1. Einleitung

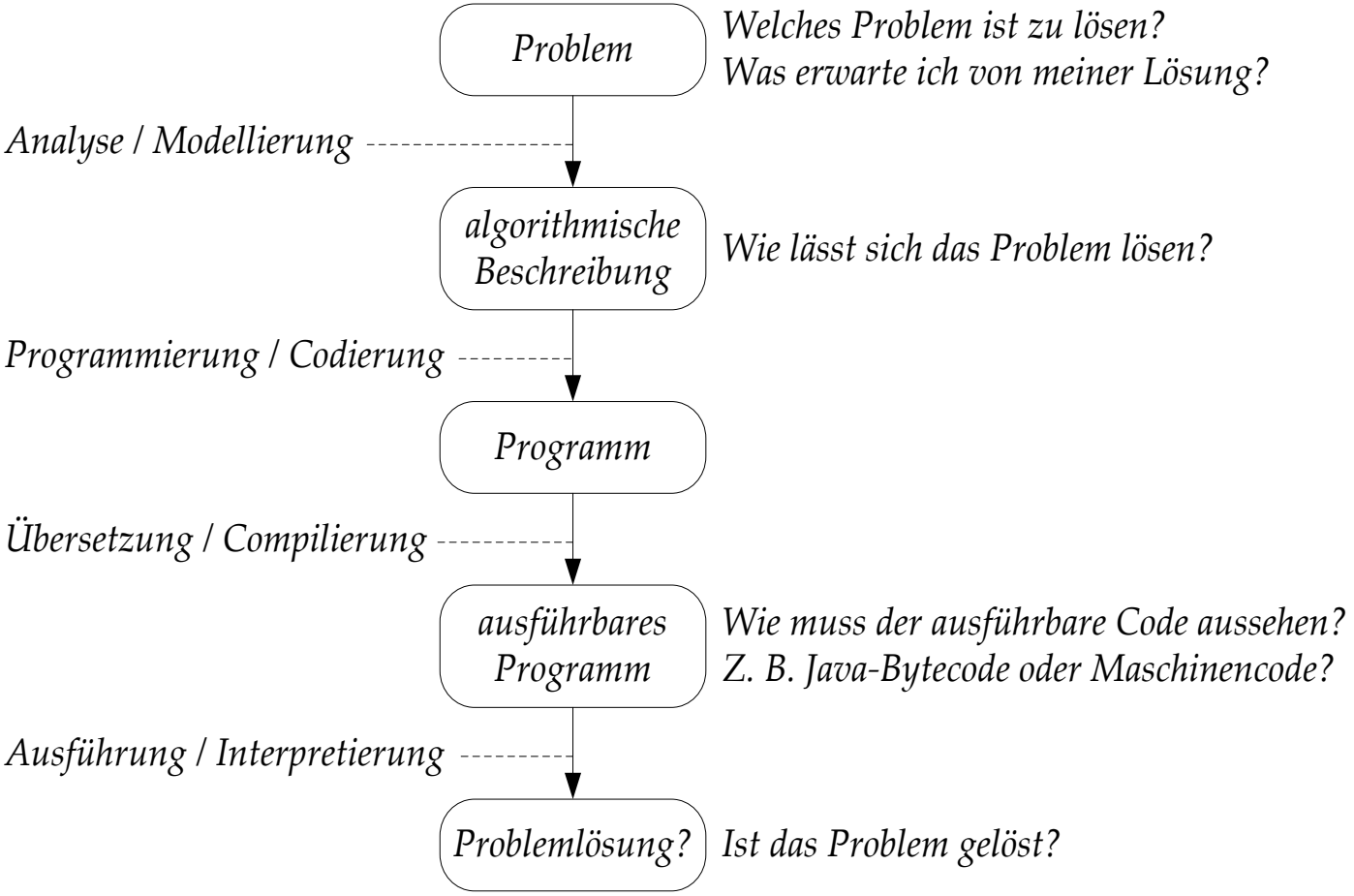
1.1 Was heißt Programmieren?

- Entwicklung eines lauffähigen Computer-Programms mit dem Ziel, ein Problem bzw. eine Menge von ähnlichen Problemen algorithmisch zu lösen
  - **Computer**: Hardware, auf der das Programm laufen soll
  - **Programm**: Software
  - **Algorithmus**: formalisierte Beschreibung eines Problemlösungsverfahrens
- Software-Entwicklung ist mehr als reines "Coden", sondern eine komplexe Prozeßkette mit u.a. folgenden Aufgaben
  - Festlegung der zu lösenden Probleme
  - Modellbildung
  - Implementierung
  - Testen / Verifikation
  - Wartung / Weiterentwicklung

→ Software-Engineering

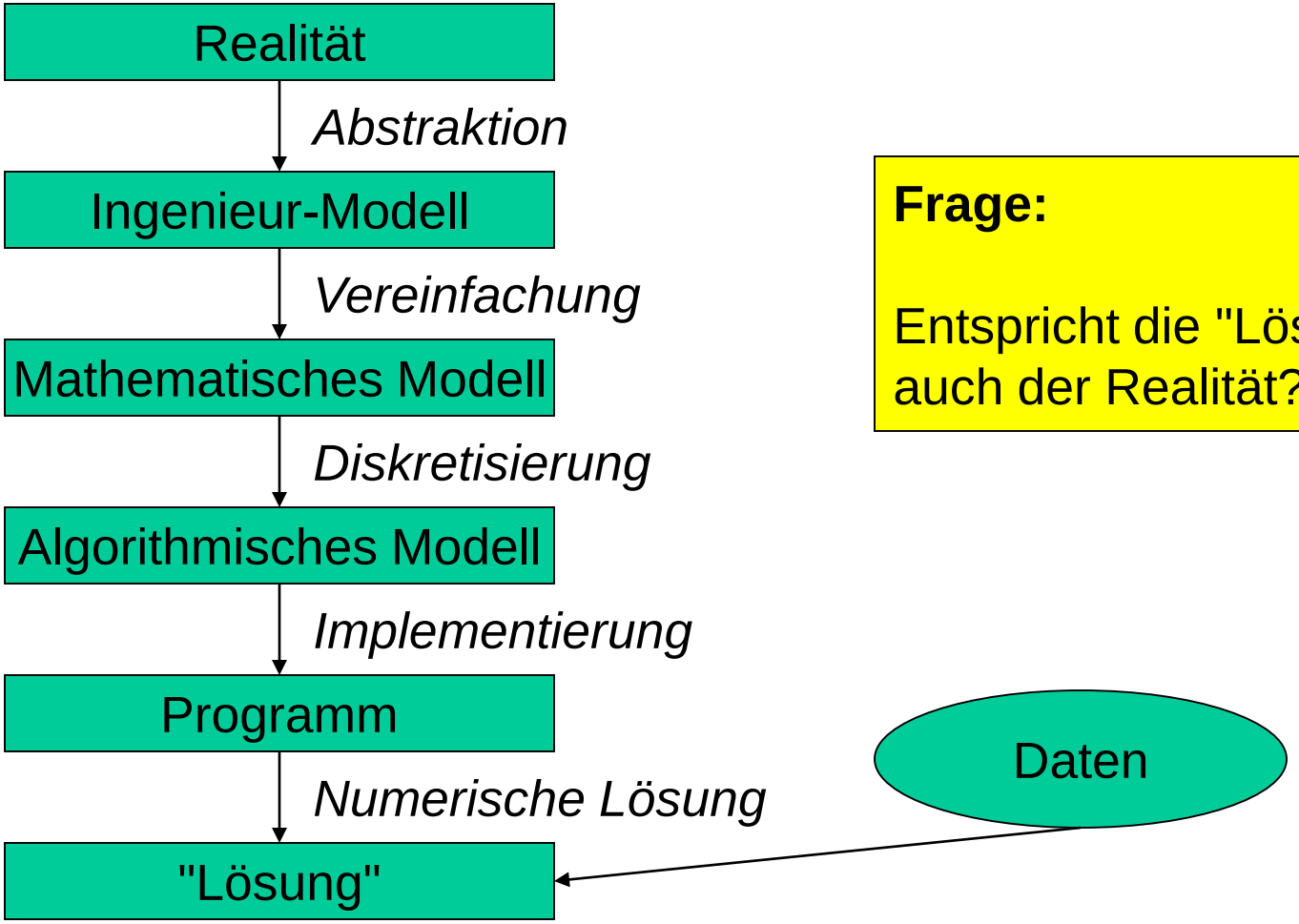
1.1 Was heißt Programmieren?

