ETH zürich

Hermann Lehner, Felix Friedrich

Informatik I

Vorlesung am D-BAUG der ETH Zürich

Herbst 2019

1. Einführung

Willkommen zur Vorlesung !

https://www.mentimeter.com/s/54775dbcef2827005cfcaa8e80bff221

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

Lino Guzzella, Präsident der ETH Zürich 2015-2018, NZZ Online, 1.9.2017

(Lino Guzzella ist übrigens nicht Informatiker, sondern Maschineningenieur und Prof. für Thermotronik ☺)

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.

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.

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
- Das **erste Programm** schreiben
- Allgemeine Informationen zur Vorlesung

1.1 Informatik und Algorithmus

Informatik, der Euklidische Algorithmus

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 $b \neq 0$

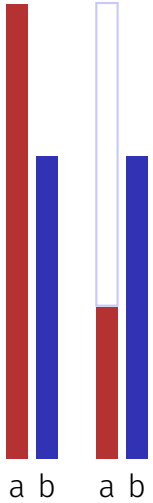
Wenn $a > b$ dann

$a \leftarrow a - b$

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 $b \neq 0$

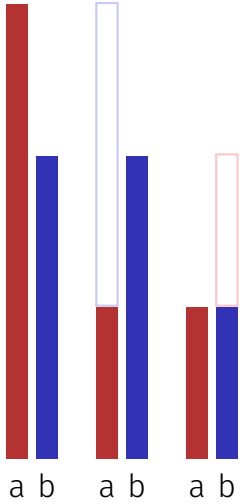
Wenn $a > b$ dann

$a \leftarrow a - b$

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 $b \neq 0$

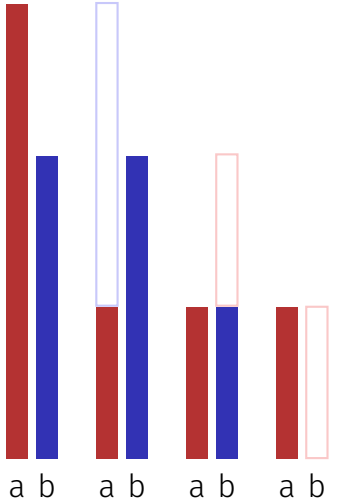
Wenn $a > b$ dann

$a \leftarrow a - b$

Sonst:

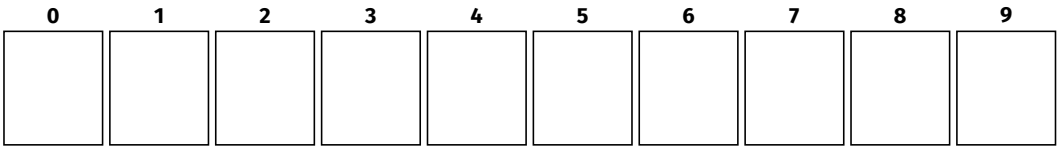
$b \leftarrow b - a$

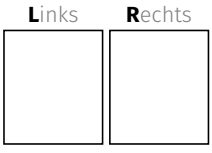
Ergebnis: a.



