Das ist keine Vorgabe von uns – wir wissen das von *Ihnen* (= Ihrem Departement).

Programmiersprachen

- Sprache, die der Computer "verstehen", ist sehr primitiv (Maschinensprache).
- Einfache Operationen müssen in viele Einzelschritte aufgeteilt werden.
- Sprache variiert von Computer zu Computer.

Höhere Programmiersprachen

darstellbar als Programmtext, der

- von Menschen *verstanden* werden kann
- vom Computermodell *unabhängig* ist
→ Abstraktion!

Java

- Basiert auf einer *virtuellen Maschine* (mit von-Neumann Architektur)

Java

- Basiert auf einer *virtuellen Maschine* (mit von-Neumann Architektur)
 - Programmcode wird in Zwischencode übersetzt

- Basiert auf einer *virtuellen Maschine* (mit von-Neumann Architektur)
 - Programmcode wird in Zwischencode übersetzt
 - Zwischencode läuft in einer simulierten Rechnerumgebung, Interpretation des Zwischencodes durch einen Interpreter

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

- Basiert auf einer *virtuellen Maschine* (mit von-Neumann Architektur)
 - Programmcode wird in Zwischencode übersetzt
 - Zwischencode läuft in einer simulierten Rechnerumgebung, Interpretation des Zwischencodes durch einen Interpreter
 - Optimierung: Just-In-Time (JIT) Kompilation von häufig genutztem Code: virtuelle Maschine → physikalische Maschine
- Folgerung, und erklärtes Ziel der Entwickler von Java: Portabilität
write once – run anywhere

1.5 Allgemeine Informationen zur Vorlesung

Organisatorisches, Tools, Übungen, Prüfung

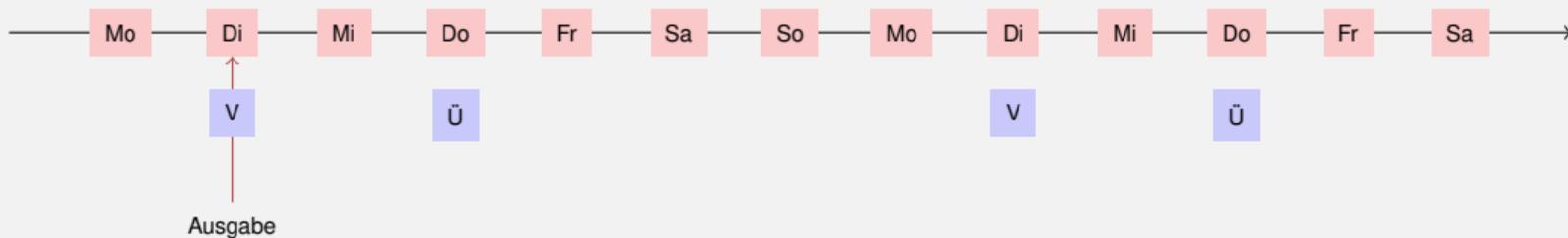
Einschreibung in Übungsgruppen

- Gruppeneinteilung selbstständig via Webseite
`http://echo.ethz.ch`
- Funktioniert nach Belegung dieser Vorlesung in myStudies.
- Die gezeigten Räume und Termine abhängig vom Studiengang.

Übungsbetrieb

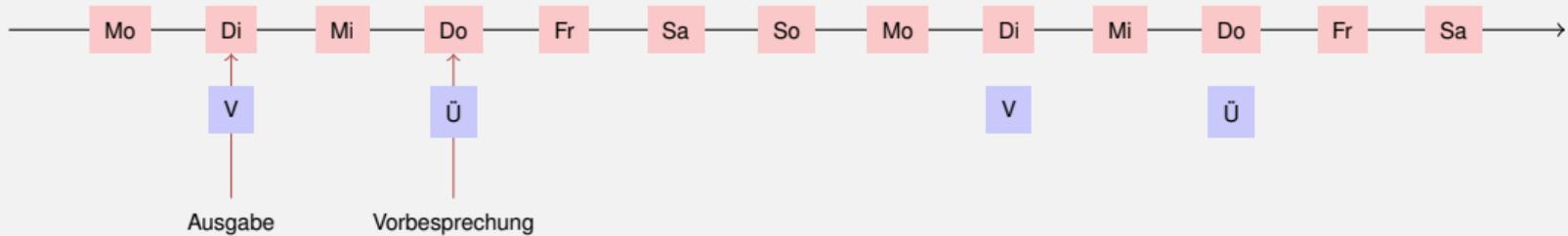


Übungsbetrieb



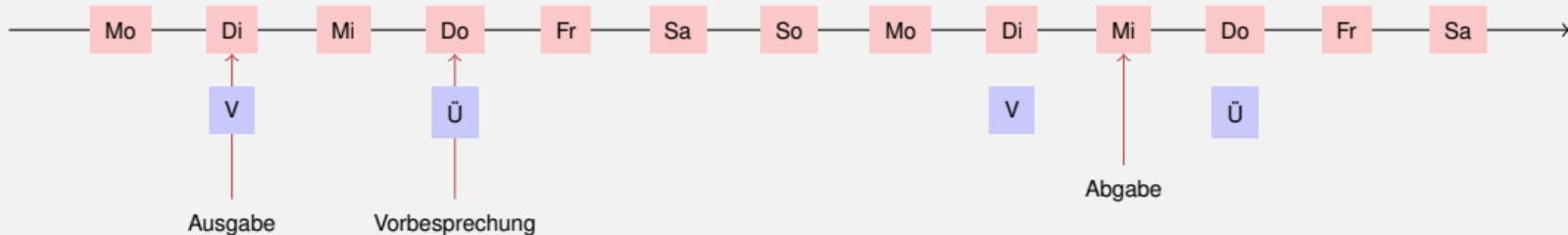
- Übungsblattausgabe zur Vorlesung (online).
- Vorbesprechung in der folgenden Übung.
- Bearbeitung der Übung bis spätestens am Tag vor der nächsten Übungsstunde (23:59h).
- Nachbesprechung der Übung in der nächsten Übungsstunde. Feedback zu den Abgaben innerhalb einer Woche nach Nachbesprechung.

