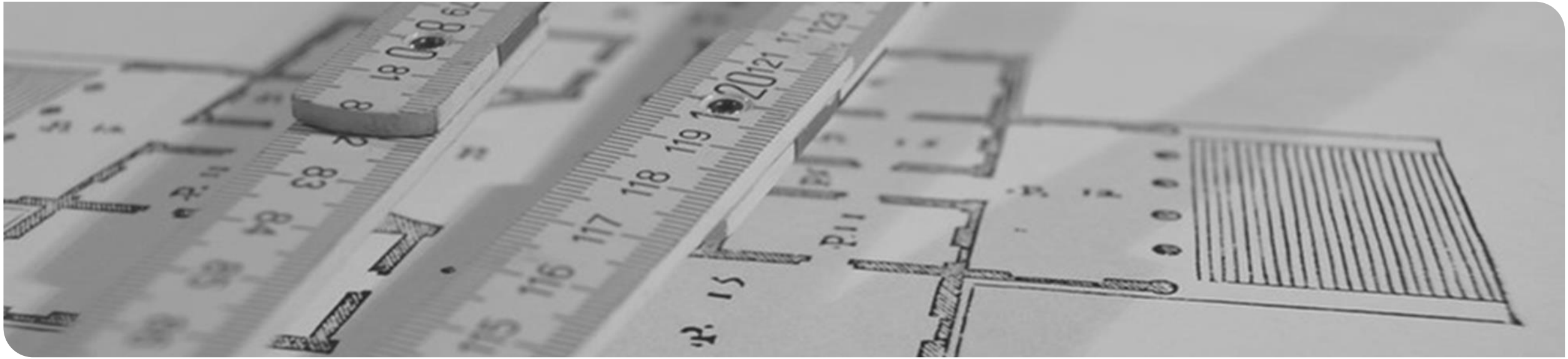
Karlsruher Institut für Technologie

Vorlesung Programmieren

1.1 Ein einfaches Programm

Prof. Dr.-Ing. Anne Koziolek



Fragen?

Beitreten über

slido.com

#Prog

Semesterplan für die Vorlesung

21.10.2024	Erstsemesterbegrüßung: Einführung
23.10.2024	Organisatorisches; Ein Einfaches Programm, Objekte und Klassen
30.10.2024	Typen und Variablen
06.11.2024	Kontrollstrukturen (+Scanner)
13.11.2024	Konstruktoren und Methoden
20.11.2024	Arrays; Konvertierung, Datenkapselung, Sichtbarkeit
27.11.2024	Listen und Abstrakte Datentypen
04.12.2024	Vererbung **~~Audimax~~ 50.34 Raum -101 und Raum -102**
11.12.2024	Exceptions; Interfaces
18.12.2024	Generics; Rekursion
08.01.2025	Java-API; Objektorientierte Design-Prinzipien
15.01.2025	Best Practices; Finden und Beheben von Fehlern
22.01.2025	Testen und Assertions
29.01.2025	JUnit; Parsen, Suchen, Sortieren
05.02.2025	Vom Programm zur Maschine; Ausblick auf zukünftige Lehrveranstaltungen
12.02.2025	Wrap-Up

Lernziele

Methodik

Grundlagen
- Ausführen
- Ein-/Ausgabe

- Sie können ein einfaches Java-Programm…
  - schreiben
  - kompilieren
  - ausführen
- Sie kennen die Grundelemente eines Java-Programms
- Sie können ein Java-Programm schreiben, das
  - Text ausgibt
  - Einfache Rechenoperationen ausführt
  - Methodenaufrufe enthält