- Umsetzen eines gegebenen Algorithmus in ein lauffähiges Computer-Programm



Was heißt Programmieren?

- Tatsächlich: Wesentlich längere Kette!



Frage:

Entspricht die "Lösung" auch der Realität?

Was heißt Programmieren?

Frage: Entspricht die "Lösung" auch der Realität?

- Versuch der Beantwortung durch
  - Modellverifikation: Überprüfen des Modells durch Experimente (Simulationen)
  - Programmverifikation: Prüfen der Korrektheit des Programms (Spezialgebiet der Informatik)
  - Ergebnisverifikation: Beweisen, dass numerische Ergebnisse auch Lösungen des algorithmischen Modells sind (Spezialgebiet der Mathematik)
- Frage: Sind alle drei Arten der Verifikation notwendig?

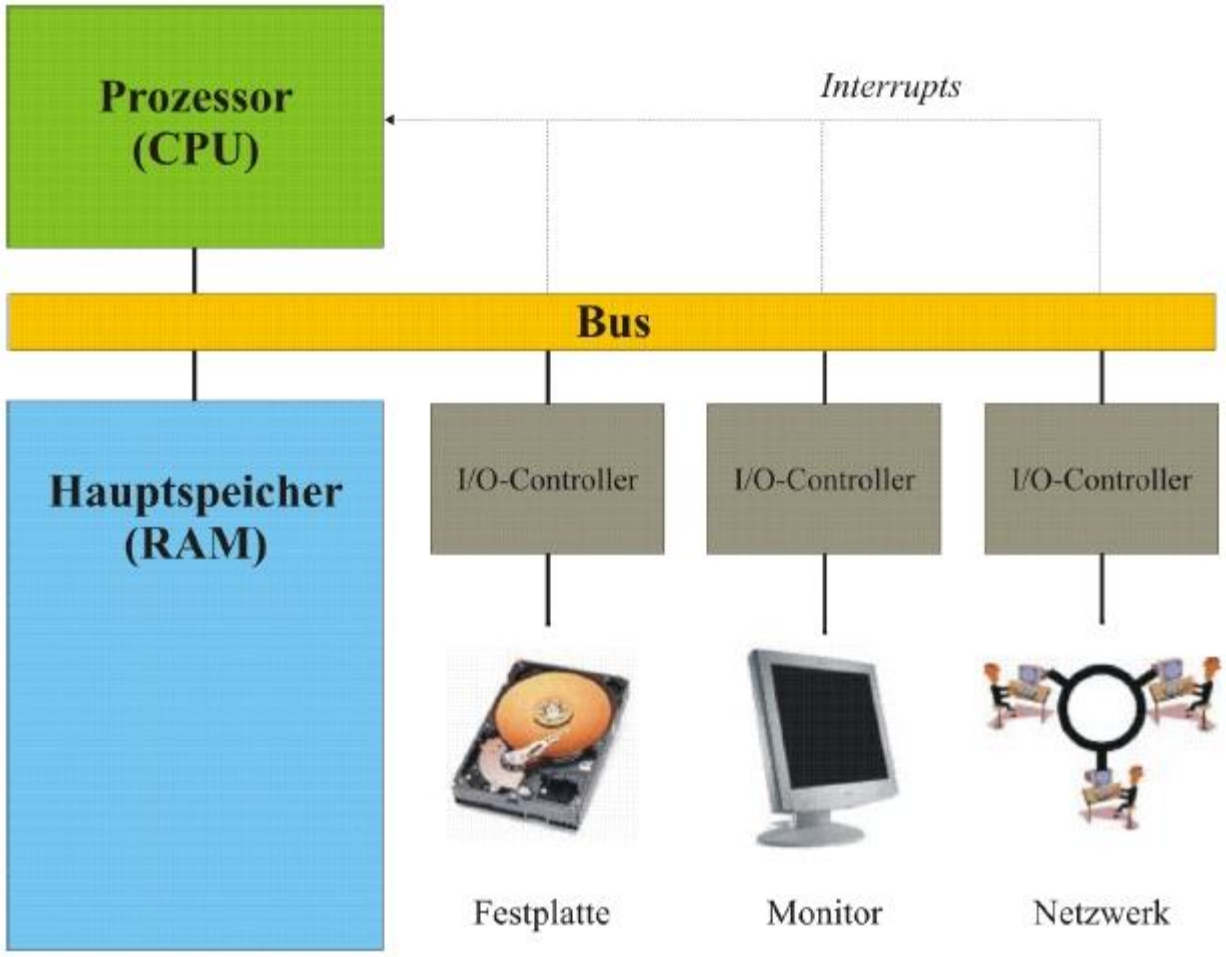
TOP 3

Rank	System	Cores	Rmax (PFlop/s)	Rpeak (PFlop/s)	Power (kW)
1	**Frontier** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE DOE/SC/Oak Ridge National Laboratory United States	8,730,112	1,102.00	1,685.65	21,100
2	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,630,848	442.01	537.21	29,899
3	**LUMI** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE EuroHPC/CSC Finland	1,110,144	151.90	214.35	2,942

(Stand: Juni 2022)

