

# **Informatik II**

**Vorlesung am D-BAUG der ETH Zürich**

**Felix Friedrich**

**FS 2017**

# Willkommen zur Vorlesung !

Vorlesungshomepage:

<http://lec.inf.ethz.ch/baug/informatik2/2017/>

# Das Team

Dozent	Felix Friedrich
Chefassistent	David Sidler
Assistenten	Giuseppe Accaputo Max Rossmannek Patrick Gruntz Thilo Weghorn Tobias Klenze

# 1. Einführung

Überblick, der Universelle Computer (Turing Maschine), Euklidischer Algorithmus

# Ziel der Vorlesung

*Problemlösen mit dem Computer*

# Ziel der Vorlesung

## *Problemlösen mit dem Computer*

Hilfsmittel:

- *Objektorientierte Programmiersprache (Java)*

## *Problemlösen mit dem Computer*

Hilfsmittel:

- *Objektorientierte Programmiersprache* (Java)
- Tools wie Matlab und *Datenbanken* (SQL)

## *Problemlösen mit dem Computer*

### Methodik

- *Kernthemen:* Algorithmen und Datenstrukturen , (Objektorientierte) Programmierung, Datenbanken.

## *Problemlösen mit dem Computer*

### Methodik

- *Kernthemen*: Algorithmen und Datenstrukturen , (Objektorientierte) Programmierung, Datenbanken.
- *Fallstudien*: Interessante Probleme aus der Informatik und angrenzenden Gebieten.

# Inhalte der Vorlesung

## Datenstrukturen / Algorithmen

Begriff der Invariante, Kostenmodell, Landau Symbole

Randomisierte Algorithmen (MCMC)

Graphen, Kürzeste Wege

Suchen und Auswahl, Sortieren

Dynamic Programming

Verkettete Strukturen    Heaps

Wörterbücher: Hashing und Suchbäume

## Programmieren mit Java

Java Basics, Rekursion

Arrays, Wert-/Referenzsemantik

Exceptions

Objektorientierte Programmierung

## Datenbanken

ER-Modell, Relationales Modell, SQL

# Warum Java?

Sehr weit verbreitete, moderne Sprache. Funktioniert auf vielen Systemen.

# Warum Java?

Sehr weit verbreitete, moderne Sprache. Funktioniert auf vielen Systemen.

Verbietet einige typische Fehler.

# Warum Java?

Sehr weit verbreitete, moderne Sprache. Funktioniert auf vielen Systemen.

Verbietet einige typische Fehler.

Datenbankanbindung

# Ziel der *heutigen* Vorlesung

Einführung / Wiederholung Computermodell und Algorithmus

# Ziel der *heutigen* Vorlesung

*Prozedurales* Programmieren mit Java, Pascal → Java.

# 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»

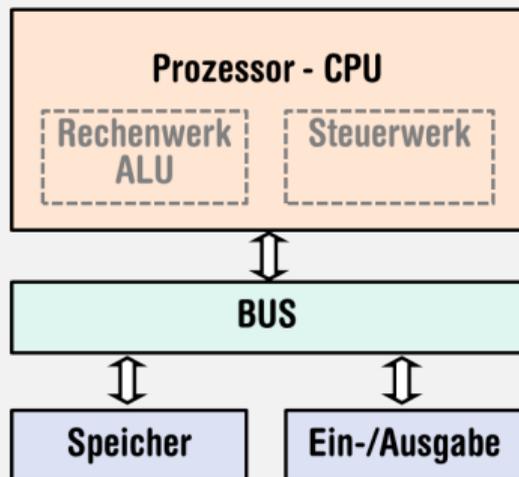


Alan Turing

# Computer – Umsetzung

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

## Von Neumann Architektur



Konrad Zuse



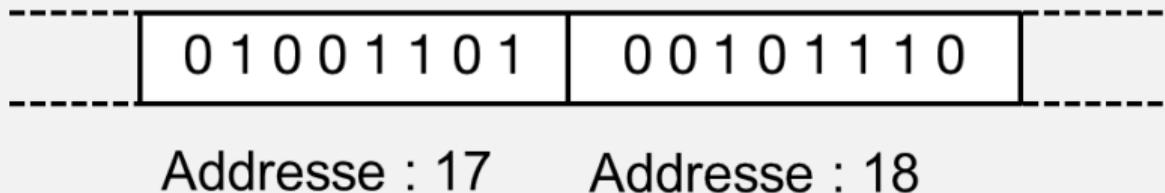
John von Neumann

# 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.



# 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

# Algorithmus: Kernbegriff der Informatik

Algorithmus:

- Handlungsanweisung zur schrittweisen Lösung eines Problems
- Ausführung erfordert keine Intelligenz, nur Genauigkeit
- 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  $b \neq 0$

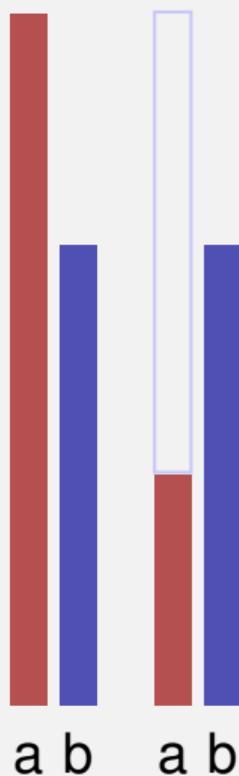
Wenn  $a > b$  dann

$$a \leftarrow a - b$$

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  $b \neq 0$

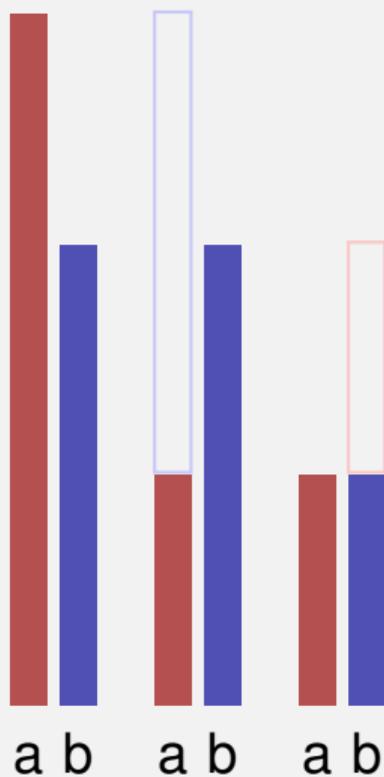
Wenn  $a > b$  dann

$$a \leftarrow a - b$$

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  $b \neq 0$

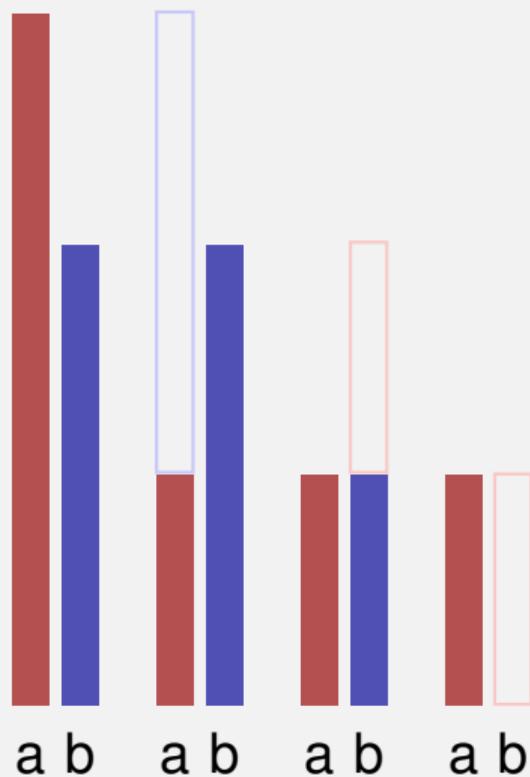
Wenn  $a > b$  dann

$$a \leftarrow a - b$$

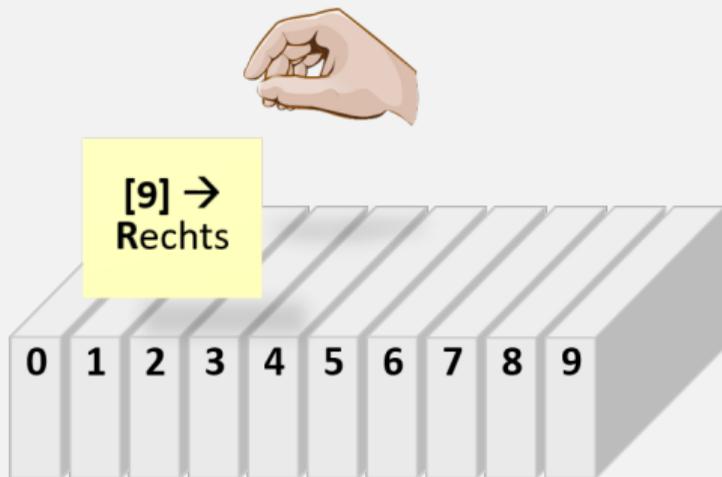
Sonst:

$$b \leftarrow b - a$$

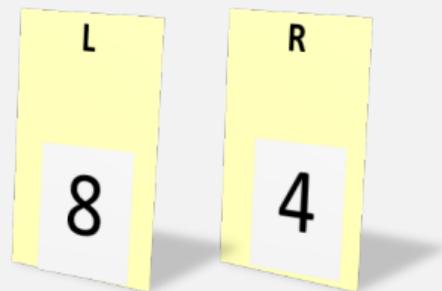
Ergebnis:  $a$ .



