

Programmieren II: Java

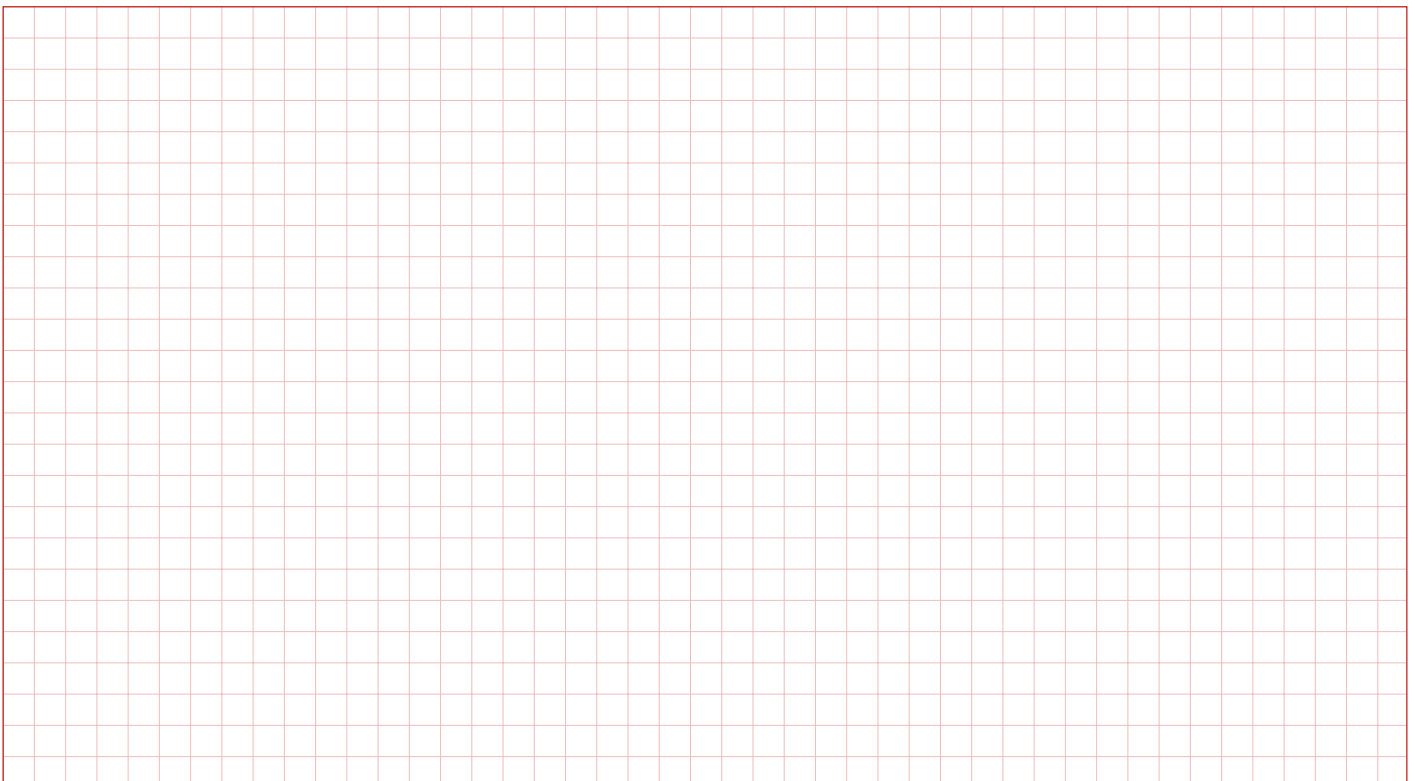
Einführung

Prof. Dr. Christopher Auer

Sommersemester 2024



Notizen



Eine kurze Geschichte von Java

Was macht Java aus?

Java-Plattformen und Implementierungen

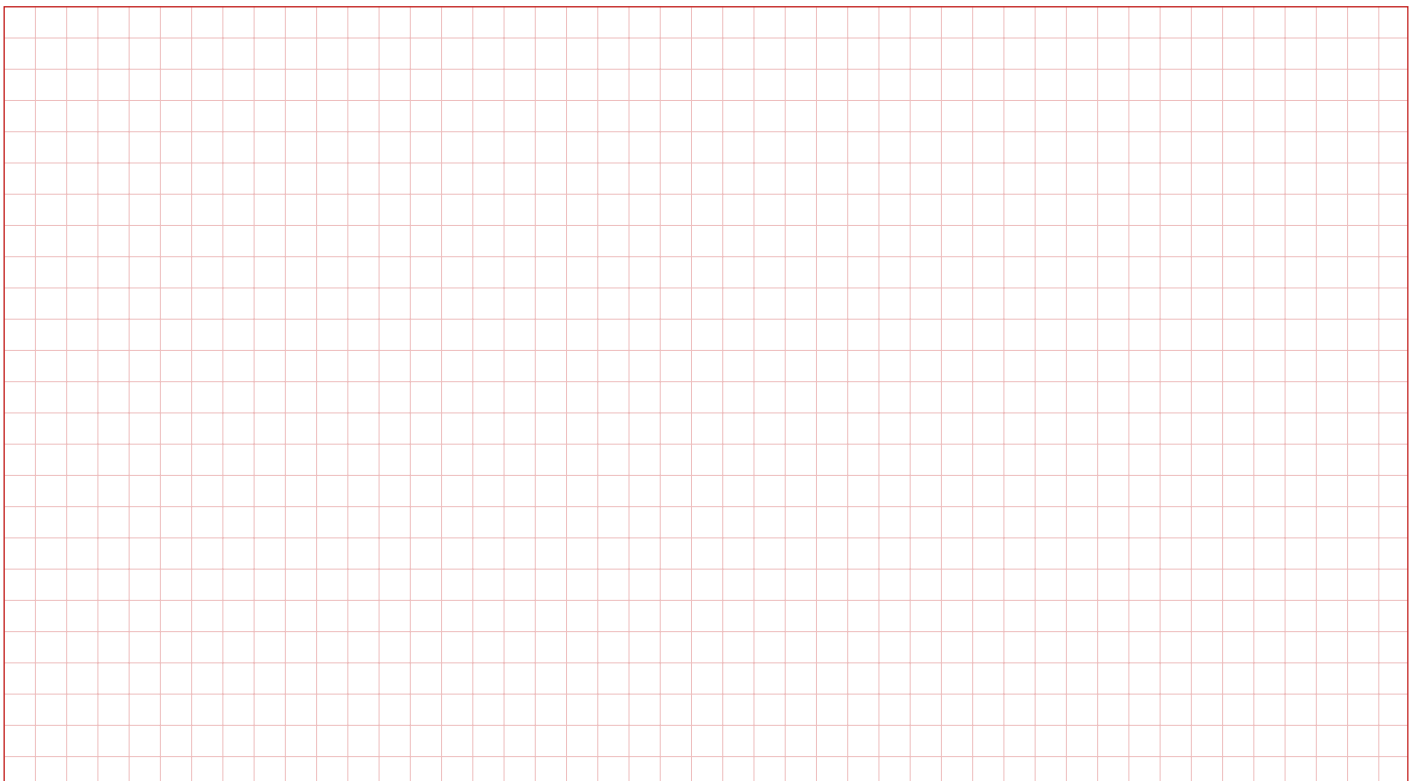
„Hello World“

Entwickeln mit Java

Literatur und Ressourcen

Überblick

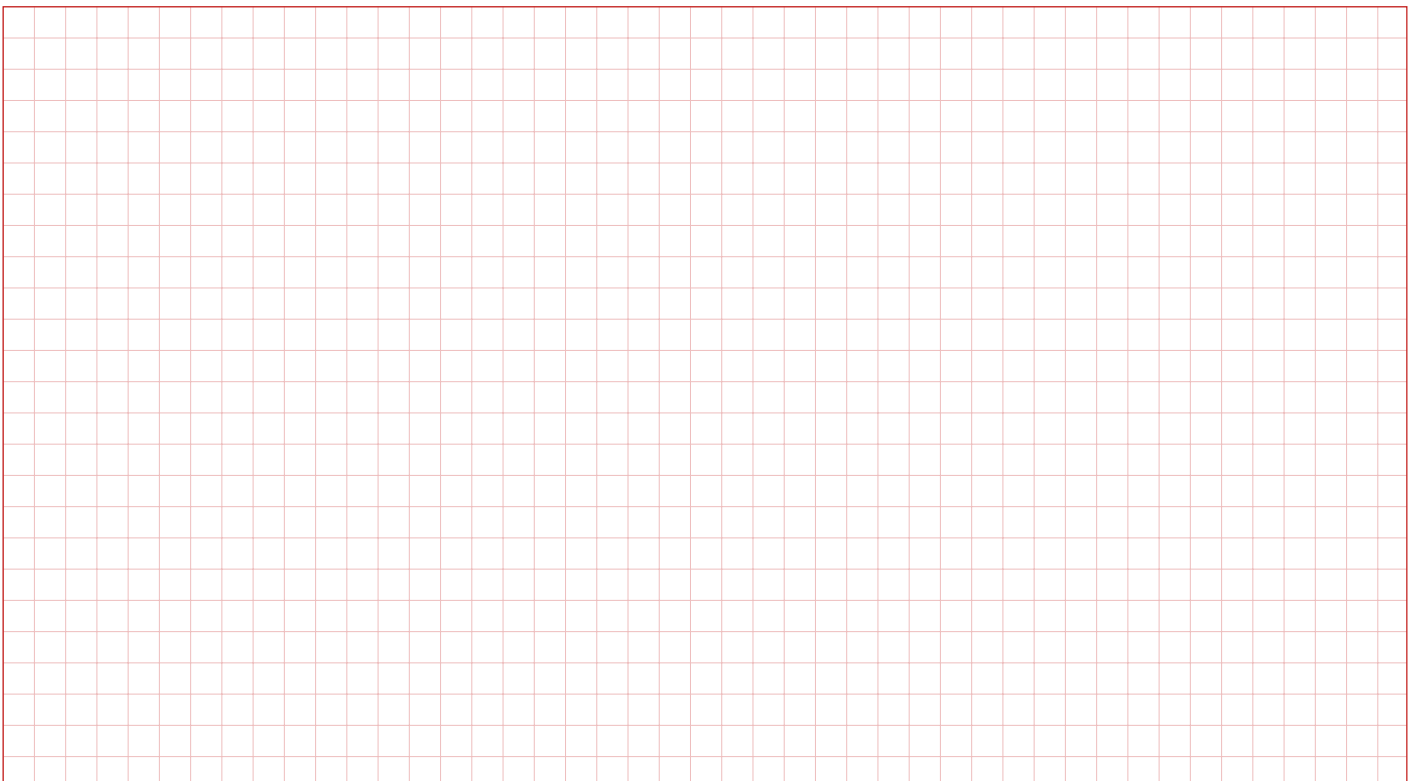
Notizen



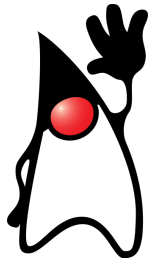
Inhalt

Eine kurze Geschichte von Java

Notizen

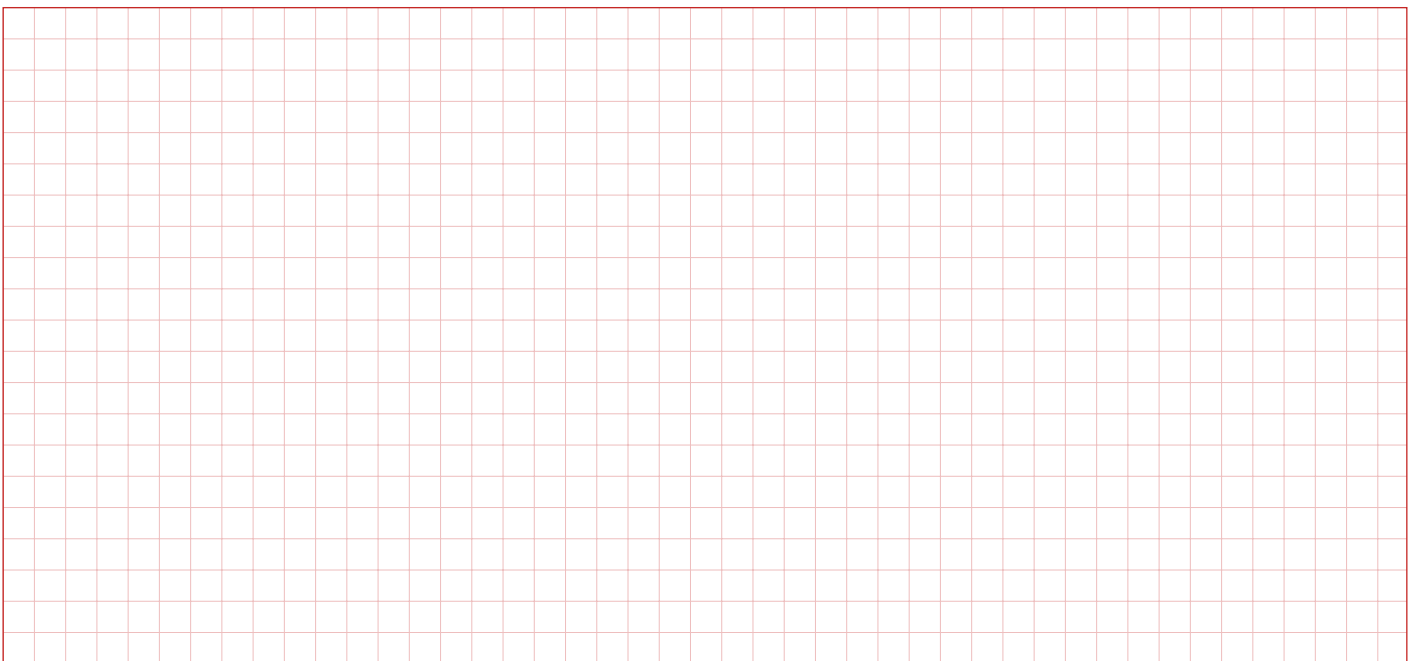


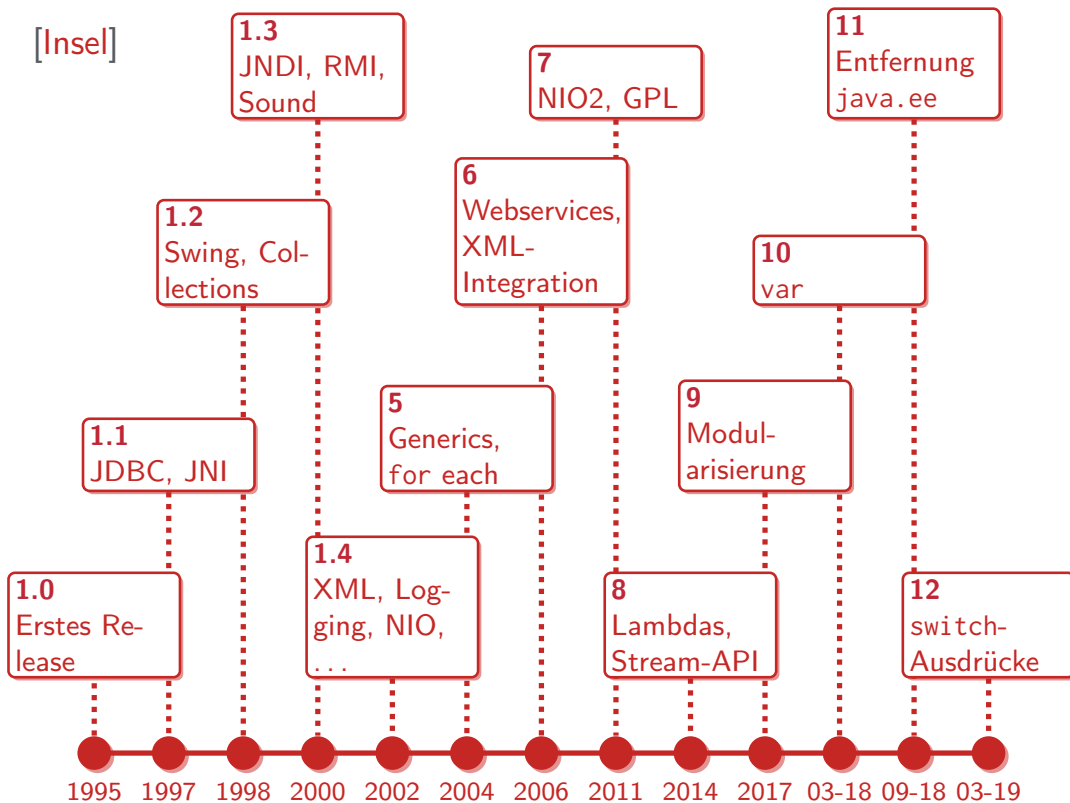
- ▶ Sun Microsystems
 - ▶ Anfang 1990er
 - ▶ Bill Joy, James Gosling
 - ▶ Unzufriedenheit mit C++ (Skalierbarkeit)
 - ▶ Oak: „Object Application Kernel“
 - ▶ Ziel (1993): Software für Set-Top-Boxen (Fernseher)
 - ▶ Neuer Name/neues Ziel (1994): Java/Web-Applets
 - ▶ 1995: Version 1.0
- ▶ Oracle
 - ▶ 2009: Oracle kauft Sun
 - ▶ Kommerzialisierung



Notizen

- Bild: Duke by [Sun Microsystems/Duke Project](#) licensed under BSD License 2.0





Notizen

JDBC „Java Database Connectivity“; Datenbankanbindung für Java-Anwendungen

JNI „Java Native Interface“; Schnittstelle zu C/C++-Programmen

Swing UI-Bibliothek

Collections Listen, Dictionaries, etc.

JNDI „Java Naming and Directory Interface“; Schnittstelle für Namensdienste

RMI „Remote Method Invocation“; Aufrufen von Java-Methoden über das Netzwerk

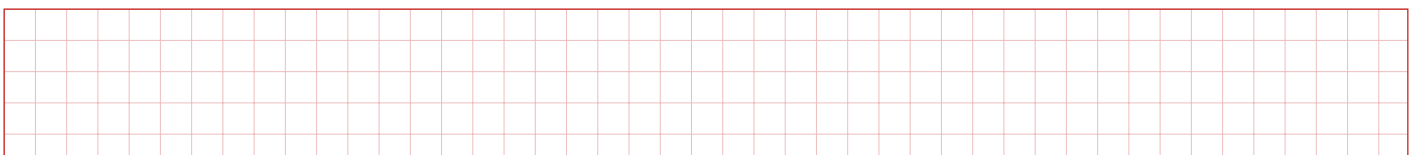
NIO „New Input/Output“; neue Schnittstelle zum Arbeiten mit Datenströmen (Dateien, Netzwerk, etc.)

Generics generische Templateklassen (ähnlich zu C++-Templates)

Lambdas Funktionen können als Parameter übergeben und Variablen zugewiesen werden

var var erlaubt Variablendeklarationen ohne explizit Typangabe

java.ee „Java Enterprise Edition“; Module für Webservices, Application-Servers, etc.

