Abschnitt 2: Einführung

Überblick

Was ist Informatik?

Franz. *informatique* (= information + mathématiques)
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 oder die Sprachen zu seiner Programmierung sondern die *Prinzipien und Techniken zur Verarbeitung von Information.*

Informatik an der Universität

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

Informatik an der Universität

- Was bedeutet Eigenverantwortung?
  - Teilweise ist der Stoff sehr schwierig/theoretisch (noch mehr in den Mathematik-Vorlesungen)
    Wenn Sie der Meinung sind, schon ganz gut programmieren zu können, haben Sie daher nicht notwendigerweise einen Vorteil.
  - 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 und Mathematik

- 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.

Einführung in die Informatik

- In dieser Vorlesung lernen wir u.a.:
  - Einführung in die Programmierung
  - Anhand der Programmiersprache Java[1]
  - Entwerfen von einfachen (hauptsächlich *imperativen*, teilweise *funktionalen*) Algorithmen
  - Einfache Datenstrukturen (Darstellungsmöglichkeiten für Daten)
  - Grundlagen der *objektorientierten* Programmierung
- In späteren Vorlesungen werden wir beispielsweise lernen
  - Andere Paradigmen, z.B. das *funktionale* oder das *logische* Programmieren
  - Software-Engineering, also Software-Entwicklung in Teams
  - Hardware-Grundlagen der Informatik
  - Analysemethoden für Probleme, Algorithmen, Programme
  - uvm.

[1] *ABER: dies ist kein Java-Programmierkurs!!!*

Der Algorithmus-Begriff

Begriff: *Algorithmus*

- Grundlage jeglicher maschineller Informationsverarbeitung
- Zentraler Begriff der Informatik
- Systematische, „schematisch“ („automatisch“, „mechanisch“) ausführbare Verarbeitungsvorschrift

Wichtige Inhalte des Informatikstudiums:

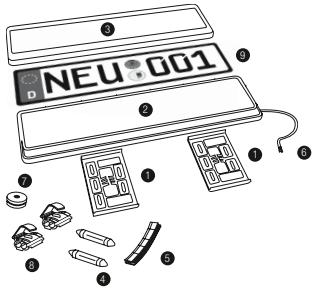
- Entwicklung von Algorithmen
- Analyse von Algorithmen (Korrektheit, Laufzeit, Eigenschaften)
- Oft weniger wichtig: Umsetzung in Programmiersprachen

Unser Alltag ist von Algorithmen geprägt . . .

Algorithmus-Beispiele aus dem Alltag

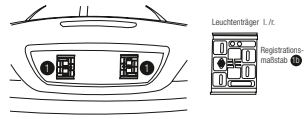
Montageanleitung

Einzelteile



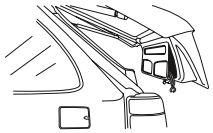
❶ Leuchtenträger
❷ Flächenleuchte
❸ Abdeckrahmen
❹ Lastwiderstände
❺ Klebemasse zur Vibrationsdämpfung
❻ Kabel
❼ Kabeldurchführung
❽ Abzweigklemmen
❾ Nummernschild (nicht in Packung enthalten)

1



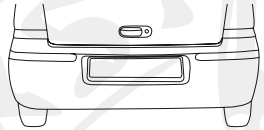
Altes Nummernschild abschrauben und die Leuchtenträger ❶ am Schraublochmuster der Karosserie befestigen. Dazu können Sie die bereits vorhandenen Schrauben benutzen. Bei Installation auf Kunststoffuntergrund empfiehlt sich die Nutzung von 4 Schrauben. Die Langlöcher ermöglichen eine flexible Anpassung, der Registrationsmaßstab ⓗ dient der vertikalen Ausrichtung. **Hinweis:** Halten Sie die genaue Position der Leuchtenträger zur Schraube fest, indem Sie die Markierungen abzählen, und stellen Sie auf beiden Seiten dasselbe Maß ein. Die Leuchtenträger sollten später möglichst nahe der rechten und linken Kante der Flächenleuchte greifen.

