DHBW Stuttgart
Duale Hochschule
Baden-Württemberg

Software Engineering: UML

Dr. Eugenie Giesbrecht

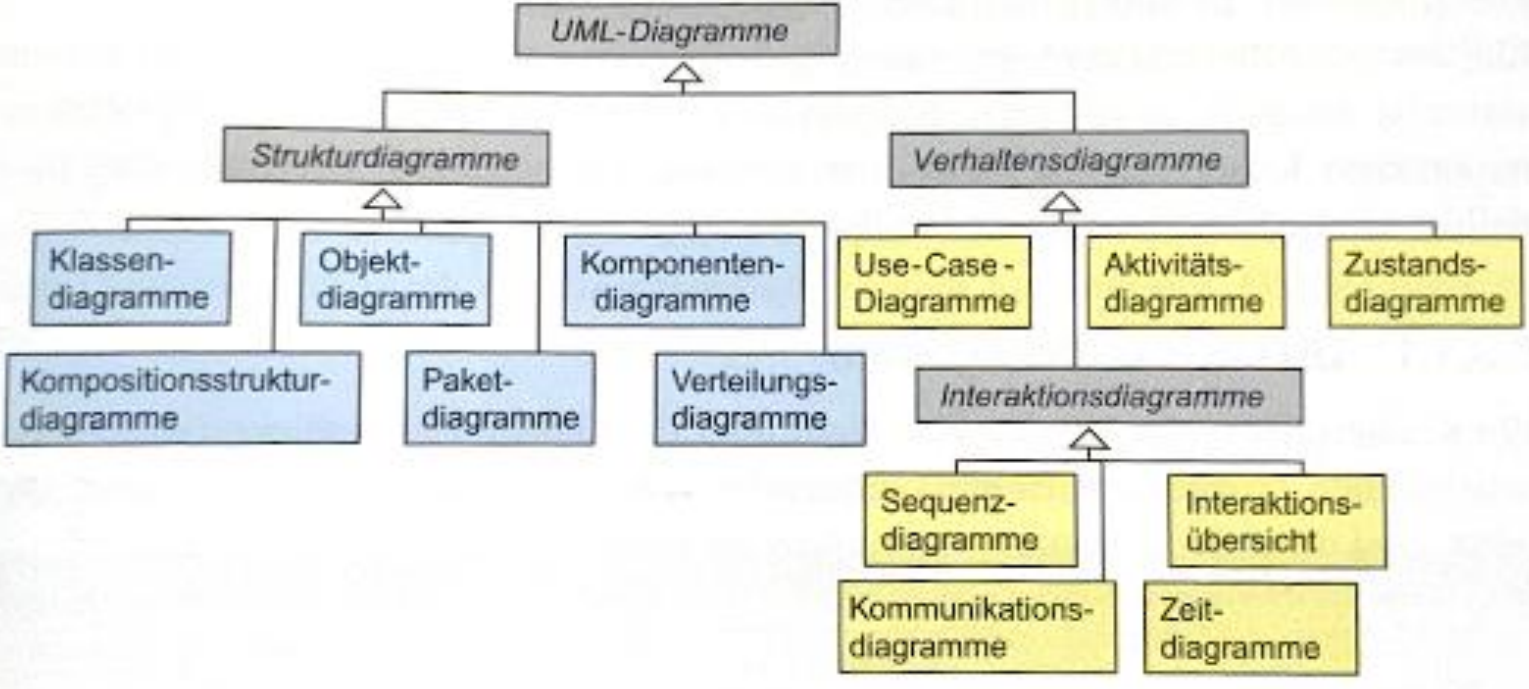
UML (Unified Modeling Language)

- Standardisierte grafische Sprache, die verwendet wird, um **Software, Systeme oder Prozesse zu modellieren**
- 1997 wurde UML als Standard von der **Object Management Group (OMG)** eingeführt
- 2005 wurde UML von der ISO (International Organization for Standardization) als Standard **ISO/IEC 19501** anerkannt

Anwendung von UML

- Präzise Anforderungserfassung (z. B. mittels Use-Case-Diagrammen)
- Entwurf von Domänenmodellen und Systemstrukturen (z. B. Klassendiagramme, Komponentenmodelle)
- Beschreibung von Abläufen und Systemzuständen (z. B. Aktivitäts- und Zustandsdiagramme)
- Nachvollziehbare Dokumentation von Architekturentscheidungen für Entwickler-Teams, Fachbereiche und externe Dienstleister

UML-Diagrammarten



[Metzner2020]

Ein einfaches Beispiel

```
public class Order {
    private String id;
    private Customer customer;
    private List<OrderItem> items;

    public Money calculateTotal() { ... }
}
```

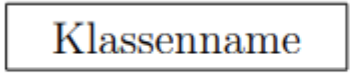
- ein **Klassendiagramm** mit Order, Customer und OrderItem samt Attributen, Operationen und Beziehungen erstellen,
- ein **Sequenzdiagramm**, das zeigt, wie ein Service eine Order erzeugt, validiert und persistiert,
- und ggf. ein **Zustandsdiagramm**, das den Lebenszyklus der Order abbildet

