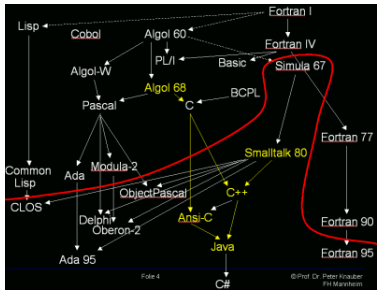
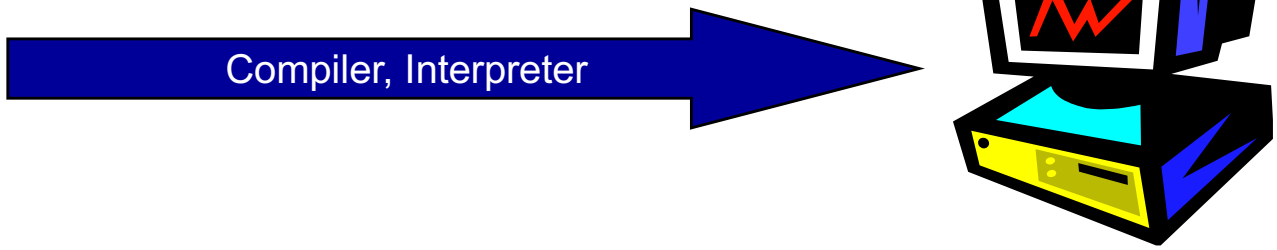
Java, Compiler und Interpreter





- Höhere Programmiersprachen
- Ausführung von Compiler-Sprachen
- Ausführung von Interpreter-Sprachen
- Vor- und Nachteile von Compilern und Interpretern
- Ausführung von Java-Programmen
- Sprachelemente von Java

Java-Historie

- Ursprünglicher (1990/91) Name: Oak
  (der Name war jedoch schon vergeben)
- Entwickler: Gosling, Joy, Steele (Sun Microsystems / Oracle)
- Geplantes Einsatzgebiet: Steuerung elektronischer Konsumgeräte
  - Kleine, portable Geräte, verteilte Systeme
  - Echtzeitanforderungen
  - -> Verteilungs-Unterstützung: sehr hilfreich für die Anwendung im Internet
- Ähnlichkeiten: die Syntax stammt weitgehend von C++, aber
  - Java hat ein sicheres Typsystem
  - Java bietet Möglichkeiten zur Modularisierung (packages)
  - Java verzichtet auf:
    - → Mehrfachvererbung
    - → Struktur- und Vereinigungstypen
    - → Zeigertypen

Programmiersprachen

- Klassisch:
  Die Menge der verfügbaren Instruktionen ist *abhängig* vom Prozessor

- *Unabhängig* vom Prozessor sind **höhere** (problemorientierte) **Programmiersprachen** (wie Java)

  Folge:
  Programme, die in höheren Programmiersprachen geschrieben sind, können auf Rechnern mit verschiedenen Prozessoren ausgeführt werden

Ausführung höherer Programmiersprachen

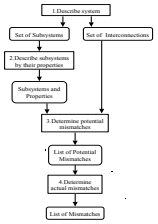
Es gibt zwei Gruppen (im Hinblick auf die Art der Ausführung) höherer Programmiersprachen:

- Compiler-Sprachen:
  Die höhere Sprache wird durch ein besonderes Programm, einen sogenannten **Compiler**, in Maschinensprache übersetzt

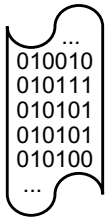
- Interpreter-Sprachen:
  Die höhere Sprache wird von einem besonderen Programm, einem sogenannten **Interpreter**, bei der Ausführung interpretiert

Ausführung von Compiler-Sprachen

Programm
(höhere
Programmiersprache)