Programmierung

Programmieren:
Einen Berechnungsprozess für den Computer präzise festlegen

Beispiel 1: Algorithmus für Summation

- Lösungsverfahren:
  - Summe um Laufvariable i erhöhen
  - Laufvariable um 1 erhöhen
  - Solange die Laufvariable kleiner gleich n ist
- Präzise festgelegtes Lösungsverfahren: Algorithmus
- Programm: Aufschreiben des Algorithmus in einer Programmiersprache (hier: Java)

$$sum(n) = \sum_{i=1}^{n} i = 1 + 2 + \ \ldots \ + n$$

```
int n = 42;
int i = 1;
int sum = 0;
while (i <= n) {
  sum = sum + i;
  i = i + 1;
}
```

n sei zu Beginn 42
i sei zu Beginn 1
sum sei zu Beginn 0
Wiederhole, solange i kleiner n ist:
    Erhöhe sum um die Laufvariable i
    Erhöhe die Laufvariable i um 1
Ende der Wiederholung

$$sum(42) = \sum_{i=1}^{42} i = 903$$

Beispiel 2: Algorithmus zur Berechnung der Summenformel

- Wie kann das Programm optimiert werden?
  - Vor dem Programmieren erstmal über das Problem nachdenken.
  - Ergebnis: Man kann die ersten n aufeinanderfolgenden natürlichen Zahlen mit der Gaußschen Summenformel direkt ausrechnen:

$$\text{sum}(n) = \sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

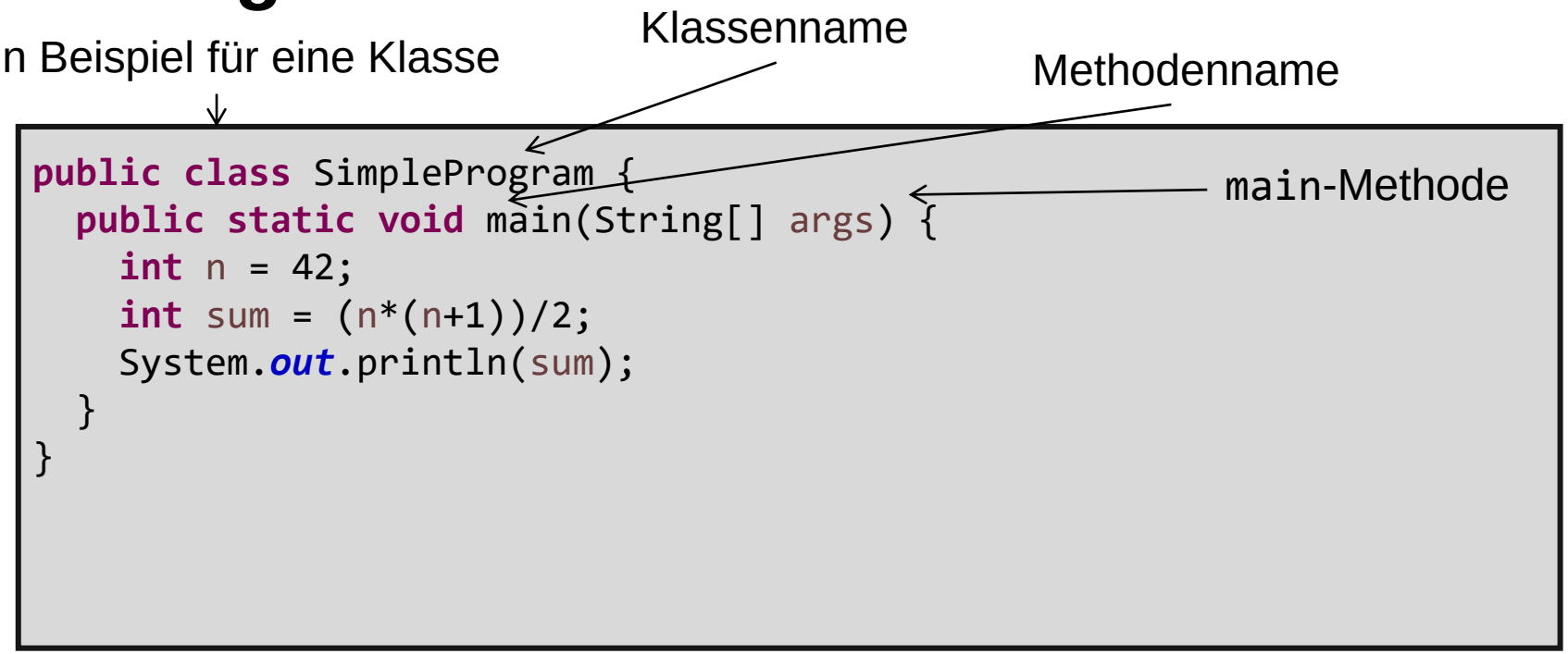
```
int n = 42;
int i = 1;
int sum = 0;
while (i <= n) {
  sum = sum + i;
  i = i + 1;
}
```