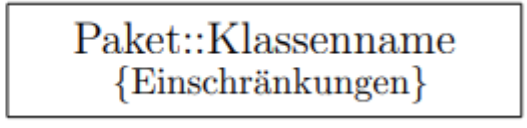
Strukturdiagramme: Klassendiagramm

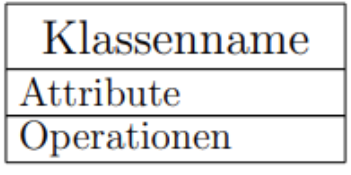
- Zeigt die Klassen, ihre Beziehungen und Methoden auf
- Eines der am häufigsten verwendeten UML-Diagramme

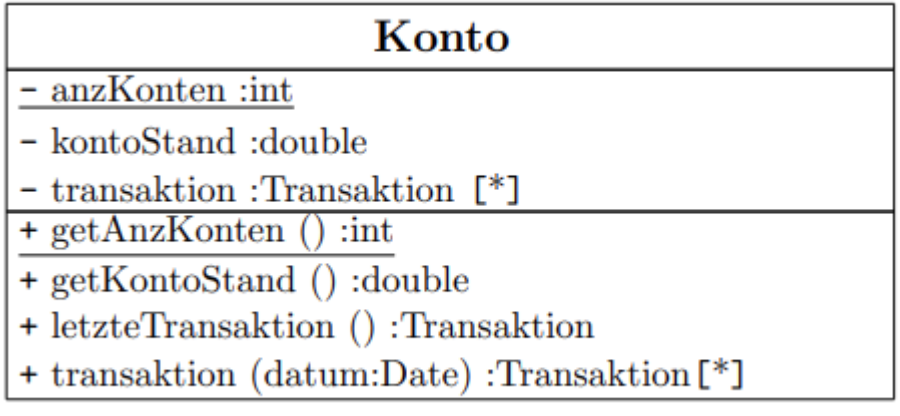
Elemente:

- Klasse
- Attribute







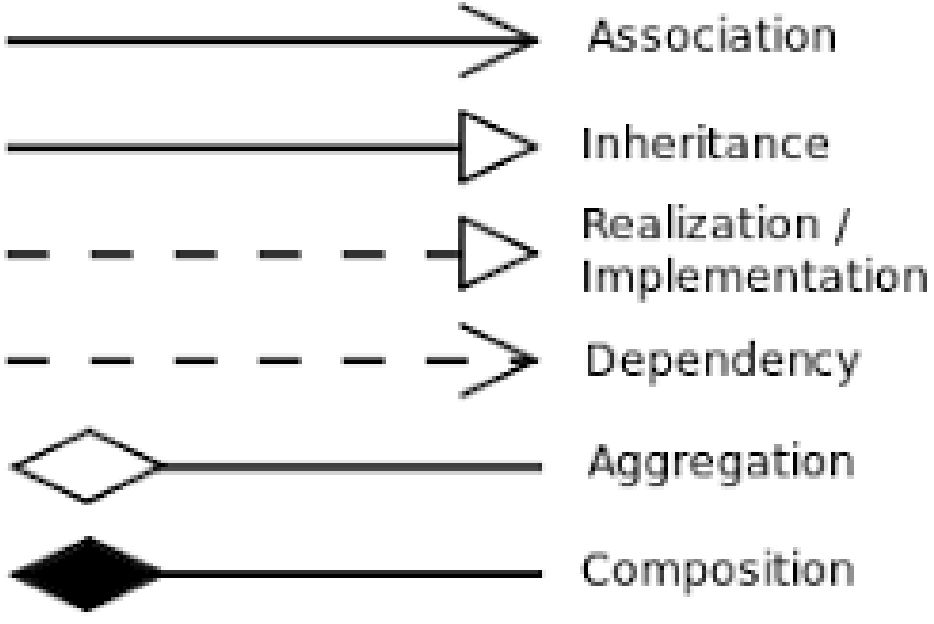


- Sichtbarkeitsbereich:

”+” (public), ”#” (protected),

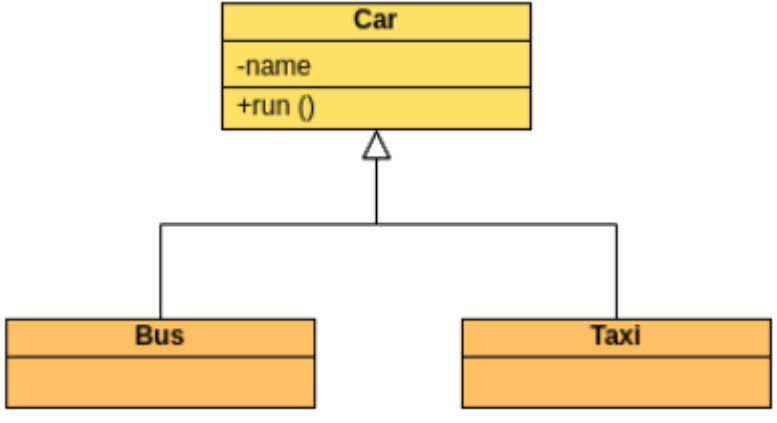
”-” (private) ”~” (package)