Übungsbetrieb



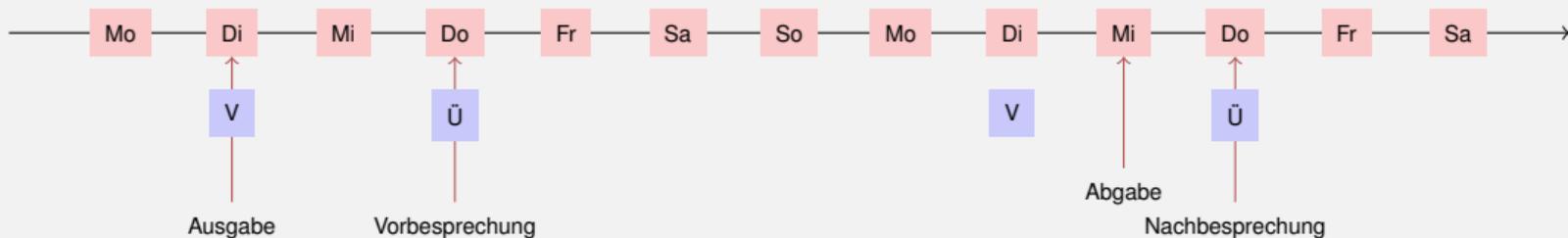
- Übungsblattausgabe zur Vorlesung (online).
- **Vorbereitung in der folgenden Übung.**
- Bearbeitung der Übung bis spätestens am Tag vor der nächsten Übungsstunde (23:59h).
- Nachbesprechung der Übung in der nächsten Übungsstunde. Feedback zu den Abgaben innerhalb einer Woche nach Nachbesprechung.

Übungsbetrieb



- Übungsblattausgabe zur Vorlesung (online).
- Vorbereitung in der folgenden Übung.
- **Bearbeitung der Übung bis spätestens am Tag vor der nächsten Übungsstunde (23:59h).**
- Nachbesprechung der Übung in der nächsten Übungsstunde. Feedback zu den Abgaben innerhalb einer Woche nach Nachbesprechung.

Übungsbetrieb



- Übungsblattausgabe zur Vorlesung (online).
- Vorbereitung in der folgenden Übung.
- Bearbeitung der Übung bis spätestens am Tag vor der nächsten Übungsstunde (23:59h).
- Nachbereitung der Übung in der nächsten Übungsstunde. Feedback zu den Abgaben innerhalb einer Woche nach Nachbereitung.

Zu den Übungen

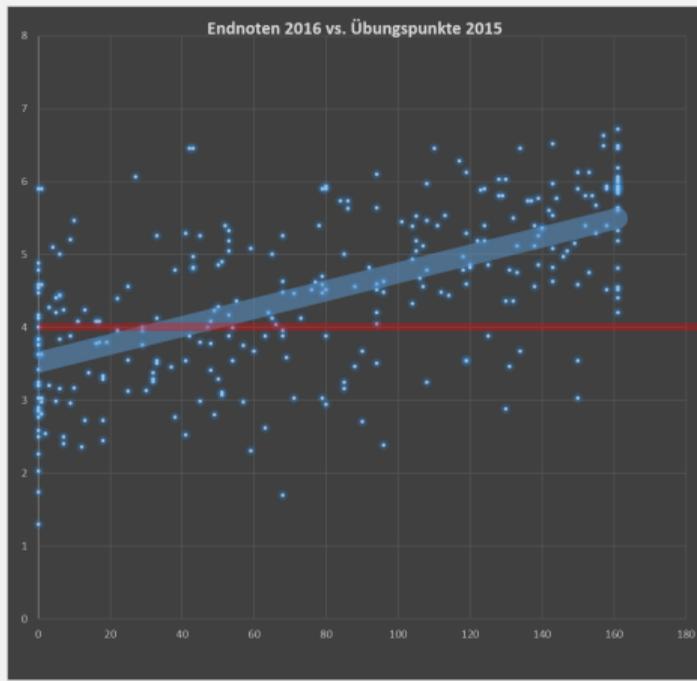
- An der ETH ist (seit HS 2013) für die Prüfungszulassung kein Testat erforderlich.

Zu den Übungen

- Bearbeitung der wöchentlichen Übungsserien ist also freiwillig, wird aber *dringend* empfohlen!

Zu den Übungen

- Bearbeitung der wöchentlichen Übungsserien ist also freiwillig, wird aber *dringend* empfohlen!



Fehlende Ressourcen sind keine Entschuldigung!

Für die Übungen verwenden wir eine Online-Entwicklungsumgebung, benötigt lediglich einen Browser, Internetverbindung und Ihr ETH Login.

Falls Sie keinen Zugang zu einem Computer haben: in der ETH stehen an vielen Orten öffentlich Computer bereit.

Tutorial

In der ersten Woche bearbeiten Sie selbständig unser *Java-Tutorial*

- Einfacher Einstieg in Java, kein Vorwissen nötig!
- Zeitbedarf: ca. zwei Stunden
- In der zweiten Woche gibt's ein *Self Assessment* zum Tutorial

Tutorial

In der ersten Woche bearbeiten Sie selbständig unser *Java-Tutorial*

- Einfacher Einstieg in Java, kein Vorwissen nötig!
- Zeitbedarf: ca. zwei Stunden
- In der zweiten Woche gibt's ein *Self Assessment* zum Tutorial

→ Das ist gut investierte Zeit!

