

Abschnitt 2: Einführung

2. Einführung

2.1 Was ist Informatik?

2.2 Die Programmiersprache Java



Überblick

2. Einführung

2.1 Was ist Informatik?

2.2 Die Programmiersprache Java

**LMU**

Was ist Informatik?

- ▶ 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
- ▶ Theorie vor Praxis
- ▶ Steilerer Anstieg, höheres Niveau
- ▶ Angebote statt Zwang und Anwesenheitspflicht

Zusätzlich: Wesentliches Lernziel an einer Universität für jedes Studienfach

Eigenverantwortung

LMU

Informatik an der Universität

- ▶ Was bedeutet Eigenverantwortung?
 - ▶ Teilweise ist der Stoff sehr schwierig und eher theoretisch abstrakt als praktisch konkret (noch mehr in den Mathematik-Vorlesungen)
Wenn Sie der Meinung sind, schon ganz gut programmieren zu können, haben Sie daher nicht notwendigerweise einen Vorteil, denn es geht nicht darum, wer „der beste Java-Programmierer“ ist.
 - ▶ Im Gegensatz zur Schule ist jetzt aber nicht mehr der Lehrer für Ihren Lernerfolg verantwortlich. (ach nein? wer denn dann????)
 - ▶ Das Erarbeiten des Stoffes ist Ihre *eigene Verantwortung*. (ach so)
- ▶ 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 Java²
 - ▶ 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.

² *ABER: dies ist kein Java-Programmierkurs!!!*

Der Algorithmus-Begriff

Begriff: *Algorithmus*

- ▶ Zentraler Begriff der Informatik
- ▶ Grundlage jeglicher maschineller Informationsverarbeitung
- ▶ 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

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 unzerlegt 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 und in kleine Würfel schneiden, Blüten und Stengelansatz entfernen; alles in den Topf geben.

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

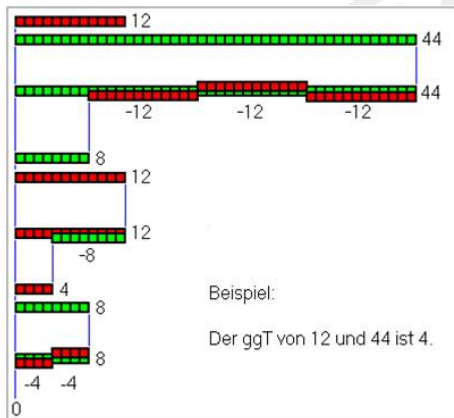
- 1 **solange** $b \neq 0$ **tue das folgende:**
- 2 **wenn** $a > b$ **dann** verändere den Wert von a zu $a - b$ (d.h. $a = b - a$)
- 3 **sonst** verändere den Wert von b zu $b - a$ (d.h. $b = b - a$)

Ausgabe: a

Z.B. ist der GGT von 12 und 44 die Zahl 4.

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?
- ▶ 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 solche Aussagen mathematisch *bewiesen* werden.



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 (durch die verarbeitende Einheit) ausführbar.
- ▶ **Elementarer Verarbeitungsschritt:** Jeder Schritt ist entweder eine Basisoperation (der verarbeitenden Einheit) oder ist selbst durch einen Algorithmus spezifiziert.

Typische Elemente in Algorithmen

Wir haben bereits in den ersten Beispielen ganz typische Elemente in Algorithmen gesehen:

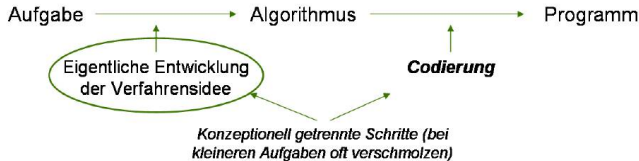
- ▶ Schrittweises Vorgehen bzw. sequentielles Abarbeiten von Verarbeitungsschritten
- ▶ Fallunterscheidungen und Wiederholungen („Schleifen“)
wenn ... dann ... sonst und **solange ...**
- ▶ Zur Berechnung erforderliche Informationen werden in „Variablen“ „zwischengespeichert“

Typische Elemente in Algorithmen

- ▶ Abstraktion von konkreten Eingabewerten:
 - ▶ Statt konkrete Werte werden Eingabeparameter (Parameter der Problemstellung) verwendet
 - ▶ z.B. die beiden Zahlen a und b deren GGT gesucht ist, anstelle nur den Algorithmus für die Berechnung des GGT von zwei konkreten Zahlen (z.B. 12 und 21) anzugeben
 - ▶ Dadurch wird eine *Klasse* von Problemen statt einem ganz speziellen Problem gelöst
 - ▶ z.B. einen Algorithmus zur Suche eines Wortes in Webseiten abstrahiert vom konkret gesuchten Wort; dadurch kann man den Algorithmus für beliebige Wörter benutzen
- ▶ Ausgabe: Lösung des Problems z.B. der GGT

