



LUDWIG-  
MAXIMILIANS-  
UNIVERSITY  
MUNICH



DEPARTMENT  
INSTITUTE FOR  
INFORMATICS



DATABASE  
SYSTEMS  
GROUP

# Kapitel 1

## Einführung

Skript zur Vorlesung  
Einführung in die Programmierung  
im Wintersemester 2012/13  
Ludwig-Maximilians-Universität München  
(c) Peer Kröger, Arthur Zimek 2009, 2012



- Was ist Informatik
- Die Programmiersprache Java

Franz.: *Informatique* (= information + mathématique)

Engl.: *Computer Science* (neuerdings auch *Informatics*)

- DUDEN Informatik:  
„Wissenschaft von der systematischen Verarbeitung von Informationen, besonders der automatischen Verarbeitung mit Computern.“
- Gesellschaft für Informatik (GI):  
„Wissenschaft, Technik und Anwendung der maschinellen Verarbeitung und Übermittlung von Informationen.“
- Association for Computer Machinery (ACM):  
„Systematic study of algorithms and data structures.“

**Primäres Lernziel:** Nicht die Komponenten eines Computers, Anwendungsprogramme oder Programmiersprachen sondern  
*Prinzipien und Techniken zur Verarbeitung von Information.*

Informatik an einer Universität ist ein *wissenschaftliches Studium*.

Was bedeutet dies?

- Neue Denkweisen
- Grundlagenorientiert
- Steilerer Anstieg, höheres Niveau
- Angebote statt Zwang und Anwesenheitspflicht

Lernziel Nummer 1 an einer Universität für jedes Studienfach  
*Eigenverantwortung*

- Was bedeutet Eigenverantwortung?
  - Teilweise ist der Stoff sehr schwierig/theoretisch (noch mehr in den Mathematik-Vorlesungen)
  - Trotzdem: Im Gegensatz zur Schule ist nicht mehr der Lehrer für Ihren Lernerfolg verantwortlich.
  - Das Erarbeiten des Stoffes ist *Ihre eigene Verantwortung*.
- Zusätzlich zum Besuch von Vorlesungen und Übungen erforderlich:
  - Recherche von Literatur zum Verstehen des Stoffes
  - Selbständiges Lösen der Aufgaben
  - Vor Besuch der Vorlesung Sichtung des Materials

- Informatik hat sich u.a. aus der Mathematik entwickelt.
- Viele Vorgehensweisen aus der Mathematik entlehnt:
  - Definition, Satz, Beweis...
  - Ein Computer hat die Aufgabe, *Berechnungen* durchzuführen.
  - Er rechnet nicht nur mit Zahlen, sondern mit *Informationen*.
    - z.B. *Wörter* aus Websites bei einer Google-Recherche.
- Analysis, Algebra, Statistik und Numerik sind wichtige Vorlesungen.

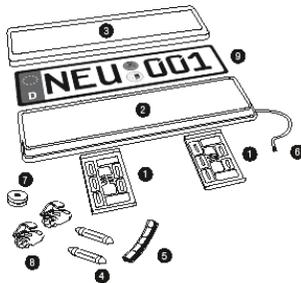
- In dieser Vorlesung lernen wir vor allem:
  - Einführung in die Programmierung
  - Anhand der Programmiersprache Java  
(ABER: dies ist kein Java-Programmierkurs!!!)
  - Entwerfen von einfachen Algorithmen
  - Einfache Datenstrukturen (Darstellungsmöglichkeiten für Daten)
  - Grundlagen der Objektorientierung
- In späteren Vorlesungen werden wir beispielsweise lernen
  - Andere Paradigmen, z.B. das „funktionale Programmieren“
  - Software-Engineering, also Software-Entwicklung in Teams
  - Hardware-Grundlagen der Informatik
  - Analysemethoden für Probleme, Algorithmen, Programme... uvm

- Algorithmus
  - Grundlage jeglicher maschineller Informationsverarbeitung
  - Zentraler Begriff der Informatik
  - Systematische, „schematisch“ („automatisch“, „mechanisch“) ausführbare Verarbeitungsvorschrift
- Wichtige Inhalte des Informatikstudiums:
  - Entwicklung von Algorithmen
  - Analyse von Algorithmen (Korrektheit, Laufzeit, Eigenschaften)
  - Oft weniger wichtig: Umsetzung in Programmiersprachen

