Java, Compiler und Interpreter

- Ausführung von Compiler-Sprachen
- Ausführung von Interpreter-Sprachen
- Vor- und Nachteile von Compilern und Interpretern
- Ausführung von Java-Programmen
- Prinzip:
  Ablauf von Java-Programmen
- Anleitung am Beispiel:
  Übersetzen und Ausführen von Java-Programmen

Ausführung höherer Programmiersprachen

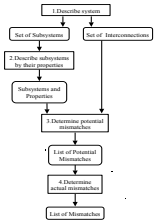
Im Hinblick auf die Art der Ausführung gibt es zwei Gruppen von (höheren) Programmiersprachen:

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

- Interpreter-Sprachen:
  Die Sprache wird von einem besonderen Programm, einem sogenannten *Interpreter*, interpretiert

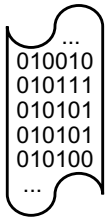
Ausführung von Compiler-Sprachen

Programm (höhere Programmiersprache)



Compiler

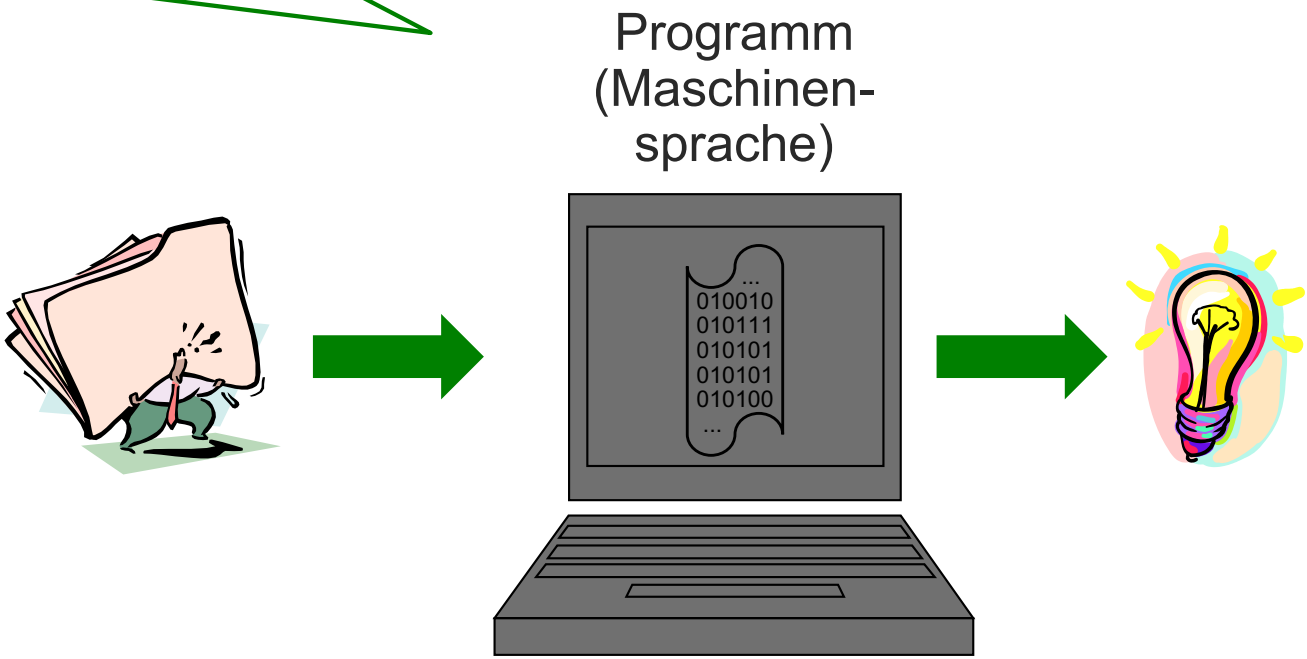
Programm (Maschinen-sprache)



Ein „Compiler“ ist auch ein Programm!

