Technische Hochschule Rosenheim
Technical University of Applied Sciences

Objektorientierte Programmierung
Kapitel 1 – Objektorientierung

Prof. Dr. Kai Höfig

Objektorientierte Programmierung (OOP) in Java

Technische Hochschule Rosenheim
Technical University of Applied Sciences

- Etwas zur Geschichte:
  - Java ist nicht die erste objektorientierte Sprache (OO-Sprache)
  - C++ war nicht die erste
  - Klassischerweise gelten Smalltalk und insbesondere Simula-67 aus dem Jahr 1967 als Stammväter aller OO-Sprachen
  - Die eingeführten Konzepte sind bis heute aktuell

1967

  - Aber sind sie auch gut und erfüllen ihren Zweck?
    - https://softwareengineering.stackexchange.com/questions/7618/does-oop-fulfill-the-promise-of-code-reuse-what-alternatives-are-there-to-achie

Warum überhaupt OOP?

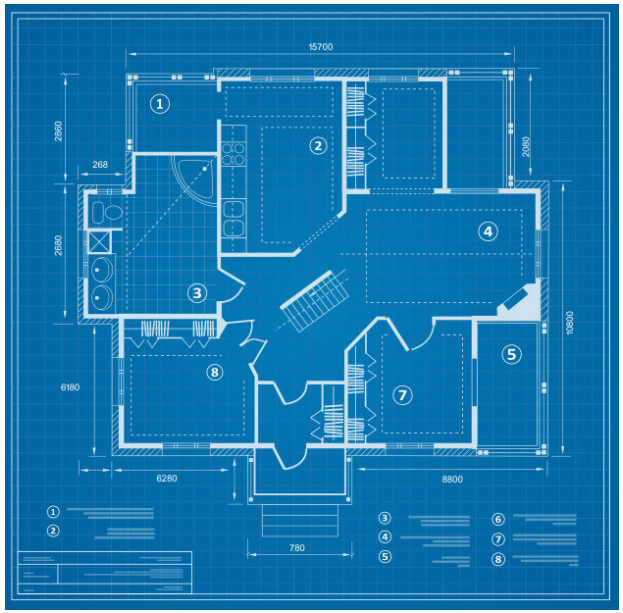
Technische Hochschule Rosenheim
Technical University of Applied Sciences

- Menschen nehmen die Welt in Objekten wahr
  - Objektorientiertes Design mit prozeduralen Systemen ist schwierig (Programme, Unterprogramme,..)
  - Programm-Design wird durch Objekte und Klassen einfacher
  - Trotzdem ist die Übertragung der Realität 1:1 in eine objektorientierte Softwarearchitektur nicht immer sinnvoll oder machbar.
  - Beispiel Hausboot: Ist das jetzt ein Objekt der Klasse Boot oder ein Objekt der Klasse Haus?

OOP Prinzipen

Technische Hochschule Rosenheim
Technical University of Applied Sciences

- OOP stützt sich auf die Konzepte von Objekten und Klassen (Typedefinition von Objekten).
  - Es gilt:
    - Alles ist ein Objekt (manchmal gibt es Ausnahmen, z.B. Basistypen)
    - Objekte kommunizieren durch das Senden und Empfangen von Nachrichten (Wie funktioniert das in Java?)
    - Jedes Objekt ist die Instanz einer Klasse.
    - Die Klasse definiert die Struktur aller ihrer Instanzen wie eine Blaupause die als Plan für verschiedene Instanzen eines Hauses dient.



Grundprinzipien der Objektorientierung Teil 1

Technische Hochschule Rosenheim
Technical University of Applied Sciences

- **Abstraktion und Generalisierung**
  - Ausschnitt aus der realen Welt
  - Relevante Objekte
  - Relevante, charakteristische Eigenschaften von Objekten.

- **Modularität**
  - Partitionieren in kleinere, weniger komplexe Einheiten
  - Strukturierung durch Objekte, Klassen und Pakete

- **Datenkapselung („Data Hiding")**
  - Zusammenfassen von Daten und Verhalten.
  - Verbergen der Implementierung hinter einer ***Schnittstelle.***
  - Zugriff nur über die Schnittstelle, damit interne Daten konsistent bleiben.