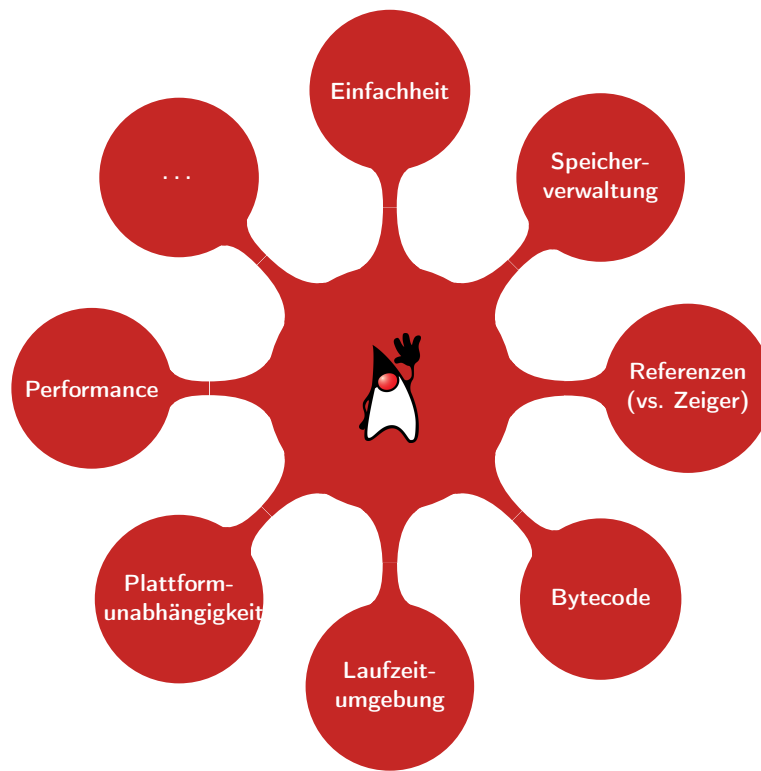


Inhalt

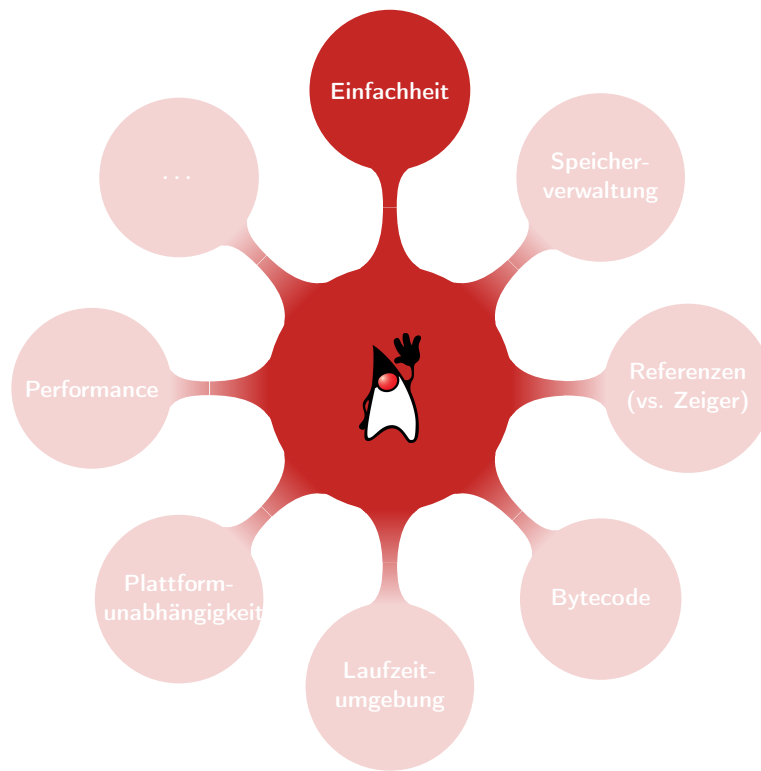
Was macht Java aus?

6

Notizen



Notizen

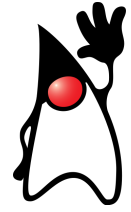


Notizen

- ▶ Syntax und Konzepte basieren auf bekannten Sprachen (C++)

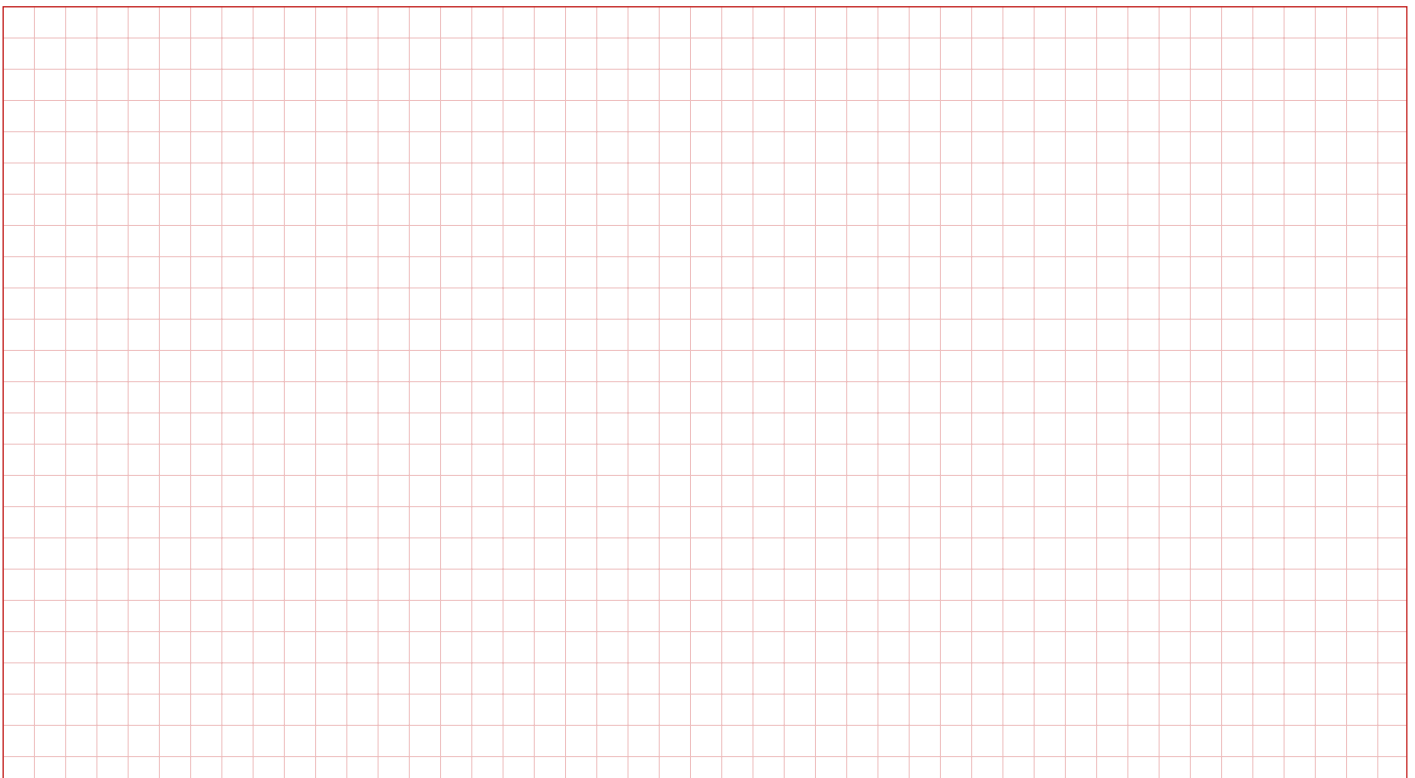
```
public class HelloWorld{  
    public static int add(int a, int b){  
        return a + b;  
    }  
    public static void main(String[] args){  
        System.out.println("Hello World!");  
    }  
}
```

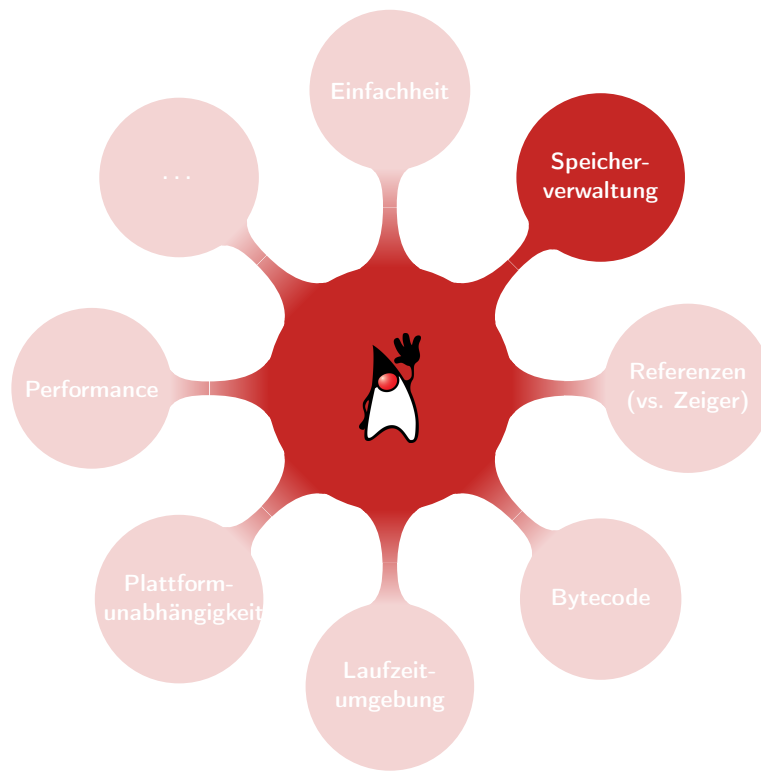
HelloWorld.java



- ▶ Aber:
 - ▶ keine Textersetzungs-Makros
 - ▶ keine Operatorenüberladung
 - ▶ keine Mehrfachvererbung

Notizen





Notizen

► C++

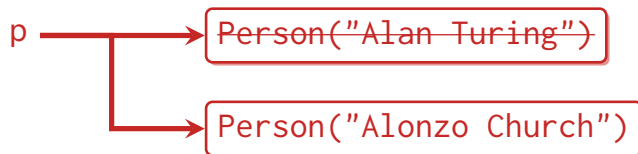
```
Person* p = new Person("Alan Turing");  
p = new Person("Alonzo Church");
```

► Java

```
Person p = new Person("Alan Turing");  
p = new Person("Alonzo Church");
```

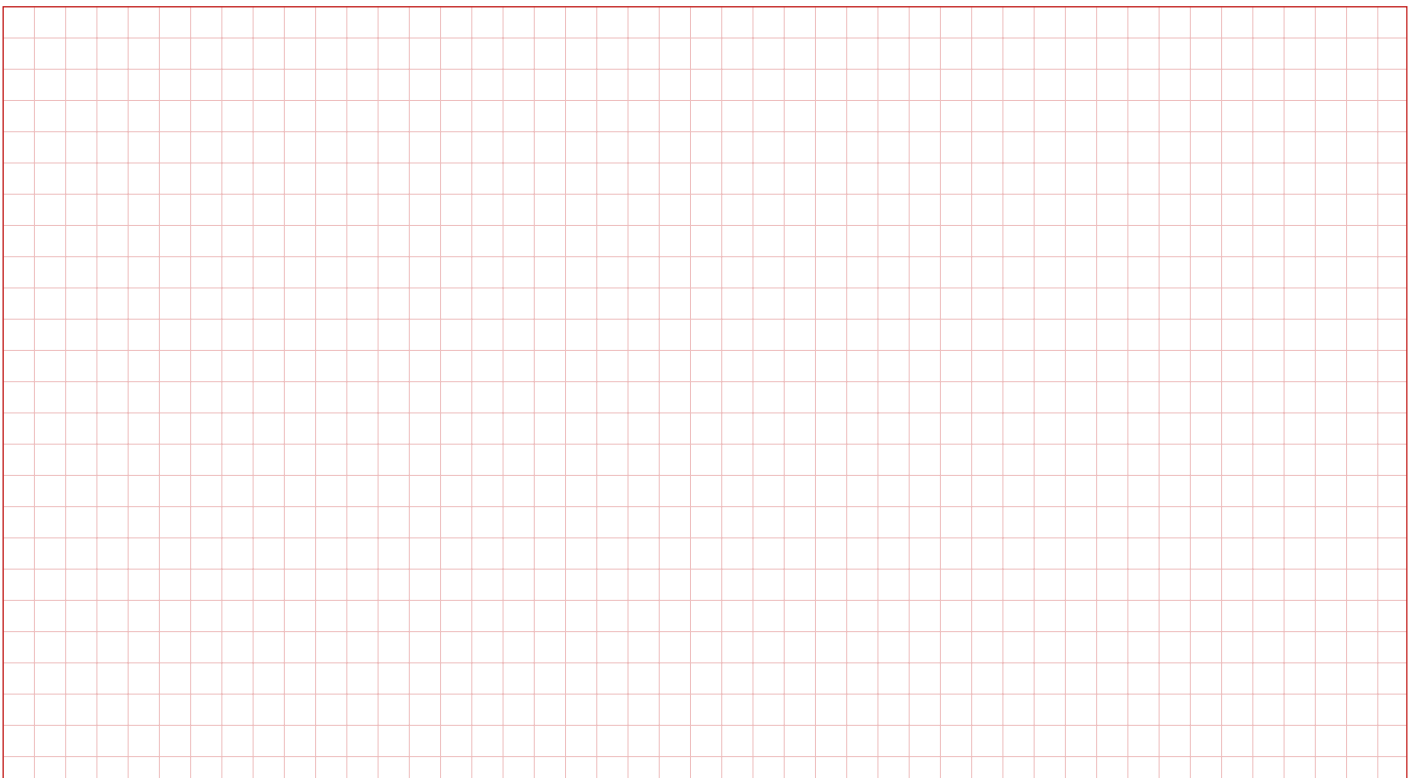
► Problem: Speicherloch

► Java: Garbage Collector



- merkt wer welche Objekte referenziert
- Objekt wird weggeworfen, wenn es nicht mehr referenziert ist
- GC läuft parallel zum Hauptprogramm (Thread)

Notizen



► Vorteile

- ✓ keine Speicherlöcher (prinzipiell, s. u.)
- ✓ reduzierte Programmkomplexität
- ✓ einfacher lesbarer Quellcode

► Nachteile

- ✗ GC braucht Ressourcen (CPU, Speicher)
- ✗ Aufräumzeitpunkt nicht vorhersehbar
- ✗ keine explizite Freigabe möglich

```
p = new Person("Kurt Goedel");  
/* ... */  
p = null; // GC kann Speicher freigeben
```

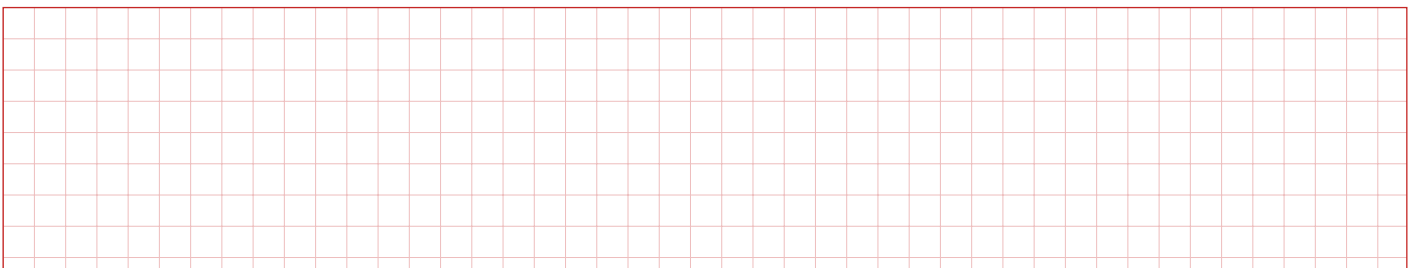
► Achtung

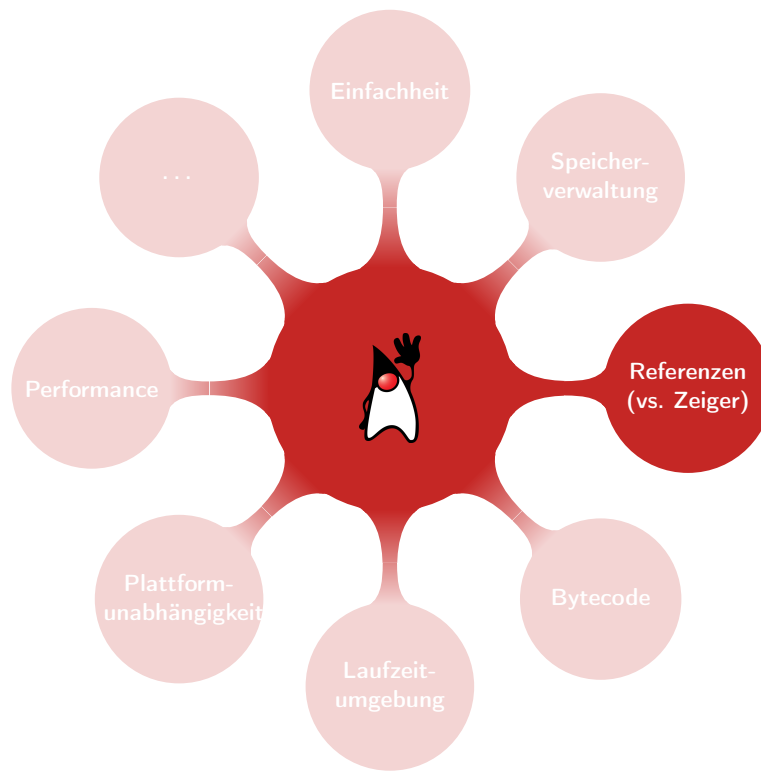
- nur eine Referenz auf ein Objekt, verhindert die Entsorgung
- GC räumt nicht immer sofort auf

```
for (int i = 0; i < 10000; i++)  
    int[] a = new int[1024*1024];
```

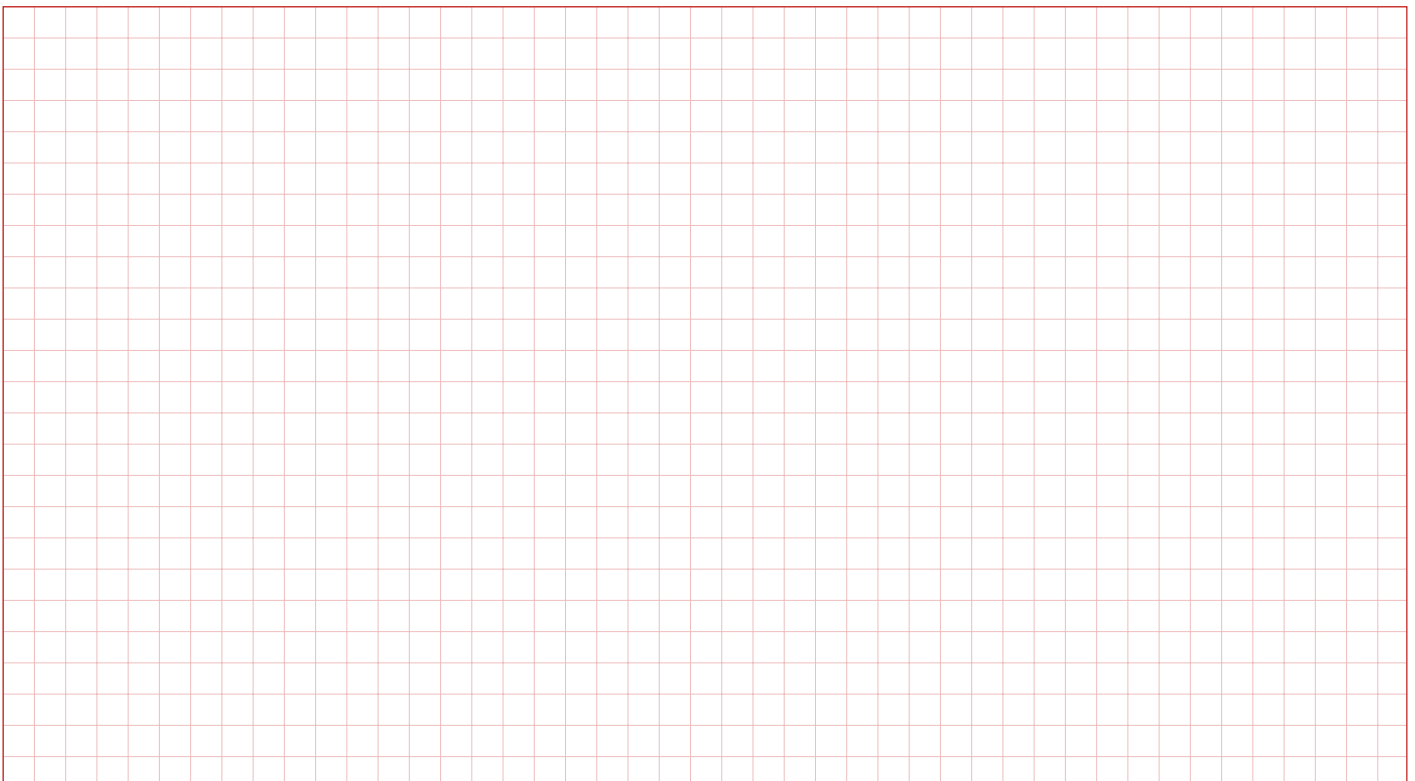
Notizen

- Die nicht Vorhersehbarkeit des Aufräumzeitpunkts ist insbesondere ein Problem für zeitkritische Anwendungen. Bspw. geht ein Spiel davon aus, dass die Framerate (Bilder pro Sekunde) möglichst konstant ist. Das Ausführen des GCs kann aber dazu führen, dass die Framerate zu unvorhersehbaren Zeitpunkten merkbar sinkt.
- Das Setzen einer Referenz auf null entfernt die Instanz nicht sofort, sondern erst nachdem der GC wieder ausgeführt wird (und natürlich nur wenn keine andere Referenz auf das Objekt zeigt). Der GC kann explizit über `System.gc()` ausgeführt werden.
- Ein Objekt selbst kann wiederum Referenzen auf andere Objekte besitzen. So kann ein ganzes Netzwerk an Objekten nur entfernt werden, wenn keine Referenz mehr auf ein Objekt dieses Netzwerks existiert.