Klassendiagramm: Beziehungen



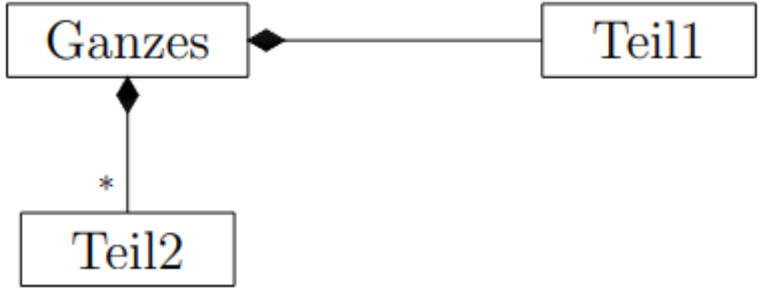
Klassendiagramm: Generalisierung

- Beziehung zwischen Eltern- und Kindklassen
- In einer Vererbungsbeziehung übernimmt die Unterklasse alle Funktionen (Methoden) der Elternklasse
- Unterklassen erweitern die Elternklasse um zusätzliche, spezifischere Eigenschaften

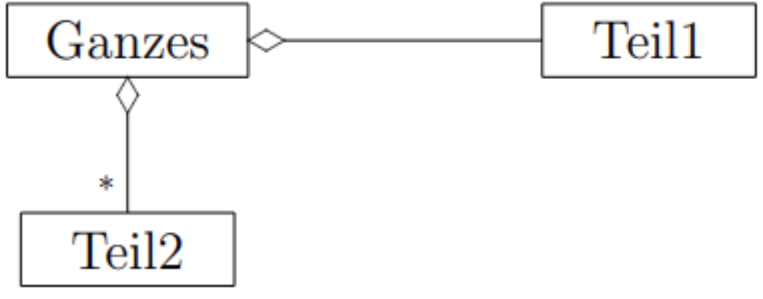


Klassendiagramm: Aggregation & Komposition

- Aggregation



- Komposition

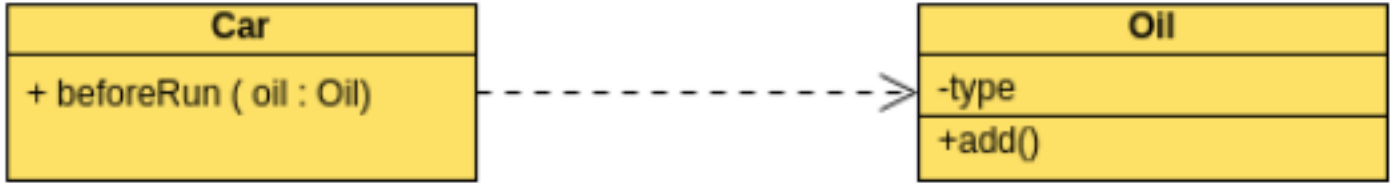


Klassendiagramm: Assoziationen

- am häufigsten verwendete Beziehung zwischen 2 Klassen
- bidirektionale Assoziationen haben zwei Pfeile oder gar keine
- unidirektionale Assoziationen oder Selbstassoziationen haben einen Pfeil
- 1..1: Nur eine
- 0..*: Null oder mehr
- 1..*:ein oder mehr
- 0..1: Nein oder nur eins
- m..n: mindestens m, höchstens n (m<=n)

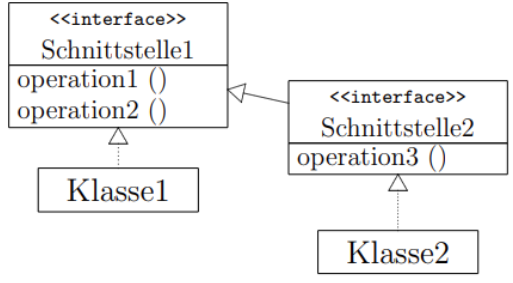
Klassendiagramm: Abhängigkeiten

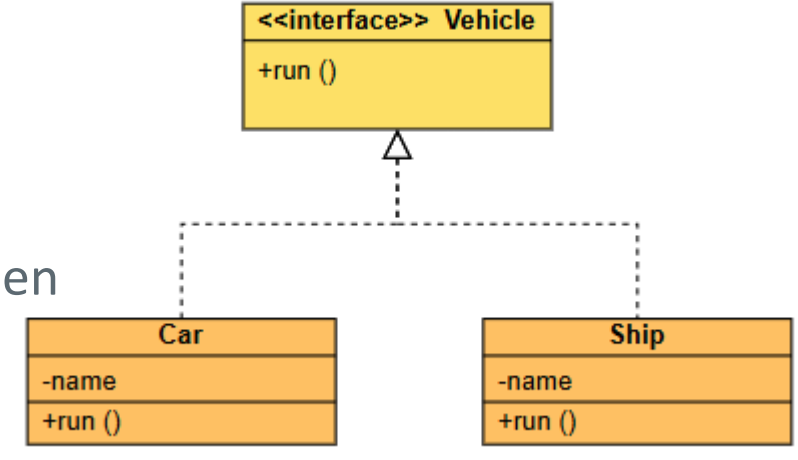
- In den meisten Fällen spiegeln sich Abhängigkeiten in Methoden einer Klasse wider, die das Objekt einer anderen Klasse als Parameter verwenden
- Eine Abhängigkeitsbeziehung ist eine „Nutzungs“-Beziehung