Programm
(Maschinensprache),
z.B. exe-Datei

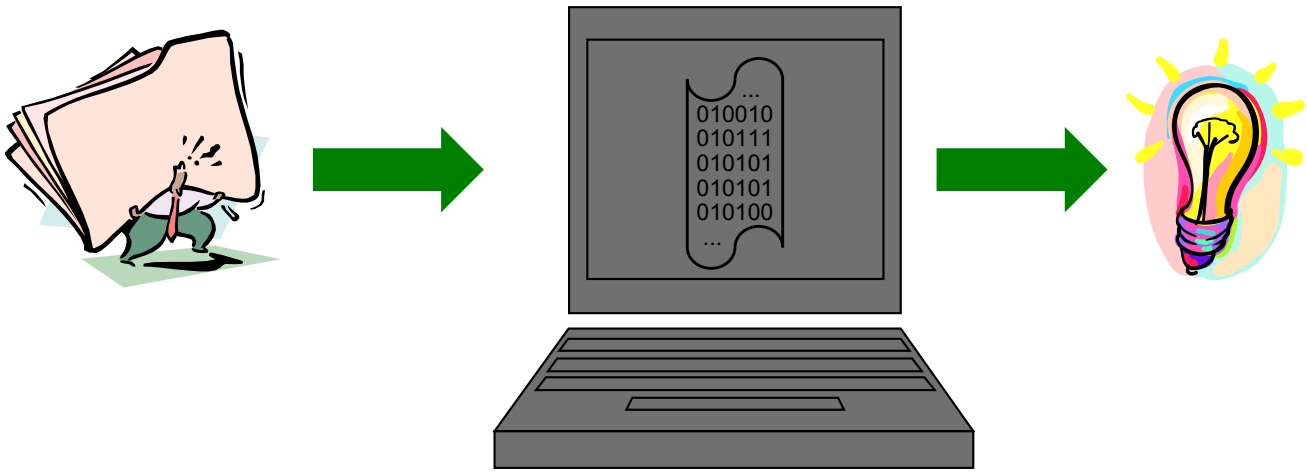


Ein „Compiler“ ist auch ein Programm!

Ausführung von Compiler-Sprachen

Maschinensprache ist nur vom Prozessor einer bestimmten „Maschine“ ausführbar: Das ist evtl. eine deutliche Einschränkung!

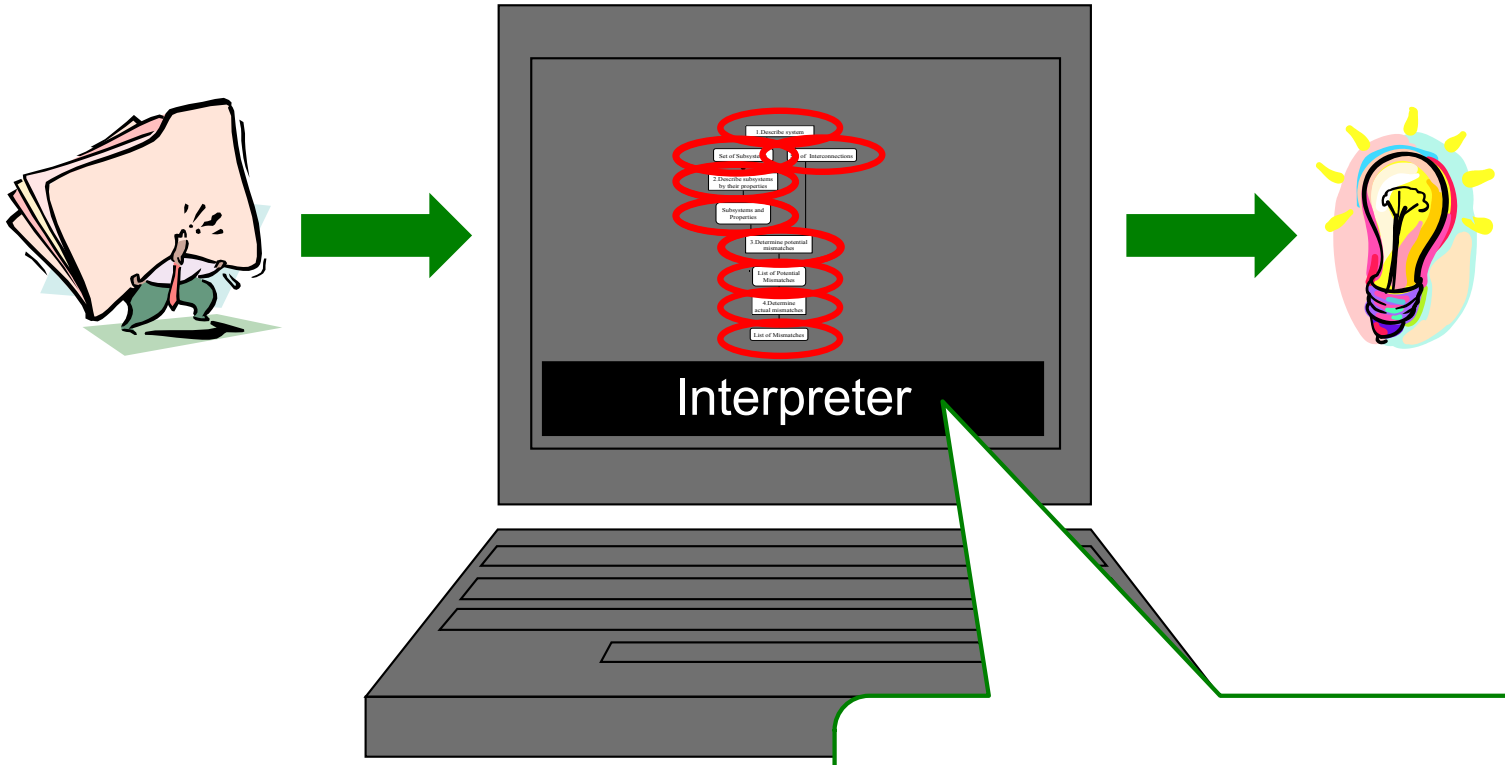
Programm (Maschinensprache), z.B. exe-Datei



