

# Programmieren II: Java

## Einführung

Prof. Dr. Christopher Auer

Sommersemester 2024



Eine kurze Geschichte von Java

Was macht Java aus?

Java-Plattformen und Implementierungen

„Hello World“

Entwickeln mit Java

Literatur und Ressourcen

Überblick

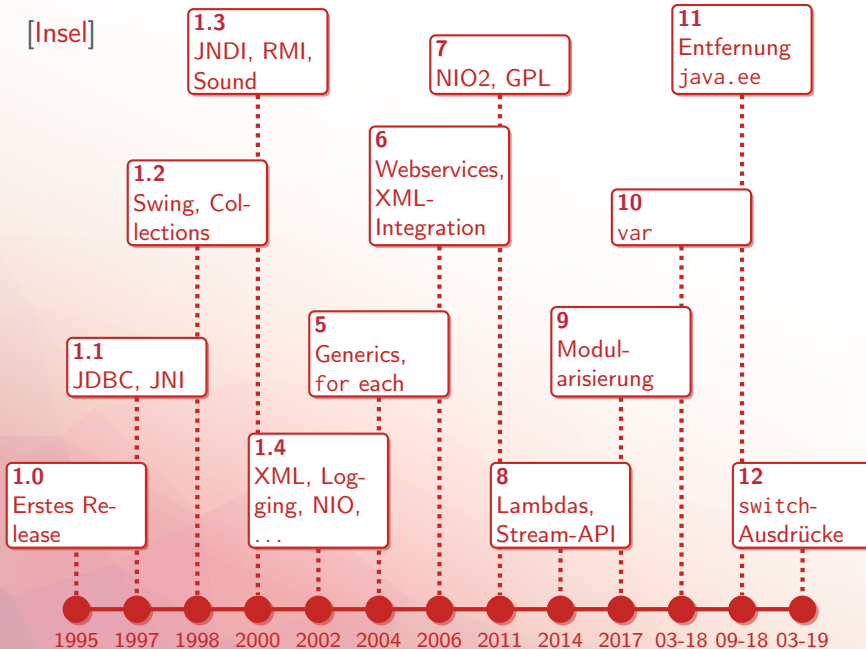
# Inhalt

## Eine kurze Geschichte von Java

- ▶ 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

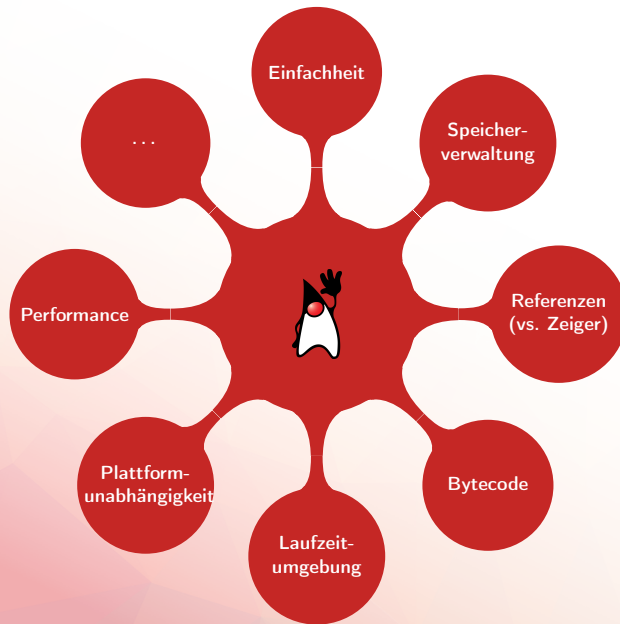


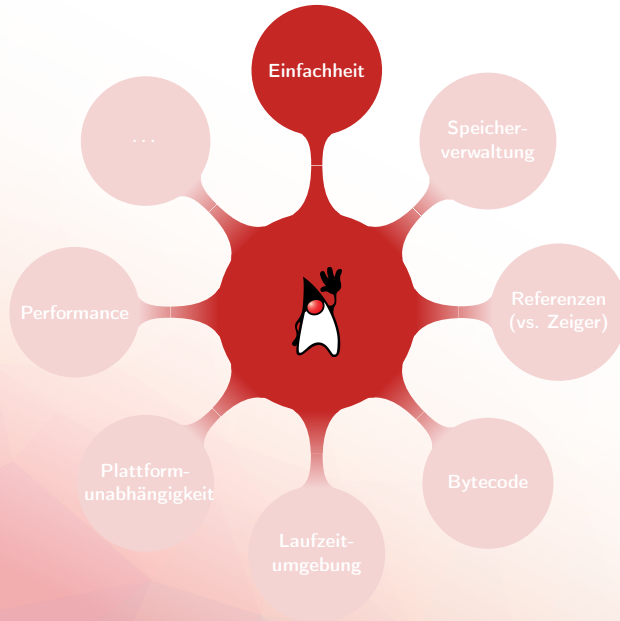
[Insel]



# Inhalt

## Was macht Java aus?







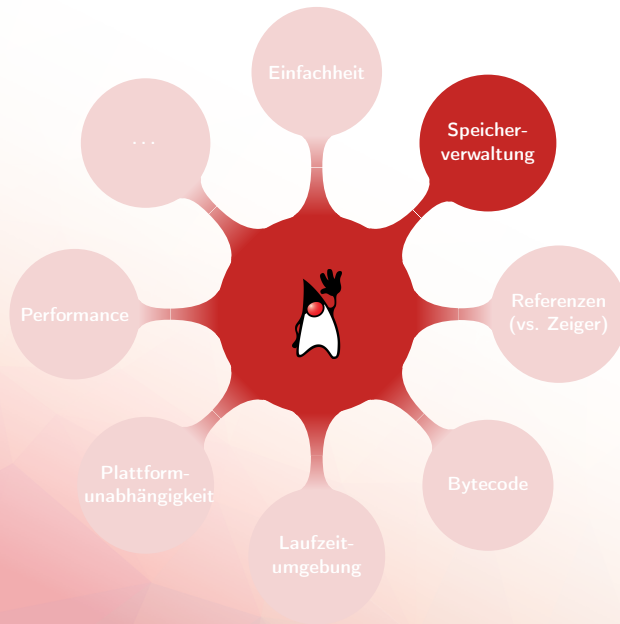
