Technische Hochschule Rosenheim
Technical University of Applied Sciences

Objektorientierte Programmierung
Kapitel 5 – Objektorientierte Modellierung

Prof. Dr. Kai Höfig

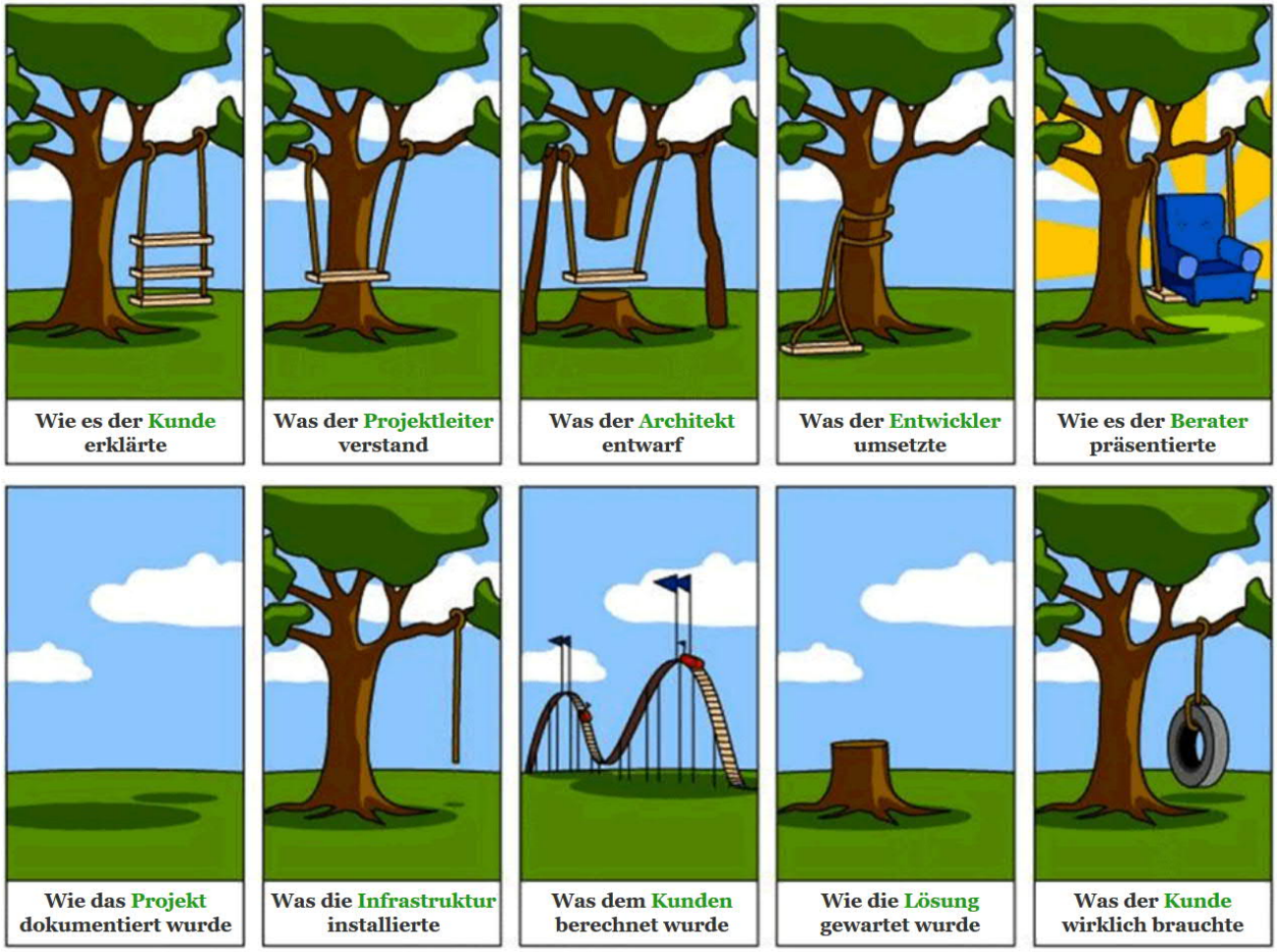
Inhalt

Technische Hochschule Rosenheim
Technical University of Applied Sciences

- Einführung Modellierung
- UML Klassen und Objektdiagramme
- Codegenerierung

Wie es der Kunde erklärt – Was der Projektleiter verstand…

Technische Hochschule Rosenheim
Technical University of Applied Sciences



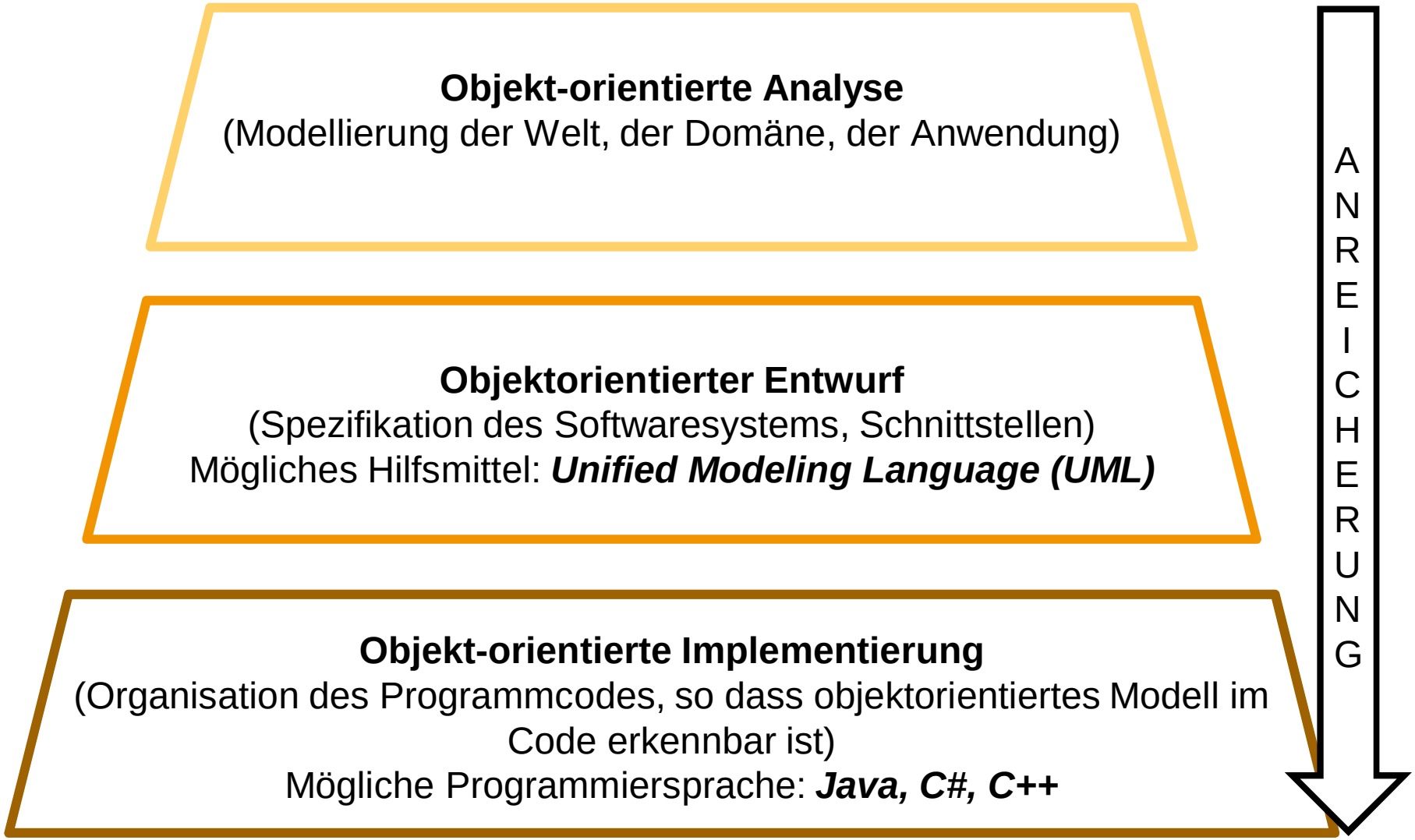
Ziel der objektorientierten Programmierung (OOP)

Technische Hochschule Rosenheim
Technical University of Applied Sciences