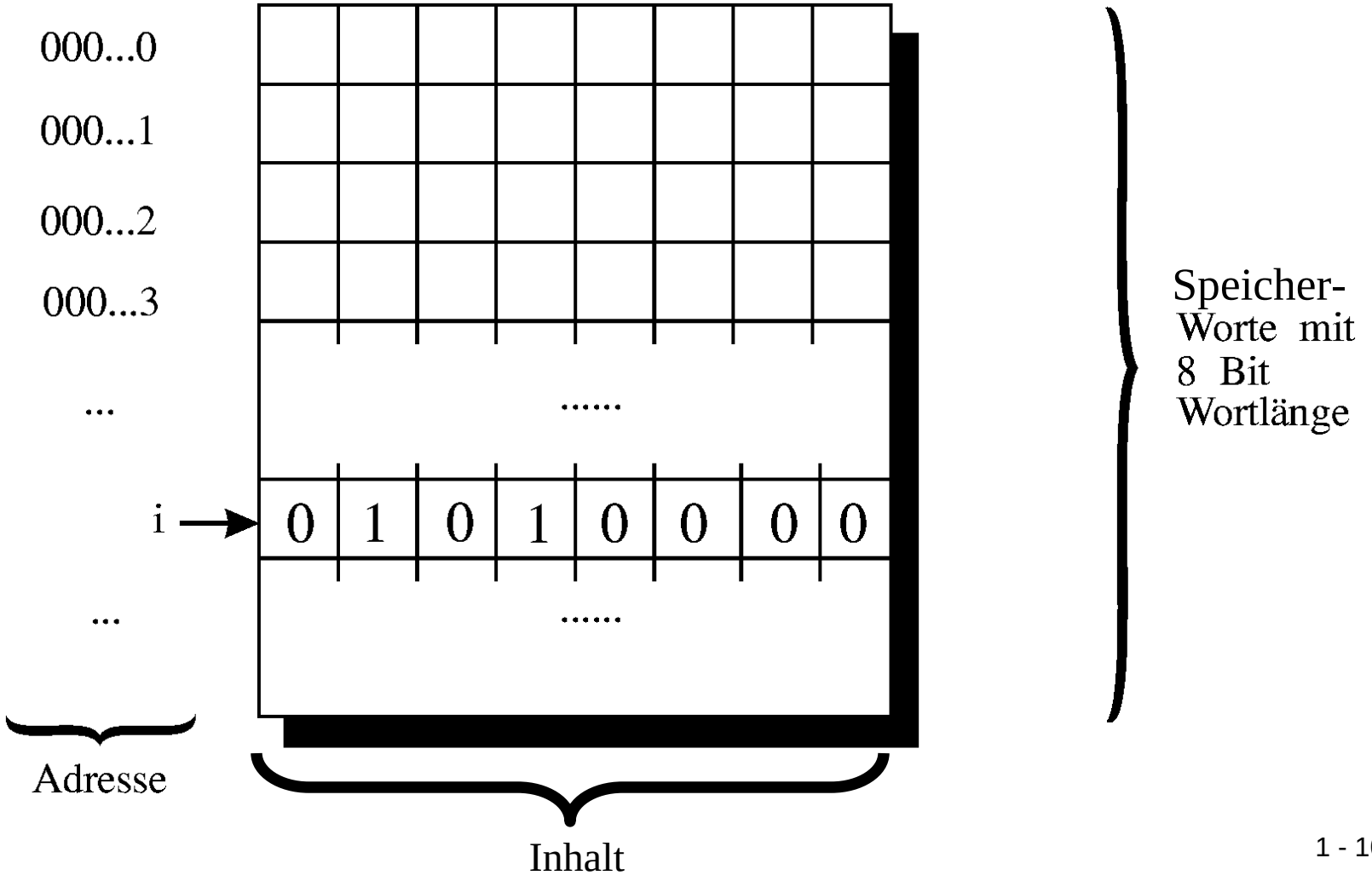
Hardware

- Minimale „schematische“ Vorstellung für einen Universalrechner



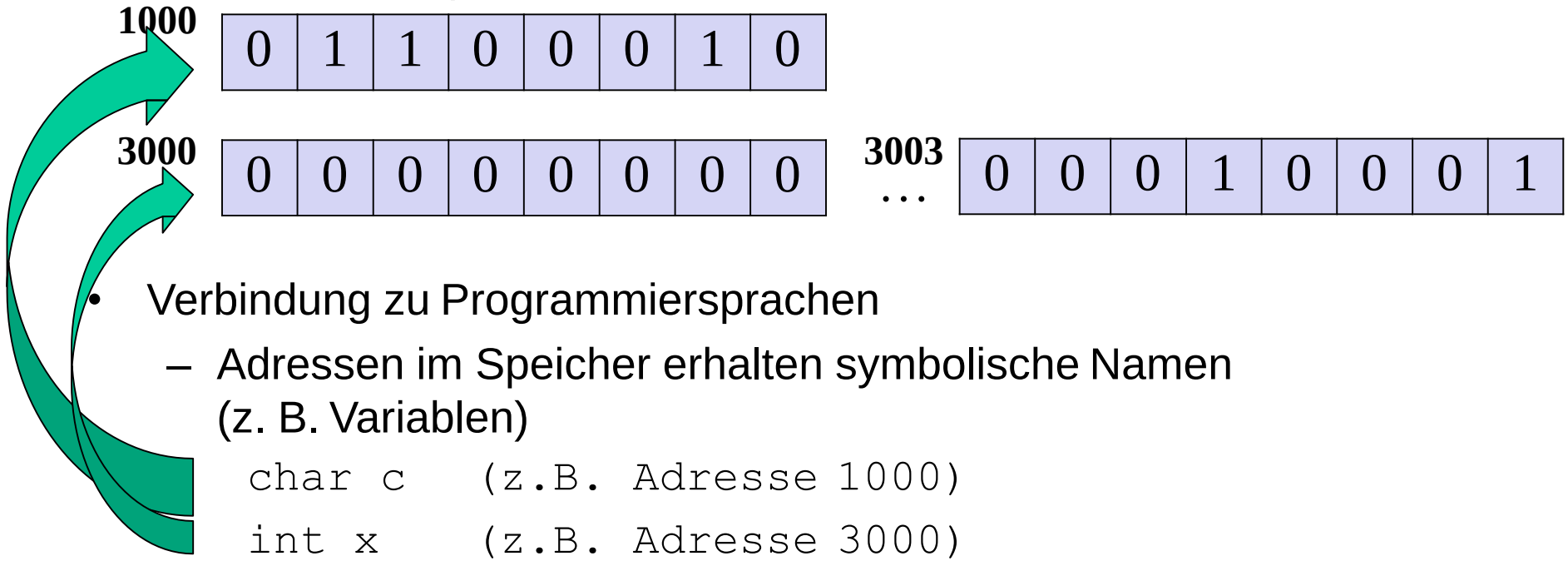
Hardware

- Minimale „schematische“ Vorstellung des Hauptspeichers (RAM)