Euklid in der Box

Speicher

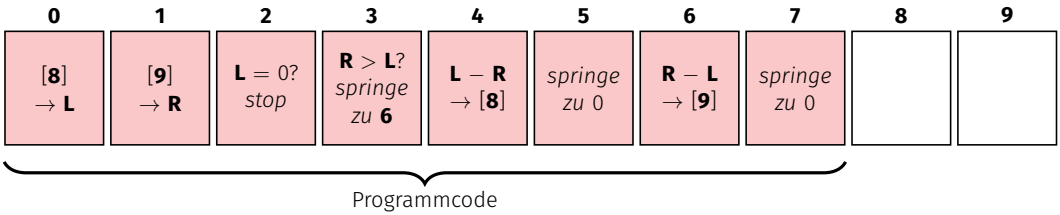


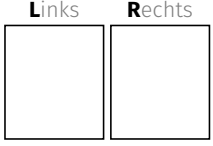


Register

Euklid in der Box

Speicher

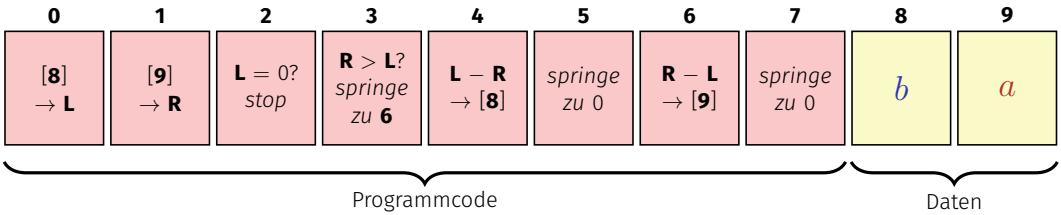


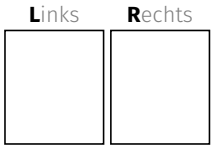


Register

Euklid in der Box

Speicher

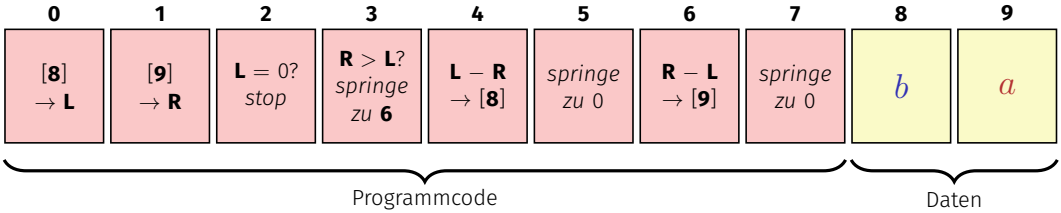


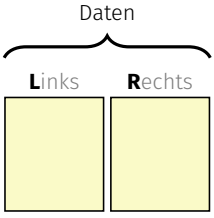


Register

Euklid in der Box

Speicher





Register

Euklid in der Box

Speicher

0	1	2	3	4	5	6	7	8	9
[**8**] → **L**	[**9**] → **R**	**L** = 0? *stop*	**R** > **L**? *springe zu* **6**	**L** − **R** → [**8**]	*springe zu* 0	**R** − **L** → [**9**]	*springe zu* 0	b	a

Links **R**echts

b	a

Register

Solange $b \neq 0$

Wenn $a > b$ dann

$a \leftarrow a - b$

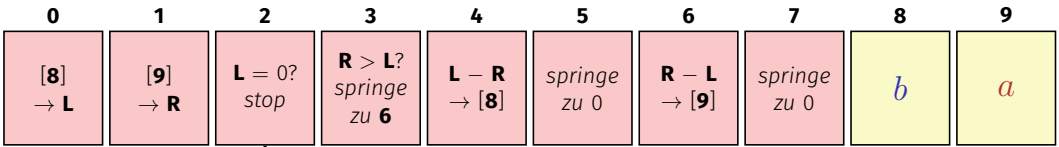
Sonst:

$b \leftarrow b - a$

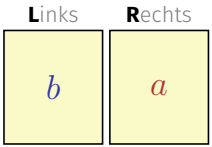
Ergebnis: a.

Euklid in der Box

Speicher



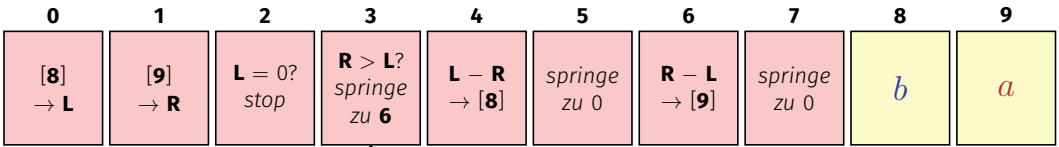
Solange $b \neq 0$
Wenn $a > b$ dann
$a \leftarrow a - b$
Sonst:
$b \leftarrow b - a$
Ergebnis: a.



Register

Euklid in der Box

Speicher



Solange $b \neq 0$

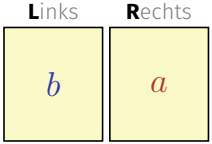
Wenn $a > b$ dann

$a \leftarrow a - b$

Sonst:

$b \leftarrow b - a$

Ergebnis: a.

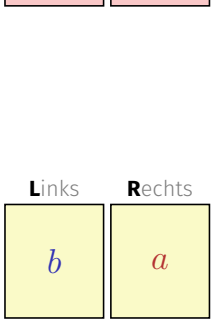


Register

Euklid in der Box

Speicher

0	1	2	3	4	5	6	7	8	9
[**8**] → **L**	[**9**] → **R**	**L** = 0? *stop*	**R** > **L**? *springe zu* **6**	**L** − **R** → [**8**]	*springe zu* 0	**R** − **L** → [**9**]	*springe zu* 0	b	a



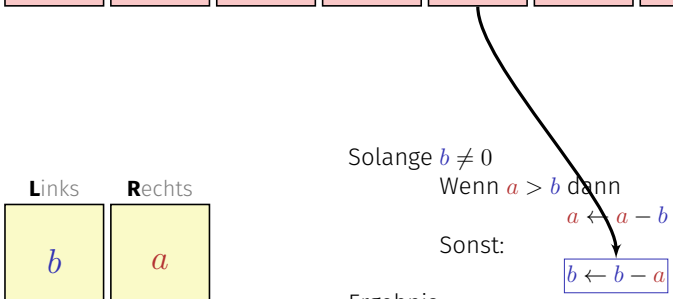
Register

Solange $b \neq 0$
Wenn $a > b$ dann
$a \leftarrow a - b$
Sonst:
$b \leftarrow b - a$
Ergebnis: a.