- Software reproduziert immer einen Ausschnitt aus der Realität.
  - Unwichtige Details können ignoriert werden.
- Objektorientierte Programmierung
  - Programmieransatz, der modellierten Teil der Realität direkt wiederspiegelt.

Objekt-orientierte Softwareentwicklung

Technische Hochschule Rosenheim
Technical University of Applied Sciences

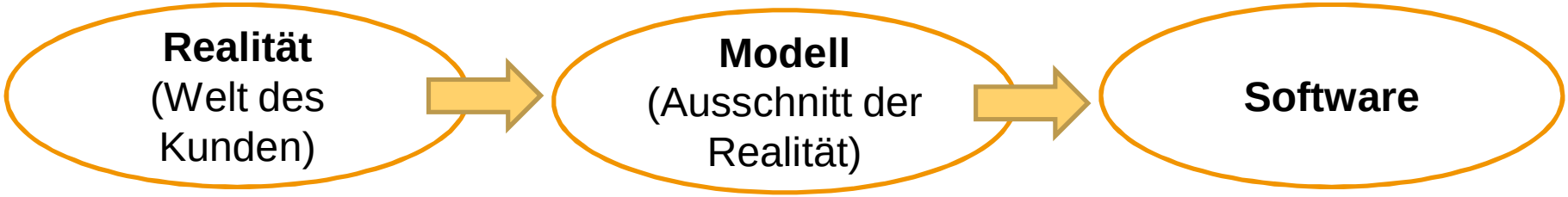


Motivation: Objektorientierter Ansatz

Technische Hochschule Rosenheim
Technical University of Applied Sciences

- Wie kommt man vom Kundenproblem zum Software-Produkt?
  - ***Modellierung*** der Realität
  - Weglassen unwichtiger Details, Konzentration auf das Wesentliche!