Ausführung von Interpreter-Sprachen

Ein interpretiertes Programm läuft auf allen „Maschinen“, auf denen es einen Interpreter für diese höhere Programmiersprache gibt

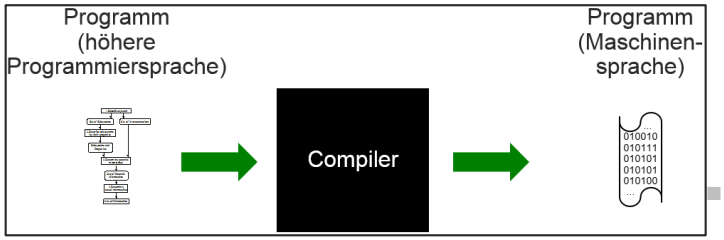
Programm (höhere Programmiersprache)

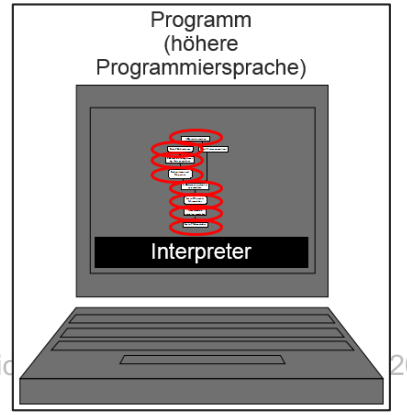


Ein „Interpreter“ ist auch ein Programm!

Vor- und Nachteile von Compilern und Interpretern 1/2

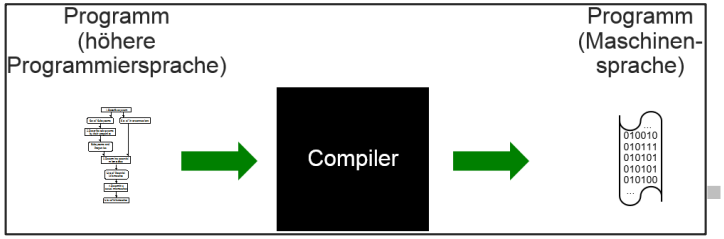
- Compiler
  - Compiler können das Programm bei der Übersetzung auf *Syntax-* und *Semantik-Fehler* überprüfen, also z. B. auf Schreibfehler und Widersprüche gegen die Sprachdefinition
  - Anweisungen werden genau einmal (beim Übersetzen) geprüft und beim Ablauf direkt vom Prozessor ausgeführt
- Interpreter
  - Semantik-Fehler werden eventuell erst bei der Programmausführung bemerkt, ein Absturz kann die Folge sein
  - Anweisungen werden unnötigerweise bei jedem Durchlauf erneut geprüft

