

Klassen und Objekte

Programmierung und Softwareentwicklung
Hörsaalübung



Allgemeines

Was ist Java?

- Objektorientierte Programmiersprache
- Syntax ähnlich zu C

Was ist objektorientiert?

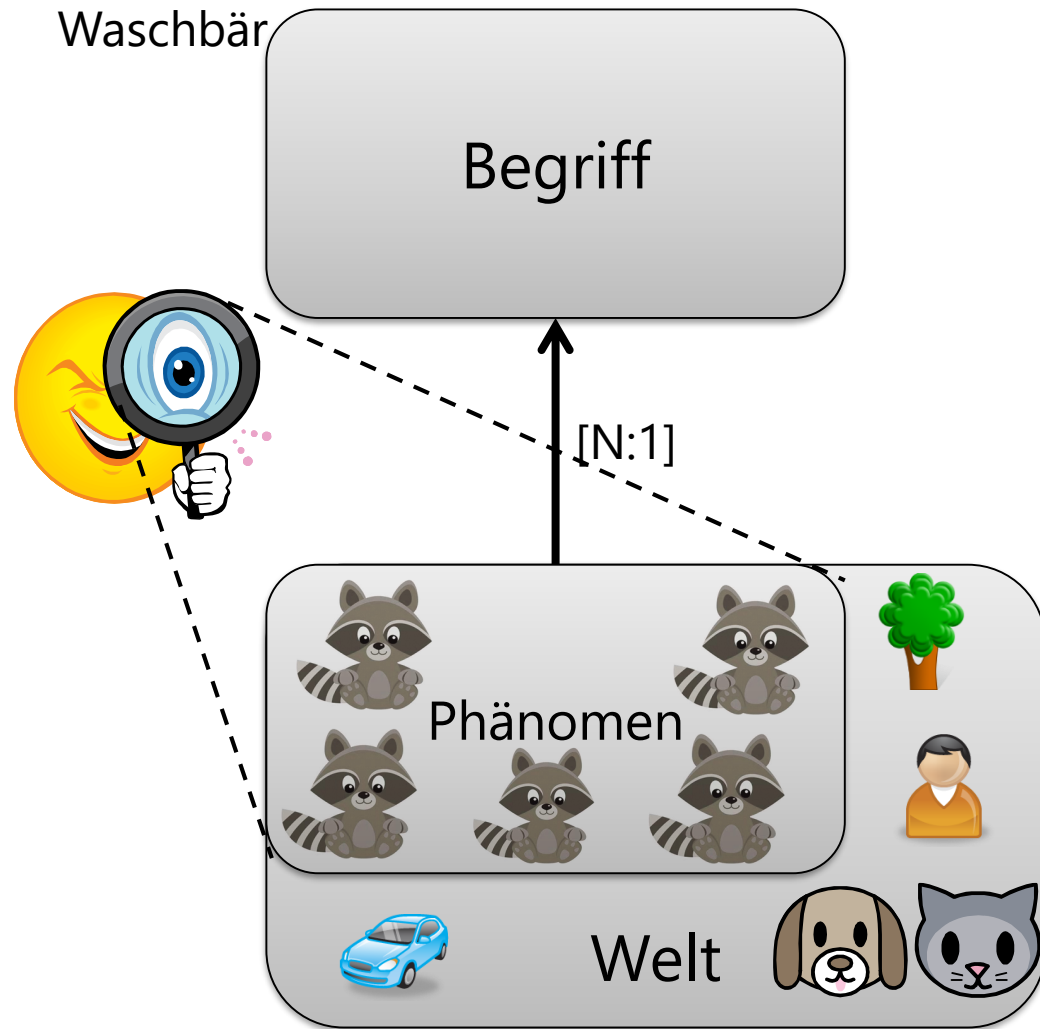
- Unterteilung der komplexen Systeme in Klassen und Objekten
- Möglichkeit der Vererbung

Wie programmiere ich objektorientiert?