# Live Demo: Turing Maschine



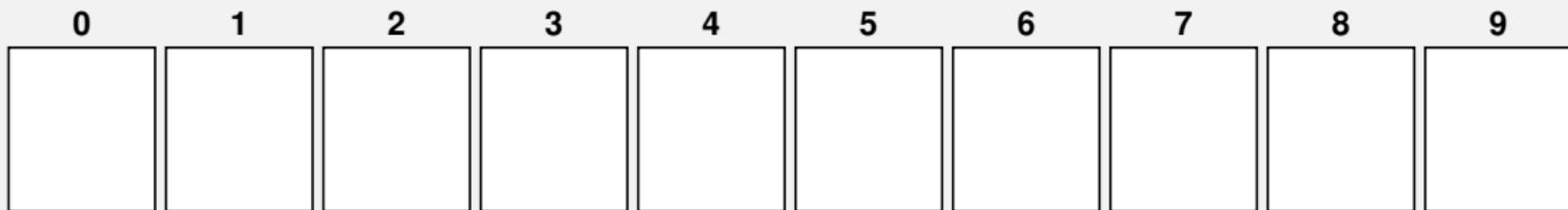
Speicher



Register

# Euklid in der Box

*Speicher*



**L**inks

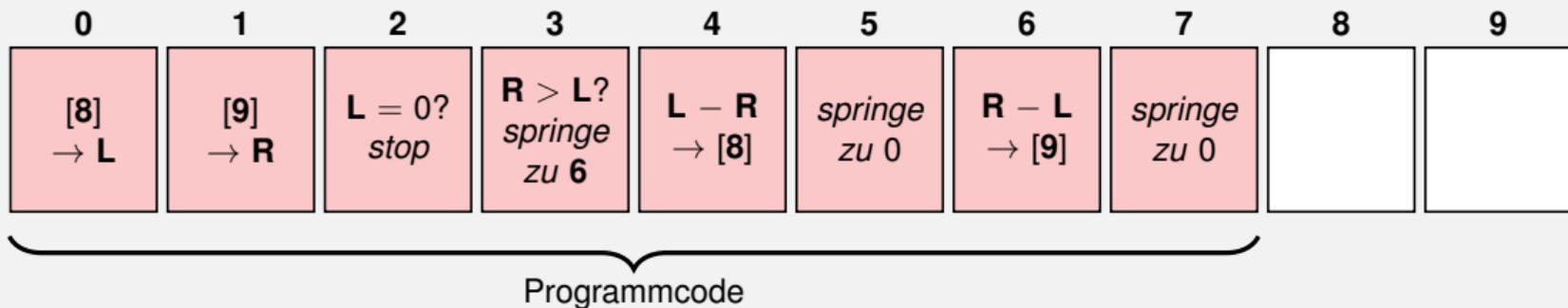
**R**echts



*Register*

# Euklid in der Box

*Speicher*



Links

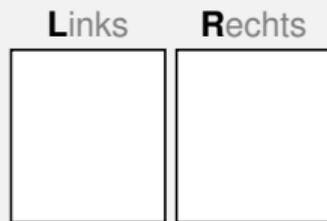
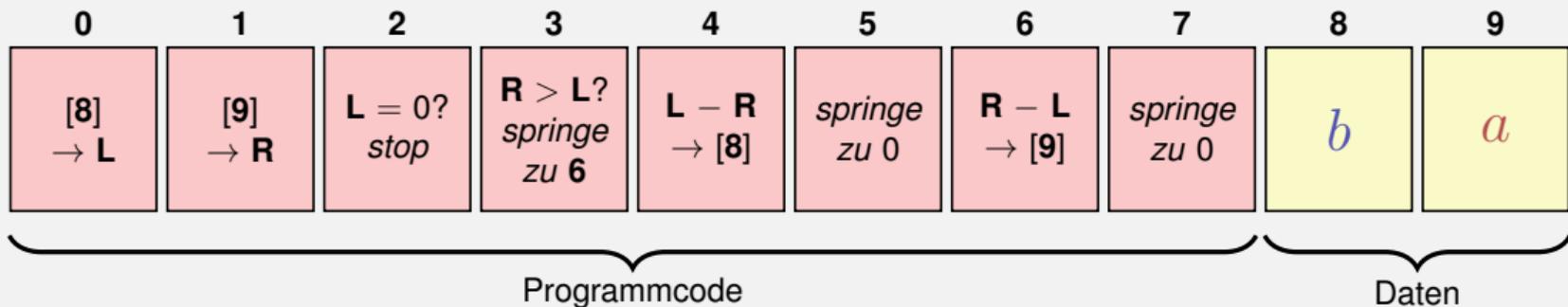
Rechts



*Register*

# Euklid in der Box

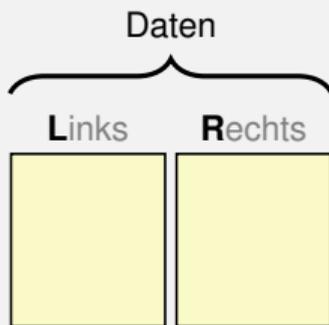
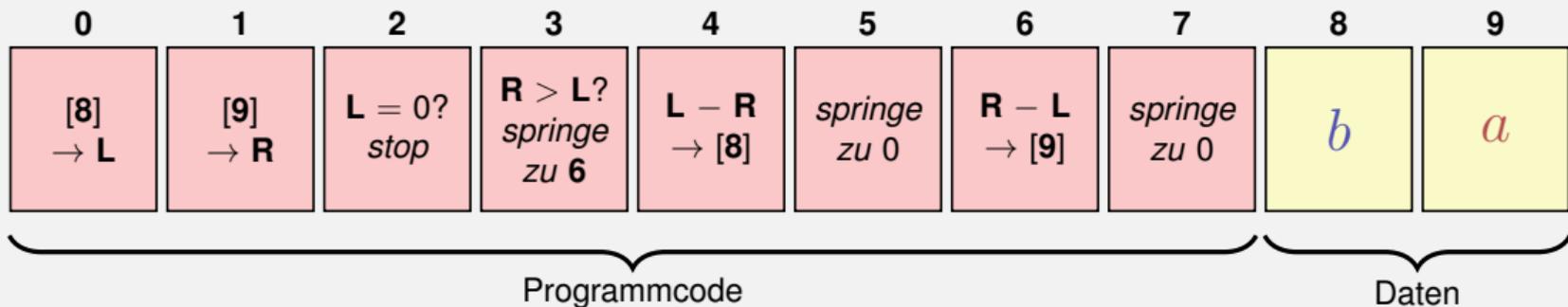
Speicher



Register

# Euklid in der Box

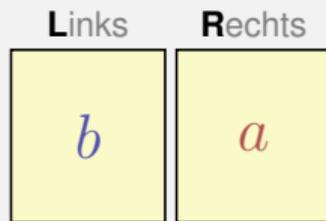
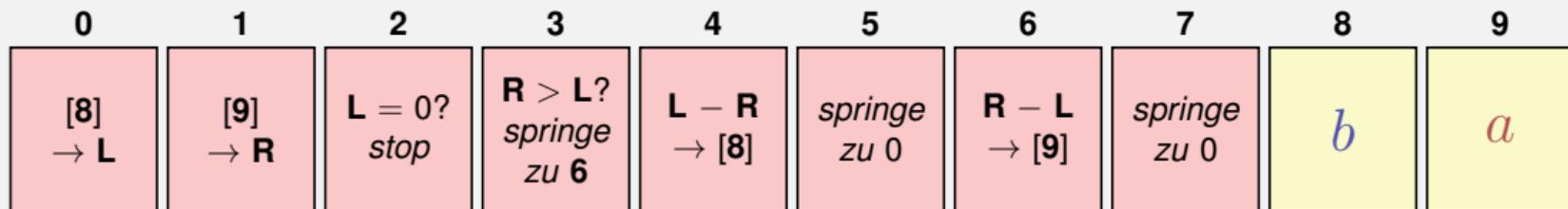
## Speicher



## Register

# Euklid in der Box

## Speicher



## Register

Solange  $b \neq 0$

Wenn  $a > b$  dann

$$a \leftarrow a - b$$

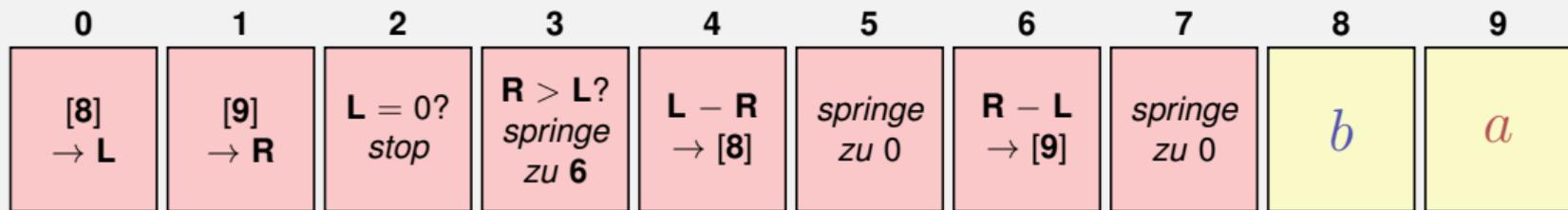
Sonst:

$$b \leftarrow b - a$$

Ergebnis:  $a$ .