2a



Montage bei Kennzeichen in der Heckklappe: Die Innenverkleidung der Heckklappe so weit entfernen, dass Sie von innen an den Kennzeichenbereich gelangen. Die Demontage ist von Fahrzeug zu Fahrzeug unterschiedlich. Fragen Sie im Zweifel einen Fachbetrieb. Bohren Sie jetzt mit einem Metallbohrer (Ø 7 mm) ein Loch für die Kabel unter der vorhandenen Kennzeichenbeleuchtung. Bohrloch entgraten und mit Lackstift gegen Rostbefall schützen. **Hinweis:** Wählen Sie für das Bohrloch eine Stelle hinter dem Kennzeichen, die von innen zugänglich ist und später von der Flächenleuchte verdeckt wird.

2b



Montage bei Kennzeichen im Stoßfänger: Bohren Sie mit einem Metallbohrer (Ø 7 mm) ein Loch für die Kabel unter der vorhandenen Kennzeichenbeleuchtung. Bohrloch entgraten und mit Lackstift gegen Rostbefall schützen. **Hinweis:** Wählen Sie für das Bohrloch eine Stelle hinter dem Kennzeichen, die von innen bzw. unten zugänglich ist und später von der Flächenleuchte verdeckt wird.

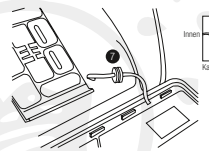
3

Platzieren Sie die umlaufende Randprägung des Nummernschildes ❾ und den Dichtungsring exakt übereinander. Abdeckrahmen ❸ durch **kräftiges Andrücken** auf der Flächenleuchte ❷ befestigen! Der Rahmen muss an 4 Stellen hörbar einrasten. Drehen Sie das Schild um und prüfen Sie, ob alle Rastnasen eingerastet sind. Der Abdeckrahmen ❸ kann zum Kennzeichenwechsel oder zur Reinigung gelöst werden. Hierzu biegen Sie mit einem Schraubendreher an den 4 dafür vorgesehenen Öffnungen den Kennzeichenrahmen nach außen.

4

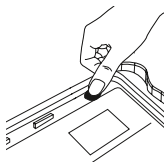
Nehmen Sie die Original-Kennzeichenbeleuchtung außer Betrieb (vorgeschrieben gemäß StVZO), indem Sie beide Leuchtmittel der Kennzeichenbeleuchtung durch die mitgelieferten Lastwiderstände ❹ ersetzen. Sollten bei Ihrem Fahrzeug andere Leuchtmittel eingesetzt werden, nehmen Sie die werkseitig angebrachte Nummernschildbeleuchtung durch Demontage der Leuchtmittel außer Betrieb.

5



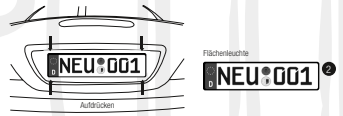
Jetzt die mitgelieferte Kabeldurchführung ❼ mit Kabel montieren und in den Innenbereich führen. Prüfen Sie, ob die Aussparung der Kabeldurchführung fest im Karosserieblech sitzt.

6



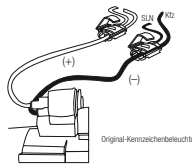
Nehmen Sie nun die Klebemasse zur Vibrationsdämpfung ❺ und bringen Sie diese auf der Rückseite des Kennzeichens an. Die genaue Menge bemessen Sie je nach Abstand zwischen Heckblech und Kennzeichen.

7



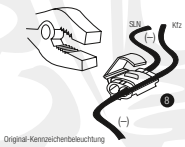
Richten Sie die Flächenleuchte ❷ auf den Leuchtenträgern horizontal aus. Jetzt kräftig aufdrücken (obere Kante zuerst) und **hörbar einrasten** lassen.

8



Verbinden Sie mit den mitgelieferten Abzweigklemmen ❽ die Kabel mit der Spannungsversorgung der ursprünglichen Kennzeichenbeleuchtung: 1 x mit „plus", 1 x mit „minus". Achtung: Nennspannung 12 V Gleichstrom, Nennleistung 2,9 Watt.

9



Pressen Sie beide Abzweigklemmen mit einer Zange fest zusammen. Führen Sie einen Funktionstest in abgedunkelter Umgebung durch. **Hinweis:** Aufgrund gesetzlicher Bestimmungen darf die Flächenleuchte bei Dämmerung und Dunkelheit nicht überstrahlen. Aus diesem Grund werden Sie unter Umständen bei Tageslichtbedingungen den Betrieb der Flächenleuchte nicht ohne weiteres erkennen. Anschließend die Innenverkleidung wieder montieren.

Algorithmus-Beispiele aus dem Alltag

Kochrezept für einen Afrikanischen Hühnertopf (von `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.

