

# Java Basics, Objekte und Klassen

**Programmieren Tutorium Nr.12**

Aleksandr Zakharov | 11. November 2025

# Java

Java  
●○○○○○○○

Basics beim Java-Programmieren  
○○○

Objekte und Klassen  
○○○○○○○○○○

Schluss  
○○

# Was ist Java?

- Objektorientierte Programmiersprache
- Plattformunabhängig
- Guter Einstieg für Beginner

Java  
○●○○○○○○

Basics beim Java-Programmieren  
○○○

Objekte und Klassen  
○○○○○○○○○○

Schluss  
○○

# Arbeitsablauf in Java

## 1. Quelltext schreiben

- Benötigt Texteditor

## 2. Quelltext kompilieren

- `javac Dateiname.java`
- Übersetzt Java Quelltext (.java) in Bytecode (.class)
- Benötigt Compiler (JDK)

## 3. Bytecode ausführen

- `java Dateiname`
- Funktioniert nur mit Klassen, die die main-Methode enthalten
- Benötigt JVM

## Neukompilieren

Programmcode und Bytecode werden nicht automatisch synchronisiert.  
Jede Änderung, die ihr an dem Programmcode macht, werden erst nach erneuter Kompilierung im ausführbaren Bytecode wieder gespiegelt!

# JRE vs JDK

## Java Runtime Environment

- Laufzeitumgebung, um Plattformunabhängigkeit zu ermöglichen
- Beinhaltet JVM, die den Bytecode ausführt und genau für die Plattformunabhängigkeit sorgt

## Java Development Kit

- Zur Erstellung von Java Programmen benötigt
- Beinhaltet JRE, Java-Compiler und weitere Werkzeuge (z.B. Bibliotheken)

Java  
○○○●○○○○

Basics beim Java-Programmieren  
○○○

Objekte und Klassen  
○○○○○○○○○○

Schluss  
○○

# Java (JDK) installieren

Artemis akzeptiert Java version 21

- Oracle JDK  
<https://www.oracle.com/java/technologies/downloads/#java21>
- OpenJDK  
<https://jdk.java.net/archive/>

## Anleitungen

- Windows: <https://docs.oracle.com/en/java/javase/21/install/installation-jdk-microsoft-windows-platforms.html>
- MacOS: <https://docs.oracle.com/en/java/javase/21/install/installation-jdk-macos.html>
- Ubuntu: <https://wiki.ubuntuusers.de/Java/Installation/OpenJDK/>

Java  
○○○○●○○○

Basics beim Java-Programmieren  
○○○

Objekte und Klassen  
○○○○○○○○○○

Schluss  
○○

# Texteditoren

## Windows

- Windows Editor
- Notepad++

## MacOS

- TextEdit
- SimpleEdit

## Linux

- nano, vim
- gedit
- Kate

## Word

Microsoft Word ist **kein** Texteditor!

# Was sind IDEs?

IDEs (Integrated Entwicklungsumgebungen) sind sehr fortgeschrittene Texteditoren

## In IDEs integrierte Funktionen

- Texteditor
- Konsole
- Compiler
- Debugger
- Code-Highlighting
- Code-Vervollständigung
- Versionskontrollsystem (bspw. Git)
- auch KI Unterstützung
- ...

# Welche IDEs gibt es?



Eclipse

<https://eclipseide.org/>

← IDE der Vorlesung



IntelliJ Idea

<https://www.jetbrains.com/idea/>

← Was ich benutze



Visual Studio Code

<https://code.visualstudio.com/download>

← Auch erwähnenswert

Java  
○○○○○○●

Basics beim Java-Programmieren  
○○○

Objekte und Klassen  
○○○○○○○○○○

Schluss  
○○

# Basics beim Java-Programmieren

Java  
○○○○○○○○

Basics beim Java-Programmieren  
●○○

Objekte und Klassen  
○○○○○○○○○○

Schluss  
○○

# Wie sieht Code eigentlich aus?

```
1 public class Something {  
2     int property;  
3  
4     //I am a one-line comment  
5     void foo() {  
6         doSomething(5);  
7     }  
8  
9     /* I am on the contrary  
10    a multiple-line comment  
11    xD  
12    */  
13    void doSomething(int number) {  
14        property = number;  
15    }  
16 }
```

Wir können hier sehen:

- Klasse
- Sichtbarkeit
  - mehr dazu nächste Woche
- Attribut
- Kommentar
  - IntelliJ: Kommentieren mit "Ctrl + /"
- Methode
- Parameter
- Anweisung
- Variable
  - mehr dazu nächste Woche
- Block (auch Rumpf genannt)

# Einstiegspunkt ins Programm

- Die **main-Methode** ist dieser Punkt
- Ein Programm darf nur *einen* Einstiegspunkt haben (also nur eine main-Methode)
- Endet die main-Methode, so kommt das Programm zum Ende

```
1 public class Main {  
2     public static void main(String[] args) {  
3         //something clever here  
4     }  
5 }
```

- `String[] args` bezeichnet die Kommandozeilen-Argumente. Die übergibt man mit dem Start des Programms

