Informatik I

Course at BAUG department of ETH Zürich

Hermann Lehner, Felix Friedrich
ETH Zürich

HS 2017

1. Introduction

Welcome to the Lecture Series!

Material

Course homepage

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

The Team

Lecturers	Hermann Lehner Felix Friedrich	
Chief assistant	Andrea Lattuada	
Assistants	Malte Schwerhoff	
	Rafael Wampfler	Gökcen Cimen
	Kai Sandbrink	Patrick Gruntz
	Irfan Bunjaku	Simon Guldimann
	Clemens Bachmann	Frederic Vogel
	Sander Staal	Philipp Schimmelfennig
	Nihat Isik	

Programming and Problem Solving

In this course you learn how to program using Java

- Software development is a handicraft
- Analogy: learn to play a musical instrument
- **The problem:** nobody has become a pianist from listening to music.

Hence this course offers several possibilities, to train. Make use of it!

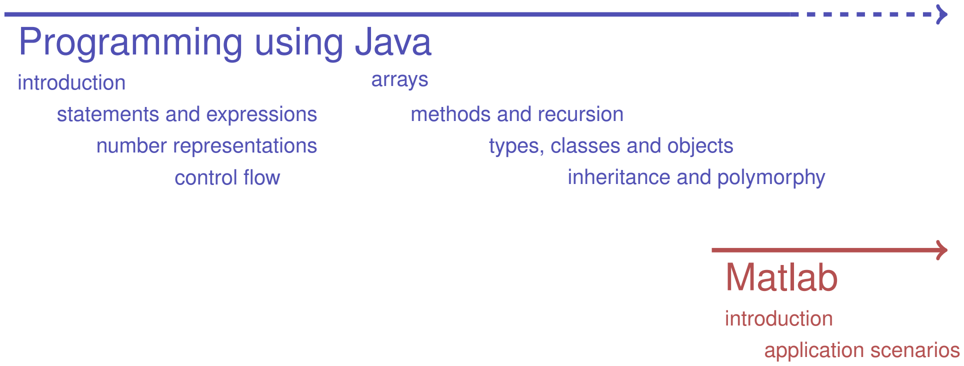
Programming and problem solving

In this course you learn to solve problems with selected algorithms and data structures

- *Fundamental knowledge* independent of the language
- Comparison: musical scale, read music, rythm skills.
- **The problem:** without musical instrument this is no fun.

Hence we combine learning problem solving with learning the programming language Java.

Course Content



Programming using Java

introduction

statements and expressions

number representations

control flow

arrays

methods and recursion

types, classes and objects

inheritance and polymorphy

Matlab

introduction

application scenarios

Goal of *today's* Lecture

- Introduction of computer model and algorithms
- General informations to the course
- Writing a *first program*

1.1 Computer Science and Algorithms

Computer Science, Euclidean Algorithm

What is Computer Science?

- The science of **systematic processing of informations**,. . .
- . . . particularly the automatic processing using digital computers.

(Wikipedia, according to “Duden Informatik”)

Computer Science $\neq$ Computer Literacy

Computer literacy: *user knowledge*

- Handling a computer
- Working with computer programs for text processing, email, presentations . . .

Computer Science *Fundamental knowledge*

- How does a computer work?
- How do you write a computer program?

Inhalt dieser Vorlesung

- Systematic problem solving using algorithms and the programming langauge Java
- Hence:
  not only
  but also programming course.

Algorithm: Fundamental Notion of Computer Science

Algorithm:

- Instructions to solve a problem step by step
- Execution does not require any intelligence, but precision (even computers can do it)
- according to *Muhammed al-Chwarizmi*, author of an arabic computation textbook (about 825)

"Dixit algorizmi…" (Latin translation)

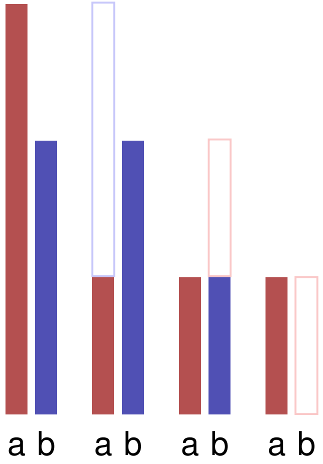
http://de.wikipedia.org/wiki/Algorithmus