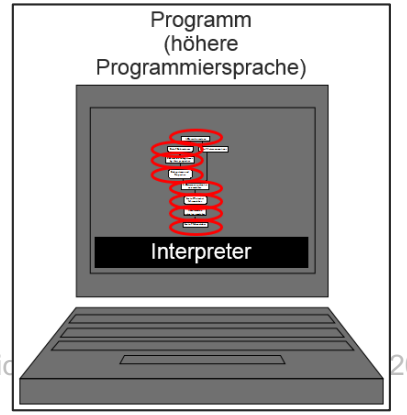




Vor- und Nachteile von Compilern und Interpretern 2/2

- Compiler
  - Treten zur Laufzeit Fehler auf (z.B. eine Division durch 0), sind diese meist schwer zu lokalisieren
  - Compiler erzeugen Code, der nur auf einem speziellen Prozessor läuft; Unterschiede z.B. in der Darstellung von Datentypen kann zu unterschiedlichen Ergebnissen des gleichen Programms führen
- Interpreter
  - \+ Interpreter können genauere Fehlermeldungen geben als Compiler, Debugging ist möglich (Informationen zum Quelltext liegen noch vor!)
  - \+ Interpreter abstrahieren vom zugrundeliegenden Prozessor, Programme können auf jedem Rechner identische Ergebnisse liefern





Ausführung von Java-Programmen 1/2

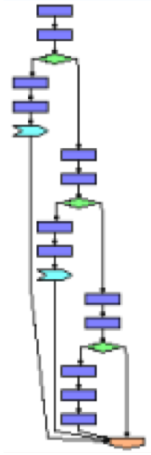
Java bemüht sich, die Vorteile beider Ansätze zu kombinieren:

- Java-Programme werden *zunächst* von einem *Compiler* übersetzt, ...
  - Dabei wird das Programm auf Syntax- und Semantik-Fehler überprüft, d.h. auf Schreibfehler und Widersprüche gegen die Sprachdefinition
  - Das Ergebnis ist ein (äußerlich) korrektes Programm in einer Zwischendarstellung, dem sogenannten **Java-Bytecode**

- ... *dann* wird der Bytecode von einem *Interpreter*, der **Virtuellen Maschine** (kurz: **VM**), ausgeführt
  - Überprüfungen können wegfallen, es liegt ein syntaktisch und semantisch korrektes Programm vor
  - Beim ersten Ausführen einer jeden Anweisung wird diese von einem (Just-In-Time-)Compiler in Maschinensprache für die aktuelle Maschine übersetzt -> das führt zu einem schnellen Programmablauf
  - Eventuelle Laufzeitfehler können komfortabel behandelt werden
  - Java-Programme laufen auf allen Prozessoren (und Betriebssystemen) identisch ab

