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)

Ziel der Vorlesung

Problemlösen mit dem Computer

Hilfsmittel:

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

Ziel der Vorlesung

Problemlösen mit dem Computer

Methodik

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

Ziel der Vorlesung

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

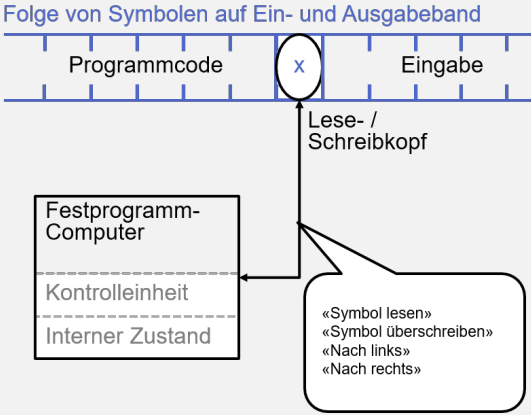
Einführung / Wiederholung Computermodell und Algorithmus

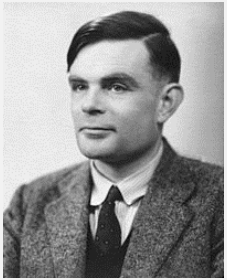
Ziel der *heutigen* Vorlesung

Prozedurales Programmieren mit Java, Pascal $\rightarrow$ Java.

Computer – Konzept

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



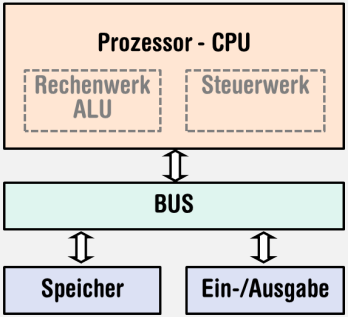


Alan Turing

Computer – Umsetzung

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

Von Neumann Architektur



Konrad Zuse

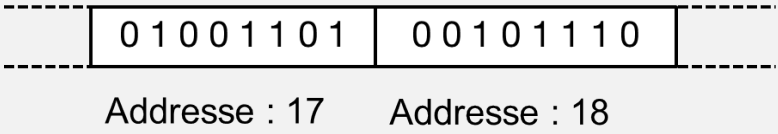
John von Neumann

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 $b \neq 0$

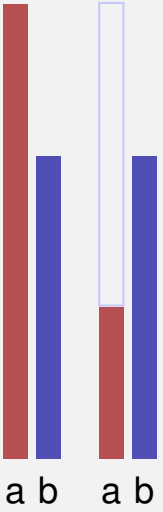
Wenn $a > b$ dann

$a \leftarrow a - b$

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 $b \neq 0$

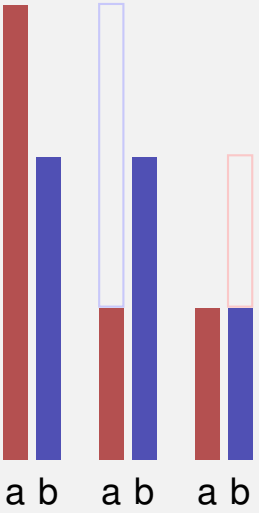
Wenn $a > b$ dann

$a \leftarrow a - b$

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 $b \neq 0$

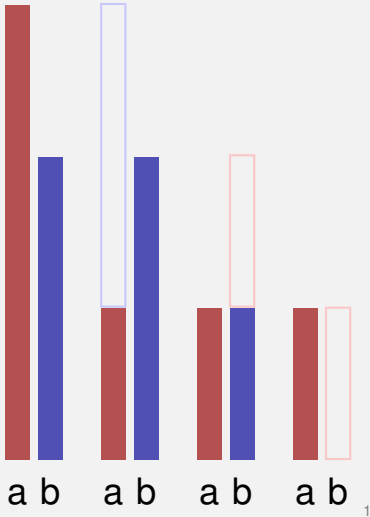
Wenn $a > b$ dann

$a \leftarrow a - b$

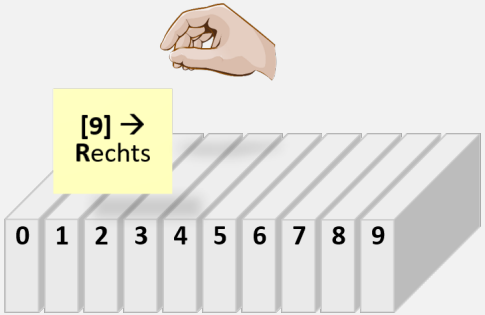
Sonst:

$b \leftarrow b - a$

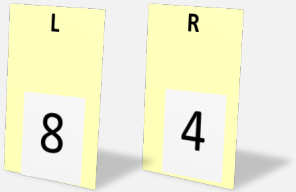
Ergebnis: a.



