Softwaretechnik

Teil 2 - Analyse und Design

Top-Down und Bottom-Up

- **Top-Down**
  Entwurf beginnt mit abstrakten Objekten, es folgt eine Konkretisierung

- **Bottom-Up**
  Beginnt mit einzelnen Detailaufgaben, die zur Erledigung übergeordneter Prozesse benötigt werden

Objektorientierte Analyse und Design

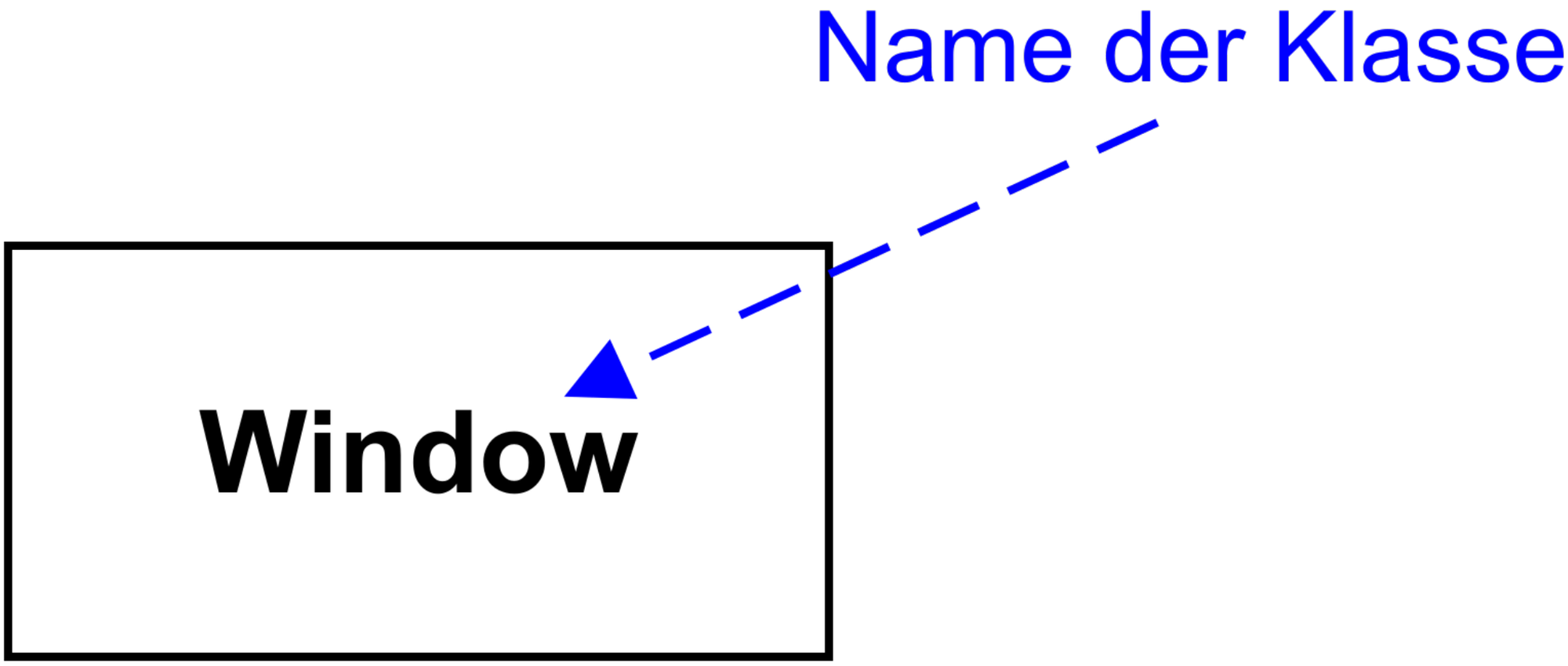
- **Domäne**
  Der Anwendungsbereich des zu entwickelnden Systems

- **Domänenmodell**
  Ein Modell des zu entwickelnden Systems als Design- und Diskussiongrundlage