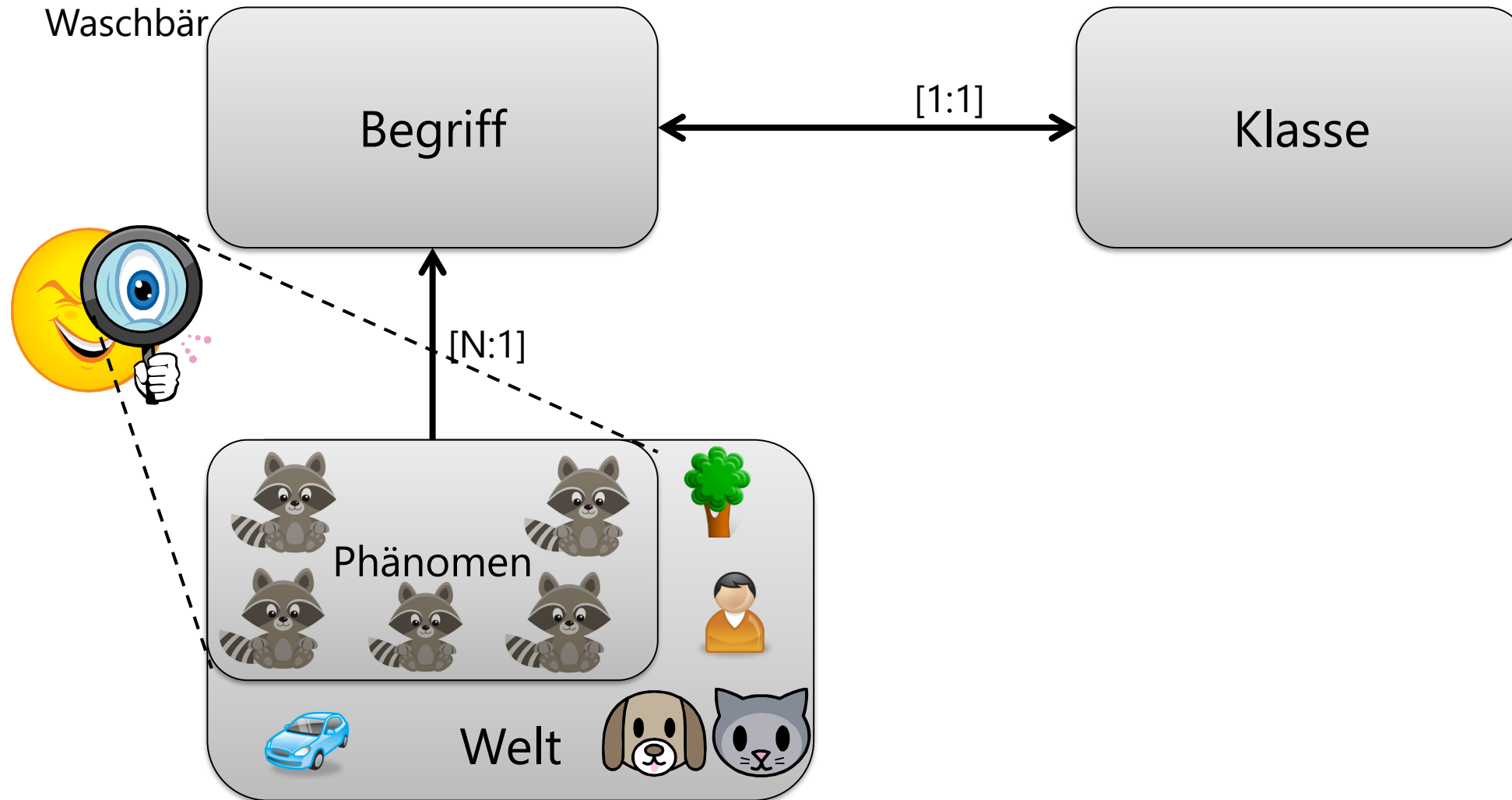
- Klasse erstellen und implementieren
- Objekte der Klasse instanziiieren und Werte zuweisen
- Operationen auf den Objekten aufrufen