⇕

```
int n = 42;
int sum = (n*(n+1))/2;
```

Einfaches Programm

Ein Beispiel für eine Klasse



Einfaches Programm

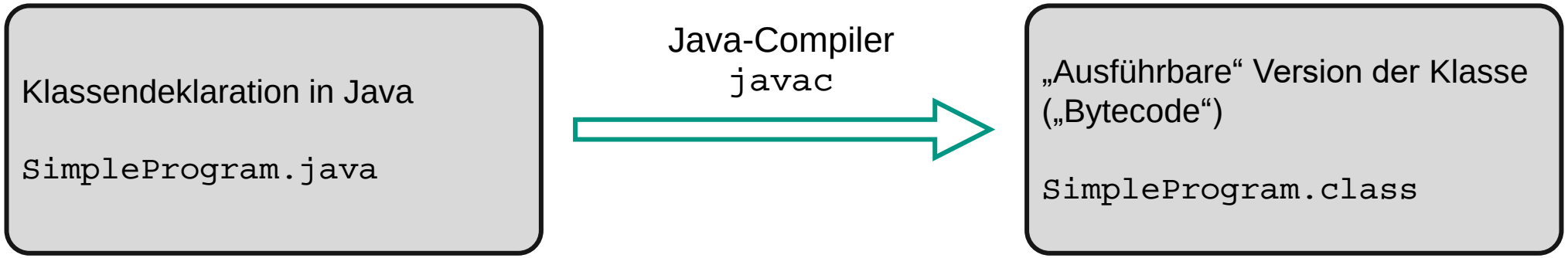
```java
public class SimpleProgram {
  public static void main(String[] args) {
    int n = 42;
    int sum = (n*(n+1))/2;
    System.out.println(sum);
  }
}
```

- Wie starte ich mein Java-Programm? (Demo)

Zur Installation von Java siehe: ILIAS-Wiki

Praxis: Übersetzen von Java-Klassen

- Der Prozessor „versteht“ kein Programm in einer höheren Programmiersprache, wie z.B. Java!
  - ➔ Er verarbeitet nur Binärcode!
- Java-Programme müssen daher zunächst übersetzt („compiliert“) werden
  - javac
- Jede Klasse in einer eigenen Datei:
  - Datei „SimpleProgram.java“ → class SimpleProgram
  - Dateiname muss dem Klassennamen entsprechen



Einfaches Programm

```java
public class SimpleProgram {
  public static void main(String[] args) {
    int n = 42;
    int sum = (n*(n+1))/2;
    System.out.println(sum);
  }
}
```

```
...$javac SimpleProgram.java
```

- Wie starte ich mein Java-Programm? (Demo)
  - Kompilieren

Praxis: Ausführen von Java-Programmen

- Mit `javac` übersetze Programme (Klassen) werden mit dem Java-Interpreter `java` ausgeführt. Im Beispiel:

```
java SimpleProgram
```

- Dabei wird in der angegebenen Klasse die Methode main gesucht und diese gestartet. `main` muss deklariert sein als

```
public static void main(String[] args) { ... }
```

- Ausgaben können mittels `System.out.print()` und `System.out.println()` vorgenommen werden; letzteres gibt einen Zeilenvorschub mit aus

Einfaches Programm

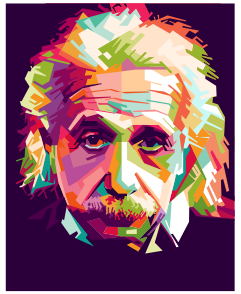
```java
public class SimpleProgram {
  public static void main(String[] args) {
    int n = 42;
    int sum = (n*(n+1))/2;
    System.out.println(sum);
  }
}
```

```
...$javac SimpleProgram.java
...$java SimpleProgram
903
...$
```