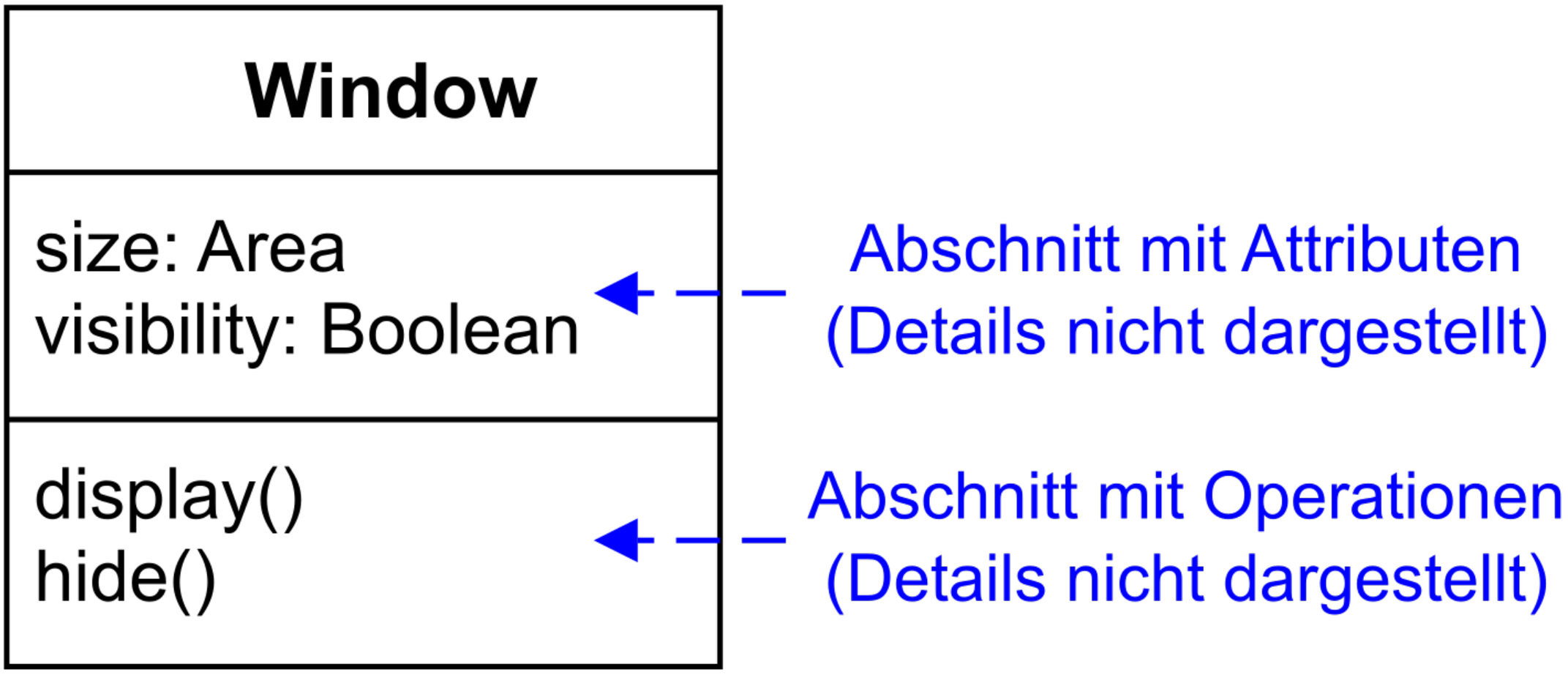
- **UML (Unified Modeling Language)**

Klassendiagramm

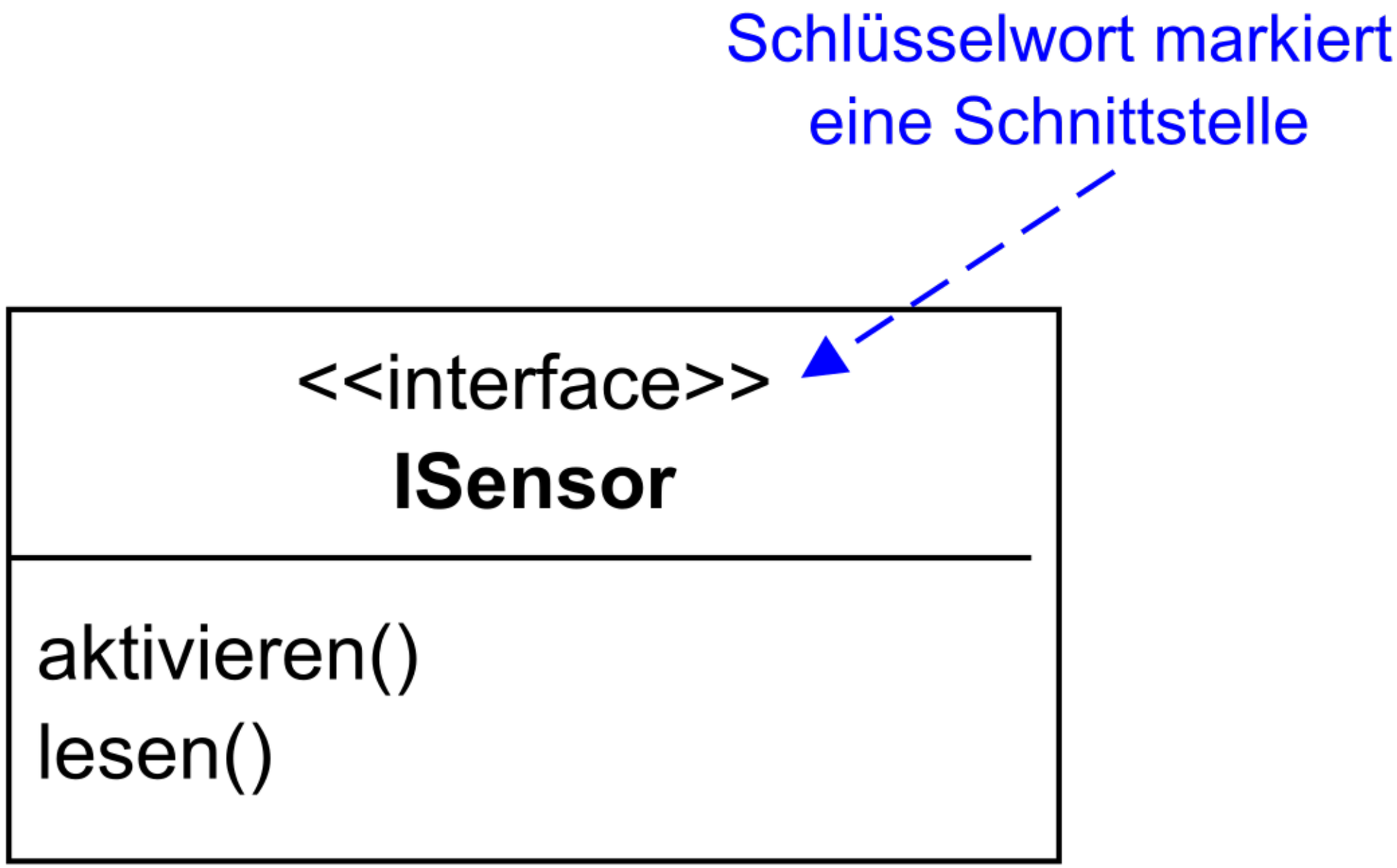
- Modellierung von Klassen, Schnittstellen und deren Beziehungen