Objektorientierung

Realwelt vs. Modell



Realwelt vs. Modell



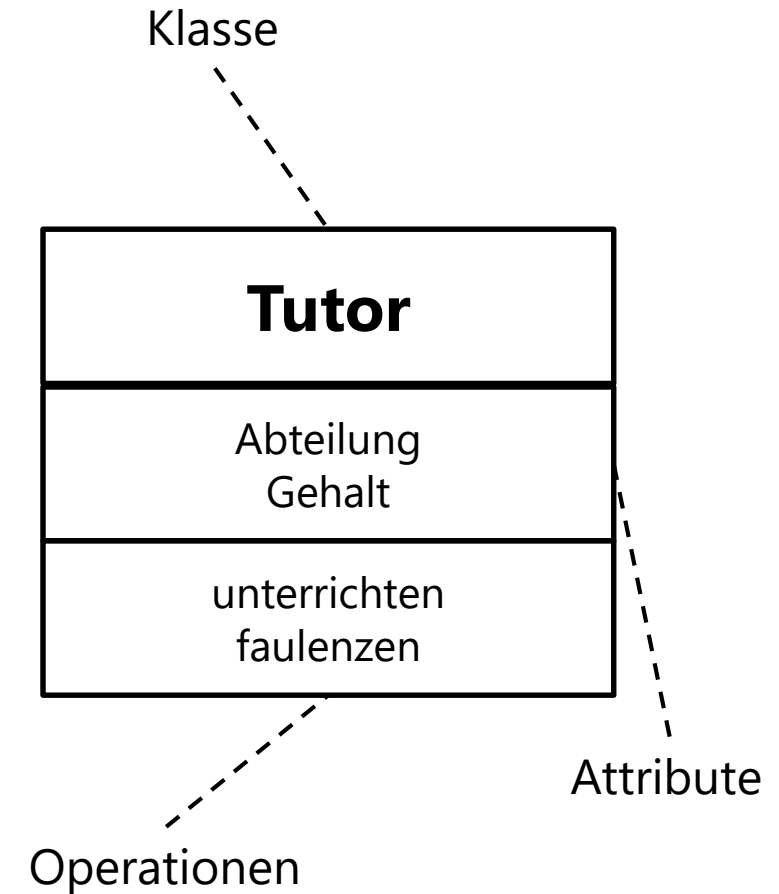
Klasse

Klasse

- Datentyp der Objekte
- Definiert Eigenschaften (Attribute) der Objekte
- Definiert Fähigkeiten (Operationen) der Objekte

Darstellung

- Vereinfachtes „Klassendiagramm“
- Rechteck mit Bereich für Klassennamen, Attribute und Operationen



Für den Waschbär

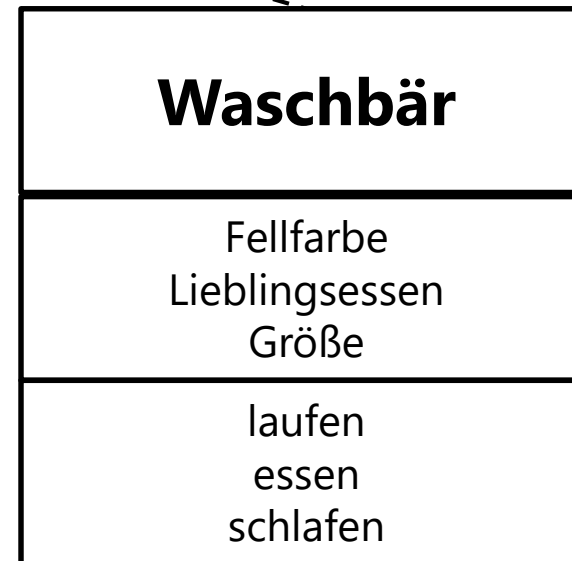
Waschbär

- Eigenschaften
 - Fellfarbe
 - Lieblingsessen
 - Größe
- Fähigkeiten
 - Laufen
 - Essen
 - Schlafen



我是一只浣熊

Klasse



----- Attribute

----- Operationen

Nur ein Waschbär? ☹️

Nein 😊

- Von einer Klasse können beliebig viele Objekte erzeugt werden
- Klasse dient nur als Bauplan
 - Objekte ist konkrete Instanz des Bauplans

Waschbär
Fellfarbe Lieblingssessen Größe
laufen essen schlafen



Nur ein Waschbär? ☹️

Nein 😊

- Von einer Klasse können beliebig viele Objekte erzeugt werden
- Klasse dient nur als Bauplan
 - Objekt ist konkrete Instanz des Bauplans