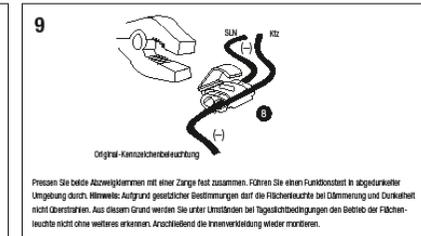
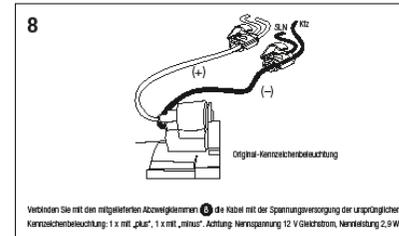
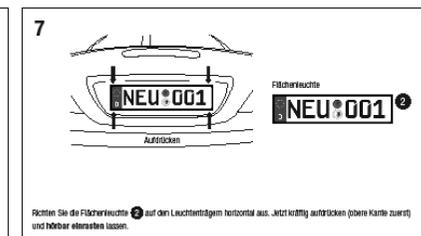
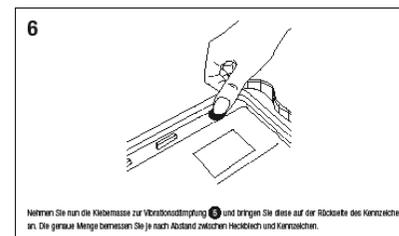
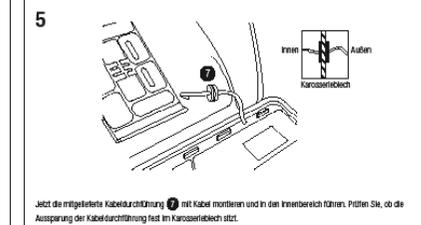
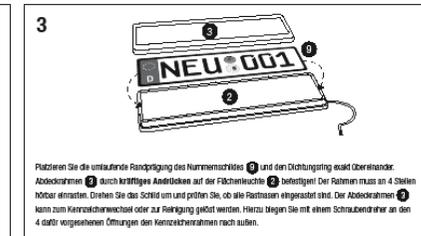
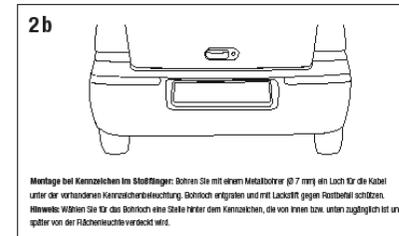
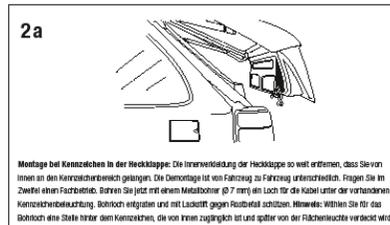
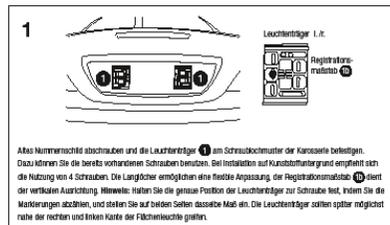
Unser Alltag ist von Algorithmen geprägt ...

- Algorithmus: Beispiele
  - Montaganleitung:

## Einzelteile



- 1 Leuchenträger
- 2 Flächenleuchte
- 3 Abdeckrahmen
- 4 Lastwiderstände
- 5 Klebmasse zur Vibrationsdämpfung
- 6 Kabel
- 7 Kabeldurchführung
- 8 Abzweigklemmen
- 9 Nummernschild (nicht in Packing enthalten)



- Kochrezept für einen Afrikanischen Hühnertopf ([www.chefkoch.de](http://www.chefkoch.de))
  - Zutaten:
    - 1 Huhn (küchenfertige Poularde), 125 ml Hühnerbrühe, 8 Tomaten, 150 g Erdnussbutter, 2 Zucchini, Salz und Pfeffer, Rosmarin.
  - Zubereitung:
    - Von der Poularde die Haut abziehen, Huhn zerteilen.
    - Hühnerteile abspülen und in einen breiten Topf legen.
    - Die Brühe angießen und langsam zum Kochen bringen.
    - Die Tomaten mit kochendem Wasser übergießen, enthäuten und unzerteilt zum Fleisch geben.
    - Zugedeckt bei geringer Hitze ca. 40 Minuten köcheln.
    - Nach 30 Minuten Garzeit die Erdnussbutter (Erdnüsse) einrühren.
    - Mit Salz, Pfeffer und Rosmarin abschmecken.
    - Die Zucchini putzen, Blüten und Stengelansatz entfernen.
    - Zucchini waschen und in kleine Würfel schneiden.
    - 10 Minuten vor Garzeitende in den Topf geben und alles noch einmal abschmecken.

