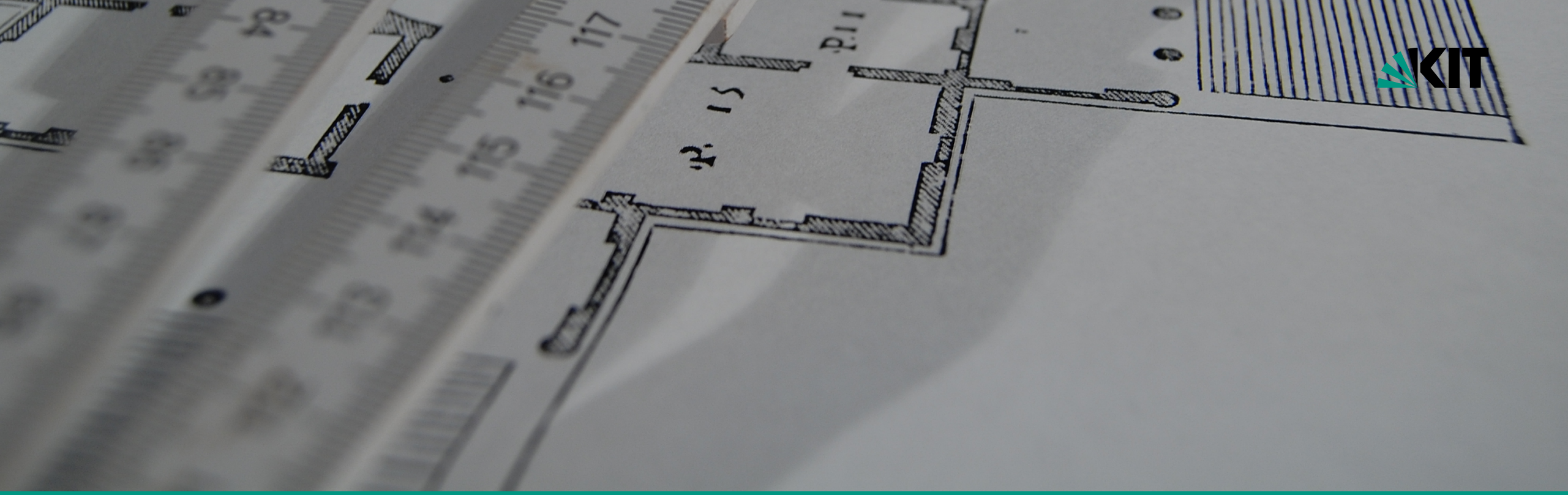

Java Basics, Objekte und Klassen

Programmieren Tutorium Nr.12

Aleksandr Zakharov | 5. November 2025

Java

Was ist Java?

- Objektorientierte Programmiersprache
- Plattformunabhängig
- Guter Einstieg für Beginner

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`-Mathode 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 **R**untime **E**nvironment

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

Java **D**evelopment **K**it

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

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/`

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
- Vesionskontrollsystem (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

Basics
beim Java-Programmieren

Wie sieht Code eigentlich aus?

```java
public class Something {
   int property;

   //I am a one-line comment
   void foo() {
      doSomething(5);
   }

   /* I am on the contrary
   a multiple-line comment
   xD
    */
   void doSomething(int number) {
      property = number;
   }
}
```

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

```java
public class Main {
   public static void main(String[] args) {
      //something clever here
   }
}
```

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

Objekte und Klassen

Was ist OOP?

- **O**bject **O**riented **P**rogramming
- Grundidee: wir modellieren Objekte aus der Realität auf dem Rechner

- Programmieren Modellieren

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

Übertragen in Java

```java
class Computer {
   //Attributes
   int memoryCapacity;
   String seriesNumber;
   double neededPower;

   //Methods
   void turnOn() {
      //turn the PC on
   }

   void signUpForExerciseCertificate() {
      //do it!!!
   }

   void launchTiktok() {
      //continue brainrotting
   }
}
```

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

Abstraktion (Kapselung)

```java
public class MemoryModule {
    //Attributes
    double memoryCapacity;
    double maximalFrequency;
    String memoryType;

    //Methods
    //e.g. boost(), changeVoltage(int value), etc.
}
```

```java
class Computer {
    //Attributes
    MemoryModule memory;
    String seriesNumber;
    double neededPower;

    //Methods...
}
```

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!

Computer ’erschaffen’

```java
public class PCCreator {
   public static void main(String[] args) {
      //create a memory module
      MemoryModule ddr5 = new MemoryModule();
      //set its attributes
      ddr5.memoryCapacity = 32.00;
      ddr5.maximalFrequency = 3200;

      //create a pc
      Computer pc = new Computer();
      //set its attributes
      pc.memory = ddr5;
      pc.neededPower = 850;
      pc.seriesNumber = "upgcv1v";

      //usual time with a computer
      pc.turnOn();
      pc.signUpForExerciseCertificate();
      pc.launchTiktok();
   }
}
```

- 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

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