# Euklid in der Box

Speicher



Solange  $b \neq 0$

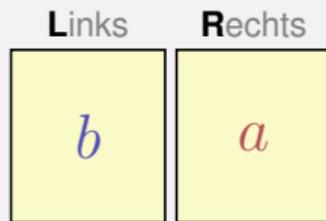
Wenn  $a > b$  dann

$$a \leftarrow a - b$$

Sonst:

$$b \leftarrow b - a$$

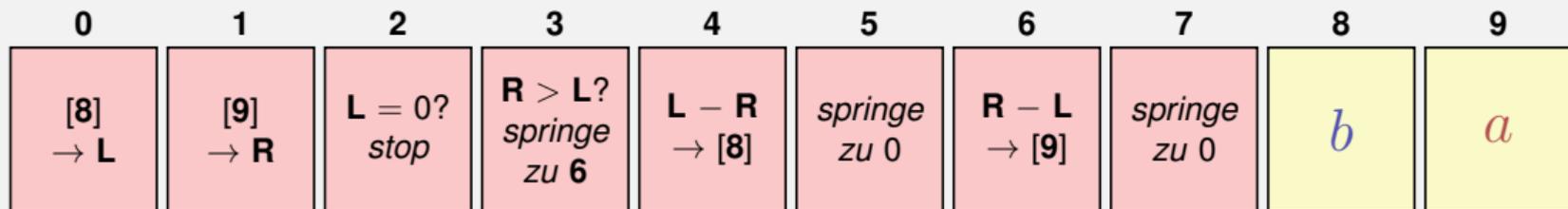
Ergebnis:  $a$ .



Register

# Euklid in der Box

Speicher



Solange  $b \neq 0$

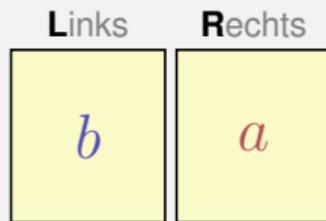
Wenn  $a > b$  dann

$$a \leftarrow a - b$$

Sonst:

$$b \leftarrow b - a$$

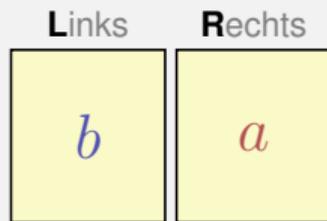
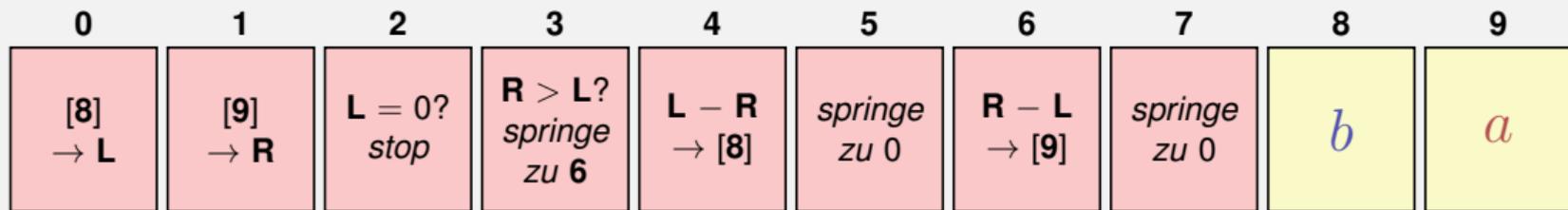
Ergebnis:  $a$ .



Register

# Euklid in der Box

Speicher



Register

Solange  $b \neq 0$

Wenn  $a > b$  dann

$$a \leftarrow a - b$$

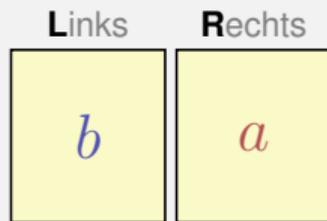
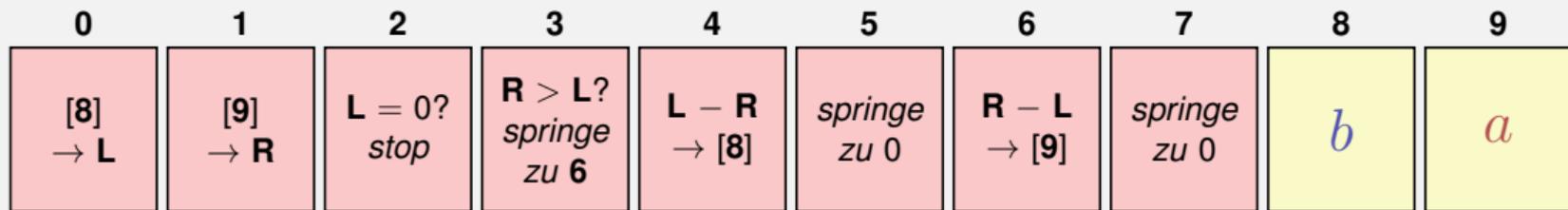
Sonst:

$$b \leftarrow b - a$$

Ergebnis:  $a$ .

# Euklid in der Box

Speicher



Register

Solange  $b \neq 0$

Wenn  $a > b$  dann

$$a \leftarrow a - b$$

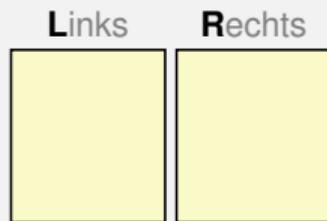
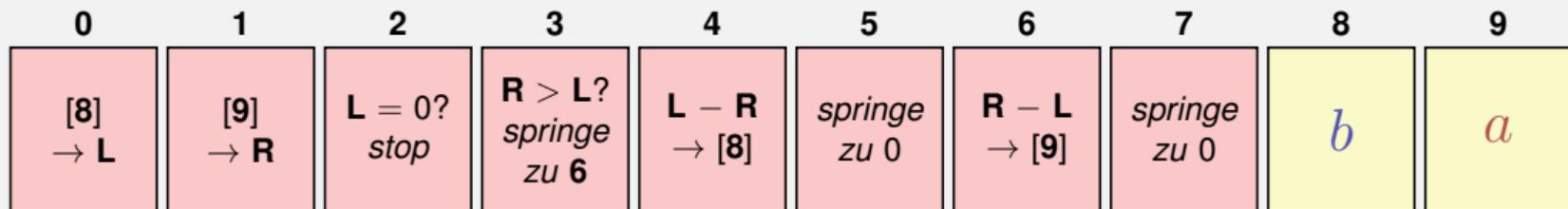
Sonst:

$$b \leftarrow b - a$$

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:  $a$ .

# Rechengeschwindigkeit

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

---

<sup>1</sup>Uniprozessor 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

---

<sup>1</sup>Uniprozessor 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.<sup>1</sup>

---

<sup>1</sup>Uniprozessor Computer bei 1GHz

# 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*

## **2. Organisation**

# Ablauf



# Ablauf



- Übungsblattausgabe zur Vorlesung (online).

# Ablauf



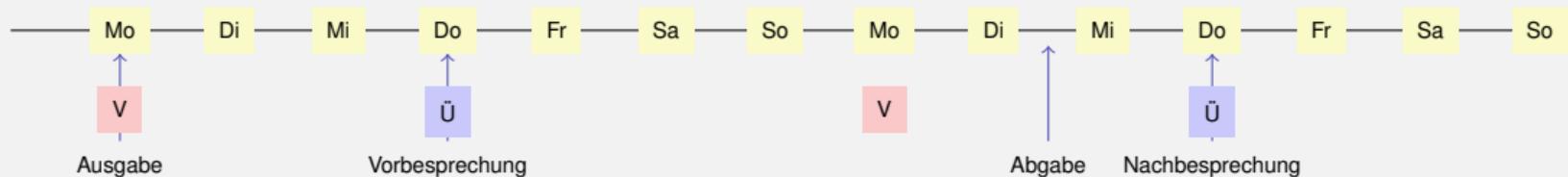
- Vorbereitung am folgenden Donnerstag.

# Ablauf



- Bearbeitung der Übung bis spätestens am Dienstag (23:59) darauf.