Hardware

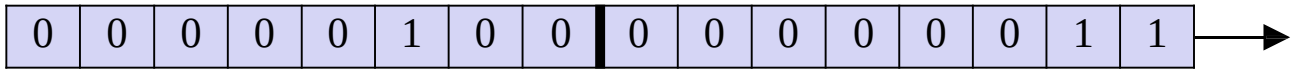
- Datenspeicherung im Arbeitsspeicher
  - einzelne Daten können sich (abhängig von ihrem Typ) aus mehreren Speicherworten zusammensetzen



- Verbindung zu Programmiersprachen
  - Adressen im Speicher erhalten symbolische Namen (z. B. Variablen)

```
char c   (z.B. Adresse 1000)
int x    (z.B. Adresse 3000)
```

  - Inhalte mehrerer Speicherworte werden kombiniert und ergeben bestimmte Werte (z. B. Werte von Variablen), Codierung je nach Typ

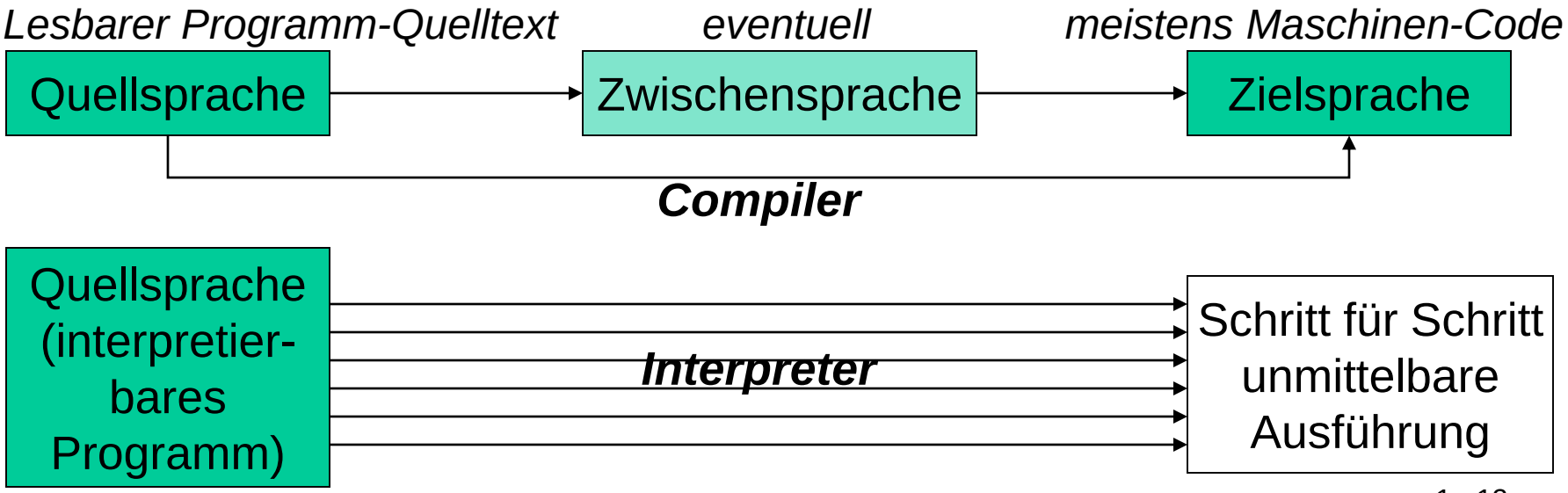


1.2.2 Software

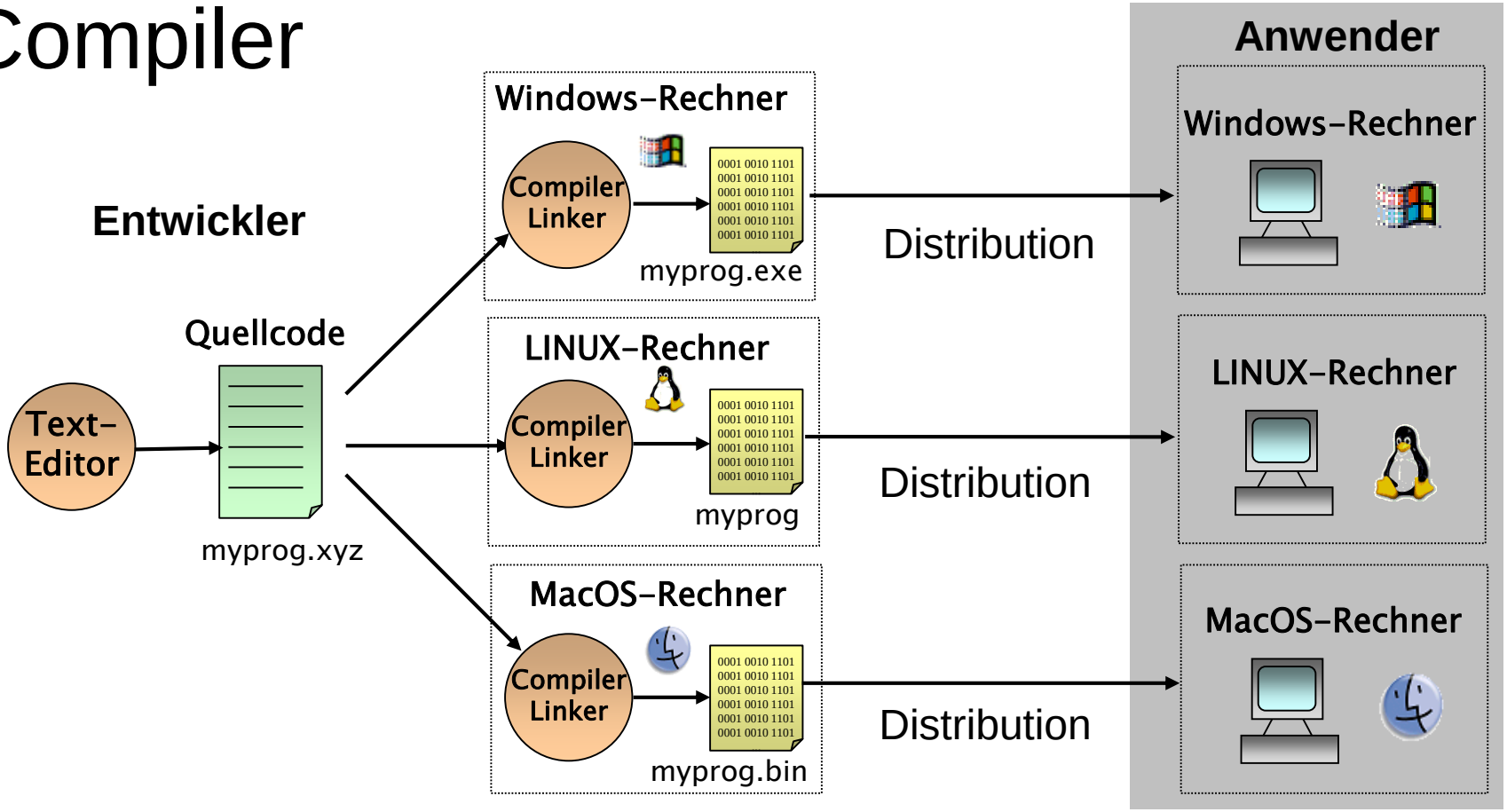