Notizen



► C++

```
Person* p = new Person("Alan Turing");
```

p beinhaltet die Speicheradresse

► Java

```
Person p = new Person("Alan Turing");
```

p beinhaltet die **Referenz**

► Java-Referenz beinhaltet...

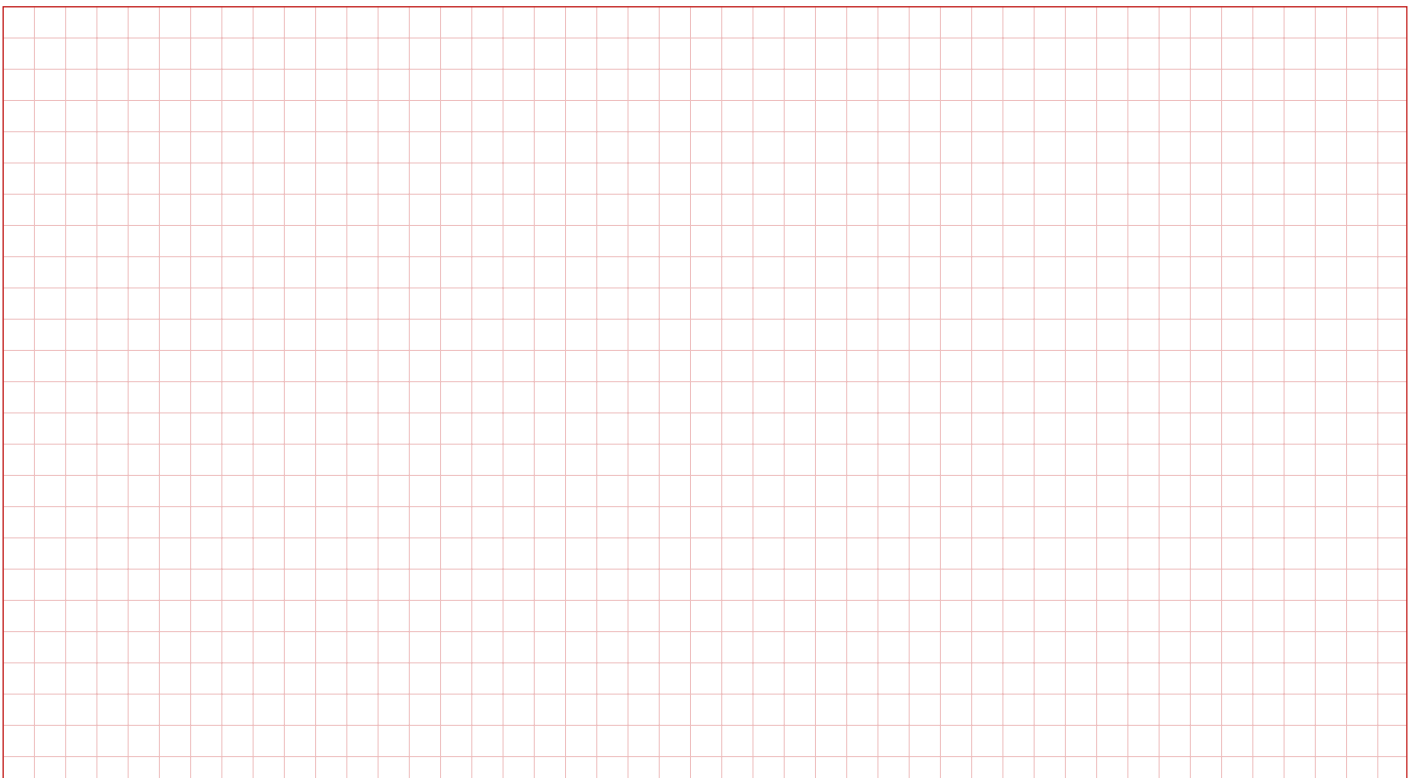
- die eigentliche Speicheradresse
- Typinformationen

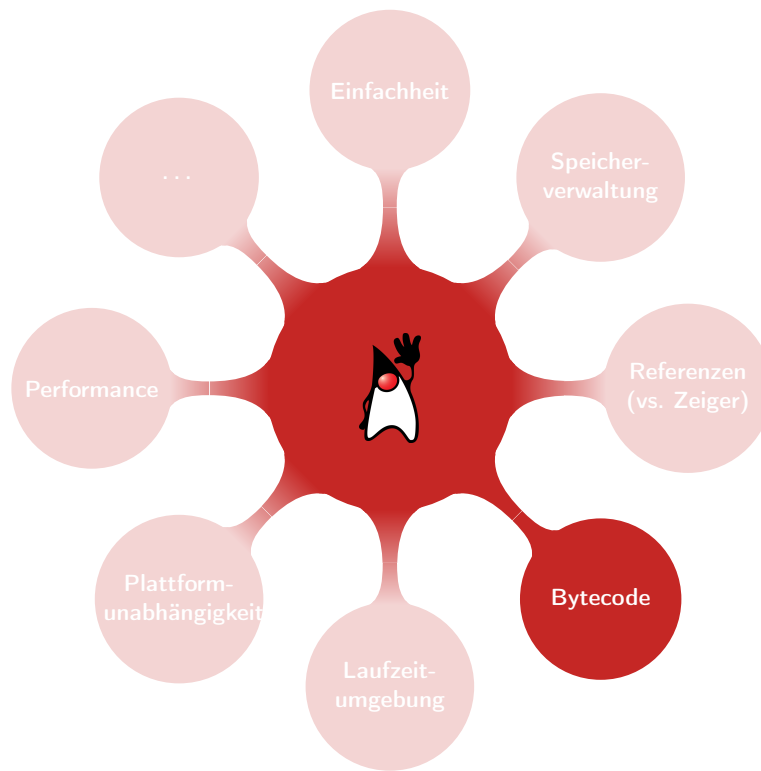
► Vorteile

- ✓ Verschiebung im Speicher möglich
- ✓ Typprüfung
- ✓ Zugriffsprüfung (**private**-Felder)

► **Nachteil**: aufwändigere Dereferenzierung

Notizen





Notizen

Java-Compiler javac

Quellcode (.java-Datei)

```
public int add(int a, int b){  
    return a+b;  
}
```

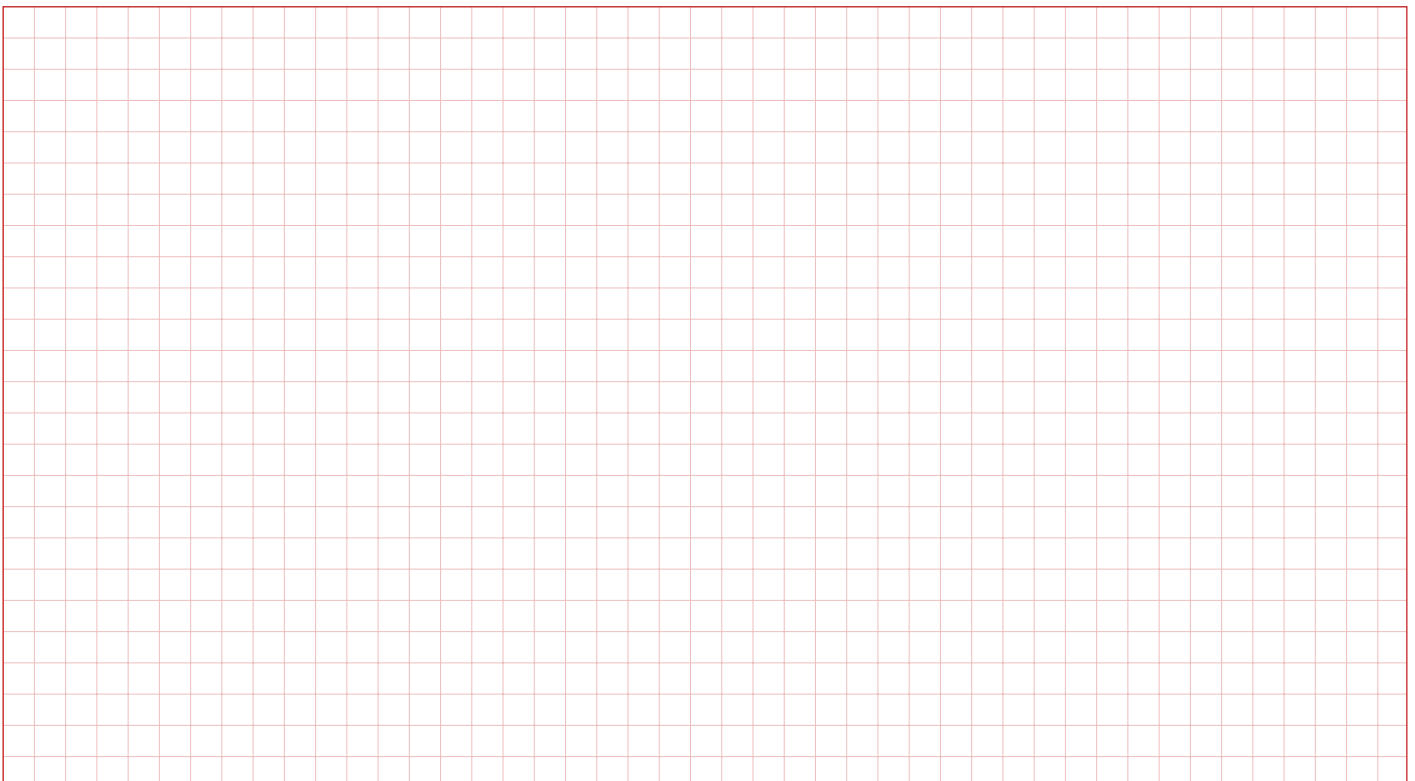
↓ javac

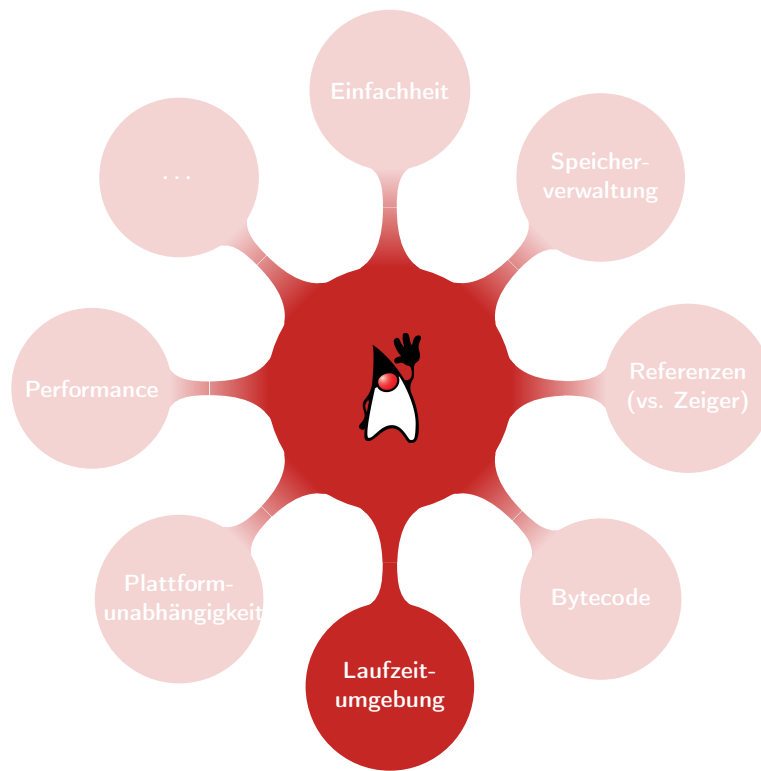
```
public static int add(int, int);  
    iload_0  
    iload_1  
    iadd  
    ireturn
```

Bytecode (.class-Datei)

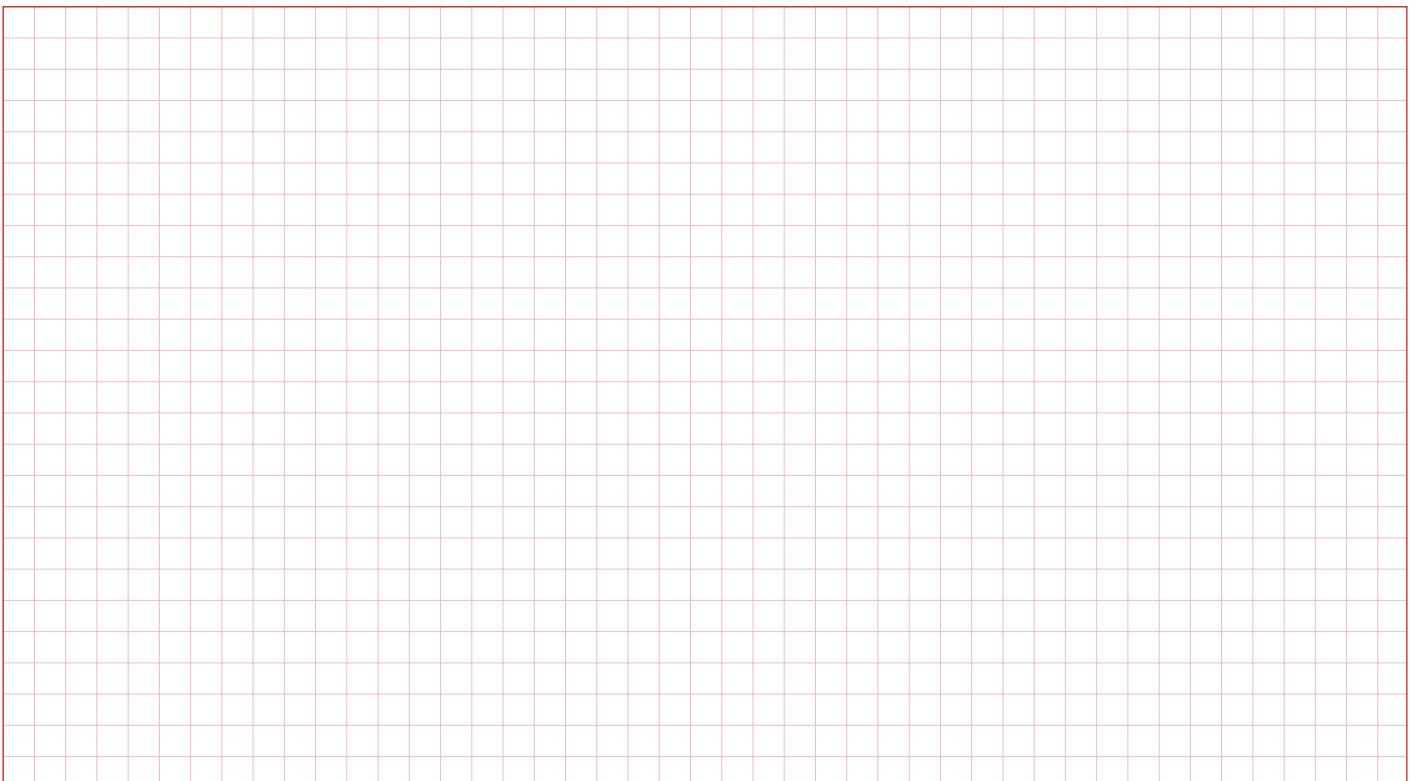
Bytecode: Instruktionen für virtuelle Java-Maschine

Notizen

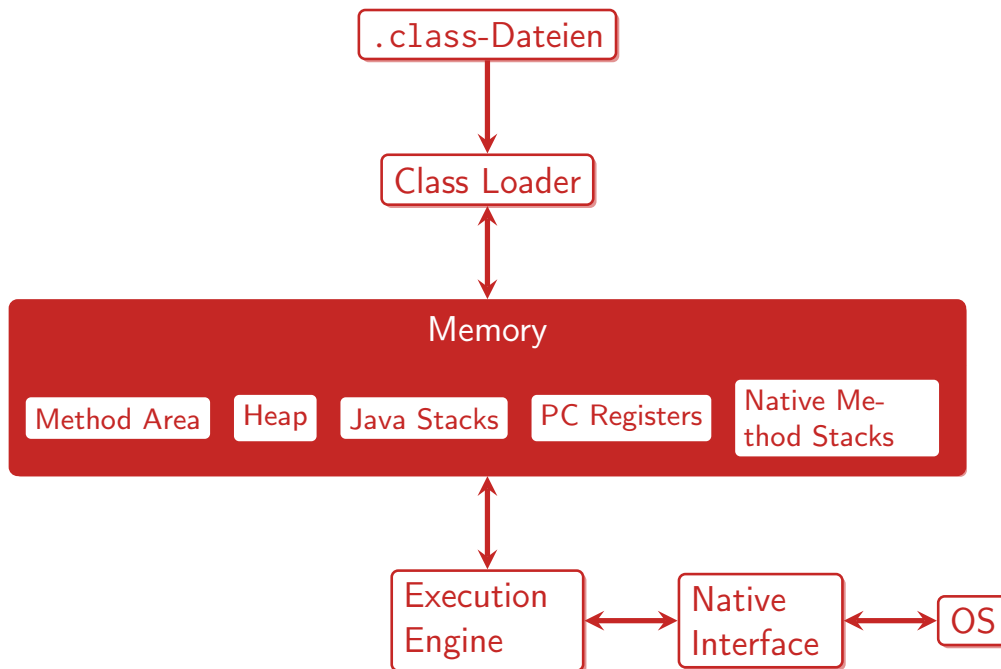




Notizen



Java Laufzeitumgebung: „Java Virtual Machine“



Notizen

Quelle Tim Lindholm u. a. *The Java Virtual Machine Specification*.

<https://docs.oracle.com/javase/specs/jvms/se12/html/index.html>. 2019

Class Loader lädt, bindet und initialisiert Java-Klassen

Method Area Daten der Klassen, wie Konstanten, Felder, sowie Code der Methoden und Konstrukturen

Heap Heap-Speicher, hält die Laufzeitdaten (Instanzen, Arrays, etc.)