# Ablauf

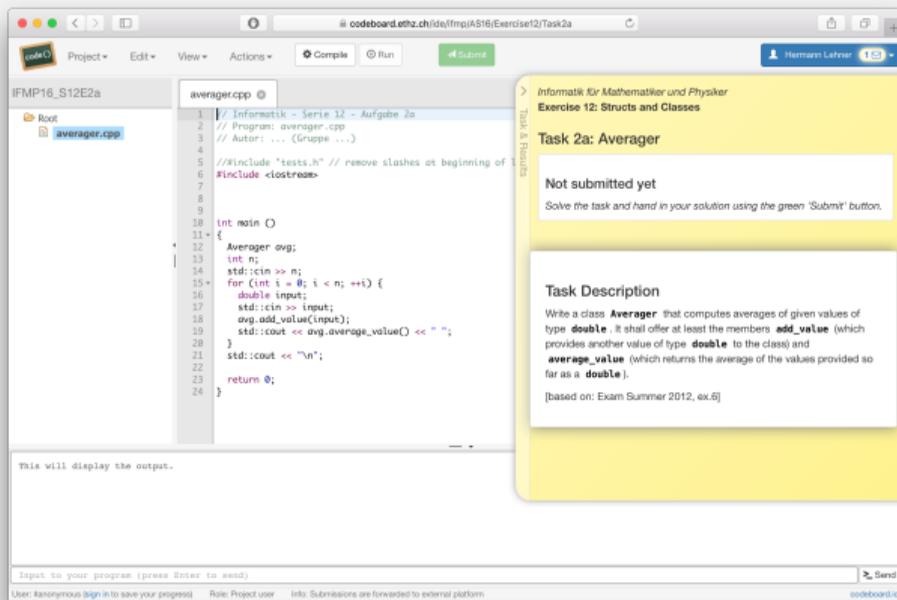


- Nachbereitung der Übung am Donnerstag. Feedback zu den Abgaben innerhalb einer Woche nach Nachbereitung.

# Codeboard

*Codeboard* ist eine Online-IDE: Programmieren im Browser!

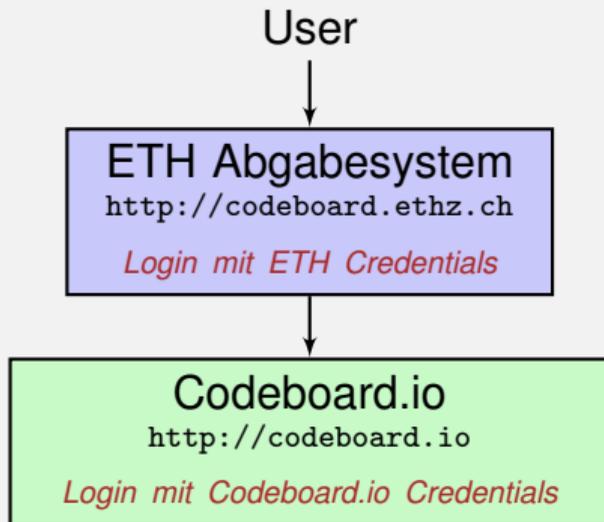
- Falls vorhanden, bringen Sie ihren Laptop/Tablet/... mit in den Unterricht.
- Sie können direkt in der Vorlesung Beispiele ausprobieren, ohne dass Sie irgendwelche Tools installieren müssen.



# Codeboard @ETH

Codeboard besteht aus zwei unabhängigen Systemen, die miteinander kommunizieren:

- **Das ETH Abgabesystem:**  
Ermöglicht es uns, ihre Aufgaben zu bewerten
- **Die Online IDE:** Die Programmierumgebung



# Codeboard

## Codeboard.io Registrierung

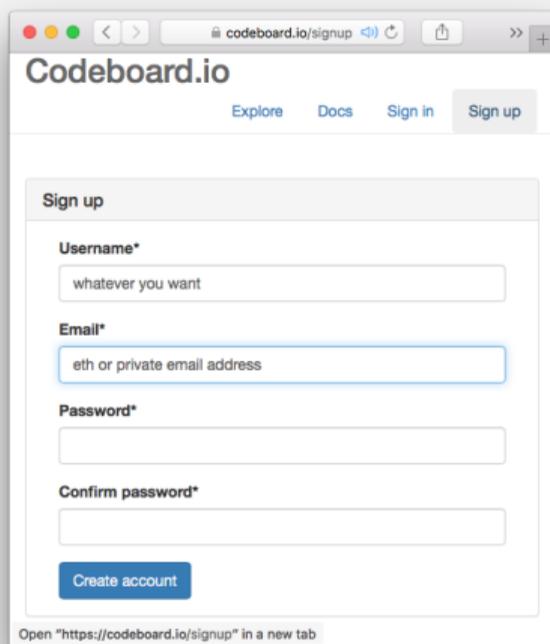
Gehen Sie auf <http://codeboard.io> und erstellen Sie dort ein Konto, bleiben Sie am besten eingeloggt.

## Einschreibung in Übungsgruppen

Gehen Sie auf <http://codeboard.ethz.ch/ifbaug2> und schreiben Sie sich dort in eine Übungsgruppe ein.

# Codeboard.io Registrierung

Falls Sie noch keinen **Codeboard.io** Account haben ...



The image shows a browser window with the URL `codeboard.io/signup`. The page title is "Codeboard.io" and the navigation menu includes "Explore", "Docs", "Sign in", and "Sign up". The "Sign up" section contains the following fields:

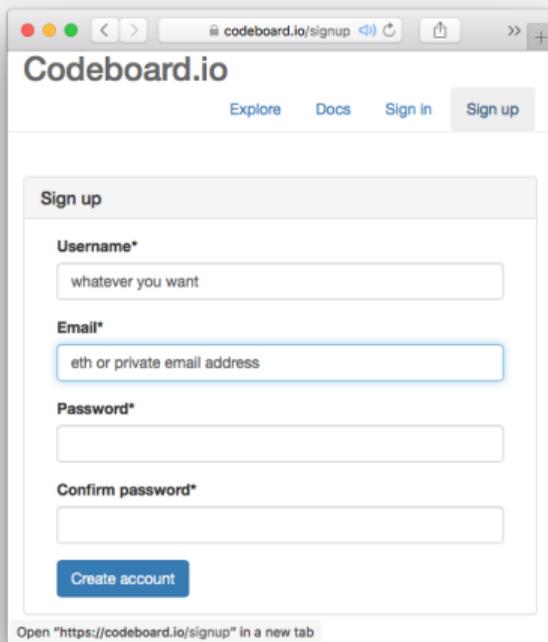
- Username\***: Input field with the placeholder text "whatever you want".
- Email\***: Input field with the placeholder text "eth or private email address".
- Password\***: Input field.
- Confirm password\***: Input field.

At the bottom of the form is a blue button labeled "Create account". Below the browser window, a status bar indicates: "Open 'https://codeboard.io/signup' in a new tab".

- Wir verwenden die Online IDE **Codeboard.io**

# Codeboard.io Registrierung

Falls Sie noch keinen **Codeboard.io** Account haben ...



The image shows a browser window with the URL `codeboard.io/signup`. The page title is "Codeboard.io" and the navigation menu includes "Explore", "Docs", "Sign in", and "Sign up". The "Sign up" section contains the following fields:

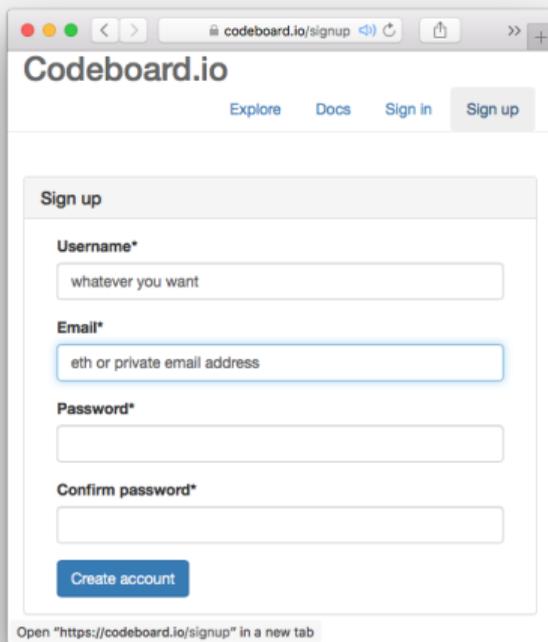
- Username\***: Input field with the placeholder text "whatever you want".
- Email\***: Input field with the placeholder text "eth or private email address".
- Password\***: Input field.
- Confirm password\***: Input field.

At the bottom of the form is a blue button labeled "Create account". Below the browser window, a status bar reads: "Open 'https://codeboard.io/signup' in a new tab".

- Wir verwenden die Online IDE **Codeboard.io**
- Erstellen Sie dort einen Account, um Ihren Fortschritt abzuspeichern und später Submissions anzuschauen

# Codeboard.io Registrierung

Falls Sie noch keinen **Codeboard.io** Account haben ...