1.2.2.1 Mögliche Klassifizierung

- Betriebssysteme (DOS, OS/2, Windows, UNIX, LINUX, MacOS)
- Dienstprogramme (Editor, Datei-Manager, Mailer, Browser)
- Übersetzer (Assembler, Compiler, Interpreter)
- Anwendungsprogramme (mit Compiler oder Interpreter erzeugt)

1.2.2.2 Compiler und Interpreter

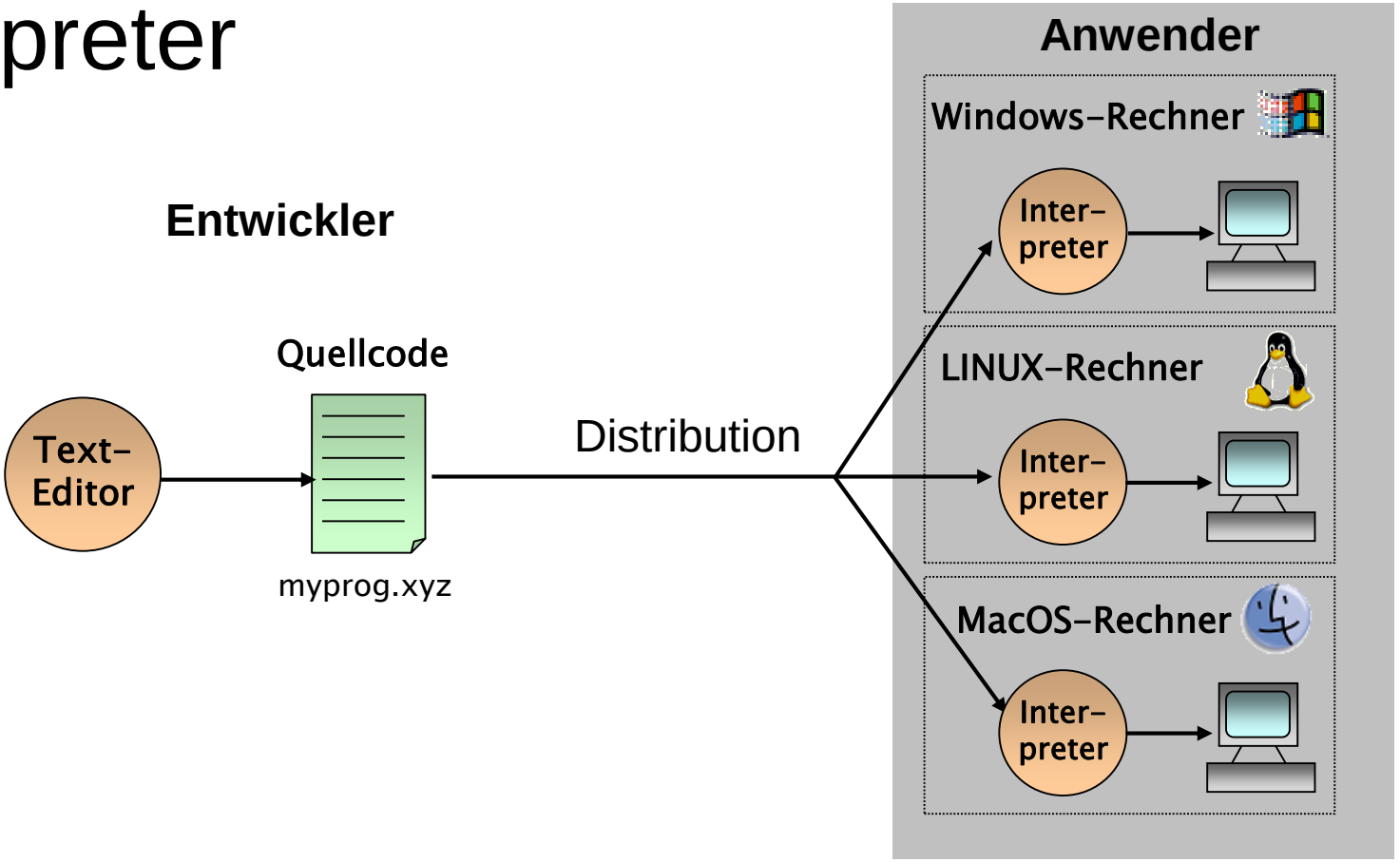


Compiler



- Vorteile: Optimale Nutzung der Prozessoreigenschaften, hohe Abarbeitungsgeschwindigkeit, Quellcode geschützt
- Nachteile: Programme sind spezialisiert auf Zielrechner, zusätzlicher Aufwand für Anpassungen (z. B. grafische Oberflächen), verschiedene Compiler notwendig