Euklid in der Box

Speicher

0	1	2	3	4	5	6	7	8	9
[**8**] → **L**	[**9**] → **R**	**L** = 0? *stop*	**R** > **L**? *springe zu* **6**	**L** − **R** → [**8**]	*springe zu* 0	**R** − **L** → [**9**]	*springe zu* 0	b	a



Euklid in der Box

Speicher

0	1	2	3	4	5	6	7	8	9
[**8**] → **L**	[**9**] → **R**	**L** = 0? *stop*	**R** > **L**? *springe zu* **6**	**L** − **R** → [**8**]	*springe zu* 0	**R** − **L** → [**9**]	*springe zu* 0	b	a

Links **R**echts

b	a

Register

Solange $b \neq 0$

Wenn $a > b$ dann

$a \leftarrow a - b$

Sonst:

$b \leftarrow b - a$

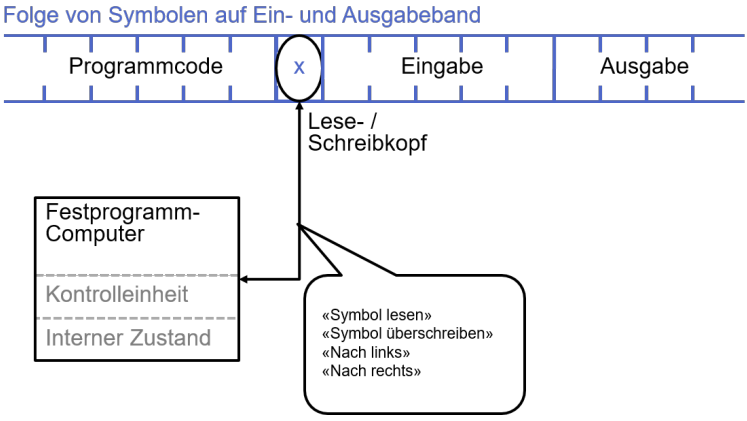
Ergebnis: a.

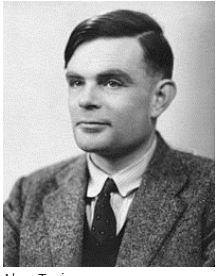
1.2 Computermodell

Turing Maschine, Von Neumann Architektur

Computer – Konzept

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

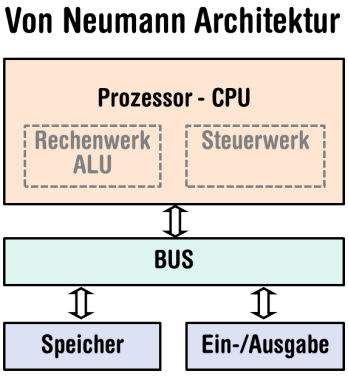




Alan Turing

Computer – Umsetzung

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



Konrad Zuse

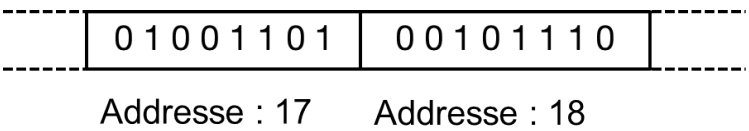
John von Neumann

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...

[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

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

Programmiersprachen

- Sprache, die der Computer "versteht", 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

  $\rightarrow$ Abstraktion!

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. 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 Maschinensprache

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 = a^2
b = b * b;     // b = a^4
```

Syntax und Semantik

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

Syntax und Semantik

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

Klasse: Ein Programm

Methode: benannte Folge von Anweisungen.

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 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);}}
```

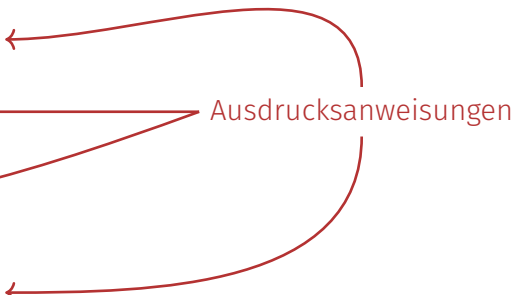
... uns aber nicht!

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);
  }
}
```



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 = a^2
    b = b * b; // b = a^4
    // 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;
    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);
  }
}
```

Deklarationsanweisungen

Typ-namen

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".

`int a;` definiert Variable mit

- Name: `a`
- Typ: `int`
- Wert: (vorerst) undefiniert
- Addresse: 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.

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*
- 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

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

Zusammengesetzer 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 );
```

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

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

Linker Operand (Variable)

Rechter Operand (Ausdruck)

Zuweisungsoperator

Multiplikationsoperator