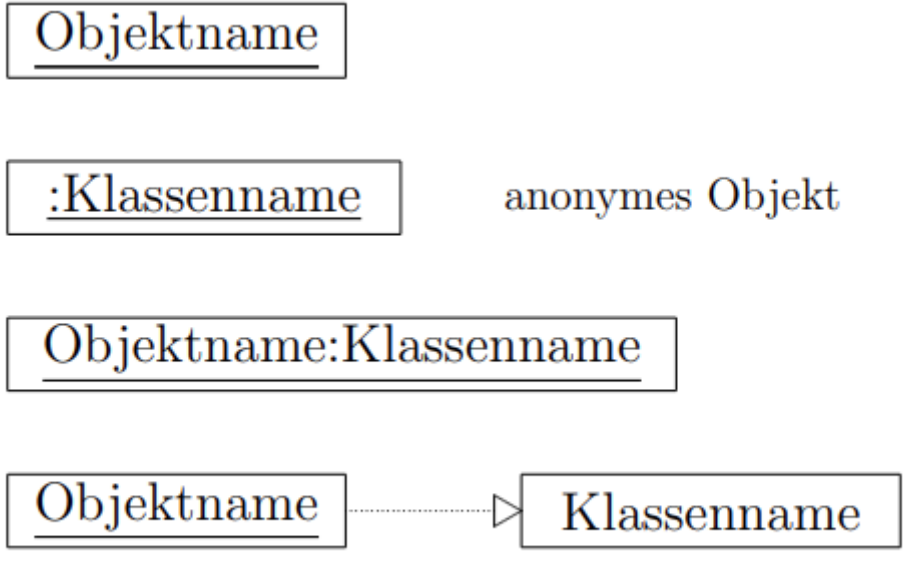
Klassendiagramm: Realisierung und Schnittstellen

- **Implementierung** (Implementation) wird hauptsächlich verwendet, um die Beziehung zwischen Schnittstellen und Implementierungsklassen zu spezifizieren
- Eine **Schnittstelle** (einschließlich einer abstrakten Klasse) ist eine Sammlung von Methoden. In einer Implementierungsbeziehung implementiert eine Klasse eine Schnittstelle, und Methoden in der Klasse implementieren alle Methoden der Schnittstellendeklaration



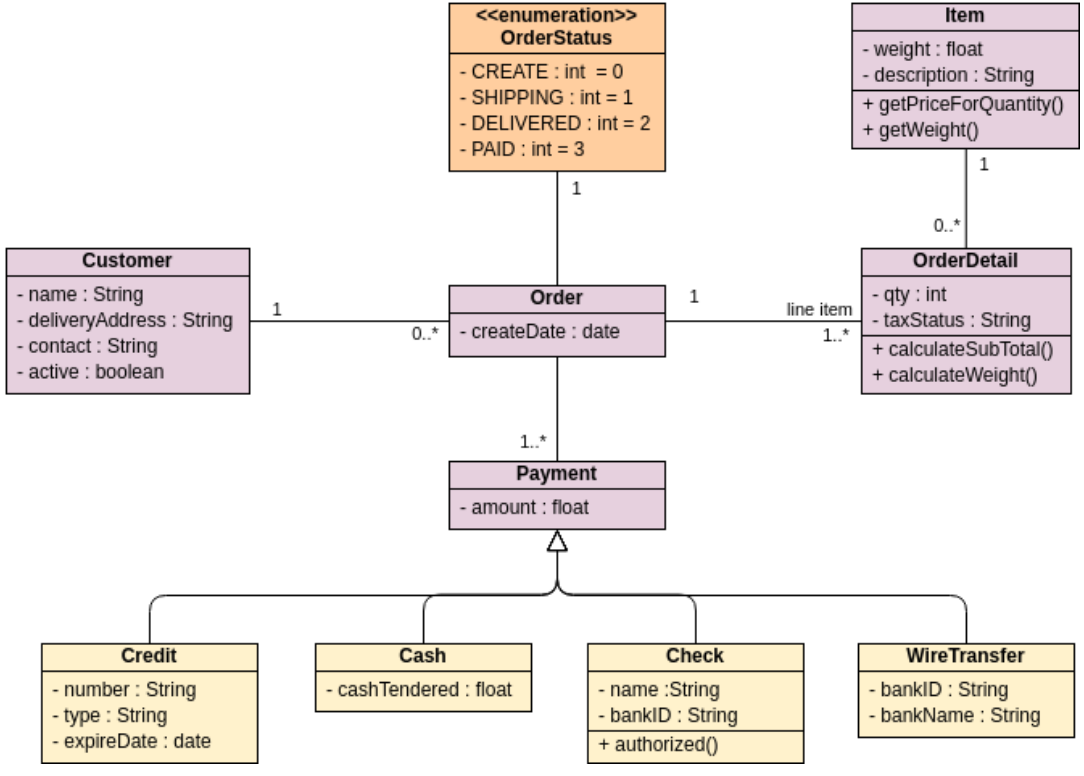


Klassendiagramm: Objekte



Instanziierung eines Objekts

Beispiel: Klassendiagramm



Übung: Teil 1 – Klassendiagramm

Erstellen Sie ein Klassendiagramm, das die wesentlichen Domänenklassen der Studierenden-TODO-App umfasst, darunter beispielsweise:

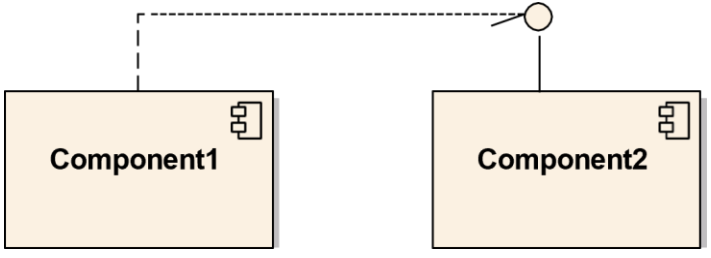
- *Student*
- *Course*
- *Task*
- *Assignment*
- *Exam*
- *Reminder*

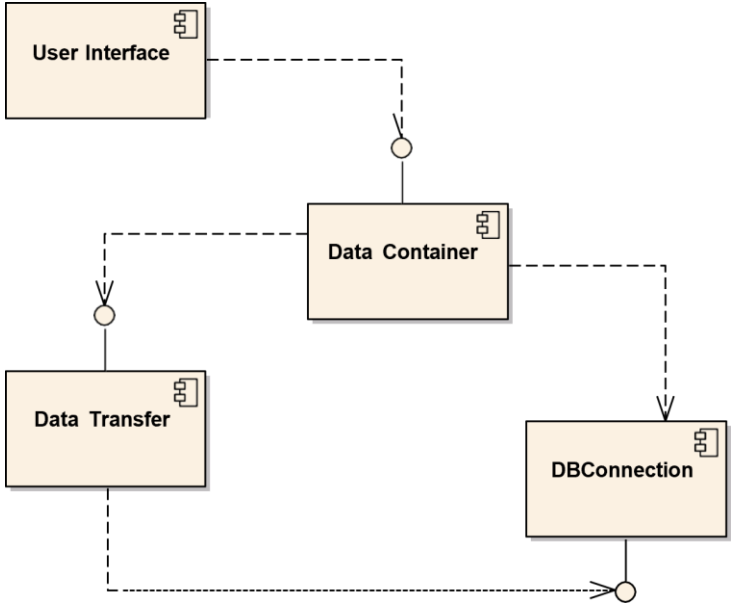
- **Bitte berücksichtigen Sie:**
  - relevante Attribute (und falls sinnvoll Methoden); Beziehungen zwischen den Klassen; korrekte Angabe der Multiplizitäten