Oldest Nontrivial Algorithm

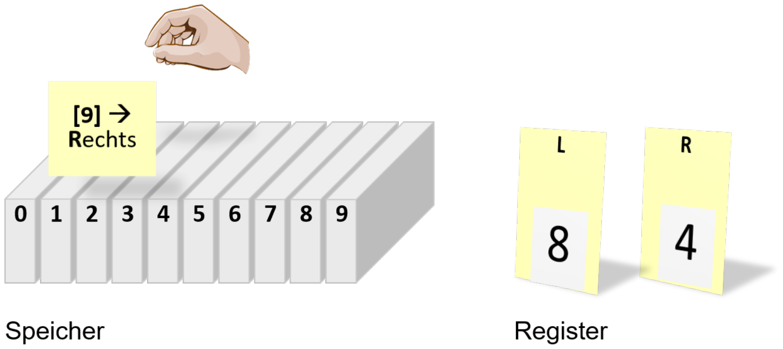
Euclidean algorithm (from the *elements* from Euklid, 3. century B.C.)

- Input: integers $a > 0$, $b > 0$
- Output: gcd of a und b

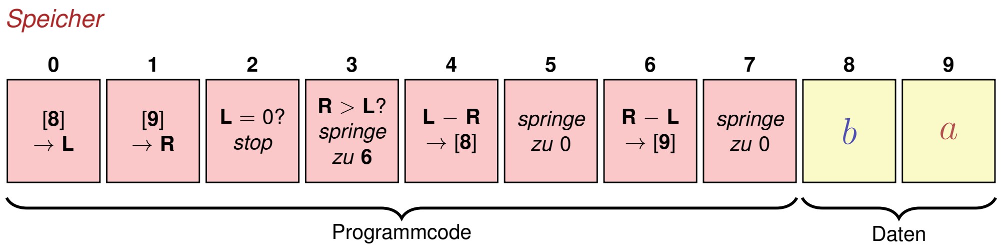
```
While $b \neq 0$
    If $a > b$ then
        $a \leftarrow a - b$
    else:
        $b \leftarrow b - a$
Result: $a$.
```



Live Demo: Turing Machine



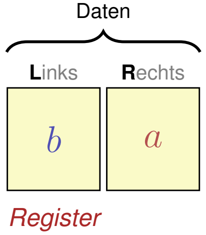
Euklid in the Box



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

Programmcode (0–7) · Daten (8–9)



Daten

Links	**R**echts
b	a

Register

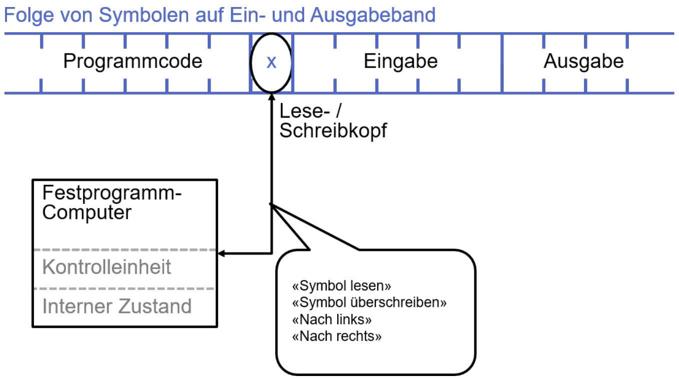
```
While $b \neq 0$
    If $a > b$ then
        $a \leftarrow a - b$
    else:
        $b \leftarrow b - a$
Ergebnis: $a$.
```

1.3 Computer Model

Turing Machine, Von Neumann Architecture

Computers – Concept

An bright idea: universal Turing machine (Alan Turing, 1936)