- Beispiel:

Algorithmus zur Berechnung des größten gemeinsamen Teilers (GGT) von zwei natürlichen Zahlen  $a$  und  $b$ .

(Beschrieben von Euklid von Alexandria (300 v.Chr.) im Werk „Die Elemente“)

## Algorithmus *GGT*

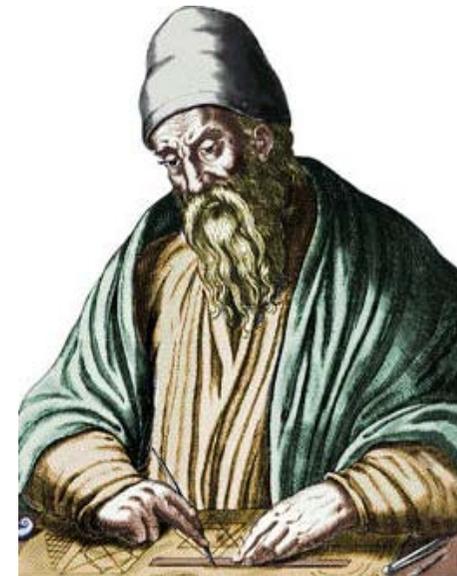
**Eingabe:**  $a, b \in \mathbb{N}$

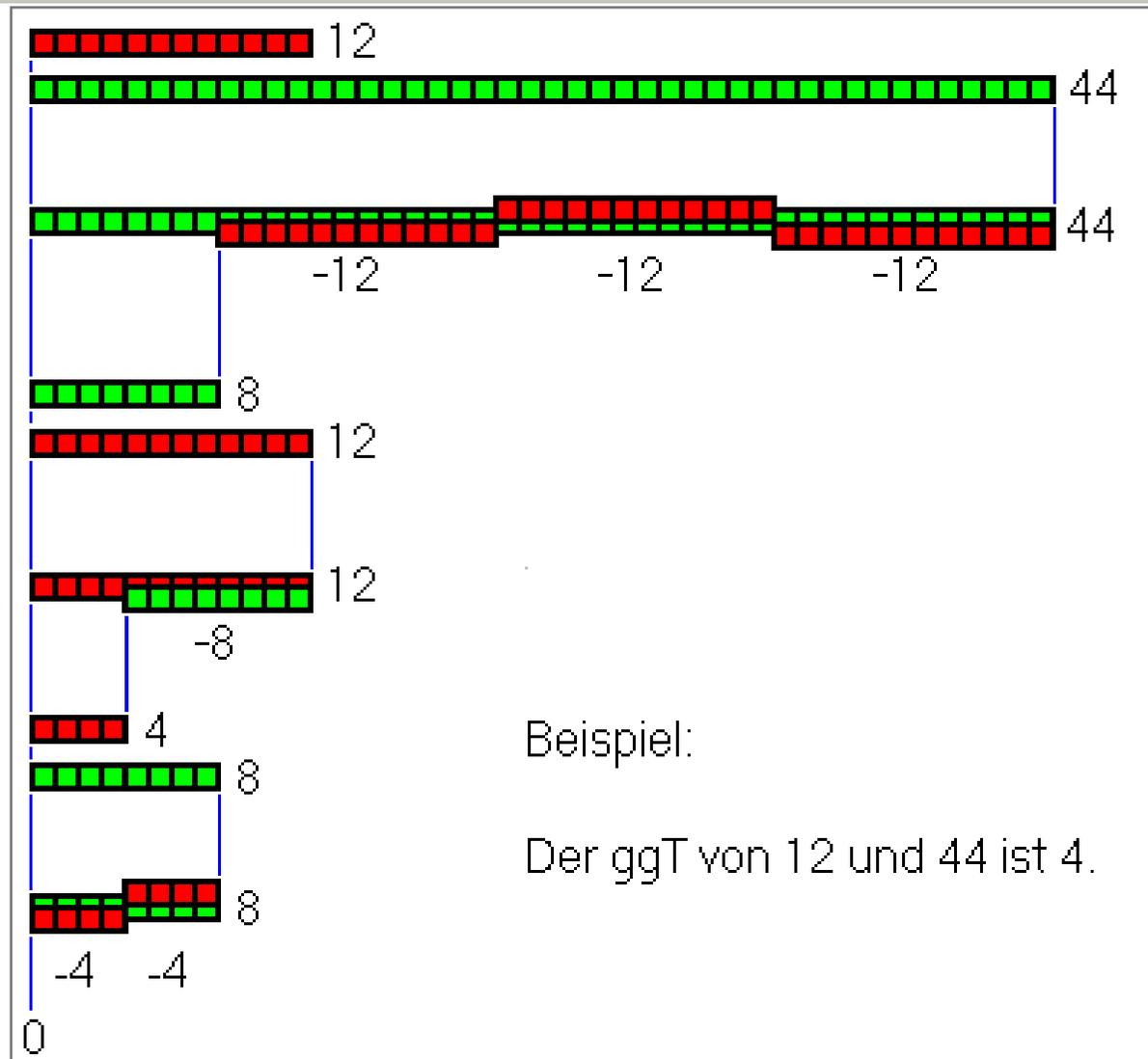
```

1  solange  $b \neq 0$ 
2      wenn  $a > b$ 
3          dann  $a := a - b$ 
4          sonst  $b := b - a$ 