- Wie können wir das Kundenproblem möglichst einfach beschreiben?
  - Im Modell **UND** im Programmcode!

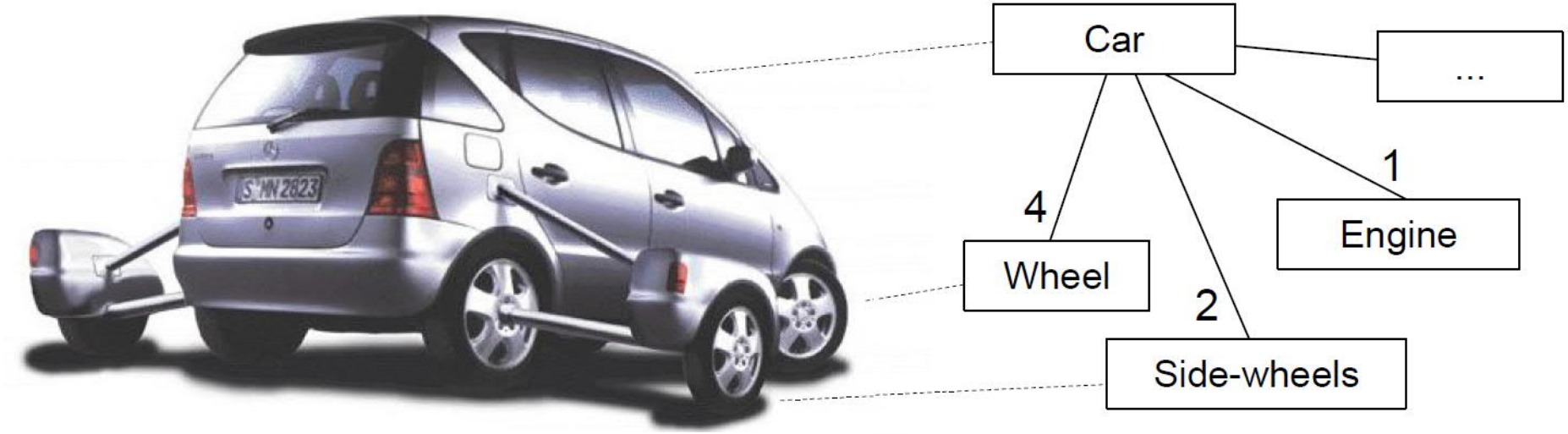


Anforderung an objektorientierte Programmierung:
In Code soll die Modellstruktur und damit auch die Welt des Kunden klar erkennbar sein!

Von der Realität zum Modell

- Objektorientierte Entwicklung und Modellierung:
  - Strukturiert die Realität in ***Objekte*** ("Dinge, die in realer Welt vorkommen").
  - Erkennt die wesentlichen ***Beziehungen zwischen Objekten***.

- Beispiel: Aus was besteht ein Auto?
  - Ein *Auto* besteht aus *Rädern*.
  - Ein *Fahrer* fährt ein Auto



Die Unified Modeling Language (UML)

Technische Hochschule Rosenheim
Technical University of Applied Sciences

- Die UML ist eine einheitliche Modellierungssprache zur *Spezifikation, Konstruktion* und *Dokumentation* von Software und Systemen
  - Objektorientiert, grafisch
  - Industriestandard der Object Management Group (OMG)
    Version 2.5.1 Dezember 2017

- Verschiedene Systemaspekte werden mit verschiedenen Diagrammarten modelliert:
  - für die Datenstrukturen (Statische Strukturen):
    - **Klassen- und Objektdiagramme** (Class and Object Diagram)
  - für das Systemverhalten (Dynamik):
    - Anwendungsfalldiagramm (Use Case Diagram)
    - Interaktionsdiagramm (Interaction Diagram, früher: Sequence Diagram)
    - Kollaborationsdiagramm (Collaboration Diagram)
    - Zustandsdiagramm (State Diagram)
    - Aktivitätsdiagramm (Activity Diagram)
  - für den Systementwurf (Statische Strukturen):
    - Komponentendiagramm (Component Diagram)
    - Einsatzdiagramm (Deployment Diagram)

