# Objektorientierte Programmierung in Java

## Vorlesung 3 - Klassen und Objekte

Emily Lucia Antosch

HAW Hamburg

10.10.2024

# Inhaltsverzeichnis

# 1. Einleitung

# 1.1 Wo sind wir gerade?

- Zuletzt haben wir uns mit den imperativen Konzepten der Programmiersprache Java beschäftigt.
- Sie können nun
  - einfache Datentypen in Java verwenden,
  - den Programmfluss mit Kontrollstrukturen und Schleifen steuern und
  - Datentypen konvertieren.
- Heute geht es um **Klassen und Objekte**.

# 1.1 Wo sind wir gerade?

1. Imperative Konzepte
2. **Klassen und Objekte**
3. Klassenbibliothek
4. Vererbung
5. Schnittstellen
6. Graphische Oberflächen
7. Ausnahmebehandlung
8. Eingaben und Ausgaben
9. Multithreading (Parallel Computing)

# 1.2 Das Ziel dieses Kapitels

- Sie implementieren Klassen und Objekte in Java, um reale Dinge abzubilden.
- Sie erzeugen Objekte einer Klasse und ändern deren Zustand über Operationen.
- Sie wenden zusätzliche Programmierrichtlinien an, um die Qualität und die Wartbarkeit Ihres Codes zu verbessern.

# 2. Klassen und Objekte

# 2.1 Klasse

- Eine **Klasse** ist ein Bauplan für Objekte. Sie enthält
  - **Attribute** (Datenfelder) und
  - **Methoden** (Operationen).
- Zusammen heißen Attribute und Methoden **Members**.

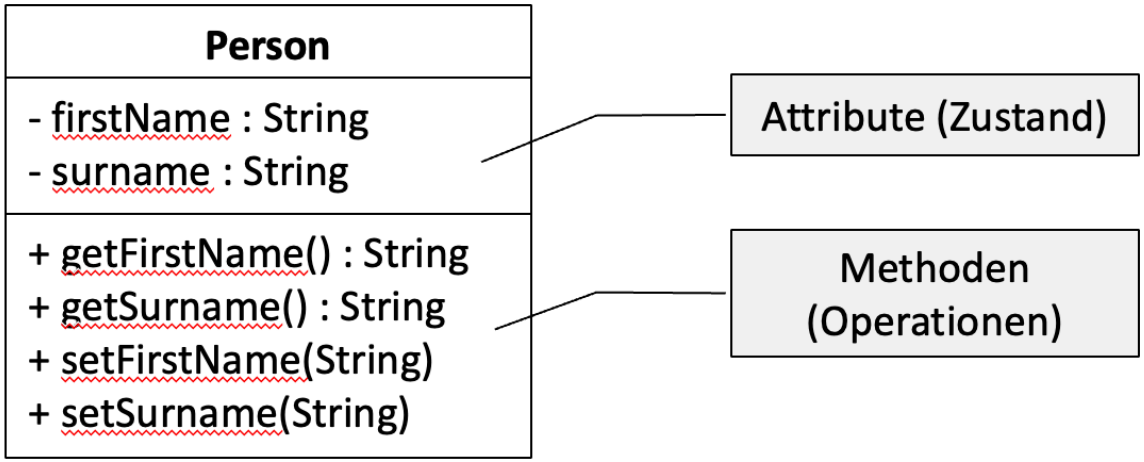



Abbildung 1: UML-Notation einer Klasse Person

# 2.1 Klasse

- Zur Laufzeit im Speicher erzeugter Datensatz einer Klasse
- Variablen beschreiben **Zustand** des Objekts
- Methoden beschreiben **Fähigkeiten** des Objekts
- Bezeichnungen für Variablen: **Attribute**, **Objektvariablen**, **Instanzvariablen**