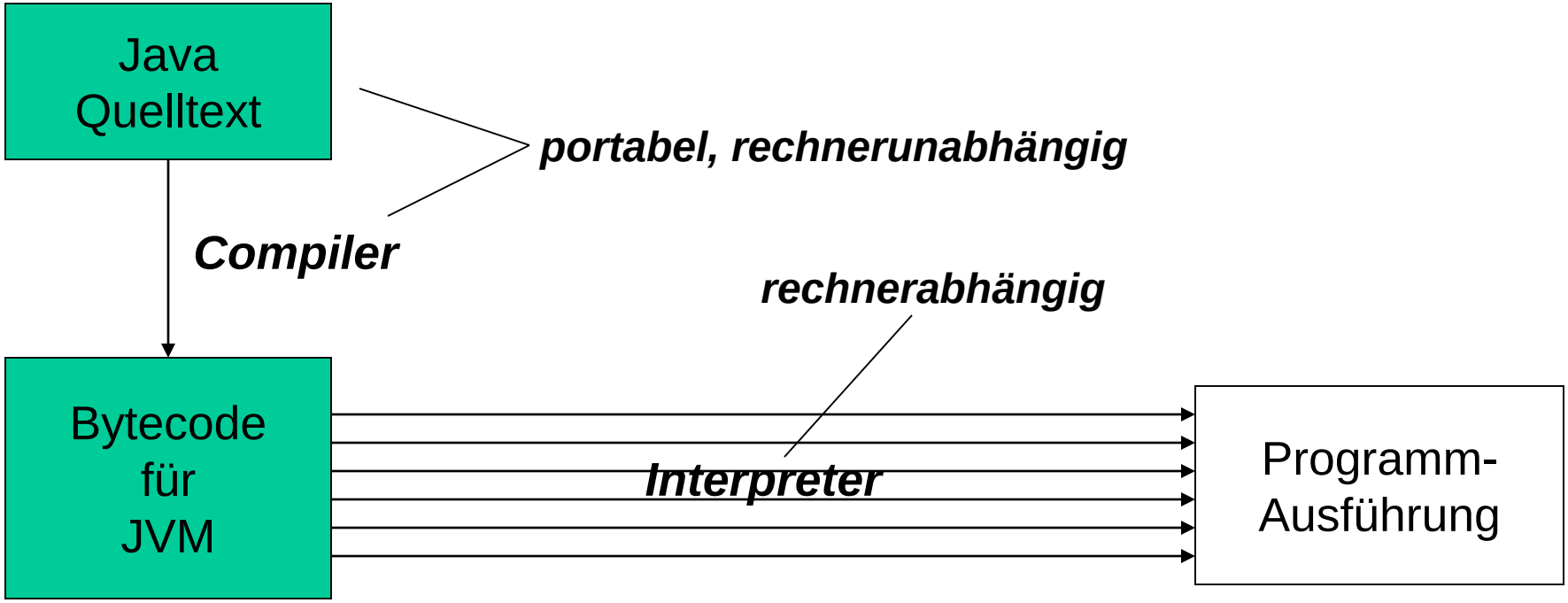
Interpreter



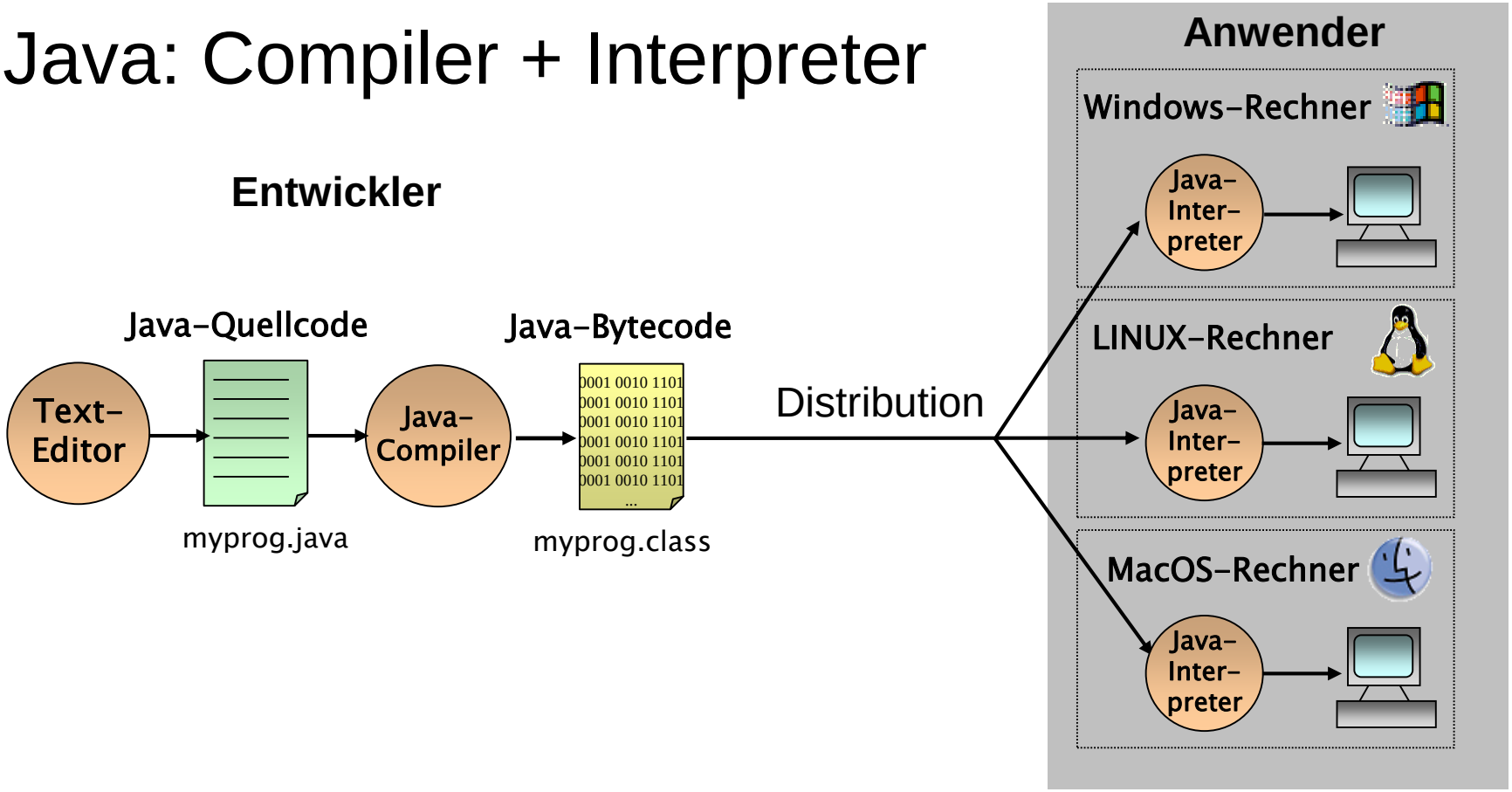
- Vorteile: Quellcode direkt ausführbar, falls Interpreter verfügbar
- Nachteile: langsame Ausführung, verschiedene Interpreter notwendig, Quellcode lesbar

1.2.2.1 Java

- Kombination von Compiler und Interpreter für Code einer virtuellen Maschine (Java Virtual Machine - JVM)