Rocket:Waschbär

Fellfarbe = „grau-braun“
Lieblingssessen = „Weichtiere“
Größe = 60



Fluffy:Waschbär

Fellfarbe = „grau-schwarz“
Lieblingssessen = „Obst“
Größe = 50



浣熊:Waschbär

Fellfarbe = „grau-schwarz“
Lieblingssessen = „alles“
Größe = 60

Objekt / Instanz

Attribute

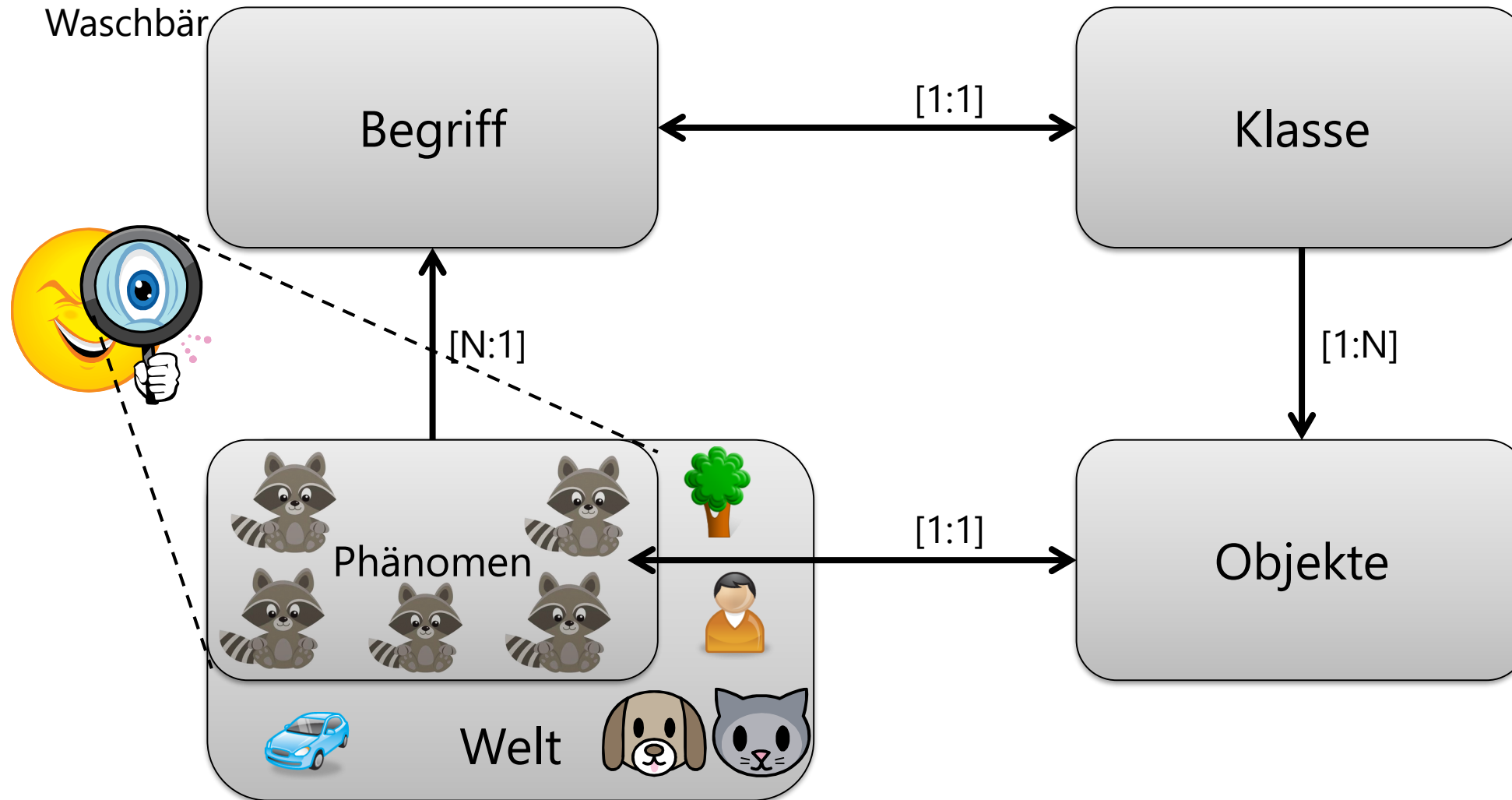


Objekt

Objekt

- Konkrete Instanz einer Klasse
 - Jedes Objekt besitzt genau eine Klasse
- Besitzt alle Operationen der Klasse
- Jedes Objekt ist einzigartig
- Attribute erhalten individuelle Werte
 - Typ der Attribute in Klasse festgelegt
 - Werte können verändert werden
- Können mit anderen Objekten interagieren

Realwelt vs. Modell



Klassen

- Was ist eine Klasse?
- Was gehört zur Deklaration einer Klasse?

Klassen

Was ist eine Klasse?

- Verallgemeinerte Sammlung von Eigenschaften

Was gehört zur Deklaration einer Klasse?

Klassen

Was ist eine Klasse?

- Verallgemeinerte Sammlung von Eigenschaften

Was gehört zur Deklaration einer Klasse?

- Name
- Attribute
- Operationen

Objekte

- Was ist ein Objekt?
- Gibt es Objekte mit undefinierten Werten der Attribute?
- Was haben Objekte für einen Zweck?

Objekte

Was ist ein Objekt?

- Konkrete Instanz einer Klasse

Gibt es Objekte mit undefinierten Werten der Attribute?

Was haben Objekte für einen Zweck?

Objekte

Was ist ein Objekt?

- Konkrete Instanz einer Klasse

Gibt es Objekte mit undefinierten Werten der Attribute?

- Nein, in Java werden alle Attribute vorbelegt

Was haben Objekte für einen Zweck?

Objekte

Was ist ein Objekt?

Gibt es Objekte mit undefinierten Werten der Attribute?

Was haben Objekte für einen Zweck?