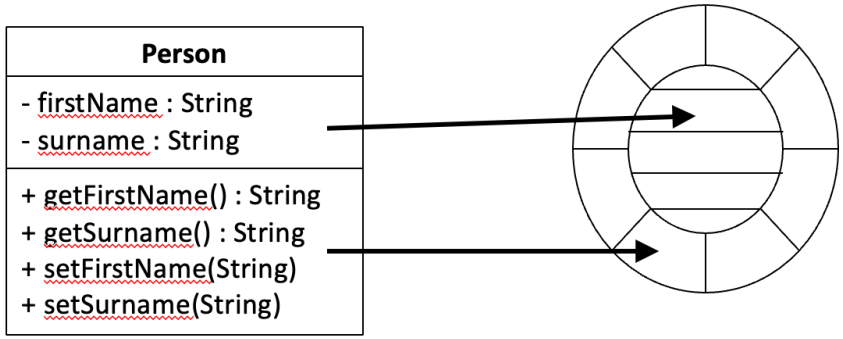



Abbildung 2: Aufteilung Methoden und Attribute

# 2.2 Zusammenhang Klasse und Objekt

- Klasse: Beschreibung („Bauplan“) eines Datentyps
- Objekt einer Klasse: Erzeugtes Element des Datentyps
- Es können beliebig viele Objekte einer Klasse erzeugt werden.

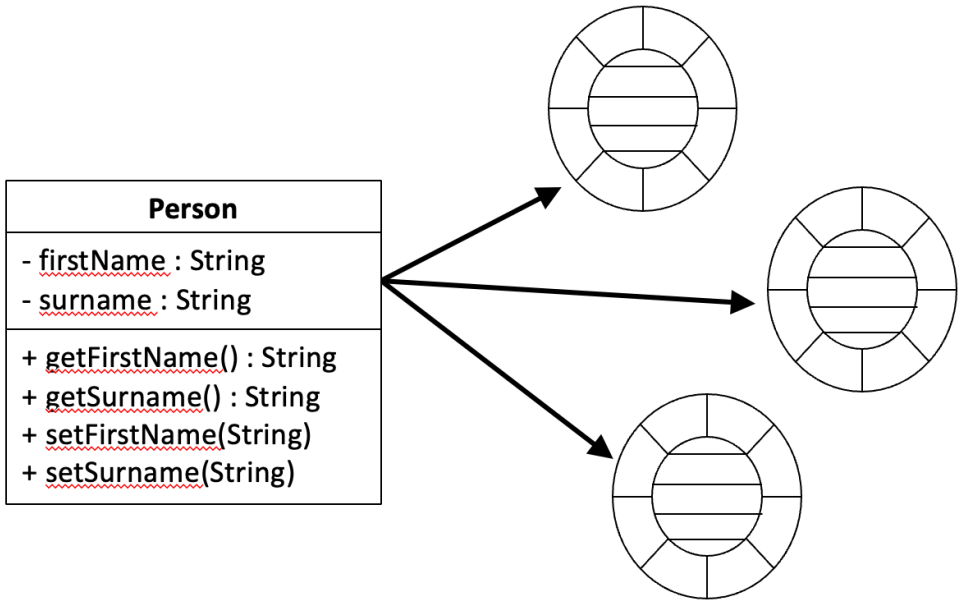



Abbildung 3: Mehrere Objekte aus einer Klasse

# 2.3 Klassen in Java

- Klassen können über den folgenden Code deklariert werden:

```java
class Klassenname {
    Attribute
    Methoden
    }
```

> 🔥 **Tipp**
>
> Legen Sie jede Klasse in einer eigenen Datei an!

# 2.4 Beispiel: Einfache Klasse

## Aufgabe 1

Lassen Sie uns diese einfache Klasse erstellen:

- Klasse Student, beschrieben durch Name, Matrikelnummer und Studienbeginn (in Jahren)

# 2.4 Beispiel: Einfache Klasse

# 2.4 Beispiel: Einfache Klasse

## Aufgabe 2

Lassen Sie uns diese einfache Klasse erstellen:

- Klasse Student, beschrieben durch Name, Matrikelnummer und Studienbeginn (in Jahren)

```java
class Student {
    String name;
    int matrNumber;
    int enrolledYear;
}
```

# 2.4 Beispiel: Einfache Klasse

- Die Klasse hat weder Methoden noch eine Datenkapselung gegen Einfluss von außen.