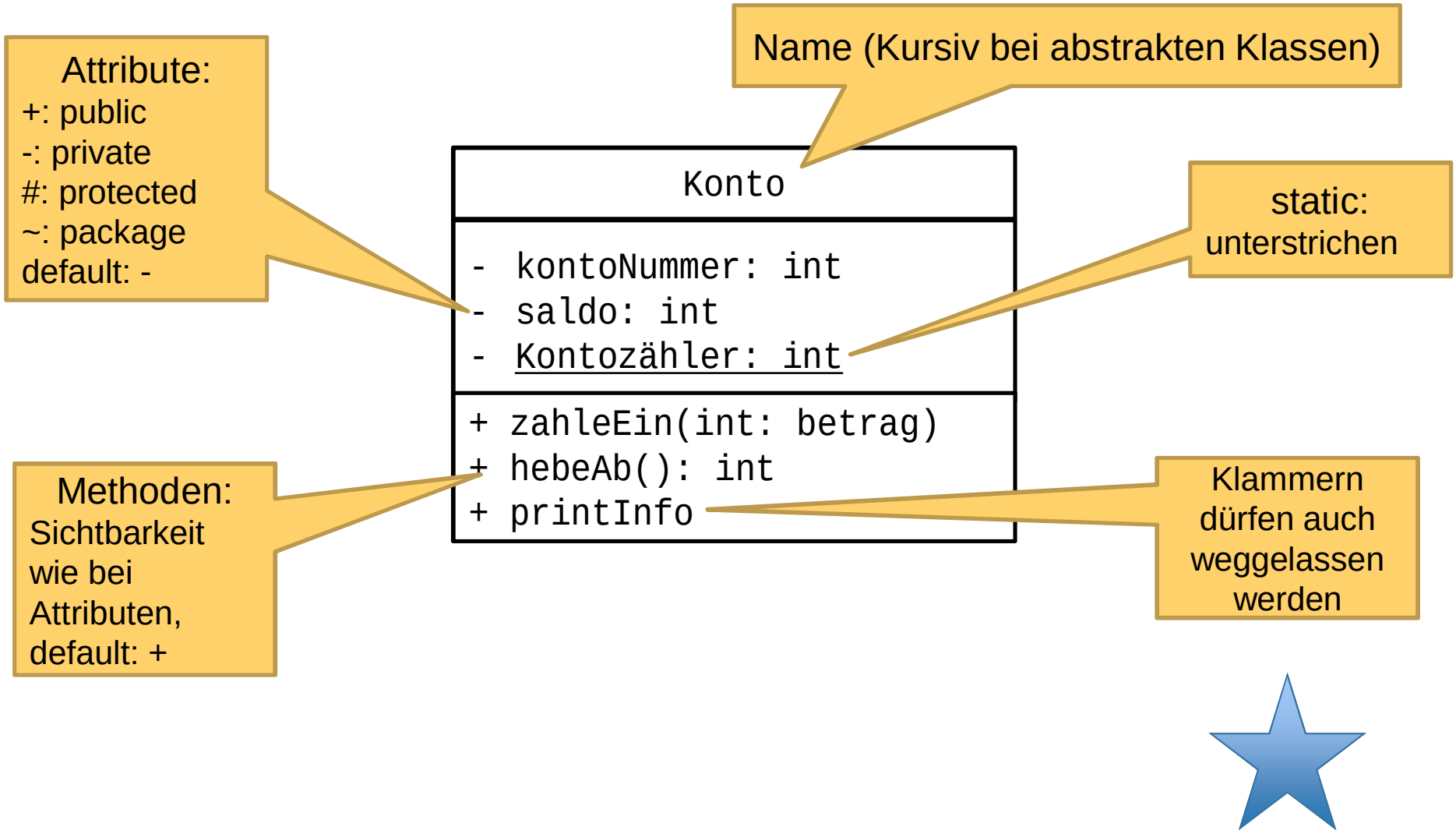
UML Klassendiagramme

Technische Hochschule Rosenheim
Technical University of Applied Sciences

- Ein Klassendiagramm
  - zeigt die statische Struktur eines Systems
  - bildet den Kern des Analysemodells
  - enthält Elemente der folgenden Arten:
    - Paket
    - Objektklasse
    - Attribut und Operation
    - Assoziation (mit Bezeichnung, Kardinalitäten und Rollen)
    - Generalisierungs-/Spezialisierungsbeziehung
  - Ist häufig das Diagramm aus dem (objektorientierter) (Java) Code generiert wird und damit das zentrale Diagramm des objektorientierten Entwurfs.