- Wie starte ich mein Java-Programm? (Demo)
  - Kompilieren
  - Ausführen

Wozu Compiler? (dt. “Übersetzer”)

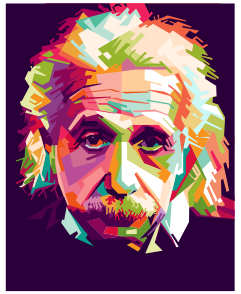
- Ein Algorithmus ist präzise festgelegte Handlungsvorschrift zur Lösung eines Problems oder (abstrakter) einer Klasse von Problemen
- Ein Programm ist ein Algorithmus, der in einer Programmiersprache geschrieben wurde.
- Die Programmierer „sprechen“ also mit dem Rechner.
- Dies kann in einer „Sprache“ erfolgen, die der Rechner direkt versteht.
  - ➔ Binärcode
    - Eine Notation mit Nullen und Einsen
    - Für Menschen schwer verständlich
- Idee 1: Maschinennahe Sprachen, auch Assemblersprache genannt.
- Assemblersprache besteht aus Buchstaben und Ziffern, z.B.:
  - MOV RG1 RG5
  - ADD RG2 RG4
- Kann sehr einfach in Binärcode umgewandelt werden.



Wozu Compiler?

- Assemblercode kann nach der leichten Umwandlung in Binärcode direkt vom Prozessor ausgeführt werden.
- Nachteil: Programm abhängig vom Prozessortyp
- Idee 2: Verwenden einer benutzernahen Sprache
  - ➔ Höhere Programmiersprachen
  - ➔ Übersetzer notwendig

- Übersetzer ist eine spezielle Software, die den Code einer höheren Programmiersprache übersetzt in
  - die Maschinensprache oder direkt Binärcode
  - oder eine Zwischensprache (dazu gleich mehr)



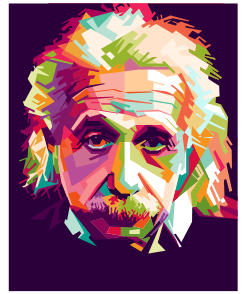
Compiler vs. Interpreter

Compiler

- Compiler transformieren die Programme in einer höheren Sprache in maschinennahe Programme.
- Die Programme werden später ausgeführt.

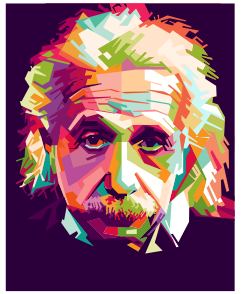
Interpreter

- Interpreter werden benutzt, wenn die Programme Anweisung für Anweisung übersetzt werden.
- Die Programme werden unmittelbar ausgeführt.



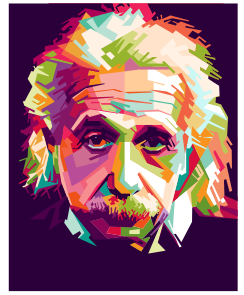
Übersetzung in maschinennahe Sprache

- Traditionelle höhere Programmiersprachen benötigen für jeden Prozessortyp einen Compiler.
- Der Quellcode wird in das Zielprogramm übersetzt.
  ➔ Das Programm muss für jeden Prozessortyp übersetzt werden.