- Sie können die Operationen der Klasse mit bestimmten Werten aufrufen
- Es können beliebig viele Objekte einer Klasse erstellt werden

Zwei Sichten eines Objektes

Zwei Sichten

- Ein Objekt hat **Daten**
- Ein Objekt ist eine Maschine, die **Operationen** anbietet
- Ein Objekt hat eine **Entität**
 - Entität identifiziert das Objekt eindeutig
 - Entität gibt einem abstrakten Objekt einen Namen

Entität:
Sandro

Operationen:

move
turnLeft
pickFood
...



Daten:

foodCount=1
Position=(1,2)
...

Operationen, die eine Maschine bereitstellt, greifen auf Objektdaten zu und modifizieren diese für ein identifiziertes Objekt

Objektorientierte Programmierung

Was bringen uns die Objekte?

- Objekte besitzen Fähigkeiten (Operationen), die ausgeführt werden können
- Objekte können mit anderen Objekten interagieren
- Wir können Eigenschaften (Attribute) der Objekte verändern
- Jedes Objekt ist einzigartig
 - Mehrere Objekte können zu derselben Klasse gehören

Es ist in der realen Welt nicht natürlich, dass es von einer Klasse nur eine Instanz gibt

- OOP ermöglicht ein realistischeres Modell
- Leichter vorzustellen

Primitive Datentypen

Was sind primitive Datentypen?

- Auch in Java gibt es Typen, die keine Objekte von Klassen sind.
- Zahlen, Zeichen und Wahrheitswerte

Welche primitive Datentypen gibt es?

- Ganzzahlen: byte, short, int, long
 - Fließkommazahlen: float, double
 - Wahrheitswerte: boolean
 - Zeichen: char
- String ist **KEIN** primitiver Datentyp

Aufgabe 1

Ein Taxiunternehmen besitzt einen CEO, Namen und mehrere Taxen. Jedes Taxi besitzt einen Fahrer, ein Taxometer, eine Fahrzeugnummer und kann mehrere Fahrgäste transportieren. Jeder Fahrer benötigt einen Taxi- und einen Führerschein. Zu jeder Fahrt gehört ein Taxi, eine Strecke, die das Taxi zurücklegt und der Fahrtpreis, der für die Strecke fällig ist.

- a) Geben Sie alle Klassen mit den zugehörigen Attributen und Operationen an
- b) Erstellen Sie für jede Klasse ein konkretes Objekt

Aufgabe 1 – Lösung (I)

Taxiunternehmen
CEO : Person namen : String taxen : List<Taxi>

Taxi
fahrer : Fahrer taxometer : int fahrzeugnummer : int

Fahrt
preis : int taxi : Taxi strecke : int

Fahrer
name : String fid : Fuehrerschein tid : Taxischein

Person

Fuehrerschein

Taxischein

Aufgabe 1 – Lösung (II)

<u>Toto: Taxiunternehmen</u>
CEO = HansDieter namen = Toto taxen = [taxi1]

<u>taxi1 : Taxi</u>
fahrer = Peter taxometer = 2 fahrzeugnummer= 33

<u>stmn : Fahrt</u>
taxometer = 2 taxi = taxi1 strecke = 32

<u>Peter : Fahrer</u>
name = Peter fid = fs tid = ts

<u>HansDieter : Person</u>

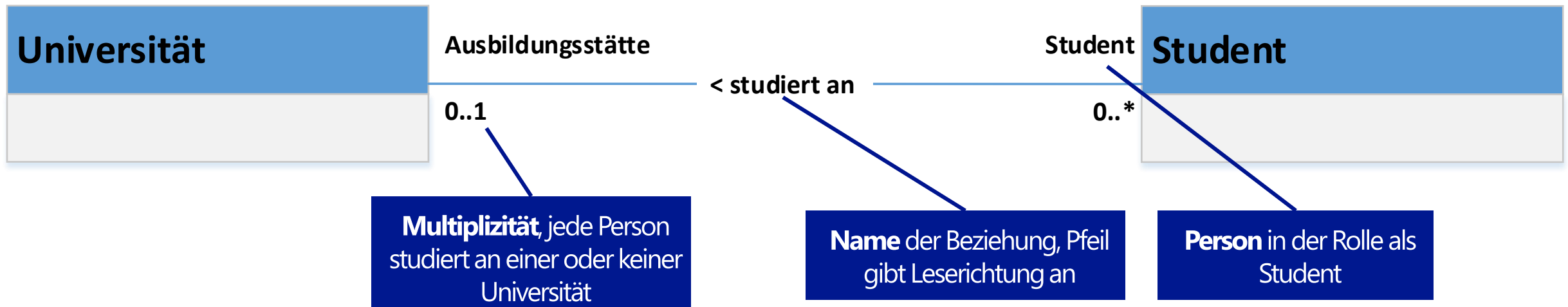
<u>fs : Fuehrerschein</u>

<u>ts : Taxischein</u>

Beziehungen zwischen Klassen

Assoziationen

- Klassen können in Beziehung zueinander stehen
 - Z.B.: n Studierende sind an 1 Universität eingeschrieben
- Beziehungen (Verbindungen) werden als Assoziationen bezeichnet
- Darstellung durch Striche zwischen den Rechtecken der Klassen
 - Beinhaltet eine Beschreibung der Assoziation