Notation für Klassen im Klassendiagramm

Technische Hochschule Rosenheim
Technical University of Applied Sciences

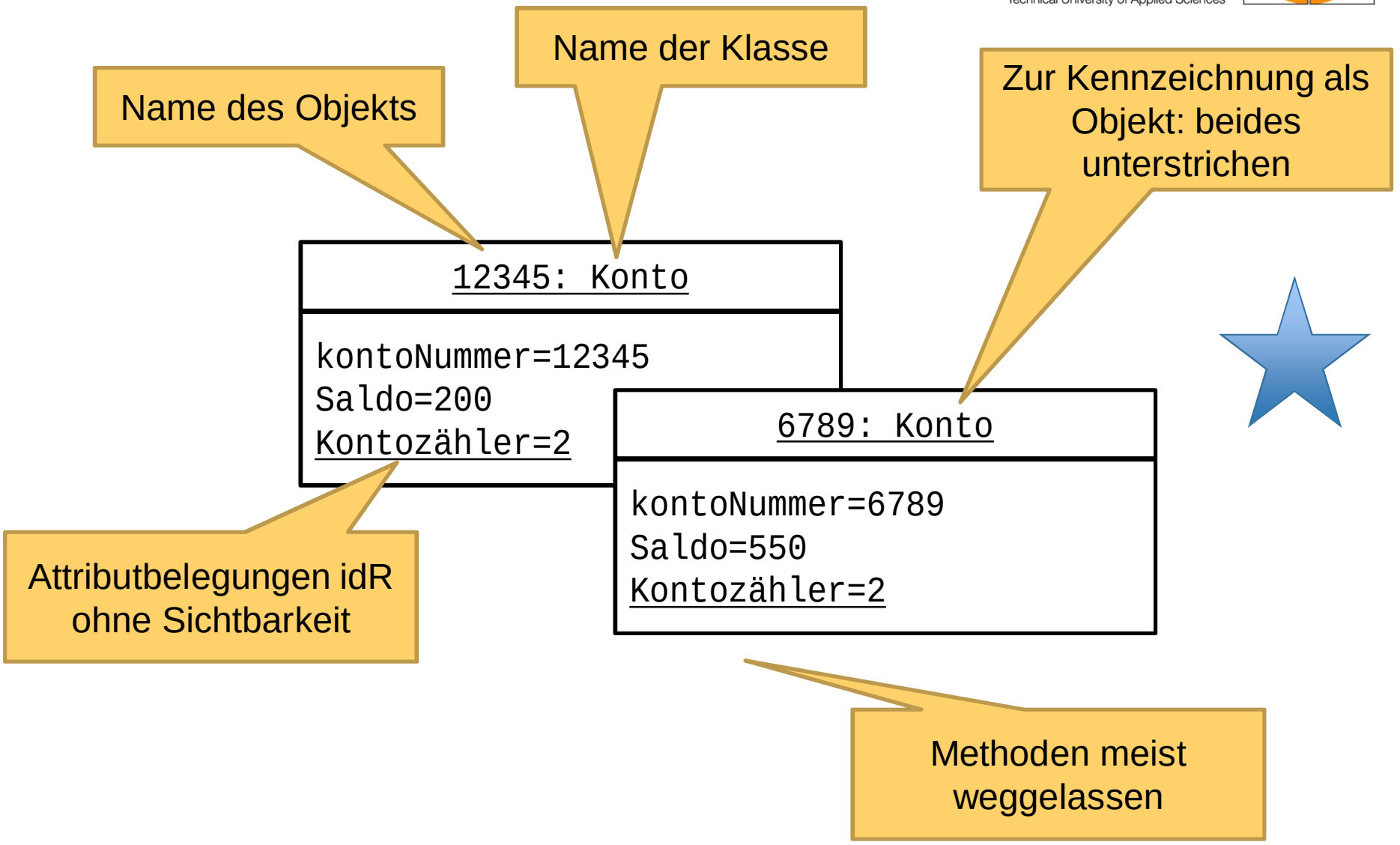


UML Objektdiagramme

Technische Hochschule Rosenheim
Technical University of Applied Sciences

- Ein Objektdiagramm
  - zeigt einen Ausschnitt eines Systems zu einem bestimmten Zeitpunkt.
  - Wird benutzt, um konkrete Beispiele oder Anwendungsszenarien grafisch darzustellen.
  - enthält Elemente der folgenden Arten:
    - Objekte mit Objektklasse
    - Attribute und Belegungen
    - Assoziation

Notation für Objekte im Objektdiagramm



Übung

Technische Hochschule Rosenheim
Technical University of Applied Sciences

- Erstellen Sie das Klassendiagramm und ein Objektdiagramm für zwei Objekte für folgenden Code:

```
public class Person {
        private String vorname;
        private String nachname;
        private static String trennzeichen = ", ";

        public Person(String vorname, String nachname){
                this.vorname = vorname;
                this.nachname = nachname;
        }
        public String getVorname(){ return this.vorname; }
        public void setVorname(String vorname){ this.vorname = vorname; }
        public String getNachname(){ return this.nachname; }
        public void setNachname(String nachname){ this.nachname = nachname; }
        public static String showTrennzeichen(){ return trennzeichen; }
        @Override
        public String toString(){
                return this.nachname  + trennzeichen + this.vorname;
        }
}
```

