**Ziel & Kontext**

* **Ziel der OO-Modellierung:** Klassen, Objekte & Beziehungen **klar strukturieren/visualisieren**, damit Teams ein gemeinsames Verständnis haben.
* **UML hilft dabei**, die **Struktur** (Klassen/Beziehungen) präzise darzustellen und sauber zu kommunizieren.

**OOP-Grundbegriffe (für Klassendiagramme)**

* **Klasse**: Schablone (Attribute/Methoden).
* **Objekt**: Instanz der Klasse.
* A screenshot of a computer

  AI-generated content may be incorrect.**Abstrakte Klasse**: unvollständig, kann nicht instanziiert werden.
* **Interface (Schnittstelle)**: reiner Vertrag (Signaturen), keine Implementierung.
* **Vererbung**: „ist-ein“ (Generalisierung).
* **Komposition/Aggregation**: „besteht-aus/hat-ein“.
* **Kapselung**: Zugriffsschutz auf interne Daten (Sichtbarkeiten).
* **Polymorphie**: einheitliche Nutzung verschiedener konkreter Typen über Oberklasse/Interface.
* **Dynamisches Binden**: Implementierung wird **zur Laufzeit** ausgewählt.

A diagram of a computer

AI-generated content may be incorrect.**Abstraktion, Interfaces & abstrakte Klassen**

**Abstraktion** = Wesentliches betonen, Details ausblenden.

**Notation in UML**

* *Abstrakte Klasse*: kursiv oder <<abstract>>.
* *Interface*: <<interface>>, gestrichelte Linie mit **offener Dreiecksspitze** (Realisierung) zur Klasse.

**Abstrakte Klasse vs. Interface (Merkpunkte)**

* A white rectangular sign with black text

  AI-generated content may be incorrect.**Abstrakte Klasse**: kann **Attribute & konkrete Methoden** haben; eine Klasse **erbt** (i. d. R.) nur **eine** Oberklasse; Basis für gemeinsame Implementierung.
* **Interface**: **nur Signaturen**, **keine Implementierung**; Klassen können **mehrere Interfaces** implementieren; ideal für Austauschbarkeit/Polymorphie.

A screenshot of a computer code

AI-generated content may be incorrect.**Kapselung & Sichtbarkeiten (UML-Legend)**

* **Zweck:** Schutz interner Zustände; Zugriff nur über methodische Schnittstellen.
* **Sichtbarkeitssymbole:** + **public**, # **protected**, - **private**.   
  *Praxis-Tipp:* Attribute meist - privat, Zugriff via Getter/Setter (oder Properties).

A diagram of a computer

AI-generated content may be incorrect.**Polymorphie & dynamisches Binden (kurz & prüfungsrelevant)**

* **Polymorphie:** Ein Referenztyp (z. B. Tier) kann zur Laufzeit unterschiedliche konkrete Objekte (Hund, Katze) halten; Aufruf läuft über die **gemeinsame Schnittstelle**.
* **Dynamisches Binden:** Welche geraeuschMachen() ausgeführt wird, entscheidet sich **zur Laufzeit** aufgrund des **tatsächlichen** Objekttyps.

**VAARAK – die 6 Hauptbeziehungen inkl. Notation, Beispiel & Merksatz**

**1) Vererbung (Generalisierung)**

* A black line on a white background

  AI-generated content may be incorrect.**Definition:** „ist-ein“; Unterklasse erbt von Oberklasse.
* **Beispiel:** Papagei → Vogel → Tier.
* **Merke:** Nutze Vererbung nur für echte **Is-a**-Beziehungen; vermeide „Missbrauch“ für Code-Sharing.

**2) Assoziation**

* A black line on a white background

  AI-generated content may be incorrect.**Definition:** Allgemeine Beziehung/Verknüpfung zwischen Klassen.
* **Beispiel:** Student — Kurs (z. B. Student 1..\* — \* Kurs).
* **Merke:** Immer **Kardinalitäten** angeben – das ist prüfungsrelevant!

**3) Abhängigkeit (Dependency)**

* A black lines on a white background

  AI-generated content may be incorrect.**Definition:** **Lockere, temporäre Nutzung**; eine Klasse **verwendet** eine andere (als Parameter, lokal, etc.).
* **Beispiel:** Textverarbeitung -----> Druckservice.
* **Merke:** Nutze für **„braucht für eine Operation“**, aber **keine dauerhafte Beziehung**.

**4) Realisierung**

* A black lines on a white background

  AI-generated content may be incorrect.**Definition:** Klasse **implementiert** ein Interface.
* **Beispiel:** Fernseher ----|> <<interface>> Geraet.
* **Merke:** **Interface = Vertrag**, Klasse liefert die konkrete Umsetzung.

**5) Aggregation (schwache Teil-Ganzes-Beziehung)**

* A black line on a white background

  AI-generated content may be incorrect.**Definition:** „hat-ein“; **Teil kann unabhängig** vom Ganzen existieren.
* **Beispiel:** Team ◇— Spieler (Spieler kann auch ohne dieses Team existieren).
* **Merke:** Organisatorische Zugehörigkeit, **keine Lebensdauer-Kopplung**.

**6) Komposition (starke Teil-Ganzes-Beziehung)**

* **Definition:** **Teil existiert nicht ohne** das Ganze; **gemeinsame Lebensdauer**.
* A black line on a white background

  AI-generated content may be incorrect.**Notation:** **Gefüllte Raute** am **Ganzen**.
* **Beispiel:** Haus ◆— Zimmer (Zimmer ohne Haus sinnlos).
* **Merke:** Nutze Komposition für **starke Besitzverhältnisse** (Containment).

**Kleine UML-Checkliste fürs Klassendiagramm (IHK-Praxis)**

1. **Domänenbegriffe sammeln** (aus Anforderungen/Use-Cases).
2. **Klassen + Schlüsselfelder** (IDs) definieren.
3. **Beziehungen** setzen **mit Kardinalitäten** und – wenn sinnvoll – Rollenbezeichnungen.
4. **Wähle Beziehungstyp**: Assoziation vs. Aggregation vs. **Komposition**; Vererbung nur bei echtem „ist-ein“.
5. **Sichtbarkeiten** angeben (+, -, #), **Interfaces** für Verträge.
6. **Einfach halten**: Nur modellieren, was fürs Verständnis/Design nötig ist.