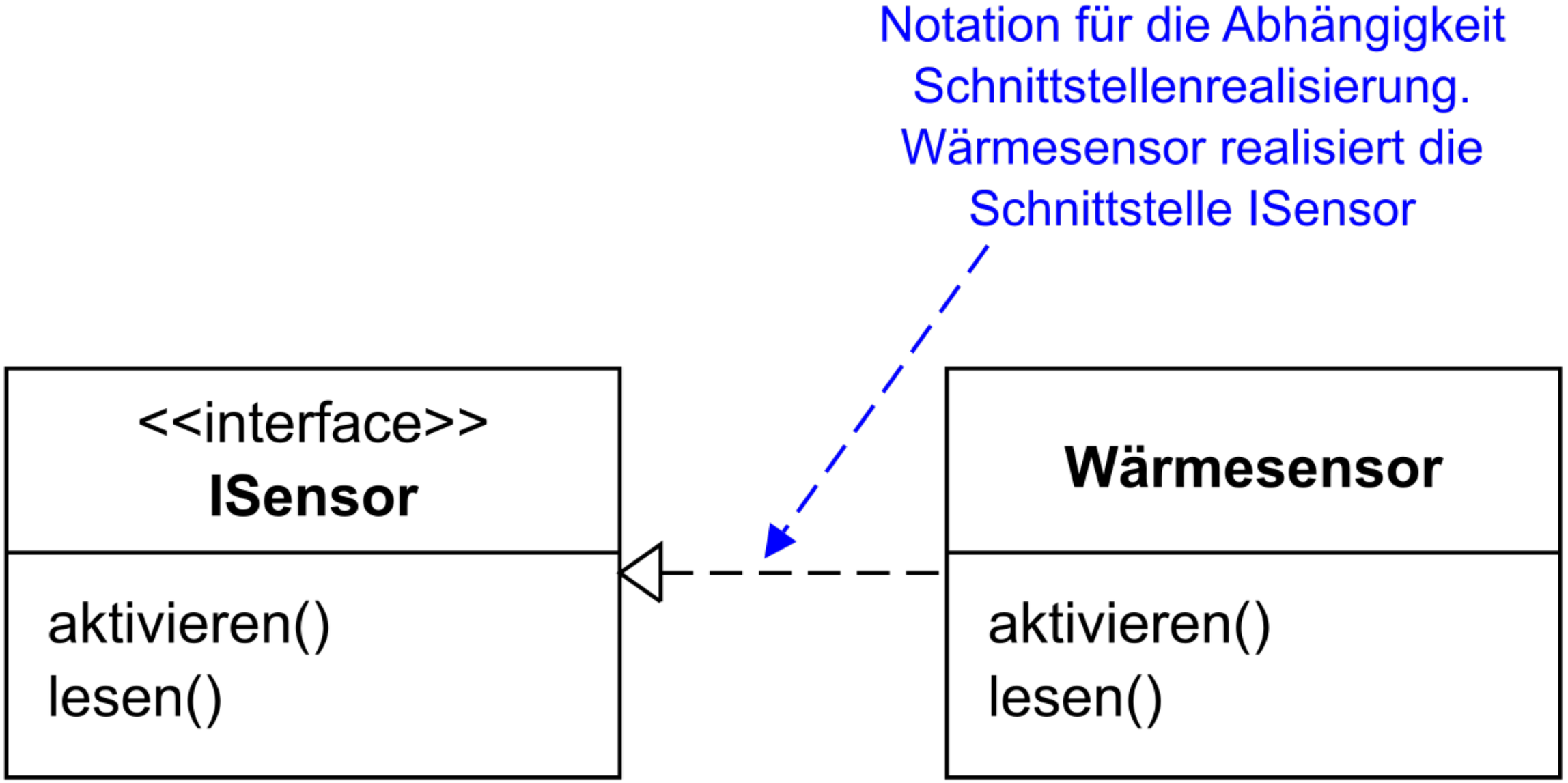
Attribute & Operationen



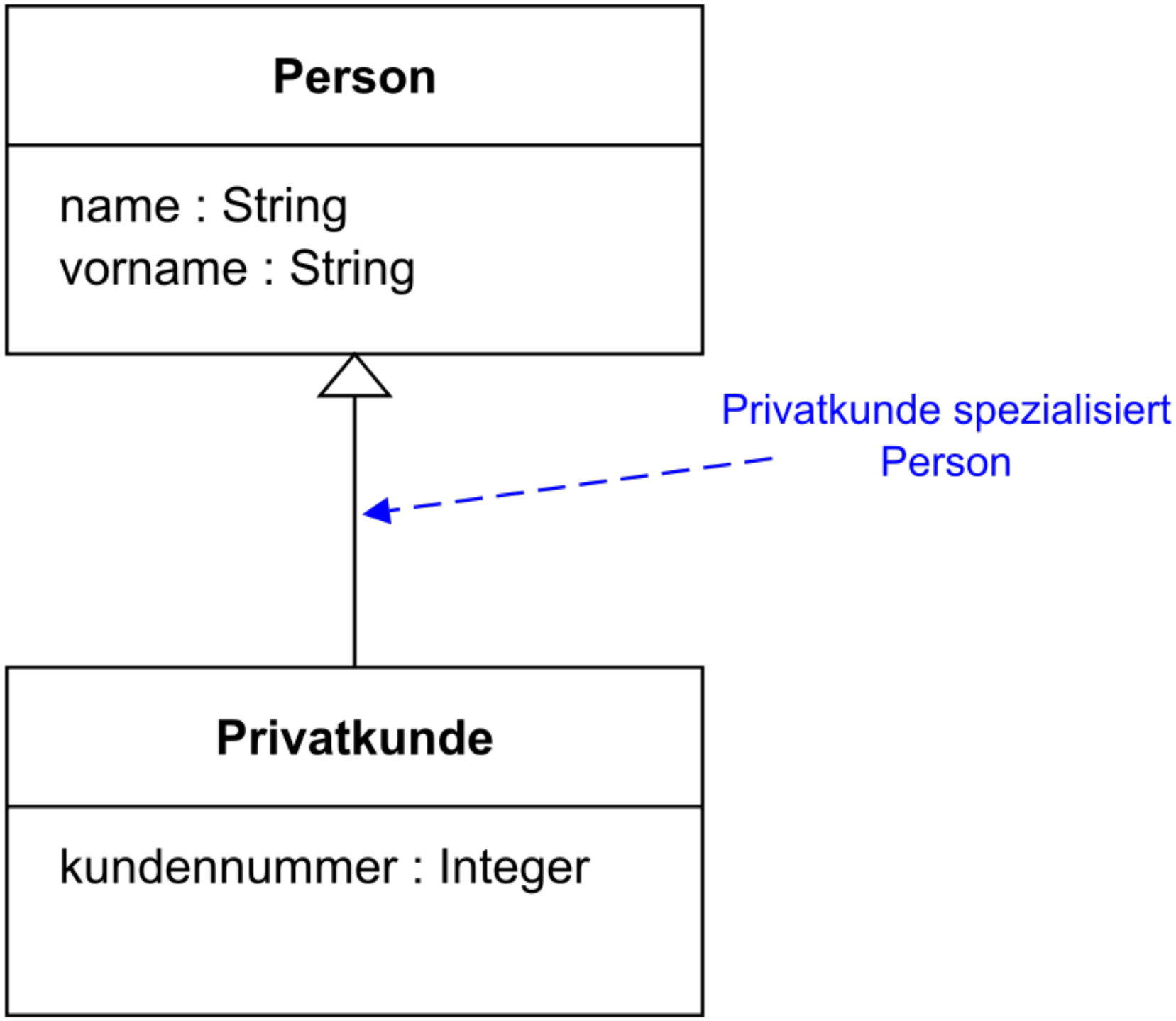
Schnittstellen



Schnittstellenrealisierung

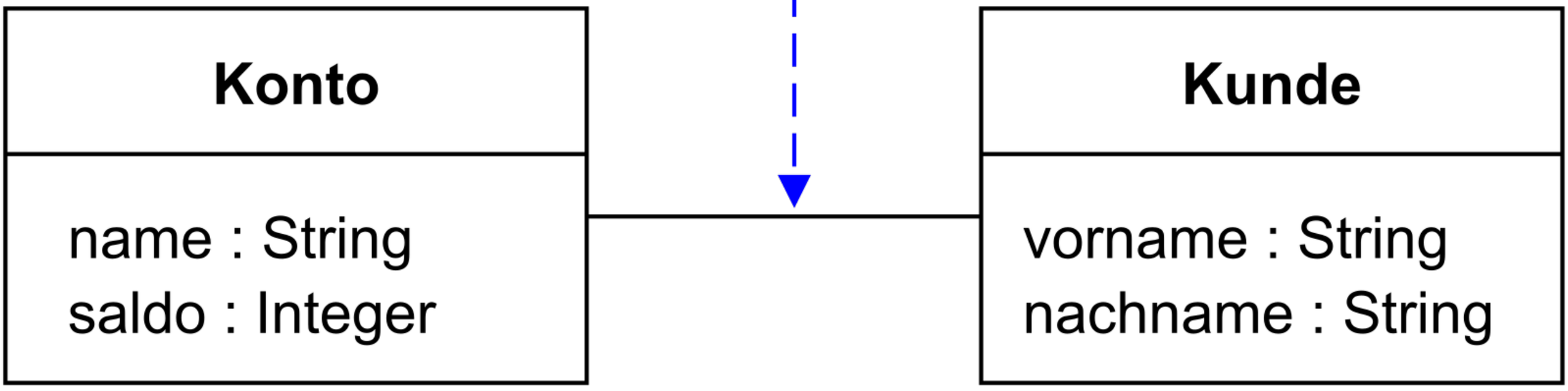


