Technische Hochschule Rosenheim
Technical University of Applied Sciences

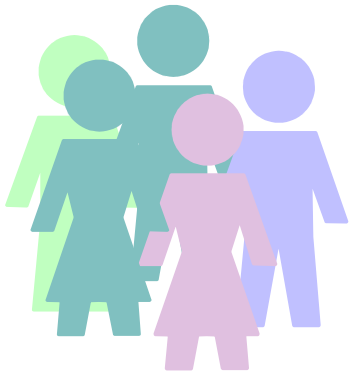
Objektorientierte Programmierung
Kapitel 10 – Vererbung

Prof. Dr. Kai Höfig

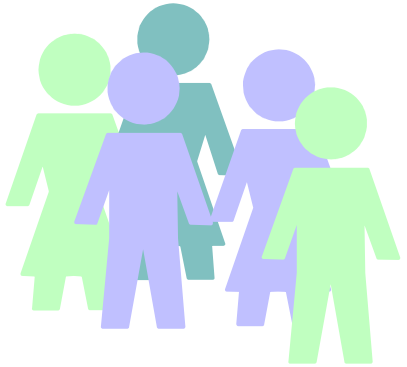
Menge ähnlicher, aber verschiedener Objekte

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Fußballfans



Schuhenthusiasten



Eigenschaften

▪ Name ▪ Alter	**Gemeinsamkeiten**	▪ Name ▪ Alter
▪ Lieblingsverein	**Unterschiede**	▪ Anzahl der Schuhpaare

Verhaltensweisen

▪ Schlafen ▪ Essen	**Gemeinsamkeiten**	▪ Schlafen ▪ Essen
▪ Fußball schauen	**Unterschiede**	▪ Schuhe kaufen

Motivation – Analyse auf Metaebene

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Analyse der beiden Gruppen

- Einige Unterschiede
  → Zusammenfassen in eine Klasse geht nicht!
- Viele Gemeinsamkeiten
  → Aufspalten in zwei getrennte Klassen bewirkt hohe Redundanz

- Wie werden derartige Sachverhalte programmiert?
  - Möglichst wenig Redundanz
  - Unterschiede deutlich machen

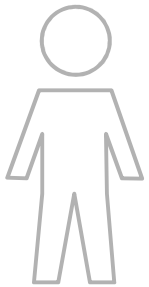
Lösungsidee

- Zentrale Definition der Gemeinsamkeiten
  - (generalisieren – allgemeine Klasse - Oberklasse)
- Spezialisierte Klasse (Unterklasse)
  - Dokumentation der Unterschiede
  - zusätzliche Attribute und/oder Methoden
  - Gemeinsamkeiten geerbt von zentraler Definition
  - Methoden können überschrieben bzw. redefiniert werden