Folge von Symbolen auf Ein- und Ausgabeband

Programmcode

x

Eingabe

Ausgabe

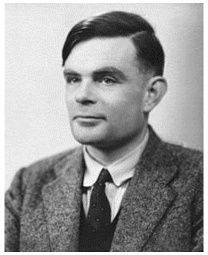
Lese- / Schreibkopf

Festprogramm-Computer

Kontrolleinheit

Interner Zustand

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

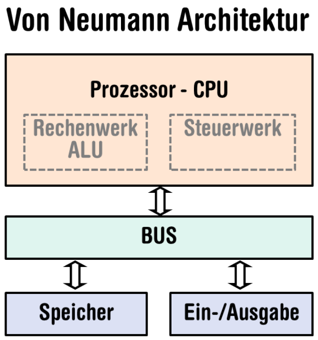


Alan Turing

http://en.wikipedia.org/wiki/Alan_Turing

Computer – Implementation

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



Von Neumann Architektur

Prozessor - CPU

Rechenwerk ALU

Steuerwerk

BUS

Speicher

Ein-/Ausgabe

Konrad Zuse

John von Neumann

http://www.hs.uni-hamburg.de/DE/GNT/hh/biogr/zuse.htm
http://commons.wikimedia.org/wiki/File:John_von_Neumann.jpg

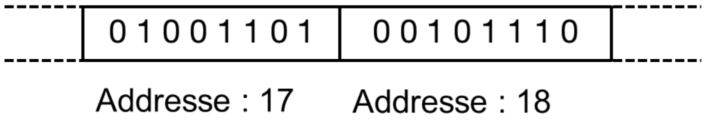
Computer

Ingredients of a *Von Neumann Architecture*

- Memory (RAM) for programs *and* data
- Processor (CPU) to process programs and data
- I/O components to communicate with the world

Memory for data *and* program

- Sequence of bits from $\{0, 1\}$.
- Program state: value of all bits.
- Aggregation of bits to memory cells (often: 8 Bits = 1 Byte)
- Every memory cell has an address.
- Random access: access time to the memory cell is (nearly) independent of its address.



0 1 0 0 1 1 0 1	0 0 1 0 1 1 1 0
Addresse : 17	Addresse : 18

Processor

The processor (CPU)

- executes instructions in machine language
- has an own "fast" memory (registers)
- can read from and write to main memory
- features a set of simplest operations = instructions (e.g. adding to register values)

Computing speed

In the time, onaverage, that the sound takes to travel from from my mouth to you ...

30 m $\widehat{=}$ more than $100.000.000$ instructions

a contemporary desktop PC can process more than 100 millions instructions [1]

[1] Uniprocessor computer at 1 GHz.

Programming

- With a *programming language* we issue commands to a computer such that it does exactly what we want.
- The sequence of instructions is the *(computer) program*

The Harvard Computers, human computers, ca.1890

http://en.wikipedia.org/wiki/Harvard_Computers

Why programming?

- Do I study computer science or what ...
- There are programs for everything ...
- I am not interested in programming ...
- because computer science is a mandatory subject here, unfortunately...
- . . .

Mathematics used to be the lingua franca of the natural sciences on all universities. Today this is computer science.

Lino Guzzella, president of ETH Zurich, NZZ Online, 1.9.2017

This is why programming!

- Any understanding of modern technology requires knowledge about the fundamental operating principles of a computer.
- Programming (with the tool computer) is evolving a cultural technique like reading and writing (using the tools paper and pencil)
- Most qualified jobs require at least elementary programming skills
- Programming is fun!

This Course is for You

- You learn the *fundamental principles* – the basics of computer science and programming – from us on a nontrivial level
- You will need to *apply the principles learned in a different context – for example for other programming languages (C++ ,Python ,Matlab , R)*
- *This is not our requirement – we know this from* you *(= your department)*

Programming Languages

- The language that the computer can understand (machine language) is very primitive.
- Simple operations have to be disassembled into many single steps
- The machine language varies between computers.

Higher Programming Languages

can be represented as program text that

- can be *understood* by humans
- is *independent* of the computer model

  → Abstraction!

Java

- is based on a *virtual machine* (with von-Neumann architecture)
  - Program code is translated into intermediate code
  - Intermediate code runs in a simulated computing envrionment, the intermediate code is executed by an interpreted
  - Optimisation: Just-In-Time (JIT) compilation of frequently used code: virtual machine → physical machine
- Consequence, and manifested goal of the Java developers:

  write once – run anywhere

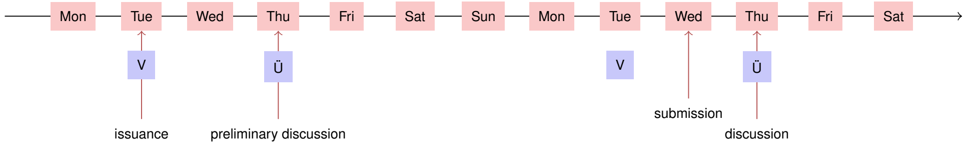
1.5 General Informations about the Course

Organisation, Tools, Exercises, Exams

Recitation Session Registry

- Registration via web page `http://echo.ethz.ch`
- Works only when enrolled for this course via myStudies.
- Available rooms depend on the course of studies.