```

**Ausgabe:**  $a$





- Es ergeben sich folgende interessante Fragen:
  - Ist der Algorithmus *korrekt*?
    - Also: berechnet er wirklich den GGT aus  $(a,b)$ ?
    - Wieso funktioniert dieser Algorithmus überhaupt?
  - Ist der Algorithmus *vollständig*?
    - Kann ich wirklich jedes Paar  $(a,b)$  eingeben?
  - Ist der Algorithmus *terminierend* (evtl. Endlosschleife)?
  - Ist der Algorithmus *effizient*?
    - Wie viel Speicher benötigt er?
    - Wie viel Zeit (d.h. Verarbeitungsschritte) benötigt er?
    - Wie hängt das von der Eingabe  $(a,b)$  ab?
- Wir müssen solche Aussagen immer mathematisch *beweisen*.

Definition: Ein *Algorithmus* ist ein Verfahren zur Verarbeitung von Daten mit einer präzisen, endlichen Beschreibung unter Verwendung effektiver, elementarer Verarbeitungsschritte

- **Daten:**  
Die Repräsentation und der Wertebereich von Eingabe und Ergebnissen müssen eindeutig definiert sein.
- **Präzise, endliche Beschreibung:**  
Die Abfolge von Schritten muss in einem endlichen Text in einer eindeutigen Sprache genau festgelegt sein.
- **Effektiver Verarbeitungsschritt:**  
Jeder einzelne Schritt ist tatsächlich ausführbar.
- **Elementarer Verarbeitungsschritt:**  
Jeder Schritt ist entweder eine Basisoperation, oder ist selbst durch einen Algorithmus spezifiziert.

- Typische Elemente, die in Algorithmen enthalten sind:
  - Schrittweises Vorgehen
  - Fallunterscheidungen und Wiederholungen („Schleifen“)
  - Zur Berechnung erforderliche Informationen in „Variablen“
  - Eingabe: Parameter der Problemstellung
    - z.B. das Wort nach dem im Web gesucht wird, bzw. die beiden Zahlen deren GGT gesucht ist
  - Ausgabe: Lösung des Problems
    - Liste der Treffer-Websites geordnet nach statist. Relevanz, bzw. der GGT