Beispiel Lösungsidee

Technische Hochschule Rosenheim
Technical University of Applied Sciences

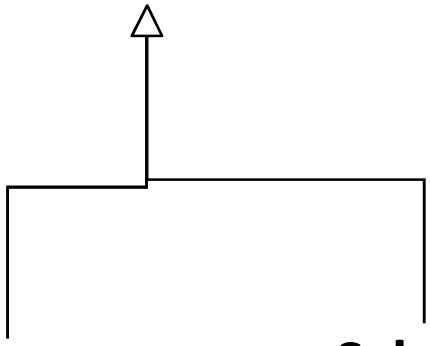
Extrahieren gemeinsamer Merkmale



Mensch

Eigenschaften
- Name
- Alter

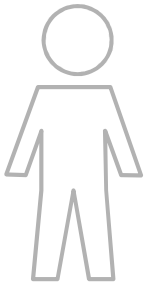
Verhaltensweisen
- Schlafen
- Essen



Fussballfan

Eigenschaften
- Lieblingsverein

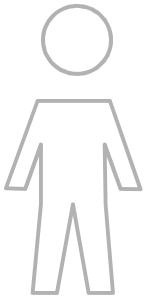
Verhaltensweisen
- Fußball schauen



Schuh-enthusiast

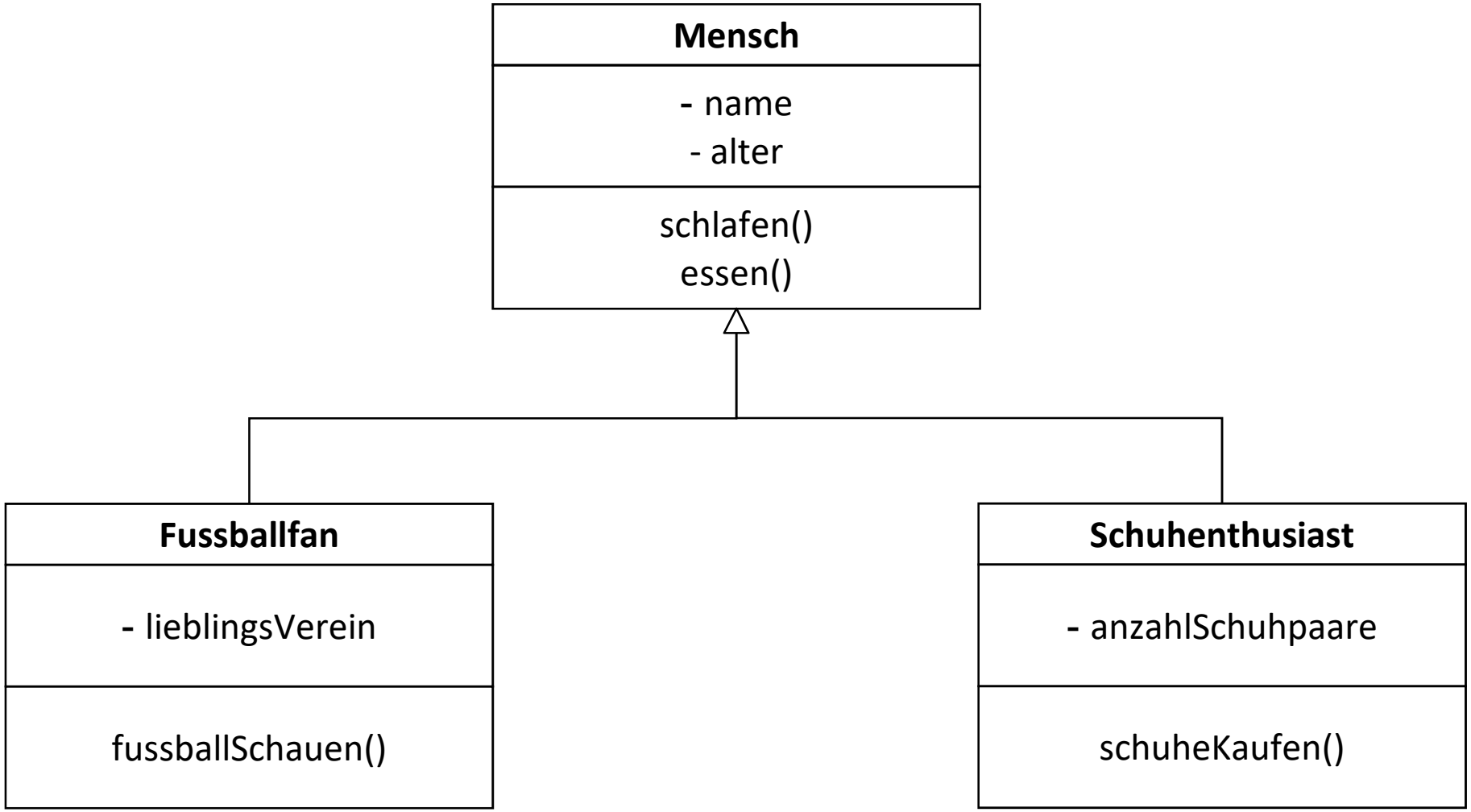
Eigenschaften
- Anzahl d. Schuhpaare

Verhaltensweisen
- Schuhe kaufen



Vererbung im UML-Klassendiagramm

Technische Hochschule Rosenheim
Technical University of Applied Sciences



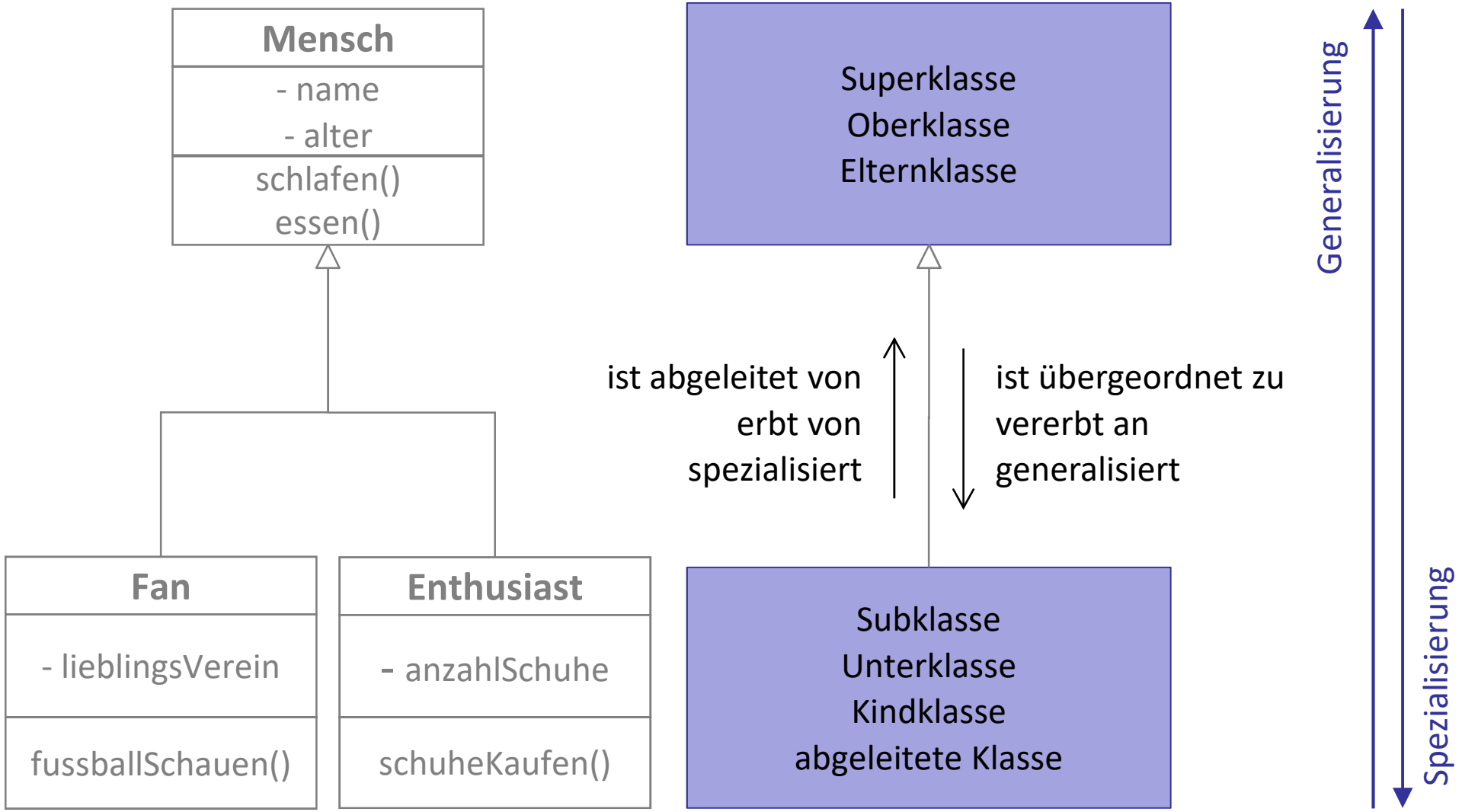
Bedeutung von Vererbung

Technische Hochschule Rosenheim
Technical University of Applied Sciences

- **Grundidee**
  - Beschreibt Ähnlichkeit zwischen Klassen
  - Spezialfall einer Beziehung zwischen Klassen
    - Jedes Objekt der Unterklasse „ist ein“ (is a) Objekt der Oberklasse
  - Strukturiert Klassen in Hierarchie von Abstraktionsebenen
  - Ermöglicht Definition einer neuen Klasse auf Basis bereits bestehender Klassen (Wiederverwendung!)