Live Demo: Turing Maschine



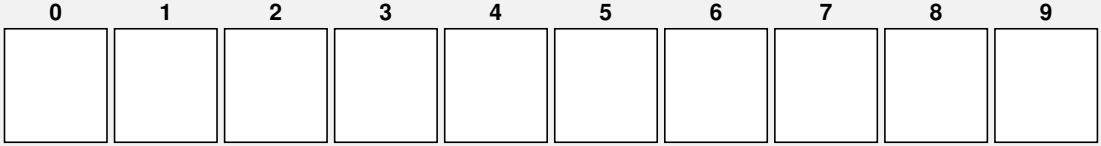
Speicher

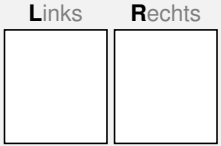


Register

Euklid in der Box

Speicher

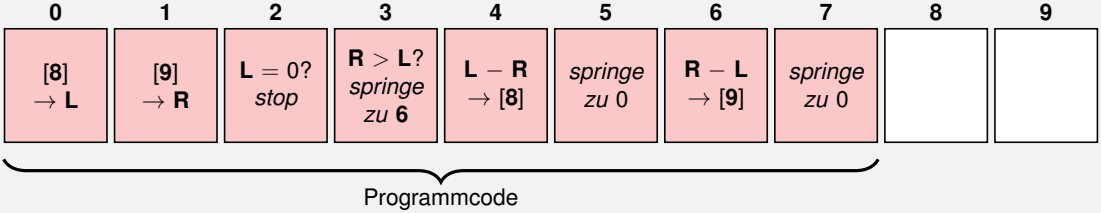


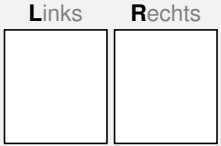


Register

Euklid in der Box

Speicher

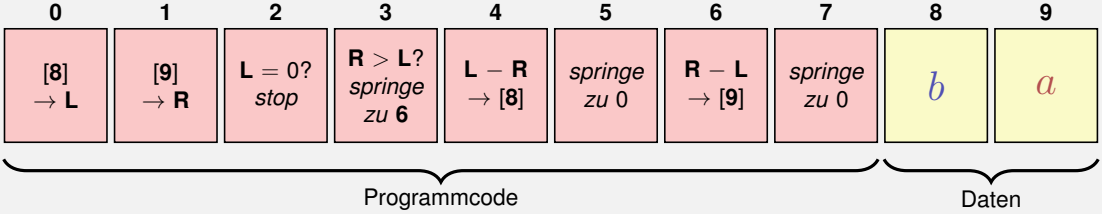


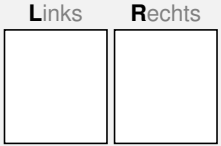


Register

Euklid in der Box

Speicher

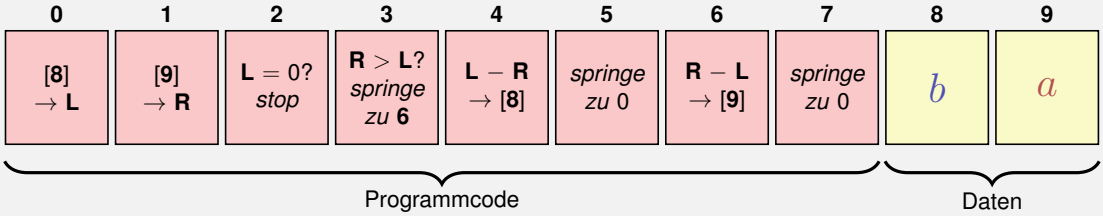


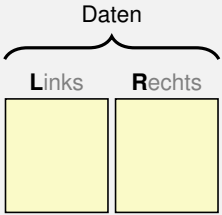


Register

Euklid in der Box

Speicher

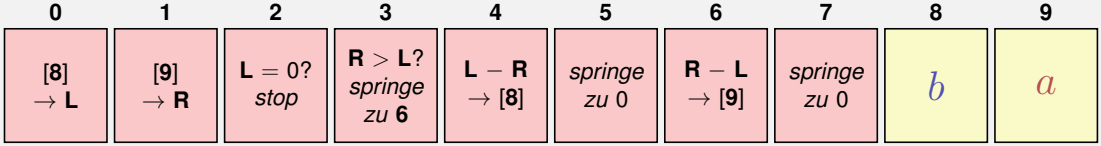


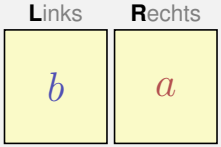


Register

Euklid in der Box

Speicher





Register

Solange $b \neq 0$

Wenn $a > b$ dann

$a \leftarrow a - b$