The image shows a browser window with the URL `codeboard.io/signup`. The page title is "Codeboard.io" and the navigation menu includes "Explore", "Docs", "Sign in", and "Sign up". The "Sign up" form contains the following fields:

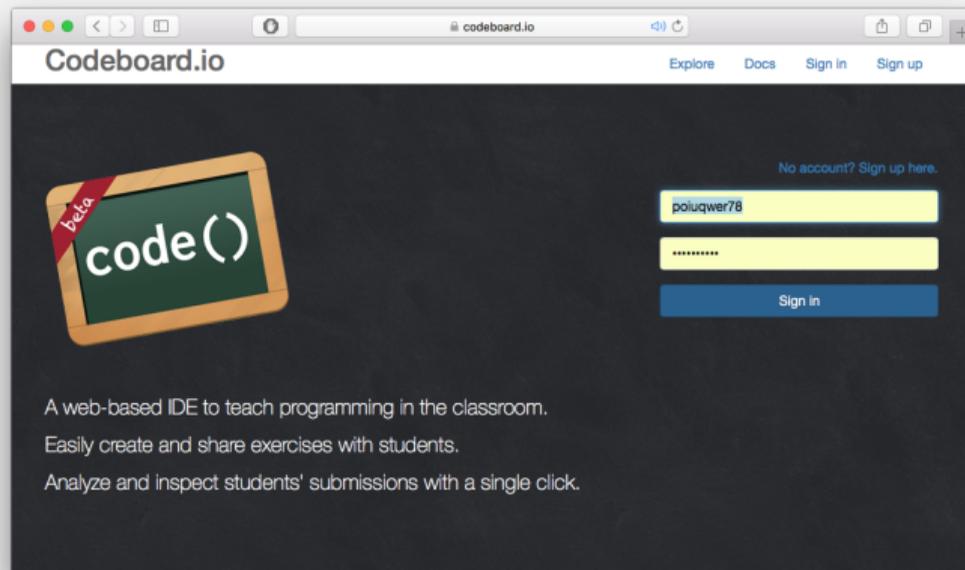
- Username\***: Input field with the placeholder text "whatever you want".
- Email\***: Input field with the placeholder text "eth or private email address".
- Password\***: Input field.
- Confirm password\***: Input field.

At the bottom of the form is a blue button labeled "Create account". Below the browser window, a status bar reads: "Open 'https://codeboard.io/signup' in a new tab".

- Wir verwenden die Online IDE **Codeboard.io**
- Erstellen Sie dort einen Account, um Ihren Fortschritt abzuspeichern und später Submissions anzuschauen
- Anmeldedaten können beliebig gewählt werden! *Verwenden Sie nicht das ETH Passwort.*

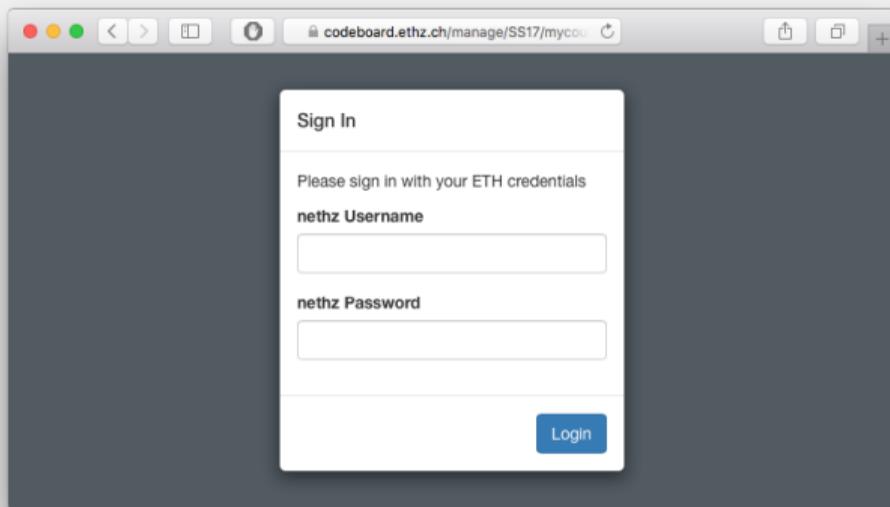
# Codeboard.io Login

Falls Sie schon einen Account haben, loggen Sie sich ein:



# Einschreibung in Übungsgruppen - I

- Besuchen Sie `http://codeboard.ethz.ch/ifbaug2`
- Loggen Sie sich mit Ihrem nethz Account ein.

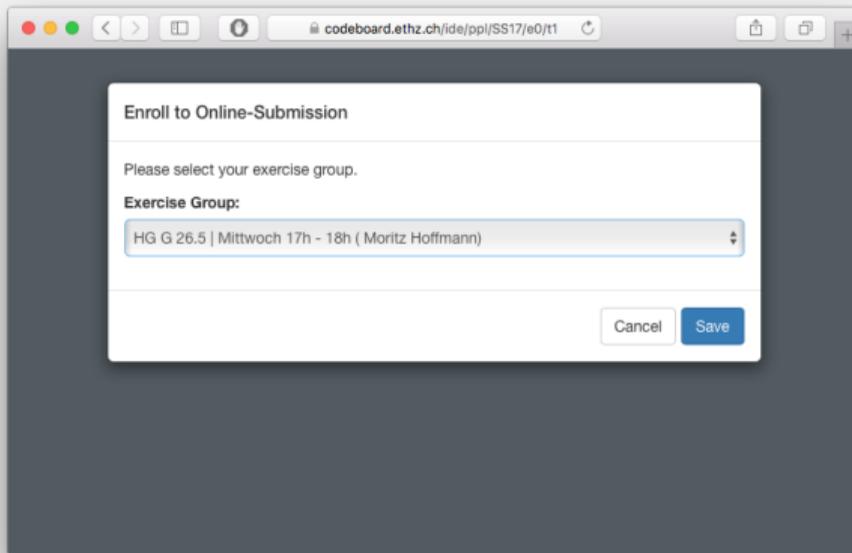


The image shows a browser window with the address bar containing `codeboard.ethz.ch/manage/SS17/mycode`. The main content area displays a white 'Sign In' form on a dark grey background. The form includes the following elements:

- Sign In** (Section Header)
- Please sign in with your ETH credentials
- nethz Username** (Label) followed by an empty text input field.
- nethz Password** (Label) followed by an empty password input field.
- Login** (Blue button)

# Einschreibung in Übungsgruppen - II

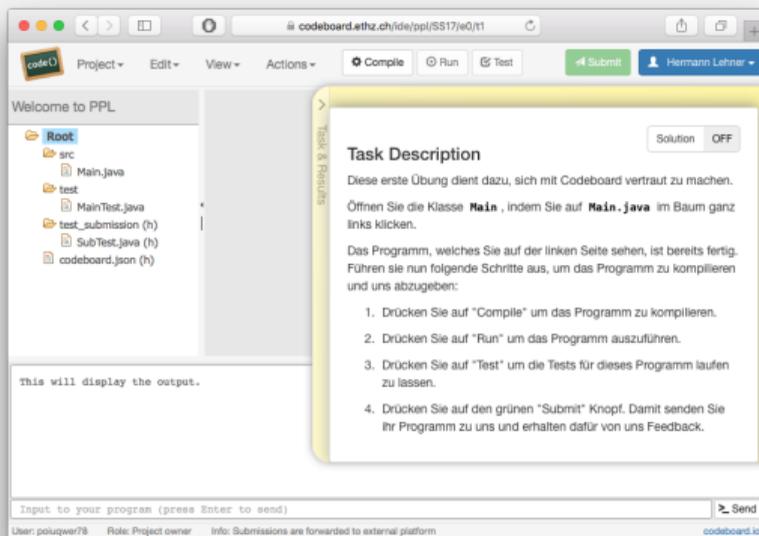
Schreiben Sie sich in diesem Dialog in eine Übungsgruppe ein.



The image shows a browser window with the URL `codeboard.ethz.ch/ide/pp/SS17/e0/t1`. A modal dialog titled "Enroll to Online-Submission" is displayed. The dialog contains the text "Please select your exercise group." followed by a label "Exercise Group:" and a dropdown menu. The dropdown menu is currently open and shows the selected option: "HG G 26.5 | Mittwoch 17h - 18h ( Moritz Hoffmann)". At the bottom right of the dialog, there are two buttons: "Cancel" and "Save".

# Die erste Übung

Sie sind nun eingeschrieben und die erste Übung ist geladen. Folgen Sie den Anweisungen in der gelben Box. Das Übungsblatt auf der Kurshomepage enthält weitere Anweisungen und Erklärungen.



The screenshot shows the Codeboard IDE interface. The browser address bar displays `codeboard.ethz.ch/ide/spl/SS17/w0/t1`. The top navigation bar includes buttons for "Compile", "Run", "Test", "Submit", and a user profile for "Herrmann Lehner". The left sidebar shows a file tree with a "Root" folder containing subfolders "src" and "test", and files "Main.java", "MainTest.java", "test\_submission (h)", "SubTest.java (h)", and "codeboard.json (h)". The main workspace area is currently empty, with a placeholder text "This will display the output." and an input field at the bottom labeled "Input to your program (press Enter to send)". A yellow box on the right side of the workspace contains the "Task Description" for the exercise.

**Task Description** Solution OFF

Diese erste Übung dient dazu, sich mit Codeboard vertraut zu machen. Öffnen Sie die Klasse **Main**, indem Sie auf **Main.java** im Baum ganz links klicken.