- ▶ Syntax und Konzepte basieren auf bekannten Sprachen (C++)

```
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



## ► C++

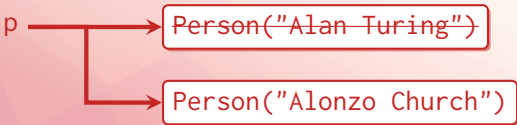
```
Person* p = new Person("Alan Turing");  
p = new Person("Alonzo Church");
```

## ► 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)

## ► Vorteile

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

## ► Nachteile

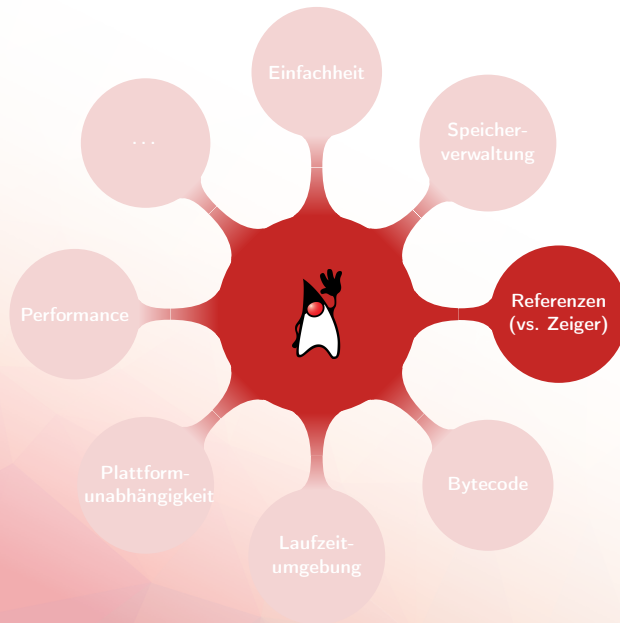
- ✗ GC braucht Ressourcen (CPU, Speicher)
- ✗ Aufräumzeitpunkt nicht vorhersehbar
- ✗ keine explizite Freigabe möglich