Generalisierung

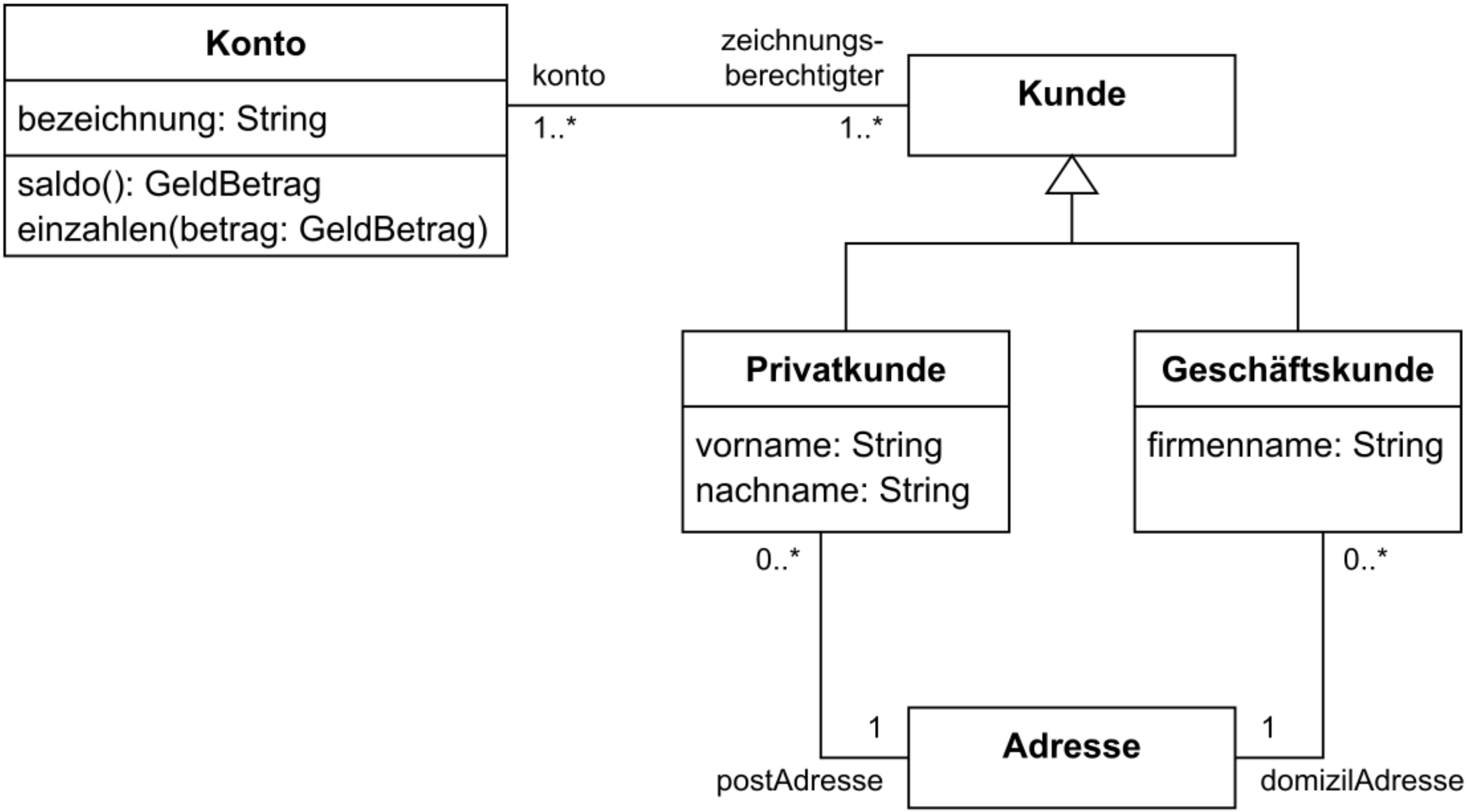


Assoziation

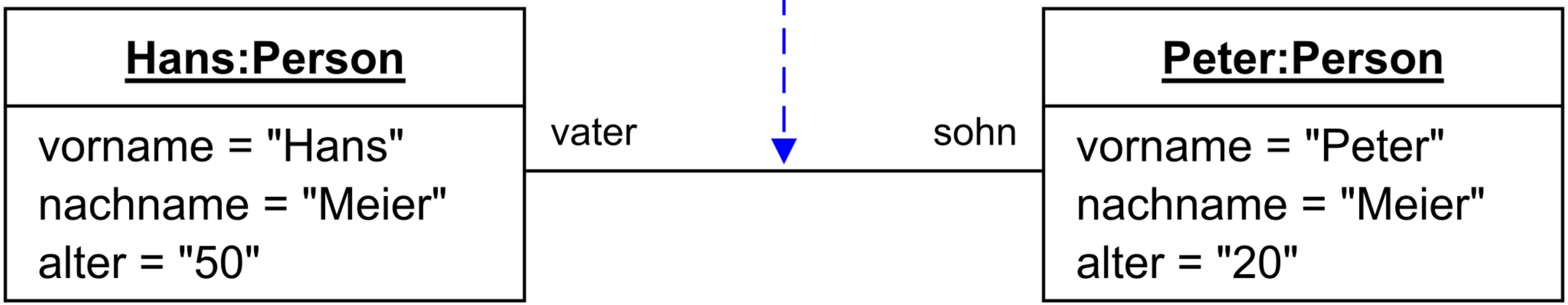
Eine Assoziation zwischen
Konto und Kunde