**Algorithmus GGT**

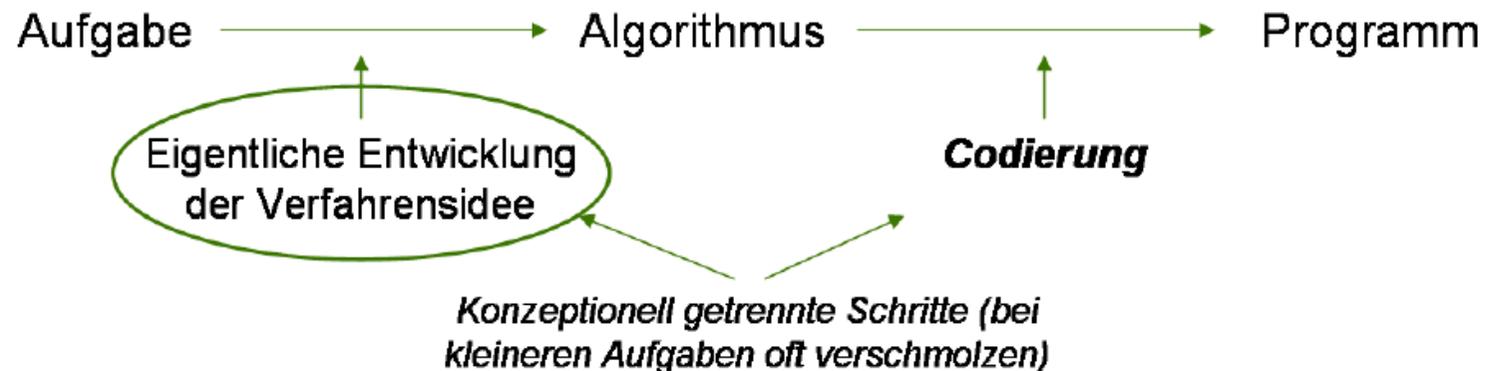
**Eingabe:**  $a, b \in \mathbb{N}$

```
1   solange  $b \neq 0$ 
2       wenn  $a > b$ 
3           dann  $a := a - b$ 
4           sonst  $b := b - a$ 
```

**Ausgabe:**  $a$

- **Zentrale Aufgabe des Informatikers:**  
Entwicklung von Algorithmen (und oft auch deren Realisierung auf dem Rechner als Programm)
- **Programm:**  
Formale Darstellung eines Algorithmus (oder mehrerer) in einer Programmiersprache
- **Programmiersprache:**  
Formale (eindeutige) Sprache, stellt insbesondere elementare Verarbeitungsschritte und eindeutig definierte Datentypen für die Ein-/Ausgabe zur Verfügung

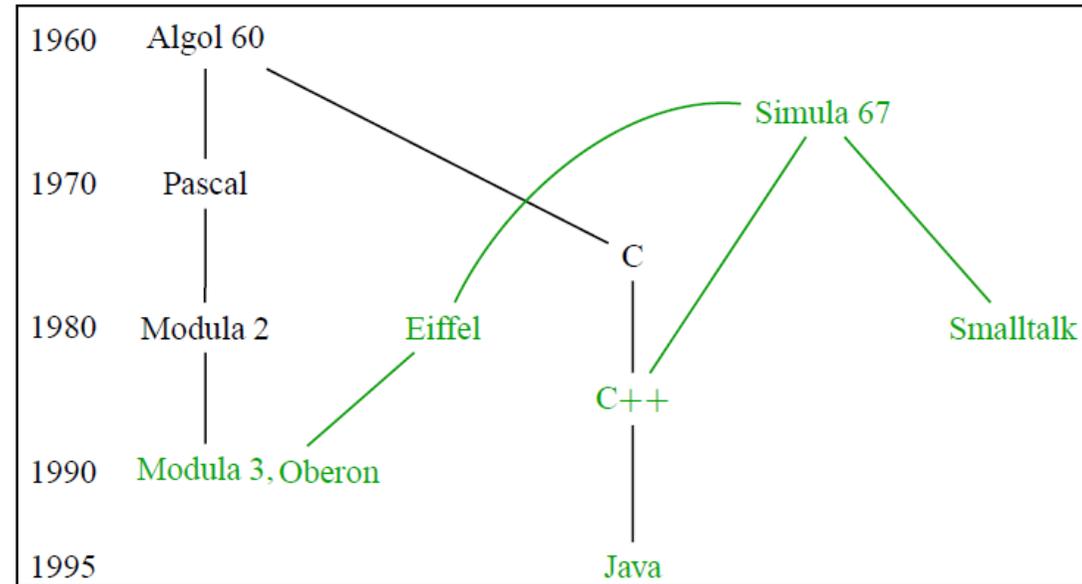
- Schematisch:



- In dieser Vorlesung
  - Konzepte, Methoden und Techniken
    - zur Darstellung und Strukturierung von Daten
    - zur Entwicklung von Algorithmen
  - KEIN Programmierkurs (aber Anwendung des Erlernten mit Java)

- Teilgebiete der Informatik
  - Theoretische Informatik:  
Theoretische Durchdringung und Grundlegung von Fragen und Konzepten der Informatik (z.B. “Berechenbarkeit” von Aufgabestellungen, “Komplexität” von Aufgabenstellungen und Lösungen).
  - Technische Informatik:  
Beschäftigung mit der Hardware (z.B. Aufbau und Wirkungsweise von Computern, maschinengerechte Darstellung von Daten und Algorithmen).
  - Praktische Informatik:  
Konstruktion, Darstellung und Ausführung von Algorithmen (Software).