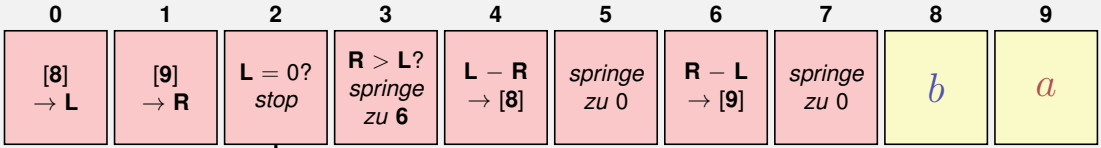
Sonst:

$b \leftarrow b - a$

Ergebnis: a.

Euklid in der Box

Speicher



Solange $b \neq 0$

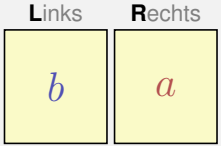
Wenn $a > b$ dann

$a \leftarrow a - b$

Sonst:

$b \leftarrow b - a$

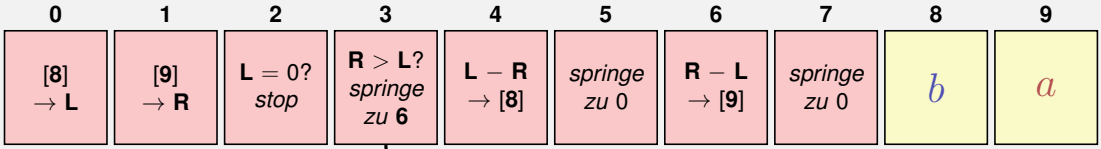
Ergebnis: a.



Register

Euklid in der Box

Speicher



Solange $b \neq 0$

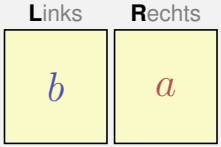
Wenn $a > b$ dann

$a \leftarrow a - b$

Sonst:

$b \leftarrow b - a$

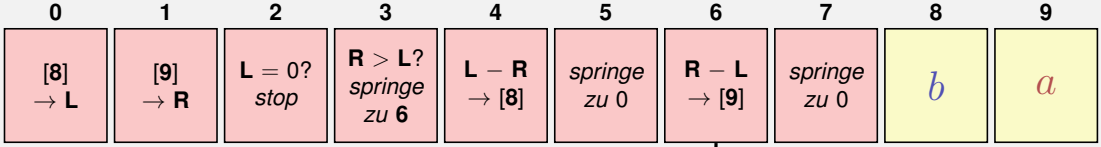
Ergebnis: a.



Register

Euklid in der Box

Speicher



Solange $b \neq 0$

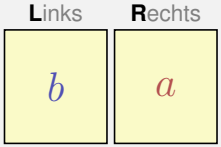
Wenn $a > b$ dann

$a \leftarrow a - b$

Sonst:

$b \leftarrow b - a$

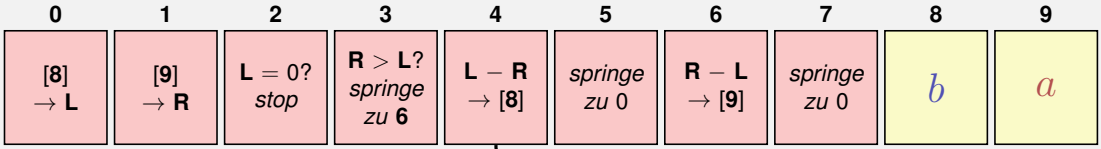
Ergebnis: a.