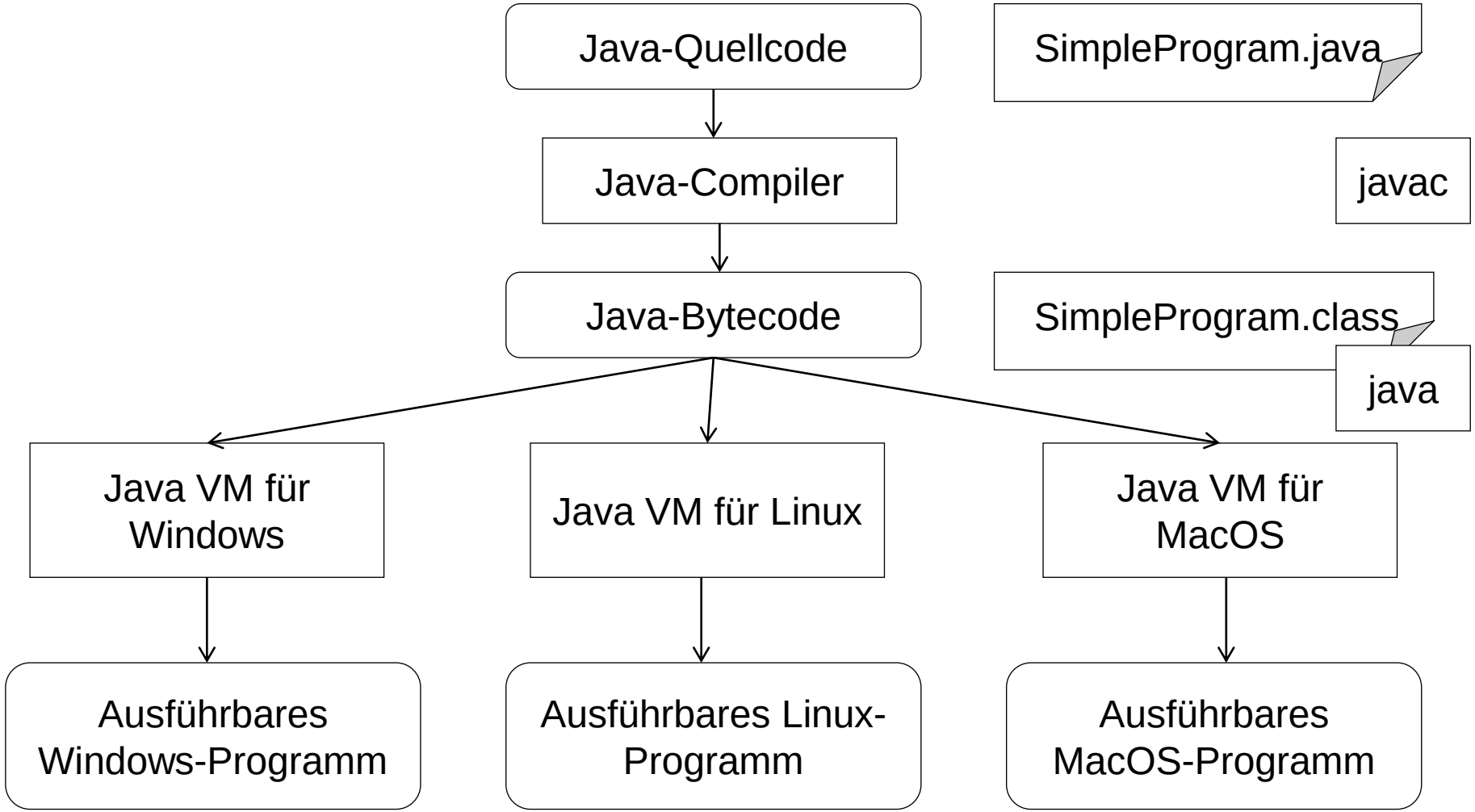


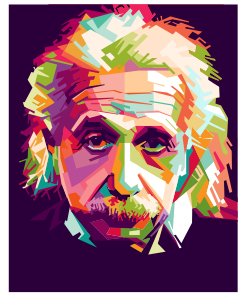
Interpretation nach Vorübersetzung

- Java hingegen ist plattformunabhängig.
  - ➔ Nur ein Compiler wird für alle Plattformen benötigt.
- Compiler übersetzt den Quellcode in Java-Bytecode.
- Java-Bytecode ist unabhängig von einem bestimmten Prozessor.
- Java-Bytecode kann nicht unmittelbar ausgeführt werden.
  - ➔ Java-Bytecode ist portabel.
  - ➔ Java-Bytecode wird für einen „virtuellen Prozessor“ gefertigt.
- Der virtuelle Prozessor wird die virtuelle Maschine (VM) genannt.
- Java-Interpreter analysiert den Java-Bytecode und führt ihn aus.