Ausführung von Compiler-Sprachen

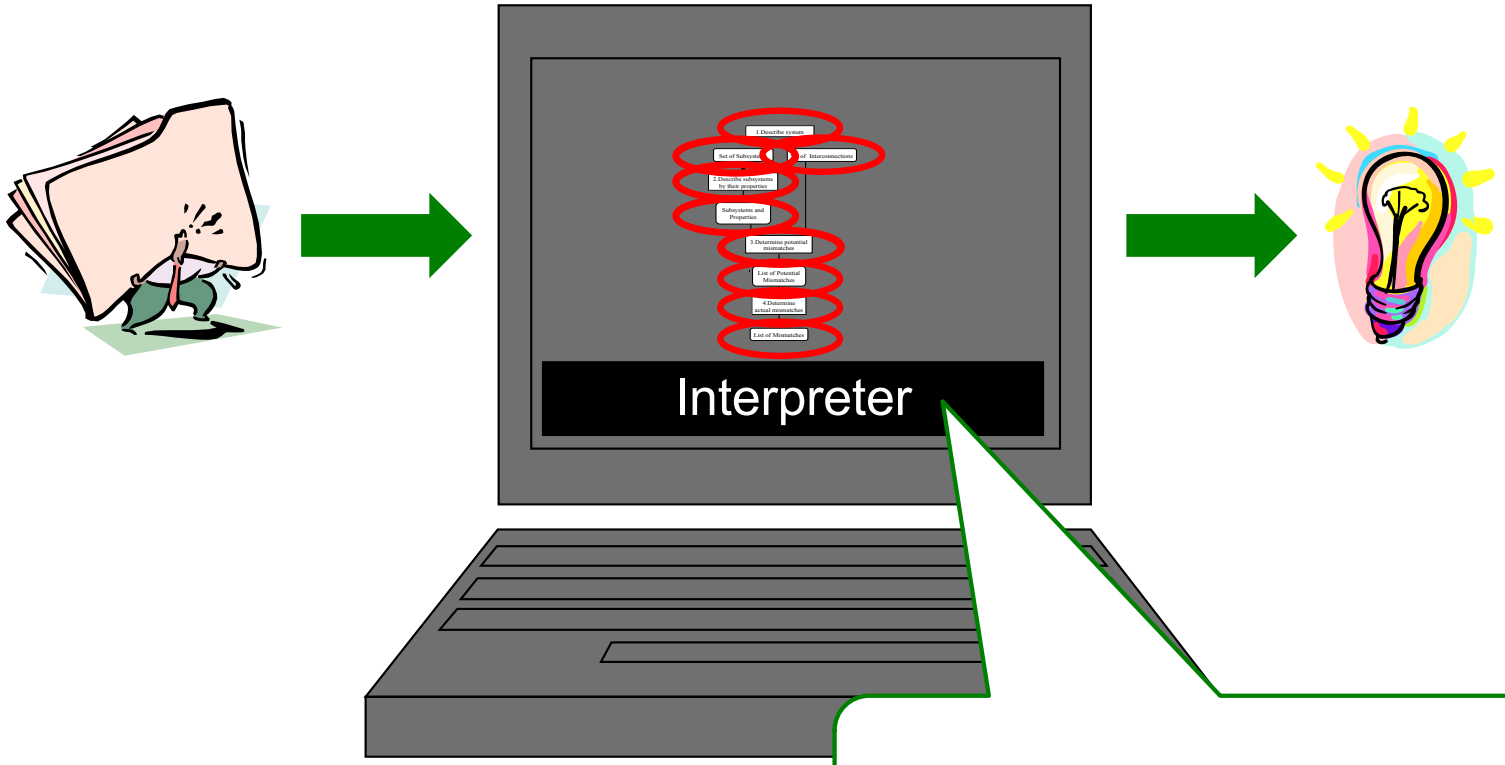
Maschinensprache ist nur vom Prozessor einer bestimmten „Maschine“ ausführbar: Das ist evtl. eine deutliche Einschränkung!



Ausführung von Interpreter-Sprachen

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

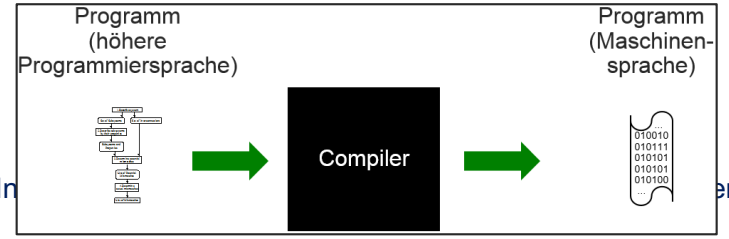
Programm (höhere Programmiersprache)

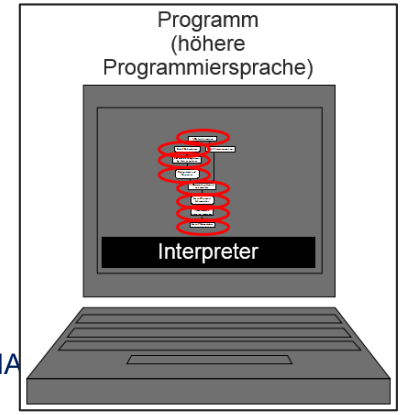


Ein „Interpreter“ ist auch ein Programm!