Beispiel: Klassendiagramm



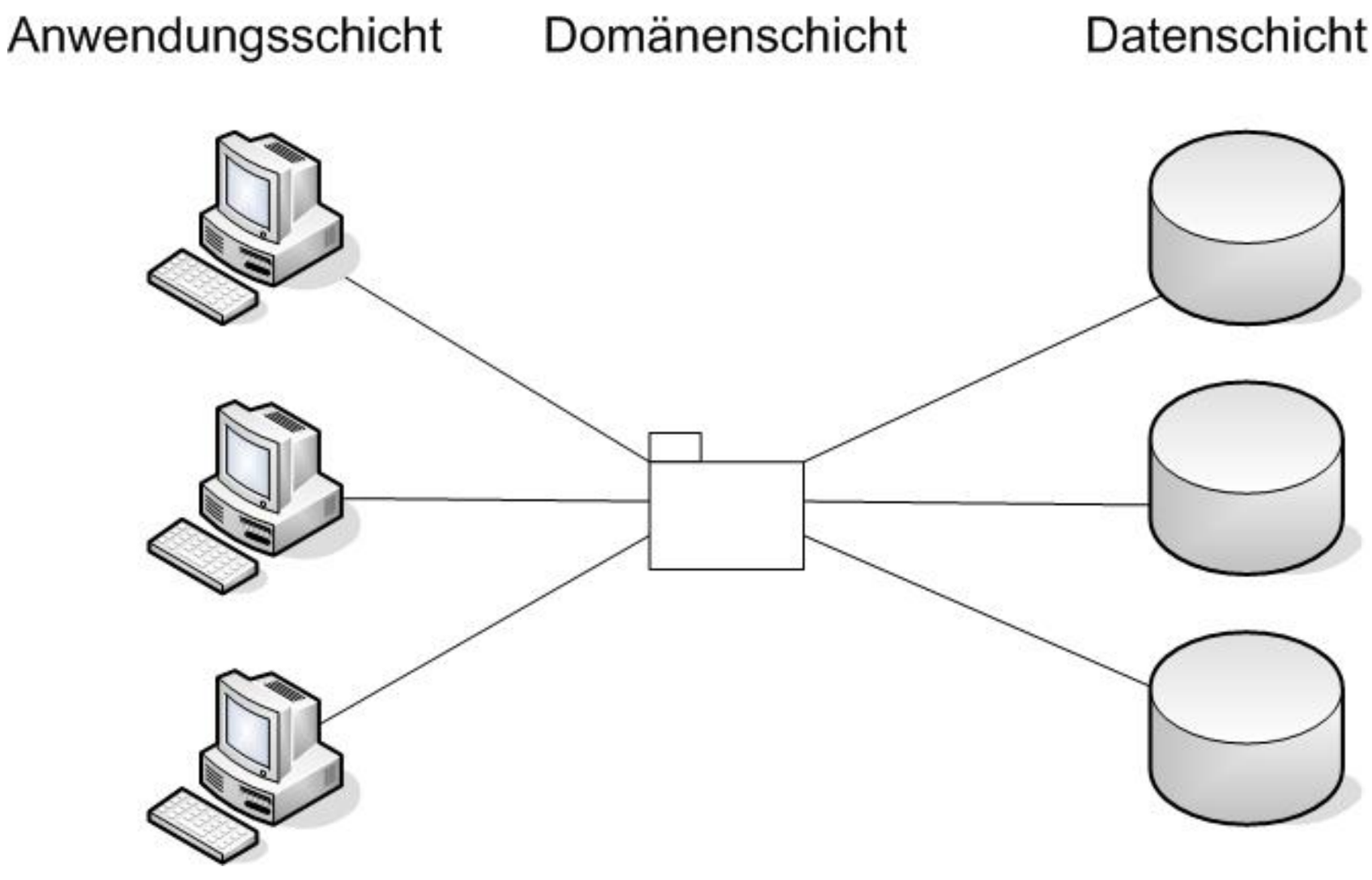
Objektdiagramm



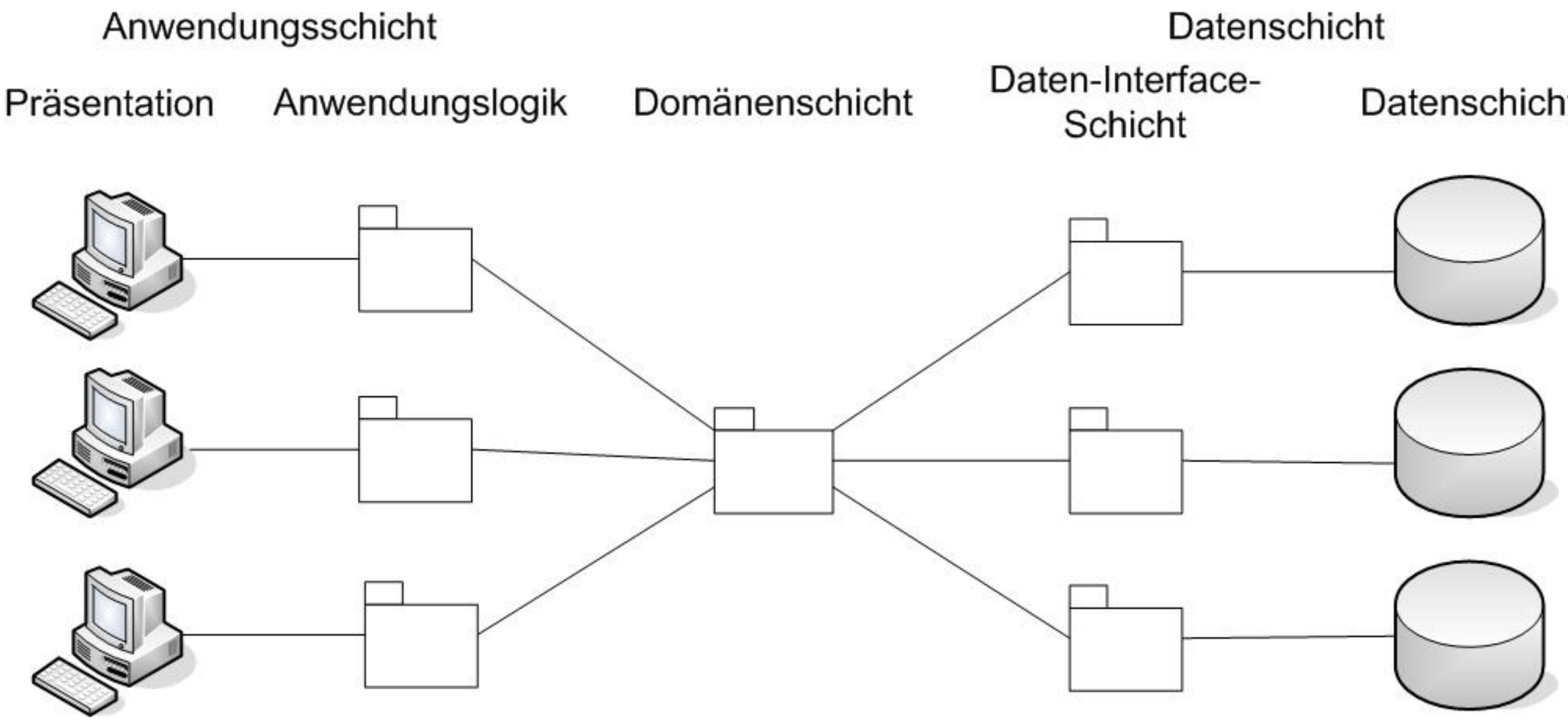
Softwarearchitekturen

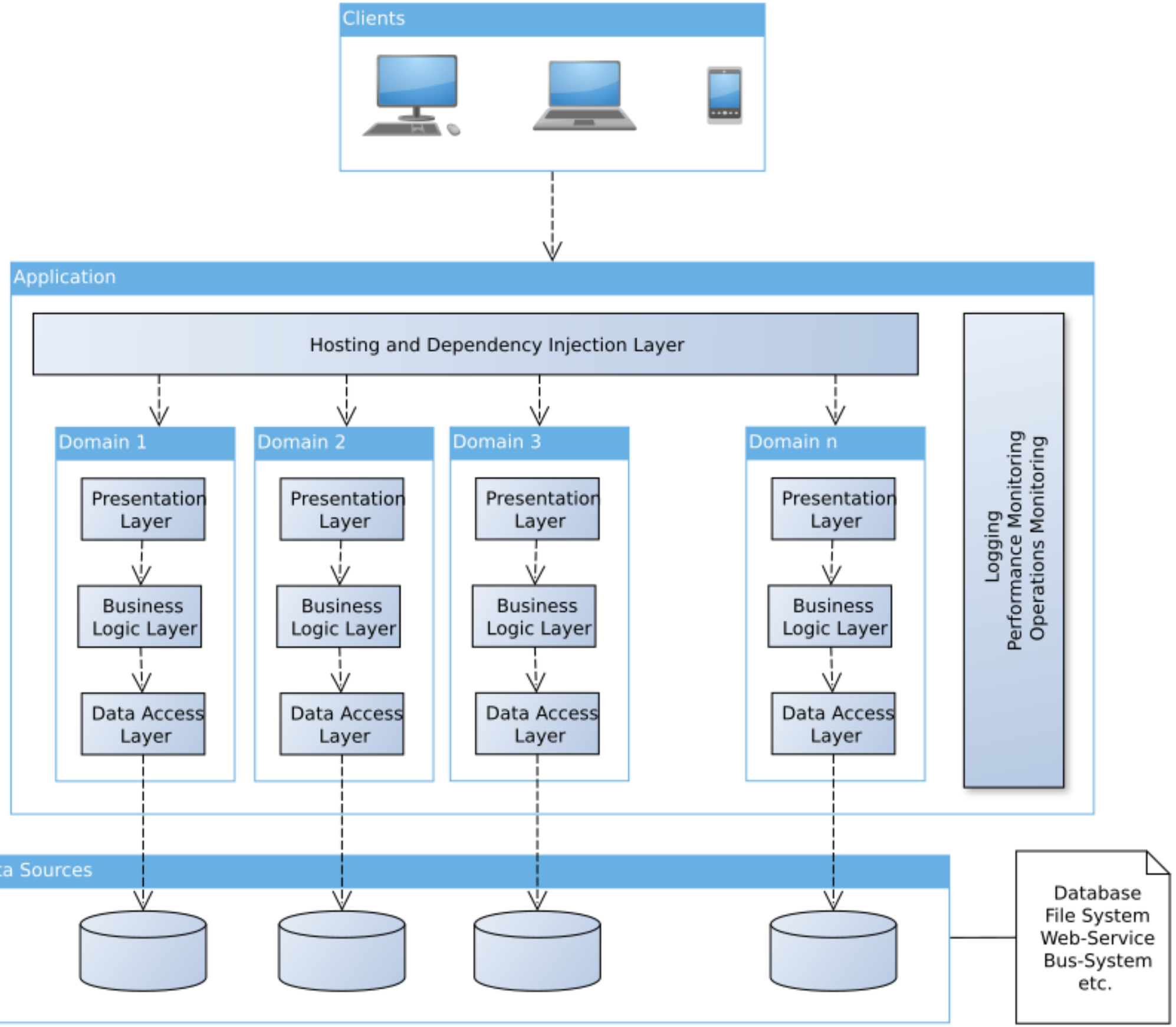
- Schichtenarchitektur
- Model, View, Controller/Presenter
- Domain Driven Design (DDD) & Naked Objects
- Monolith vs. Micro-Services