Tutorial - Url

Java Tutorial

Hier finden Sie das Tutorial

<https://frontend-1.et.ethz.ch/sc/WKrEKYAuHvaeTqLzr>

Buch zur Vorlesung

Sprechen Sie Java?

Hanspeter Mössenböck

dpunkt.verlag

- Gut aufgebautes Lernmaterial
- Vertiefte Diskussion der Themen
- Übungsaufgaben mit Lösungen



- *An der Prüfung werden wir 1-2 Aufgaben aus dem Buch bringen*

Relevantes für die Prüfung

Prüfungsstoff für die Endprüfung (in der Prüfungssession 2019) schliesst ein

- Vorlesungsinhalt (Vorlesung, Handout) und
- Übungsinhalte (Übungsstunden, Übungsaufgaben).

Relevantes für die Prüfung

Prüfung ist schriftlich - kann eventuell am Computer stattfinden

Es wird sowohl praktisches Wissen (Programmierfähigkeit als auch theoretisches Wissen (Hintergründe, Systematik) geprüft.

Unser Angebot

- Ihre Programmierübungen werden (halb)automatisch bewertet. Durch Bearbeitung der wöchentlichen Übungsserien kann ein Bonus von maximal 0.25 Notenpunkten erarbeitet werden, der an die Prüfung mitgenommen wird.
- Der Bonus ist proportional zur erreichten Punktzahl von speziell markierten Bonus-Aufgaben, wobei volle Punktzahl einem Bonus von 0.25 entspricht. Die Zulassung zu speziell markierten Bonusaufgaben hängt von der erfolgreichen Absolvierung anderer Übungsaufgaben ab. Der erreichte Notenbonus verfällt, sobald die Vorlesung neu gelesen wird.

Akademische Lauterkeit

Regel: Sie geben nur eigene Lösungen ab, welche Sie selbst verfasst und verstanden haben.

Wir prüfen das (zum Teil automatisiert) nach und behalten uns insbesondere mündliche Prüfungsgespräche vor.

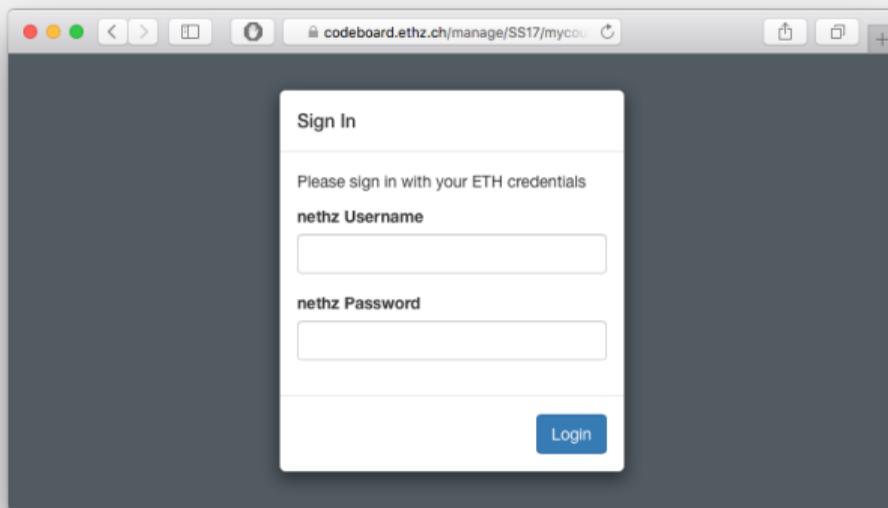
Sollten Sie zu einem Gespräch eingeladen werden: geraten Sie nicht in Panik. Es gilt primär die Unschuldsvermutung. Wir wollen wissen, ob Sie verstanden haben, was Sie abgegeben haben.

Einschreibung in Übungsgruppen - I

- Besuchen Sie

`http://expert.ethz.ch/enroll/AS18/inf1baug`

- Loggen Sie sich mit Ihrem nethz Account ein.