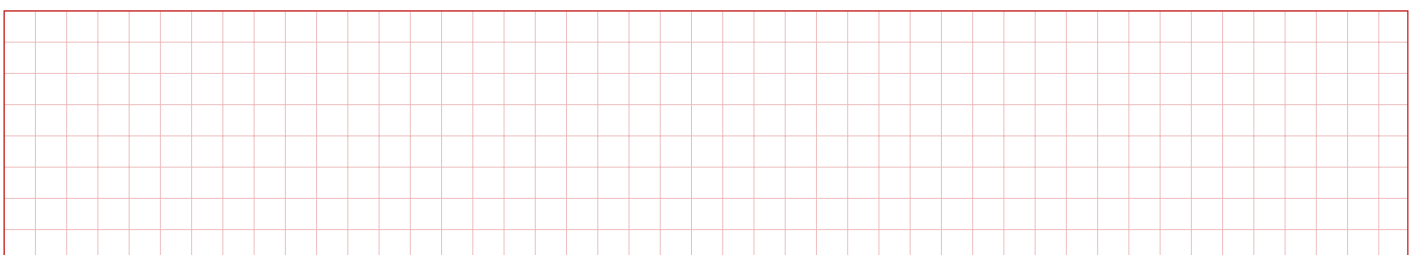
Java Stacks Laufzeit-Stacks der einzelnen Java-Threads

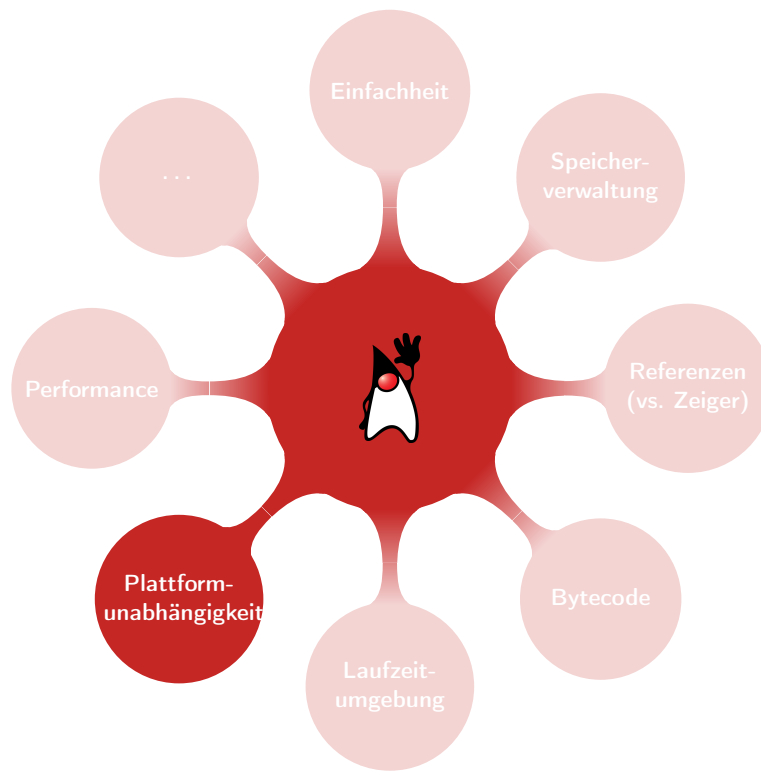
PC Registers „Program Counters“ der einzelnen Java-Threads (aktuell auszuführende Bytecode-Instruktion)

Native Method Stacks Laufzeit-Stacks der nativen Methoden (C/C++)

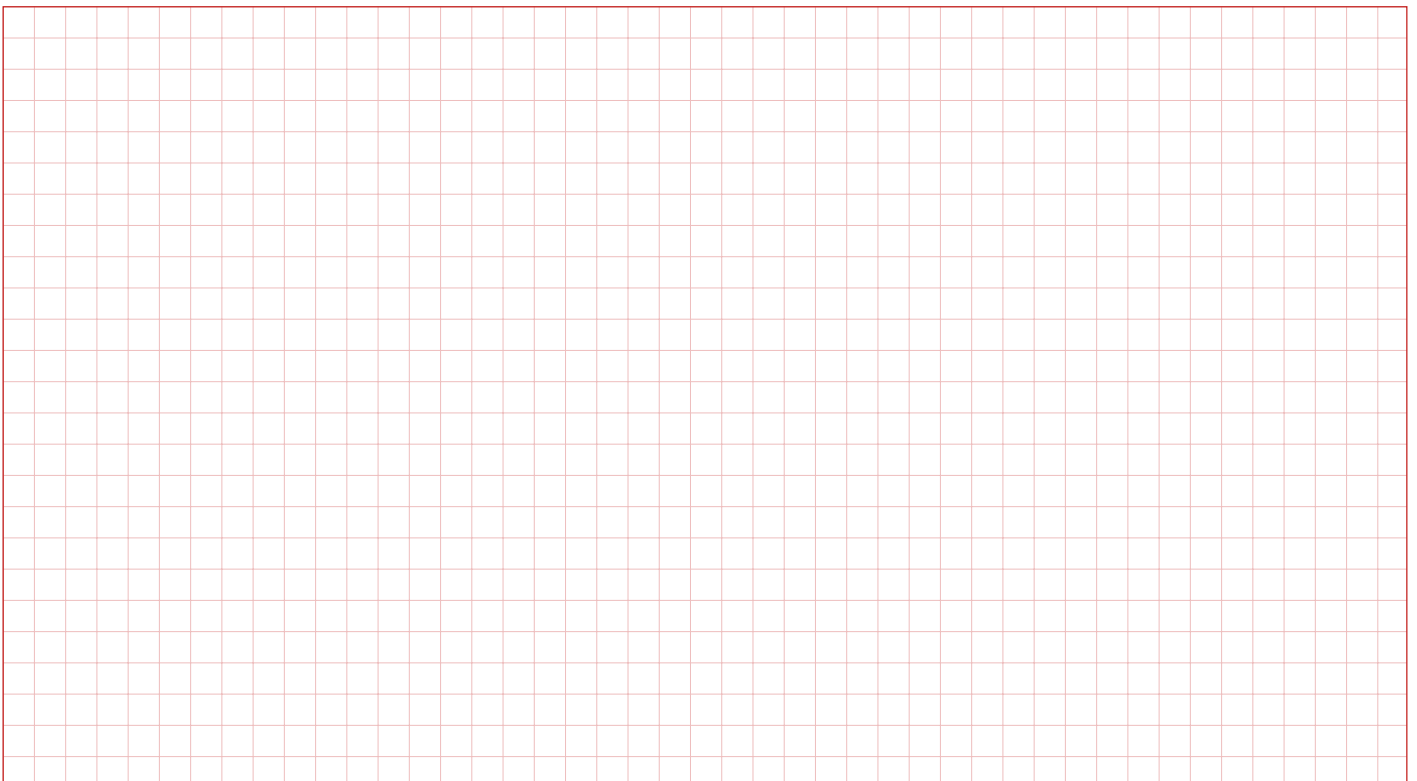
Execution Engine führt den Java-Code aus (Interpretation)

Native Interface Schnittstelle zu nativen Bibliotheken und zum Betriebssystem

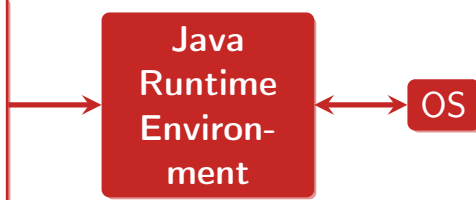




Notizen

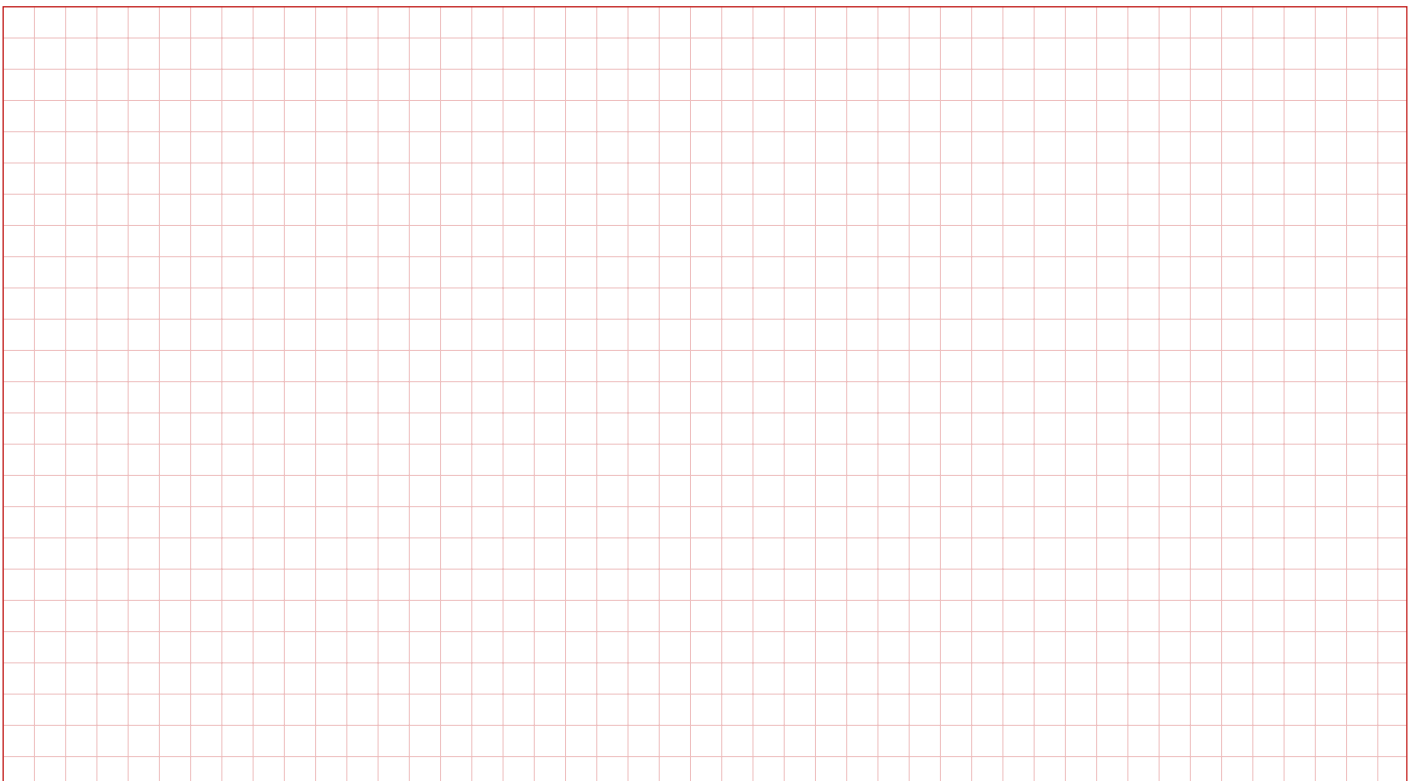


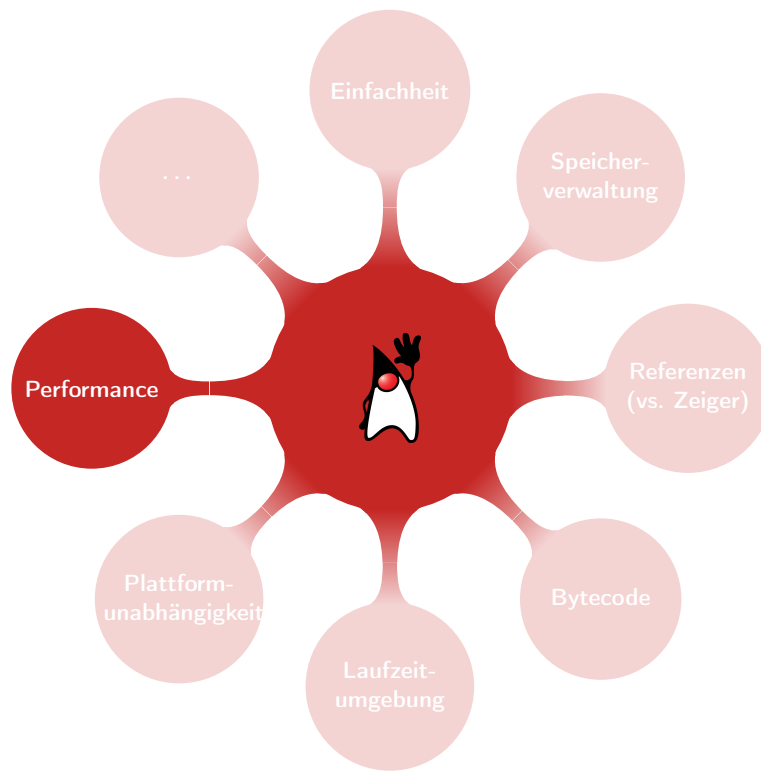
```
public static int add(int, int);  
  iload_0  
  iload_1  
  iadd  
  ireturn
```



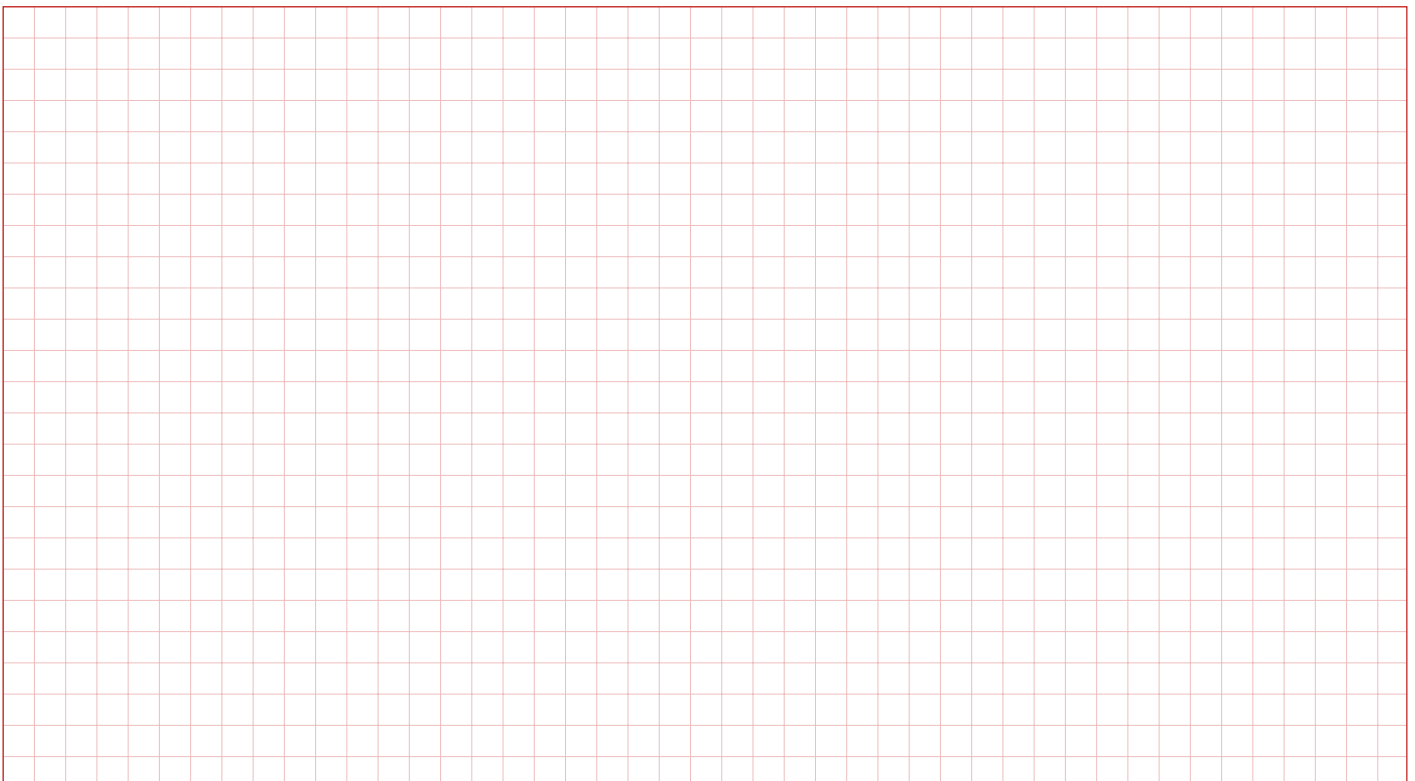
- ▶ Java Runtime Environment (JRE)
 - ▶ interpretiert Bytecode in Java Virtual Machine
 - ▶ reicht Aufrufe an Betriebssystem weiter
- ▶ JRE bildet eine Abstraktionsschicht
- ▶ verschiedene JRE-Implementierungen möglich
- ▶ Bytecode ist **plattformunabhängig**

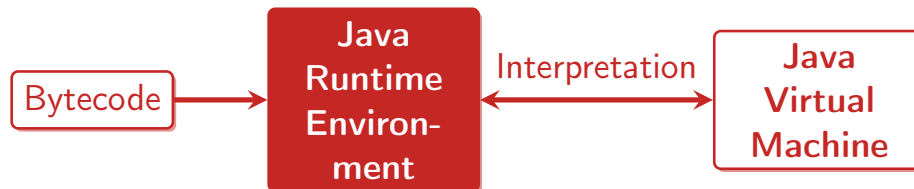
Notizen





Notizen

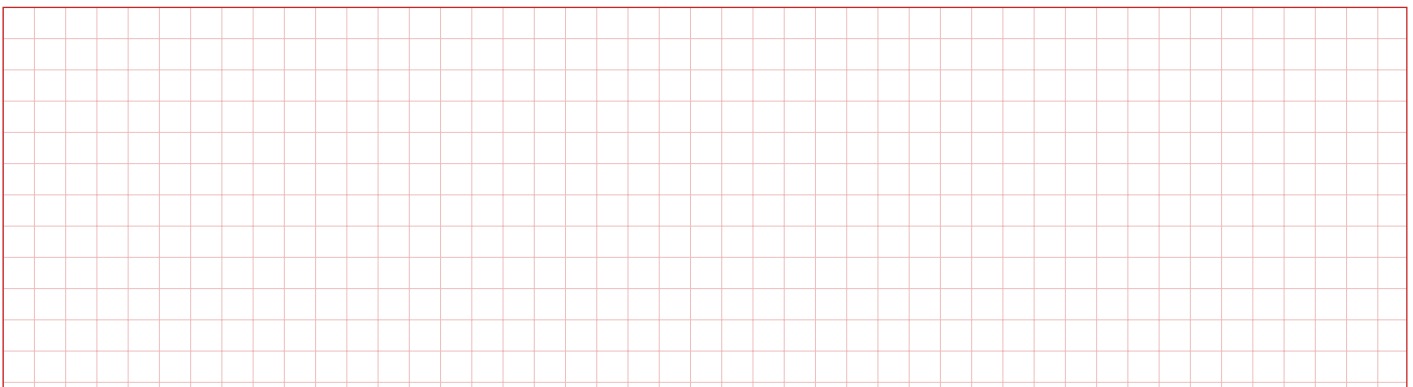


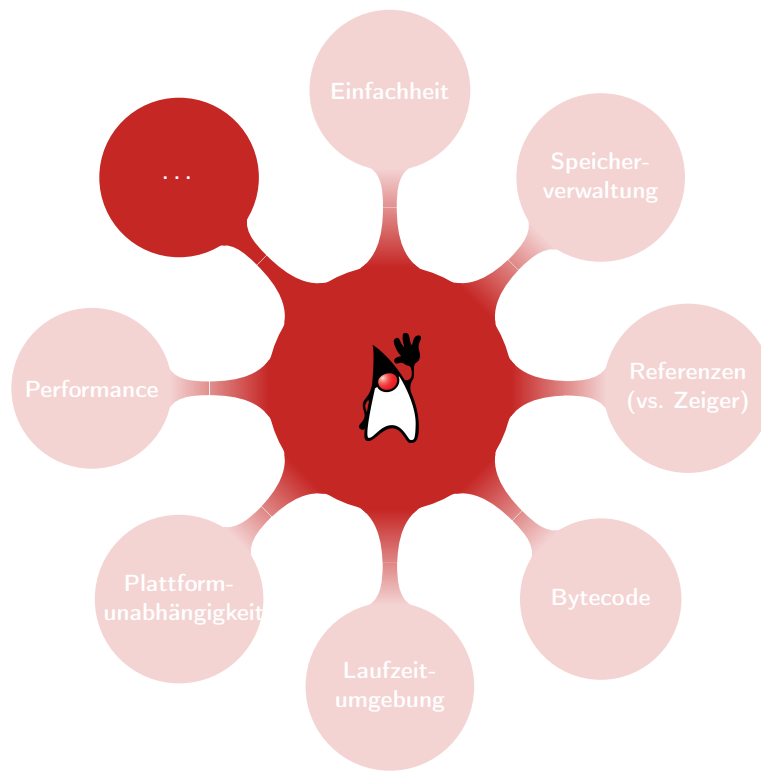


- ▶ Interpretation
 - ▶ Erkennen
 - ▶ Dekodieren
 - ▶ Ausführen
- ▶ Problem: langsam!
- ▶ Just-in-Time-Compilation
 - ▶ Ausführungseinheit erkennt „Hotspots“ (oft ausgeführte Programmteile)
 - ▶ Übersetzung in Maschinencode zur Laufzeit
- ▶ Vorteile
 - ✓ Kontextinformationen (Programm, Prozessor, etc.)
 - ✓ Code-Optimierung zur Laufzeit
- ▶ Nachteile
 - ✗ Übersetzung kostet Zeit
 - ✗ komplexe JRE-Implementierung