Register

Euklid in der Box

Speicher



Solange $b \neq 0$

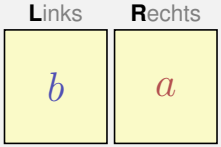
Wenn $a > b$ dann

$a \leftarrow a - b$

Sonst:

$b \leftarrow b - a$

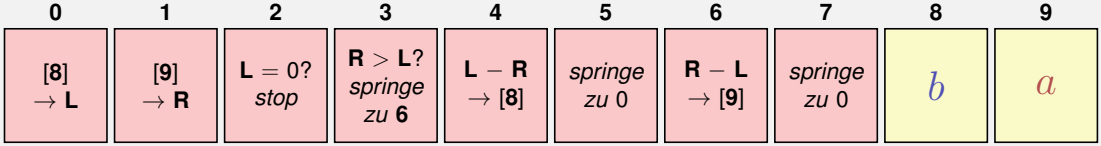
Ergebnis: a.

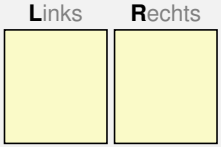


Register

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.

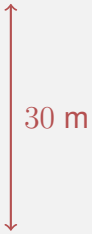
Rechengeschwindigkeit

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

[1] Uniprozessor Computer bei 1GHz

Rechengeschwindigkeit

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



arbeitet ein heutiger Desktop-PC mehr als 100

[1] Uniprozessor Computer bei 1GHz

Rechengeschwindigkeit

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

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

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

[1] 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

Java

- 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

Java

- 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 $\rightarrow$ physikalische Maschine

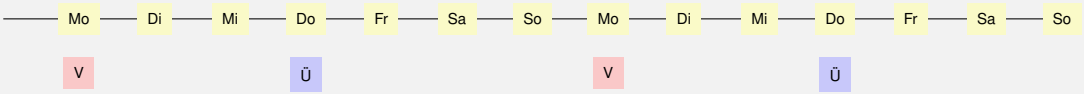
Java

- 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 $\rightarrow$ physikalische Maschine
- Folgerung, und erklärtes Ziel der Entwickler von Java: Portabilität

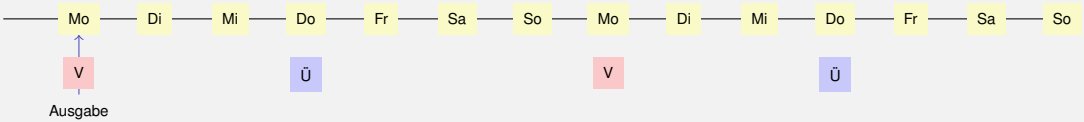
  write once – run anywhere

2. Organisation

Ablauf

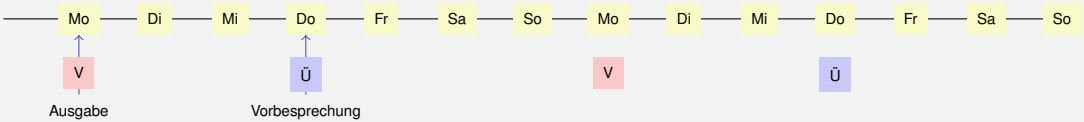


Ablauf



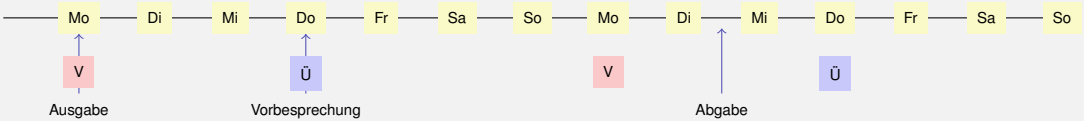
- Übungsblattausgabe zur Vorlesung (online).

Ablauf



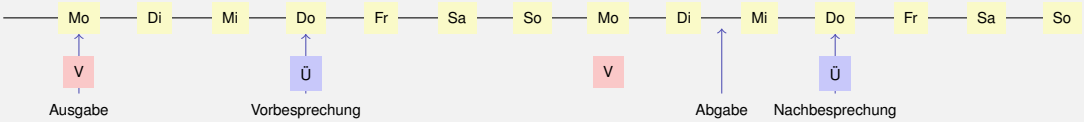
- Vorbesprechung am folgenden Donnerstag.