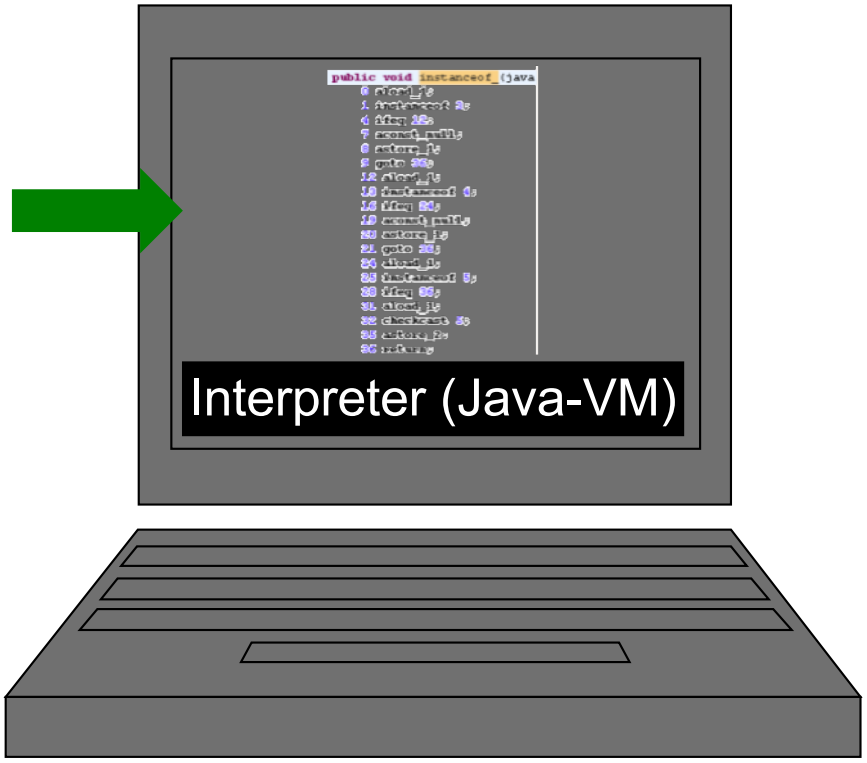
Ausführung von Java-Programmen 2/2

Java-Programm (höhere Programmiersprache)