Grundprinzipien der Objektorientierung Teil 2

Technische Hochschule Rosenheim
Technical University of Applied Sciences

- **Vererbung**
  - Repräsentieren von Abstraktionsebenen
  - Klassifizieren von Gemeinsamkeiten und Unterschieden
  - Ordnungsprinzip Vererbung: Ermöglicht die Definition neuer Klassen auf Grundlage von bereits bestehenden Klassen

- **Polymorphie (**"Vielgestaltigkeit")
  - ***Beispiel:*** Lampe / Glühbirne
    - Man kann jede Glühbirne einschrauben, die in die Lampenfassung passt.
    - Verschiedene Glühbirnen verhalten sich dennoch unterschiedlich (brennen hell oder weniger hell).
  - Objektorientierung erlaubt einfaches **Austauschen von Code** ("Glühbirne") solange die ***Schnittstelle*** ("Fassung") gleich bleibt.

Zentrale Ziele

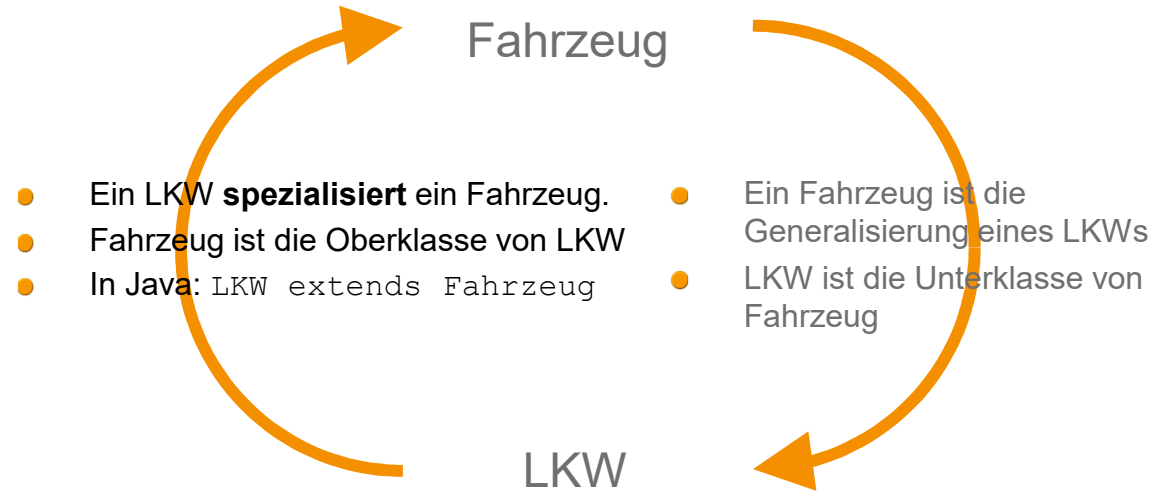
Technische Hochschule Rosenheim
Technical University of Applied Sciences

- Wie macht man Code ***wiederverwendbar***?
  - Erweiterung von bestehendem Code
  - Modifikation von bestehendem Code

- Wie vermeidet man ***Redundanzen*** im Code?
  - ***Prinzip der einzigen Verantwortung***
  - Schwierige Wartung, falls 2 Module existieren, die die gleiche Aufgabe erfüllen.

Zentrales Vorgehen (im Alltag): **Generalisierung** und **Spezialisierung**

- Beispiel: Ein `Auto` und ein `Motorrad` sind Spezialisierungen eines `Fahrzeugs`. Ein `Fahrzeug` ist Generalisierung eines `Autos/Motorrads`.
- Spezialisierungen verfügen über alle Merkmale der Generalisierung, haben aber weitere Merkmale.



OOP in Java

Technische Hochschule Rosenheim
Technical University of Applied Sciences

- **Das Konzept der OOP lehnt sich stark an Strukturierungs- und Klassifizierungsmethoden aus der alltäglichen (menschlichen) Betrachtungsweise unserer realen Welt an.**

- OOP wird in Java mittels *Klassen* und *Objekten* realisiert.