Ablauf



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

Ablauf

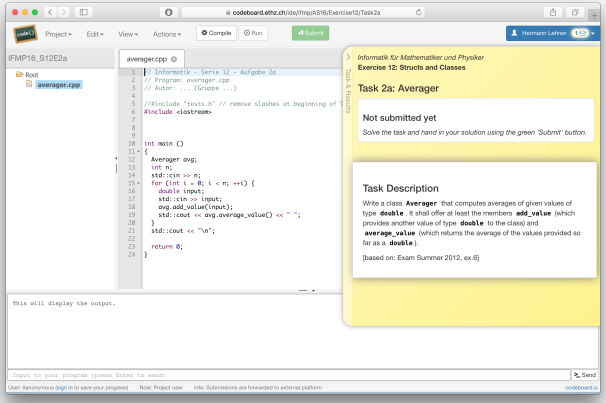


- Nachbesprechung der Übung am Donnerstag. Feeback zu den Abgaben innerhalb einer Woche nach Nachbesprechung.

Codeboard

Codeboard ist eine Online-IDE: Programmieren im Browser!

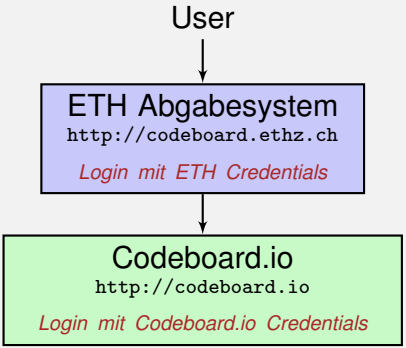
- Falls vorhanden, bringen Sie ihren Laptop/Tablet/... mit in den Unterricht.
- Sie können direkt in der Vorlesung Beispiele ausprobieren, ohne dass Sie irgendwelche Tools installieren müssen.



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

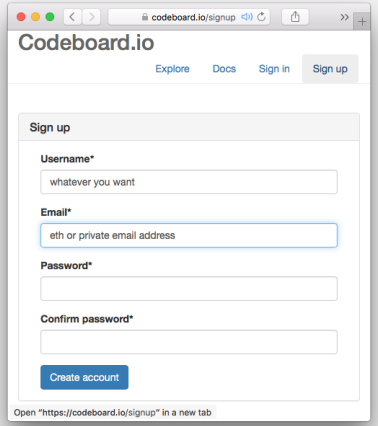
Gehen Sie auf `http://codeboard.io` und erstellen Sie dort ein Konto, bleiben Sie am besten eingeloggt.

Einschreibung in Übungsgruppen

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

Codeboard.io Registrierung

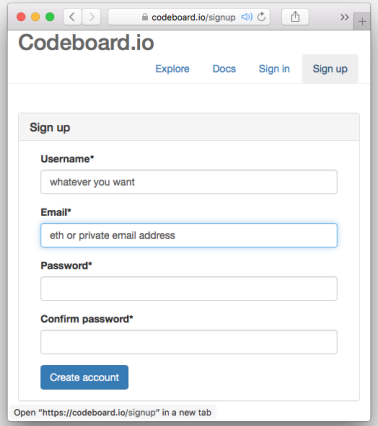
Falls Sie noch keinen **Codeboard.io** Account haben ...



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

Codeboard.io Registrierung

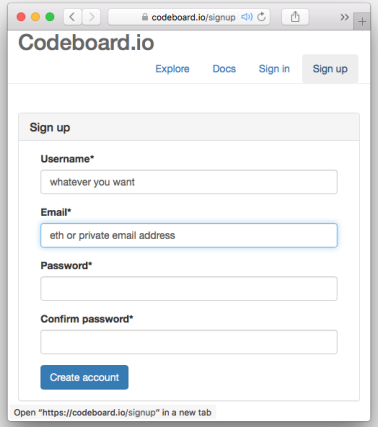
Falls Sie noch keinen **Codeboard.io** Account haben ...



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

Codeboard.io Registrierung

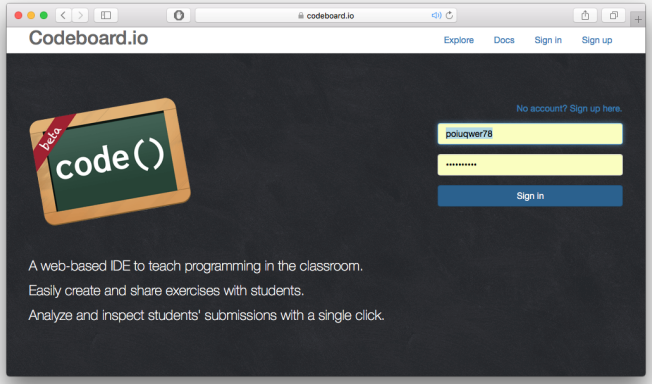
Falls Sie noch keinen **Codeboard.io** Account haben ...