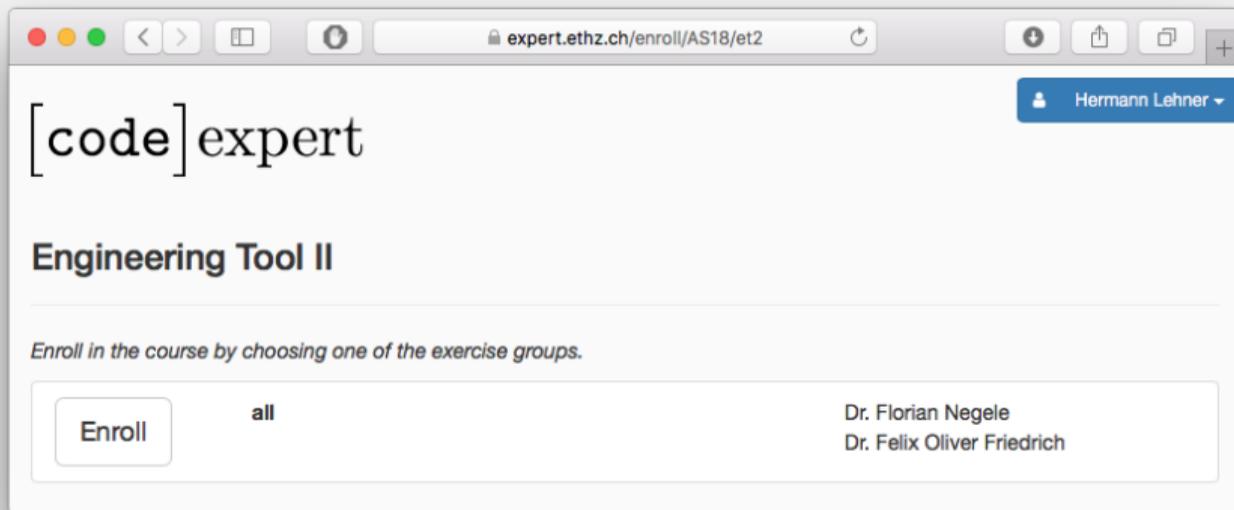


The image shows a browser window with the address bar containing `codeboard.ethz.ch/manage/SS17/mycoo`. The main content area displays a 'Sign In' form with the following elements:

- Title: Sign In
- Instruction: Please sign in with your ETH credentials
- Label: nethz Username
- Input field: A text box for the username.
- Label: nethz Password
- Input field: A text box for the password.
- Button: A blue button labeled 'Login'.

Einschreibung in Übungsgruppen - II

Schreiben Sie sich im folgenden Dialog in eine Übungsgruppe ein.



The screenshot shows a web browser window with the URL `expert.ethz.ch/enroll/AS18/et2`. The page content includes the logo `[code]expert`, the course title **Engineering Tool II**, and a user profile for Hermann Lehner. Below the course title, there is a text instruction: *Enroll in the course by choosing one of the exercise groups.* At the bottom, there is a table with one row containing an **Enroll** button, the text **all**, and the names of the lecturers: **Dr. Florian Negele** and **Dr. Felix Oliver Friedrich**.

Enroll	all	Dr. Florian Negele Dr. Felix Oliver Friedrich
---------------	------------	--

Übersicht

[code]expert

Felix Oliver Friedrich ▾

Autum 2017 ▾

Enrolled Courses

My Exercise Groups

My Courses

Demo Course Demo Group - Dr. Hermann Lehner [change...](#)

Coding Demo Exercise

Earned XP

Submissions

Handout Date

Due Date

Tasks

Solutions

1,000 / 1,000

9. Sep. 2017 00:00

31. Dez. 2027 00:00

Quadratic Equations In C++

1,000 ✓

100%

Hand In now

Markdown Editor Manual

Submissions

Handout Date

Due Date

Tasks

Solutions

1. Aug. 2017 00:00

1. Aug. 2017 00:01

Basic Markdown Syntax

Code Blocks and Inline Code

Programmierübung

```
1 #include <iostream>
2
3 int main () {
4     int min; int max;
5     std::cin >> min;
6     max = min;
7     for (int i = 0; i < 8; ++i){ // (there is a bug here)
8         int v;
9         std::cin >> v;
10        if (v<min) min = v;
11        if (v>max) max = v;
12    }
13    std::cout << min << "/" << max << std::endl;
14 }
```

A

B

C

>_ Console

Felix Oliver Friedrich

Status Not submitted yet

Create new Submission

Minimax

Write a program that outputs the minimum and maximum of a series of ten integers.

- Input format: 10 consecutive integers
number:int, example:

```
0 100250 45 0 0 1 -1000001 45 -25065 1
```

- Expected output format: minimum:int
"/" maximum:int, example:

```
-1000001/100251
```

Programmierübung

```
1 #include <iostream>
2
3 int main () {
4     int min; int max;
5     std::cin >> min;
6     max = min;
7     for (int i = 0; i < 8; ++i){ // (there is a bug here)
8         int v;
9         std::cin >> v;
10        if (v<min) min = v;
11        if (v>max) max = v;
12    }
13    std::cout << min << "/" << max << std::endl;
14 }
```

A
B
C

A: compile
B: run
C: test

>_ Console

Felix Oliver Friedrich

Status Not submitted yet

Create new Submission

Minimax

Write a program that outputs the minimum and maximum of a series of ten integers.

- Input format: 10 consecutive integers
number:int, example:

```
0 100250 45 0 0 1 -1000001 45 -25065 1
```

- Expected output format: minimum:int
"/" maximum:int, example:

```
-1000001/100251
```

Programmierübung

```
1 #include <iostream>
2
3 int main () {
4     int min; int max;
5     std::cin >> min;
6     max = min;
7     for (int i = 0; i < 8; ++i){ // (there is
8         int v;
9         std::cin >> v;
10        if (v<min) min = v;
11        if (v>max) max = v;
12    }
13    std::cout << min << "/" << max << std::endl;
14 }
```

D: Beschreibung
E: History

maximum and
maximum of a series of ten integers.

- Input format: 10 consecutive integers
number:int , example:
0 100250 45 0 0 1 -1000001 45 -25065 1
- Expected output format: minimum:int
"/" maximum:int , example:
-1000001/100251

A
B
C

>_ Console

Testen und Abgeben

The screenshot shows a C++ IDE interface with three main panels:

- Project Files:** Shows a file named `main.cpp`.
- Minimax - Student Attempt:** Contains the following C++ code:

```
1 #include <iostream>
2
3 int main () {
4     int min; int max;
5     std::cin >> min; std::cin >> max;
6     max = min-1;
7     for (int i = 0; i < 8; ++i){
8         int v;
9         std::cin >> v;
10        if (v<min) min = v;
11        if (v>max) max = v;
12    }
13    std::cout << min << "/" << max << std::endl;
14 }
```
- Console:** Shows test results:

```
Running tests.....
min_first passed
min_last passed
min_middle passed
max_first failed
input:
100251 -25065 45 -1000001 1 0 0 45 100250 0
expected output:
-1000001/100251
actual output:
-1000001/100250
-----
max_last passed
max_middle passed
unique passed

Tests result: passed 6 of 7 / score: 86% [ ]
```
- Right Panel:** Shows user information for Felix Oliver Friedrich, status "Not submitted yet", and a "Create new Submission" button. It also includes "Filter Snapshots" and "Create Snapshot" options, along with "First Working Version" (2 minutes ago) and "Initial Snapshot" (13 minutes ago) sections.