```
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];
```



## ► C++

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

p beinhaltet die Speicheradresse

## ► Java

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

p beinhaltet die **Referenz**

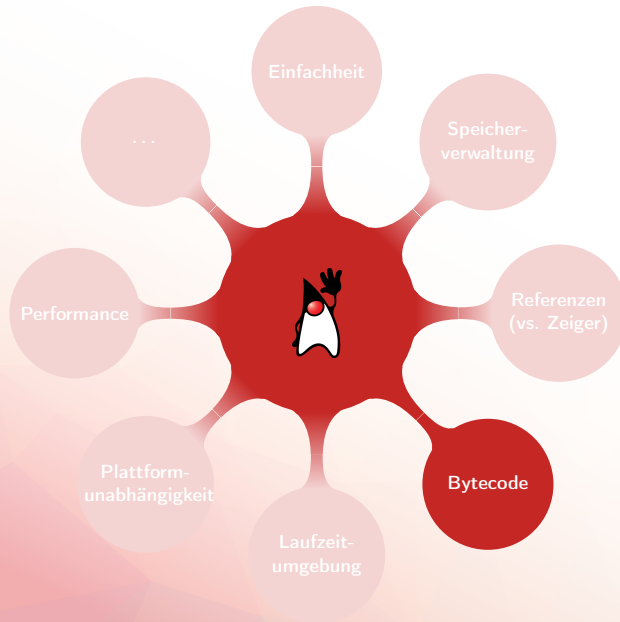
## ► Java-Referenz beinhaltet...

- die eigentliche Speicheradresse
- Typinformationen

## ► Vorteile

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