- 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.*

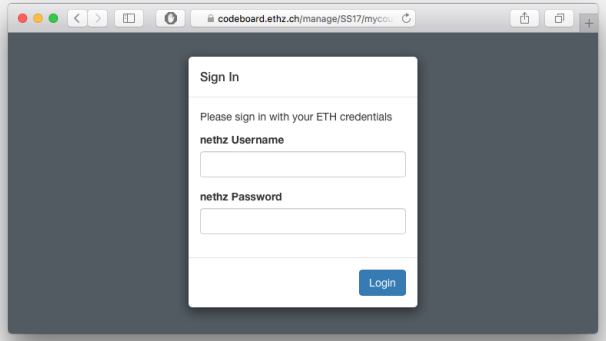
Codeboard.io Login

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



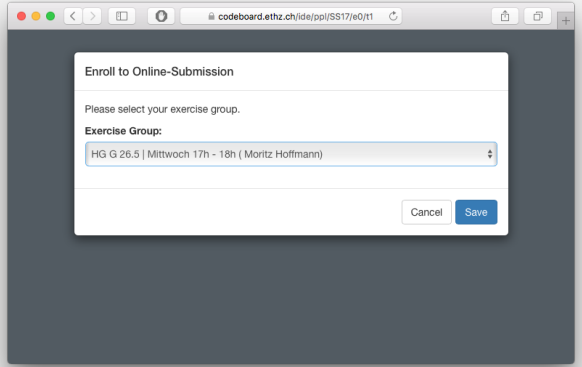
Einschreibung in Übungsgruppen - I

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



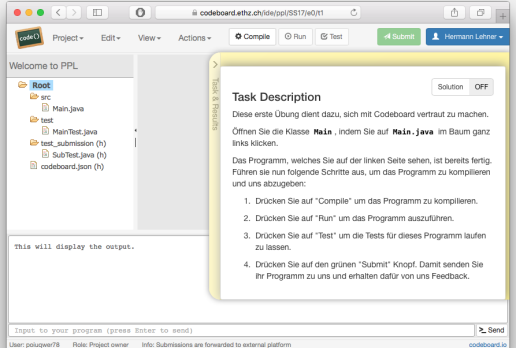
Einschreibung in Übungsgruppen - II

Schreiben Sie sich in diesem Dialog in eine Übungsgruppe ein.



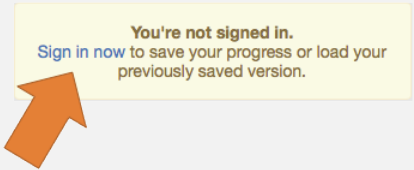
Die erste Übung

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.



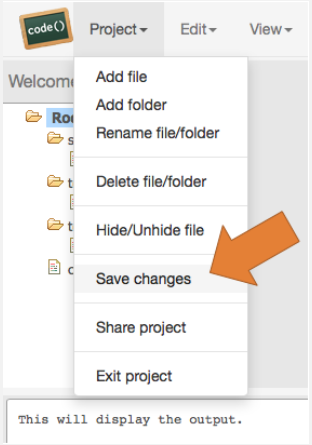
Die erste Übung - Codeboard.io Login

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

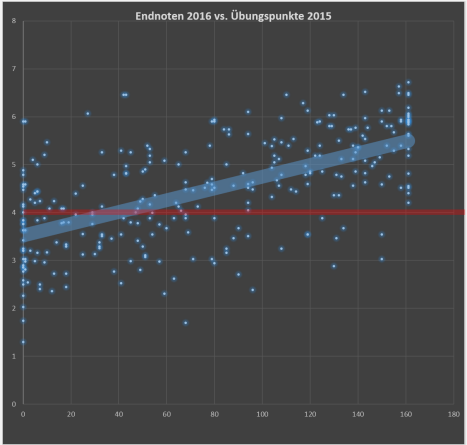
- Seit HS 2013 für Prüfungszulassung kein Testat mehr erforderlich.

Zu den Übungen

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

Zu den Übungen

- Bearbeitung der wöchentlichen Uebungsserien 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:[2]

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

[2] 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.