Testen und Abgeben

The screenshot displays a C++ IDE interface with three main sections:

- Project Files:** Shows a file named `main.cpp`.
- Minimax - Student Attempt:** Contains the following C++ code:

```
1 #include <iostream>
2
3 int main () {
4     int min; int max;
5     std::cin >> min; std::cin >> max;
6     max = min-1;
7     for (int i = 0; i < 8; ++i){
8         int v;
9         std::cin >> v;
10        if (v<min) min = v;
11        if (v>max) max = v;
12    }
13    std::cout << min << "/" << max << std::endl;
14 }
```
- Console:** Shows test results for a Minimax problem:

```
Running tests.....
min_first passed
min_last passed
min_middle passed
max_first failed
input:
100251 -25065 45 -1000001 1 0 0 45 100250 0
expected output:
-1000001/100251
actual output:
-1000001/100250
-----
max_last passed
max_middle passed
unique passed

Tests result: passed 6 of 7 / score: 86% [ ]
```
- Submission Panel:** Shows the user `Felix Oliver Friedrich` with a status of `Not submitted yet`. It includes a `Create new Submission` button and options to `Filter Snapshots`, `Create Snapshot`, `First Working Version` (2 minutes ago), and `Initial Snapshot` (13 minutes ago).

Testen

Testen und Abgeben

The screenshot displays a coding environment with the following components:

- Project Files:** A sidebar on the left shows a folder named "Project Files" containing a file named "main.cpp".
- Code Editor:** The main area shows C++ code for finding the minimum and maximum of an array. The code is:

```
1 #include <iostream>
2
3 int main () {
4     int min; int max;
5     std::cin >> min; std::cin >> max;
6     max = min-1;
7     for (int i = 0; i < 8; ++i){
8         int v;
9         std::cin >> v;
10        if (v<min) min = v;
11        if (v>max) max = v;
12    }
13    std::cout << min << "/" << max << std::endl;
14 }
```
- Test Results:** Below the code, a terminal window shows the output of running tests:

```
Running tests.....
min_first passed
min_last passed
min_middle passed
max_first failed
input:
100251 -25065 45 -1000001 1 0 0 45 100250 0
expected output:
-1000001/100251
actual output:
-1000001/100250
-----
max_last passed
max_middle passed
unique passed

Tests result: passed 6 of 7 / score: 86% [ ]
```
- Submission Panel:** On the right, a panel for user "Felix Oliver Friedrich" shows submission options. A red box labeled "Abgeben" (Submit) points to the "Create new Submission" button. Below it, there are sections for "Filter Snapshots" (with a "Create Snapshot" button), "First Working Version" (2 minutes ago), and "Initial Snapshot" (13 minutes ago).

Wo ist der Save Knopf?

- Das Filesystem ist transaktionsbasiert und es wird laufend gespeichert ("autosave"). Beim Öffnen eines Projektes findet man immer den zuletzt gesehenen Zustand wieder.
- Der derzeitige Stand kann als (benannter) *Snapshot* festgehalten werden. Zu gespeicherten Snapshots kann jederzeit zurückgekehrt werden.
- Der aktuelle Stand kann als Snapshot abgegeben (submitted) werden. Zudem kann jeder gespeicherte Snapshot abgegeben werden.

Snapshots

The screenshot displays a C++ IDE interface. On the left, a file explorer shows 'Project Files' and 'main.cpp'. The main editor area shows the following code:

```
1 #include <iostream>
2
3 int main () {
4     int min; int max;
5     std::cin >> min; std::cin >> max;
6     max = min;
7     for (int i = 0; i < 8; ++i){ // (there is a bug here)
8         int v;
9         std::cin >> v;
10        if (v<min) min = v;
11        if (v>max) max = v;
12    }
13    std::cout << min << "/" << max << std::endl;
14 }
```

Below the code, the console shows the following test results:

```
Running tests.....
min_first passed
min_last passed
min_middle passed
max_first passed
max_last passed
max_middle passed
unique passed

Tests result: passed 7 of 7 / score: 100% [ ]
[]
```

On the right side, a sidebar contains the following information:

- History: Go back to current version
- User: Felix Oliver Friedrich
- Status: Already submitted
- Buttons: Create new Submission, Create Snapshot
- Filter Snapshots
- Really Working Version (2 minutes ago)
- First Working Version (6 minutes ago)
- Initial Snapshot (16 minutes ago)

At the bottom left, the console prompt is '>_ Console'.

Snapshots

The screenshot displays a code editor interface with the following components:

- Project Files:** A sidebar on the left showing a folder named "Project Files" containing a file named "main.cpp".
- Code Editor:** The main area shows C++ code for a program named "Minimax - Student Attempt". The code includes headers, variable declarations, and a loop with a comment indicating a bug: "For (int i = 0; i < 8; ++i){ // (there is a bug here)".
- Console:** Below the code editor, a console window shows the output of running tests: "Running tests.....", followed by a list of test cases (min_first, min_last, min_middle, max_first, max_last, max_middle, unique) all marked as "passed". A summary line reads "Tests result: passed 7 of 7 / score: 100%".
- History Panel:** On the right side, a panel titled "History" shows the user's name "Felix Oliver Friedrich" and the status "Already submitted". It features a "Create new Submission" button and a "Filter Snapshots" section with a "Create Snapshot" button. Below this, three snapshots are listed: "Really Working Version" (2 minutes ago), "First Working Version" (6 minutes ago), and "Initial Snapshot" (16 minutes ago). Each snapshot has a magnifying glass icon and a download icon.
- Callout:** A pink rectangular box with the text "Snapshot betrachten" is overlaid on the code editor. A red arrow points from this box to the "First Working Version" entry in the history panel.