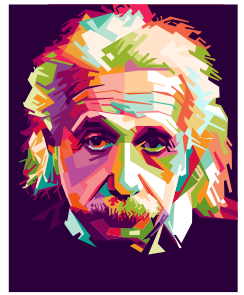
Übersetzung und Ausführung in Java





Just In Time Compilation

- Ein leichtgewichtiger Compiler
- Findet während der Ausführung des Codes statt
- Ziel: Optimierung des Codes zum Verbessern der Performance
  ➔ Optimierung des häufig auszuführenden Codes, wie z.B. Schleifen
- Beispiele für die Code-Optimierung:
  - Entfernen von redundanten Aufrufen (z. B.: durch Ersetzen mit einer lokalen Variable)
  - Copy Propagation (z. B.: Ersetzen von y=f und z=x+y durch z=x+f)
  - Entfernen vom toten Code (Code, der nie aufgerufen wird oder dessen Ergebnis nie benutzt wird)



Einfaches Programm

```java
public class SimpleProgram {
  public static void main(String[] args) {
    int n = 42;
    int sum = (n*(n+1))/2;
    System.out.println(sum);
  }
}
```

```
...$javac SimpleProgram.java
...$java SimpleProgram
903
...$
```

- Wie ändere ich die Ausgabe?

Einfaches Programm

```java
public class SimpleProgram {
  public static void main(String[] args) {
    int n = 43;
    int sum = (n*(n+1))/2;
    System.out.println(sum);
  }
}
```

```
...$javac SimpleProgram.java
...$java SimpleProgram
903
...$
```

- Wie ändere ich die Ausgabe?

Einfaches Programm

```java
public class SimpleProgram {
  public static void main(String[] args) {
    int n = 43;
    int sum = (n*(n+1))/2;
    System.out.println(sum);
  }
}
```

```
...$javac SimpleProgram.java
...$java SimpleProgram
903
...$java SimpleProgram
903
```

- Wie ändere ich die Ausgabe?
- Muss ich das Programm wieder kompilieren?
- Ja

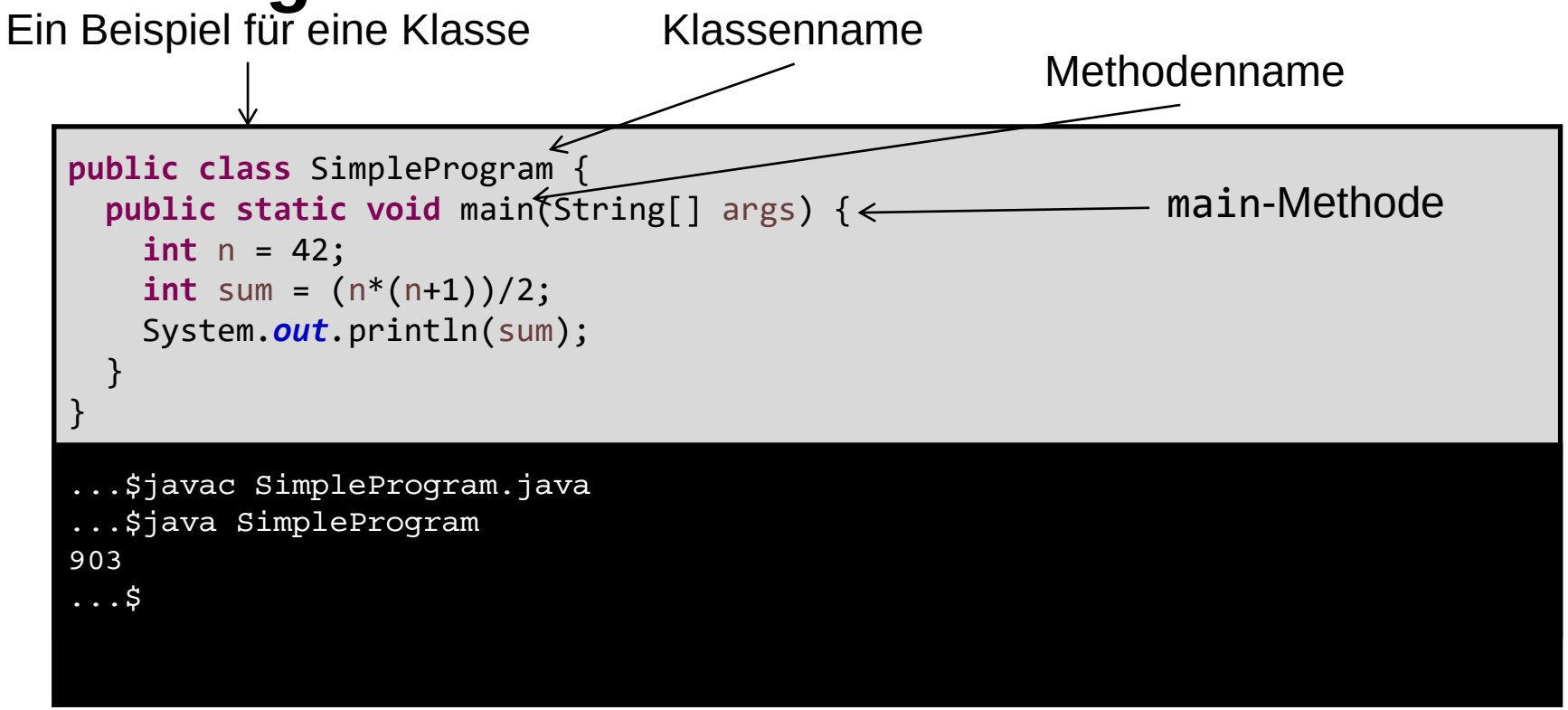
Einfaches Programm

```java
public class SimpleProgram {
  public static void main(String[] args) {
    int n = 43;
    int sum = (n*(n+1))/2;
    System.out.println(sum);
  }
}
```

```
...$javac SimpleProgram.java
...$java SimpleProgram
903
...$javac SimpleProgram.java
...$java SimpleProgram
946
```