3 Schichten

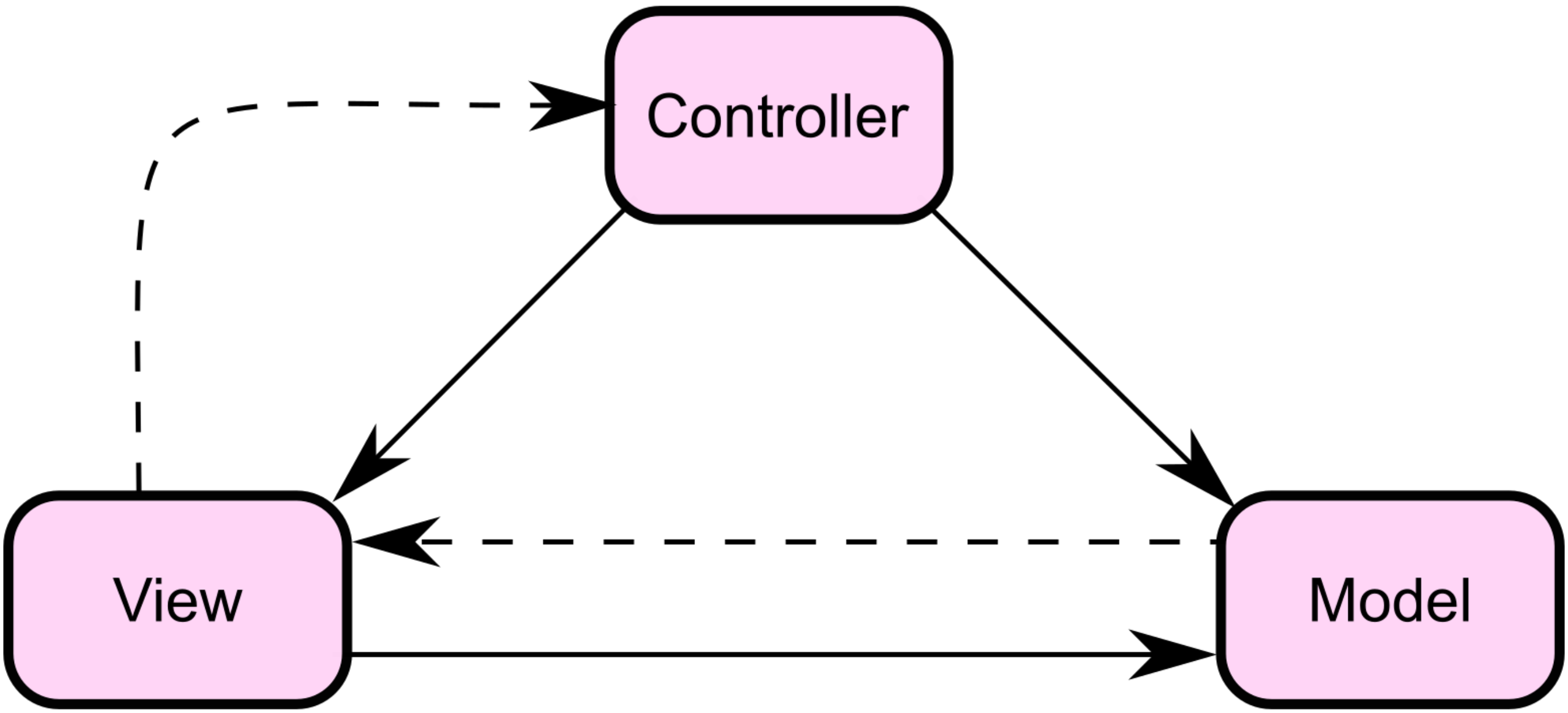


Erweiterte Schichtenarchitektur

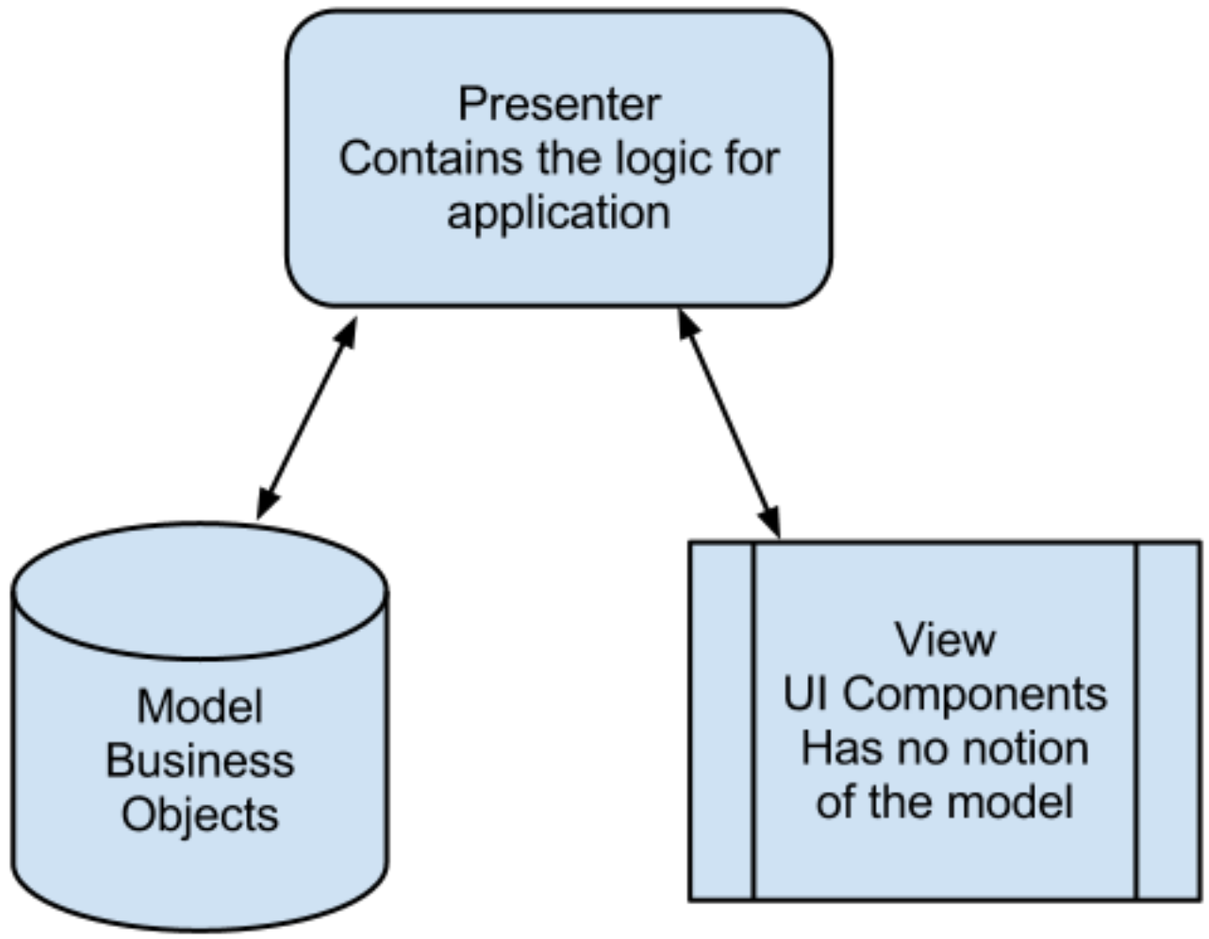



Clients
Application
Hosting and Dependency Injection Layer
Domain 1
Domain 2
Domain 3
Domain n
Presentation Layer
Business Logic Layer
Data Access Layer
Logging
Performance Monitoring
Operations Monitoring
Data Sources
Database
File System
Web-Service
Bus-System
etc.