- Geschichte



- Entwickelt von J. Gosling, Bill Joy, P. Naughton, u.a.
- Erste plattformunabhängige OO-Sprache, insbesondere zur Programmierung von Internet-Applikationen
- Erste Version 1.0 1995, heute Java 1.7 (auch: 7.0)
  - ACHTUNG: wir benutzen die Version 1.6 (auch 6.0)
- Ursprünglicher Name: OAK

- Eigenschaften
  - objektorientiert: Klassenkonzept, strenge Typisierung
  - plattformunabhängig: Übersetzung in Virtuelle Maschine (JVM)
  - netzwerkfähig, nebenläufig
  - Sicherheitsaspekt in der Entwicklung der Sprache wichtig
  - Nachteile:
    - Laufzeithandicap durch Interpretation (JVM), wird aber stetig verbessert
  - Vorteile:
    - Plattformunabhängigkeit
    - verteilte Anwendungen, Web-Anwendungen
    - Rechnerunabhängigkeit von Graphikanwendungen

- Grober Aufbau eines Java-Programms
  - Ein Java-Programm besteht aus Klassen und Schnittstellen
  - Eine Klasse besteht aus
    - Klassenvariablen: beschreiben Eigenschaften aller Objekte dieser Klasse.
    - Attributen (fields, Instanzvariablen): beschreiben den Zustand eines Objekts dieser Klasse.
    - statischen Methoden: Prozeduren einer Klasse, unabhängig vom Zustand eines Objekts
    - Objekt-Methoden: Operationen, die ein Objekt ausführen kann, abhängig vom Zustand des Objektes
    - Konstruktoren: Operationen zur Erzeugung von Objekten einer bestimmten Klasse

Wir betrachten nun zunächst nur die statischen Elemente. Zu den Aspekten der Objektorientierung kommen wir zu einem späteren Zeitpunkt der Vorlesung.

Ein einfaches imperatives Java-Programm besteht aus nur einer Klassendeklaration und einer Methode "main":

```
public class KlassenName
{
    public static void main(String[] args)
    {
        // Anweisungen ...
    }
}
```

Die Textdatei, die den Java-Code enthält, heißt `KlassenName.java`, also genauso wie die enthaltene Klasse, mit der Endung `.java`.

- Das kanonische Beispiel HelloWorld.java

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

(Es ist zum jetzigen Zeitpunkt noch nicht wichtig, das alles genau zu verstehen. Es reicht, wenn Sie diesen Programm-„Container“ zunächst einfach anwenden können. Versuchen Sie es doch einfach mal nur abzuschreiben und ein bisschen damit zu „spielen“)

In der Java-Programmierung gibt es einige Konventionen. Deren Einhaltung erleichtert das Lesen von Programmen. Beispiele solcher Konventionen sind:

- Klassennamen beginnen mit großen Buchstaben: `HelloWorld`
- Methodennamen, Attributnamen und Variablennamen beginnen mit kleinen Buchstaben, z.B.
  - Methoden: `main`, `println`,
  - Klassenvariable: `out`,
  - Parametervariable: `args`
- Zusammengesetzte Namen werden zusammengeschrieben, jeder innere Teilname beginnt mit einem großen Buchstaben:  
Klassenname `HelloWorld`

*(Immer noch: Sie müssen noch nicht wissen, was das alles genau ist, aber merken Sie sich`s trotzdem schon einmal)*

- Plattformunabhängigkeit
  - Bei vielen Sprachen (z.B. C/C++) erzeugt der Compiler Plattform-abhängigen Maschinencode
  - Dieser Maschinencode kann nur auf bestimmten Rechnerarchitekturen/Betriebssystemen ausgeführt werden.
  - Sog. Skript-Sprachen (z.B. Perl, PHP, auch SML) werden interpretiert von einem plattformspezifischen Interpreter
  - Die Programme sind plattformunabhängig, aber der Sourcecode bleibt unübersetzt und sichtbar, was in vielen Anwendungen nicht erwünscht ist.

- Plattformunabhängigkeit (cont.)
  - Plattformunabhängigkeit eines Java-Programmes wird durch einen Kompromiss erreicht:
  - Der Sourcecode wird übersetzt in Bytecode, der plattformunabhängig verwendet werden kann.
  - Bytecode wird von einer virtuellen Maschine ausgeführt (interpretiert).
  - Die virtuelle Maschine gibt es in verschiedenen Versionen für verschiedene Plattformen (JVM = Java Virtual Machine, Teil des JRE = Java Runtime Environment).