- Wie ändere ich die Ausgabe?
- Muss ich das Programm wieder kompilieren?
- Ja

Einfaches Programm

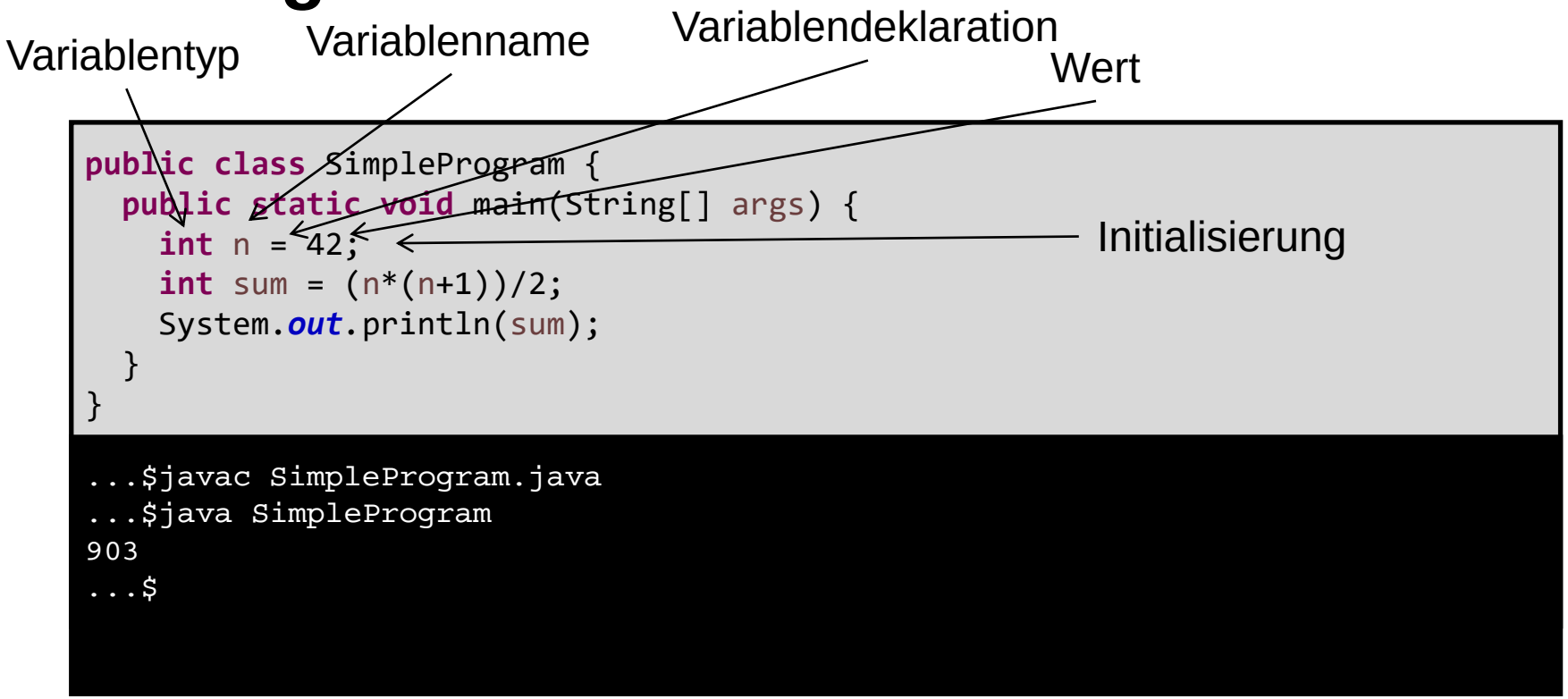



```
public class SimpleProgram {
  public static void main(String[] args) {
    int n = 42;
    int sum = (n*(n+1))/2;
    System.out.println(sum);
  }
}
```

```
...$javac SimpleProgram.java
...$java SimpleProgram
903
...$
```