Werkzeug: draw.io

Strukturdiagramme: Komponentendiagramm

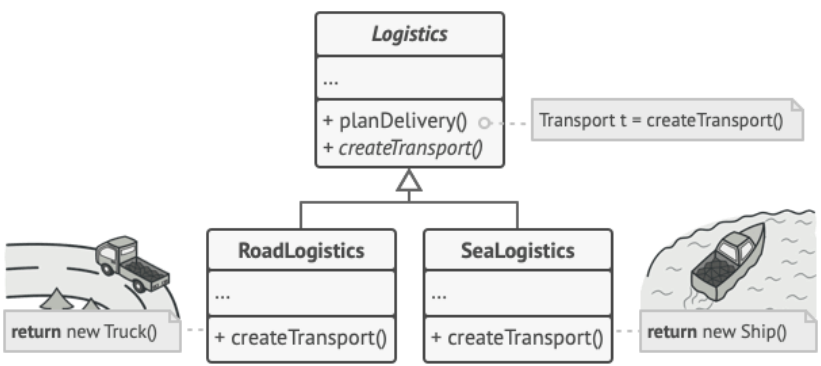
- Komponente und deren Zusammenspiel

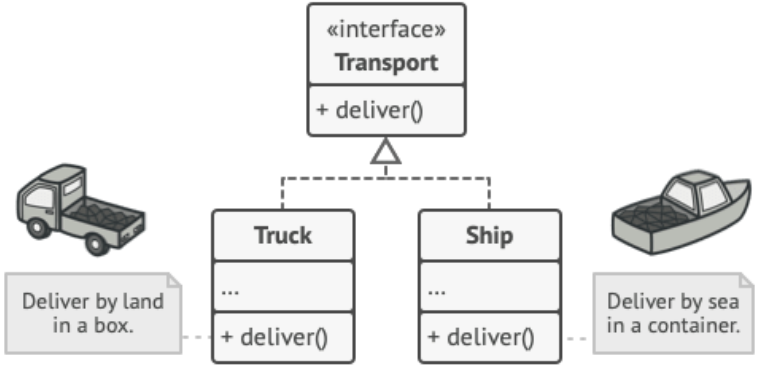




Logistikmanagement-Anwendung: Komponentendiagramm

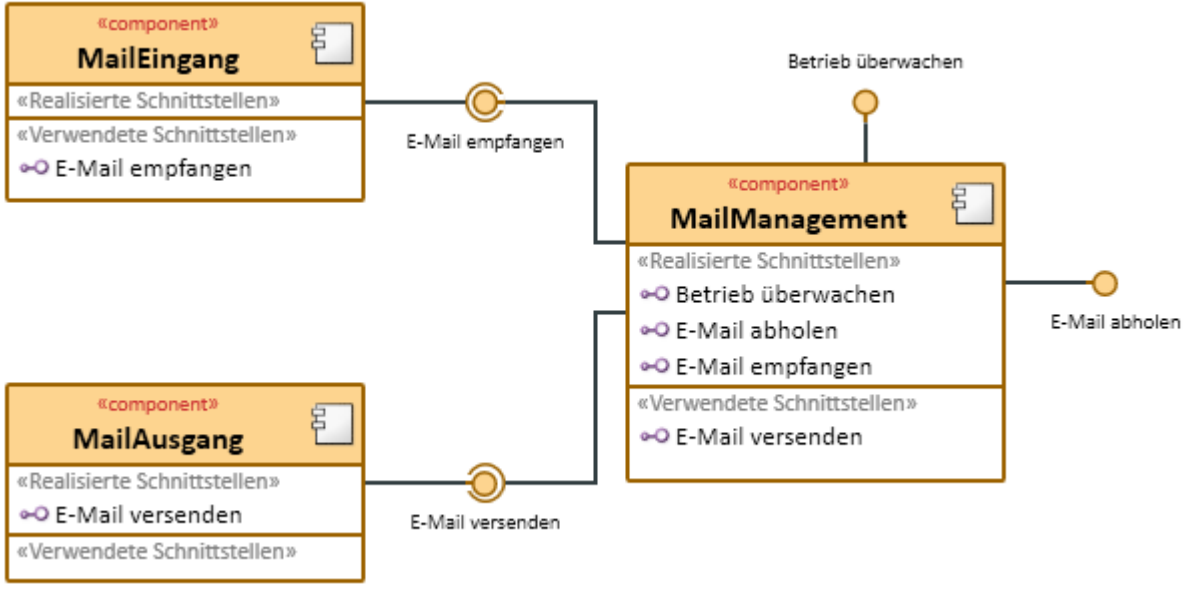
Logistikmanagement-Anwendung





Strukturdiagramme: Komponentendiagramm

- Komponente und deren Zusammenspiel



Übung Teil 2 – Komponentendiagramm für TODO App

Erstellen Sie anschließend ein Komponentendiagramm, das die wichtigsten Softwarekomponenten der App darstellt, z. B.:

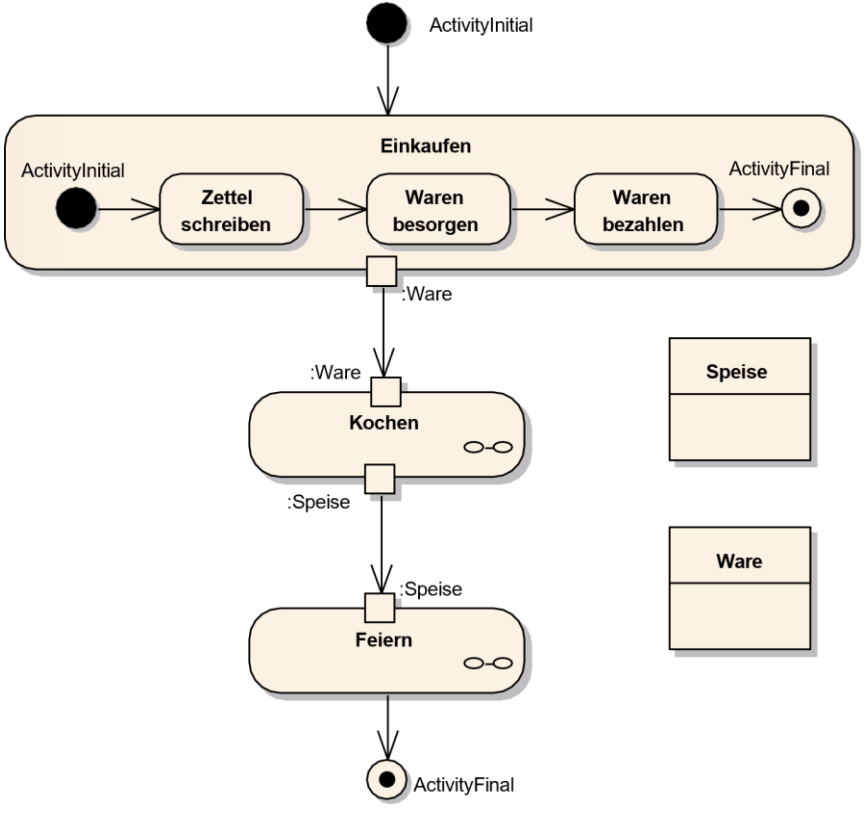
- Benutzeroberfläche (UI-Komponenten)
- Backend-Services (z. B. Reminder Service)
- Persistenzschicht (z.B. lokale Datenbank)

Achten Sie darauf, dass klar ersichtlich wird:

