Objektorientierte Programmierung in Java

Vorlesung 1 - Organisation und Einführung

Emily Lucia Antosch

HAW Hamburg

08.10.2024

Inhaltsverzeichnis

1. Organisation

1.1 Das Ziel dieses Kapitels

- Ich will mich bei Ihnen vorstellen und mit Ihnen den Ablauf dieses Moduls besprechen.
- Sie bekommen einen Überblick über die Voraussetzungen in diesem Modul und können diese erfüllen.
- Sie wissen, wie sich mich erreichen können.

1.2 Kurz zu mir

- Emily Lucia Antosch, 24 Jahre alt
- Bachelor in Elektro- und Informationstechnik
- Zurzeit tätig als Anwendungsentwicklerin bei NVL
- Nächste Station: Masterstudium Informatik
- Mail: emilylucia.antosch@haw-hamburg.de

Info

Ich mach das zum ersten Mal, seien Sie also bitte nachsichtig.

1.3 Vorlesungsablauf

- Vorlesungen teilen sich in Termine am Dienstag und Donnerstag auf
  - Am Anfang sind sehr viele Termine, die Sie auf das Praktikum vorbereiten sollen
- Ich würde Sie bitten, sich an der Vorlesung aktiv zu beteiligen
- Es wird kleine Fragen und Aufgaben geben, die Sie live beantworten und mitprogrammieren können

! Merke

Falls Sie etwas nicht verstehen, fragen Sie bitte sofort! Ich wiederhole gerne Inhalte auf Deutsch oder Englisch!

1.4 Vorlesungsfokus

- Wir wollen hier eine Brücke aus Ihren Vorkenntnissen bauen.
- Am Ende der Vorlesung sollten Sie in der Lage sein, in Java einfache Programme zu erstellen.
- Außerdem sollten Sie dann Objektorientierte Programmierung beherrschen und die Unterschiede zu anderen Paradigmen in der Programmierung herausstellen können.
- Den genauen Stoff können Sie außerhalb der Vorlesung auch im Modulhandbuch nachlesen

1.5 Vorraussetzungen

- Sie brauchen eine Installation des **Java SDK**.
  - Dafür habe ich Ihnen eine Anleitung geschrieben, die Sie im Moodle-Raum finden.
- Außerdem wird die Vorlesung mit dem Tool **JetBrains IntelliJ** sein.
  - Dies ist, wie ich finde, eine sehr gute und einfache IDE für den Anfang.
  - Auch hierfür finden Sie eine Anleitung im Moodle.

2. Einführung

2.1 Das Ziel dieses Kapitels

- Sie sollen ihr bekanntes Wissen aus vorhergehenden Vorlesungen auf neue Inhalte anwenden.
- Sie kennen die grundlegenden Ideen der Objektorientierten Programmierung und kennen den Unterschied zu der Programmierung in C.
- Wir erstellen ein einfaches Programm in der Entwicklungsumgebung Intellij IDEA und führen dieses aus.

2.2 Themenübersicht: Grundlagen

Die ersten Vorlesungen beziehen sich auf die folgenden Prinzipien:

1. Imperative Konzepte
2. Klassen und Objekte
3. Klassenbibliothek
4. Vererbung
5. Schnittstellen

2.3 Themenübersicht: Weiterführende Konzepte

Aus den Grundlagen wollen wir dann weitere Konzepte ableiten:

6. Graphische Oberflächen
7. Ausnahmebehandlung
8. Eingaben und Ausgaben
9. Multithreading (Parallel Computing)

2.4 Objekte und Klassen

In der echten Welt werden oft Dinge über ihre Eigenschaften bestimmt und beschrieben:

- Ein Auto hat Eigenschaften wie
  - einen Hersteller
  - eine Farbe
  - einen Verbrauch

Idee

Mithilfe der Objektorientierten Programmierung können wir diesen intuitiven Ansatz auch bei der Programmierung anwenden!

2.4 Objekte und Klassen

? Frage

Über welche Eigenschaften könnten Sie zum Beispiel eine Person beschreiben? Wie passt das vielleicht in den Programmierungskontext?

2.4 Objekte und Klassen

? Frage

Über welche Eigenschaften könnten Sie zum Beispiel eine Person beschreiben? Wie passt das vielleicht in den Programmierungskontext?

- Für Studenten:
  - Name, Anschrift, Immatrikulationsnummer
- Für Programme/Webseiten:
  - Username, Passwort, Beitrittsdatum

2.4 Objekte und Klassen

- Um aus diesem ähnlichen Bauplan dann mehrere gleichartige Objekte zu erstellen, wird eine Klasse erstellt,
  - Sie enthält alle Eigenschaften, die wir gerade definiert haben, also zum Beispiel Variablen
  - Aus ihr lassen sich ganz verschiedene Objekte erstellen, die diese Eigenschaften unterschiedlich gefüllt haben

Beispiel

Aus der Klasse **Student** lassen sich zum Beispiel die beiden Studenten **Max** und **Ines** erstellen, die beide unterschiedlich heißen und eine eigene Immatrikulationsnummer haben.