Wesentlicher Mechanismus, der objektorientierte Sprachen von funktionalen/prozeduralen Sprachen unterscheidet!

Begriffe


Mensch
- name
- alter
schlafen()
essen()
Fan
- lieblingsVerein
fussballSchauen()
Enthusiast
- anzahlSchuhe
schuheKaufen()
Superklasse
Oberklasse
Elternklasse
Subklasse
Unterklasse
Kindklasse
abgeleitete Klasse
ist abgeleitet von
erbt von
spezialisiert
ist übergeordnet zu
vererbt an
generalisiert
Generalisierung
Spezialisierung

Vorgehensweise

Technische Hochschule Rosenheim
Technical University of Applied Sciences

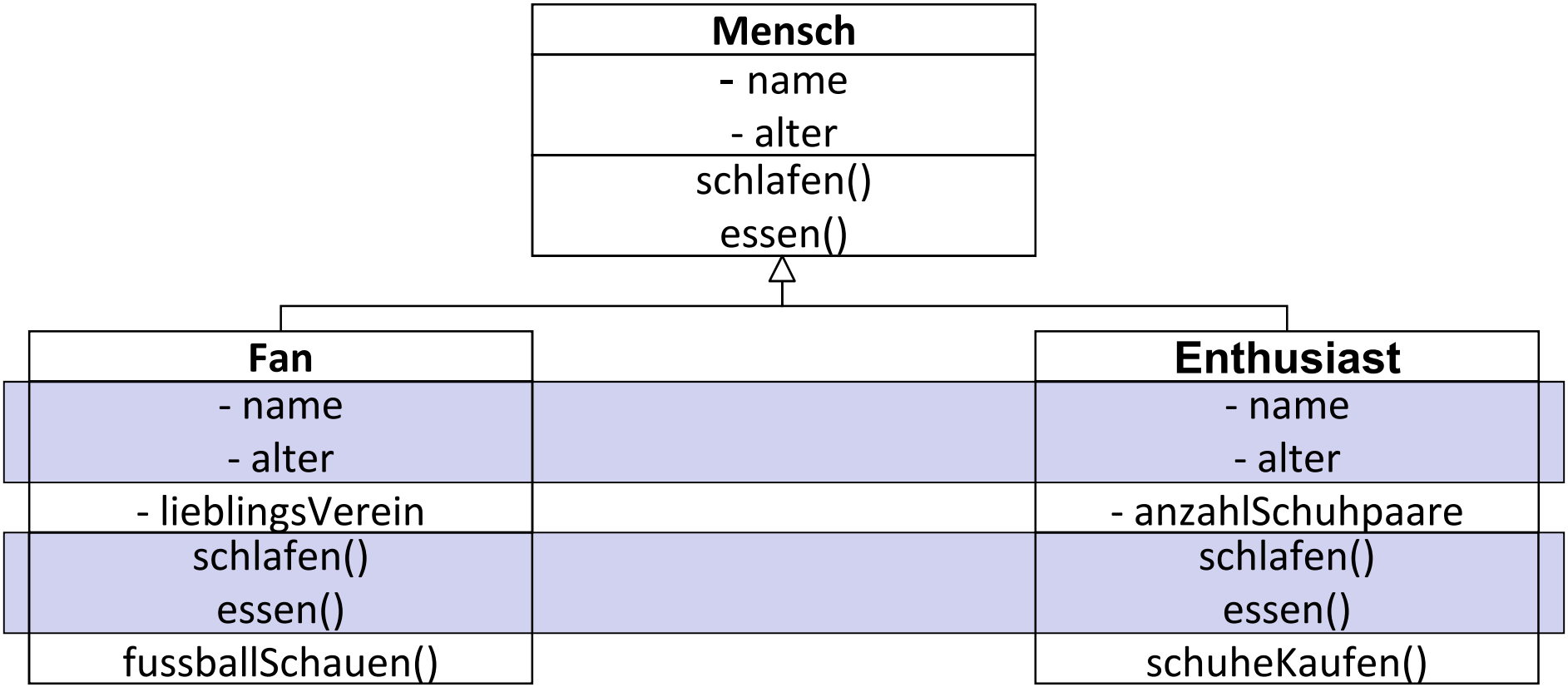
- **Zwei mögliche Vorgehensweisen:**
  - Bottom-up: Vom Speziellen zum Allgemeinen
  - Top-down: Vom Allgemeinen zum Speziellen

- **Wann nimmt man was?**
  - Botton-up:
    Wenn Gemeinsamkeiten erst in teilfertiger Lösung auffallen
  - Top-down:
    Wenn man schon vorab weiß, dass es Gemeinsamkeiten gibt

Vorgehensweise – Bottom-up

Technische Hochschule Rosenheim
Technical University of Applied Sciences

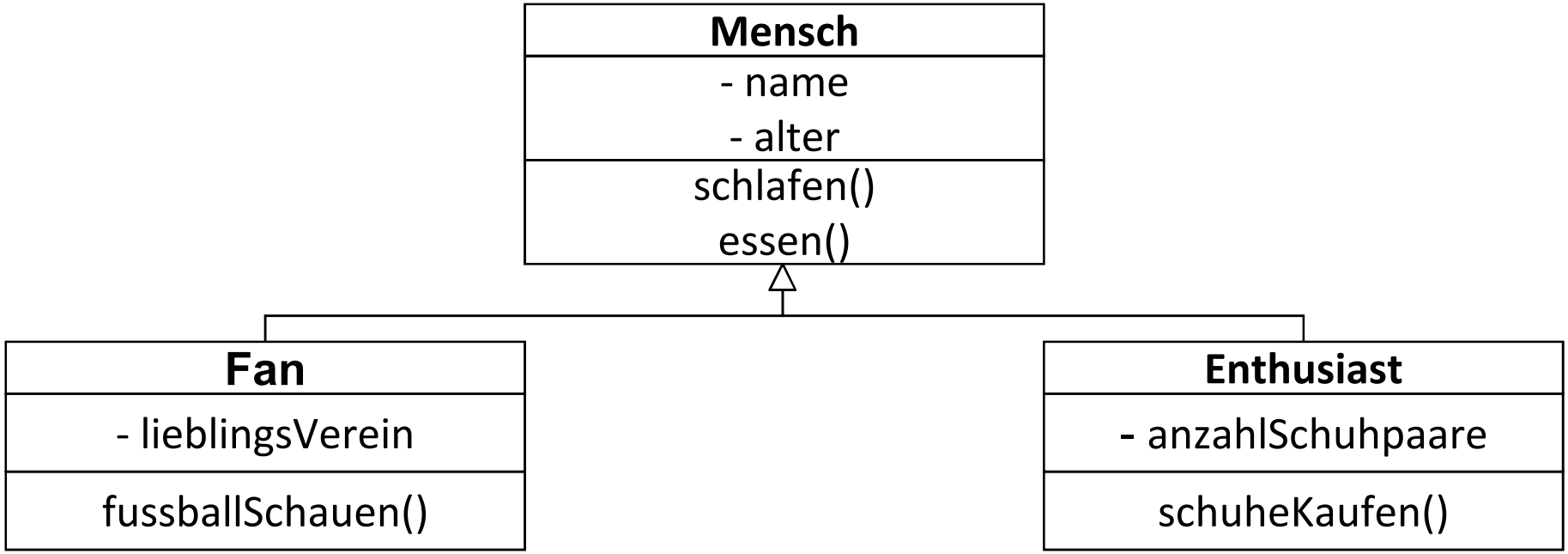
1. Zunächst einzelne Klassen modellieren
2. Redundanzen feststellen
3. Gemeinsamkeiten auslagern in Oberklasse
4. Ursprüngliche Klassen von Oberklasse ableiten und "ausmisten"