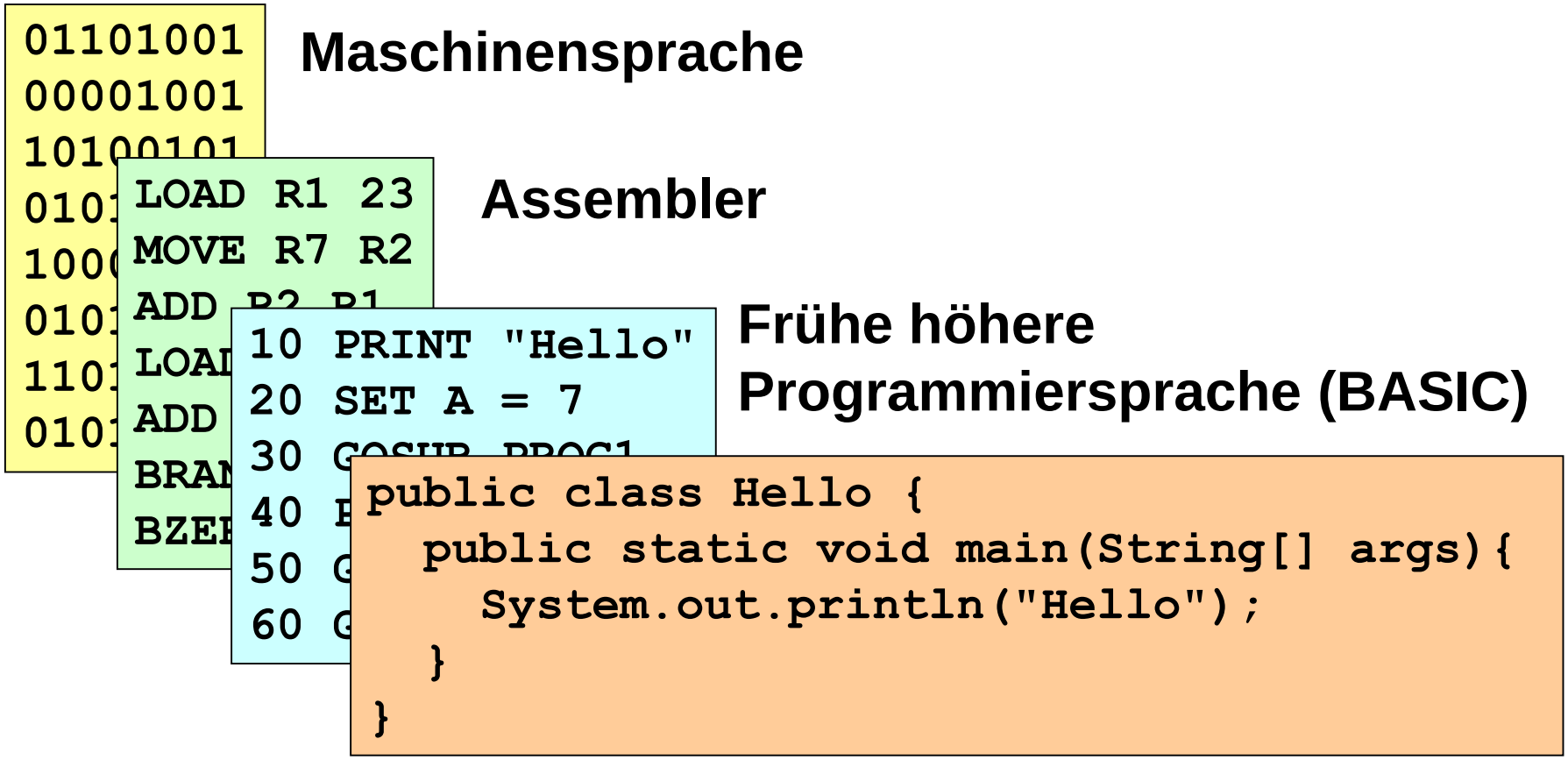


Java: Compiler + Interpreter



- Vorteile: nur ein Compiler, Sprache plattformunabhängig, Quellcode teilweise geschützt
- Nachteile: langsame Ausführung, verschiedene Interpreter notwendig

1.2.2.2 Ein klein wenig Geschichte



Maschinensprache

Assembler

Frühe höhere Programmiersprache (BASIC)

Moderne höhere Programmiersprache (Java)

1.2.2.3 Literatur

Die Vorlesung basiert auf:

Ratz, Scheffler, Seese, Wiesenberger
Grundkurs Programmieren in Java
6. Auflage, 2011, Hanser-Verlag

Literatur

- Guido Krüger, Thomas Stark:
  Handbuch der Java-Programmierung,
  Addison Wesley

- Sharon Zakhour, Scott Hommel, Jacob Royal:
  Das Java Tutorial. Eine Einführung in die Grundlagen
  Addison Wesley

- Kathy Sierra, Bert Bates, Lars Schulten, Elke Buchholz:
  Java von Kopf bis Fuß,
  O'Reilly

Literatur

- Christian Ullenboom:
  Java ist auch eine Insel,
  Galileo Computing

- …

1.3 Algorithmen und ihre Darstellung

- *Programm* und *Algorithmus* sind eng verwandte Begriffe
- **Algorithmus**
  - Vorschrift zur Bearbeitung (Lösung?) eines Problems
  - geforderte **Eigenschaften**
    - *endlich* beschrieben
    - *eindeutig* im Sinne
      - aus Operationen mit *eindeutiger* Wirkung zusammengesetzt
      - Reihenfolge der Operationen ist *eindeutig* bestimmt
    - *abbrechend* (terminierend)
      - ist nach *endlich* vielen Schritten beendet
  - geforderte **Bestandteile**
    - zu bearbeitende "Objekte" (Größen, Werte, Gegenstände, ...)
    - Angaben über Anfangs- und Endzustände
    - auszuführende Anweisungen

Algorithmen und ihre Darstellung

- Beispiele
  - Kochrezept = Algorithmus ≠ Programm (Koch = Prozessor? ☺)
  - Bauanleitung

- **Programm**
  - Reihe von Anweisungen (an den Prozessor), die (im Prinzip) sequentiell abgearbeitet werden

- Darstellungsformen für Algorithmen
  - als Text (verbal, Pseudo-Code bzw. -Sprache)
  - als Flussdiagramm (Programm-Ablauf-Plan, PAP)
  - als Struktogramm (Nassi-Shneiderman-Diagramm)

Algorithmen und ihre Darstellung

- Beispiel für einen "Nicht-Algorithmus"

  Wiederhole die Schritte 1, 2 und 3 so lange, bis du eine Million Euro gewonnen hast:

  1. Fülle einen Lottoschein aus.
  2. Gib den Schein bei der Annahmestelle ab.
  3. Warte auf die nächste Ziehung und kassiere deinen Gewinn.