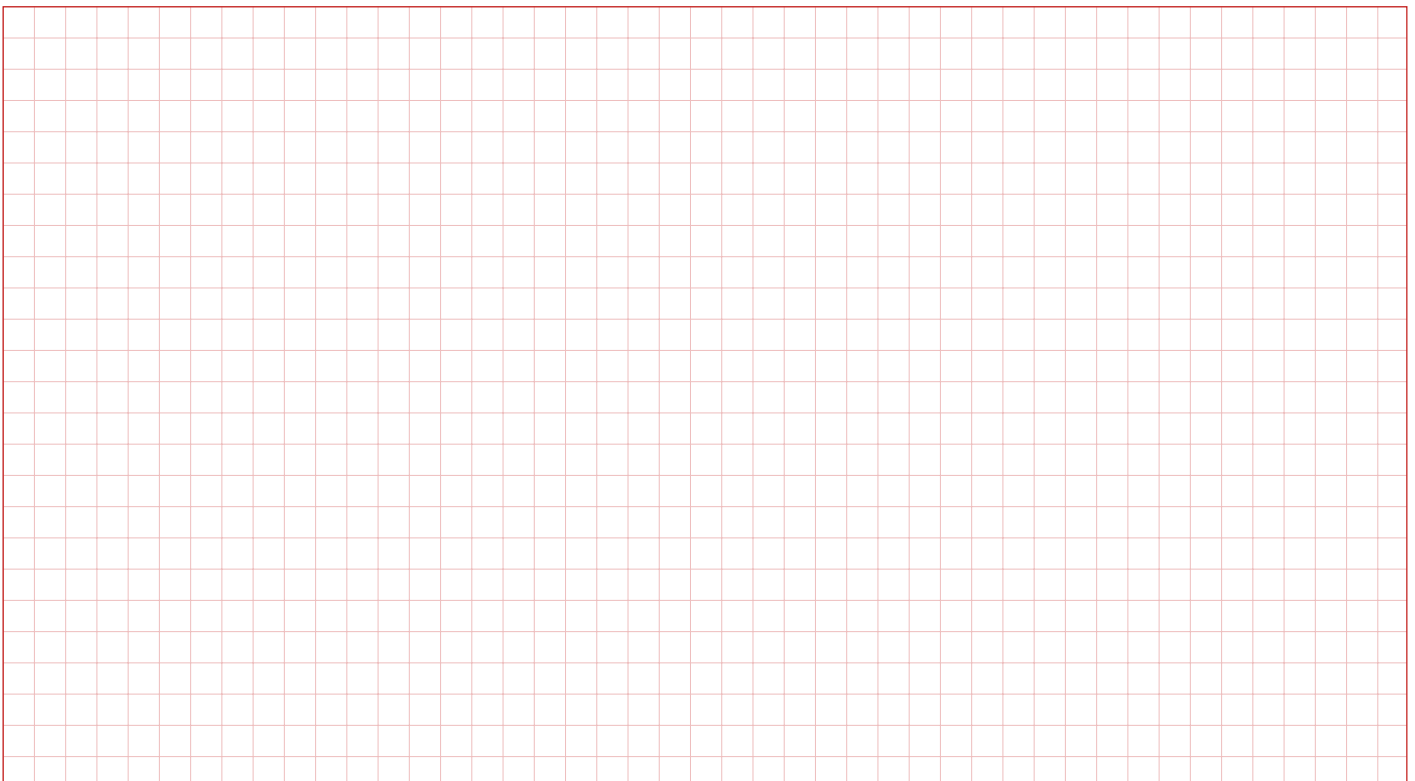
Notizen

- Interessante Artikel zum Thema Java-Performanz
 - https://en.wikipedia.org/wiki/Java_performance
 - <https://benchmarksgame-team.pages.debian.net/benchmarksgame/fastest/java-gpp.html>
- Die JRE übersetzt nicht sofort alle Programmteile, da dies zu lange dauern würde. Stattdessen beginnt sie mit Interpretation und entscheidet mit Hilfe von Heuristiken welche oft ausgeführten Programmteile (sog. „Hotspots“, daher der Name Hotspot-Engine) übersetzt werden sollen.



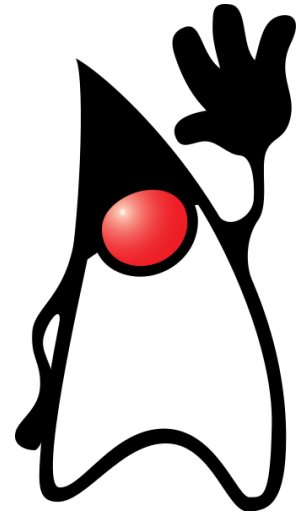


Notizen



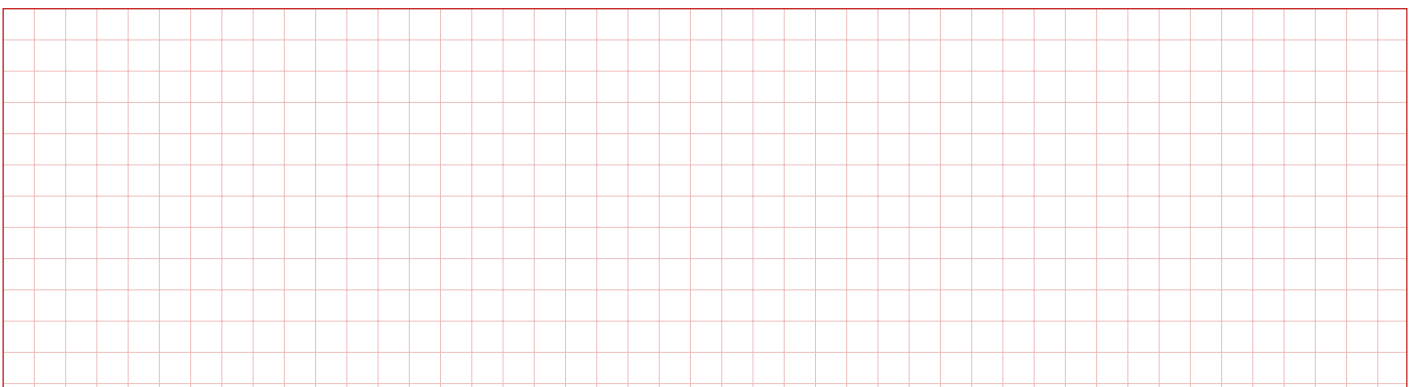
Java ist. . .

- ▶ **objektorientiert**
 - ▶ rein objektorientiert, bis auf primitive Typen (Zahlen, Character)
 - ▶ keine Mehrfachvererbung
 - ▶ Interfaces
 - ▶ objektorientierte Ausnahmebehandlung
- ▶ (prinzipiell) **sehr sicher**
 - ▶ JRE bildet eine „sandbox“
 - ▶ starke Typ- und Zugriffsprüfung
 - ▶ konfigurierbarer Security Manager für Datei-/Netzwerkzugriffe
- ▶ **open source** (OpenJDK)
- ▶ **konservativ** in der Weiterentwicklung
 - ▶ funktionale Konstrukte (Streams, Lambdas) erst ab Java 8
 - ▶ überschaubarer Sprachkern (vgl. C++)



Notizen

- Java bietet, im Vergleich zu C/C++ und C#, z.B. keine structs
- Java-Interfaces ist vergleichbar mit einer rein abstrakten Klasse, d.h. alle Methoden sind abstrakt. Durch Interfaces bietet Java eine weitere Abstraktions- und Strukturierungsmöglichkeit und vermeidet Probleme der Mehrfachvererbung (s. „deadly diamond of death“-Problem in C++). Eine Java-Klasse kann mehrere Interfaces implementieren und Interfaces können von anderen Interfaces erben.
- OpenJDK: <https://openjdk.java.net/>

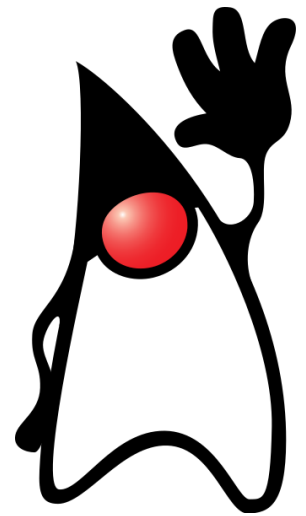


Java eignet sich für. . .

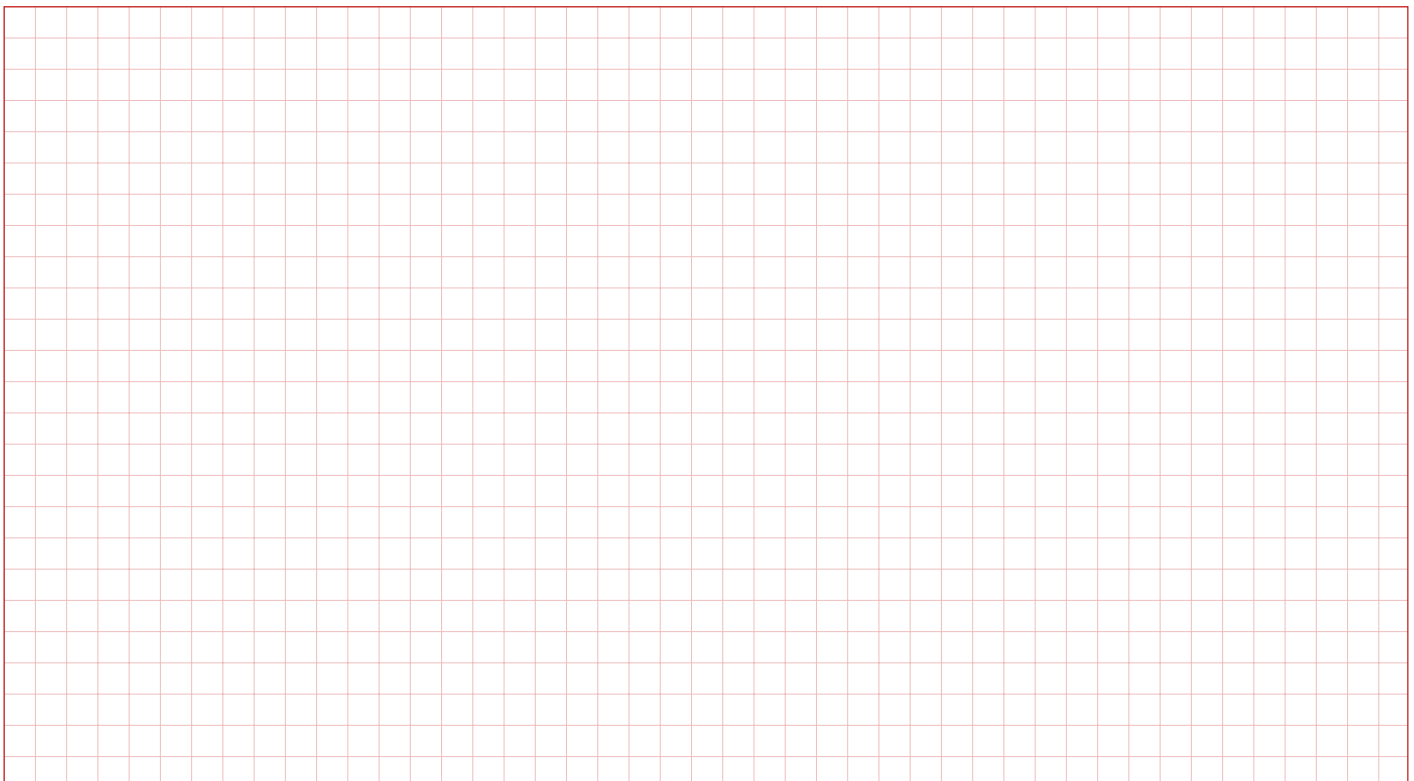
- ✓ **Webserver**-Anwendungen
- ✓ **plattformunabhängige** (UI-)Anwendungen
- ✓ das **Erlernen** objektorientierter Programmierung

Java eignet sich **nicht** für. . .

- ✗ **hardwarenahe** Entwicklung: Zugriff auf USB, Hardwareschnittstellen
- ✗ **betriebssystemnahe** Entwicklung: Kernel-Erweiterung, Systemschnittstellen
- ✗ **„low-level“**-Anwendungen: Netzwerkprotokolle (ICMP), (erweiterte) Konsolenein-/ausgaben



Notizen



Inhalt

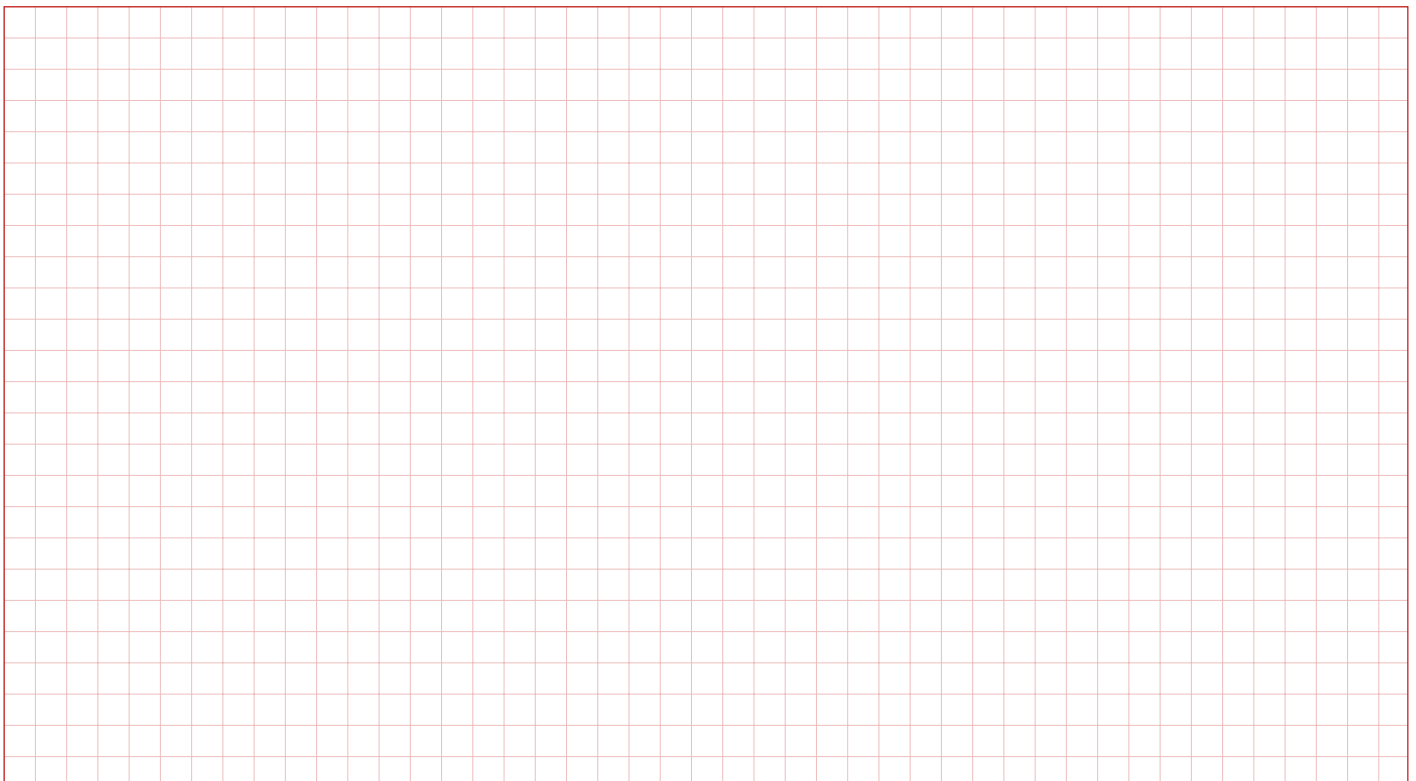
Java-Plattformen und Implementierungen

Plattformen

Implementierungen

Java als Plattform für Sprachen

Notizen

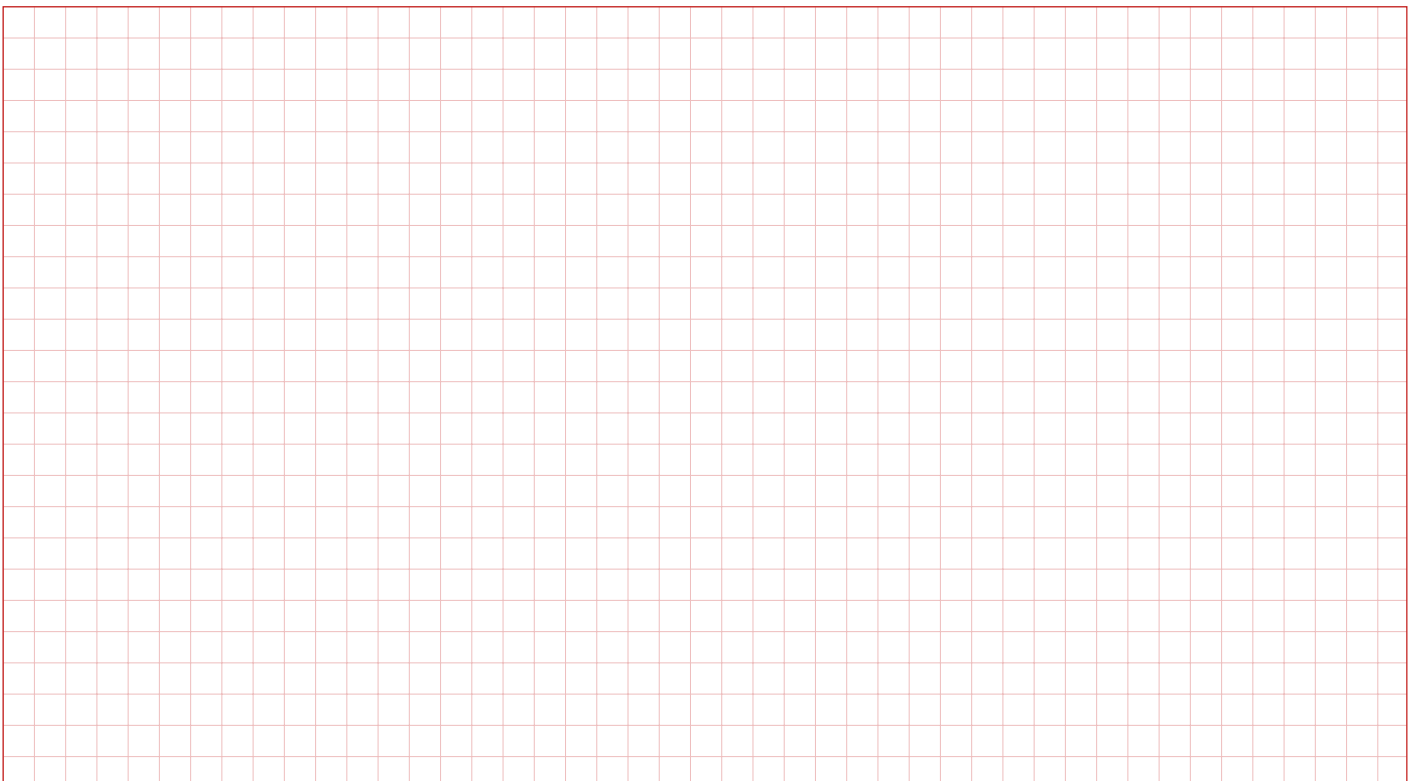


Inhalt

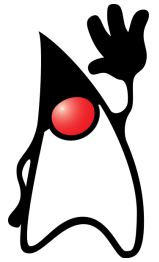
Java-Plattformen und Implementierungen

Plattformen

Notizen



- ▶ Java SE (Standard Edition)
 - ▶ Interpreter, Compiler, Debugger
 - ▶ Datenbankschnittstelle (JDBC)
 - ▶ UI-Entwicklung (AWT, Swing)
 - ▶ Datenströme: Dateien, Netzwerk
 - ▶ ...
- ▶ Java ME (Micro Edition): eingebettete System, Smartphones
- ▶ Java Card: JVM auf Chipkarten (z.B. SIM-Karten)
- ▶ Java EE/Jakarta
 - ▶ früher Java EE (Enterprise Edition)
 - ▶ Webseiten/-dienste (JSP, JSF)
 - ▶ JavaMail API
 - ▶ Applikationsserver Glassfish
- ▶ Real-Time Java
 - ▶ zeitkritische Anwendungen (z.B. Sensorverarbeitung)
 - ▶ Garbage Collector kann gesteuert werden
 - ▶ (manuellere) Speicherverwaltung



Notizen

- Java ME hat über die Jahre stark an Bedeutung verloren. Vor allem auch wegen der übermächtigen Konkurrenz seitens Google (Android) und Apple (iOS).
- Android verwendet zur Entwicklung ebenfalls Java. Allerdings mit einem proprietären Bytecode-Format und einer eigenen virtuellen Maschine namens *Dalvik*.
- JSP („Java Server Pages“) und das neuere JSF („Java Server Faces“) werden zur Erstellung von dynamischen Webseiten verwendet.
- Bei Glassfish handelt es sich um die Referenzimplementierung. Weiter verbreitet ist Apache's Tomcat (s. <http://tomcat.apache.org/index.html>)
- Real-Time Java setzt ein Betriebssystem voraus, das Echtzeitverarbeitung unterstützt.