Vorgehensweise – Top-down

Technische Hochschule Rosenheim
Technical University of Applied Sciences

1. Erst die Gemeinsamkeiten in zentraler Oberklasse definieren
2. Spezialisierende Klassen definieren, von Oberklasse ableiten
3. Dann die Spezifika der abgeleiteten Klassen definieren
4. Gegebenenfalls Zahl der abgeleiteten Klassen sukzessive erweitern



Schlüsselwort "extends"

Technische Hochschule Rosenheim
Technical University of Applied Sciences

- Verweis auf **Oberklasse** durch Schlüsselwort ***extends*** im Kopf der **abgeleiteten Klasse** (Unterklasse)
  - Beispiel:

```
Class Cat extends Pet {…}
```

- Abgeleitete Klasse erbt *alle* Variablen und *alle* Methoden der Oberklasse.

- Ändern der Funktionalität der Oberklasse möglich durch
  - Hinzufügen neuer Elemente (Attribute, Methoden, …)
  - Überladen der vorhandenen Methoden
    - Bsp: `public String getName(String greeting)`
  - Redefinieren (Überschreiben) der vorhandenen Methoden

Sichtbarkeiten im Überblick

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Modifier	Klasse	Paket	Unterklasse	Welt
`public`	Ja	Ja	Ja	Ja
`protected`	Ja	Ja	Ja	Nein
kein Attribut	Ja	Ja	Nein	Nein
`private`	Ja	Nein	Nein	Nein

http://docs.oracle.com/javase/tutorial/java/javaOO/accesscontrol.html

- Attribute in der Regel `private`
  - …außer guter Grund für `protected` oder `public`
- Methoden in der Regel `public`
  - …außer guter Grund für `protected` oder `private`

Implementierung – Definition der Oberklasse

Technische Hochschule Rosenheim
Technical University of Applied Sciences

```
public class Person {

  // Gemeinsame Eigenschaften aller Unterklassen
  private String name;
  private int age;

  // Gemeinsame Funktionalität aller Unterklassen
  public String sleep() {
   return "sleep: Chrrrrr.... chrrrr...";
  }
  public String eat() {
        return "eat  : Mmmh, lecker.";
  }
}
```

Implementierung – Unterklasse definieren (1)

Technische Hochschule Rosenheim
Technical University of Applied Sciences

```
public class Fussballfan extends Person {

  // Neues Attribut
  private String favoriteClub;

  // Neue Funktionalität
  public String watchSoccerGame() {
    return "play : ja... Ja... TOOOOOOOR!!!";
  }
}
```

Implementierung – Unterklasse definieren (2)

Technische Hochschule Rosenheim
Technical University of Applied Sciences

```
public class Schuhenthusiast extends Person {

  // Neues Attribut
  private int pairsOfShoes;

  // Neue Funktionalität
  public String buyShoes() {
    pairsOfShoes++;
    return "shop : DIE sind ja schick..., " +
           "Paar Nummer" + pairsOfShoes;
  }
}
```

Implementierung – Hauptklasse definieren

Technische Hochschule Rosenheim
Technical University of Applied Sciences

```
public class Main {
  public static void processPerson(Person person) {
    person.eat();
  }
  public static void main(String[] args) {
    Fussballfan eva = new Fussballfan();
    Schuhenthusiast adam = new Schuhenthusiast();
    System.out.println("Das macht Eva:");
    eva.sleep();
    processPerson(eva);
    eva.watchSoccerGame();
    System.out.println();
    System.out.println("Das macht Adam:");
    adam.sleep();
    processPerson(adam);
    adam.buyShoes();
    System.out.println();
  }
}
```