Literatur - Englisch

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 $\rightarrow$ Java, der euklidische Algorithmus in Java, Java Klassen

Erstes Pascal Programm

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

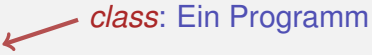
Erstes Java Programm

```
public class Hello {

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

Aufruf: `java Hello`

Erstes Java Programm




```
public class Hello {

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

Aufruf: `java Hello`

Erstes Java Programm

class: Ein Programm

Methode: benannte Folge von Anweisungen.

```
public class Hello {

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

Aufruf: `java Hello`

Erstes Java Programm

```
public class Hello {

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

class: Ein Programm

Methode: benannte Folge von Anweisungen.

Aufruf einer Methode (Prozedur) einer anderen Klasse.

Aufruf: `java Hello`

Erstes Java Programm

```
public class Hello {

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

class: Ein Programm

Methode: benannte Folge von Anweisungen.

Anweisung: Kommando, welches auszuführen ist.

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 $\rightarrow$ Java

Deklarationen, Ausdrücke und Anweisungen:separate Tabelle		
Programm	$\rightarrow$	Klasse mit `public static void main`
Prozedur	$\rightarrow$	`static` Methode in dieser Klasse
Globale Variablen	$\rightarrow$	`static` Variablen in dieser Klasse
Records	$\rightarrow$	Klassen (Achtung: Referenzsemantik[3])
Arrays	$\rightarrow$	Arrays (Achtung: Referenzsemantik)
Var-Parameter	$\rightarrow$	es gibt in Java kein Pass-by-Reference!

[3] 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;
    int b = 20;
    while (b != 0) {
      if (a>b)
        a = a - b;
      else
        b = b - a;
    }
    System.out.println("ggt(24,20)= " + a);
  }
}
```

Variablen müssen vor Gebrauch deklariert werden. Keine separate Deklarationssequenz nötig.

Anweisungsblöcke sind durch geschweifte Klammern gekennzeichnet.

Bedingungen stehen bei if und while immer in runden Klammern.

Nachteil dieser Variante

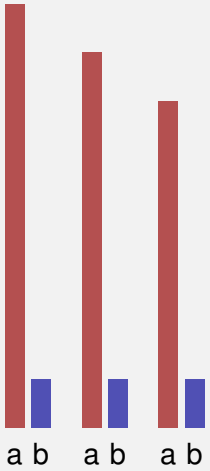
```
while (b != 0) {
   if (a>b)
     a = a – b;
   else
     b = b – a;
 }
```

Nachteil dieser Variante

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

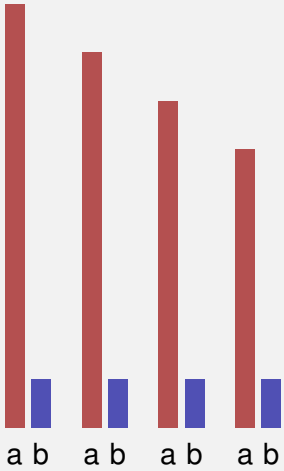
Nachteil dieser Variante

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



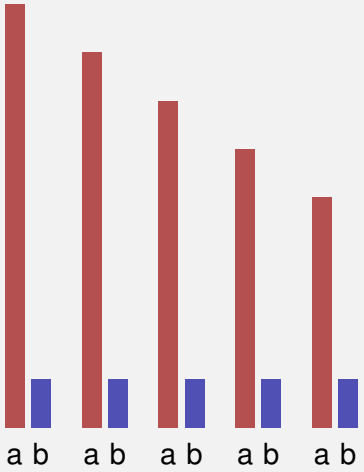
Nachteil dieser Variante

```
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;
 }
```