Model, View, Controller



Model, View, Presenter



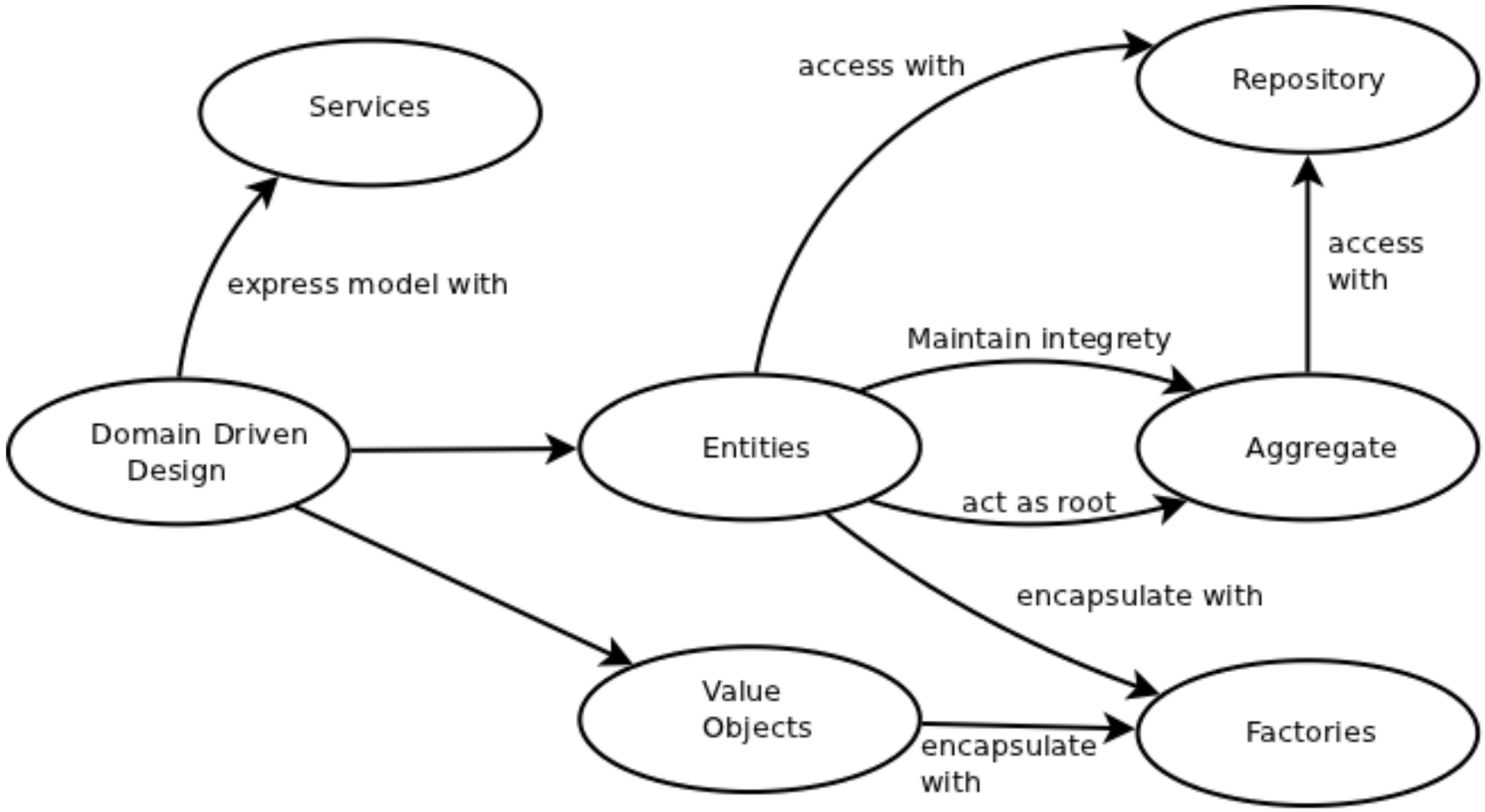
Domain Driven Design

- Der Schwerpunkt des Softwaredesigns liegt auf der **Fachlichkeit** und der Fachlogik.

- Der Entwurf komplexer fachlicher Zusammenhänge sollte auf einem **Modell der Anwendungsdomäne**, dem Domänenmodell basieren.

- Ubiquitäre **Sprache** (“ubiquitous language”) (allgemein verwendeten, allgegenwärtigen)

DDD Map


Services
express model with
access with
Repository
access
with
Maintain integrety
Domain Driven
Design
Entities
Aggregate
act as root
encapsulate with
Value
Objects
encapsulate
with
Factories

DDD (1)

- **Entitäten** (“entities”)
  Werden nicht durch Eigenschaften sondern durch ihre Identität definiert
  Bsp.: Person

- **Wertobjekte** (“value objects”)
  Werden durch ihre Eigenschaften definiert.
  Unveränderbar (“immutable”)
  Bsp.: Adresse

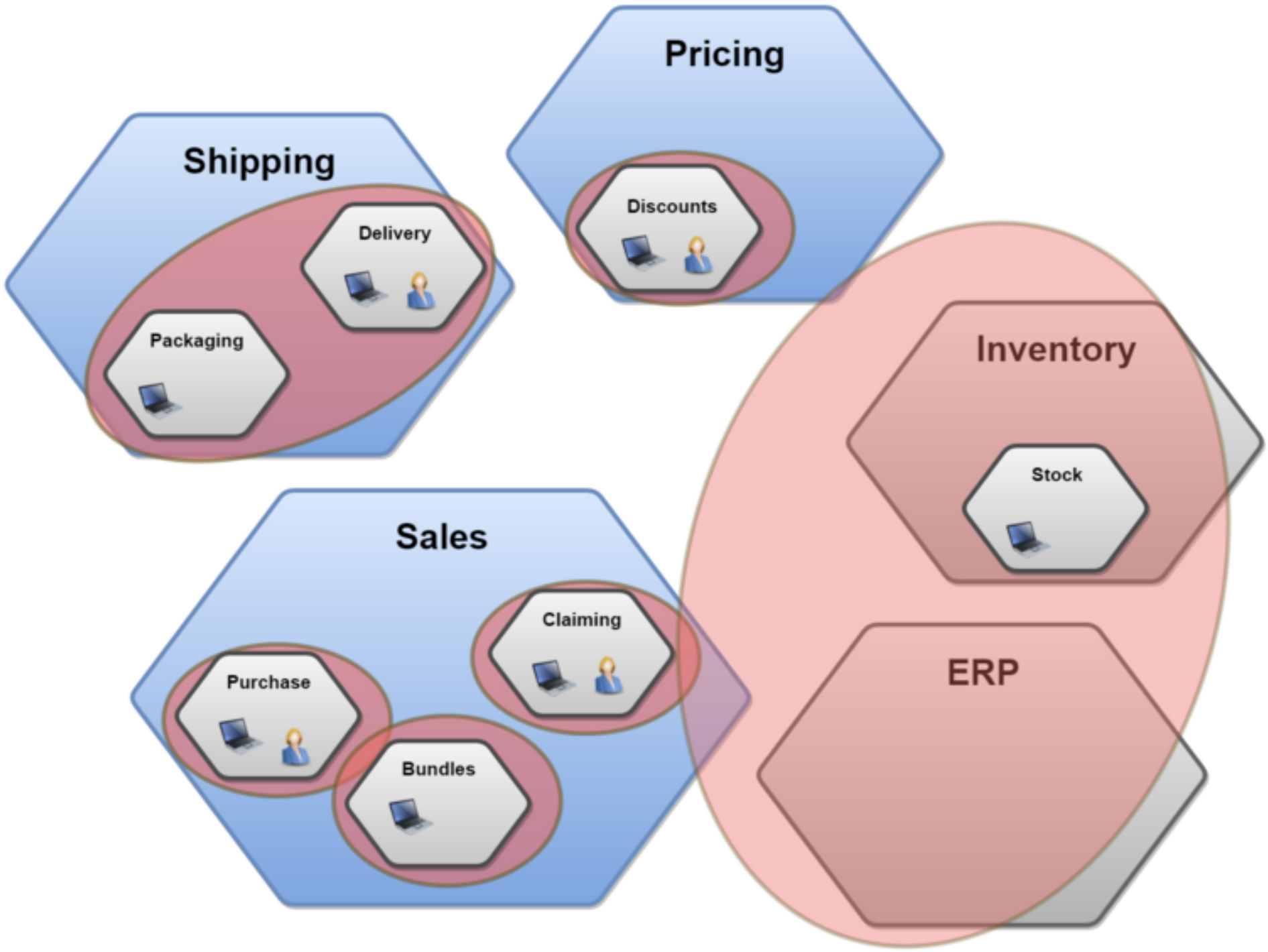
- **Aggregate** (“aggregates”)
  Zusammenfassung von Entitäten und Wertobjekten zu einer Einheit. Zugriff nur via Root-Entität möglich