Snapshots

The screenshot displays a code editor with a C++ program named "Minimax - Student Attempt". The code is as follows:

```
1 #include <iostream>
2
3 int main () {
4     int min; int max;
5     std::cin >> min; std::cin >> max;
6     max = min;
7     for (int i = 0; i < 8; ++i){ // (there is a bug here)
8         int v;
9         std::cin >> v;
10        if (v<min) min = v;
11        if (v>max) max = v;
12    }
13    std::cout << min << " " << max << endl;
14 }
```

Below the code, the test results are shown:

```
Running tests.....
min_first passed
min_last passed
min_middle passed
max_first passed
max_last passed
max_middle passed
unique passed

Tests result: passed 7 of 7 / score: 100% [██████████]
[]
```

The right sidebar shows the "History" section with the following items:

- Really Working Version (2 minutes ago)
- First Working Version (6 minutes ago)
- Initial Snapshot (16 minutes ago)

Annotations in red text with arrows point to specific elements:

- "Snapshot betrachten" points to the "First Working Version" entry.
- "Abgabe" points to the "Initial Snapshot" entry.
- "Zurück zu" points to the "Initial Snapshot" entry.

2. Java Einführung

Programmieren – Ein erstes Java Programm

Was braucht es zum Programmieren?

- **Editor:** Programm zum Ändern, Erfassen und Speichern vom Java-Programmtext
- **Compiler:** Programm zum Übersetzen des Programmtexts in Maschinsprache

Was braucht es zum Programmieren?

- **Computer:** Gerät zum Ausführen von Programmen in Maschinensprache
- **Betriebssystem:** Programm zur Organisation all dieser Abläufe (Dateiverwaltung, Editor-, Compiler- und Programmaufruf)

Deutsch vs. Programmiersprache

Deutsch

*Es ist nicht genug zu wissen,
man muss auch anwenden.
(Johann Wolfgang von Goethe)*

Java / C / C++

```
// computation  
int b = a * a; // b = a2  
b = b * b;    // b = a4
```

Syntax und Semantik

- Programme müssen, wie unsere Sprache, nach gewissen Regeln geformt werden.
 - **Syntax**: Zusammenfügingsregeln für elementare Zeichen (Buchstaben).
 - **Semantik**: Interpretationsregeln für zusammengefügte Zeichen.

Syntax und Semantik

- Entsprechende Regeln für ein Computerprogramm sind einfacher, aber auch strenger, denn Computer sind vergleichsweise dumm.

Syntax und Semantik von Java

Syntax

- Was *ist* ein Java Programm?
- Ist es *grammatikalisch* korrekt?

Semantik

- Was *bedeutet* ein Programm?
- Welchen Algorithmus realisiert ein Programm?

Erstes Java Programm

```
// input
Out.print("Compute a^8 for a= ?");
int a;
a = In.readInt();
// computation
int b = a * a; // b = a^2
b = b * b; // b = a^4
// output b*b, i.e. a^8
Out.println(a + "^8 = " + b*b);
```

Erstes Java Programm

```
public static void main(String[] args) {  
    // input  
    Out.print("Compute a^8 for a= ?");  
    int a;  
    a = In.readInt();  
    // computation  
    int b = a * a; // b = a^2  
    b = b * b; // b = a^4  
    // output b*b, i.e. a^8  
    Out.println(a + "^8 = " + b*b);  
}
```

Erstes Java Programm

```
// Program to raise a number to the eighth power
public class Main {

    public static void main(String[] args) {
        // input
        Out.print("Compute a^8 for a= ?");
        int a;
        a = In.readInt();
        // computation
        int b = a * a; // b = a^2
        b = b * b; // b = a^4
        // output b*b, i.e. a^8
        Out.println(a + "^8 = " + b*b);
    }
}
```

Erstes Java Programm

```
// Program to raise a number to the eighth power
```

```
public class Main { ← Klasse: Ein Programm
```

```
    public static void main(String[] args) { ← Methode: benannte  
                                           Folge von Anweisungen.
```

```
        // input
```

```
        Out.print("Compute a^8 for a= ?");
```

```
        int a;
```

```
        a = In.readInt();
```

```
        // computation
```

```
        int b = a * a; // b = a^2
```

```
        b = b * b; // b = a^4
```

```
        // output b*b, i.e. a^8
```

```
        Out.println(a + "^8 = " + b*b);
```

```
    }
```

```
}
```

Verhalten eines Programmes

Zur Compilationszeit:

- vom Compiler akzeptiertes Programm (syntaktisch korrektes Java)
- Compiler-Fehler

Verhalten eines Programmes

Zur Laufzeit:

- korrektes Resultat
- inkorrektes Resultat
- Programmabsturz
- Programm *terminiert* nicht (Endlosschleife)

Kommentare

```
// Program to raise a number to the eighth power
public class Main {

    public static void main(String[] args) {
        // input
        Out.print("Compute a^8 for a= ?");
        int a;
        a = In.readInt();
        // computation
        int b = a * a; // b = a^2
        b = b * b; // b = a^4
        // output b*b, i.e. a^8
        Out.println(a + "^8 = " + b*b);
    }
}
```