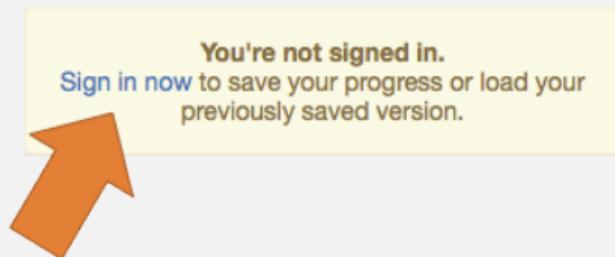
Das Programm, welches Sie auf der linken Seite sehen, ist bereits fertig. Führen sie nun folgende Schritte aus, um das Programm zu kompilieren und uns abzugeben:

1. Drücken Sie auf "Compile" um das Programm zu kompilieren.
2. Drücken Sie auf "Run" um das Programm auszuführen.
3. Drücken Sie auf "Test" um die Tests für dieses Programm laufen zu lassen.
4. Drücken Sie auf den grünen "Submit" Knopf. Damit senden Sie Ihr Programm zu uns und erhalten dafür von uns Feedback.

Footer: User: poluqwe78 Role: Project owner Info: Submissions are forwarded to external platform codeboard.io

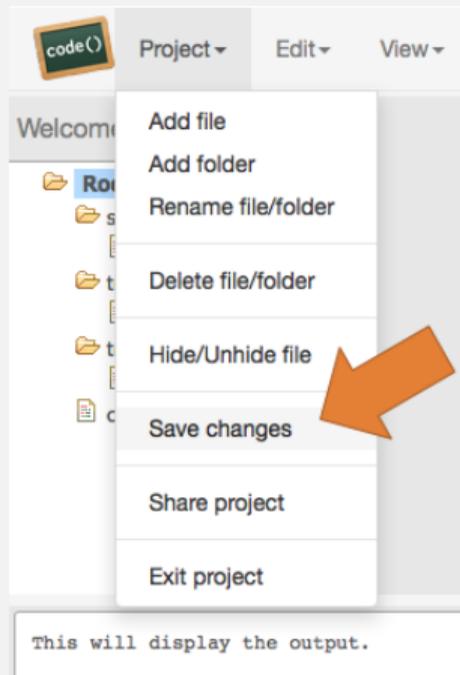
# Die erste Übung - Codeboard.io Login

Falls Sie diese Nachricht sehen, klicken Sie auf [Sign in now](#) und melden Sie sich dort mit ihrem **Codeboard.io** Account ein.



# Die erste Übung - Fortschritt speichern!

*Achtung!* Speichern Sie ihren Fortschritt regelmässig ab. So können Sie jederzeit an einem anderen Ort weiterarbeiten.



# Zu den Übungen

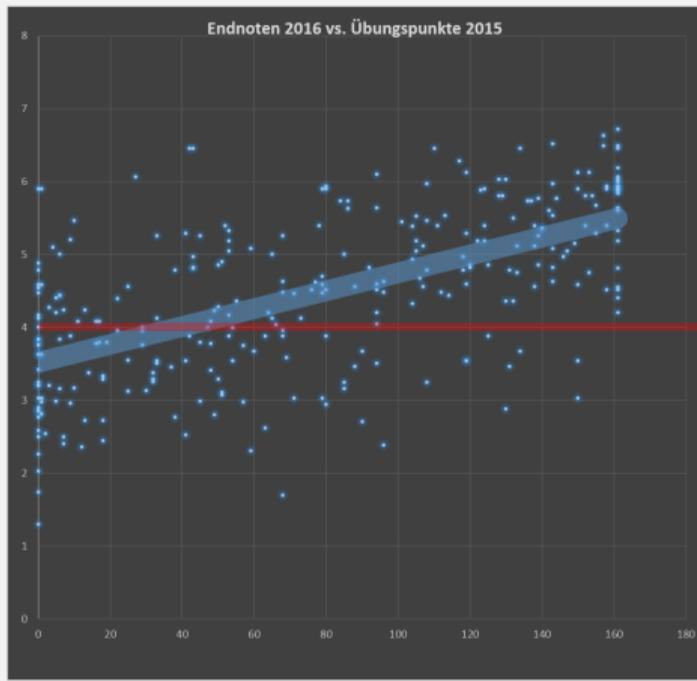
- Seit HS 2013 für Prüfungszulassung kein Testat mehr erforderlich.

# Zu den Übungen

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

# Zu den Übungen

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



# Tipps

- Üben Sie!

# Tipps

- Wenn Sie zu zweit an Übungen arbeiten, stellen Sie Gleichverteilung des aktiven Parts sicher.

# Tipps

- Lassen Sie sich nicht frustrieren. Schlafen Sie eine Nacht, wenn Sie nicht vorwärtskommen.

# Tipps

- Holen Sie sich Hilfe, wenn Sie nicht vorwärtskommen.

# Tipps

- Üben Sie!

# 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
- Auch das Tutorial ist basierend auf **Codeboard.io**

# 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
- Auch das Tutorial ist basierend auf **Codeboard.io**

→ Das ist gut investierte Zeit!

# Tutorial - Url

## Java Tutorial

Hier finden Sie das Tutorial

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

# Relevantes für die Prüfung

Prüfungsstoff für die Endprüfung (in der Prüfungssession 2017) schliesst ein

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

*Programmierkenntnisse sind essentiell zum Lösen der Prüfung.*

# Relevantes für die Prüfung

Prüfung (120 min) ist schriftlich. Hilfsmittel: vier A4-Seiten (bzw. 2 A4-Blätter doppelseitig) entweder handgeschrieben oder mit Fontgrösse minimal 11 Punkt.

*Programmierkenntnisse sind essentiell zum Lösen der Prüfung.*

# Unser Angebot

Ein freiwilliges Midterm (MT).

Warum?

- 1 Damit Sie am Ball bleiben !
- 2 Damit Sie auf die Prüfung vorbereitet sind.

Termin: vorr. 4.5.2017 zur Übungsstunde. Bitte Kollisionen melden.

# Unser Angebot

Seien

$B$ =Note Basisprüfung Informatik II,

$M$ =Note Midterm Informatik II,

$N$ =Endnote Basisprüfung Informatik II,

dann:<sup>2</sup>

$$N = \begin{cases} 0.7 \cdot B + 0.3 \cdot M & \text{falls } M > B \\ B & \text{sonst.} \end{cases}$$

---

<sup>2</sup>In anderen Worten: Das Midterm zählt genau dann zu 30% wenn es zu Ihrem Vorteil ist.

# Literatur

**Sprechen Sie Java?**, Hanspeter Mössenböck, dpunkt Verlag, 5. Auflage 2014.

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

**Algorithmen und Datenstrukturen**, T. Ottmann, P. Widmayer, Spektrum-Verlag, 5. Auflage, 2011

**Algorithmen - Eine Einführung**, T. Cormen, C. Leiserson, R. Rivest, C. Stein, Oldenbourg, 2010

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

**Introduction to Programming in Java: An Interdisciplinary Approach**, *Robert Sedgewick, Kevin Wayne* , Addison-Wesley, 2008

**Introduction to Algorithms**, *T. Cormen, C. Leiserson, R. Rivest, C. Stein* , 3rd ed., MIT Press, 2009

# In Ihrem und unserem Interesse

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.



## 3. Von Pascal nach Java

erstes Java Programm, Pascal → Java, der euklidische Algorithmus in Java, Java Klassen

# Erstes Pascal Programm

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

# Erstes Java Programm

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

Aufruf: java Hello

# Erstes Java Programm

```
public class Hello {
```



*class: Ein Programm*

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

```
    {
```

```
        System.out.println("Hello World.");
```

```
    }
```

```
}
```

Aufruf: java Hello

# Erstes Java Programm

```
public class Hello {
```

*class:* Ein Programm

*Methode:* benannte  
Folge von Anweisungen.

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

Aufruf: java Hello

# Erstes Java Programm

```
public class Hello {
```

*class:* Ein Programm

*Methode:* benannte Folge von Anweisungen.

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

```
{
```

```
    System.out.println("Hello World.");
```

```
}
```

```
}
```

*Aufruf* einer Methode (Prozedur) einer anderen Klasse.

Aufruf: java Hello

# Erstes Java Programm

```
public class Hello {
```

*class:* Ein Programm

*Methode:* benannte Folge von Anweisungen.

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

*Anweisung:* Kommando, welches auszuführen ist.

```
{
```

```
    System.out.println("Hello World.");
```

```
}
```

*Aufruf* einer Methode (Prozedur) einer anderen Klasse.

Aufruf: java Hello

# 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<sup>3</sup>)

Arrays → Arrays (Achtung: Referenzsemantik)

Var-Parameter → es gibt in Java kein Pass-by-Reference!