Implementierung – Ausgabe

Technische Hochschule Rosenheim
Technical University of Applied Sciences

- Ausgabe des Hauptprogramms

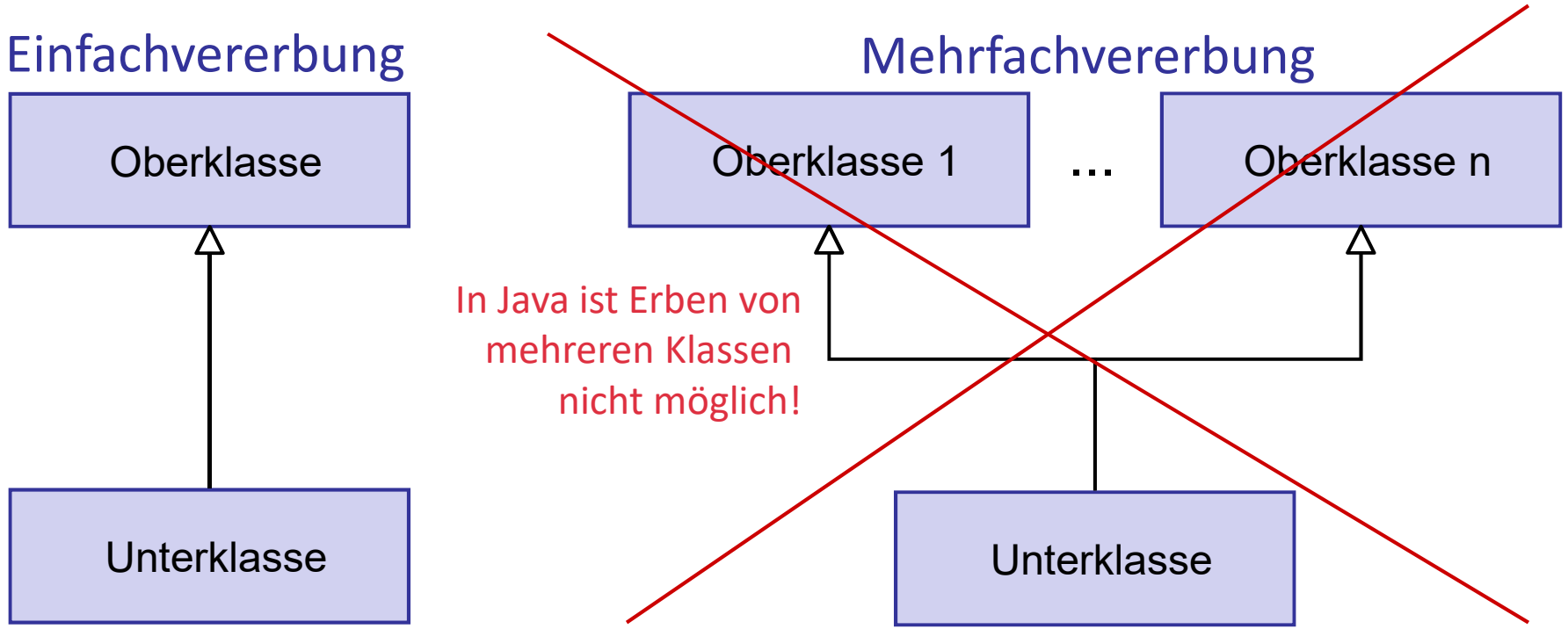
```
Das macht Eva:
sleep: Chrrrrr.... chrrrr...
eat  : Mmmmh, lecker.
play : Ja... JAA... TOOOOOOOR!!!
Das macht Adam:
sleep: Chrrrrr.... chrrrr...
eat  : Mmmmh, lecker.
shop : DIE sind ja schick...
```

Arten von Vererbung

Technische Hochschule Rosenheim
Technical University of Applied Sciences

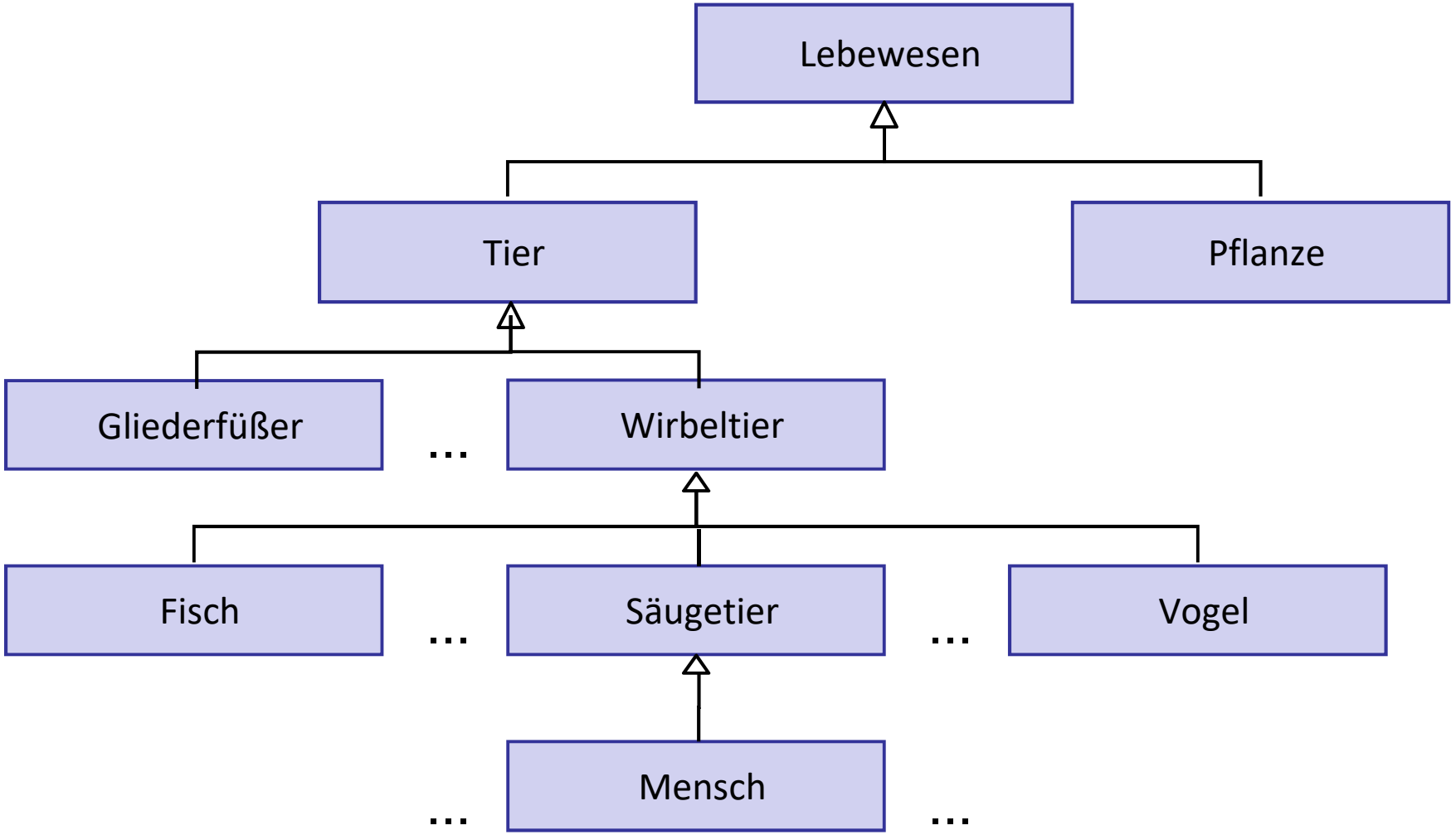
Einfachvererbung
- Unterklasse erbt von genau einer Oberklasse