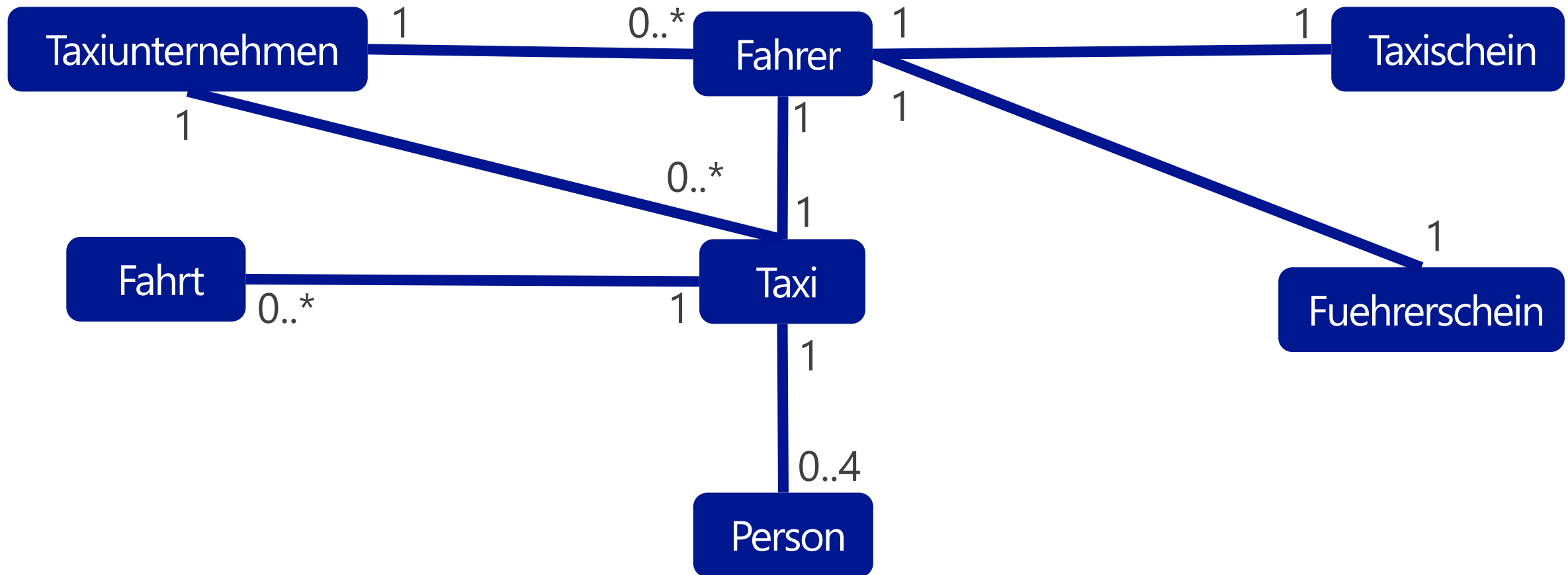


Aufgabe 1

Ein Taxiunternehmen besitzt einen CEO, Namen und mehrere Taxen. Jedes Taxi besitzt einen Fahrer, ein Taxometer, eine Fahrzeugnummer und kann mehrere Fahrgäste transportieren. Jeder Fahrer benötigt einen Taxi- und einen Führerschein. Zu jeder Fahrt gehört ein Taxi, eine Strecke, die das Taxi zurücklegt und der Fahrtpreis der für die Strecke fällig ist.

c) Geben Sie die Beziehungen zwischen den Klassen an

Aufgabe 1 – Lösung (II)