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, die insbesondere elementare Verarbeitungsschritte und eindeutig definierte Datentypen für die Ein-/Ausgabe zur Verfügung stellt



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 Aufgabenstellungen, “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

2. Einführung

2.1 Was ist Informatik?

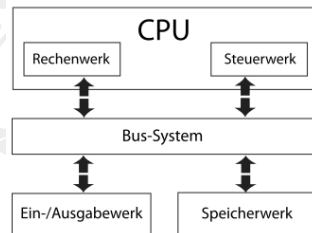
2.2 Die Programmiersprache Java



Aufbau von Computern

- ▶ In der Architektur moderner Rechenanlagen (von-Neumann Modell) ist der Prozessor (CPU) für die „Berechnungen“ zuständig.
- ▶ Die CPU besitzt eine kleine Menge (ca. 50) an *Maschinenbefehlen* (plattformspezifisch), das sind typ. logische und arithmetische Operationen.
- ▶ Die CPU kann Befehle sequentiell ausführen.
- ▶ Befehle und Daten sind *binär* repräsentiert.

„Von Neumann Rechner“



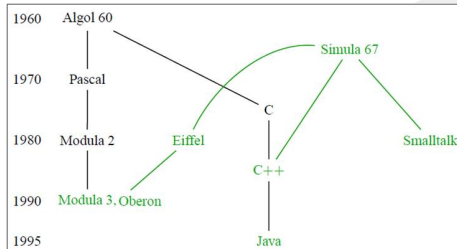
Quelle: Wikipedia



Programmiersprachen

- ▶ Die binären Maschinenbefehle, die in der CPU verarbeitet werden, sind für den Menschen praktisch unlesbar.
- ▶ Durch die Zuordnung von symbolische Namen (sog. *Assembler-Befehle*, z.B. `ADD` für die Addition zweier Werte) für Maschinenbefehle entstand eine erste abstrakte „Programmiersprache“. In dieser wurde GAAAAANZ früher auch programmiert (für komplexe Programme ist das aber absolut unbrauchbar).
- ▶ Mit der Zeit wurden noch höhere Abstraktionen dieser Maschinensprache entwickelt: die höheren Programmiersprachen.
- ▶ Zur Umsetzung eines Programms einer höheren PGS in Maschinenbefehle benötigt man ein „Übersetzungsprogramm“ (z.B. einen *Compiler*).

Die Entstehungsgeschichte 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 8
- ▶ Ursprünglicher Name: OAK



Eigenschaften von Java

- ▶ Objektorientiert: Klassenkonzept, strenge Typisierung
- ▶ Plattformunabhängig: 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

Plattformunabhängigkeit

Ist gar nicht so einfach . . . :

- ▶ Bei vielen Sprachen (z.B. C/C++) erzeugt ein Compiler vor dem Ausführen des Programms Plattform-abhängigen Maschinencode, der nur auf bestimmten Rechnerarchitekturen/Betriebssystemen ausgeführt werden kann (das ist die Abbildung auf eine spezifische Menge an Maschinenbefehlen).
- ▶ Andere Sprachen (sog. Skript-Sprachen wie Python, Perl, PHP, auch SML) werden von einem plattformspezifischen Interpreter (zur Ausführungszeit) interpretiert (in Maschinenbefehle übersetzt); die Programme sind plattformunabhängig, aber der Sourcecode bleibt unübersetzt und sichtbar, was in vielen Anwendungen nicht erwünscht ist.

Plattformunabhängigkeit

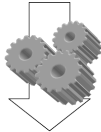
Ist gar nicht so einfach . . . :

- ▶ 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).
 - ▶ Der Bytecode sorgt dafür, dass das Programm nicht mehr (Menschen-)lesbar ist, die JVM übersetzt den Bytecode in die speziellen Maschinenbefehle der CPU.

Übersetzung und Ausführung

Programmierer:

```
HelloWorld.java  
public class HelloWorld  
{  
    ...  
}
```



Compiler:

```
javac HelloWorld.java
```

```
HelloWorld.class  
0100011101010010001001  
0010100010000010000010  
1110010111101010111010  
1111010111101101101011
```

Anwender:



JVM Windows



JVM Linux



JVM Mac

Aufruf und
Ausgabe

```
java HelloWorld  
Hello World!
```

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

Java unterstützt das Kommentieren von Source-Code in einer besonderen Qualität. Tun Sie sich und anderen einen Gefallen und nutzen Sie dieses Potential möglichst vollständig aus.