Mehrfachvererbung
- Unterklasse erbt von mehr als einer Oberklasse



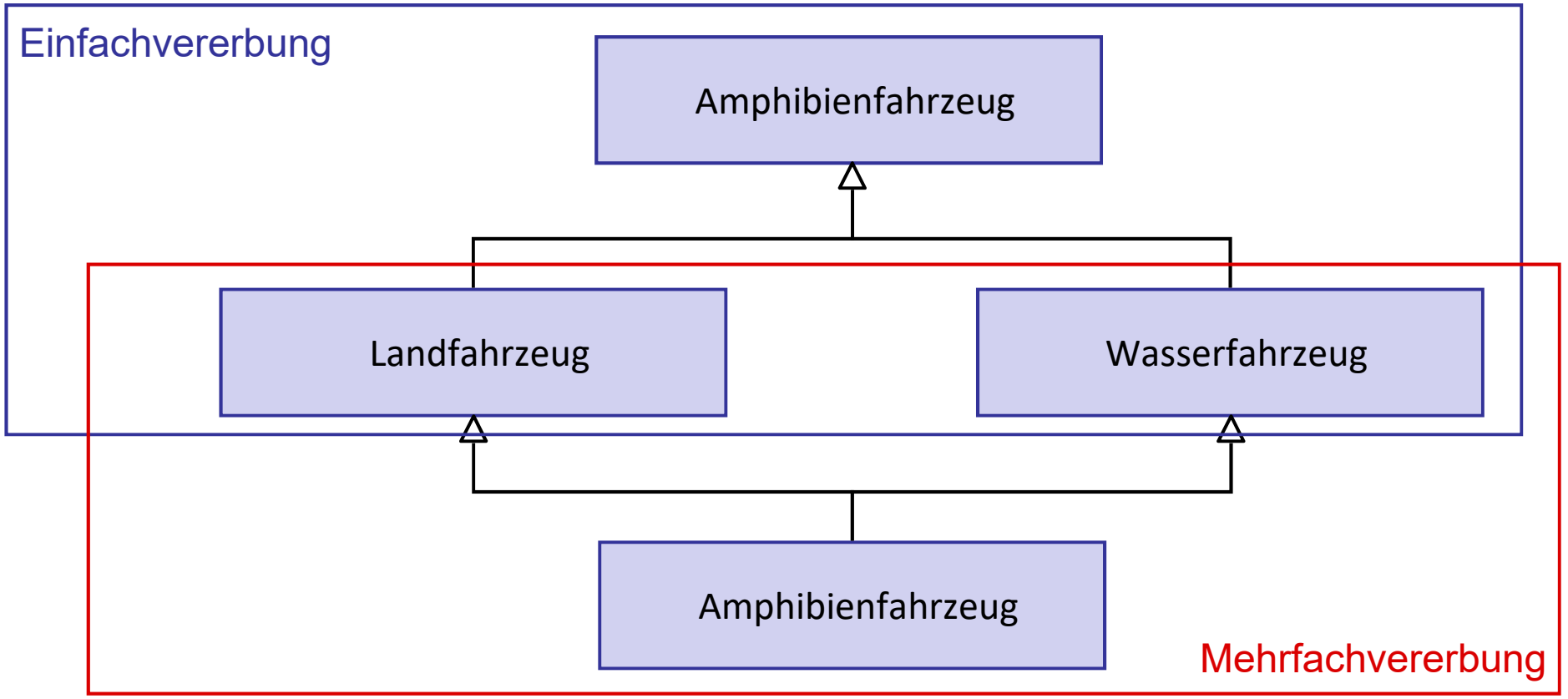
Einfachvererbung über mehrere Stufen

Technische Hochschule Rosenheim
Technical University of Applied Sciences



Einfach- und Mehrfachvererbung

Technische Hochschule Rosenheim
Technical University of Applied Sciences



Was wird vererbt?

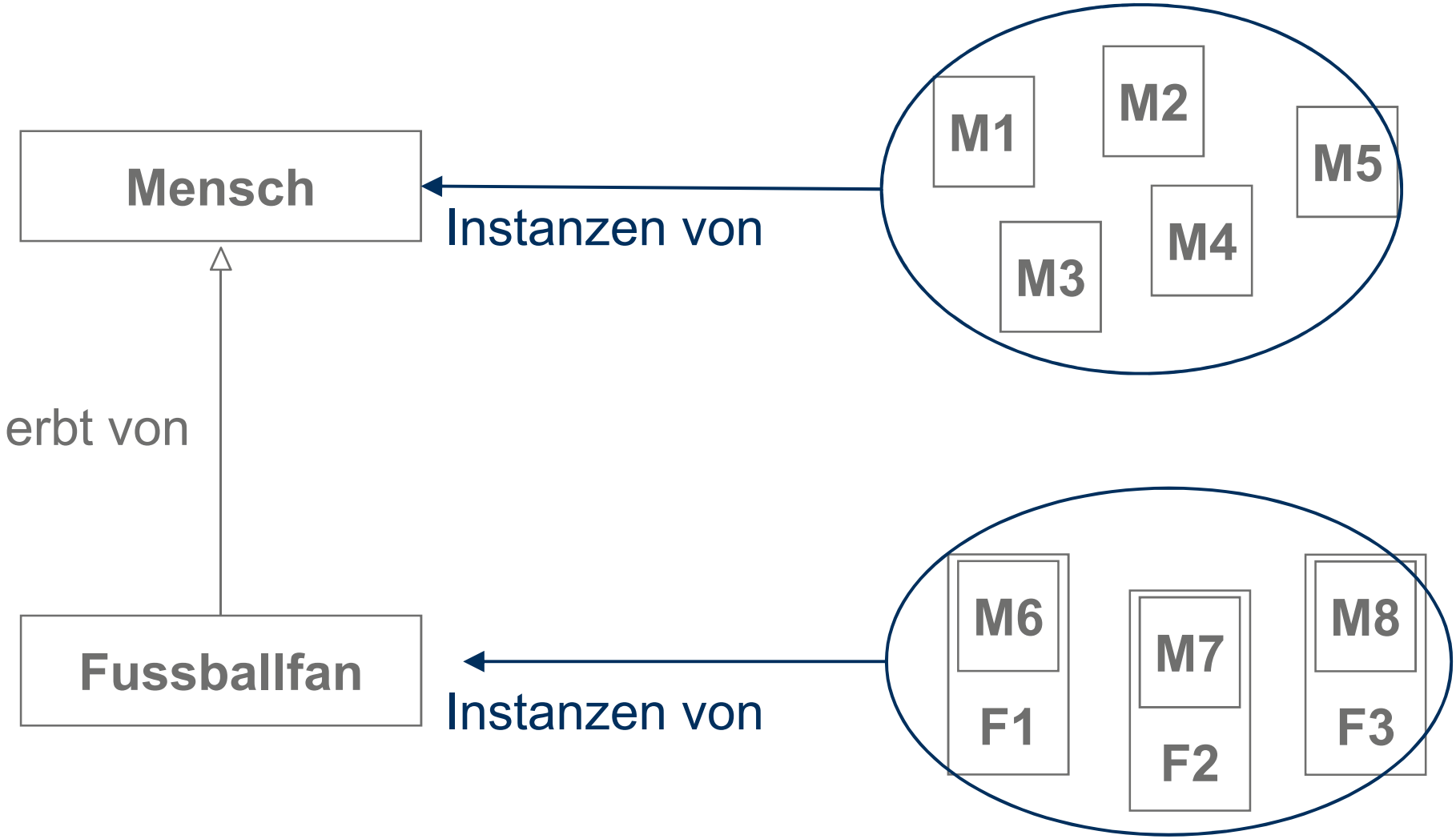
Technische Hochschule Rosenheim
Technical University of Applied Sciences