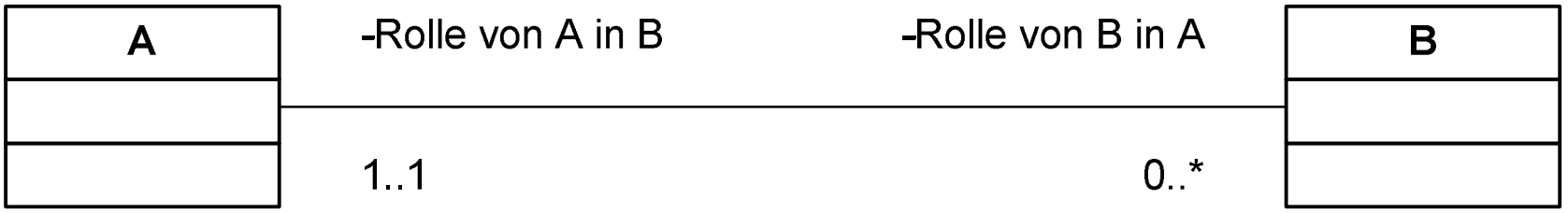
Codegenerierung aus UML Klassendiagrammen

Technische Hochschule Rosenheim
Technical University of Applied Sciences

- Aus UML Klassendiagrammen lassen sich Codegerüste generieren, z.B.
  - Attribute mit Sichtbarkeiten
  - Schlüsselwörter wie `abstract` oder `static`
  - Vererbungshierarchien mit `extends`
  - Interfaces mit `implements`
  - Und vieles mehr

- Aber: es existiert zwar eine standardisierte Modellierung, aber ***keine*** standardisierte Semantik. Die Codebeispiele in diesem Kapitel geben daher nur mögliche Interpretationen für die Generierung von Code aus UML Diagrammen wieder.

Beziehungen im Klassendiagramm – Bidirektional

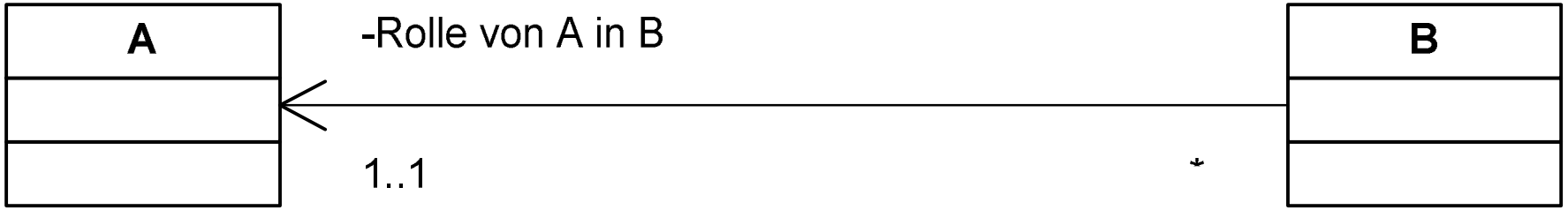



```
public class A {
        private B[] Rolle_von_B_in_A;
}
```

```
public class B {
        private A Rolle_von_A_in_B;
}
```

- Eine Bidirektionale Beziehung wird verwendet, falls Objekte einer anderen Klasse als Attribute referenziert werden.
- Rollen werden zu Attributbezeichnern. Es werden Multiplizitäten mit angegeben.

Beziehungen im Klassendiagramm – Unidirektional

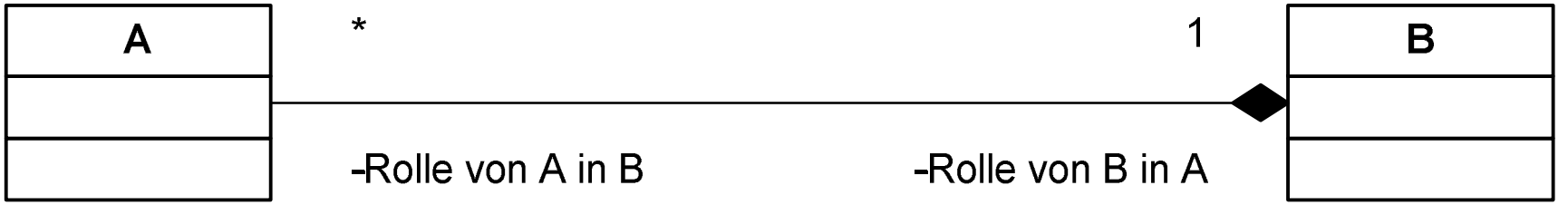



```
public class A {
        //keine Referenz zu B
}
```

```
public class B {
        private A Rolle_von_A_in_B;
}
```

- Eine Unidirektionale Beziehung wird verwendet, falls Objekte einer anderen Klasse als Attribute referenziert werden, dies aber nur in eine Richtung vorgesehen ist.
- Rollen werden zu Attributbezeichnern. Es werden Multiplizitäten mit angegeben.