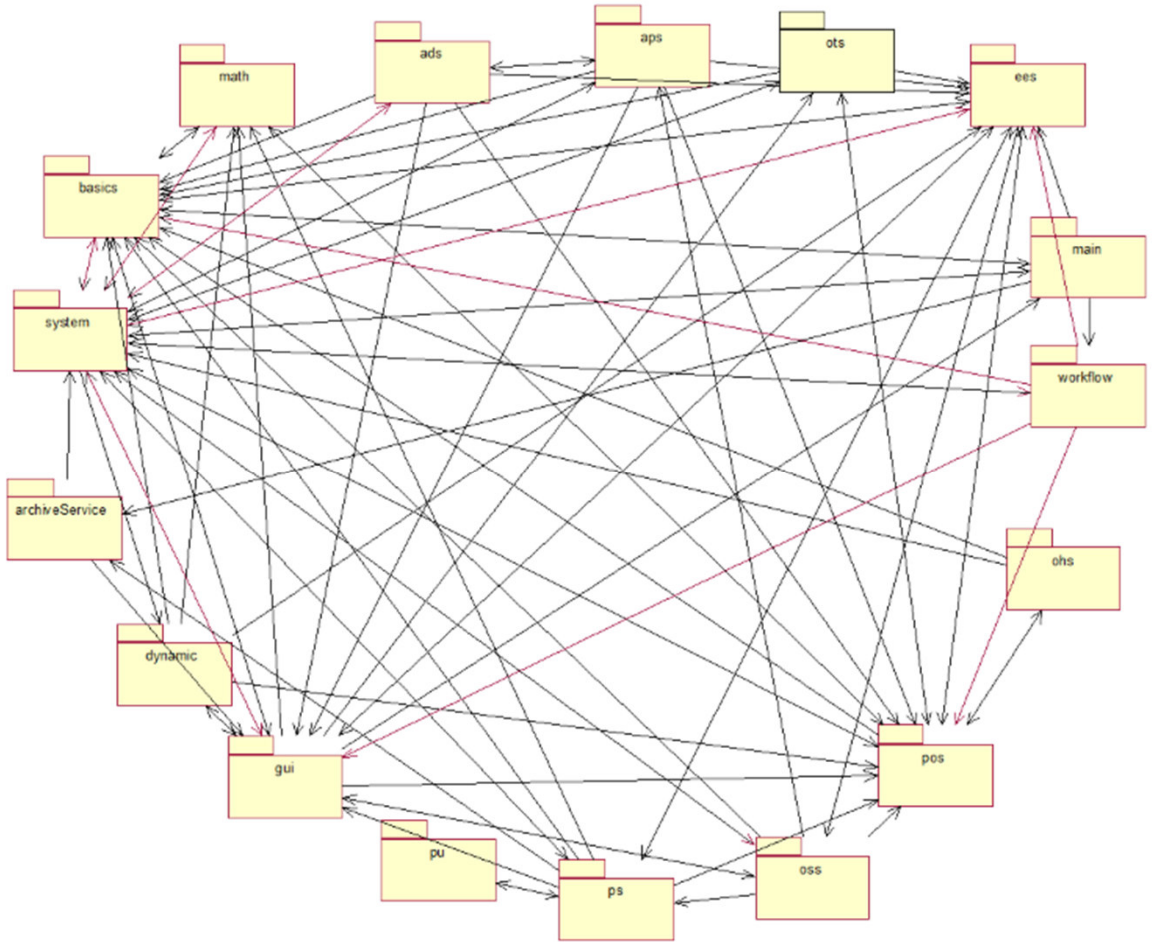
- *Klassen* spezifizieren
  - die Struktur (*Attribute*)
  - die Hierarchie (*Vererbung*)
  - und Abhängigkeiten (*Referenzen*)

- Ein *Objekt* ist die Instanz einer *Klasse* und somit die konkrete Ausprägung einer Klasse.
  - Jedes Objekt hat eine **Identität** (bleibt erhalten während der Lebenszeit!)
  - Jedes Objekt hat einen **Zustand** (Bildet eine Einheit von Daten und Funktionalität)
  - Jedes Objekt hat ein **Verhalten**
  - Jedes Objekt bietet eine Schnittstelle (Interface) zur Interaktion

Motivation: "Bad design smells!"