Algorithmus-Beispiel aus der Mathematik

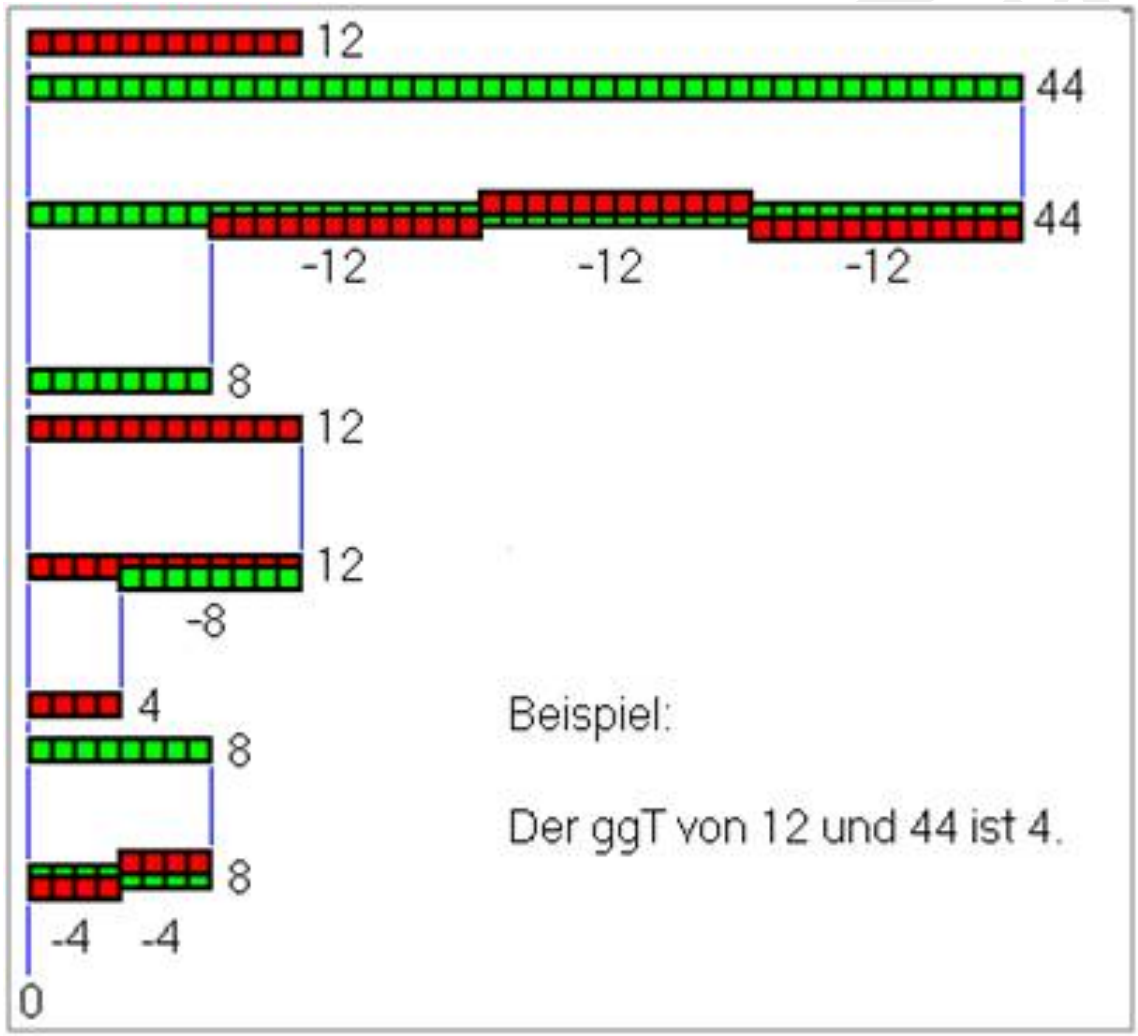
Berechnung des größten gemeinsamen Teilers (GGT) von zwei natürlichen Zahlen a und b
(Beschrieben von Euklid von Alexandria (300 v.Chr.) im Werk *Die Elemente*)

Algorithmus 1 (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

Algorithmus-Beispiel aus der Mathematik

Ablaufbeispiel:



Eigenschaften von Algorithmen

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*?
  - Hört die Berechnung für jedes Paar (a, b) immer auf oder gibt es evtl. eine 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?

Um wirklich sicher zu sein, müssen wir solche Aussagen immer mathematisch *beweisen*.