Syntax

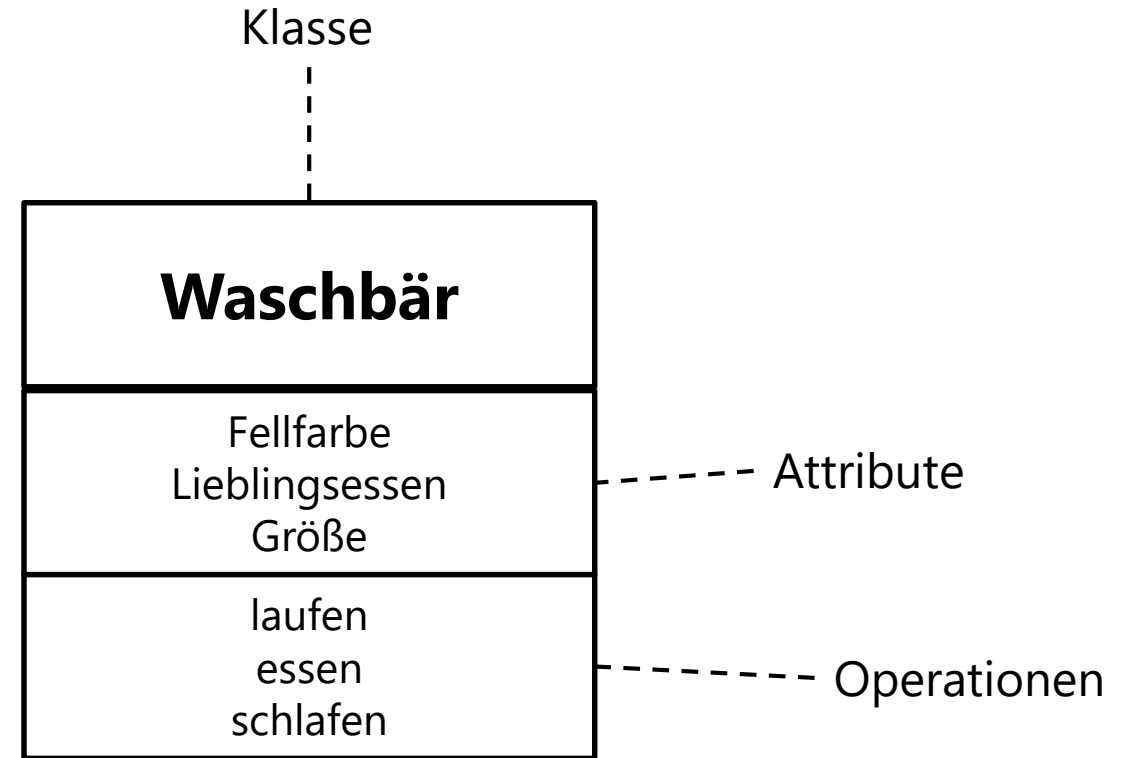
Struktur einer Java Klasse

Klassenstruktur

Klasse

- Schlüsselwort `class`
- Klassenname
- Block

```
public class Raccoon {  
    // Attributes  
  
    // Operations  
}
```



Klassenstruktur

Klassenname

- Styleguide: PascalCase für Klassen- und Schnittstellennamen

Block

- Beinhaltet Attribute und Operationen
- Definiert Funktionalität der Klasse

Pakete

- Was sind Pakete?
- Wozu dienen Pakete?

Pakete

Was sind Pakete?

- Sammlungen von Klassen unter einem bestimmten Namen
- Styleguide: Name als umgedrehte Domain

Wozu dienen Pakete?

Pakete

Was sind Pakete?

- Sammlungen von Klassen unter einem bestimmten Namen

Wozu dienen Pakete?

- Zur Strukturierung größerer Programme

Klassenstruktur

Pakete

- Paket befindet sich als erstes in der Datei

```
package de.unistuttgart.pse.hu;  
  
public class Raccoon {  
    // Attributes  
  
    // Operations  
}
```

Programmcode

Klasse

```
public class Raccoon extends Pet {  
    // ...  
    public void move() {  
        // ...  
    }  
}
```

Name der Klasse

Programmcode

```
public class Raccoon extends Pet {  
    // ...  
    public void move () {  
        // ...  
    }  
}
```

Erweitert eine
existierende Klasse
(Pet)

Programmcode

Blockanfang

```
public class Raccoon extends Pet {  
  
    // ...  
  
    public void move () {  
        // ...  
    }  
}
```

Block/
Körper

Blockende

Programmcode

```
public class Raccoon extends Pet {  
    private Color colorOfFur;  
    private double size;  
  
    // ...  
}
```



Attribute

Klassenstruktur

Attribute

- Datentyp
 - Name
 - Wert
- ```
private Color colorOfFur = Color.gray;
private double size = 1;
private Food lovelyFood = new Food("Cake");
```

## Variablen

- Wie Attribute, nur innerhalb von Operationen
- Keine Sichtbarkeit (das `public`, `private` Zeug)

# Programmcode

```
public class Raccoon extends Pet {
```

```
// ...
```

```
public void eat(final Food food) {
```

```
// ...
```

```
}
```

```
public void move() {
```

```
// ...
```

```
}
```

```
}
```

Operationen

# Programmcode

```
public class Raccoon extends Pet {

 // ...
 public void eat (final Food food) {
 // ...
 }

 public void move () {
 // ...
 }
}
```

Namen der  
Argumente sind  
nicht Teil der  
Signatur

Operationensignatur

Schlüsselwörter (class, extends, void, final, public, private, ...) haben eine spezielle Bedeutung