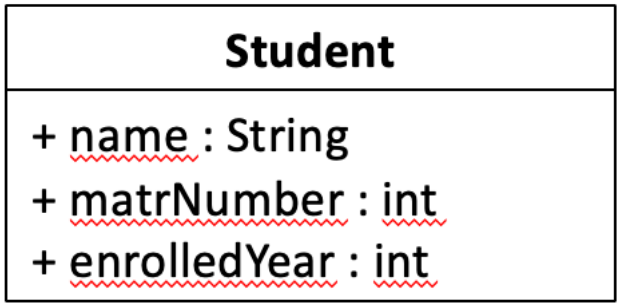



Abbildung 4: UML-Darstellung der Klasse, die wir eben erstellt haben

# 2.5 Beispiel: Eine Klasse, viele Objekte

- Klasse („Eine Klasse für alle Studierenden“):
  - Die Klasse ist ein neuer Datentyp.
  - Legt fest, durch welche Daten Studierende beschrieben werden
- Objekte („Für jede/n Studierende/n ein eigenes Objekt“):
  - Objekte sind Instanzen im Speicher.
  - Besitzen Struktur der Klasse, sind aber mit Daten gefüllt
  - Es können beliebig viele Objekte erzeugt werden.

# 2.5 Beispiel: Eine Klasse, viele Objekte

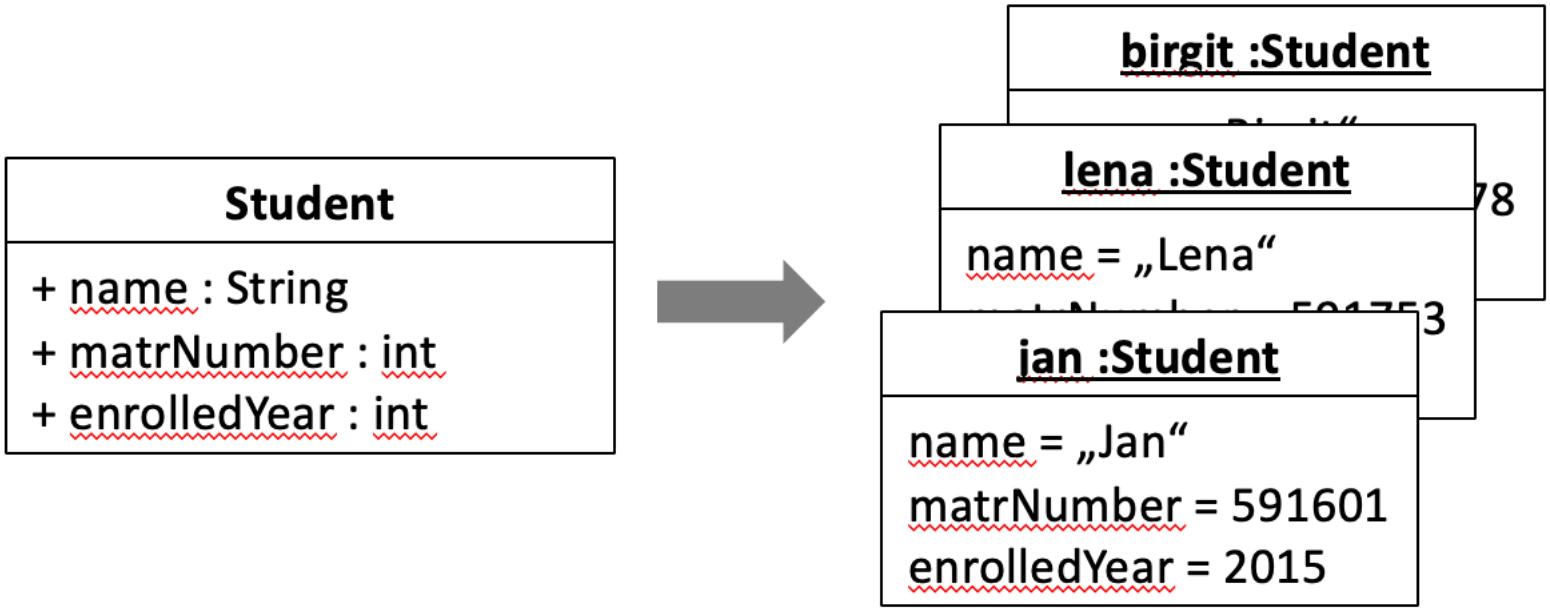



Abbildung 5: Aus einer Klasse lassen sich mehrere Objekte erstellen

# 2.6 Beispiel: Lokale Variablen

**? Frage**

Welche Werte haben die Variablen `count`, `jan` und `lena`?

