**Wofür?**

* **Ziel:** Statische **Struktur** eines Systems zeigen: Klassen, Attribute, Methoden, **Beziehungen** (inkl. Kardinalitäten).
* **Nutzen:** Klärt Datenmodell & Verantwortlichkeiten; reduziert Missverständnisse (z. B. wer darf was sehen/ändern im Krankenhaus-System).

**Hauptbestandteile**

* **Klasse**: Name | Attribute | Methoden.
* **Attribute** (Eigenschaften), **Methoden** (Verhalten).
* **Sichtbarkeit**: + public, - private, # protected.
* **Beziehungen**: Vererbung, Assoziation, Aggregation, Komposition, Abhängigkeit, Realisierung.

A black arrow pointing to a black line

AI-generated content may be incorrect.**Assoziationen (Richtung)**

* **Unidirektional**: A „kennt“ B → **Linie mit Pfeil** zu B.
* A black text on a white background

  AI-generated content may be incorrect.**Bidirektional**: Beide kennen sich → **Linie ohne Pfeile**.
* **Tipp (IHK):** Immer **Rollenname** und **Multiplizitäten** angeben.

A diagram of a computer

AI-generated content may be incorrect.**Vererbung (Generalisierung)**

* **„Ist-ein“**: Unterklasse erbt von Oberklasse.
* **Notation:** Durchgezogene Linie mit **offener Dreiecksspitze** zur Oberklasse.

A screenshot of a computer

AI-generated content may be incorrect.**Multiplizitäten (Kardinalitäten)**

* **Werte:** 0..1, 1, 0..\*, 1..\*, m..n.
* **Beispiel:** Haus 1 — 0..\* Bewohner (ein Haus hat beliebig viele Bewohner; Bewohner lebt in **genau 1** Haus).

A computer screen shot

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.**Aggregation vs. Komposition**

* **Aggregation (◇)**: „hat-ein“, **Teil kann unabhängig** existieren (z. B. Team—Spieler).
* **Komposition (◆)**: **starke** Teil-Ganzes-Beziehung; Teil **ohne Ganzes nicht sinnvoll** (z. B. Haus—Zimmer).

**Abhängigkeit & Realisierung (der Vollständigkeit halber)**

* **Abhängigkeit (Dependency):** **gestrichelte Linie mit Pfeil** → temporäre Nutzung (z. B. Service ruft Druckfunktion).
* **Realisierung:** Klasse implementiert Interface → **gestrichelte Linie mit offener Dreiecksspitze** zum <<interface>>.

**Vorgehen: Klassendiagramm erstellen (Checkliste)**

1. **Domänenbegriffe sammeln** (aus Anforderungen/Use-Cases).
2. **Klassen identifizieren**, Schlüsselattribute (IDs) festlegen.
3. **Attribute & grundlegende Methoden** ergänzen (nur was fürs Design nötig ist).
4. **Beziehungen wählen** (Assoziation/Aggregation/Komposition/Vererbung) und **Multiplizitäten** setzen.
5. **Rollen benennen**, ggf. Navigationsrichtung festlegen.
6. **Sichtbarkeiten** und **Interfaces/abstrakte Klassen** dort, wo sinnvoll.
7. **Review**: passt das Modell zur Fachsprache? Überlappungen/Fehlstellen?