2.4 Objekte und Klassen

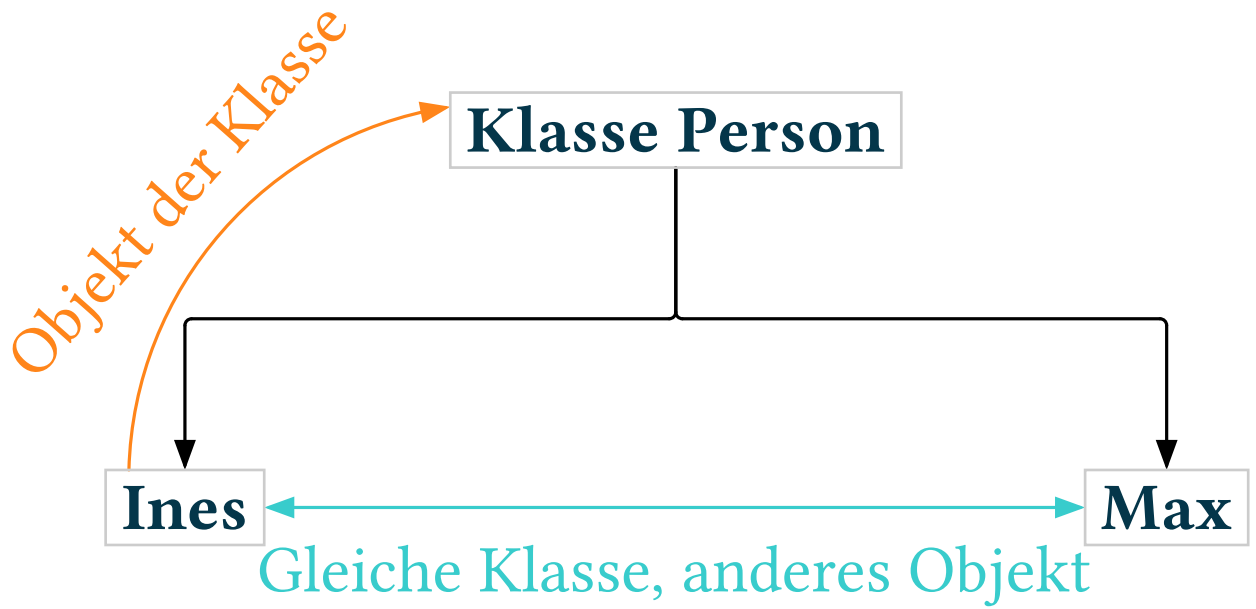


Abbildung 1: Relation zwischen Klassen und Objekten dieser Klasse

2.4 Objekte und Klassen

- Variablen und Funktionen werden also zu einer Klasse zusammengefasst.
  - Es wird eine Menge an Variablen definiert.
  - Für diese Variablen werden wiederum Funktionen eingeführt, die diese lesen und ändern können.

! Merke

- Variablen werden als **Attribute** bezeichnet.
- Die Werte dieser Variablen beschrieben den **Zustand** des Objekts.
- Funktionen werden als **Methoden** bezeichnet.

2.4 Objekte und Klassen

- Über sogennante UML Klassendiagramme lassen sich Klassen mit ihren **Attributen** und **Methoden** beschreiben.