Exercises



Mon Tue Wed Thu Fri Sat Sun Mon Tue Wed Thu Fri Sat

V

Ü

V

Ü

issuance

preliminary discussion

submission

discussion

- Exercises availabe at lectures.
- Preliminary discussion in the following recitation session
- Solution of the exercise until the day before the next recitation session.
- Dicussion of the exercise in the next recitation session.

Exercises

- At ETH an exercise certificate is not required in order to subscribe for the exams.
- The solution of the weekly exercises is thus voluntary but *stronly* recommended.

Lacking Resources are no Excuse!

For the exercises we use an online development environment that requires only a browser, internet connection and your ETH login.

If you do not have access to a computer: there are a a lot of computers publicly accessible at ETH.

Tutorial

In the first week you work through our *Java-tutorial* on your own

- Simple introduction to Java, no foreknowledge required
- Time needed: about two hours
- In the second week recitation session there will be a *self assessment* about the tutorial
- The tutorial is using **codeboard.io**

→ This time is well-invested!

Tutorial - Url

Java Tutorial

Here you find the tutorial:
`https://frontend-1.et.ethz.ch/sc/WKrEKYAuHvaeTqLzr`

Book to the Lecture

Sprechen Sie Java?

Hanspeter Mössenböck

dpunkt.verlag

['dʒa:və]
→ 5., überarbeitete und erweiterte Auflage
Hanspeter Mössenböck
Sprechen Sie Java?
Eine Einführung in das systematische Programmieren
Bestseller in der 5. Auflage
dpunkt.verlag

- Well structured learning material
- In-depth discussion of the topics
- Exercise tasks with solutions

- *Our exam will include 1-2 questions from the book*

Exams

The exam (in examination period 2018) will cover

- Lectures content (lectures, handouts)
- Exercise content (recitation hours, exercise tasks).

Written exam.

We will test your practical skills (programming skills [2]) and theoretical knowledge (background knowledge, systematics).

[2] as far as possible in a written exam

Offer

- During the semester we offer weekly programming exercises that are graded. Points achieved will be taken as a bonus to the exam.
- The achieved grade bonus is proportional to the achieved points of all exercise series. Achieving all points corresponds to 1/4 grade.

Academic integrity

Rule: You submit solutions that you have written yourself and that you have understood.

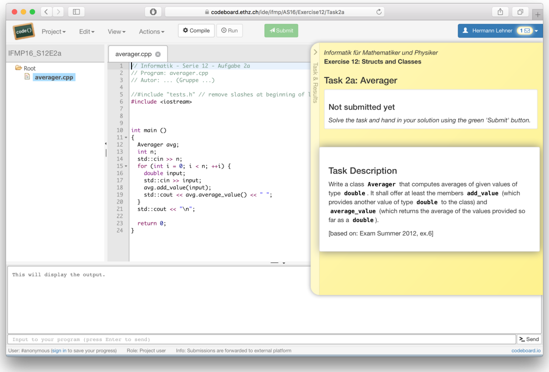
We check this (partially automatically) and reserve our rights to invite you to interviews.

Should you be invited to an interview: don’t panic. Primary we presume your innocence and want to know if you understood what you have submitted.

Codeboard

Codeboard is an online IDE: programming in the browser!

- Bring your laptop / tablet / . . . along, if available.
- You can try out examples in class without having to install any tools.



```
// Informatik - Serie 12 - Aufgabe 2a
// Program: averager.cpp
// Autor: ... (Gruppe ...)

//#include "tests.h" // remove slashes at beginning of l
#include <iostream>



int main ()
{
  Averager avg;
  int n;
  std::cin >> n;
  for (int i = 0; i < n; ++i) {
    double input;
    std::cin >> input;
    avg.add_value(input);
    std::cout << avg.average_value() << " ";
  }
  std::cout << "\n";

  return 0;
}
```

```
This will display the output.
```

Input to your program (press Enter to send)

User: #anonymous (sign in to save your progress) Role: Project user Info: Submissions are forwarded to external platform

Informatik für Mathematiker und Physiker
Exercise 12: Structs and Classes

Task 2a: Averager

Not submitted yet

Solve the task and hand in your solution using the green 'Submit' button.

