Lernskript: Zugriffsmodifikatoren

Teil 1: Die grundlegende Analogie - Das "Warum"

Zugriffsmodifikatoren steuern die **Sichtbarkeit** von Klassen und ihren Mitgliedern. Sie sind das Kernwerkzeug der **Kapselung**, um Implementierungsdetails zu verstecken und nur eine kontrollierte Schnittstelle anzubieten. Die Analogie ist ein Auto.

- `public` **(Öffentlich):** Das Lenkrad und die Pedale. Für jeden sichtbar und benutzbar. Gilt projektweit.
- `private` **(Privat):** Die Verkabelung im Motorsteuergerät. Nur innerhalb der eigenen Klasse sichtbar. Der stärkste Schutz und der Standard für Felder.
- `default` **(Package-Private):** Spezialwerkzeuge für den Mechaniker. Sichtbar für alle Klassen im **selben Paket**.
- `protected` **(Geschützt):** Eine Diagnose-Schnittstelle. Sichtbar im selben Paket **UND für alle Subklassen**, egal in welchem Paket sie sind.

Sichtbarkeitshierarchie (von streng nach locker): `private` -> `default` -> `protected` -> `public`

Teil 2: Praktische Anwendungsfälle - Das "Wofür"

Use Case 1: Kapselung (`private`)

- **Problem:** Sicherstellen, dass Felder wie `kontostand` nicht unkontrolliert von außen verändert werden können.
- **Lösung:** Das Feld `private` machen und den Zugriff nur über `public` Methoden wie `einzahlen()` erlauben, die Regeln durchsetzen können.

Use Case 2: Paket-interne Helfer (`default`)

- **Problem:** Eine Helferklasse (`DatenbankConnector`) soll nur von Klassen im selben Paket verwendet werden.
- **Lösung:** Die Klasse **ohne** `public` deklarieren (`class DatenbankConnector`).

Use Case 3: Vererbung (`protected`)

- **Problem:** Ein Feld (`energie` in `Tier`) soll nur von Subklassen direkt bearbeitet werden dürfen.
- **Lösung:** Das Feld `protected` deklarieren.

Teil 3: Wichtige Regeln & Vertiefung (Profi-Tipps)

- **Regel der geringsten Rechte:** Beginne **IMMER** mit der restriktivsten Sichtbarkeit (`private`) und lockere sie nur bei Bedarf. Felder sollten fast immer `private` sein.

- `public` **vs.** `default` **bei Klassen:**

    - `public class MeineKlasse`: Eine **Hauptklasse**, die eine öffentliche Schnittstelle für das ganze Projekt darstellt.
    - `class MeineKlasse`: Eine **Hilfsklasse**, die ein Implementierungsdetail ihres Pakets ist.

- **`public` Klassen und Dateinamen:** Pro `.java`-Datei darf es **maximal eine `public` Klasse** geben. Der Name der `.java`-Datei muss exakt mit dem Namen dieser `public` Klasse übereinstimmen.

- **Interfaces:** Methoden in einem `interface` sind standardmäßig `public abstract`. Felder sind `public static final`.