## ► **Nachteil:** aufwändigere Dereferenzierung



## Java-Compiler javac

Quellcode (.java-Datei)

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

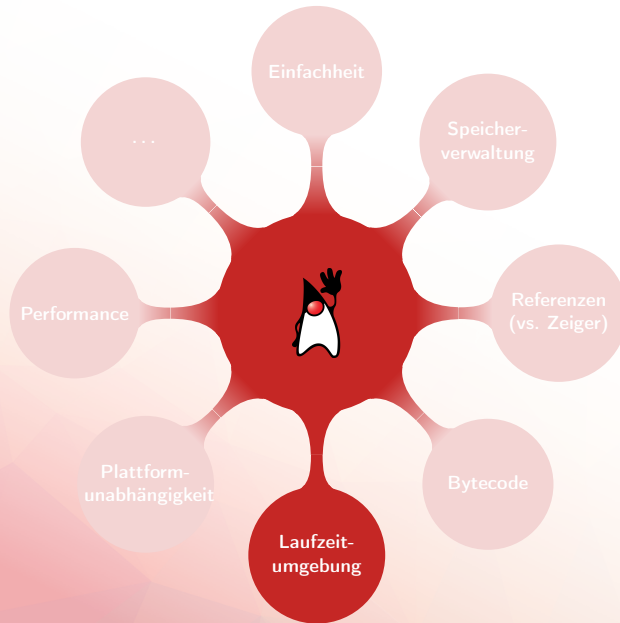
↓  
javac

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

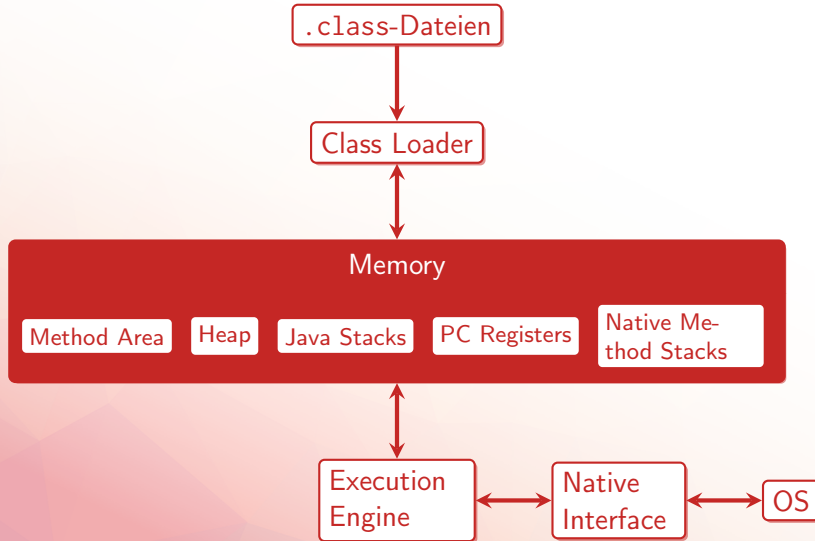
Bytecode (.class-Datei)

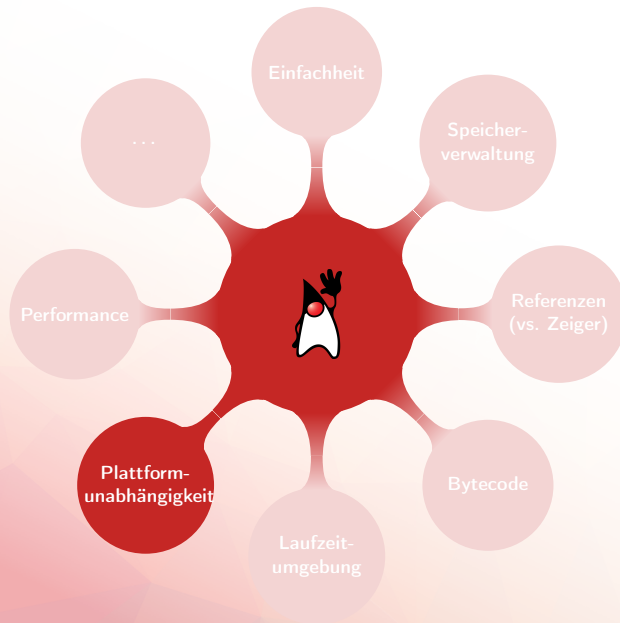
Bytecode: Instruktionen für virtuelle Java-Maschine





## Java Laufzeitumgebung: „Java Virtual Machine“

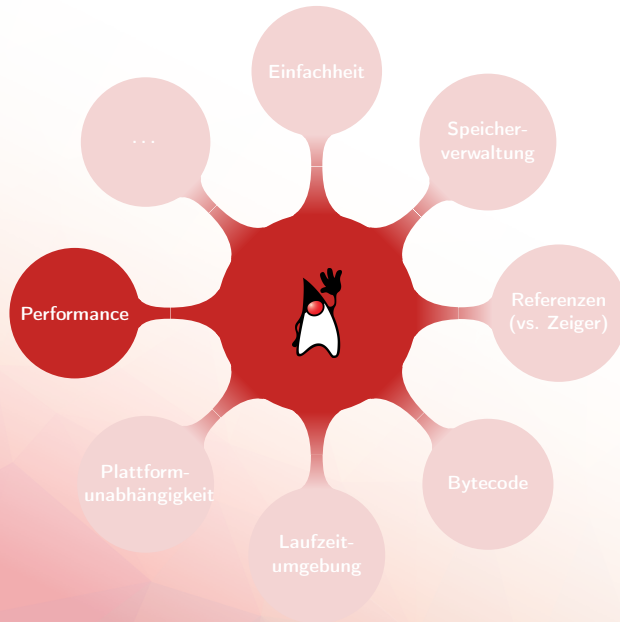


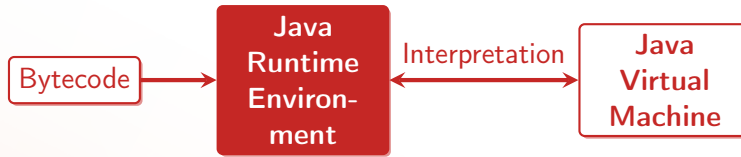