Task Description

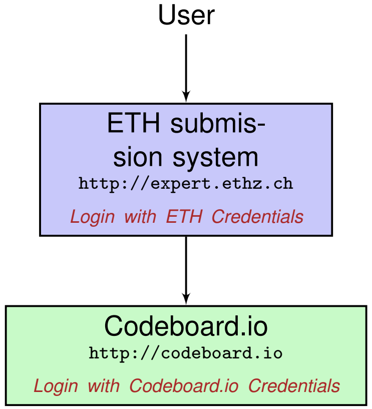
Write a class **`Averager`** that computes averages of given values of type **`double`** . It shall offer at least the members **`add_value`** (which provides another value of type **`double`** to the class) and **`average_value`** (which returns the average of the values provided so far as a **`double`**).

[based on: Exam Summer 2012, ex.6]

Expert

Our exercise system consists of two independent systems that communicate with each other:

- **The ETH submission system:** Allows us to evaluate your tasks.
- **The online IDE:** The programming environment



Exercise Registration

Codeboard.io Registration

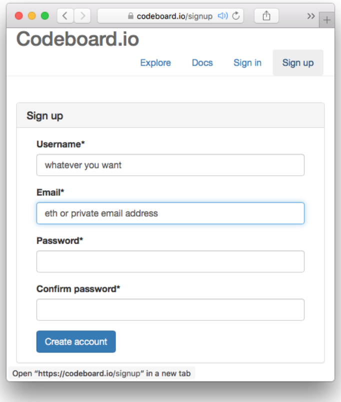
Go to `http://codeboard.io` and create an account, stay logged in.

Registration for exercises

Go to `http://expert.ethz.ch/baugi1_2017e00t01` and inscribe for one of the exercise groups there.

Codeboard.io Registration

If you do not yet have an **Codeboard.io** account ...



codeboard.io/signup

Codeboard.io

Sign up

Username*

whatever you want

Email*

eth or private email address

Password*

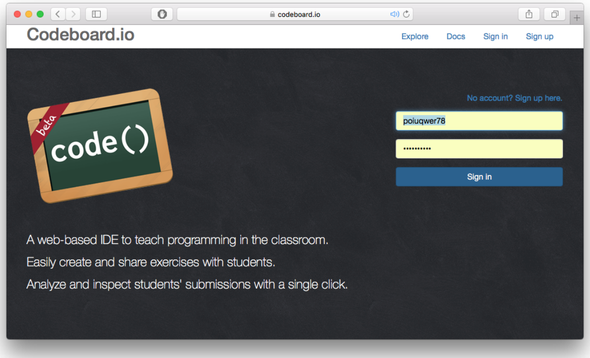
Confirm password*

Open “https://codeboard.io/signup” in a new tab

- We use the online IDE **Codeboard.io**
- Create an account to store your progress and be able to review submissions later on
- Credentials can be chose arbitrarily *Do not use the ETH password.*

Codeboard.io Login

If you have an account, log in:



codeboard.io

Codeboard.io

poiuqwer78

••••••••••

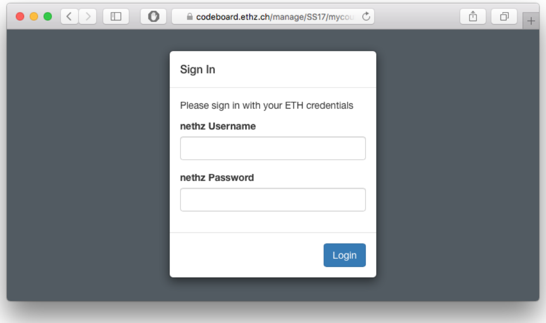
A web-based IDE to teach programming in the classroom.

Easily create and share exercises with students.

Analyze and inspect students' submissions with a single click.