Inhalt

Java-Plattformen und Implementierungen

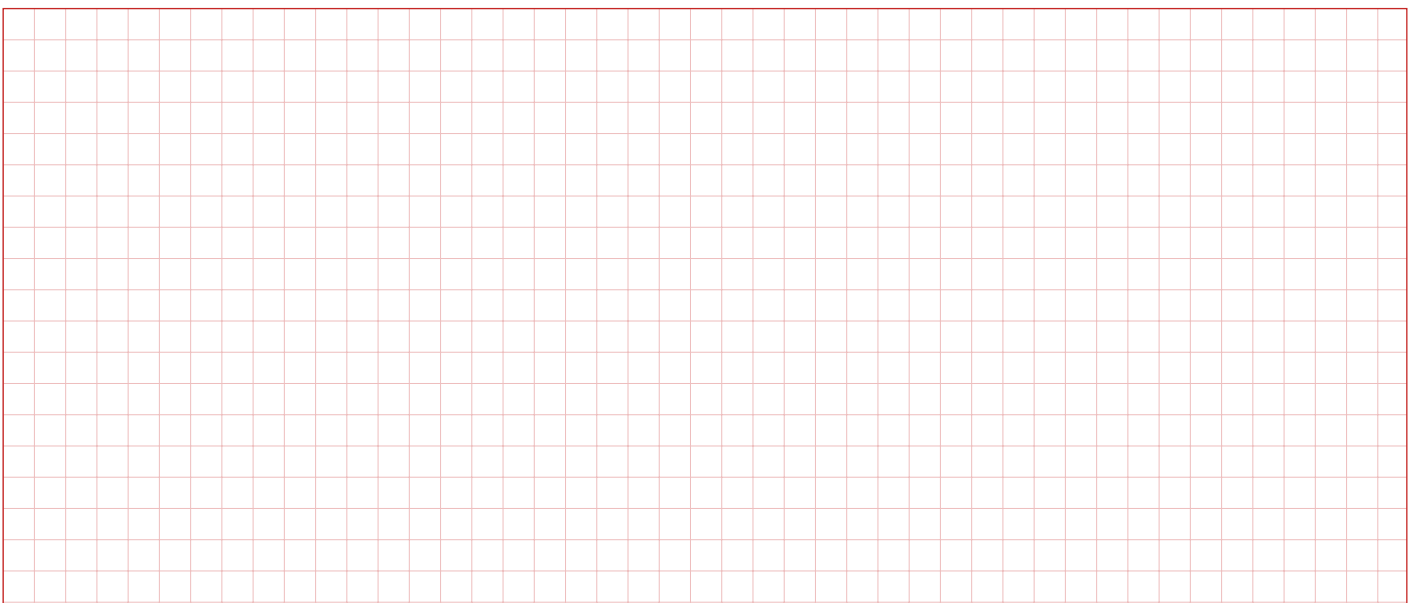
29

Notizen

- ▶ Oracle JDK
 - ▶ <https://www.oracle.com/java/technologies/javase-downloads.html>
 - ▶ alle drei Jahre **LTS-Version** („long time support“)
 - ▶ **Achtung**: nur für „development, testing, protoyping und demonstration purposes“
 - ▶ sonst **monatliche Lizenzgebühren**
- ▶ OpenJDK
 - ▶ Oracle <https://openjdk.java.net/>
 - ▶ GPL (v2)
 - ▶ mehrere Anbieter

Notizen

- Eine LTS-Version wird weiterhin mit Sicherheits-Updates und Bugfixes gepflegt.
- Eine Liste mit OpenJDK-Anbietern findet man auf <https://en.wikipedia.org/wiki/OpenJDK>.

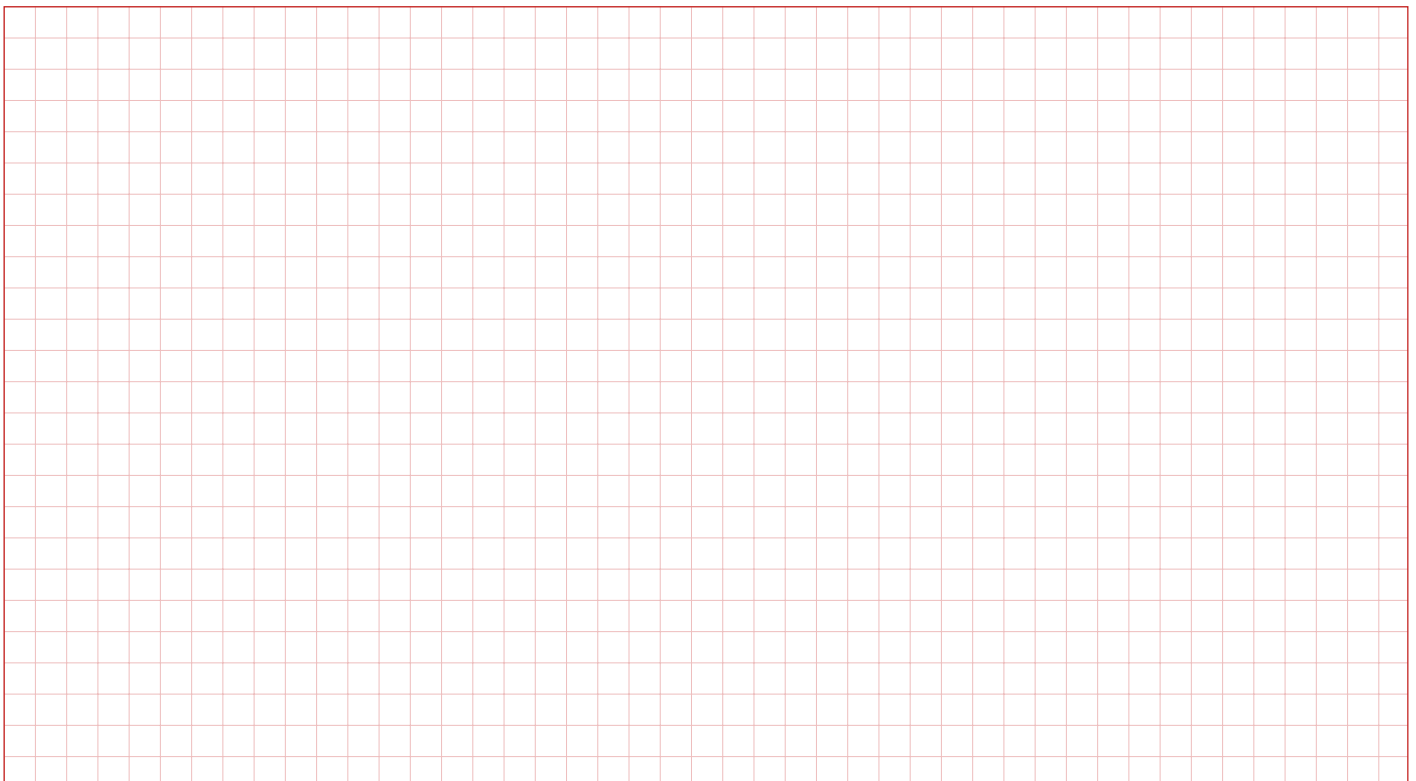


Inhalt

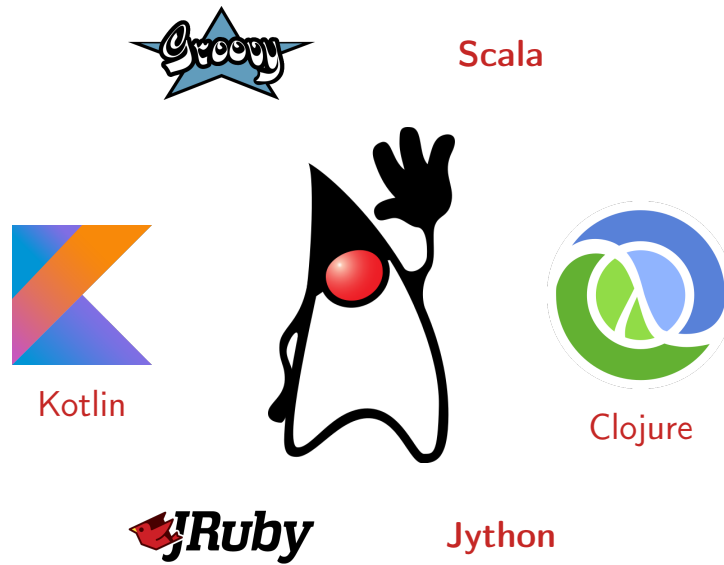
Java-Plattformen und Implementierungen

Java als Plattform für Sprachen

Notizen



Java als Plattform für Sprachen





Notizen

Clojure eine funktionale LISP (<http://clojure.org>); Clojure Logo, Tom Hickey und Rich Hickey, Public Domain

Scala funktionale und objektorientierte Sprache (<https://www.scala-lang.org/>)

Groovy dynamische Sprache, die auch als Skriptsprache verwendet werden kann (<https://groovy-lang.org/>); Logo  Groovy Logo by Zorak1103 licensed under  CC BY-SA 3.0

Kotlin statisch getypte Programmiersprache zur Entwicklung auf mehreren Plattformen, wie z.B. Android (<https://kotlinlang.org/>); Logo by  JetBrains

JRuby Implementierung der Programmiersprache Ruby auf der JVM (<https://www.jruby.org/>); Logo  <https://github.com/jruby/collateral> by Tony Price licensed under  CC BY-ND 3.0

Jython Implementierung der Programmiersprache Python auf der JVM (<https://www.jython.org/>)



Inhalt

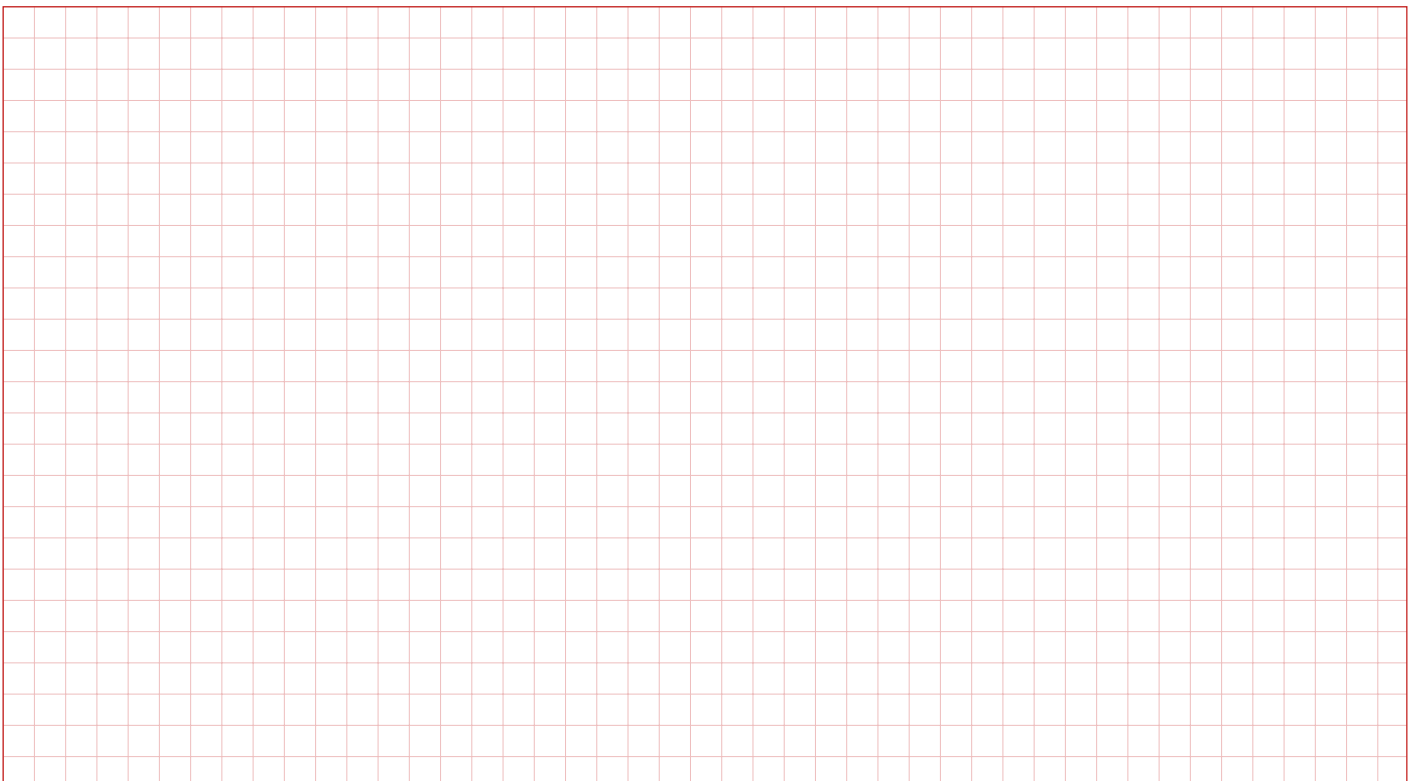
„Hello World“

Notizen

```
public class HelloWorld{  
    public static int add(int a, int b){  
        return a + b;  
    }  
    public static void main(String[] args){  
        System.out.println("Hello World!");  
    }  
}
```

📄 HelloWorld.java

Notizen



- **Voraussetzung:** installiertes JDK

```
$ javac -version  
javac 1.8.0_144
```

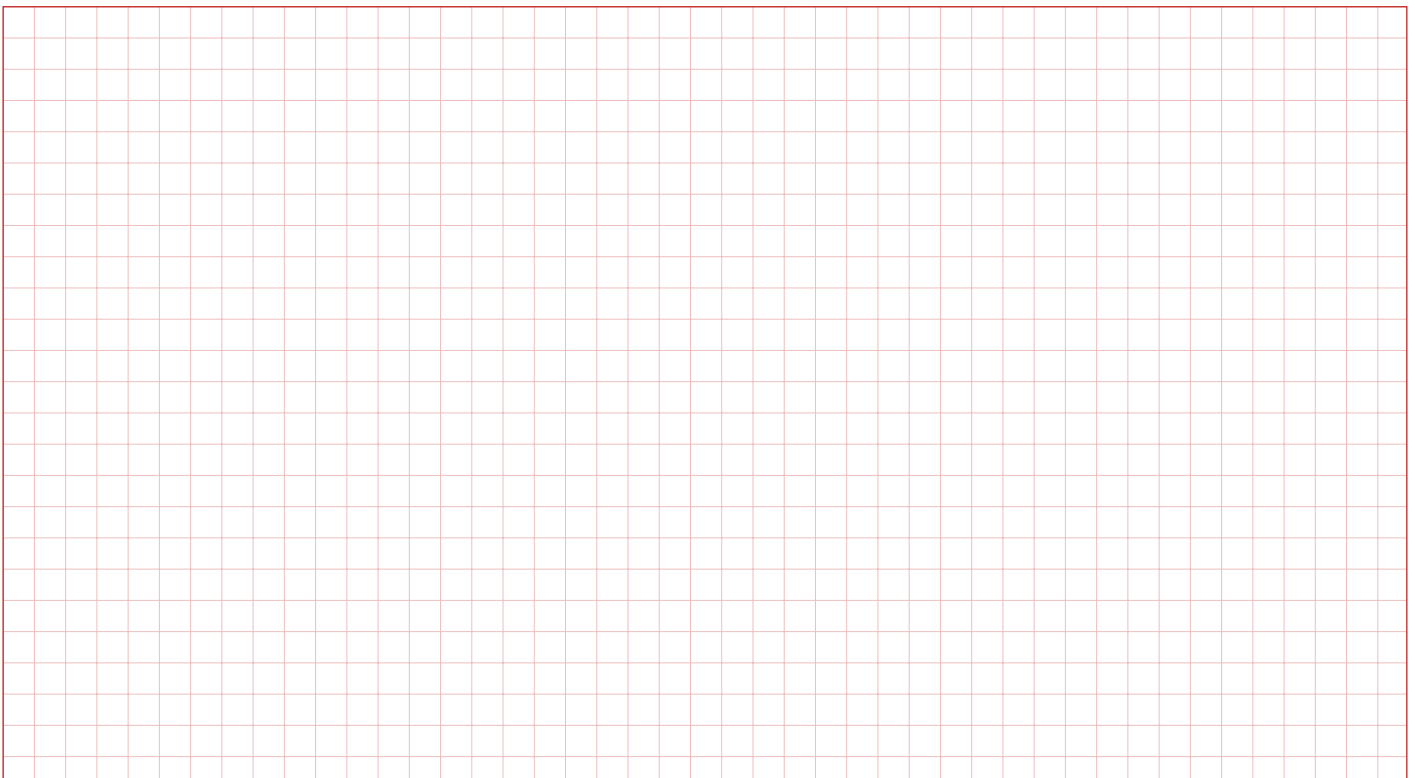
- **Übersetzen**

```
$ cd Examples/src/main/java  
$ ls  
HelloWorld.java  
$ javac HelloWorld.java  
$ ls  
HelloWorld.class    HelloWorld.java
```