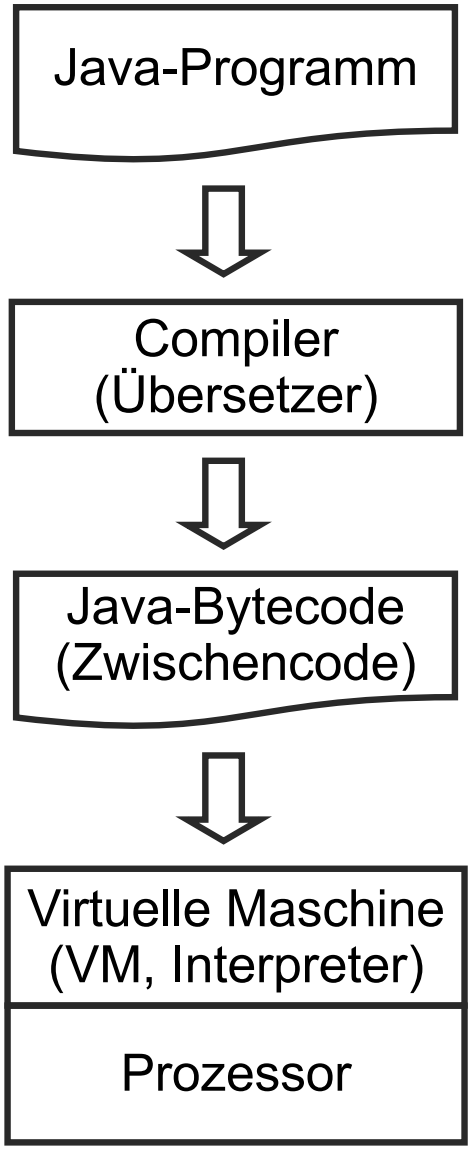


Compiler

Java-Bytecode

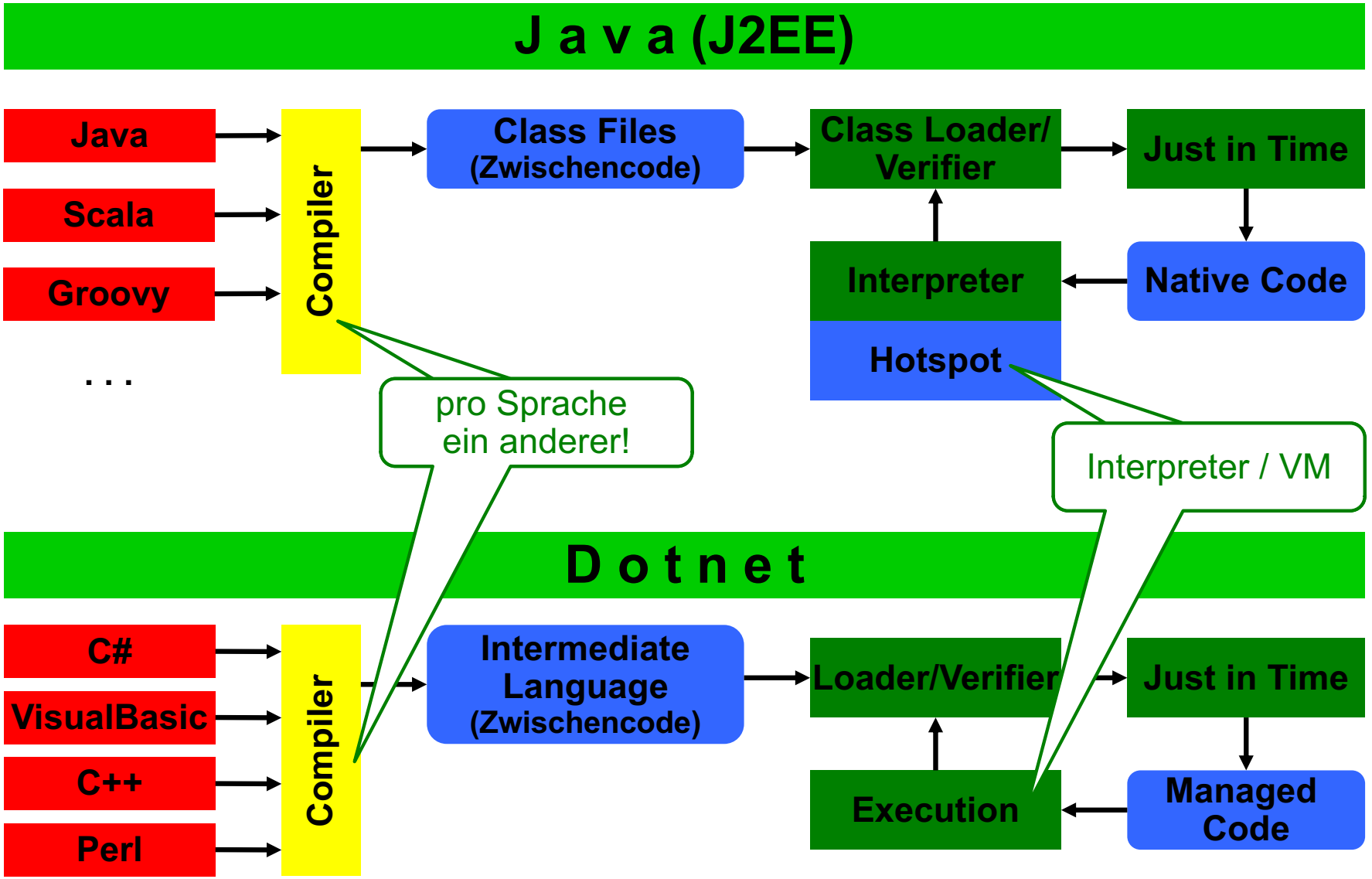


Ausführung von Java-Programmen 2/2



J2EE als Vorlage für MS-Dotnet (.net)

[orientiert an COMPUTER ZEITUNG 22/2002]



Programmiersprachen

- Programmiersprachen werden benötigt, um Algorithmen für Computer verständlich zu formulieren
- Wir unterscheiden Maschinensprache (und Assembler) von höheren (oder problemorientierten) Programmiersprachen
- Höhere Sprachen enthalten folgende **Sprachkonstrukte** (Sprachelemente)
  - Anweisungen: Addiere, Öffne Fenster, ...
  - Ausdrücke: 3+5, istRechnungBezahlt, ...
  - Vereinbarungen für Typen, Variablen, Unterprogramme etc.

- Diese Konstrukte existieren in fast allen Sprachen, jedoch in unterschiedlicher Ausprägung
- Es gibt einige weitere Hilfskonstrukte und Bibliotheken

