

[zurück](#)

### Aufgabe: Klassendiagramm entwerfen

Du sollst ein System für eine Bibliothek modellieren. Folgende Anforderungen sind gegeben:

- Eine Bibliothek verwaltet mehrere Bücher.
- Jedes Buch hat einen Titel, Autor, ISBN und Status (verfügbar/verliehen).
- Ein Benutzer kann Bücher ausleihen.
- Die Klasse Ausleihe dokumentiert, welches Buch wann von welchem Benutzer ausgeliehen wurde.

### Aufgabenstellung:

1. Entwirf ein UML-Klassendiagramm mit allen relevanten Klassen, Attributen und Beziehungen.
2. Kennzeichne die Sichtbarkeit der Attribute.
3. Welche Klasse sollte Methoden enthalten – z. B. ausleihen() oder zurückgeben()?

### Lösung:

```
classDiagram class Bibliothek { name buchHinzufuegen() benutzerRegistrieren() ausleihen()
zurueckgeben() findeBuch() } class Buch { titel autor isbn status istVerfuegbar() } class Benutzer {
benutzerId name anzahlAktiverAusleihen() } class Ausleihe { ausleihDatum faelligAm
rueckgabeDatum istUeberfaellig() dauerInTagen() } Bibliothek "1" o-- "0.."
Buch Bibliothek "1" o-- "0.."
Benutzer Bibliothek "1" o-- "0.."
Ausleihe Ausleihe "1" --> "1" Buch Ausleihe "1" --> "1" Benutzer
Benutzer "1" --> "0.."
Ausleihe
```

### Aufgabe : Beziehungstypen erkennen

Gegeben sind folgende Klassen:

- Rechnung
- Position
- Produkt

Beschreibung:

- Eine Rechnung besteht aus mehreren Positionen.
- Jede Position bezieht sich auf genau ein Produkt.
- Ein Produkt kann auf mehreren Rechnungen erscheinen.

### Aufgabenstellung:

1. Zeichne das Klassendiagramm mit den passenden Beziehungstypen.
2. Welche Beziehung besteht zwischen Rechnung und Position? Aggregation oder Komposition?
3. Wie würde sich die Modellierung ändern, wenn Position ohne Rechnung nicht existieren kann?

### Lösung

#### 1) Klassendiagramm (mit Beziehungstypen & Multiplizitäten)

**Empfohlen (fachlich korrekt: Komposition zwischen Rechnung und Position):** < mermaid>  
classDiagram class Rechnung class Position class Produkt

Rechnung „1“ - „1..“ Position Position „0..\*“ -

> "1" Produkt </mermaid>

Hinweis: Dadurch ergibt sich indirekt eine **n:m-Beziehung** zwischen *Rechnung* und *Produkt* (über *Position*).

## 2) Aggregation oder Komposition?

**Komposition.** Positionen sind Teil der Rechnung (Lebenszyklus gebunden); löscht man die Rechnung, verschwinden die Positionen.

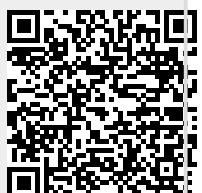
## 3) Wenn Position ohne Rechnung nicht existieren kann ...

... dann ist genau das die **Komposition** (gefüllter Diamant). Falls du zuvor Aggregation modelliert hattest, tausche einfach o-- gegen \*-- aus:

**Alternative (Aggregation, nur falls Positionen eigenständig existieren dürften):**

classDiagram class Rechnung class Position class Produkt Rechnung "1" o-- "1..\*" Position Position "0..\*"  
"0..\* --> "1" Produkt

From:  
<http://wiki.nctl.de/dokuwiki/> - □ **Veni. Vidi. sudo rm -rf / vici.**



Permanent link:  
<http://wiki.nctl.de/dokuwiki/doku.php?id=allgemein:test:uebungsaufgaben&rev=1759406828>

Last update: **02.10.2025 14:07**