Definition: Algorithmus

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

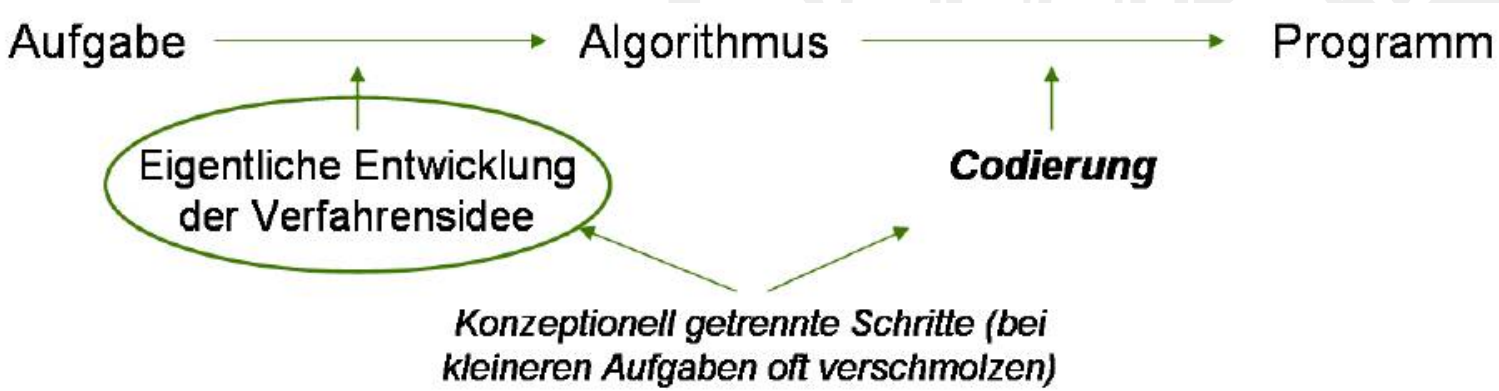
- **Daten**: Die Repräsentation und der Wertebereich von Eingabe und Ergebnissen müssen eindeutig definiert sein.
- **Präzise, endliche Beschreibung**: Die Abfolge von Schritten muss in einem endlichen Text in einer eindeutigen Sprache genau festgelegt sein.
- **Effektiver Verarbeitungsschritt**: Jeder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

Typische Elemente in Algorithmen

- Schrittweises Vorgehen
- Fallunterscheidungen und Wiederholungen („Schleifen“)
  wenn . . . **dann** . . . **sonst** und **solange** . . .
- Zur Berechnung erforderliche Informationen in „Variablen“
- Eingabe: konkrete Werte werden in Eingabeparameter (Parameter der Problemstellung) abstrahiert; dadurch wird eine Klasse von problemen statt einem ganz speziellen Proglem gelöst
  z.B. die beiden Zahlen a und b deren GGT gesucht ist, anstelle nur den GGT von zwei konkreten Zahlen (z.B. 12 und 21) zu berechnen
  z.B. das Wort nach dem im Web gesucht wird
- Ausgabe: Lösung des Problems
  z.B. der GGT
  z.B. Liste der Treffer-Webseiten