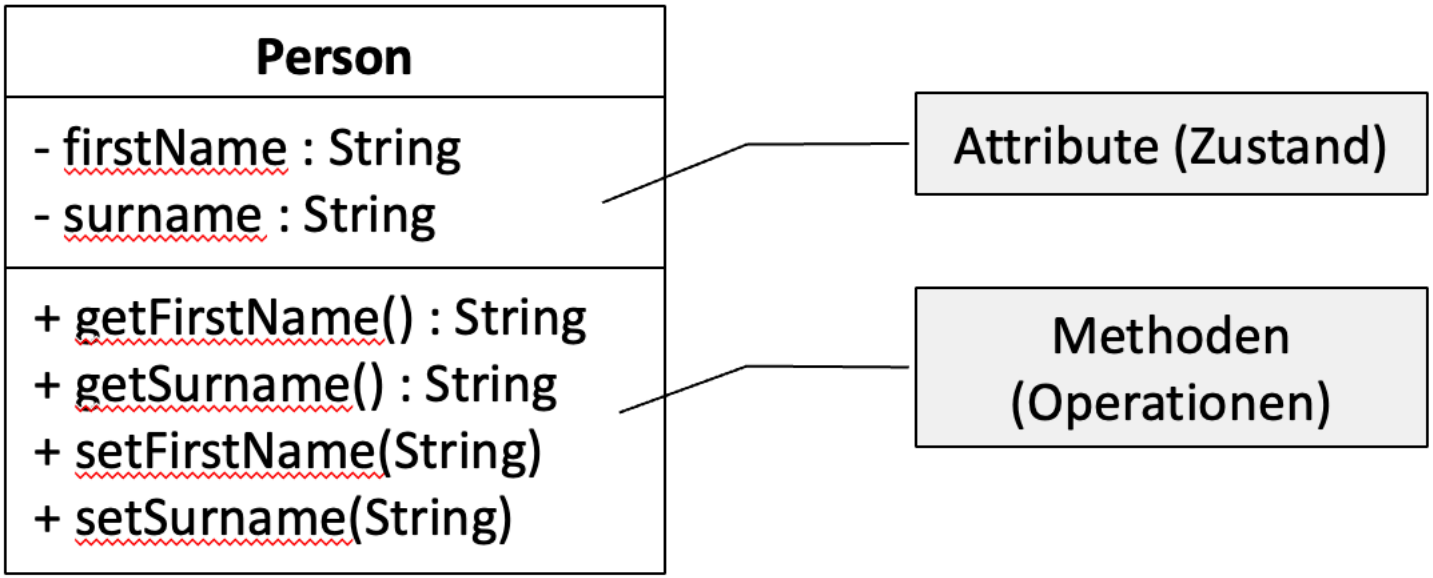


Abbildung 2: UML Klassendiagramm

2.5 Datenkapselung

- Mithilfe von Attributen und Methoden von Klassen lassen sich Daten kapseln.