Kommentare

```
// Program to raise a number to the eighth power  
public class Main {
```

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

```
        // input
```

```
        Out.print("Compute a^8 for a= ?");
```

```
        int a;
```

```
        a = In.readInt();
```

```
        // computation
```

```
        int b = a * a; // b = a^2
```

```
        b = b * b; // b = a^4
```

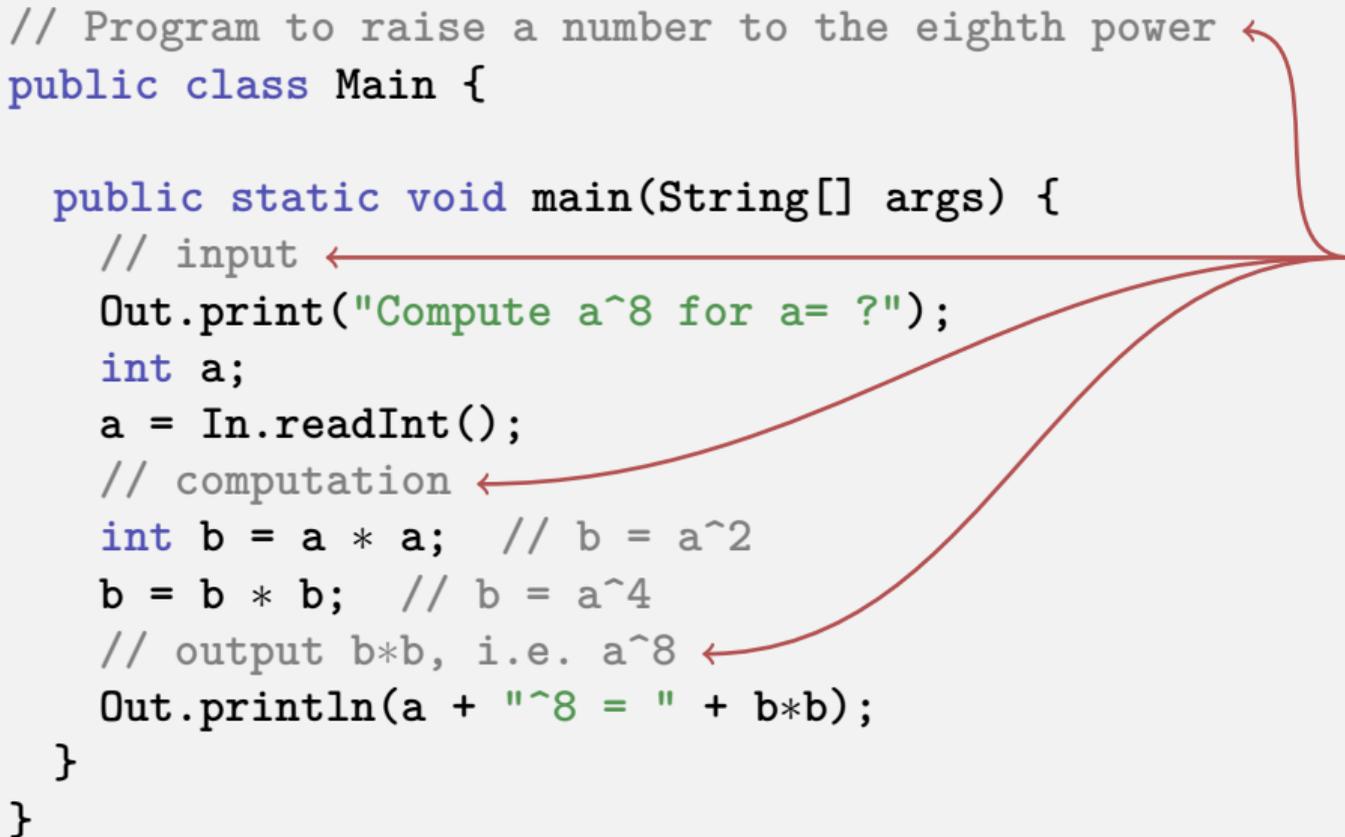
```
        // output b*b, i.e. a^8
```

```
        Out.println(a + "^8 = " + b*b);
```

```
    }
```

```
}
```

Kommentare



Kommentare und Layout

Dem Compiler ist's egal...

```
public class Main{public static void main(String[] args){Out.print
("Compute a^8 for a= ?");int a;a = In.readInt();int b = a*a;b =
b * b;Out.println(a + "^8 = " + b*b);}}
```

Kommentare und Layout

Dem Compiler ist's egal...

```
public class Main{public static void main(String[] args){Out.print  
("Compute a^8 for a= ?");int a;a = In.readInt();int b = a*a;b =  
b * b;Out.println(a + "^8 = " + b*b);}}
```

... uns aber nicht!

Anweisungen (Statements)

```
// Program to raise a number to the eighth power
public class Main {

    public static void main(String[] args) {
        // input
        Out.print("Compute a^8 for a= ?");
        int a;
        a = In.readInt();
        // computation
        int b = a * a; // b = a^2
        b = b * b; // b = a^4
        // output b*b, i.e. a^8
        Out.println(a + "^8 = " + b*b);
    }
}
```

Anweisungen (Statements)

```
// Program to raise a number to the eighth power  
public class Main {
```

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

```
        // input
```

```
        Out.print("Compute a^8 for a= ?");
```

```
        int a;
```

```
        a = In.readInt();
```

```
        // computation
```

```
        int b = a * a; // b = a^2
```

```
        b = b * b; // b = a^4
```

```
        // output b*b, i.e. a^8
```

```
        Out.println(a + "^8 = " + b*b);
```

```
    }
```

```
}
```

Ausdrucksanweisungen

Anweisungen – Werte und Effekte

```
// Program to raise a number to the eighth power
public class Main {

    public static void main(String[] args) {
        // input
        Out.print("Compute a^8 for a= ?");
        int a;
        a = In.readInt();
        // computation
        int b = a * a;
        b = b * b;
        // output b*b, i.e. a^8
        Out.println(a + "^8 = " + b*b);
    }
}
```

Effekt: Ausgabe des Strings Compute ...