Exercise group registration I

- Visit `http://expert.ethz.ch/baugi1_2017e00t01`
- Log in with your nethz account.



codeboard.ethz.ch/manage/SS17/mycou

Sign In

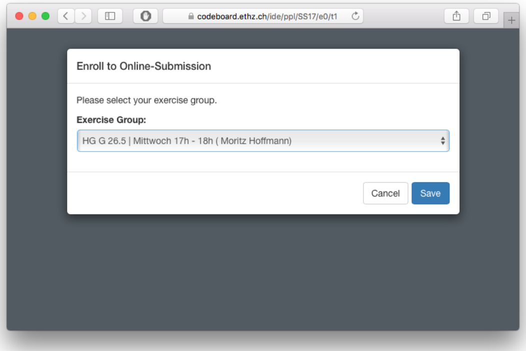
Please sign in with your ETH credentials

nethz Username

nethz Password

Exercise group registration II

Register with this dialog for an exercise group.



codeboard.ethz.ch/ide/ppl/SS17/e0/t1

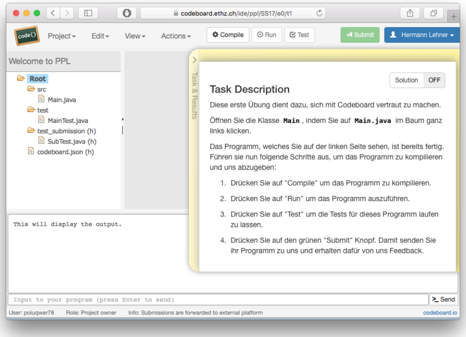
Enroll to Online-Submission

Please select your exercise group.

Exercise Group: HG G 26.5 | Mittwoch 17h - 18h (Moritz Hoffmann)

The first exercise.

You are now registered and the first exercise is loaded. Follow the instructions in the yellow box.



codeboard.ethz.ch/ide/ppl/SS17/e0/t1

Task Description

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.

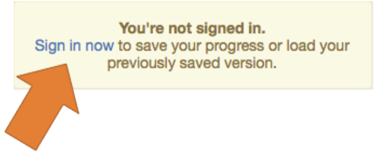
```
This will display the output.
```

```
Input to your program (press Enter to send)
```

User: poiuqwer78 Role: Project owner Info: Submissions are forwarded to external platform

The first exercise – codeboard.io login

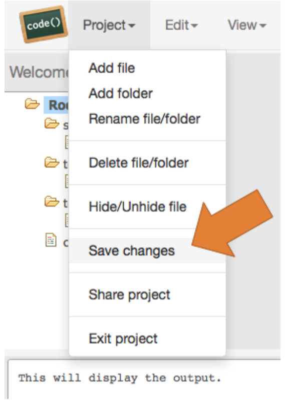
Attention If you see this message, click on Sign in now and register with you **codeboard.io** account.



You're not signed in.
Sign in now to save your progress or load your previously saved version.

The first exercise – store progress

Attention! Store your progress regularly. So you can continue working at any different location.



```
This will display the output.
```

2. Introduction to Java

Programming – a first Java Program

Programming Tools

- **Editor:** Program to modify, edit and store Java program texts
- **Compiler:** program to translate a program text into machine language
- **Computer:** machine to execute machine language programs
- **Operating System:** program to organize all procedures such as file handling, editor-, compiler- and program execution.

German vs. Programming Language

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 and Semantics

- Like our langauge, programs have to be formed according to certain rules.
  - Syntax: Connection rules for elementary symbols (characters)
  - Semantics: interpretation rules for connected symbols.
- Corresponding rules for a computer program are simpler but also more strict because computers are relatively stupid.

Syntax and Semantics of Java

Syntax

- What *is* a Java program?
- Is it *grammatically* correct?

Semantics

- What does a program *mean*?
- What kind of algorithm does a program implement?

First Java Program

```
// 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);
  }
}
```

← *Class*: a program

← *Method*: named sequence of statements.