Vor- und Nachteile von Compilern und Interpretern 1/2

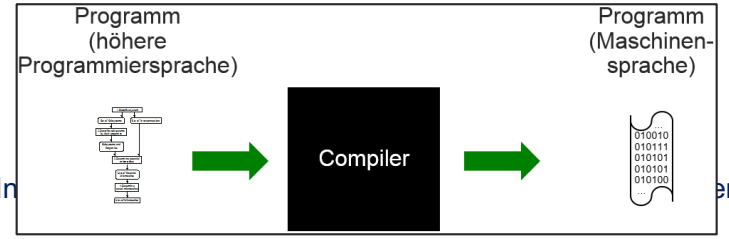
- Compiler
  - ✚ Compiler überprüfen das Programm bei der Übersetzung *vor* der Ausführung auf Syntax- und (viele) Semantik-Fehler
  - ✚ Anweisungen werden genau einmal (beim Übersetzen) geprüft und beim Ablauf direkt vom Prozessor ausgeführt
- Interpreter
  - Solche Fehler werden eventuell erst *bei* der Programmausführung bemerkt, ein Absturz kann die Folge sein
  - Anweisungen werden unnötigerweise bei jedem Durchlauf erneut geprüft (gilt nicht für Just-In-Time-Compiler

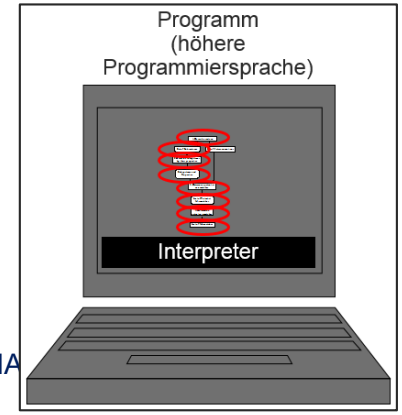




Vor- und Nachteile von Compilern und Interpretern 2/2

- 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 Zahlen 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

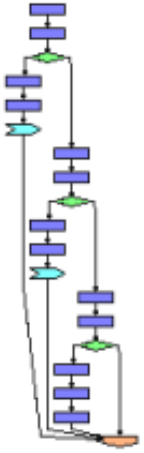
Java bemüht sich, die Vorteile beider Ansätze zu kombinieren:

- 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

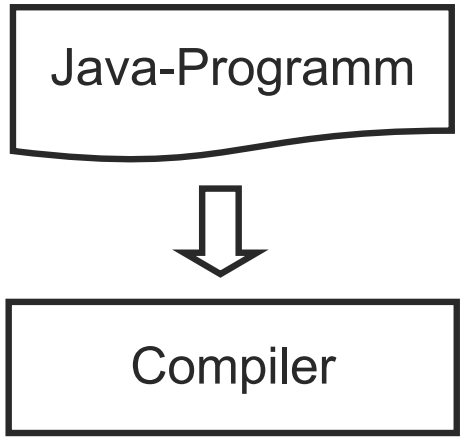
Java-Programm (höhere Programmiersprache)