- Unterklasse erbt von Oberklasse …
  - die Operationen (das Verhalten)
  - die Attribute (die möglichen Zustände)
  - die Semantik!
    (d.h. anstelle eines Objekts der Oberklasse kann immer auch ein Objekt einer beliebigen Unterklasse verwendet werden!
    => Substitutionsprinzip)

- Beispiele in Java:
  - ```
    Person p = new Man();
    p = new Woman();
    ```
    ```

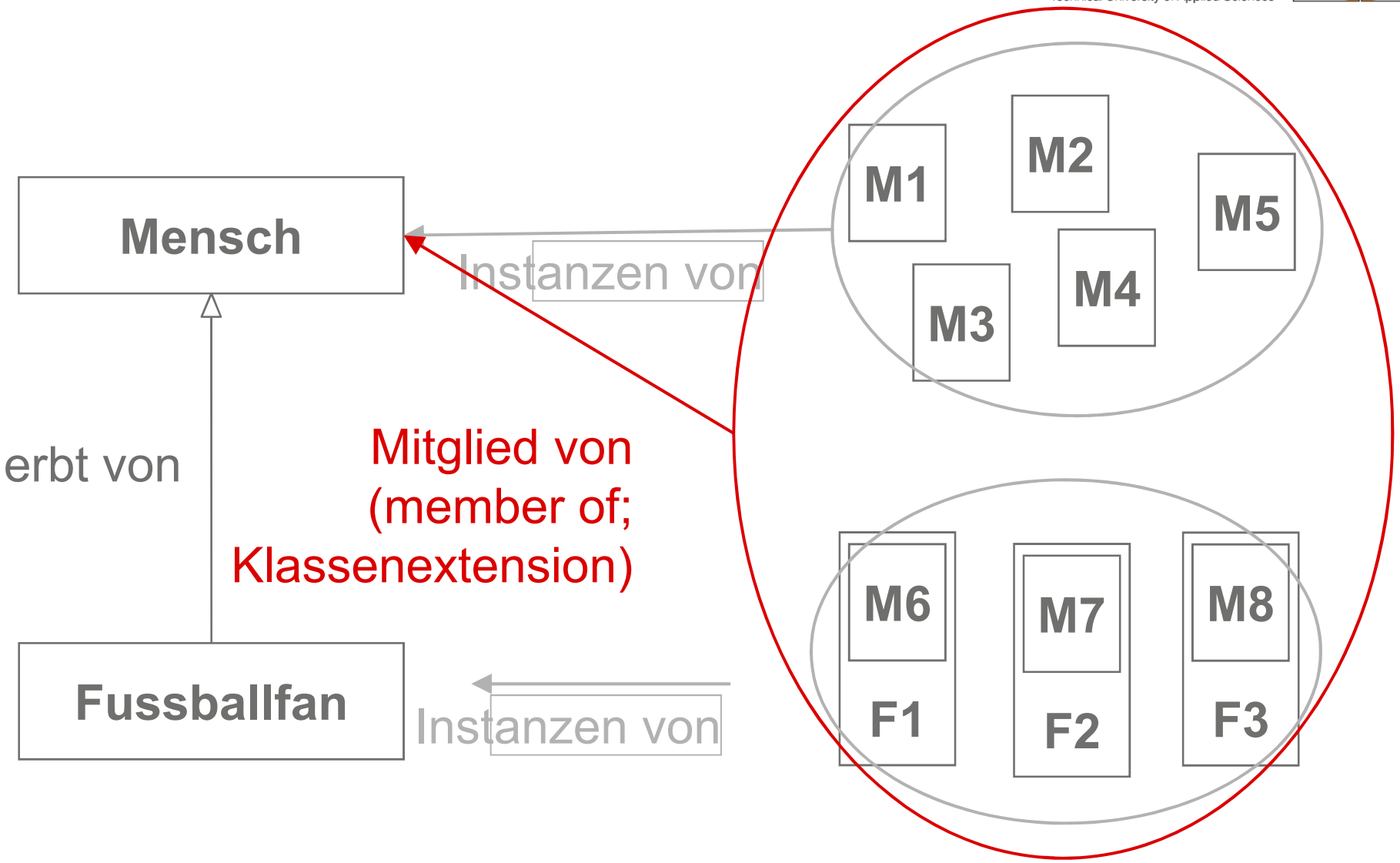
Syntaktische Vererbung

Technische Hochschule Rosenheim
Technical University of Applied Sciences



Semantische Vererbung

Technische Hochschule Rosenheim
Technical University of Applied Sciences



Konstruktoren in der Vererbung

Technische Hochschule Rosenheim
Technical University of Applied Sciences

- Jeder Konstruktor einer abgeleiteten Klasse sollte einen Konstruktor der Oberklasse aufrufen.
  - Ansonsten würden Attribute der Oberklasse gegebenenfalls niemals initialisiert.

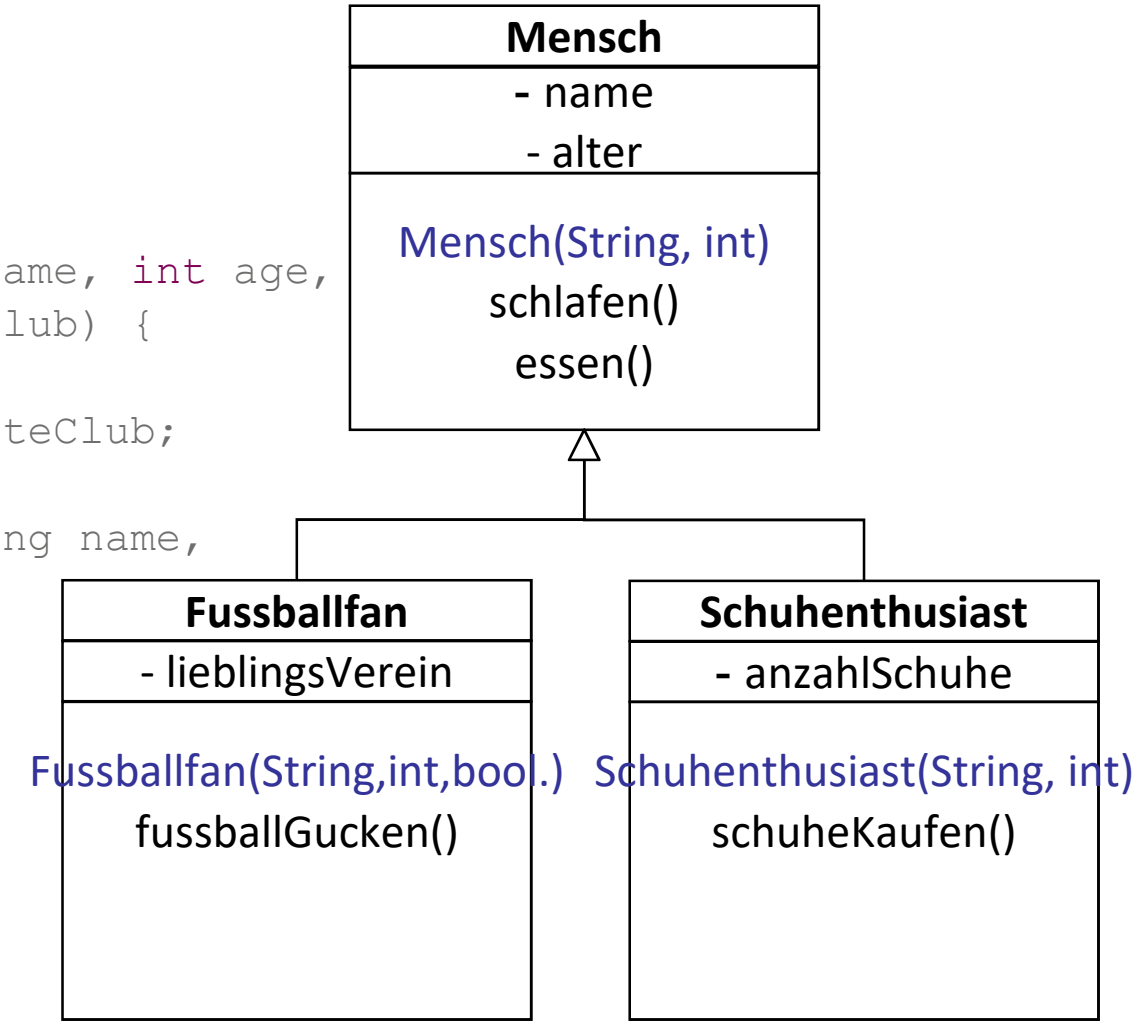
- Expliziter Aufruf des Default-Konstruktors der Oberklasse:
  -
```
super();
```
- Expliziter Aufruf eines Werte-Konstruktors der Oberklasse:
  -
```
super(name,…);
```
- Bei fehlendem explizitem Aufruf:
  - Impliziter Aufruf des Default-Konstruktors der Oberklasse. Dieser muss explizit angegeben werden, sonst tritt ein Fehler auf.