Java

```java
public class StudentDemo {
    public static void main(String[] args) {
        int count;
        Student lena, jan;
    }
}
```

# 2.6 Beispiel: Lokale Variablen

## ? Frage

Welche Werte haben die Variablen `count`, `jan` und `lena`?

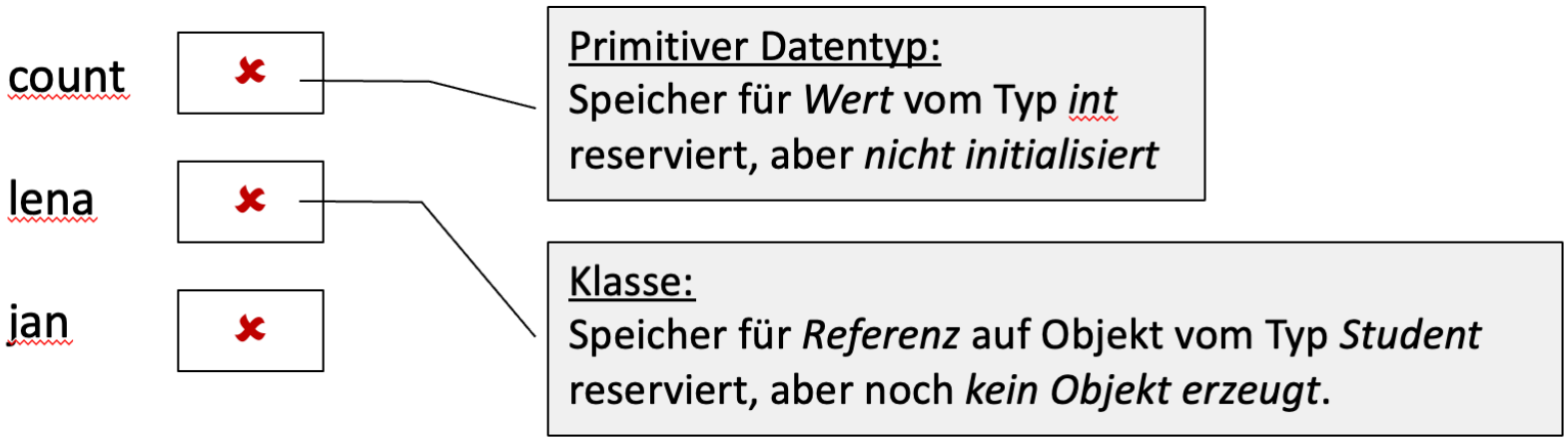


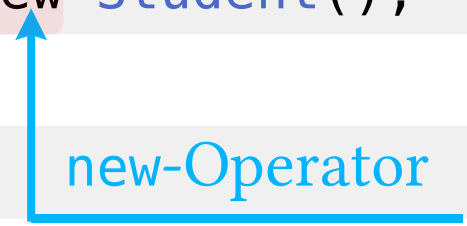

Abbildung 6: Primitive Datentypen vs. Objekte

# 2.7 Beispiel: new-Operator

- Objekte werden durch den new-Operator erzeugt.

Java

```java
public class StudentDemo {
    public static void main(String[] args) {
        int count;
        Student lena, jan;
        lena = new Student();
    }
}
```



# 2.7 Beispiel: new-Operator

- Schritt 1: new-Operator erzeugt Objekt.
  - Speicherplatz für Objekt (mit Objektvariablen) reservieren.
  - Objektvariablen mit Standardwerten initialisieren (mehr dazu gleich).