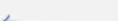
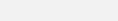
---

<sup>3</sup>wird noch erklärt

# 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);  
    }  
}
```

# Der euklidische Algorithmus in Java

```
public class Euclidean {  
  
    public static void main(String[] args){  
        int a = 24;  Variablen müssen vor Gebrauch deklariert werden.  
        int b = 20;  Keine separate Deklarationssequenz nötig.  
        while (b != 0) {  Anweisungsblöcke sind durch geschweifte  
            if (a>b)  Klammern gekennzeichnet.  
                a = a - b;  Bedingungen stehen bei if und while im-  
                b = b - a;  mer in runden Klammern.  
            }  
        System.out.println("ggT(24,20)= " + a);  
    }  
}
```

# Nachteil dieser Variante

```
while (b != 0) {  
    if (a>b)  
        a = a - b;  
    else  
        b = b - a;  
}
```



# Nachteil dieser Variante

```
while (b != 0) {  
    if (a > b)  
        a = a - b;  
    else  
        b = b - a;  
}
```



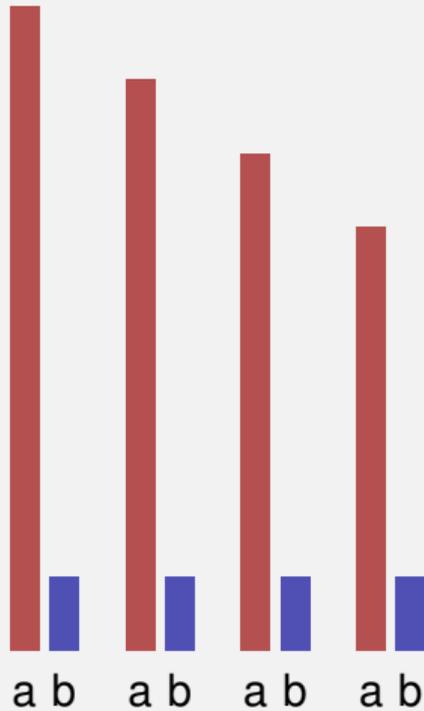
# Nachteil dieser Variante

```
while (b != 0) {  
    if (a>b)  
        a = a - b;  
    else  
        b = b - a;  
}
```



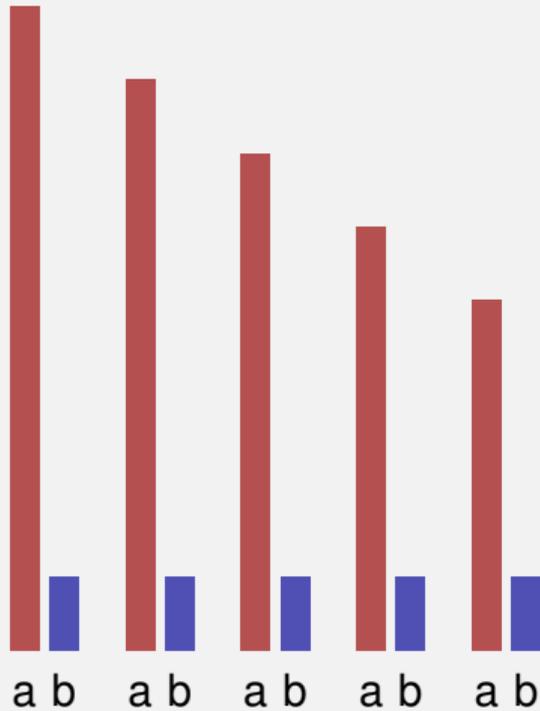
# Nachteil dieser Variante

```
while (b != 0) {  
    if (a>b)  
        a = a - b;  
    else  
        b = b - a;  
}
```



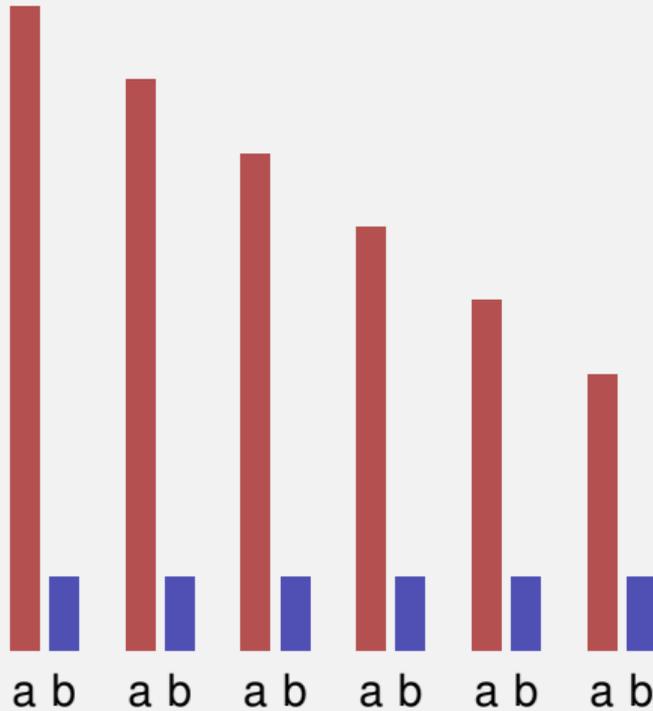
# Nachteil dieser Variante

```
while (b != 0) {  
    if (a>b)  
        a = a - b;  
    else  
        b = b - a;  
}
```



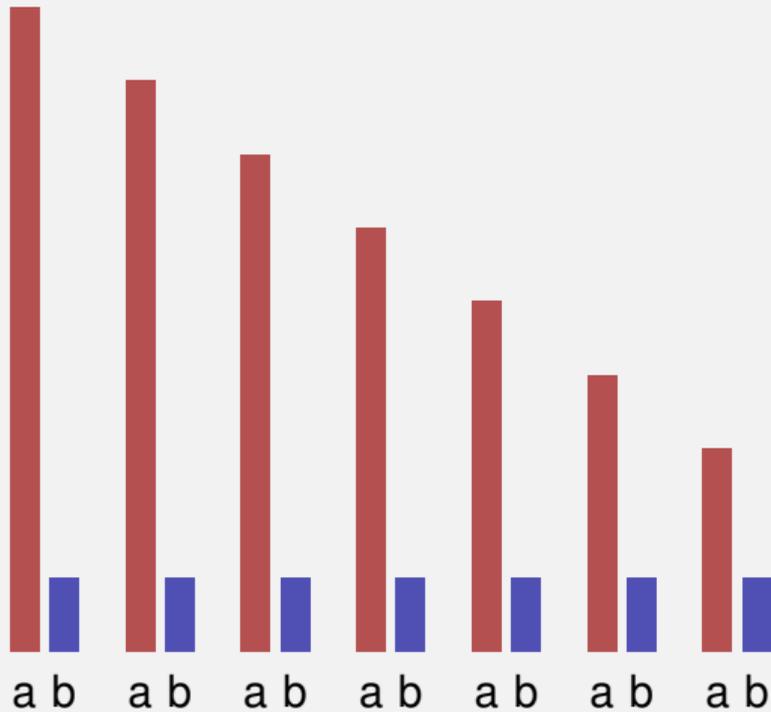
# Nachteil dieser Variante

```
while (b != 0) {  
    if (a>b)  
        a = a - b;  
    else  
        b = b - a;  
}
```



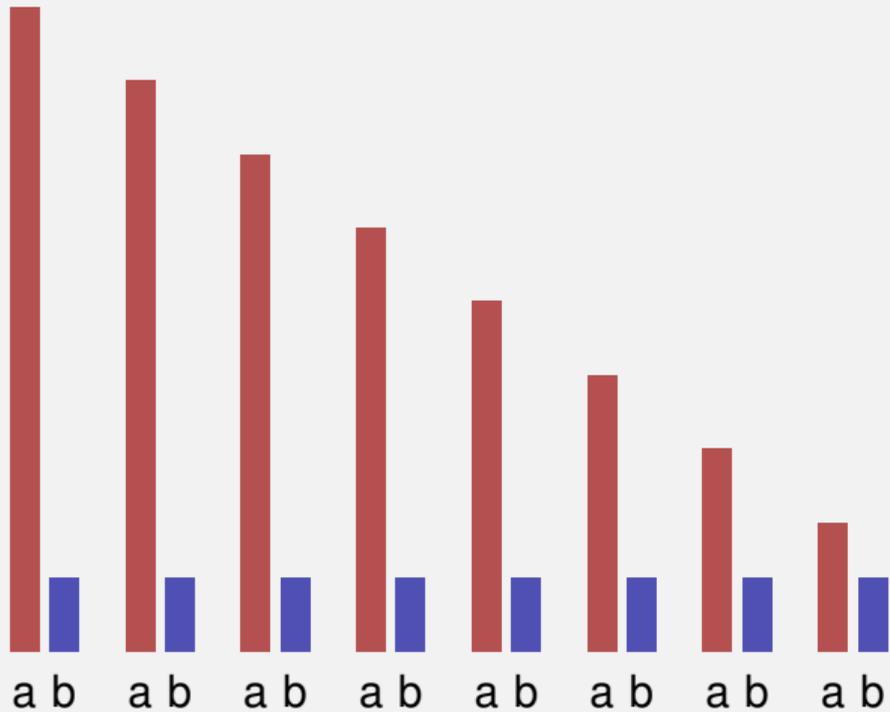
# Nachteil dieser Variante