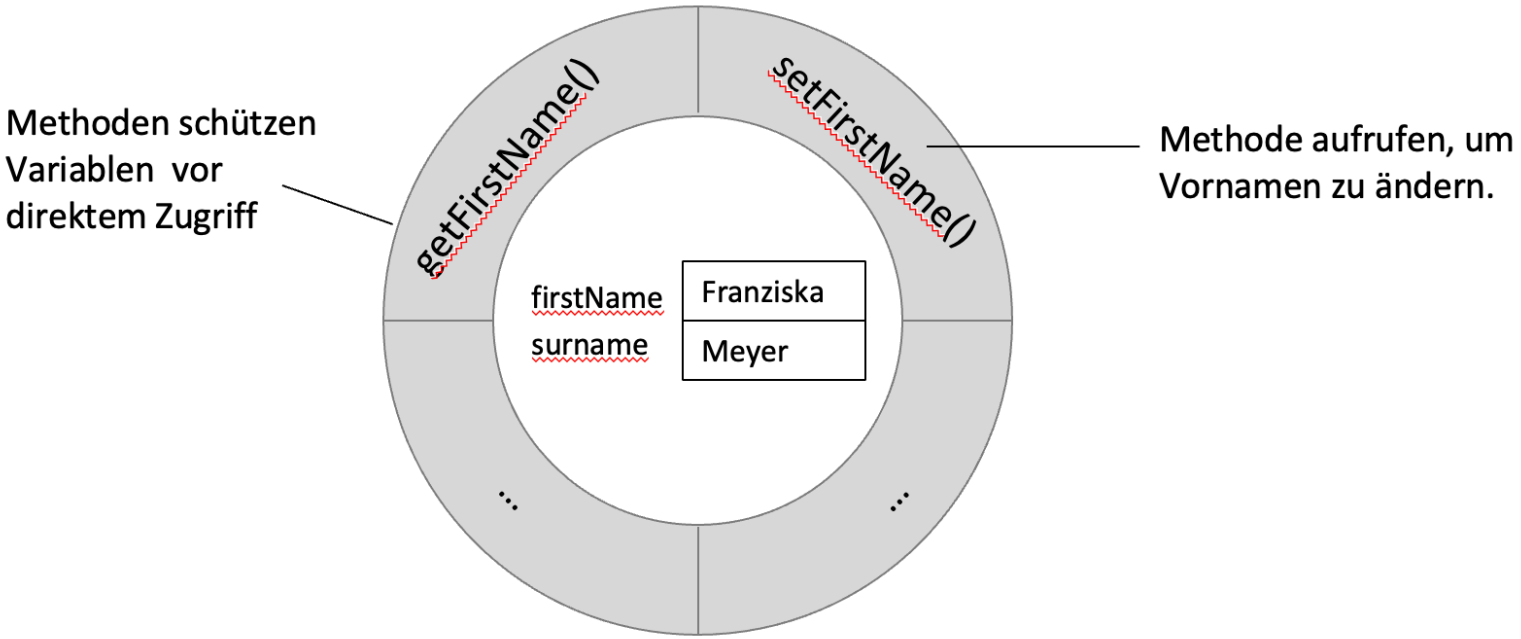


Abbildung 3: Datenkapselung durch Klassen

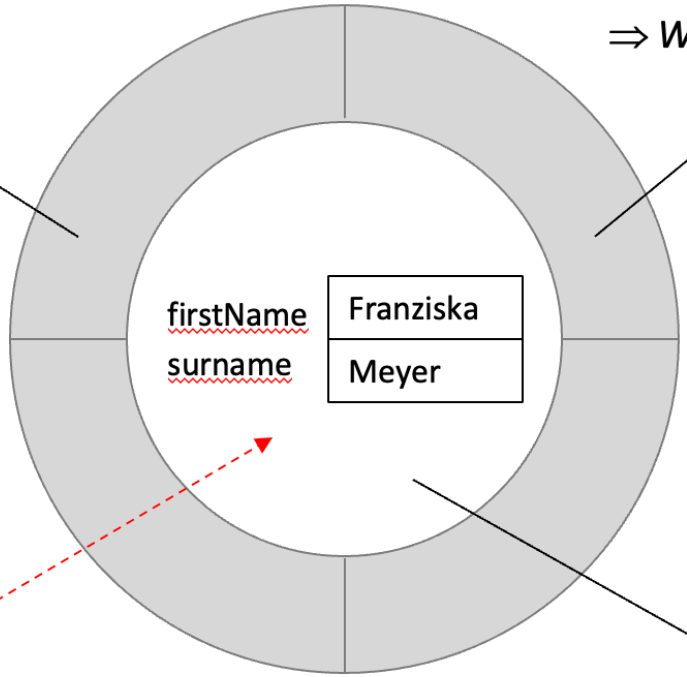
2.5 Datenkapselung

- Mithilfe von Attributen und Methoden von Klassen lassen sich Daten kapseln.
  - Auf gekapselte Daten können nicht alle Teile des Programms zugreifen, was die Sicherheit erhöht.
  - Außerdem lassen sich so Attribute vor fehlerhaften Werten schützen.

2.5 Datenkapselung

1. Über spezielle Methoden (*Konstruktoren*) erzeugt und initialisiert
⇒ *Anfangs* sinnvolle Datenwerte

2. Kontrollierte Änderung der Datenwerte nur über Methoden des Objektes
⇒ *Weiterhin* sinnvolle Datenwerte



Satz an Daten bzw. Variablen (*Zustand*)

Kein direkter Zugriff von außen auf die Variablen

2.5 Datenkapselung

? Frage

Wie ist das im Gegensatz dazu in C?

2.5 Datenkapselung

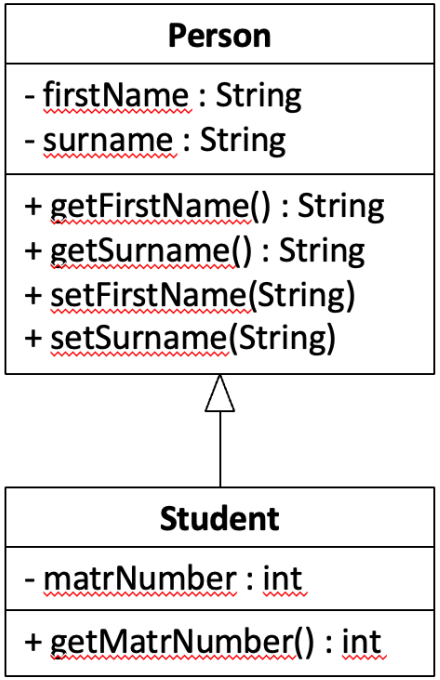
? Frage

Wie ist das im Gegensatz dazu in C?

- Die Datenstruktur (also das **struct**) muss für den Zugriff auf die Elemente öffentlich gemacht werden.
- Die Daten werden nicht geschützt.
- Es gibt keine Zuordnung zwischen den Daten und den Funktionen.

2.6 Vererbung

- Durch Vererbung lassen sich neue Klassen aus anderen Klassen erzeugen.



2.6 Vererbung

- Durch Vererbung lassen sich neue Klassen aus anderen Klassen erzeugen.
- Dabei werden die Methoden und Attribute der Basisklasse übernommen und um weiteren Code erweitert
- Keine duplizierter Code

- Klassen können auch durch andere Klassen zusammengesetzt sein.
- Sowas nennt man dann **Komposition**.
- So wäre zum Beispiel die Klasse **Haus** zusammengesetzt aus zum Beispiel **Fenstern**, **Wänden** und **Türen**.