Technische Hochschule Rosenheim
Technical University of Applied Sciences

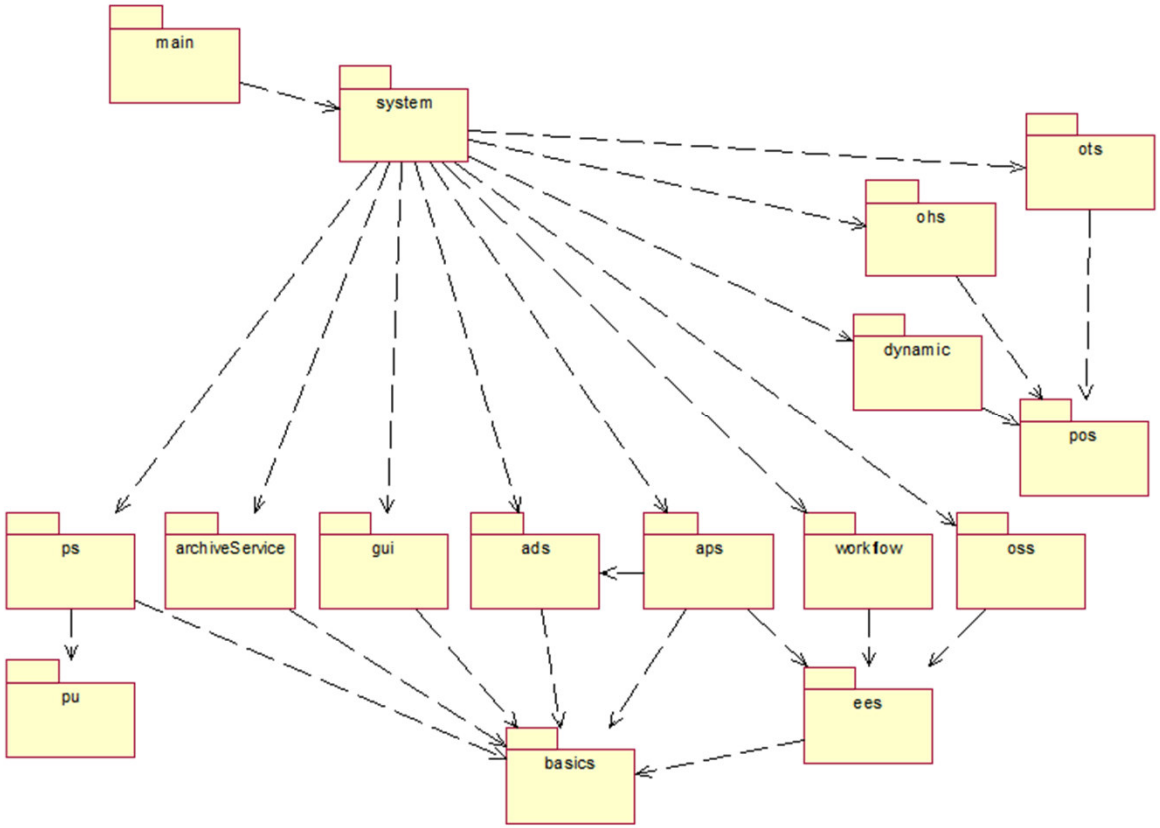
- Beispiel hohe Kopplung



Good Design