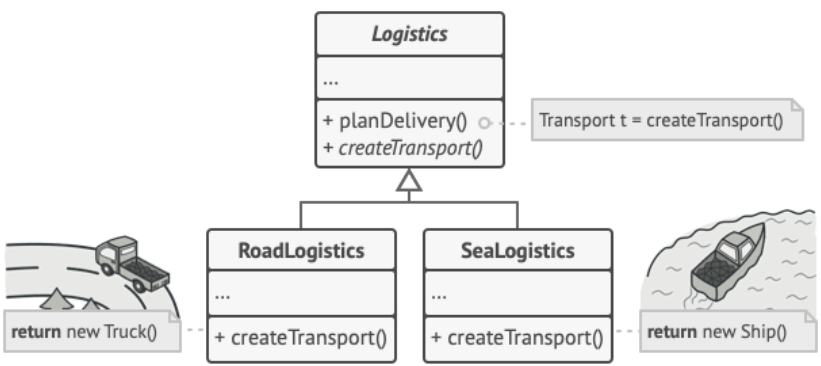
- welche Komponenten miteinander interagieren
- welche Abhängigkeiten bestehen
- welche Aufgaben bzw. Verantwortlichkeiten jede Komponente besitzt

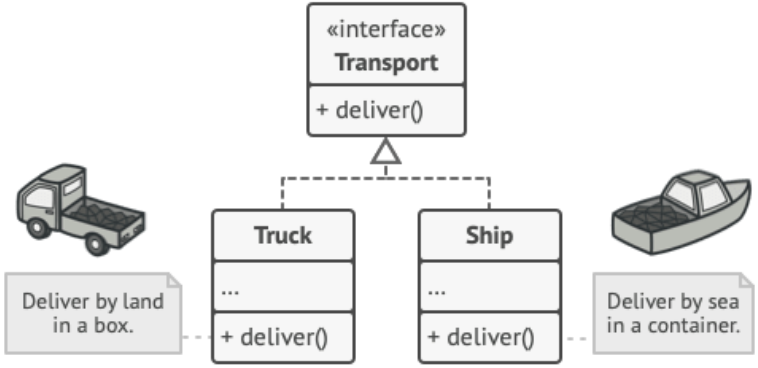
Verhaltensdiagramme: Aktivitätsdiagramm

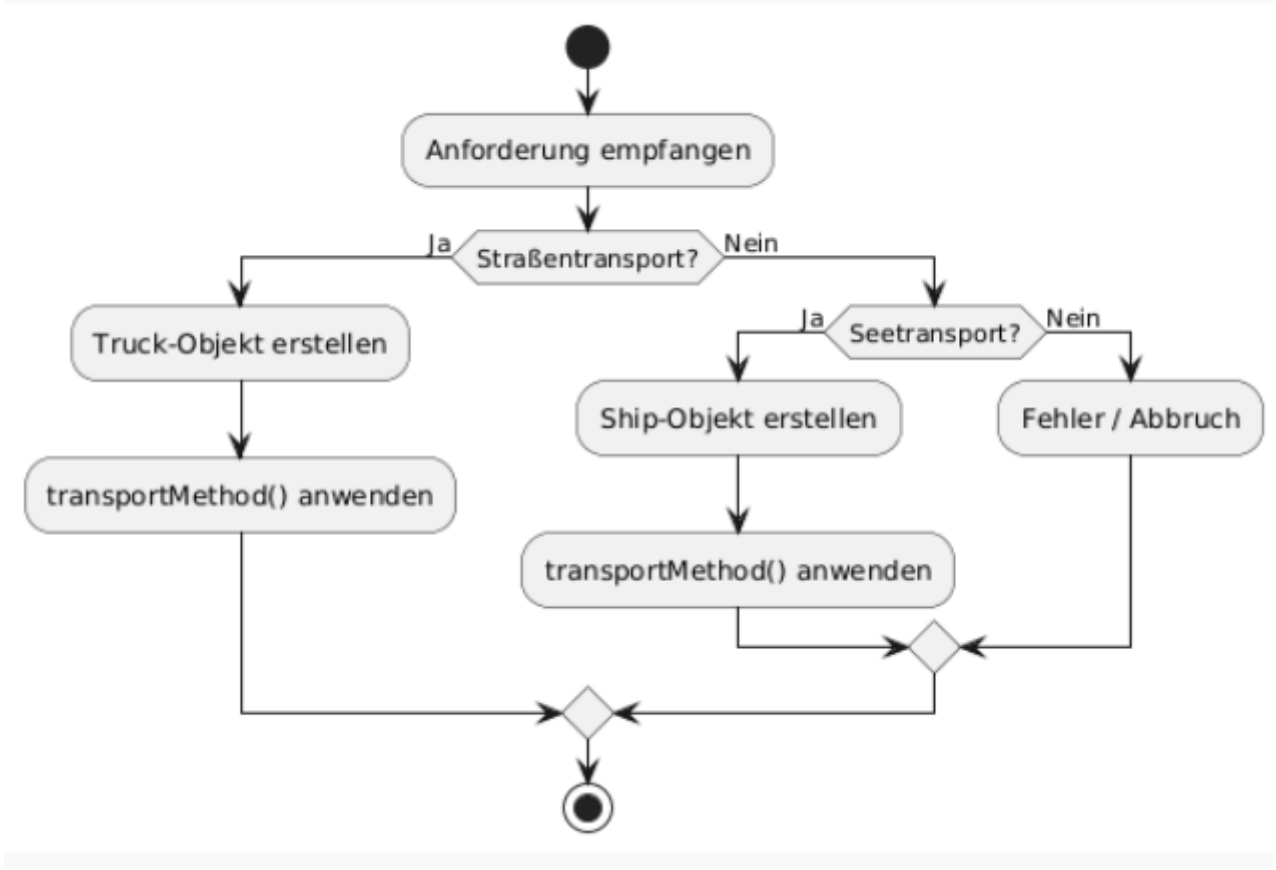
- dient zur Beschreibung von Prozessen, Workflows und Algorithmen
- *Beispiel: Prozess zur Durchführung einer Feier*