Compiler

Java-Bytecode

Klassen

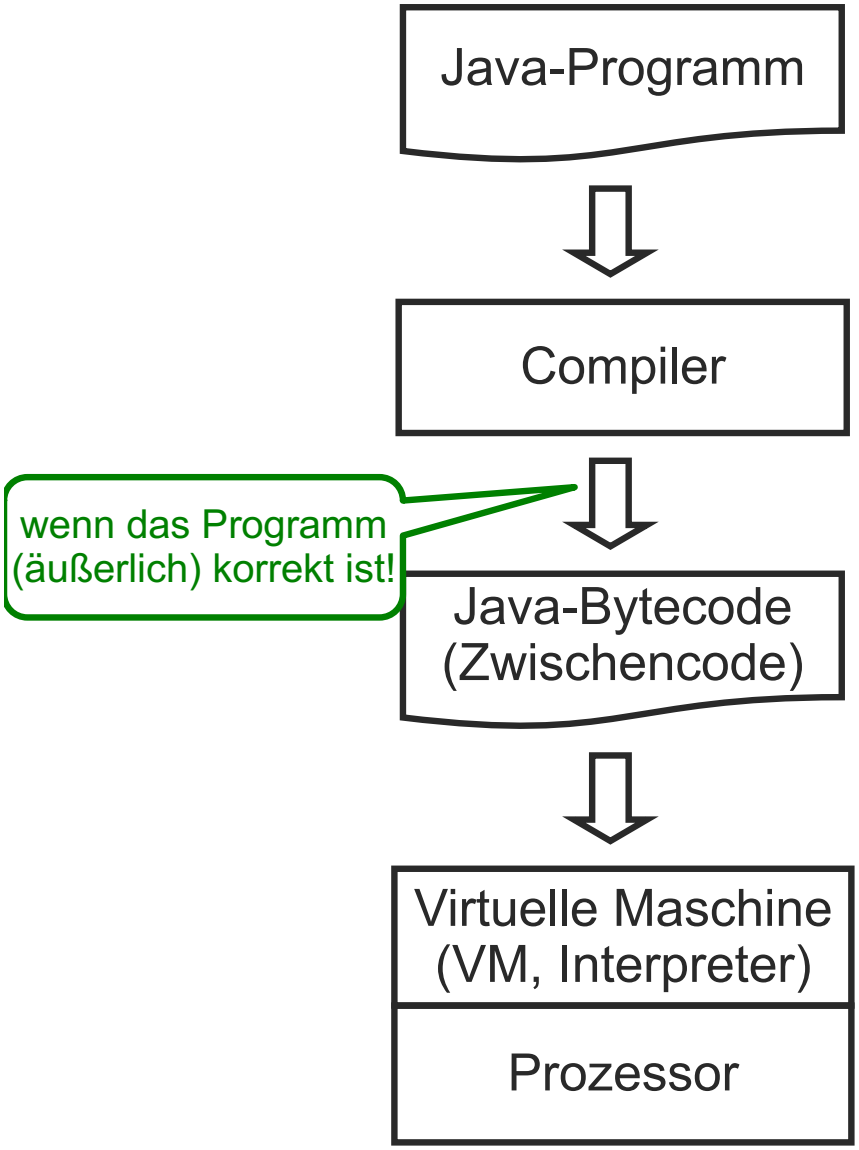


- Java-Programme liegen in Form von sogenannten *Klassen* vor

- Klassen beginnen mit dem Schlüsselwort „class“:

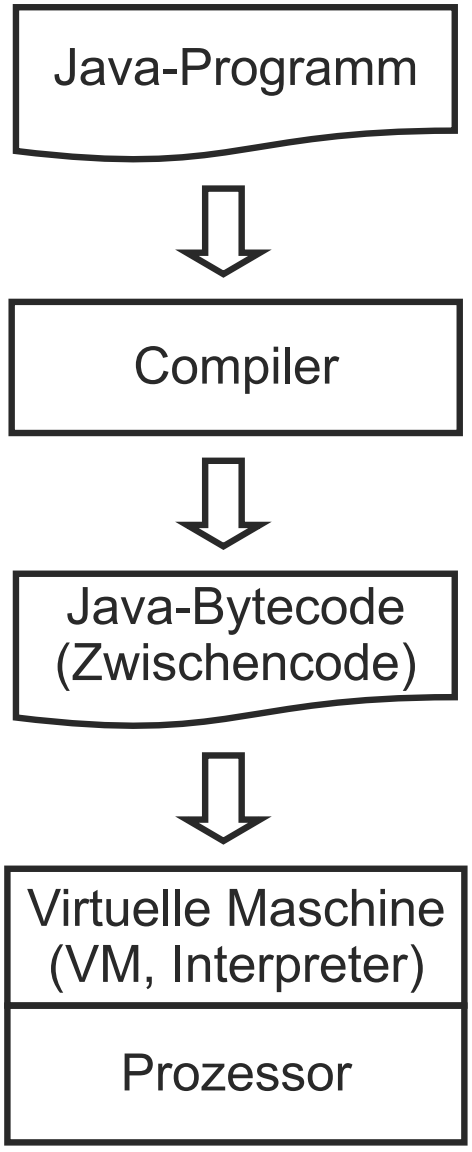
```
class Helloworld {

        public static void main( String[] args ) {
                println("Helloworld!");
        }

}
```

Klassen



- Java-Programme liegen in Form von sogenannten *Klassen* vor
- Ein Programm kann aus *mehreren* Klassen bestehen
- Eine der Klassen wird als Start-Programmteil festgelegt
  - Wir werden später sehen, wie das funktioniert
  - Bisher haben wir nur mit *Programmen* gearbeitet, die aus *einer einzelnen Klasse* bestehen