# Objekte und Klassen

Java  
○○○○○○○○

Basics beim Java-Programmieren  
○○○

Objekte und Klassen  
●○○○○○○○○

Schluss  
○○

# Was ist OOP?

- Object **O**riented **P**rogramming
- Grundidee: wir modellieren Objekte aus der Realität auf dem Rechner
- *Programmieren*  $\neq$  *Modellieren*

Java  
○○○○○○○○

Basics beim Java-Programmieren  
○○○

Objekte und Klassen  
○●○○○○○○○○

Schluss  
○○

# Warum OOP?

Alternativ: Ein Script wird ohne weiteres von oben nach unten durchgelaufen

- Was passiert, wenn das Programm erweitert werden muss?
- Was passiert, wenn ein Code-Block mehrmals verwendet werden muss?

## Vorteile von OOP

- Modularität
- Anpassungsfähigkeit
- ...
- Wiederverwendbarkeit
- Verständlichkeit (v.a. bei großen Projekten)

# Beispiel



- PC → Identität
- Welche Eigenschaften bzw. Komponente hat der Rechner? → Zustand
- Was kann der PC leisten? → Verhalten

# Übertragen in Java

```
1 class Computer {
2     //Attributes
3     int memoryCapacity;
4     String seriesNumber;
5     double neededPower;
6
7     //Methods
8     void turnOn() {
9         //turn the PC on
10    }
11
12    void signUpForExerciseCertificate() {
13        //do it!!!
14    }
15
16    void launchTiktok() {
17        //continue brainrotting
18    }
19 }
```

- Klassendeklaration mit `class Computer`
- Zustandsbeschreibung mithilfe von **Attributen**
- Verhalten ist definiert durch die **Methoden**

# Abstraktion (Kapselung)

```
1 public class MemoryModule {  
2     //Attributes  
3     double memoryCapacity;  
4     double maximalFrequency;  
5     String memoryType;  
6  
7     //Methods  
8     //e.g. boost(), changeVoltage(int value), etc.  
9 }
```

```
1 class Computer {  
2     //Attributes  
3     MemoryModule memory;  
4     String seriesNumber;  
5     double neededPower;  
6  
7     //Methods...  
8 }
```

Warum abstrahieren wir?

- Kürzere und übersichtlichere Klassendeklaration
- Intuitive Struktur (Kapselung)
- Wiederverwendbarkeit
- ...

# Jetzt seid ihr dran!

## Aufgabe 1

Modelliert die KIT-Bib!

- Welche gleichartige Objekte (Klassen) gibt es?
- Welche Eigenschaften haben diese Klassen? (Attribute)
- Welches Verhalten ist von jeder Klasse zu erwarten? (Methoden)

## Aufgabe 2

Modelliert das Spiel Monopoly!

Java  
○○○○○○○○

Basics beim Java-Programmieren  
○○○

Objekte und Klassen  
○○○○○○●○○○

Schluss  
○○

# Computer 'erschaffen'

```
1 public class PCCreator {
2     public static void main(String[] args) {
3         //create a memory module
4         MemoryModule ddr5 = new MemoryModule();
5         //set its attributes
6         ddr5.memoryCapacity = 32.00;
7         ddr5.maximalFrequency = 3200;
8
9         //create a pc
10        Computer pc = new Computer();
11        //set its attributes
12        pc.memory = ddr5;
13        pc.neededPower = 850;
14        pc.seriesNumber = "upgcv1v";
15
16        //usual time with a computer
17        pc.turnOn();
18        pc.signUpForExerciseCertificate();
19        pc.launchTiktok();
20    }
21 }
```

- Instanz einer Klasse ist ein Objekt
- Ein Objekt wird mit `new` `Klassenname()` erzeugt
- Zugriff auf Attribute mit `variablenname.attribut`
- Ausführen von Methoden mit `variablenname.methode()`

# Naming Conventions

## Naming von Variablen und Methoden

Das sind optionale aber stark empfohlene **Vorschriften** wie man Klassen, Interfaces, Variablen, Methoden usw benennt. Das bringt Konsistenz und gute Lesbarkeit des Codes.

Beispiele:

- camelCase bzw. lowerCamelCase: Attribute, Methoden
- UpperCamelCase: Klassen, Interfaces
- ALL\_CAPS: Konstanten
- ...

# Weitere Conventions

- Kein nicht englischer Code
  - Folglich keine deutsche Buchstaben wie ß, ä, ö usw
- Die Klassennamen entsprechen den Dateinamen (`class Car` → `Car.java`)

# Schluss

Java  
○○○○○○○○

Basics beim Java-Programmieren  
○○○

Objekte und Klassen  
○○○○○○○○○○

Schluss  
●○

# Zusammenfassung

- Quellcode schreiben → Kompilieren → Bytecode ausführen
- Java Programme beginnen in der und enden nach der main-Methode
- OOP ist ein Ansatz für die Modellierung von Software
- Klassen sind "Baupläne" für Objekten
- Objekte sind Instanzen von Klassen
- Objekte werden mit **new** erzeugt
- Naming Conventions sind hilfreiche Vorschriften, schönen Code zu schreiben