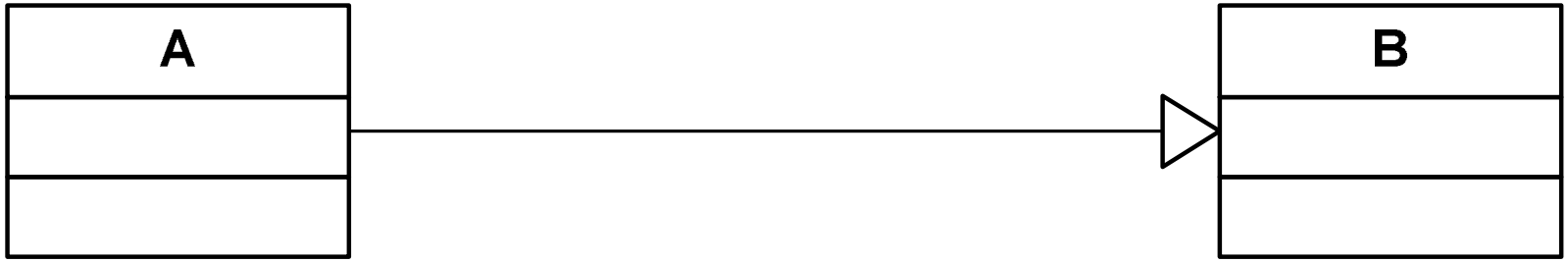
Beziehungen im Klassendiagramm – Komposition




```
class B {
   class A {
      private B[] Rolle_von_B_in_A
   }
   private A Rolle_von_A_in_B;
}
```

- Eine Komposition wird verwendet, falls Objekte aus anderen Objekten bestehen (B besteht aus A).
- Die Klassen sind dann idR ineinander geschachtelt. Eine Sichtbarkeit von A außerhalb von B kann durch `public` erreicht werden.

Beziehungen im Klassendiagramm – Vererbung

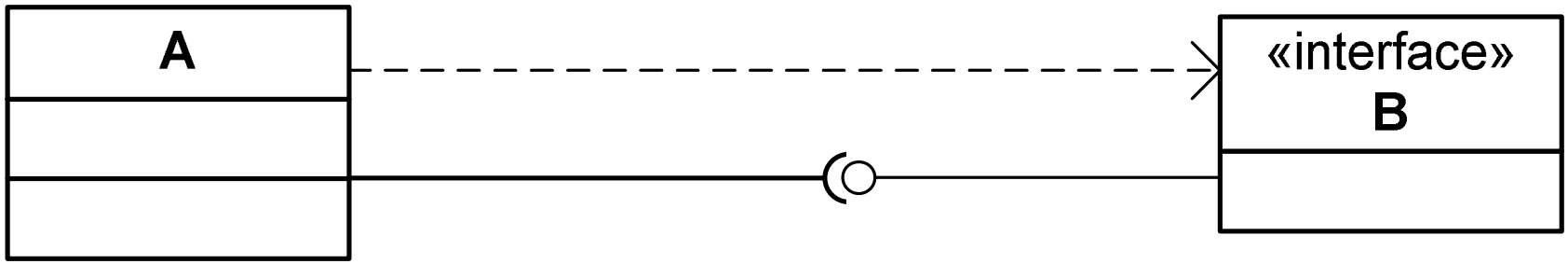



```
class A extends B {
}
```

- Eine Vererbung wird verwendet, falls Klassen die Eigenschaften voneinander erben.

Beziehungen im Klassendiagramm – Interfaces

Technische Hochschule Rosenheim
Technical University of Applied Sciences




```
class A implements B {
}
```

- Zwei Varianten. Mit gestrichelter Linie (use) oder durch Lolipop-Notation (provided interface, requested interface)
- Vorteil der Lolipop Notation: Interfaces können angefragt werden auch wenn sie noch nicht modelliert sind.