- **Ausführen**

```
$ java HelloWorld  
Hello World!
```

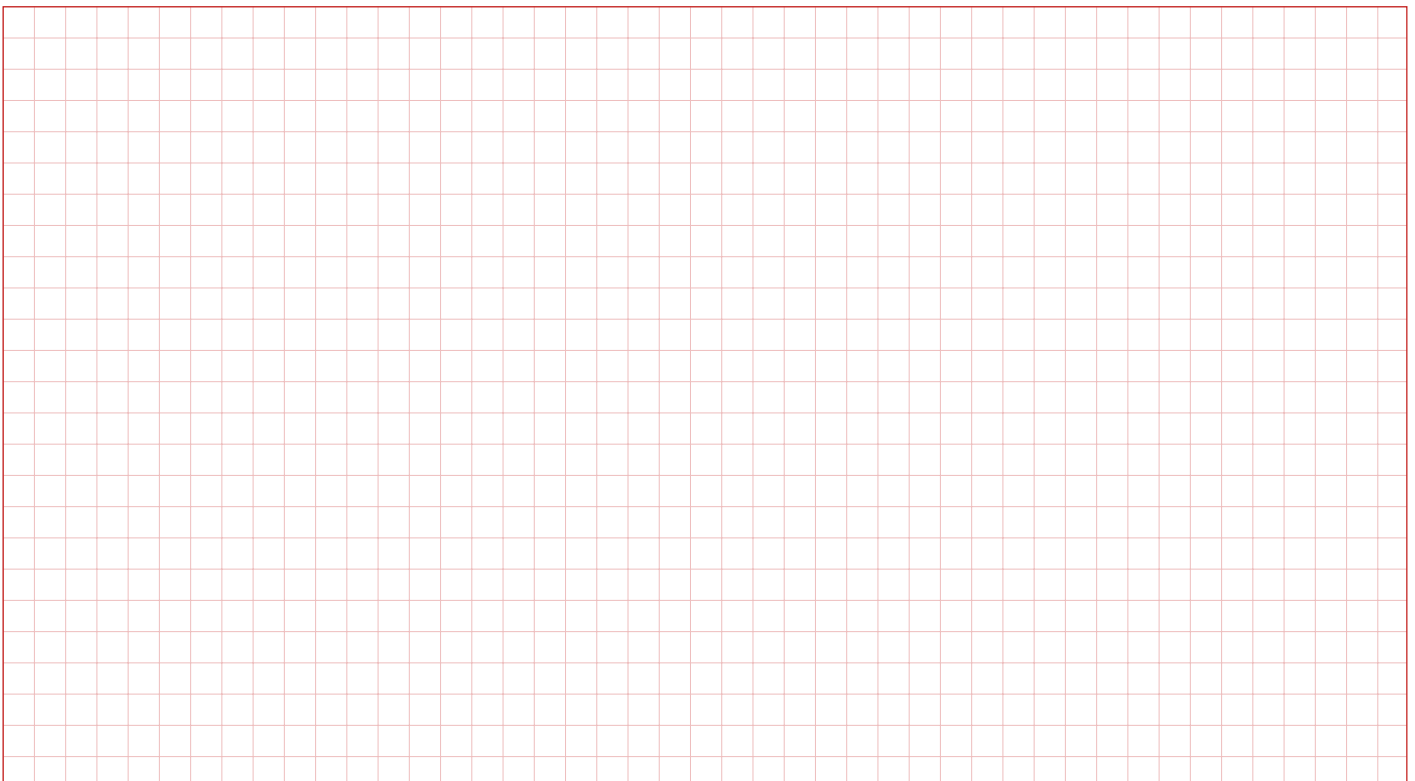
Notizen



Bytecode anzeigen

```
$ javap -c HelloWorld.class
Compiled from "HelloWorld.java"
public class HelloWorld {
    public HelloWorld();
        0: aload_0
        1: invokespecial java/lang/Object."<init>":()V
        4: return
    public static int add(int, int);
        0: iload_0
        1: iload_1
        2: iadd
        3: ireturn
    /* ... */
}
```

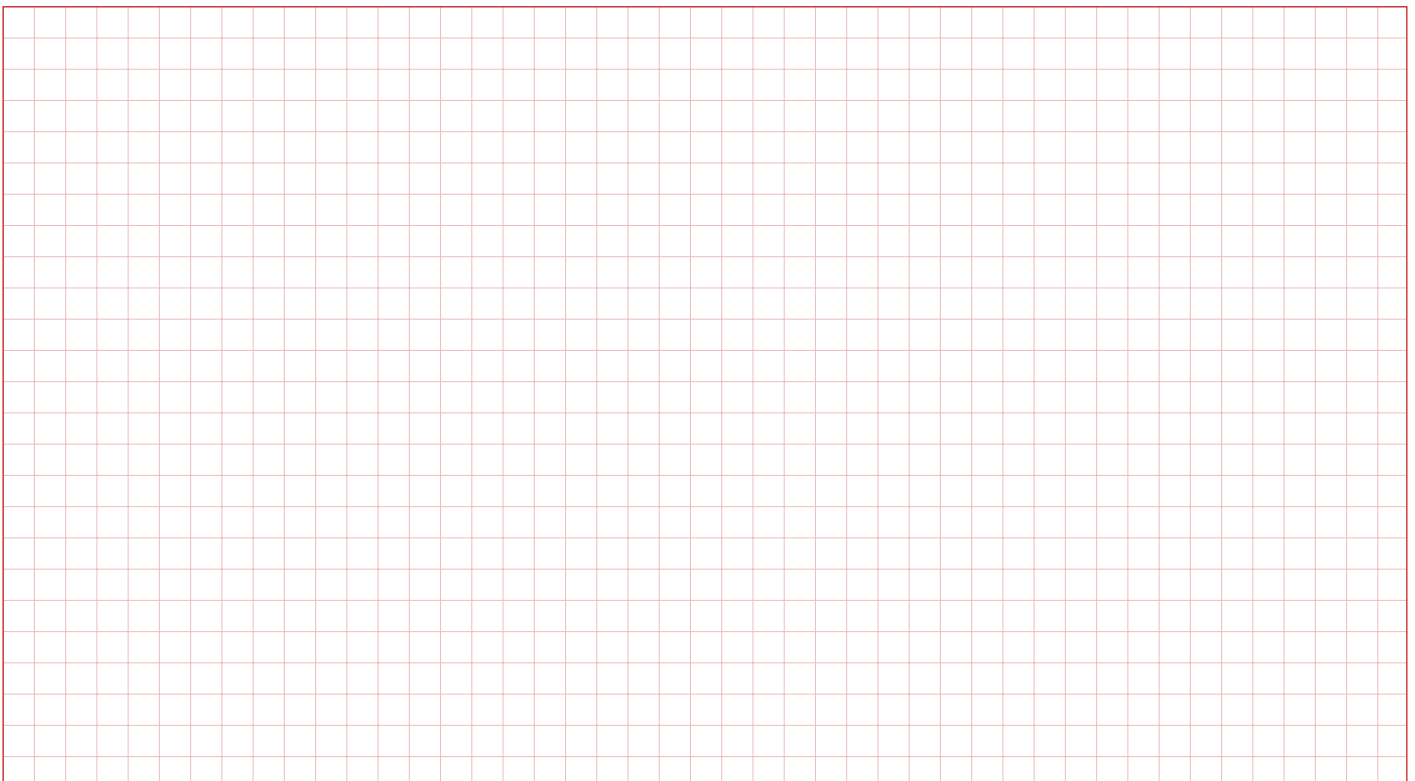
Notizen



Bytecode (Teil 2)

```
/* ... */  
public static void main(java.lang.String[]);  
  0: getstatic java/lang/System.out:Ljava/io/PrintStream;  
  3: ldc "Hello World!"  
  5: invokevirtual  
    java/io/PrintStream.println:(Ljava/lang/String;)V  
  8: return  
}
```

Notizen



Inhalt

Entwickeln mit Java

- IDEs
- Build Tools
- jshell

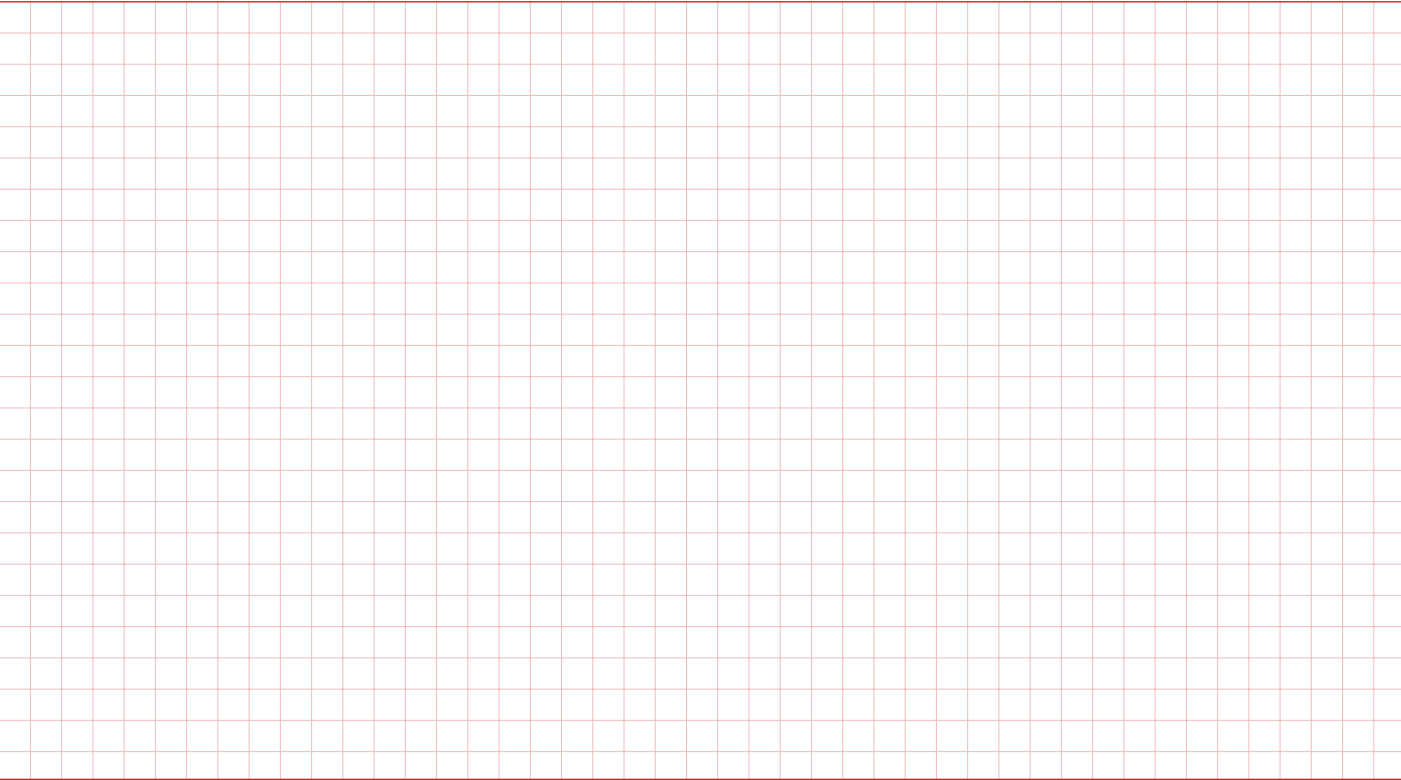
Notizen

A large rectangular area filled with a light gray grid pattern, intended for taking notes. The grid consists of small squares, with a slightly larger margin at the top for a header.

Inhalt

Entwickeln mit Java
IDEs

Notizen



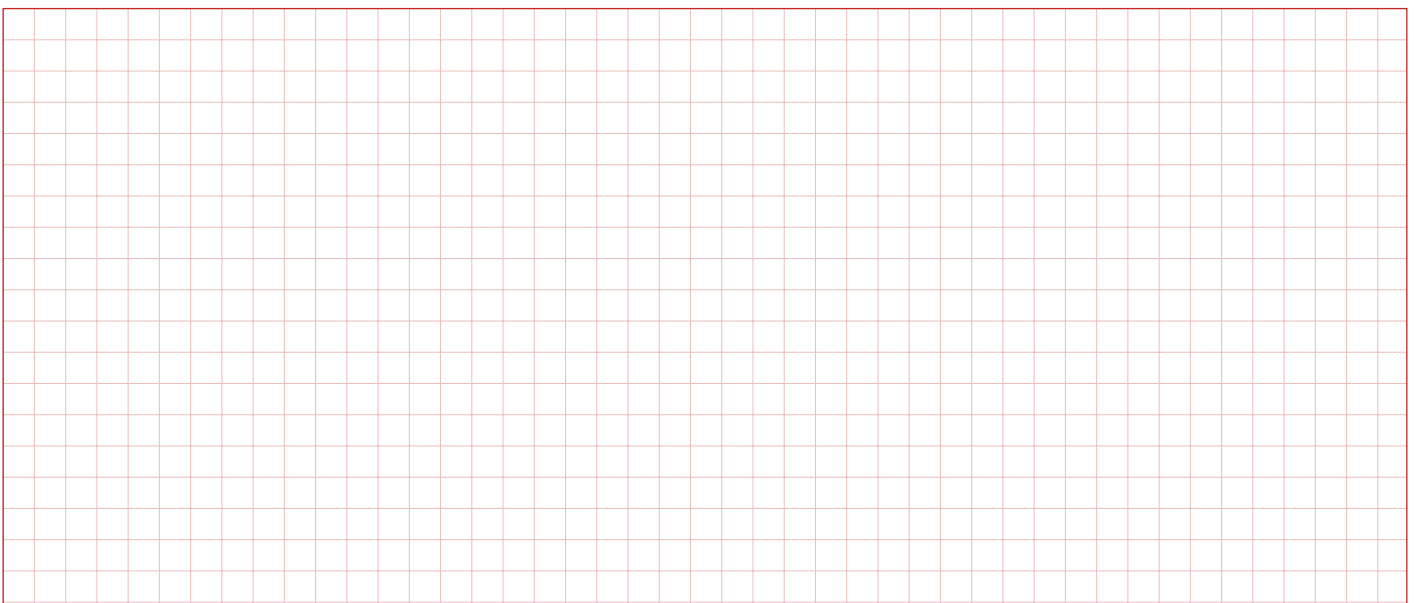
Entwicklungsumgebungen (IDEs):

- ▶ Eclipse
 - ▶ <https://www.eclipse.org/ide/>
 - ▶ frei (Eclipse Foundation)
- ▶ NetBeans
 - ▶ <https://netbeans.apache.org/>
 - ▶ frei (Apache Software Foundation)
- ▶ IntelliJ IDEA
 - ▶ <https://www.jetbrains.com/idea/>
 - ▶ kommerziell (JetBrains), kostenlos in einer Community-Version
- ▶ Visual Studio Code
 - ▶ <https://code.visualstudio.com/>
 - ▶ frei (Microsoft, MIT-Lizenz)
 - ▶ Java-Entwicklung über Extensions
- ▶ Praktikum
 - ▶ **Visual Studio Code** (siehe Tutorial Videos Moodle)
 - ▶ **Andere IDEs selber** verantwortlich



Notizen

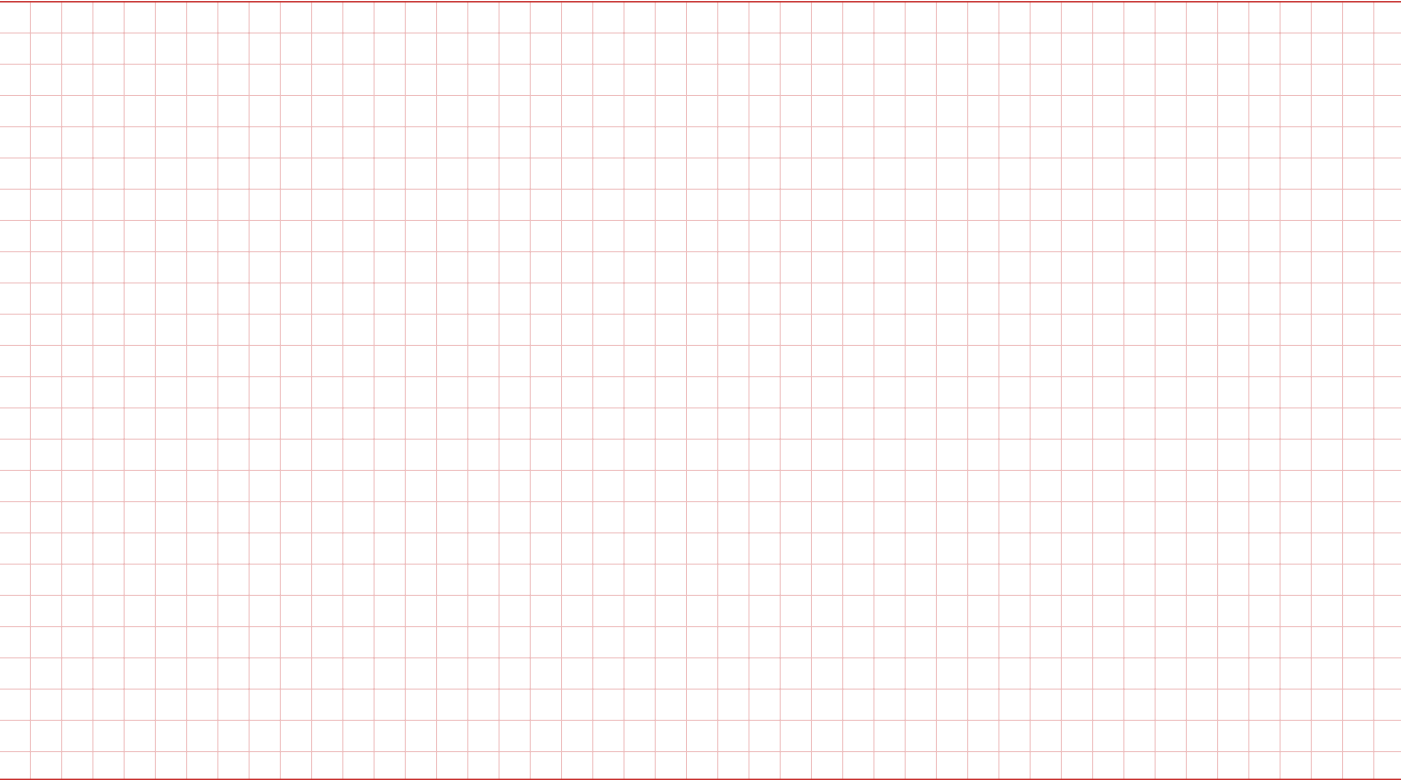
- Apache NetBeans Logo by Apache NetBeans licensed under [↗ Apache License, Version 2.0](#)
- IntelliJ IDEA Logo by [↗ JetBrains](#)