DDD (2)

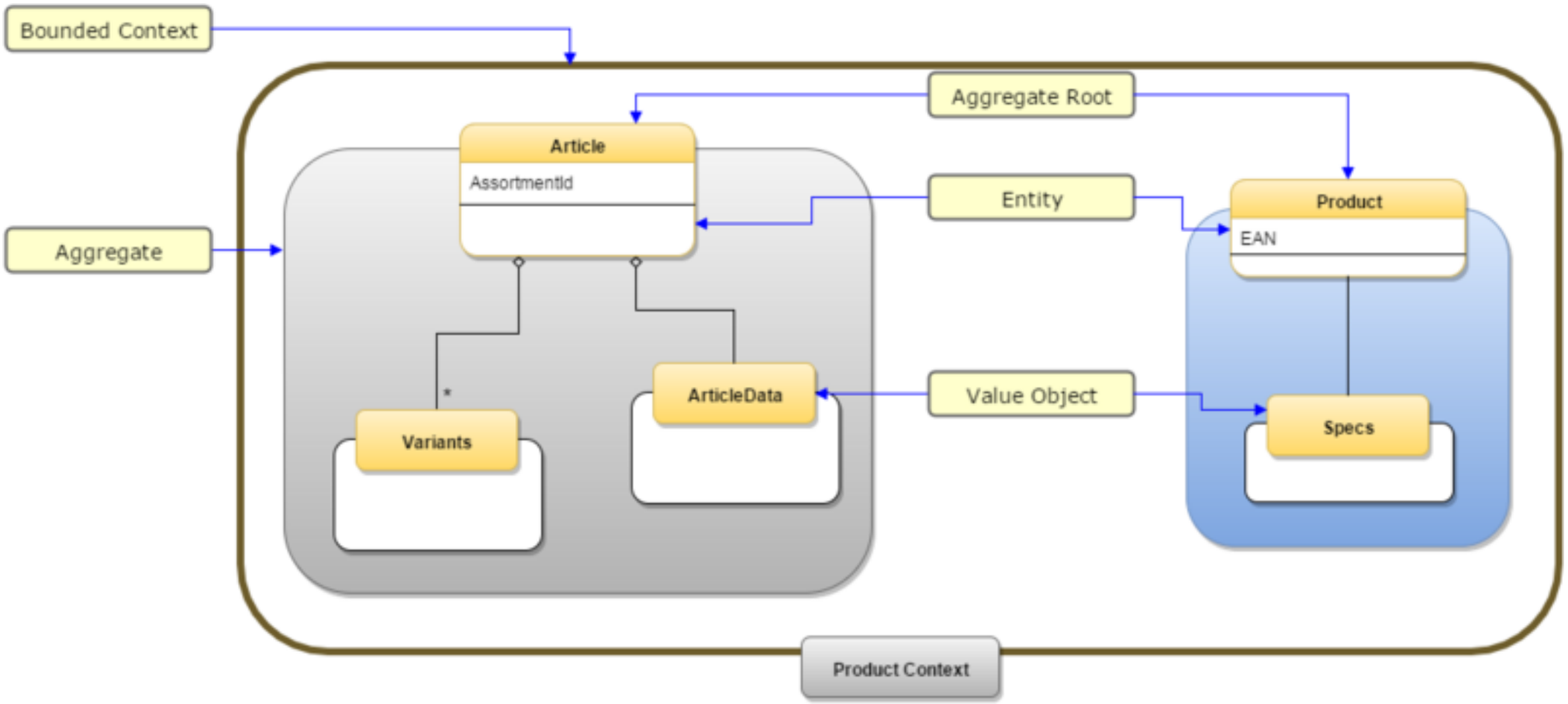
- **Serviceobjekte** (“services”)
  Funktionen der Fachlichkeit die konzeptionell zu mehreren Objekten des Domänenmodells gehören. Sind Zustandslos (“stateless”) und daher wiederverwendbar. Methoden erhalten Entitäten und Wertobjekte.
  Bsp.: Rechnung um Steuerbeträge erweitern

- **Fachliche Ereignisse** (“domain events”)
  Werden direkt in ein Log geschrieben (audit) und dann verarbeitet. Unveränderbar (“immutable”)
  Bsp.: Jemand hat etwas gekauft

Context Map


Pricing
Discounts
Shipping
Delivery
Packaging
Inventory
Stock
Sales
Claiming
Purchase
Bundles
ERP

Beispiel: Product



Entwurfsmuster

- Gang of Four (“Viererbande”)
  Erich Gamma, Richard Helm, Ralph Johnson und John Vlissides

- Entwurfsmuster. Elemente wiederverwendbarer objektorientierter Software
  (Originaltitel Design Patterns. Elements of Reusable Object-Oriented Software)

- Mehr unter: https://de.wikipedia.org/wiki/Prinzipien_objektorientierten_Designs

Kleine Auswahl an Mustern

- **Factory**
  Kapselt Erzeugung von Objekten
- **Singleton**
  Nur eine Objektinstanz
- **Builder**
  Zusammenbau von komplexen Objekten
- **Adapter**
  Brücke zwischen Schnittstellen
- **Decorator**
  Erweitern von Objektfunktionialität ohne Vererbung
- **Facade**
  Erlaubt einfache Verwendung komplexer Systeme
- **Command**
  Befehle als Objekte
- **Observer**
  Benachrichtigt andere Objekte über Änderungen am Objekt
- **Strategy**
  Strategie als Interface und Algorithmen als austauschbare Implementierungen
- **Visitor**
  Iterative Verarbeitung von Objekten mit wechselnden Operationen
- **Dependency Injection**
  Eigene Schicht für das zentrale Erzeugen von Abhängigkeiten

Antipatterns

- Big ball of mud / Spaghetti Code
  http://www.laputan.org/mud/

- Gas factory / Over-engineering

- God Object

- Sumo Marriage (Fachlogik in der DB)

- Integration Database (Geteilte DB)

- Mehr unter: https://de.wikipedia.org/wiki/Anti-Pattern

Mock-Ups - Vorführmodelle

- Einfach und schnell zu bauen
- Wegwerf-Prototypen
- Bieten den Anwendern die Möglichkeit die UI zu testen
- Erleichtert Kommunikation über UI Entwurf
- z.B. Online: https://moqups.com/ oder http://www.framebox.org/