Fallbeispiel: Eine einfache Bank-Software

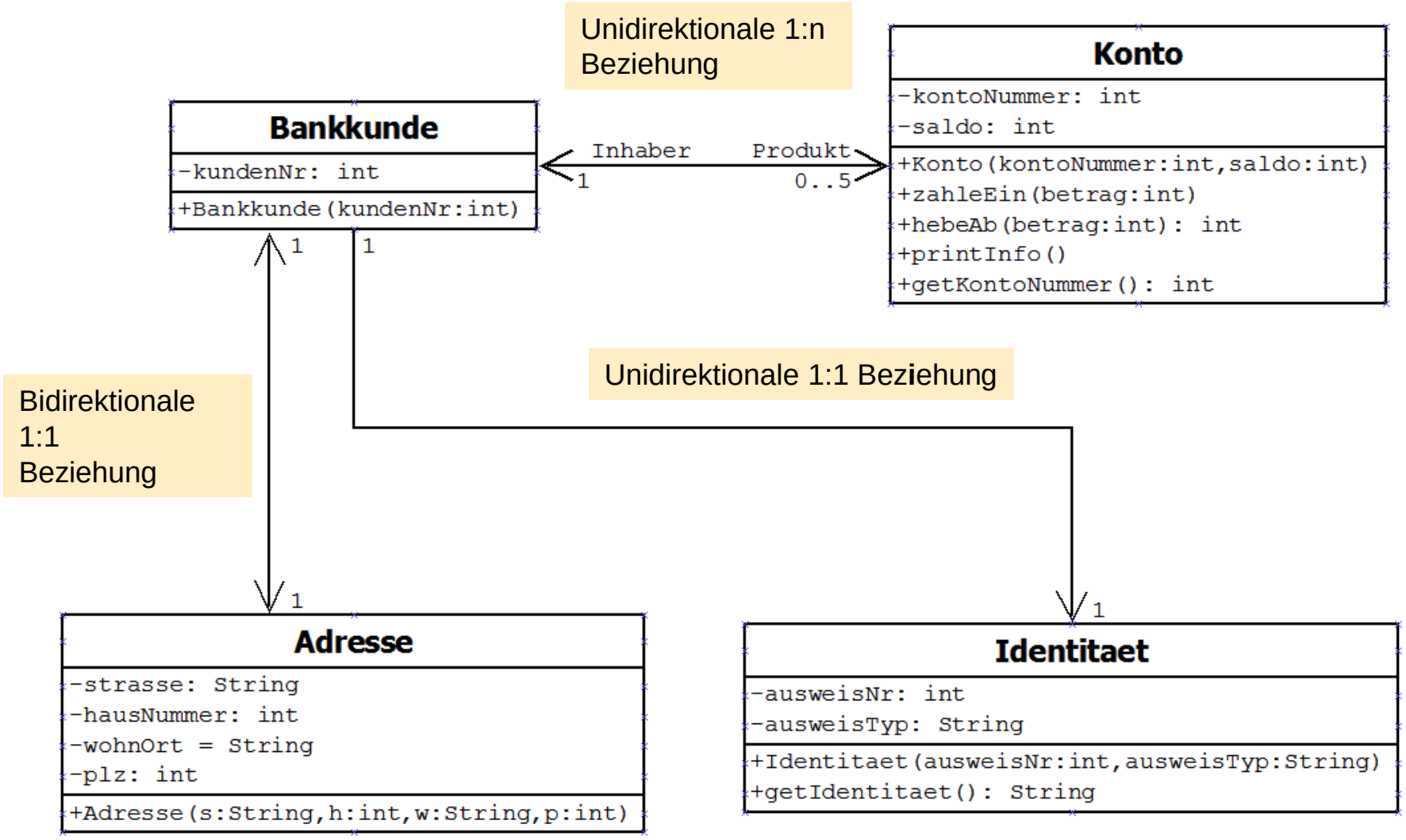
Technische Hochschule Rosenheim
Technical University of Applied Sciences

- **Mögliche Anforderungen**
  - Man soll bei Konto Geld abheben und einzahlen können, siehe Kapitel 2.
  - Jeder Bankkunde darf maximal 5 Konten haben.
  - Von jedem Bankkunden benötigt die Bank die genaue Anschrift. Die Anschrift kann sich aber während der Kundenbeziehung verändern.
  - Jeder Bankkunde hat eine eindeutige Identität.
  - ...
  - + zahlreiche weiteren Anforderungen!!

- **Fragen**
  - Welche Stakeholder / logischen Entitäten gibt es in der Realität?
  - Welche Objekte benötigen Sie in Ihrer Software?

- **Brainstorming:** Klassendiagramm gemeinsam entwickeln.

Verfeinertes UML-Klassendiagramm



IntelliJ Codegenerator

Technische Hochschule Rosenheim
Technical University of Applied Sciences

- IntelliJ Idea bietet Diagramm Unterstützung, allerdings nur in der Ultimate Edition.
- Diese ist bei Registrierung mit der FH Emailadresse kostenfrei erhältlich.

- https://www.jetbrains.com/help/idea/working-with-diagrams.html