Klassen




```
class HelloWorld {
```

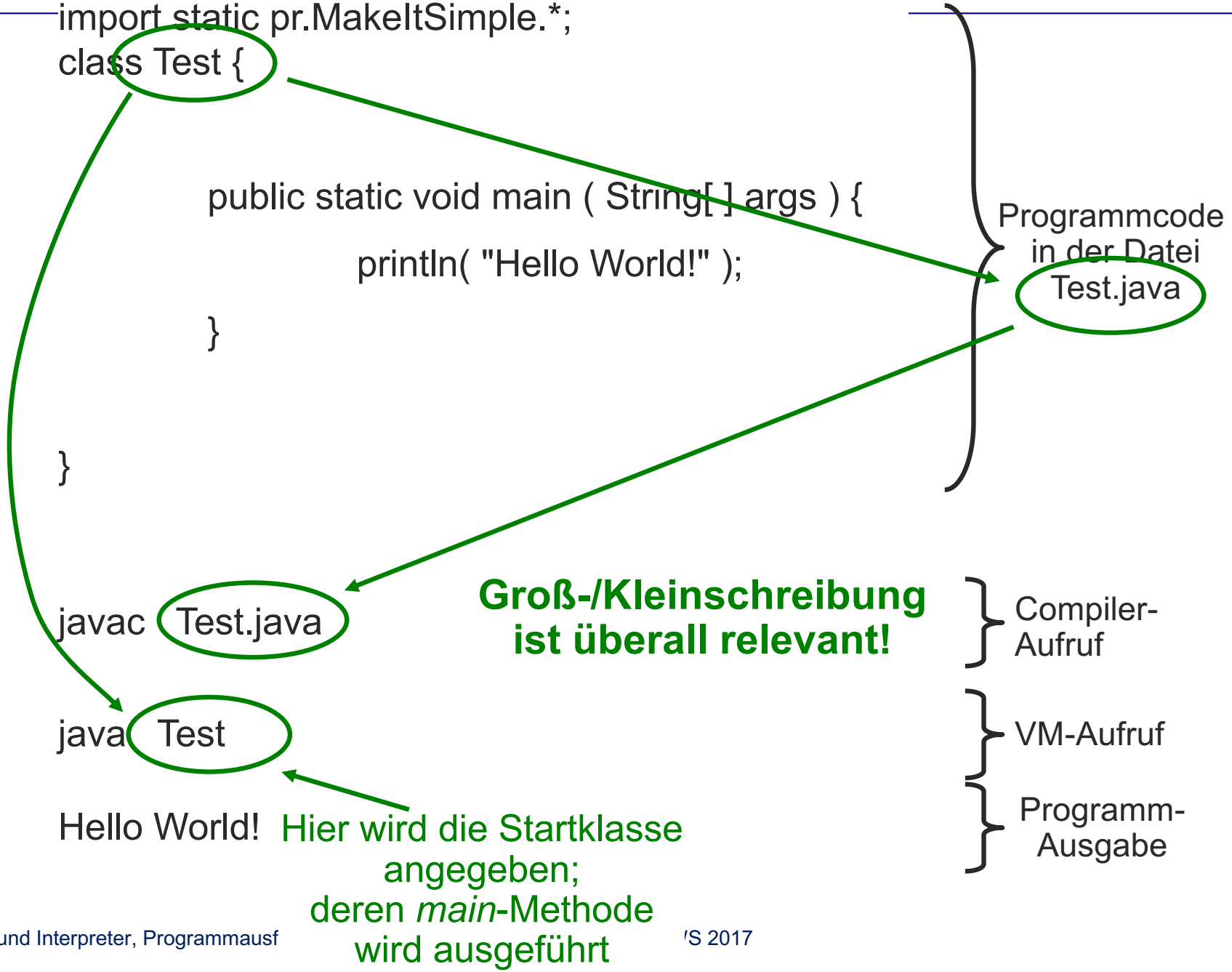
Dateiname: HelloWorld.java

javac HelloWorld.java

Soll mehr als eine zusammengehörige Datei übersetzt werden, kann man alle Dateinamen angegeben

java HelloWorld

Übersetzen und Start eines Java-Programms von der Kommandozeile (Unix, Linux, DOS)



Demo

Merk-Regel:

1 Klasse = 1 Datei