Technische Hochschule Rosenheim
Technical University of Applied Sciences

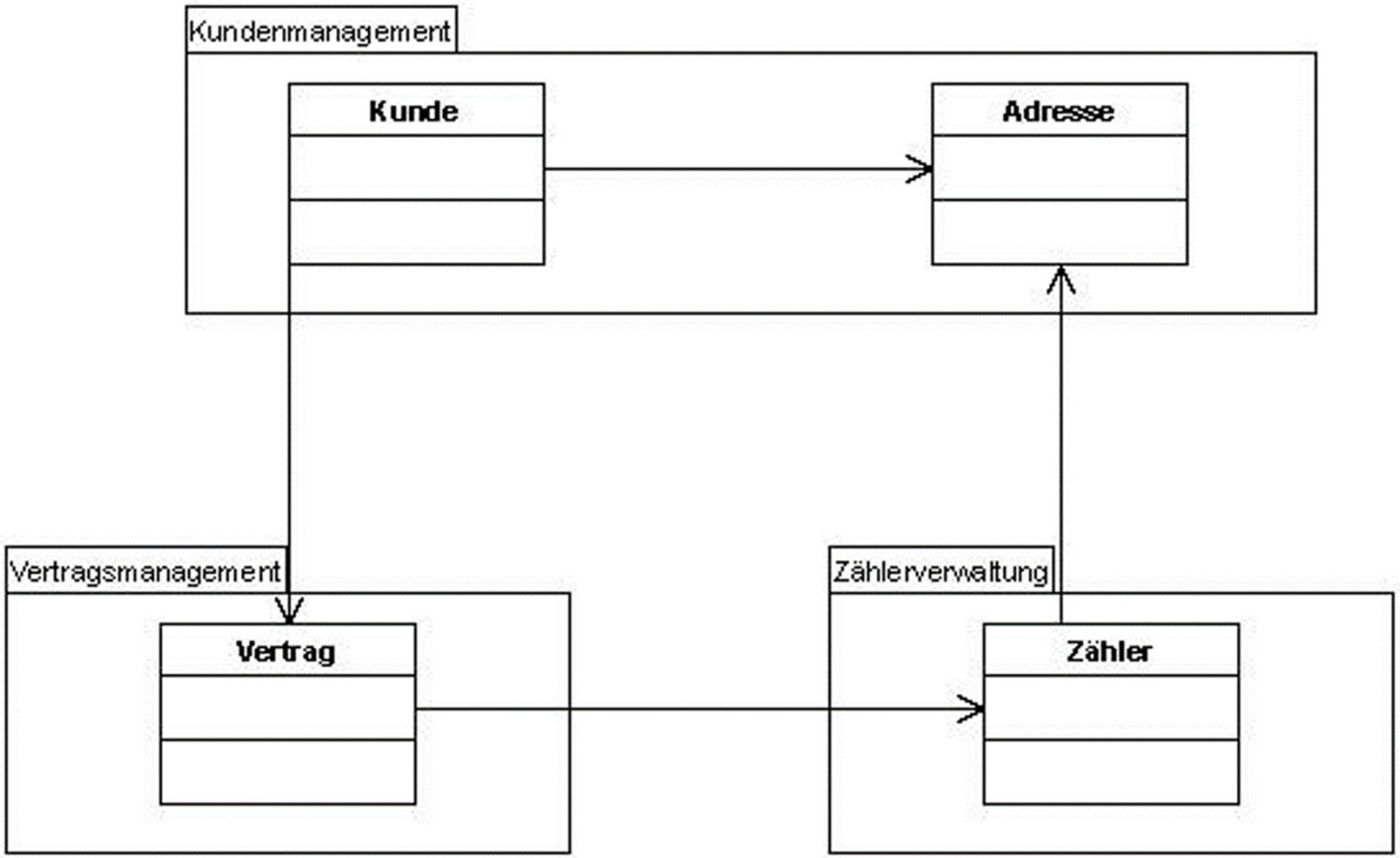
- Beispiel geringe Kopplung



Bad Design: Wie kann das passieren?