Aktivitätsdiagramm für Logistikmanagement

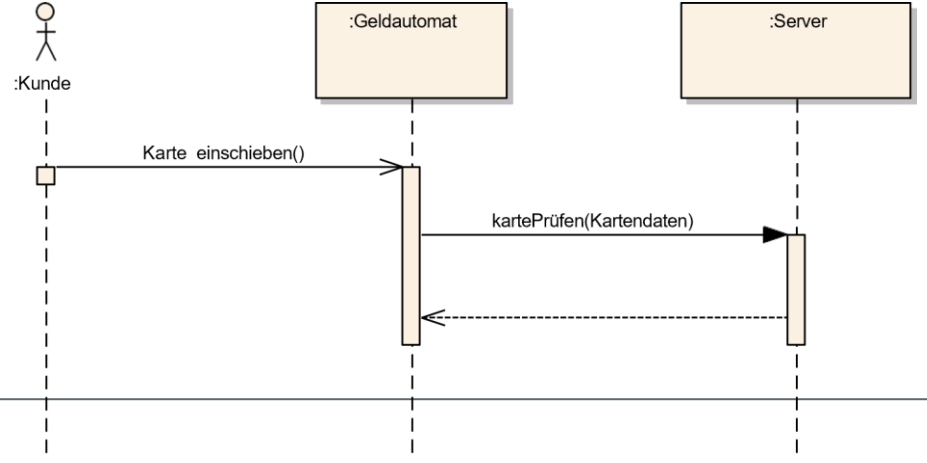





Anforderung empfangen
Straßentransport?
Ja
Nein
Truck-Objekt erstellen
transportMethod() anwenden
Seetransport?
Ja
Nein
Ship-Objekt erstellen
transportMethod() anwenden
Fehler / Abbruch

Interaktionsdiagramme: Sequenzdiagramm

- Zeigt den Ablauf von Nachrichten (Methodenaufrufen) zwischen Objekten in einer bestimmten Reihenfolge
- Die Reihenfolge der Nachrichten entspricht ihrer horizontalen Position im Diagramm. Es wird festgelegt, wann ein Objekt erstellt wird und wann Nachrichten zu welchem Objekt gesendet werden.



Übung – Aktivitätsdiagramm

Aufgabe: Erstellen Sie ein **Aktivitätsdiagramm**, das den Ablauf beschreibt, wie ein *Student* in der TODO-App eine neue Aufgabe (Task) anlegt.

Das Aktivitätsdiagramm soll mindestens folgende Schritte enthalten:

- Öffnen der Task-Ansicht
- Eingabe der Aufgabendetails (Titel, Beschreibung, Fälligkeitsdatum etc.)
- Optionales Hinzufügen eines Reminders
- Speichern der Aufgabe in einer lokalen Datenbank oder Datei
- Rückkehr zur Task-Übersicht

Die Kontrollflüsse, Entscheidungen (z. B. Reminder ja/nein) und Aktionen sollen klar dargestellt werden.

Übung: Sequenzdiagramm

Aufgabe: Erstellen Sie ein **Sequenzdiagramm**, das den Ablauf der Interaktion zwischen Benutzeroberfläche, Task-Service und Datenbank beschreibt, wenn ein Student eine neue Aufgabe erstellt.

Das Diagramm soll folgende Interaktionen enthalten:

- Benutzer gibt Daten in der UI ein
- UI sendet Anfrage an den Task-Service
- Task-Service validiert die Daten
- Task-Service speichert die Aufgabe in der Datenbank
- Bestätigung wird an die UI zurückgegeben
- UI zeigt Erfolgsmeldung an

Zentrale Vorteile von UML

- Einheitliche Sprache über Rollen hinweg
- Klarheit bei komplexen Systemen
- Bessere Kommunikation & Dokumentation

- Unterstützung durch Tools & Standards
  (z. B. Enterprise Architect, Visual Paradigm, Lucidchart, draw.io)

Typische Herausforderungen

- Übermodellierung und Komplexität
- Lernkurve und Konsistenz
- Synchronisation mit dem Code

Alternative Lösungen

- BPMN (Business Process Model and Notation)
- SysML (Systems Modeling Language)
- ArchiMate
- C4-Modell
- Domänenspezifische Sprachen (DSLs)

What‘s next?

Software-Architekturen