Ausdrücke

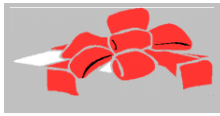
Anweisungen

Vereinbarungen

Zugriff auf geschachtelte Arrays `a[3][4]`

`0.25` Gleitkomma-Zahlen `3.14159...`

`buch.autor` Zugriff auf Objekte `cd.titel`



weitere Schleifen

`fakultät(8)` Funktionsaufrufe `f(x)`

`println(...);` Prozeduraufrufe `openFile(...);`

Zeichen `'x'` `'U'`

Zugriff auf Arrays `adresse[17]`

`ja / nein` Wahrheitswerte `true / false`

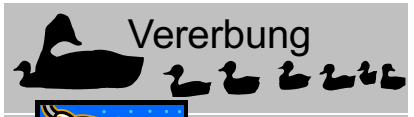
Zuweisungen `a = 5` `s = "Hallo!"`

abweisende Schleifen

Zeichenketten (String) `"Hallo!"`

`buch` `vorname` Variablen `text` `i`

Anw. folgen `{...}`



Interface

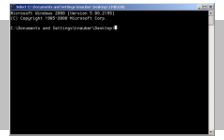
Klassen, Felder, Methoden

Unterprogramm-Definition

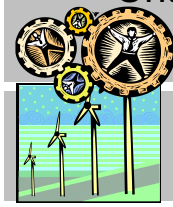
`4712` Ganze Zahlen

`-` `+` `++` `>=` `&` `*` Operatoren

Bedingte Anw. `If...`



Konsole



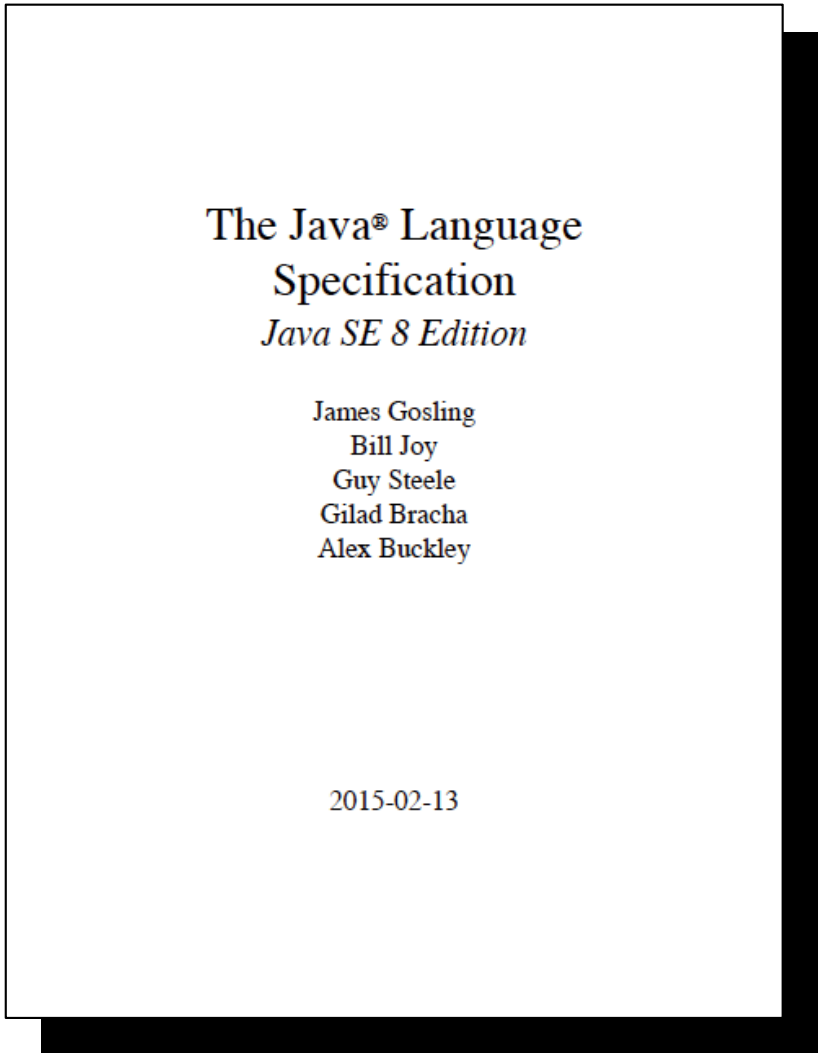
Arrays

HOCHSCHULE DER WIRTSCHA

Editoren

Compiler, Interpreter

Die Sprachbeschreibung von Java



The Java® Language Specification

Java SE 8 Edition

James Gosling
Bill Joy
Guy Steele
Gilad Bracha
Alex Buckley

2015-02-13

HOCHSCHULE DER WIRTSCHAFT FÜR MANAGEMENT
UNIVERSITY OF APPLIED MANAGEMENT STUDIES

F R A G E N