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



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

In dieser Vorlesung

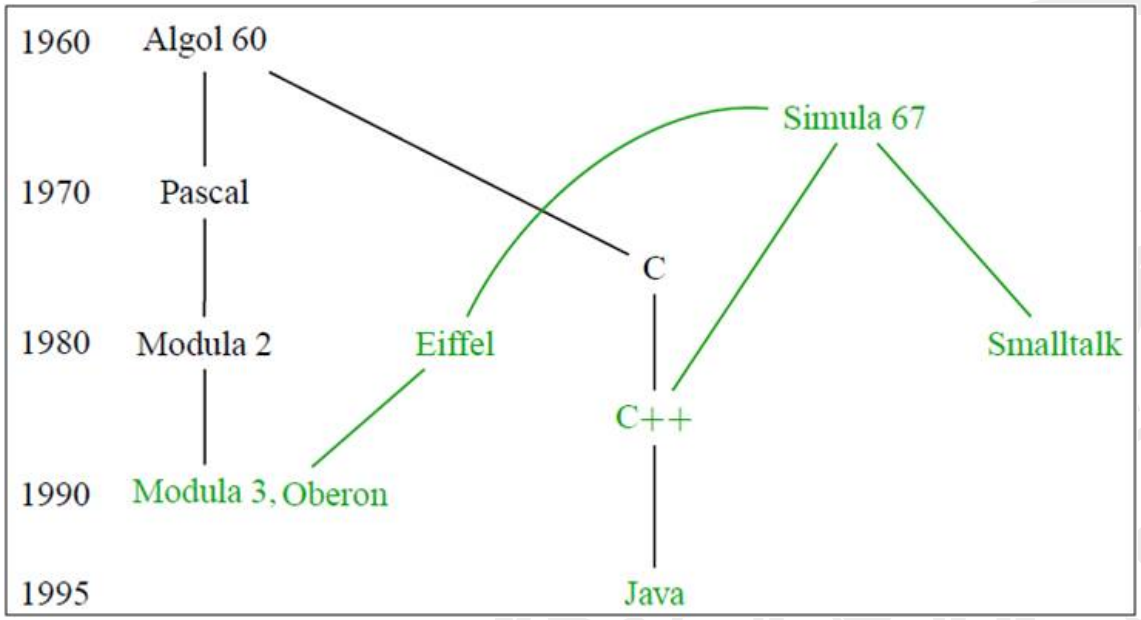
- Konzepte, Methoden und Techniken
  - zur Darstellung und Strukturierung von Daten
  - zur Entwicklung von Algorithmen
- *KEIN* Programmierkurs (*ABER* Anwendung des Erlernten mit Java)

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. maschinengerechte Darstellung von Daten und Algorithmen).
- Praktische Informatik:
  Konstruktion, Darstellung und Ausführung von Algorithmen (*Software*).

Überblick

Die Entstehugs-Geschichte von Java



- Entwickelt von J. Gosling, Bill Joy, P. Naughton, u.a.
- Erste plattformunabhängige, objektorientierte Sprache, insbesondere zur Programmierung von Internet-Applikationen
- Erste Version 1.0 1995, heute Java 9 (beta), bzw. Java 8
- ACHTUNG: wir benutzen die Version Java 7
- Ursprünglicher Name: OAK

Eigenschaften von Java

- Objektorientiert: Klassenkonzept, strenge Typisierung
- Pattformunabhängig: Übersetzung in Virtuelle Maschine (JVM)
- Netzwerkfähig, nebenläufig
- Sicherheitsaspekt in der Entwicklung der Sprache wichtig

Nachteil:

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 statische Eigenschaften aller Objekte dieser Klasse.
  - Attributen (fields, Instanzvariablen): beschreiben den Zustand eines Objekts.
  - statischen Methoden: Funktionen/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 zunächst nur die statischen Elemente. Zu den Aspekten der Objektorientierung kommen wir zu einem späteren Zeitpunkt der Vorlesung.

Ein einfaches imperatives Java-Programm

Keine Angst!!!

All das lernen wir später noch genauer kennen. Für den Anfang merken wir uns: ein einfaches (imperatives) Java-Programm besteht aus der Deklaration nur einer Klasse und einer Methode “main”:

```
public class KlassenName
{
  public static void main(String[] args)
  {
    // Hier geht's los mit
    // Anweisungen (elementare Verarbeitungsschritte)
    // ...
  }
}
```