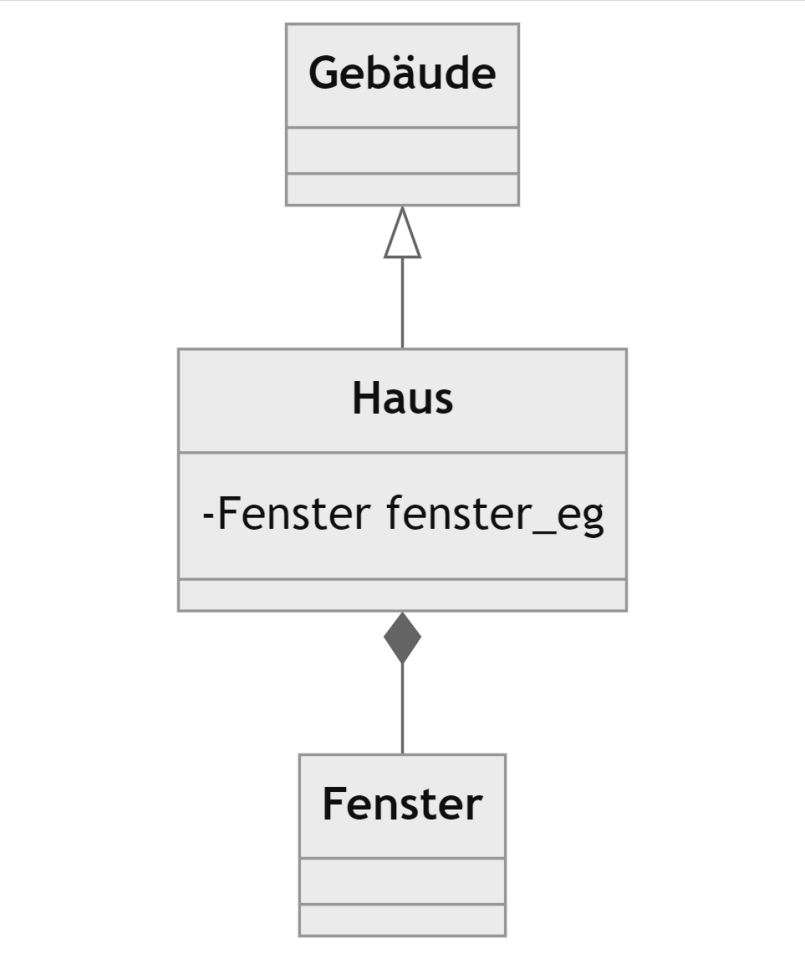


Abbildung 6: Komposition der Klasse **Haus**, die von **Gebäude** erbt

2.7 Ablauf eines Programms in Java

1. Bei Programmstart wird eine besondere Methode **main** im *Hauptobjekt* ausgeführt.
2. In dieser Methode werden dann Objekte erzeugt und die **Referenzen** auf diese Objekte in Variablen gespeichert.
3. Über diese Variablen kann dann auf das jeweilige Objekt zugegriffen werden.
4. Objekte in dem Programm können dann wiederum weitere Objekte erzeugen und Methoden aufrufen.
5. Sobald die **main**-Methode beendet wurde, endet das Programm.

2.7 Ablauf eines Programms in Java

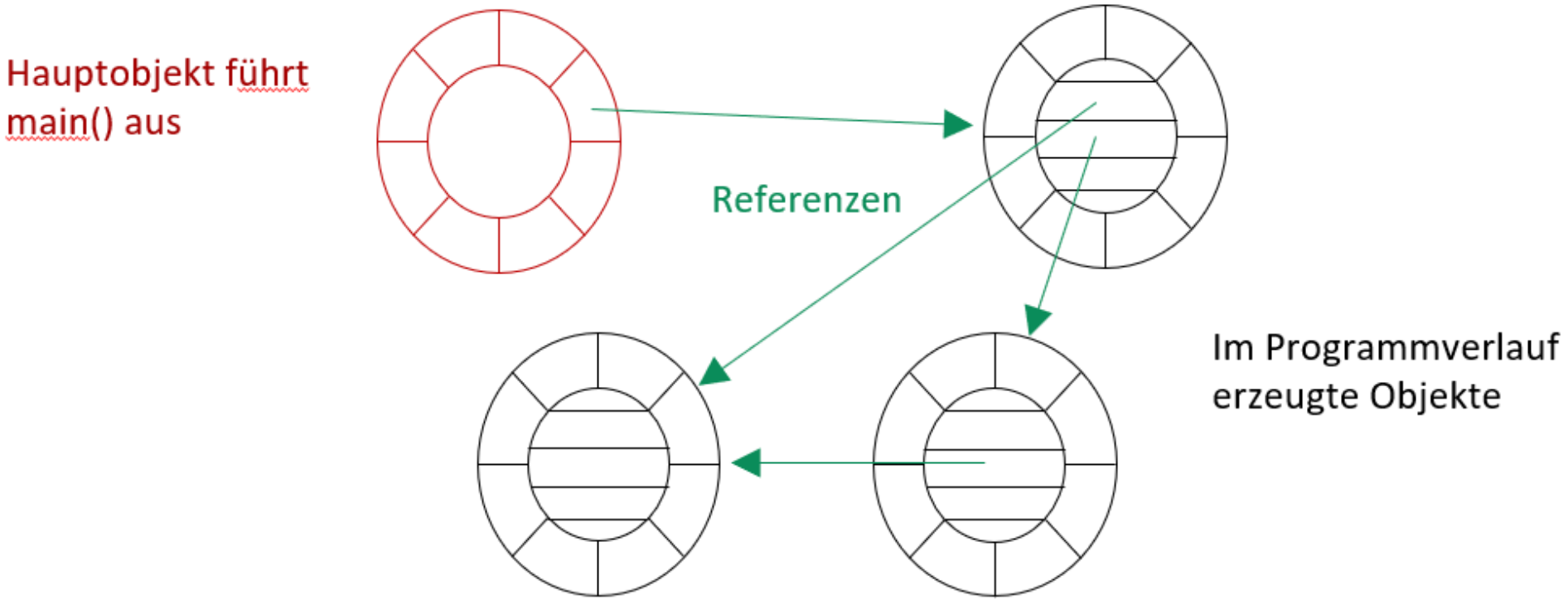


Abbildung 7: Referenzen in der Lebenszeit eines Programms

3. Die Programmiersprache Java

Aufgabe 1

Lassen Sie uns erst einmal ein paar einfache Aufgaben in der Programmiersprache C schreiben:

- Summe der Zahlen 1 bis n über **for**-Schleife.
- Maximum zweier Zahlen über **if**-Anweisung.
- Maximum zweier Zahlen über die Funktion **getMax()** bestimmen.