- Regel: Ein Konstruktoraufruf **muss** immer *erstes* Statement im Konstruktor der Unterklasse sein

Konstruktoren mit `super()`

Technische Hochschule Rosenheim
Technical University of Applied Sciences

```
public Person (String name,
               int age) {
  this.name = name;
  this.age = age;
}
public Fussballfan (String name, int age,
            String favoriteClub) {
  super (name, age);
  this.favoriteClub = favoriteClub;
}
public Schuhenthusiast (String name,
              int age) {
       (name, age);
  pairsOfShoes = 0;
}
```



Konstruktoren mit `this()`

Technische Hochschule Rosenheim
Technical University of Applied Sciences

- Zur Erinnerung
  - Aufruf eines anderen Konstruktor der gleichen Klasse: `this()`
  - Muss als erste Anweisung im Konstruktorrumpf stehen
  - Nützlich, um Redundanzen in den Konstruktoren zu vermeiden

- Beispiel:

```
public Schuhenthusiast (String name, int pairsOfShoes)
{
  this.name = name;
  this.pairsOfShoes = pairsOfShoes;
}

public Schuhenthusiast (String name) {
  this (name, 0);
}
```

Zusammenfassung

Technische Hochschule Rosenheim
Technical University of Applied Sciences

- Vererbung
- Generalisierung und Spezialisierung
- Botom-up und Top-down Ansatz beim Entwurf
- Sichtbarkeiten
- Mehrfachvererbung
- Syntaktische und semantische Vererbung
- Konstruktoren mit `super` und `this`