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

Beispiel: `HelloWorld.java`

Das kanonische Beispiel:

```
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“, beobachten Sie, was passiert, wenn es ausgeführt wird, etc.)

Konventionen

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

- Klassennamen beginnen mit großen Buchstaben, z.B. `HelloWorld`
- Methodennamen, Attributnamen und Variablennamen beginnen mit kleinen Buchstaben, z.B.
  - Methoden: `main`, `println`,
  - Klassenvariable: `out`,
  - Variable: `args` (weder Instanz- noch Klassenvariable, sondern Parametervariable)
- Zusammengesetzte Namen werden zusammengeschrieben, jeder innere Teilname beginnt mit einem großen Buchstaben, z.B. Klassennamen `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, der nur auf bestimmten Rechnerarchitekturen/Betriebssystemen ausgeführt werden kann.
- Andere Sprachen (sog. Skript-Sprachen wie Perl, PHP, auch SML) werden von einem plattformspezifischen Interpreter interpretiert; die Programme sind plattformunabhängig, aber der Sourcecode bleibt unübersetzt und sichtbar, was in vielen Anwendungen nicht erwünscht ist.
- 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.

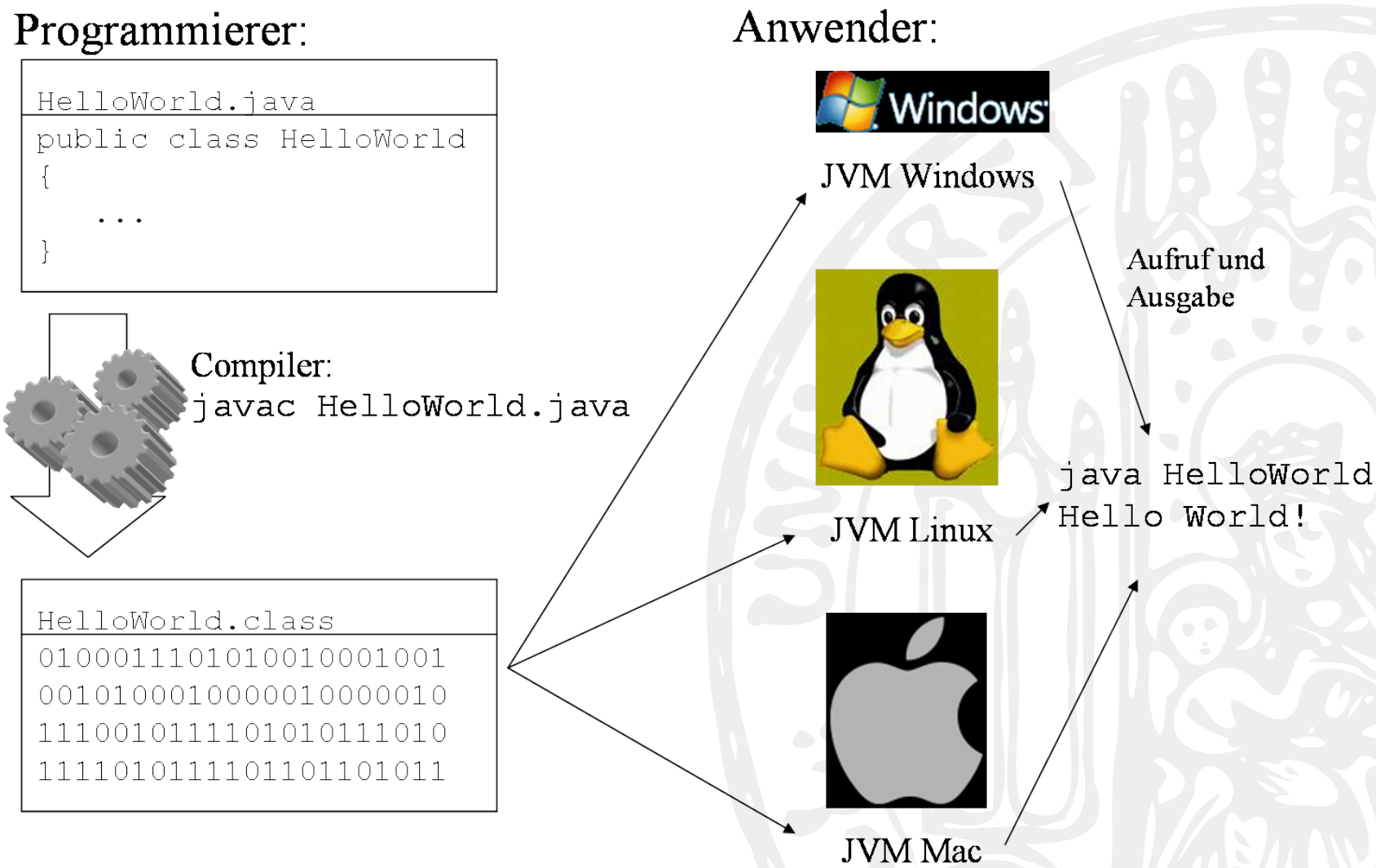
Ausführen des Programms

- 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:

```
java HelloWorld
```

gibt “Hello, World!” auf dem Bildschirm aus.

Übersetzung und Ausführung



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):
Hints on Programming Language Design, 1973

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.

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
     * <code>java HelloWorld</code>
     * 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!");
    }

}
```

Erzeugen der Dokumentation

- 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
- Diese Dokumentation kann auch für viele Klassen gleichzeitig erfolgen (`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

Auch die Bibliothek der Standard Edition, in der viele Algorithmen und Datenstrukturen zur Verfügung gestellt werden (*Java Standard Edition API Dokumentation*), 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.