Effekt: Eingabe einer Zahl und Speichern in a

*Effekt: Speichern des berechneten Wertes von a*a in b*

*Effekt: Speichern des berechneten Wertes von b*b in b*

*Effekt: Ausgabe des Wertes von a und des berechneten Wertes von b*b*

Anweisungen – Variablendefinitionen

```
// Program to raise a number to the eighth power
public class Main {

    public static void main(String[] args) {
        // input
        Out.print("Compute a^8 for a= ?");
        int a;
        a = In.readInt();
        // computation
        int b = a * a; // b = a^2
        b = b * b; // b = a^4
        // output b*b, i.e. a^8
        Out.println(a + "^8 = " + b*b);
    }
}
```

Anweisungen – Variablendefinitionen

```
// Program to raise a number to the eighth power
public class Main {

    public static void main(String[] args) {
        // input
        Out.print("Compute a^8 for a= ?");
        int a; ← Deklarationsanweisungen
        a = In.readInt();
        // computation
        int b = a * a; ← // b = a^2
        b = b * b; // b = a^4
        // output b*b, i.e. a^8
        Out.println(a + "^8 = " + b*b);
    }
}
```

Typ-
namen

Deklarationsanweisungen

Variablen

- repräsentieren (wechselnde) Werte,
- haben
 - *Name*
 - *Typ*
 - *Wert*
 - *Adresse*
- sind im Programmtext "sichtbar".

Variablen

- repräsentieren (wechselnde) Werte,
- haben
 - *Name*
 - *Typ*
 - *Wert*
 - *Adresse*
- sind im Programmtext "sichtbar".

Beispiel

`int a;` definiert Variable mit

- Name: a
- Typ: `int`
- Wert: (vorerst) undefiniert
- Adresse: durch Compiler bestimmt

Objekte

- repräsentieren Werte im Hauptspeicher
- haben *Typ*, *Adresse* und *Wert* (Speicherinhalt an der Adresse),
- können benannt werden (Variable) ...
- ... aber auch anonym sein.

Anmerkung

Ein Programm hat eine *feste* Anzahl von Variablen. Um eine variable Anzahl von Werten behandeln zu können, braucht es "anonyme" Adressen, die über temporäre Namen angesprochen werden können.

Ausdrücke (Expressions)

- repräsentieren *Berechnungen*

Ausdrücke (Expressions)

- repräsentieren *Berechnungen*
- sind entweder **primär** (b)

Ausdrücke (Expressions)

- repräsentieren *Berechnungen*
- sind entweder **primär** (b)
- oder **zusammengesetzt** ($b*b$). . .

Ausdrücke (Expressions)

- repräsentieren *Berechnungen*
- sind entweder **primär** (b)
- oder **zusammengesetzt** ($b*b$)...
- ... aus anderen Ausdrücken, mit Hilfe von **Operatoren**

Ausdrücke (Expressions)

- repräsentieren *Berechnungen*
- sind entweder **primär** (b)
- oder **zusammengesetzt** ($b*b$)...
- ... aus anderen Ausdrücken, mit Hilfe von **Operatoren**

Analogie: Baukasten

Ausdrücke (Expressions)

```
// input
Out.print("Compute a^8 for a= ?");
int a;
a = In.readInt();

// computation
int b = a * a; // b = a^2
b = b * b;    // b = a^4

// output b*b, i.e. a^8
Out.println(a + "^8 = " + b * b | ); ||
```

Ausdrücke (Expressions)

```
// input
Out.print("Compute a^8 for a= ?");
int a;
a = In.readInt();
// computation
int b = a * a; // b = a^2
b = b * b; // b = a^4
// output b*b, i.e. a^8
Out.println(a + "^8 = " + b * b);
```

Variablenname, primärer Ausdruck

Variablenname, primärer Ausdruck

Ausdrücke (Expressions)

```
// input
Out.print("Compute a^8 for a= ?");
int a;
a = In.readInt();

// computation
int b = a * a; // b = a^2
b = b * b;     // b = a^4

// output
Out.println(a + "^8 = " + b * b | ); ||
```

Zusammengesetzter Ausdruck

Operatoren und Operanden

```
// input
Out.print("Compute a^8 for a= ?");
int a;
a = In.readInt();

// computation
int b = a * a; // b = a^2
b = b * b;     // b = a^4

// output b*b, i.e. a^8
Out.println(a + "^8 = " + b * b );
```

Operatoren und Operanden

```
// input
Out.print("Compute a^8 for a= ?");
int a;
a = In.readInt();

// computation
int b = a * a; // b = a^2
b = b * b;     // b = a^4

// output b*b, i.e. a^8
Out.println(a + "^8 = " + b * b );
```

Linker Operand (Variable)

Rechter Operand (Ausdruck)

Operatoren und Operanden

```
// input
Out.print("Compute a^8 for a= ?");
int a;
a = In.readInt();
```

Linker Operand (Variable)

```
// computation
int b = a * a; // b = a^2
```

Rechter Operand (Ausdruck)

```
b = b * b; // b = a^4
```

```
// ou  
Zuweisungsoperator ;
```

```
Out.println(a + "^8 = " + b * b );
```

Multiplikationsoperator