```
public static int add(int, int);  
    iload_0  
    iload_1  
    iadd  
    ireturn
```

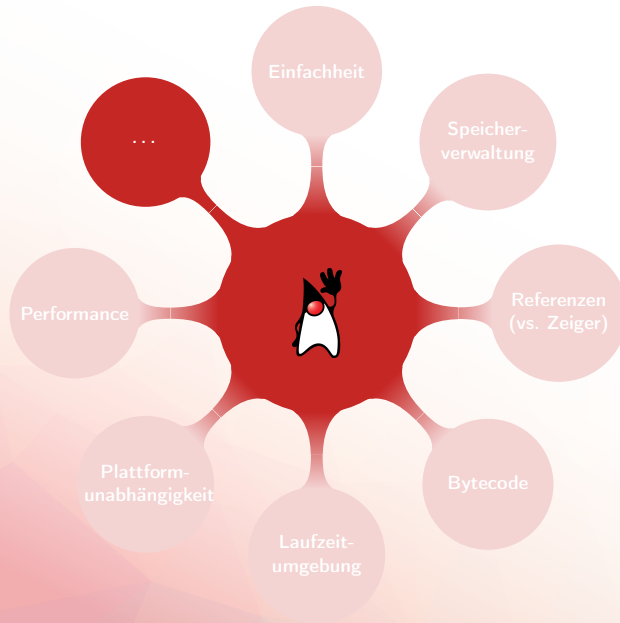


- ▶ Java Runtime Environment (JRE)
  - ▶ interpretiert Bytecode in Java Virtual Machine
  - ▶ reicht Aufrufe an Betriebssystem weiter
- ▶ JRE bildet eine Abstraktionsschicht
- ▶ verschiedene JRE-Implementierungen möglich
- ▶ Bytecode ist **plattformunabhängig**



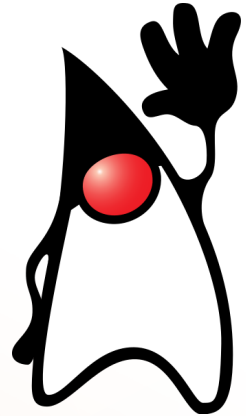


- ▶ Interpretation
  - ▶ Erkennen
  - ▶ Dekodieren
  - ▶ Ausführen
- ▶ Problem: langsam!
- ▶ Just-in-Time-Compilation
  - ▶ Ausführungseinheit erkennt „Hotspots“ (oft ausgeführte Programmteile)
  - ▶ Übersetzung in Maschinencode zur Laufzeit
- ▶ Vorteile
  - ✓ Kontextinformationen (Programm, Prozessor,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



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++)



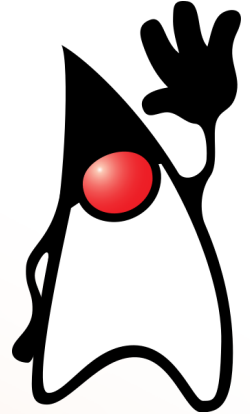


Java eignet sich für...

- ✓ **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



# Inhalt

## Java-Plattformen und Implementierungen

Plattformen

Implementierungen

Java als Plattform für Sprachen

# Inhalt

## Java-Plattformen und Implementierungen

### Plattformen



- ▶ 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

# Inhalt

## Java-Plattformen und Implementierungen

### Implementierungen

## ▶ 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

# Inhalt

## Java-Plattformen und Implementierungen

Java als Plattform für Sprachen

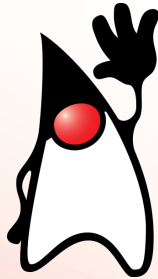
# Java als Plattform für Sprachen



Scala



Kotlin



Jython



Clojure



# Inhalt

„Hello World“

```
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

- ▶ **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!
```

## 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
    /* ... */
}
```

## 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  
}
```

# Inhalt

## Entwickeln mit Java

IDEs

Build Tools

jshell

# Inhalt

## Entwickeln mit Java IDEs



## 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





# Inhalt

## Entwickeln mit Java Build Tools



# 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



# Inhalt

## Entwickeln mit Java

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
```

## 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);
```

# Inhalt

## Literatur und Ressourcen

# 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

- ▶ 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/>



# Inhalt

## Überblick

## 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