3.1 Java vs. C

- Ich habe gute Nachrichten: Diesen Code hätten Sie auch in Java problemlos ausführen können!
- Die **Syntax**, also die Schlüsselworte und der Aufbau der Sprache, ist sehr nahe an C und C++!
- Deshalb wollen wir auf Ihre Vorkenntnis weiter aufbauen.

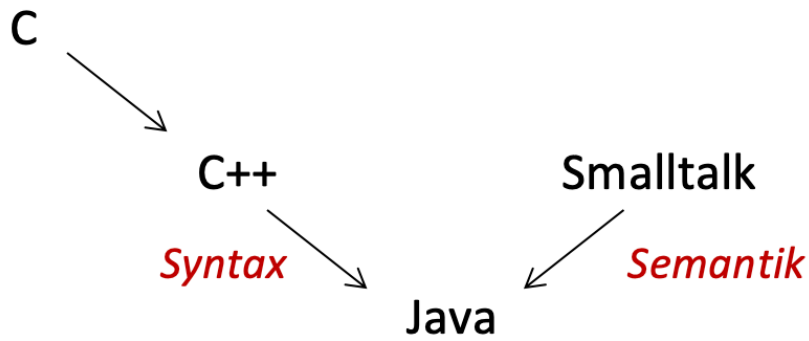


Abbildung 8: Die Einflüsse auf die Programmiersprache Java

3.2 Kompilieren

1. Entwicklung
   - Quelltext wird am PC geschrieben
   - Compiler kompiliert Quelltext in einen **Bytecode**
2. Ausführung
   - Bytecode wird auf **JVM** (Java Virtual Machine) ausgeführt
   - Ausführung benötigt keine neue Kompilierung für die jeweilige Zielplatform

3.2 Kompilieren

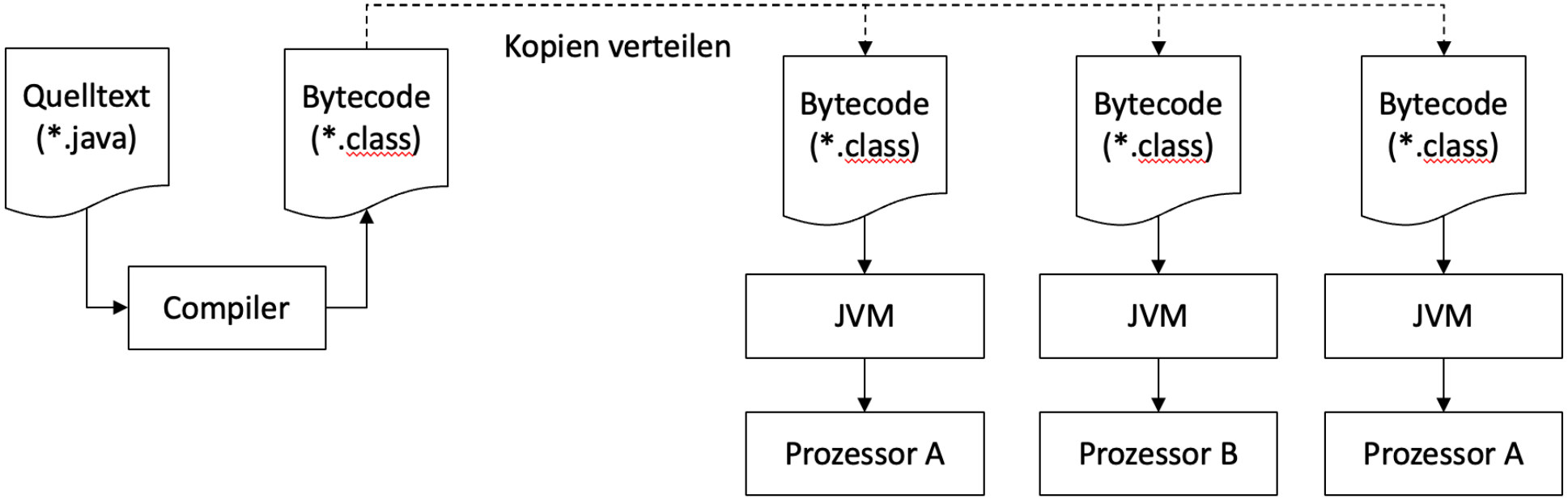


Abbildung 9: Die Ausführung eines Programms mit der JVM

3.2 Kompilieren

- Unterschiede in anderen Programmiersprachen, die kompiliert oder interpretiert sind:
  - **Kompilierte Sprachen** müssen für jede Zielplattform neu kompiliert werden.
  - **Interpretierte Sprachen** müssen durch einen eigenen Interpreter auf der Zielplattform selbst interpretiert werden.