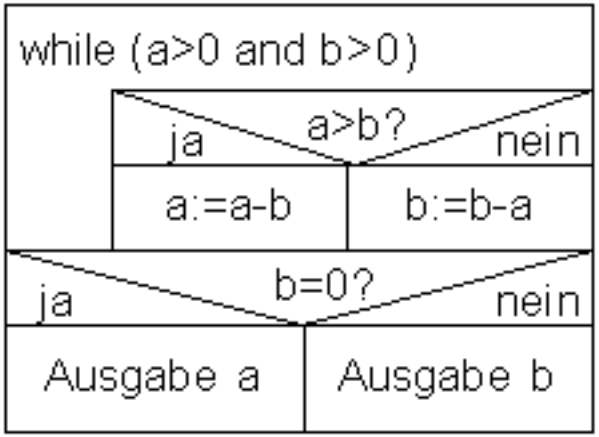
  Terminierung höchst unwahrscheinlich! ;-)

Algorithmen und ihre Darstellung

- Beispiel: **Euklidischer Algorithmus** Berechnung des größten gemeinsamen Teilers zweier natürlicher Zahlen *a* und *b*
  - als Text:

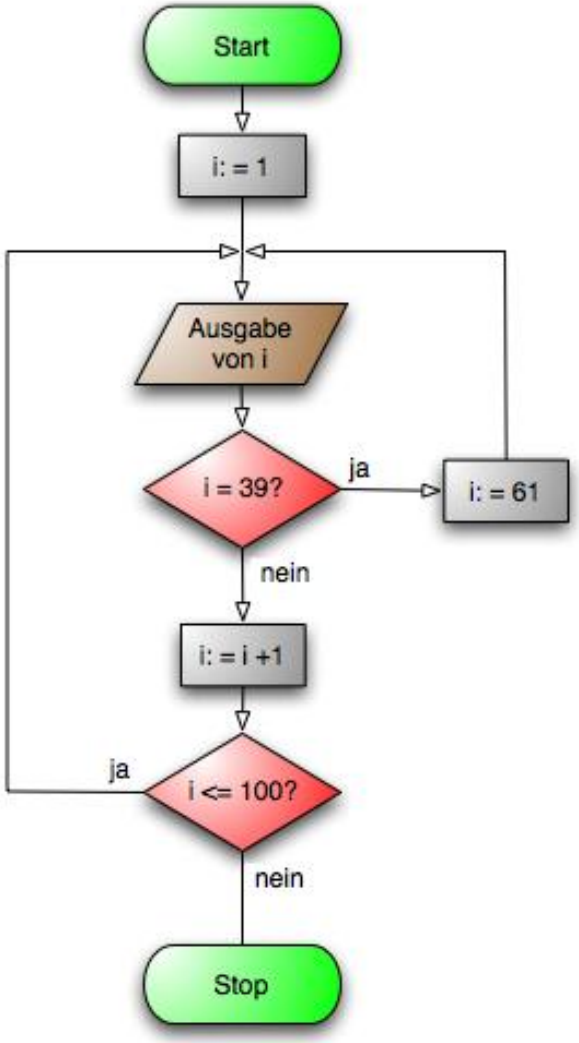
solange $b \neq 0$
  falls $a > b$
    $a := a - b$
  andernfalls
    $b := b - a$
gib *a* zurück

  - als Struktogramm (Nassi-Shneiderman-Diagramm):



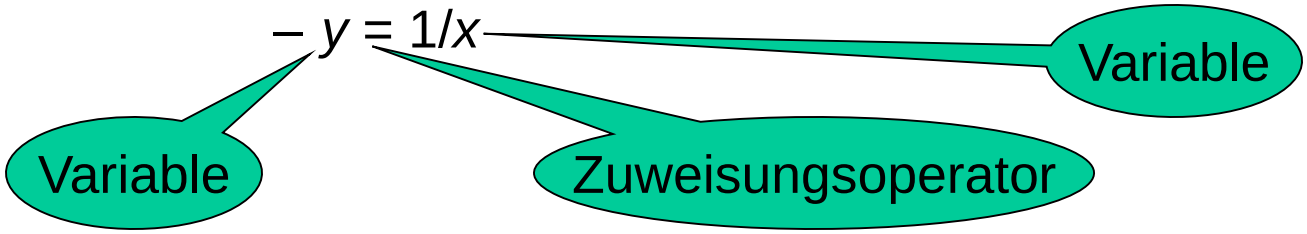
Algorithmen und ihre Darstellung

- Beispiel
  Flussdiagramm
  (Programm-Ablauf-Plan)
  - genormte Elemente



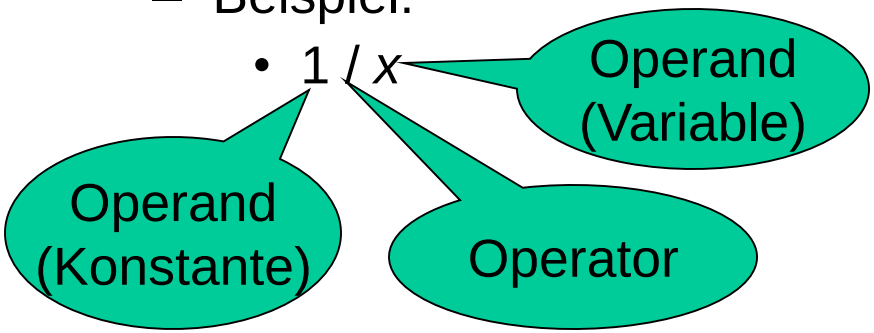
1.4 Grundbegriffe der Programmierung

- **Variable**
  - "Platzhalter„ für einen Wert (Analogie: „Gefäß“)
  - Beispiel:
    - Algorithmus A: "Kehrwert von 4"
      - dividiere 1 durch 4
    - Algorithmus B: "Kehrwert einer Zahl"
      - setze *x* auf den Wert der Zahl
      - dividiere 1 durch *x*
- **Zuweisung (Wertzuweisung)**
  - weist einer Variable einen Wert zu (Analogie: „Gefäß füllen“)
  - Beispiel:
    - Algorithmus B': "Kehrwert einer Zahl"
      - *x* = Wert einer Zahl
      - *y* = 1/*x*



Grundbegriffe der Programmierung

- **Ausdruck**
  - Kombination von **Operanden** und **Operatoren** als "Vorschrift" zur Berechnung eines **Werts**
  - liefert immer einen Wert (Ergebniswert) ab
  - Beispiel:
    - 1 / *x*



- **Anweisung**
  - Kombination von Ausdrücken und Methoden als "Vorschrift" zur Ausführung einer Aktion
  - Beispiele:
    - *x* = 5 Wertzuweisung
    - *y* = 1 / *x* Wertzuweisung
    - print(*y*) Ausgabeanweisung (Methodenaufruf "Drucke *y*")

Grundbegriffe der Programmierung

- **Anweisungs-Sequenz**
  - mehrere Anweisungen, die nacheinander ausgeführt werden

- **Anweisungs-Block**
  - logisch zusammengefasste Anweisungen bzw. Programmteile, die als ***eine*** Anweisung aufgefasst werden können

- **Bedingte Anweisung** (**Entscheidungsanweisung**)
  - Anweisung mit mehreren Alternativen

- **Wiederholungsanweisung** (**Schleife**)
  - mehrfach ausgeführter Anweisungsteil (Block)

Grundbegriffe der Programmierung

- **Datentyp** (**Typ**)
  - Bauplan für Daten (Variablen- oder Konstanten-Werte), der festlegt,
    - wie die Darstellung der Werte im Speicher erfolgt,
    - welche Operationen für die Werte erlaubt sind,
    - welche Standardwerte (Default-Werte) festgelegt sind.
  - Beispiele (Variablen im Speicher):