Inhalt

Entwickeln mit Java
Build Tools

Notizen



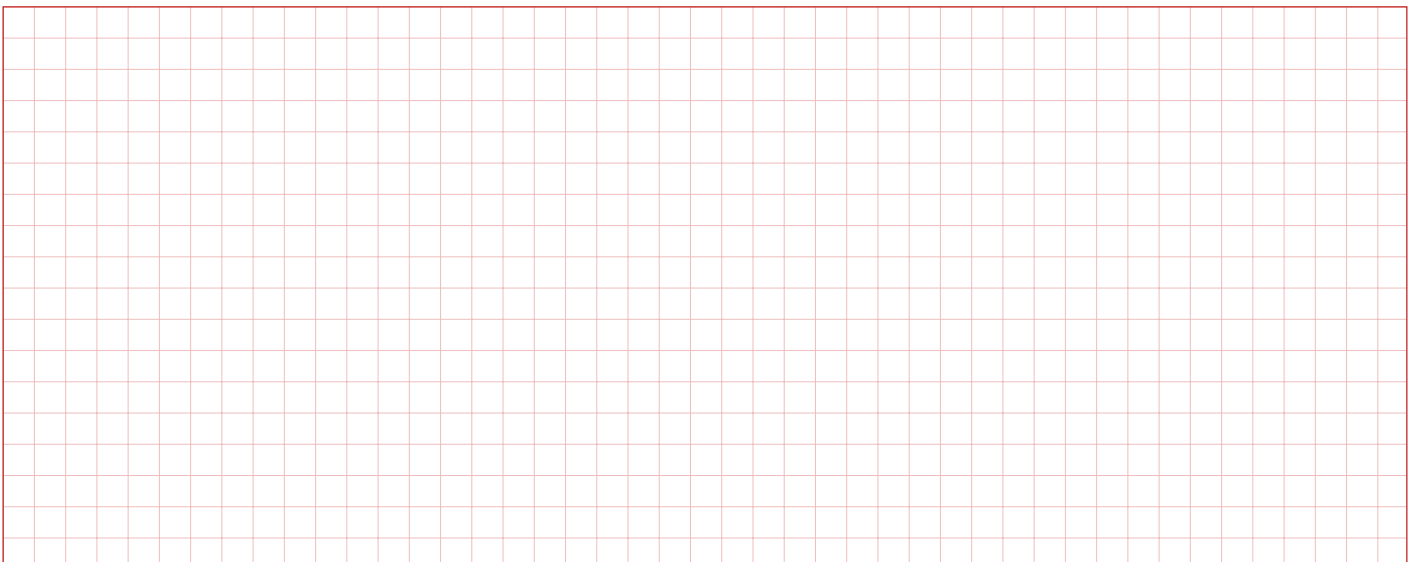
Build Tools

- ▶ Build Tools?
 - ▶ Compilieren, testen, verwalten von komplexen Java-Anwendungen
 - ▶ automatisches Auflösen von Abhängigkeiten
 - ▶ Kommandozeile oder IDE-Integration
- ▶ Apache ANT
 - ▶ <https://ant.apache.org/>
 - ▶ Projektbeschreibung: XML
 - ▶ sehr flexibel
- ▶ Apache Maven
 - ▶ <https://maven.apache.org/>
 - ▶ Projektbeschreibung: XML
 - ▶ komfortabler als ANT
- ▶ Gradle
 - ▶ <https://gradle.org/>
 - ▶ Projektbeschreibung: Groovy/Kotlin
 - ▶ ähnlich zu Maven
 - ▶ verwendet für Programmierbeispiele



Notizen

- Apache Ant Logo by [↗ Apache Ant Projekt Team](#) licensed under [↗ Apache License, Version 2.0](#)
- Apache Maven Logo by Apache Software Foundation licensed under [↗ Apache License, Version 2.0](#)



Inhalt

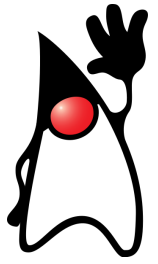
Entwickeln mit Java

jshell

Notizen

jshell

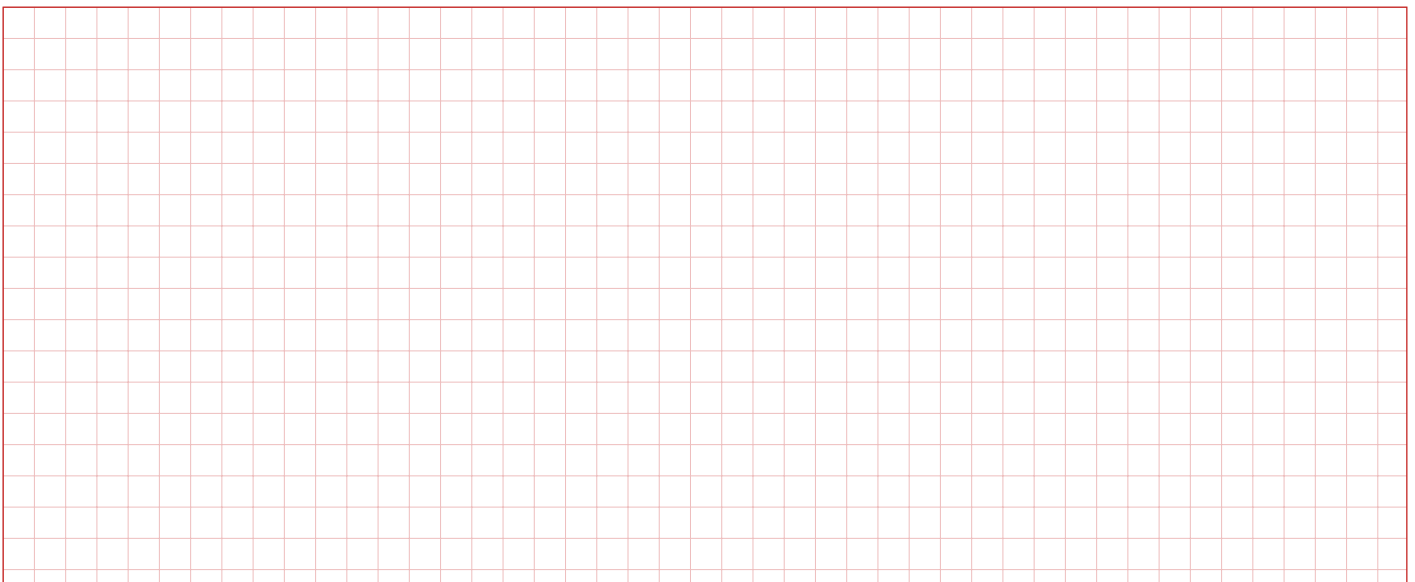
- ▶ jshell: **Java REPL** („read-evaluate-print-loop“)
- ▶ Wird mit JDK installiert (ab 9)
- ▶ Java Konsole zum...
 - ▶ Ausprobieren
 - ▶ Testen
 - ▶ „rapid prototyping“



```
jshell> System.out.println("Hello World!");
Hello World!
jshell> long fib(int n){
...> return (n<=1 ? 1 : fib(n-1) + fib(n-2));
...> }
created method fib(int)
jshell> fib(10);
$3 ==> 89
```

Notizen

- Die jshell bietet auch die Möglichkeit Code-Snippets aus Dateien zu laden und in Dateien zu speichern. Eine Auflistung der Funktionen ist hier zu finden:
<https://docs.oracle.com/javase/9/tools/jshell.htm>

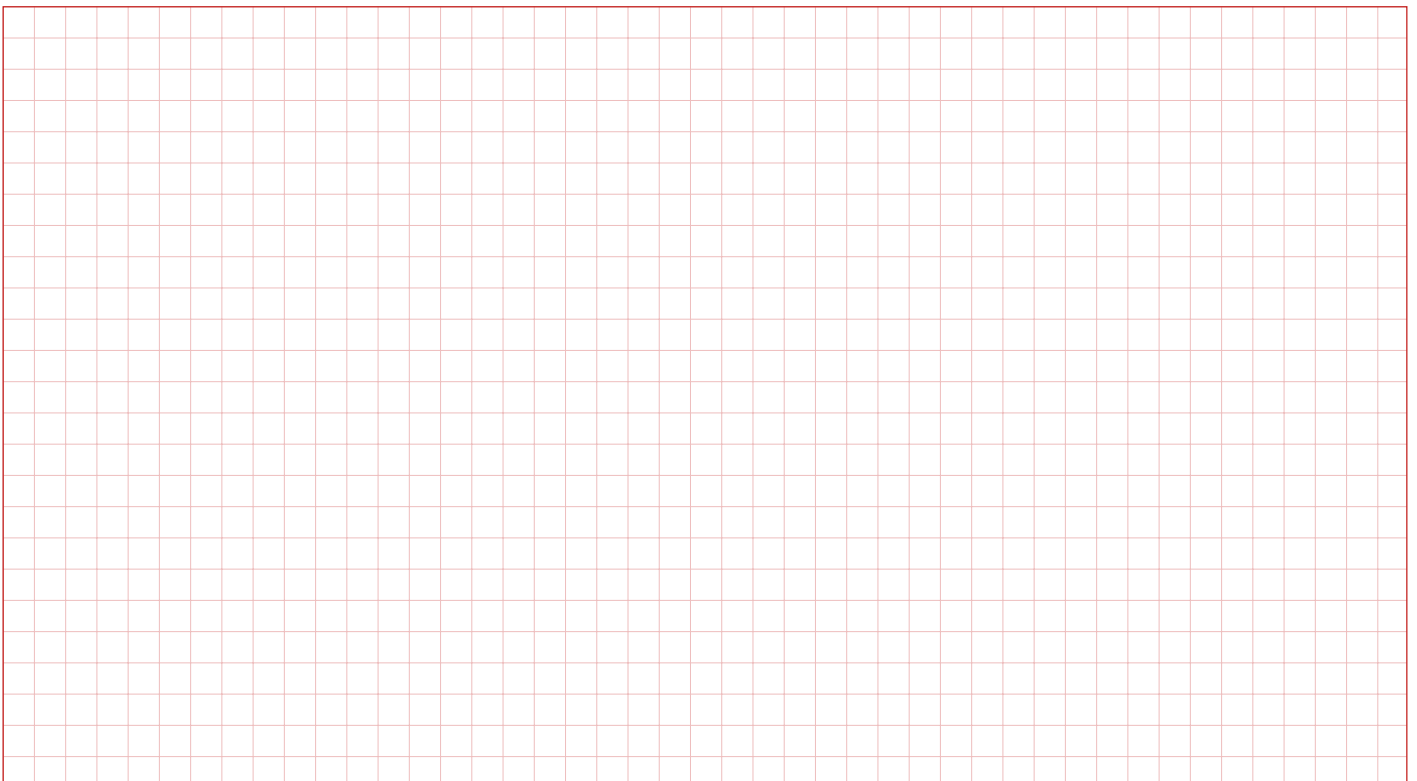


jshell: Noch ein Beispiel

Öffnet ein Fenster mit einer Schaltfläche:

```
import javax.swing.*;  
var frame = new JFrame("Hello Java!");  
var button = new JButton("Press me!");  
button.addActionListener(  
    event -> System.out.println("Button pressed!") );  
frame.add(button);  
frame.pack();  
frame.setVisible(true);
```

Notizen



Inhalt

Literatur und Ressourcen

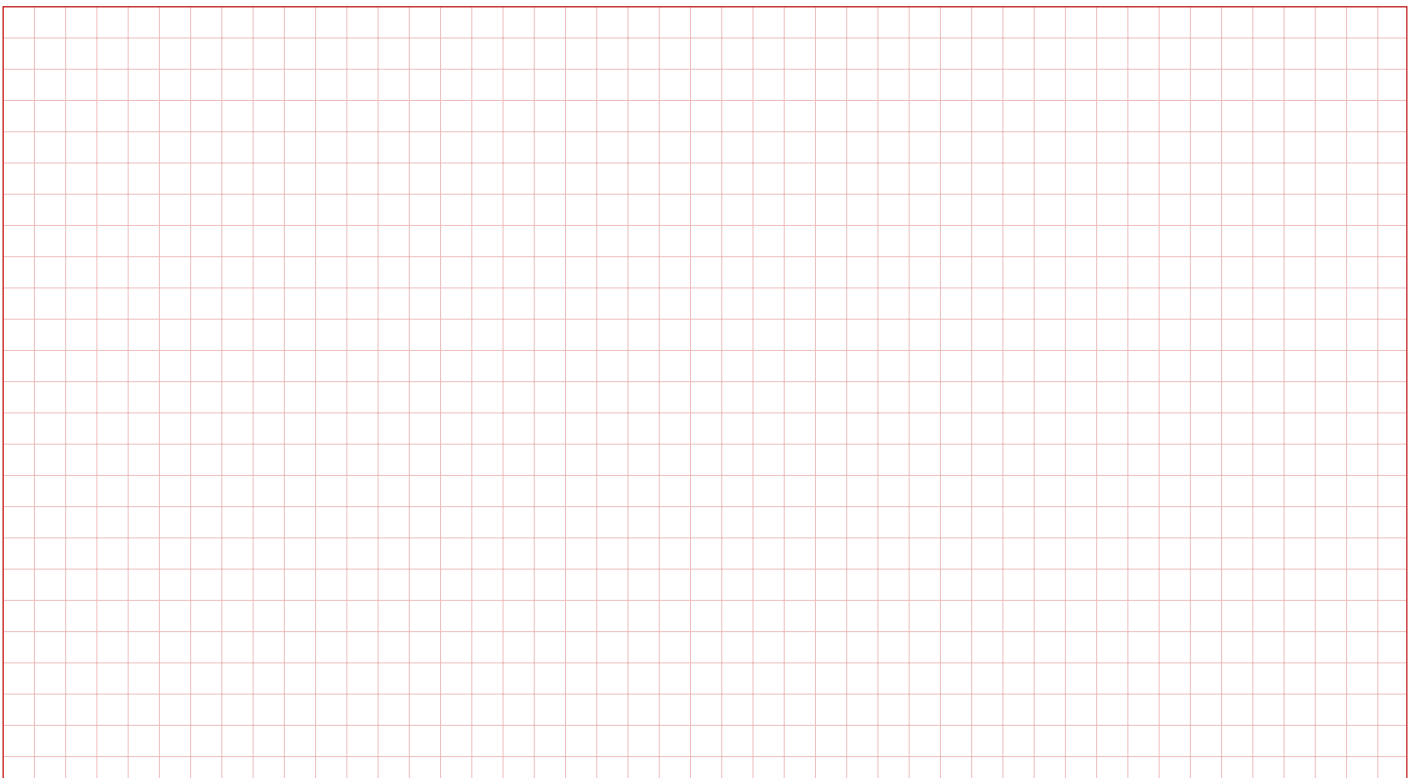
46

Notizen

Literatur

- ▶ **[Insel]**
Christian Ullenboom. *Java ist auch eine Insel*. 14. Aufl. Bonn: Galileo Press, 2018. ISBN: 978-3-8362-6721-2. URL: <http://openbook.rheinwerk-verlag.de/javainsel/>
- ▶ **[Schiedermeier]**
Reinhard Schiedermeier. *Programmieren mit Java*. 2. Aufl. Hallbergmoos: Pearson Studium, 2010. ISBN: 978-3-8689-4031-2. URL: <https://sol.cs.hm.edu/4031/>
- ▶ **[Indena]**
Michael Inden. *Der Weg zum Java-Profi*. 3. Aufl. Heidelberg: dpunkt.verlag, 2015. ISBN: 978-3-86490-203-1
- ▶ **[Ind15]**
Michael Inden. *Java 8 — Die Neuerungen*. 2. Aufl. Heidelberg: dpunkt.verlag, 2015. ISBN: 978-3-86490-290-1
- ▶ **[Indenb]**
Michael Inden. *Java — Die Neuerungen von Version 9 bis 12*. 1. Aufl. Heidelberg: dpunkt.verlag, 2019. ISBN: 978-3-86490-672-5

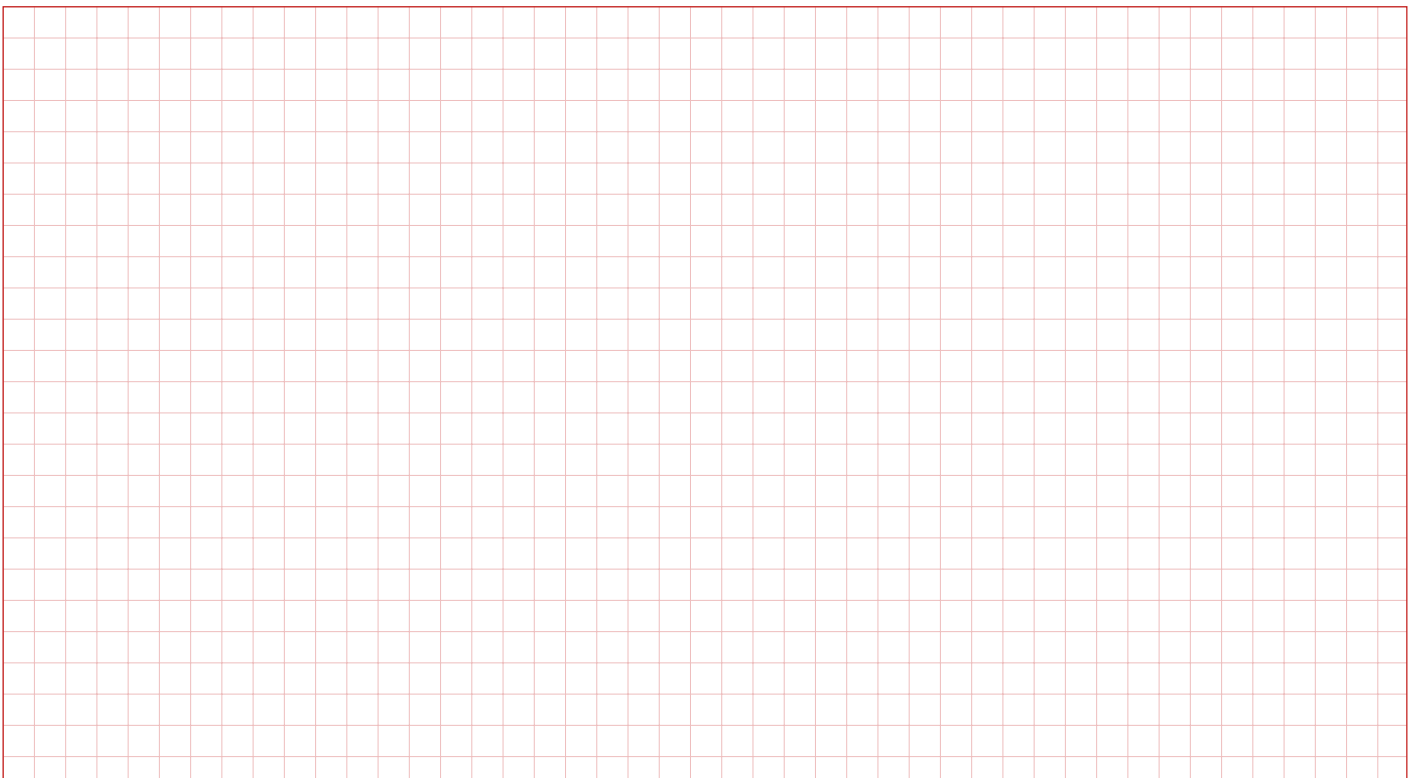
Notizen



Online-Ressourcen

- ▶ Christian Ullenboom — „Java ist auch eine Insel“, (12. Auflage) als Online-eBook:
<http://openbook.rheinwerk-verlag.de/javainsel/>
- ▶ Java API Dokumentation (12):
<https://docs.oracle.com/en/java/javase/12/docs/api/index.html>
- ▶ Download Oracle JDK:
<https://www.oracle.com/java/technologies/javase-downloads.html>
- ▶ Download OpenJDK: <https://openjdk.java.net/>
- ▶ Download Visual Studio Code <https://code.visualstudio.com/>

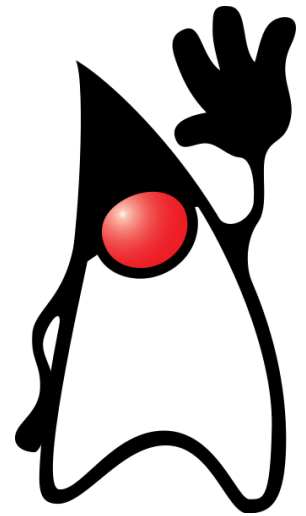
Notizen



Notizen

Überblick

- ▶ **Grundlagen:** imperative Sprachkonstrukte, primitive Typen
- ▶ **Objektorientierte Programmierung:** Klassen, Referenzen, **enums**, javadoc
- ▶ **Strings:** Characters, Zeichenketten, arbeiten mit Strings
- ▶ **Arrays:** eindimensional, mehrdimensional, arbeiten mit Arrays
- ▶ **Objektorientierung vertieft:** Packages, Vererbung, Interfaces
- ▶ **Ausnahmenbehandlung:** try-catch, (un)geprüfte Ausnahmen, Ausnahmen definieren
- ▶ **Collections:** Java-Collections, Iteratoren
- ▶ **Ein-/Ausgabe:** Datenströme, arbeiten mit Dateien und Verzeichnissen



Notizen