3.2 Kompilieren

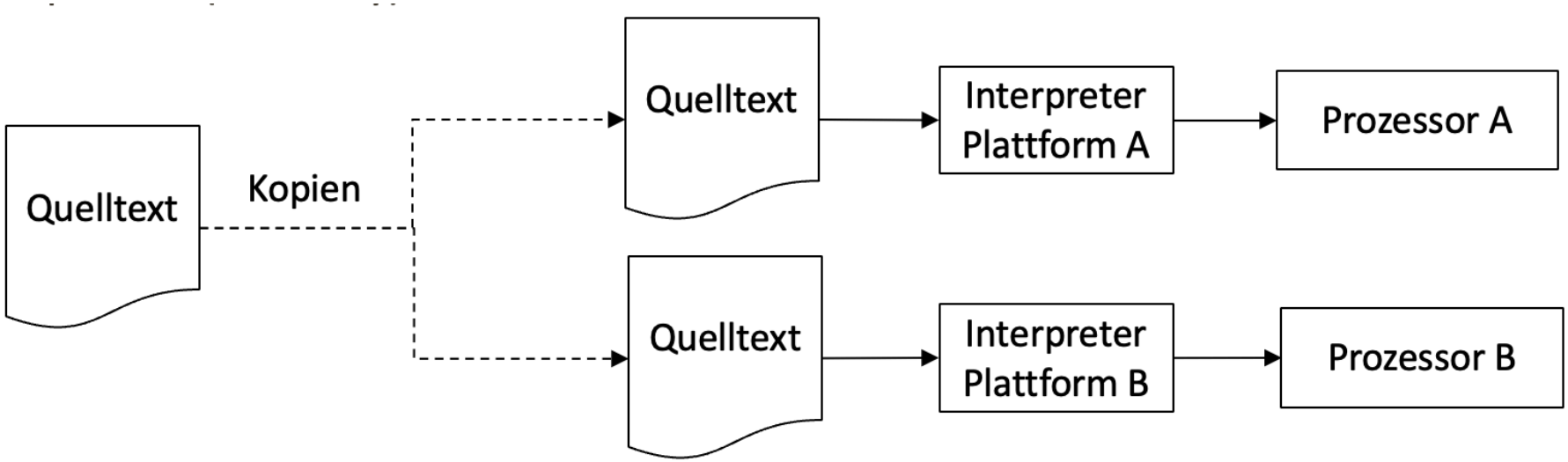


Abbildung 10: Ausführung von kompilierten Sprachen

3.2 Kompilieren

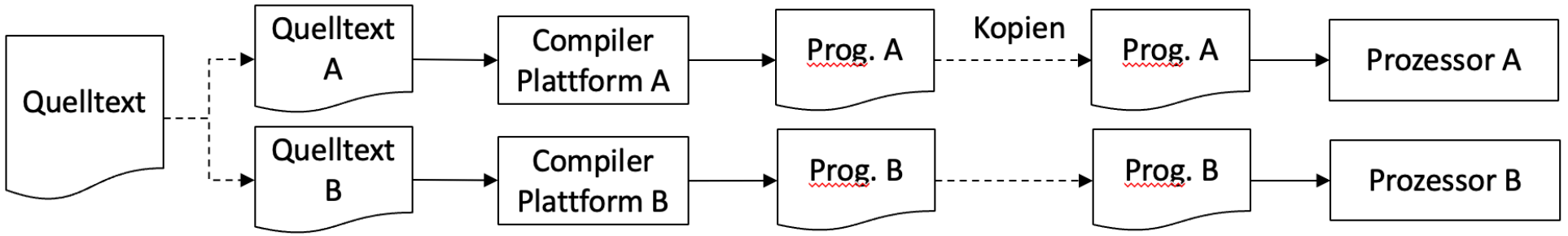


Abbildung 11: Ausführung von interpretierten Sprachen

3.3 Eigenschaften von Java

? Frage

Wenn Sie eine neue Sprache entwicklen könnten, was wäre Ihnen wichtig? Was würden Sie an C/C++ ändern?

3.3 Eigenschaften von Java

? Frage

Wenn Sie eine neue Sprache entwicklen könnten, was wäre Ihnen wichtig? Was würden Sie an C/C++ ändern?

- Java
  - Objektorientierte Sprache (also Klassen, Objekte und Vererbung)
  - Plattformunabhängig (über **JVM**)
  - Stark typisiert (feste Typen wie **int** und **String**)
  - Robust (also Garbage Collector)

3.3 Eigenschaften von Java

? Frage

Welches ist die bessere Programmiersprache: C oder Java?