- Übersetzung in Bytecode
  - Aus einer Textdatei `KlassenName.java` erzeugt der Java Compiler `javac` eine Binärdatei `KlassenName.class`
    - Beispiel:

```
javac HelloWorld.java
```

erzeugt die Binärdatei `HelloWorld.class`
  - Die Binärdatei `KlassenName.class` enthält den Bytecode für die JVM
  - Der Compiler `javac` ist Teil des JDK (= Java Development Kit). Das JDK enthält JRE, Sie benötigen also das JDK für die Übungen zu dieser Vorlesung.

- Ausführung

- Die Binärdatei `KlassenName.class` wird der JVM übergeben und von dieser ausgeführt (interpretiert).

- Durch den Aufruf

```
java KlassenName
```

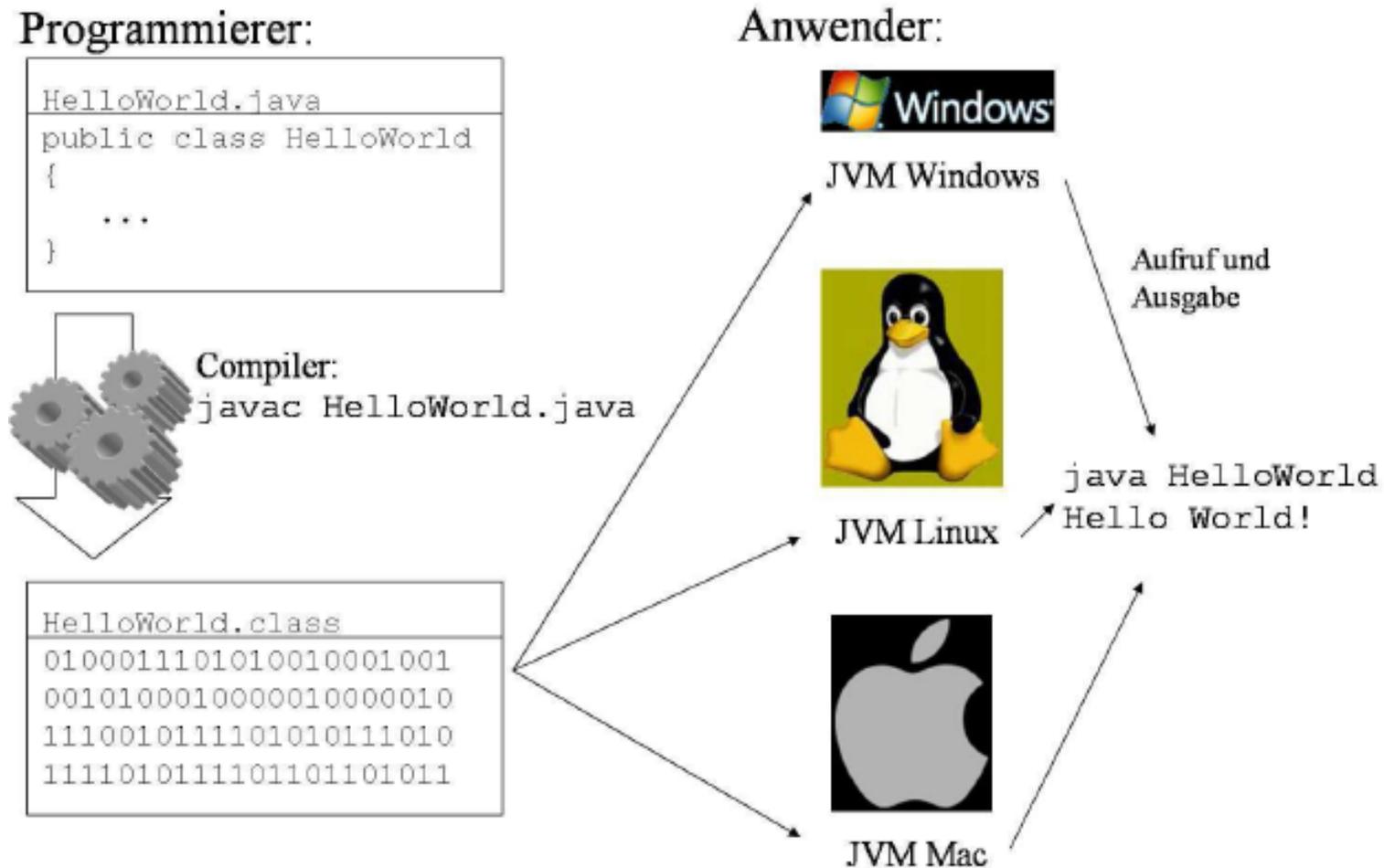
wird die `main`-Methode der Klasse `KlassenName` aufgerufen.