- Woraus besteht das einfache Beispiel?

Einfaches Programm



- Woraus besteht das einfache Beispiel?

Einfaches Programm

```java
public class SimpleProgram {
  public static void main(String[] args) {
    int n = 42;
    int sum = (n*(n+1))/2;
    System.out.println(sum);
  }
}
```

```
...$javac SimpleProgram.java
...$java SimpleProgram
903
...$
```

- Wie können Parametern von der Kommandozeile aus an ein Programm übergeben werden?

Einfaches Programm

Zeichenkette umwandeln in eine natürliche Zahl

Kommandozeilen-parameter

Erster Kommando-zeilenparameter

```
public class SimpleProgram {
  public static void main(String[] args) {
    int n = Integer.parseInt(args[0]);
    int sum = (n*(n+1))/2;
    System.out.println(sum);
  }
}
```

```
...$javac SimpleProgram.java
...$java SimpleProgram 42
903
...$
```

- Wie können Parametern von der Kommandozeile aus an ein Programm übergeben werden?
- `args` ist die Liste der Kommandozeilenparameter

Literaturhinweis - Weiterlesen

- Dietmar Ratz, Jens Scheffler, Detlef Seese und Jan Wiesenberger „Grundkurs Programmieren in Java“, 7. Auflage, 2014 (mit Java 8), Hansa-Verlag
  - Abschnitt 2.2. „Was heißt Programmieren?“
  - Abschnitt 3.1. „Mein erstes Programm“
  - Online aus dem KIT-Netz: http://doi.org/10.3139/9783446453845

Zusammenfassung

- Programme sind
  - Präzise festgelegte Prozesse
  - Eindeutige Instruktionen
  - Allgemeiner Plan, der angibt, welcher Schritt als nächstes zu tun ist

- Einfaches Java-Programm
  - Klasse mit main-Methode
  - Kompilieren und Ausführen
  - Ausgabe eines Textes mit `System.out.println(…)`
  - Rechnen mit Variablen

Ihre Programmiererfahrung?

Beitreten unter
slido.com
#Prog

Multiple choice
Votes: 0

Haben Sie bereits programmiert (in irgendeiner Sprache)? Wenn ja, wie lange bereits? Bitte rechnen Sie Pausen, also Jahre in denen Sie sich nicht mit dem Thema beschäftigt haben, nicht mit.

Multiple choice
Votes: 0

Haben Sie an einem BoGy-Praktikum an der KIT-Fakultät für Informatik teilgenommen?