Java Classes

A Java program comprises at least one class with main-method. The sequence of statements in this method is executed when the program starts.

```
public class Test{
  // Potentiell weiterer Code und Daten

  public static void main(String[] args) {
    // Hier beginnt die Ausfuehrung
    ...
  }
}
```

Behavior of a Program

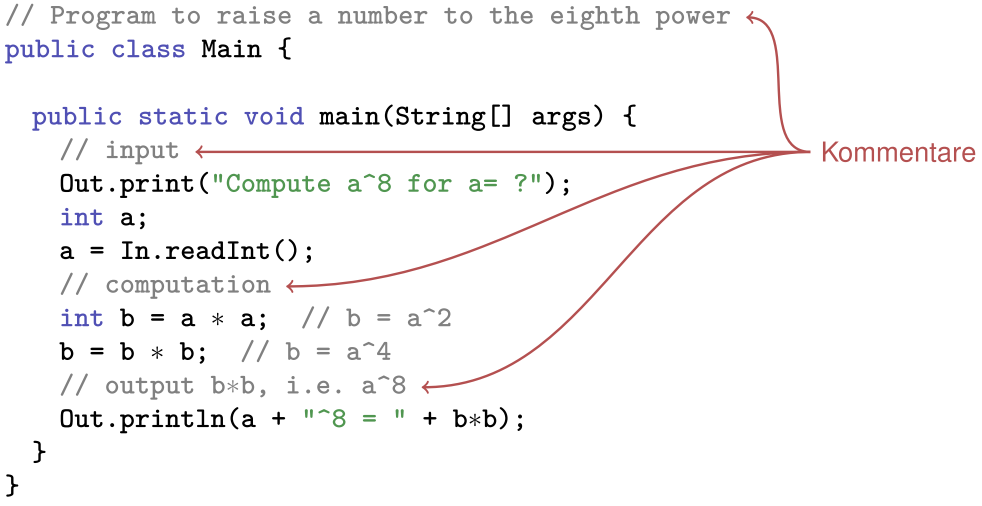
At compile time:

- program accepted by the compiler (syntactically correct)
- Compiler error

During runtime:

- correct result
- incorrect result
- program crashes
- program does not terminate (endless loop)

Comments



```
// 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

Comments and Layout

Comments

- are contained in every good program.
- document, *what* and *how* a program does something and how it should be used,
- are ignored by the compiler
- Syntax: “double slash” `// until the line end.`

The compiler *ignores* additionally

- Empty lines, spaces,
- Indendations that should reflect the program logic

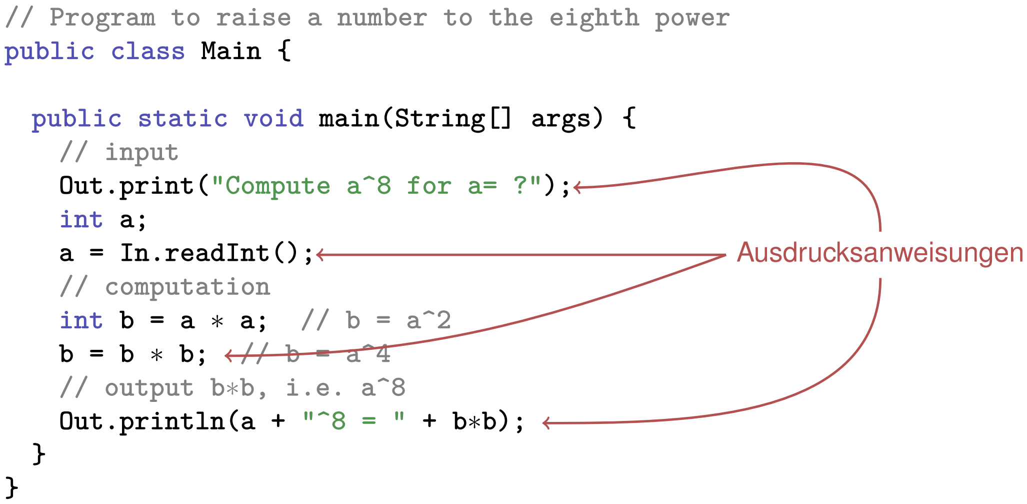
Comments and Layout

The compiler does not care...