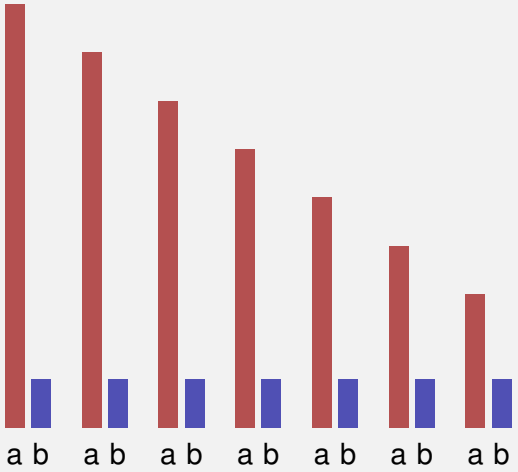


Nachteil dieser Variante

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

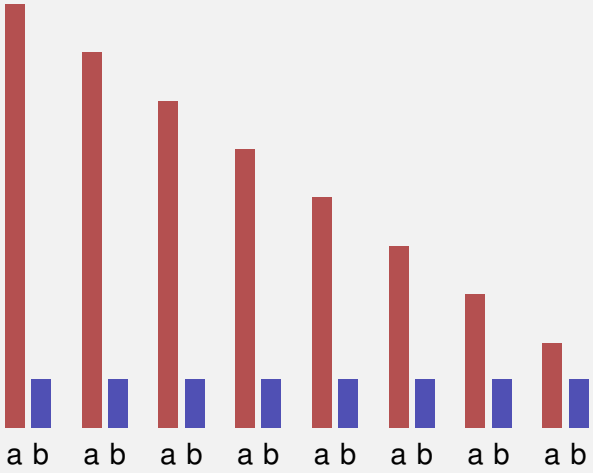
Nachteil dieser Variante

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



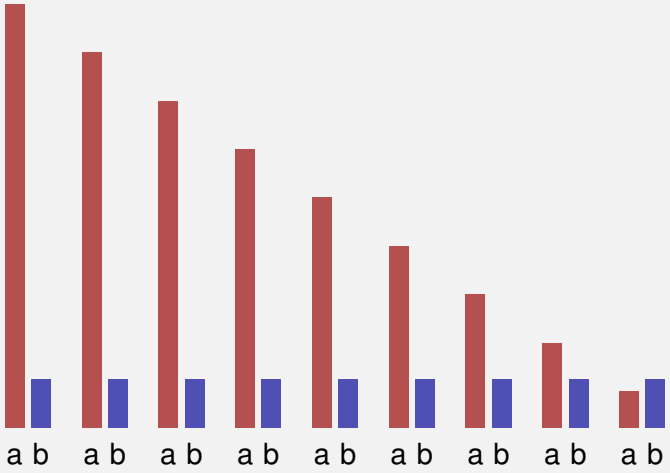
Nachteil dieser Variante

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



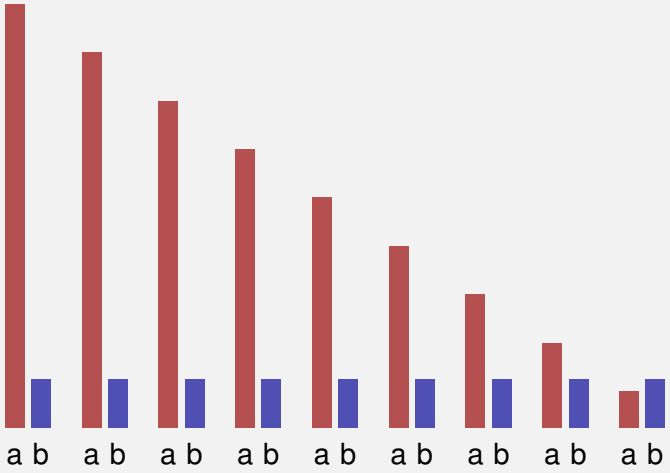
Nachteil dieser Variante

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



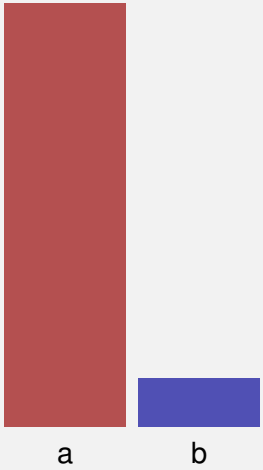
Nachteil dieser Variante

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

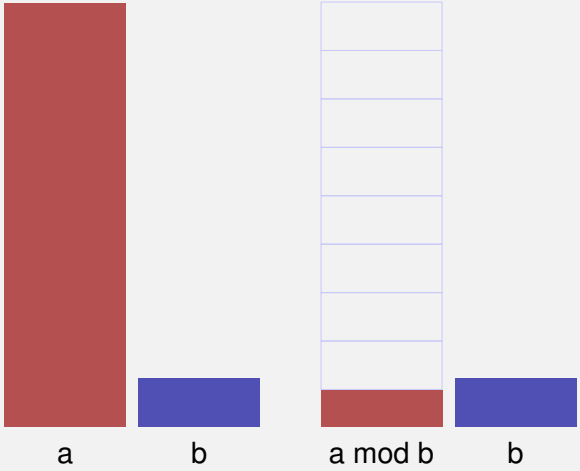


Schneller so!

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

```java
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[4]

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

Verkettung

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

[4] Nur lesend. Strings sind unveränderlich