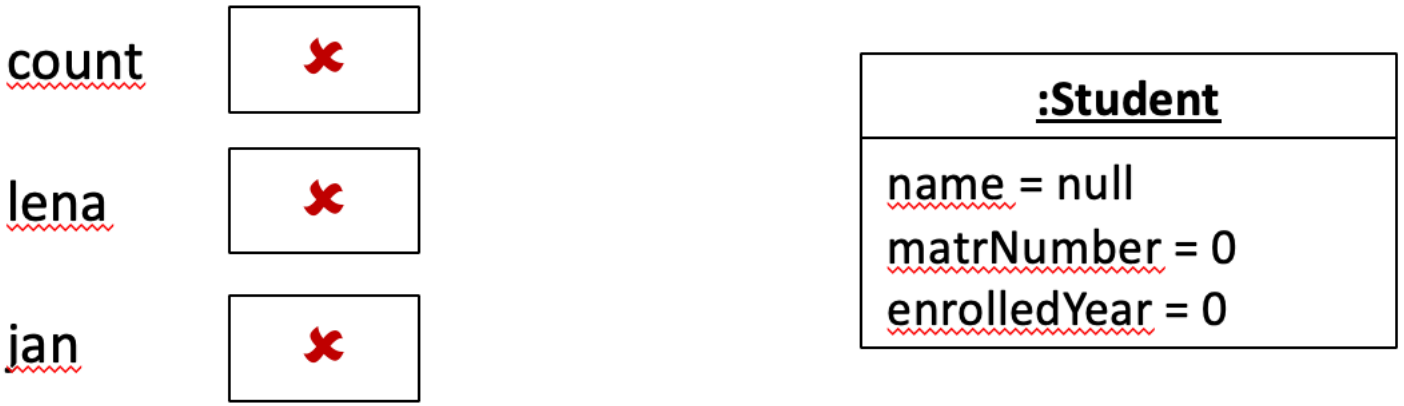



Abbildung 7: Erstellen von Referenz mit new

# 2.7 Beispiel: new-Operator

- Schritt 2: Zuweisung
  - Schreibt Referenz („Adresse“) des neuen Objekts in Variable `lena`.
  - Ist unabhängig vom new-Operator und der Erzeugung des Objekts

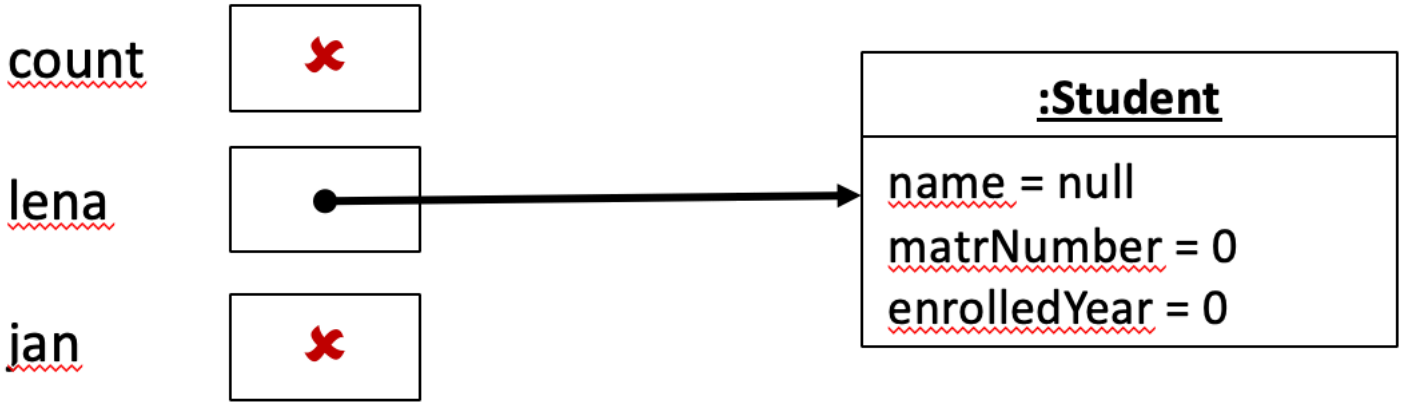



Abbildung 8: Zuweisung von Referenz an Variable

# 3. Variablen und Speicher

# 3.1 Zielsetzung

## Zusammenfassung

- Das haben wir uns bereits angeschaut:
  - Was sind Klassen und Objekte?
  - Wie deklariert man Klassen?
  - Wie erzeugt man Objekte?

- Im Folgenden wollen wir uns folgende Aspekte anschauen:
  - Zugriff auf Objektvariablen
  - Initialisierung von Objektvariablen
  - Zuweisung von Referenzen
  - Automatische Speicherbereinigung

# 3.2 Zugriff auf Objektvariablen

- Zugriff auf Objektvariablen erfolgt mittels des Punkt-Operators:

```java
Objektreferenz.Member
```

- Dabei ist die `Objektreferenz` eine Referenz auf ein Objekt, die in einer Variable gespeichert ist.
- `Member` ist z.B. ein Attribut/Objektvariable

# 4. License Notice

# 4.1 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.