- Beispiel: der Aufruf

```
java HelloWorld
```

gibt "Hello, World!" auf dem Bildschirm aus.

- Überblick



- Kommentare in Java

*"The view that documentation is something that is added to a program after it has been commissioned seems to be wrong in principle, and counterproductive in practice. Instead, documentation must be regarded as an integral part of the process of design and coding."*

C. A. R. Hoare (Turing-Preisträger 1980): Hints on Programming Language Design, 1973

- Es gibt in Java 3 Arten von Kommentaren:
  - Einzeilige Kommentare beginnen mit `//` und enden am Ende der aktuellen Zeile
  - Mehrzeilige Kommentare beginnen mit `/*` und enden mit `*/`
  - Dokumentationskommentare beginnen mit `/**` und enden mit `*/` und können sich ebenfalls über mehrere Zeilen erstrecken.
- Kommentare derselben Art sind nicht schachtelbar. Ein Java-Compiler akzeptiert aber einen einzeiligen innerhalb eines mehrzeiligen Kommentars und umgekehrt.
- Dokumentationskommentare dienen dazu, Programme im Quelltext zu dokumentieren. Sie werden in den mit dem Befehl `javadoc` erzeugten Report mit aufgenommen.

- Beispiel: Die Klasse HelloWorld dokumentiert

```
/**
 * HelloWorld Klasse um eine einfache Benutzung einer java-Klasse zu illustrieren.
 *
 * Diese Klasse dient nur dem Anzeigen des Strings
 * "Hello, world!" auf dem Bildschirm
 *
 * @author Arthur Zimek
 */
public class HelloWorld
{
    /**
     * Die main-Methode wird automatisch aufgerufen, wenn die Klasse mit
     * java HelloWorld
     * aufgerufen wird.
     *
     * Die Methode main druckt "Hello, world!" auf die Standard-Ausgabe.
     *
     * @param args Array mit Parametern - wird von dieser Methode nicht verwendet.
     */
    public static void main(String[] args)
    {
        // Ausgabe von "Hello World!" auf die Standard-Ausgabe
        System.out.println("Hello World!");
    }
}
```

- Verarbeiten von Kommentaren zu Dokumentationen

- Mit dem Befehl

```
javadoc HelloWorld.java
```

wird automatisch eine Beschreibung der Klasse HelloWorld erzeugt und in die Datei

```
HelloWorld.html
```

geschrieben.

- Die Klassenbeschreibung wird eingebettet in eine organisierte Menge von html-Dokumenten:

Siehe Vorlesungshomepage

- Diese Dokumentation kann auch für viele Klassen gleichzeitig erfolgen (z.B. mit dem Befehl `javadoc *.java`).

- Spezielle Tags
  - Die durch @ eingeleiteten Elemente in einem Dokumentationskommentar haben eine besondere Bedeutung, z.B.:
    - @see für Verweise
    - @author für Name des Autors / Namen der Autoren
    - @version für die Version
    - @param für die Methodenparameter
    - @return für die Beschreibung des Ergebnisses einer Methode
- Java Application Programming Interface (API)
  - Auch die Bibliothek der Standard Edition ist mit javadoc erzeugt:  
<http://docs.oracle.com/javase/6/docs/api/>
  - Für das fortgeschrittene Programmieren mit Java ist diese API Doc ein sehr wichtiges Hilfsmittel.

- Nocheinmal:
  - Wenn Sie bisher noch keine Erfahrungen mit Java haben, und nur Bahnhof verstehen: macht nix
  - Aber versuchen Sie sich „reinzubeißen“, denn Programmieren (also einen Algorithmus auch in einer Programmiersprache aufzuschreiben) gehört nun einmal zum Handwerkszeug eines Informatikers
- Zur Überprüfung: Sie sollten jetzt folgendes können
  - ein einfaches, imperatives Java-Programm (z.B. HelloWorld) schreiben,
  - ein Java-Programm in Bytecode übersetzen,
  - ein kompiliertes Java-Programm ausführen,
  - ein Java-Programm dokumentieren.