4. Das erste Programm

4.1 IDE

- Ich würde Ihnen IntelliJ IDEA von JetBrains als IDE empfehlen.
  - Dieses Tool wird auch in der Klausur verwendet werden.
  - Die IDE enthält auch das Java JDK, was sie zum Programmieren brauchen.
- Wählen Sie die Community Edition auf https://www.jetbrains.com/idea/download/?section=windows

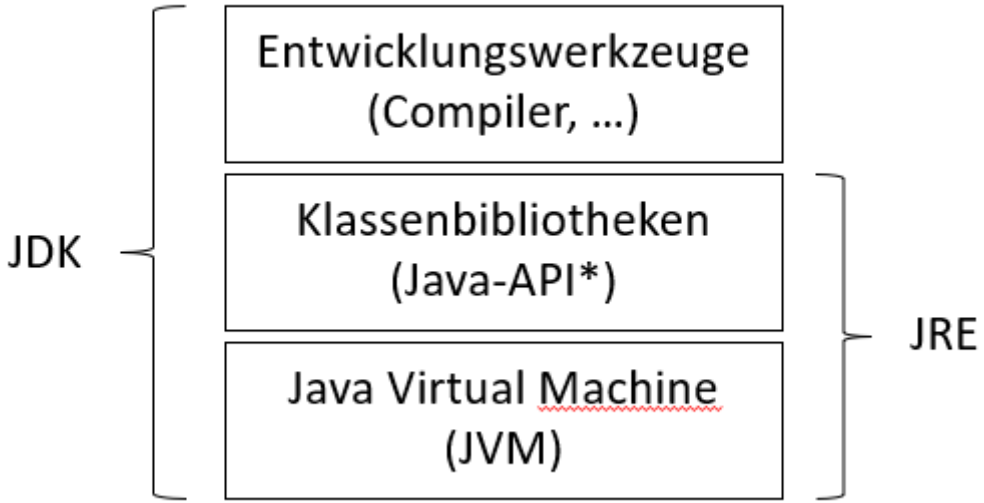


Abbildung 8: Der Aufbau der Java Toolchain*

*Application Programming Interface

Aufgabe 2

- Vorbereitung:
  1. Öffnen Sie zuerst ein Verzeichnis, in dem Sie die Dateien zu Programmieren ablegen werden.
  2. Öffnen Sie IntelliJ IDEA.
- Projekt anlegen:
  1. File > New > Project auswählen.
  2. Vergeben Sie einen Namen und einen Ort.
  3. Wählen Sie Java und IntelliJ und das entsprechende JDK
  4. Auf „Create“ klicken

Aufgabe 3

- Paket erstellen
  1. Rechtsklick auf src
  2. New > Package auswählen
  3. Name eintragen
- Klasse erstellen
  1. Rechtsklick auf Paket
  2. New > Java Class wählen

4.2 Das erste Programm

```java
public static void main(String[] args){
  System.out.println("Hello World!");
}
```

Java

Idee

Tragen Sie den Code in die gerade erstellte Datei ein. Wenn Sie schon soweit sind, programmieren Sie gerne mit!

4.3 Anwendungen in Java

Eine Java-Datei kann ausgeführt werden, wenn sie eine öffentliche (**public**) Klasse hat: `public class MyApplication {...}`

4.3 Anwendungen in Java

Eine Java-Datei kann ausgeführt werden, wenn sie eine öffentliche (**public**) Klasse hat: `public class MyApplication {...}`

Die Klasse muss außerdem den gleichen Namen wie die Datei haben, also bspw. `MyApplication.java`

4.3 Anwendungen in Java

Eine Java-Datei kann ausgeführt werden, wenn sie eine öffentliche (**public**) Klasse hat: `public class MyApplication {...}`

Die Klasse muss außerdem den gleichen Namen wie die Datei haben, also bspw. `MyApplication.java`

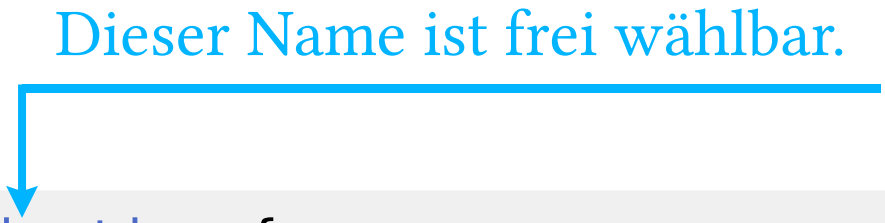
Die Klasse besitzt die Methode: `public static void main(String[] args)`

4.3 Anwendungen in Java

Java

```java
public class MyApplication {
  public static void main(String[] args) {
    System.out.println("Hello World!");
  }
}
```

4.3 Anwendungen in Java



Java

```java
public class MyApplication {
  public static void main(String[] args) {
    System.out.println("Hello World!");
  }
}
```

4.3 Anwendungen in Java

Dieser Name ist frei wählbar.

```java
public class MyApplication {
  public static void main(String[] args) {
    System.out.println("Hello World!");
  }
}
```

Diese Methode muss immer main heißen.

5. Literatur

5.1 Literaturempfehlungen

- Einige Bücher, die Ihnen vielleicht im Verlauf der Veranstaltung helfen könnten:
  - D. Abts: Grundkurs JAVA, Springer-Vieweg
  - H.-P. Habelitz: Programmieren lernen mit Java, Rheinwerk Computing

6. License Notice

6.1 Attribution

- This work is shared under the CC BY-NC-SA 4.0 License and the respective Public License
- https://creativecommons.org/licenses/by-nc-sa/4.0/
- This work is based off of the work Prof. Dr. Marc Hensel.
- Some of the images and texts, as well as the layout were changed.
- The base material was supplied in private, therefore the link to the source cannot be shared with the audience.