# Klassenstruktur

## Operationen

- Operationssignatur
  - Definiert die Rahmen der Operation nach außen
  - Beinhaltet Typ der Rückgabe
  - Beinhaltet Argumente
- Block
  - Beinhaltet Logik der Operation

```
public void walk () {
 // some implementation
}
```

```
public void eat (Food food) {
 // some implementation
}
```

# Anweisung

Fundamentales Prinzip der objektorientierten Programmierung: Führe eine Operation auf einem Objekt aus

```
public void moveRaccoon3Steps (final Raccoon raccoon) {
 raccoon.move ();
 raccoon.move ();
 raccoon.move ();
}
```

; als Abschluss einer  
Instruktion

Objekt (Ziel des  
Operationsaufrufs)

Operation, die auf dem  
Ziel ausgeführt wird

# Programmcode-Formatierung

- Einrückungen, um das Programm zu formatieren
  - Eine Ebene = 1 Tab oder 4 Spaces
- Leerzeichen füllen Bereiche zwischen Code
  - Spaces, Tabs, Newlines, ...
  - Kein Whitespace vor oder nach einem Aufruf einer Methode
    - » Also kein: `entity. move()` oder `entity .move()`
- Kommentare zur Erläuterung schwieriger Code-Abschnitte oder Operationen
  - Erleichtert Lesbarkeit und Wartbarkeit des Codes
- Formatierungen werden ignoriert
  - Keine Auswirkung auf den Effekt des Programms

# Main-Methode

- Operation mit spezieller Bedeutung
- Einstiegspunkt in den Programmfluss
- Jedes Java-Programm besitzt genau einen Einstiegspunkt
  - Ein Projekt kann jedoch mehrere Main-Methoden besitzen
- `public static void main(String[] args) {}`



# Aufgabe 3

## Code

```
public class CommonMorning {
 public static void main(String[] args) {
 Alarm alarm = new Alarm();
 Dozent dozent = new Dozent();
 Meal breakfast = new Meal("Spiegelei mit Speck");
 Lecture pseVU = new Lecture("Java, Objekte");

 alarm.ring(28800);
 dozent.wakeUp();
 System.out.println("Awake: " + dozent.isAwake());
 dozent.dismiss(alarm);
 dozent.getUp();
 dozent.eat(breakfast);
 dozent.giveLecture(pseVU);
 }
}
```

## Betrachte den Code

- I. Welche Klassen gibt es?
- II. Welche Attribute haben die Klassen?
- III. Welche Operationen haben die Klassen?
- IV. Was für Objekte gibt es?

# Aufgabe 3 – Lösung (I)

## Code

```
public class CommonMorning {
 public static void main(String[] args) {
 Alarm alarm = new Alarm();
 Dozent dozent = new Dozent();
 Meal breakfast = new Meal("Spiegelei mit Speck");
 Lecture pseVU = new Lecture("Java, Objekte");

 alarm.ring(28800);
 dozent.wakeUp();
 System.out.println("Awake: " + dozent.isAwake());
 dozent.dismiss(alarm);
 dozent.getUp();
 dozent.eat(breakfast);
 dozent.giveLecture(pseVU);
 }
}
```

## Klassen

| Alarm     |
|-----------|
|           |
| ring(int) |

| Lecture          |
|------------------|
| content : String |
|                  |

| Meal          |
|---------------|
| name : String |
|               |

| Dozent                                                                                  |
|-----------------------------------------------------------------------------------------|
| isAwake : boolean                                                                       |
| wakeUp()<br>dismiss(Alarm)<br>getUp()<br>eat(Meal)<br>giveLecture(Lecture)<br>isAwake() |

# Aufgabe 3 – Lösung (II)

## Code

```
public class CommonMorning {
 public static void main(String[] args) {
 Alarm alarm = new Alarm();
 Dozent dozent = new Dozent();
 Meal breakfast = new Meal("Spiegelei mit Speck");
 Lecture pseVU = new Lecture("Java, Objekte");

 alarm.ring(28800);
 dozent.wakeUp();
 System.out.println("Awake: " + dozent.isAwake());
 dozent.dismiss(alarm);
 dozent.getUp();
 dozent.eat(breakfast);
 dozent.giveLecture(pseVU);
 }
}
```

## Objekte

|                         |
|-------------------------|
| <u>alarm</u> : Alarm    |
|                         |
| <u>pseVU</u> : Lecture  |
| content = "Java"        |
| <u>breakfast</u> : Meal |
| name = "Spiegelei"      |
| <u>dozent</u> : Dozent  |
| isAwake = false         |

# Git Tutorial

Live Demo

# Fragen?