Technische Hochschule Rosenheim
Technical University of Applied Sciences

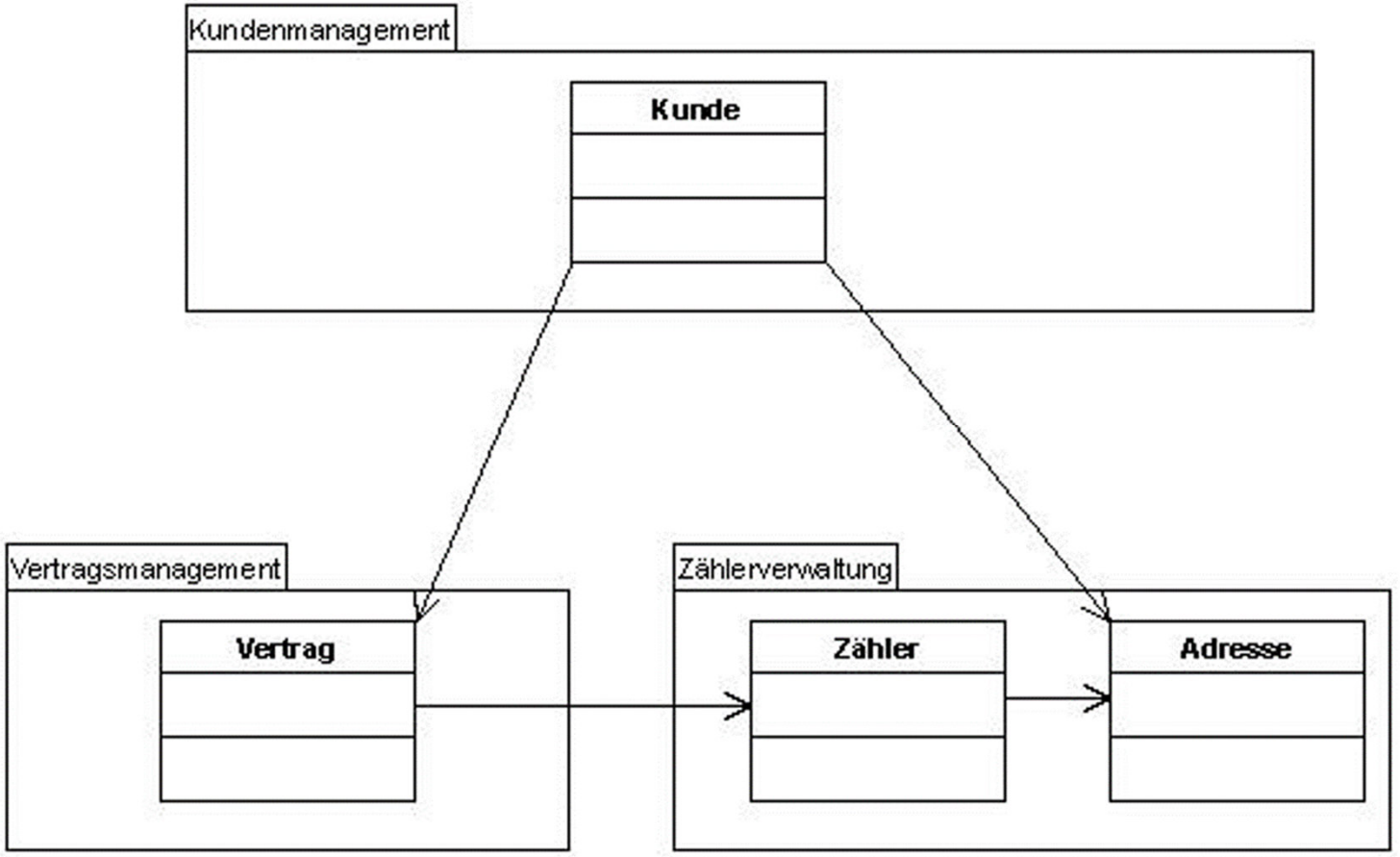
- Die Adresse wird auch zur Darstellung des Zählerstandortes verwendet.