```
while (b != 0) {  
    if (a>b)  
        a = a - b;  
    else  
        b = b - a;  
}
```



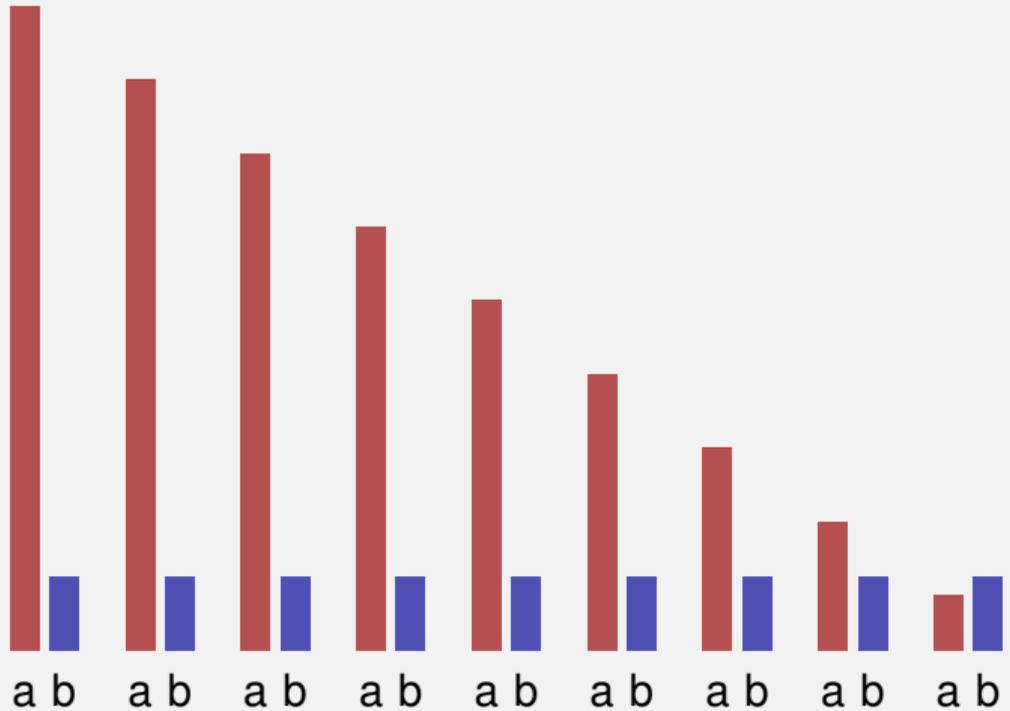
# Nachteil dieser Variante

```
while (b != 0) {  
    if (a>b)  
        a = a - b;  
    else  
        b = b - a;  
}
```



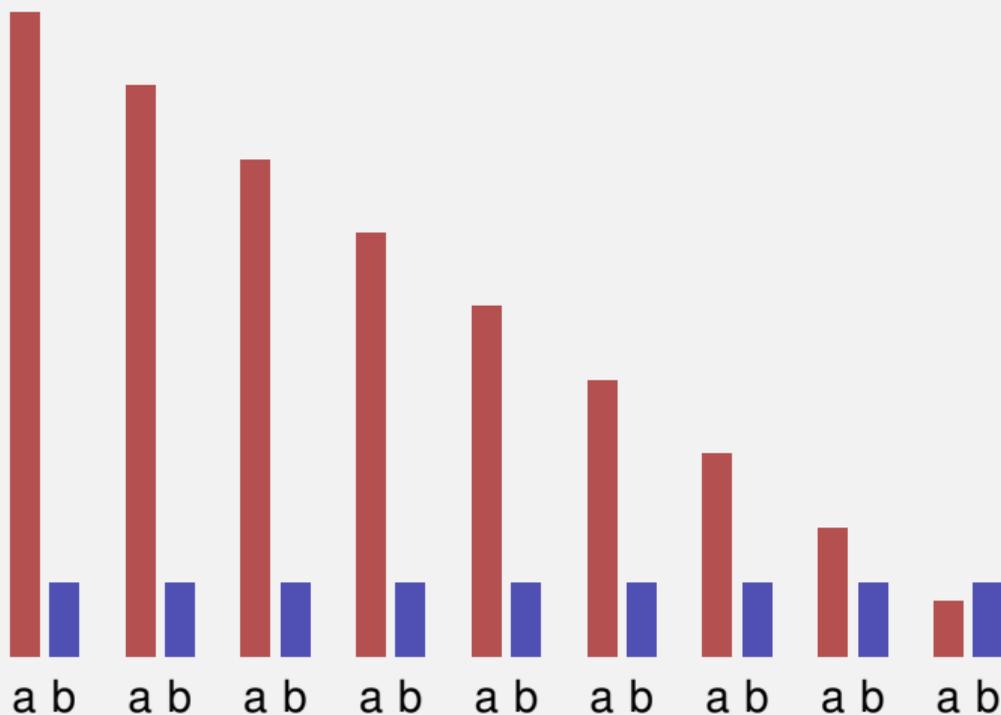
# Nachteil dieser Variante

```
while (b != 0) {  
    if (a>b)  
        a = a - b;  
    else  
        b = b - a;  
}
```



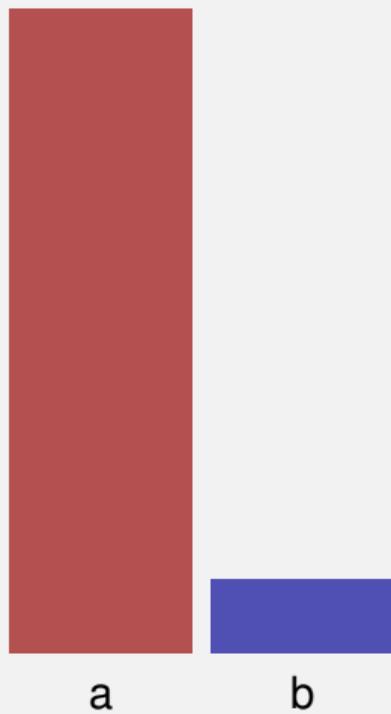
# Nachteil dieser Variante

```
while (b != 0) {  
    if (a>b)  
        a = a - b;  
    else  
        b = b - a;  
}
```

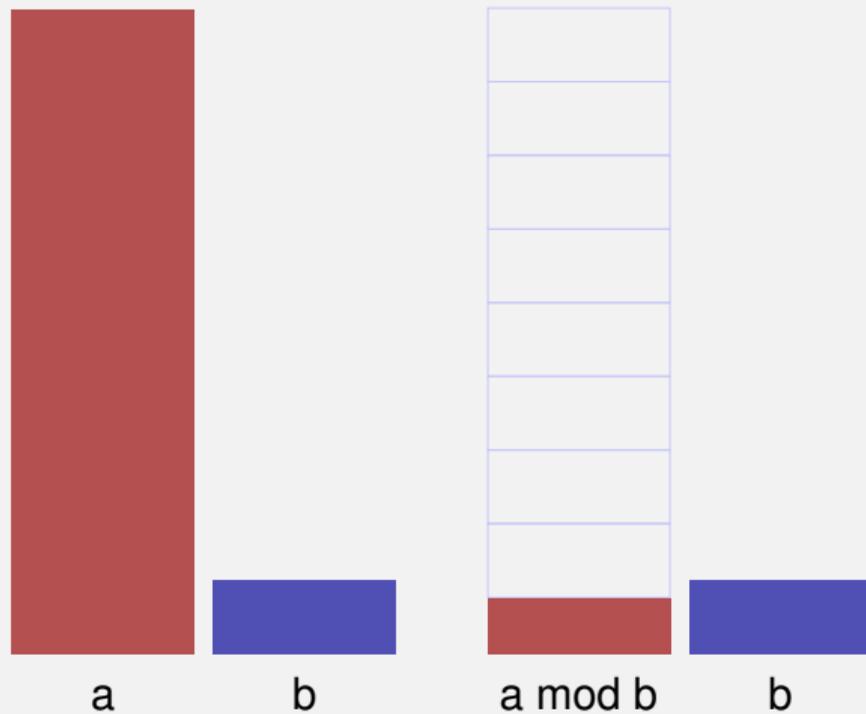


**Schneller so!**

# Schneller so!



# Schneller so!



# Moderne (und effizientere) Variante

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

# Vergleich mit Pascal

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

```
program Euklid;  
var a, b, h: integer;  
begin  
    a := 24;  
    b := 20;  
    while b <> 0 do begin  
        h := a mod b;  
        a := b;  
        b := h;  
    end;  
    writeln('ggt(24,20)=',a:5);  
  
end.
```

# Mit einer Funktion (Methode)

```
public class Euclidean {
    // PRE: a, b >= 0
    // POST: gibt GGT(a,b) zur\"{u}ck
    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));
    }
}
```

# Ein- und Ausgabe

Ausgabe via `System.out` mit

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

Eingabe via `System.in` mit einem Scanner:

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

benötigt einen Import:

```
import java.util.Scanner;
```

# Mit Eingabe

```
import java.util.Scanner;

public class Euclidean {

    static int ggt(int a, int b){ ... } // wie oben

    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();
```

Elementzugriff<sup>4</sup>

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

Verkettung

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

---

<sup>4</sup>Nur lesend. Strings sind unveränderlich