```
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);}}
```

... but we do!

Statements



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

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

Ausdrucksanweisungen

Statements

- building blocks of a Java program
- are *executed* (sequentially)
- end with a semicolon
- Any statement provide an *effect* (potentially)

Expression Statements

- have the following form:

  expr;

  where *expr* is an expression
- Effect is the effect of *expr*, the value of *expr* is ignored.

Example: `b = b*b;`

Statements – Values and Effects

```
// 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

Values and Effects

- determine what a program does,
- are purely semantical concepts:
  - Symbol 0 means Value $0 \in \mathbb{Z}$
  - `a = In.readInt();` means effect "read in a number"
- depend on the program state (memory content, inputs)

Variable Definitions

```
// 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

Declaration Statements

- introduce new names in the program,
- consist of declaration and semicolon

Example: `int a;`

- can initialize variables

Example: `int b = a * a;`

Types and Functionality

`int`:

- Java integer type
- corresponds to $(\mathbb{Z}, +, \times)$ in math

In Java each type has a name and

- a domain (e.g. integers)
- functionality (e.g. addition/multiplication)

Fundamental Types

Java comprises fundamental types for

- integers (`int`)
- real numbers (`float`, `double`)
- boolean values (`boolean`)
- ...

Literals

- represent constant values
- have a fixed *type* and *value*
- are "syntactical values".

Examples:

- `0` has type `int`, value 0.
- `1.2e5` has type `double`, transWertvalue $1.2 \cdot 10^5$.

Variables

- represent (varying) values,
- have
  - *name*
  - *type*
  - *value*
  - *address*
- are "visible" in the program context.

Beispiel

`int a;` defines a variable with

- name: `a`
- type: `int`
- value: (initially) undefined
- Address: determined by compiler

Objects

- represent values in main memory
- have *type*, *address* and *value* (memory content at the address)
- can be named (variable) ...
- ... but also anonymous.

Remarks

A program has a *fixed* number of variables. In order to be able to deal with a variable number of value, it requires "anonymous" addresses that can be address via temporary names.

Identifiers and Names

(Variable-)names are identifiers

- allowed: A,...,Z; a,...,z; 0,...,9;_
- First symbol needs to be a character.

There are more names:

- `Out.println` (Qualified identifier)

Expressions

- represent *Computations*
- are either primary (`b`)
- or composed (`b*b`). . .
- . . . from different expressions by operators

Analogy: building blocks

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| ); ||
```

Expressions

- represent *computations*
- are *primary* or *composite* (by other expressions and operations)

  `a * a`
  composed of
  variable name, operator symbol,variable name
  variable name: primary expression

- can be put into parantheses

  `a * a` is equivalent to `(a * a)`

Expressions

have *type*, *value* und *effect* (potentially).

Example

`a * a`

- type: `int` (type of the operands)
- Value: product of `a` and `a`
- Effect: none.

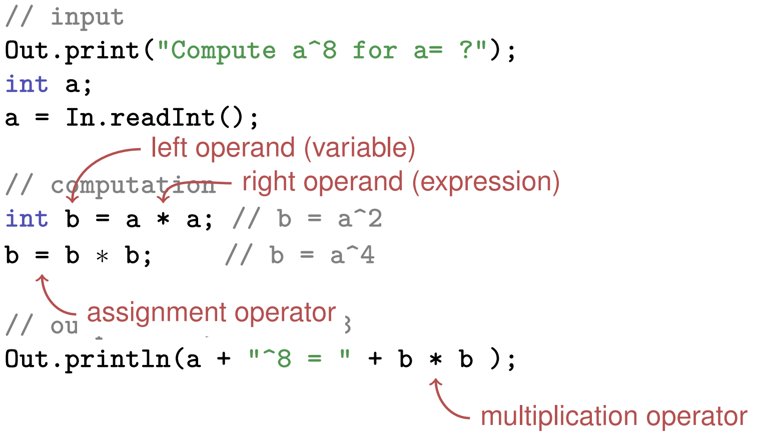
Example

`b = b * b`

- type: `int` (Typ der Operanden)
- Value: product of `b` and `b`
- effect: assignment of the product value to `b`

The type of an expression is fixed but the value and effect are only determined by the *evaluation* of the expression

Operators and Operands



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

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

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

left operand (variable)

right operand (expression)

assignment operator

multiplication operator

Operators

Operators

- make expressions (*operands*) into new composed expressions
- specify the required and resulting types for the operands and the result
- have an arity

Multiplication Operator *

- expects to R-values of the same type as operands (arity 2)
- "returns the product as value of the same type", that means formally:
  - The composite expression is value of the product of the value of the two operands

Examples: `a * a` and `b * b`

Assignment Operator =

- Assigns to the left operand the value of the right operand and returns the left operand

Examples: `b = b * b` and `a = b`

Attention, Trap!

The operator = corresponds to the assignment operator of mathematics (:=), not to the comparison operator (=).