Good Design

Technische Hochschule Rosenheim
Technical University of Applied Sciences

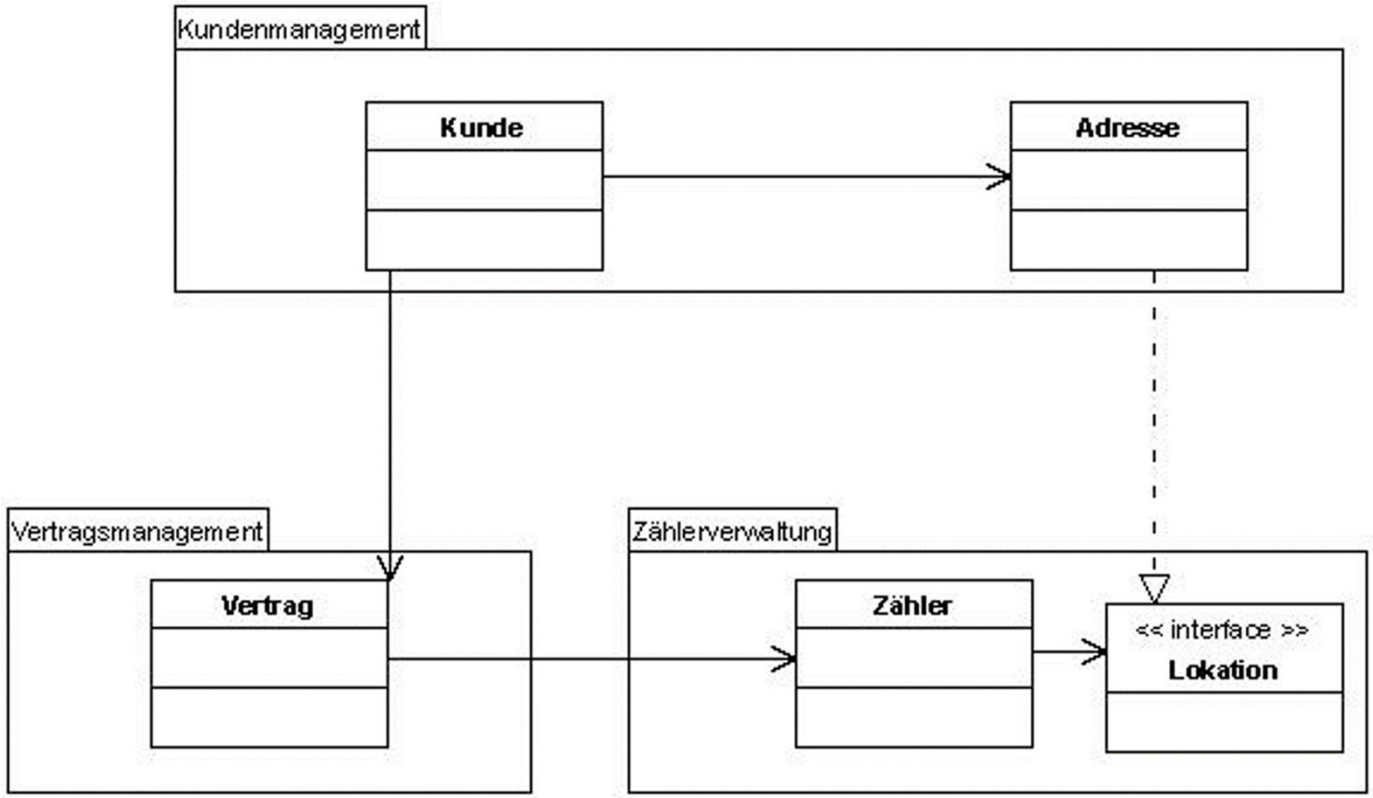
- Adresse gehört hier starker zum Zähler als zu einem Kunden.



Better Design

Technische Hochschule Rosenheim
Technical University of Applied Sciences

- Besser ist hier die Trennung über ein Interface, dann ist die Implemetierung austauschbar (Polymorphie)



Why does it matter?

Technische Hochschule Rosenheim
Technical University of Applied Sciences

- **Klare Struktur – klare Sprache**
  - Eindeutige Abhängigkeiten
  - modular
- **Effekt**
  - Definierte Verantwortlichkeiten
  - Einfachere Wartung
  - Einfachere Änderungen
  - Effizienter
  - Besser zu testen