Lernskript: Konstruktoren, `this` und `super`

Teil 1: Die grundlegende Analogie - Das "Warum"

- **Konstruktor:** Eine spezielle Methode, die bei `new` aufgerufen wird, um ein Objekt in einen **gültigen Anfangszustand** zu versetzen. Er ist der "Chef-Monteur" am Fließband, der sicherstellt, dass jedes neue Objekt die essenziellen Teile hat.
- `this` **(Dieses hier):** Eine Referenz auf das **aktuelle Objekt**, an dem gerade gearbeitet wird. Wird verwendet, um Instanzfelder von Parametern zu unterscheiden (`this.feld = parameter`).
- `super` **(Der von oben):** Eine Referenz auf die **Superklasse**. `super()` ruft einen Konstruktor der Superklasse auf, was in der Vererbung entscheidend ist.

Teil 2: Praktische Anwendungsfälle - Das "Wofür"

Use Case 1: Der Standard-Konstruktor und Pflichtfelder

- **Problem:** Sicherstellen, dass ein Objekt immer mit den notwendigen Daten erstellt wird.
- **Lösung:** Einen eigenen Konstruktor definieren, der diese Daten als Parameter erzwingt.
- **Die wichtigste Konstruktor-Regel:** Sobald du auch nur **einen eigenen Konstruktor** definierst, **löscht Java den unsichtbaren, parameterlosen Standard-Konstruktor**. Ein Aufruf mit `new MeineKlasse()` führt dann zu einem Compiler-Fehler, wenn nur ein Konstruktor wie `public MeineKlasse(String name)` existiert.

Use Case 2: Konstruktor-Überladung und `this()`

- **Problem:** Mehrere Wege zur Objekterstellung anbieten, ohne Code zu duplizieren.
- **Lösung:** Mehrere Konstruktoren definieren (**Überladung**) und mit `this()` einen Konstruktor aus einem anderen heraus aufrufen, um die Logik zu zentralisieren.
- **Regel:** `this()` muss immer die **allererste Anweisung** im aufrufenden Konstruktor sein.

Use Case 3: Konstruktoren in der Vererbung und `super()`

- **Problem:** Sicherstellen, dass der Superklassen-Teil eines Objekts korrekt initialisiert wird, bevor die Subklasse ihre eigenen Felder setzt.
- **Lösung:** Der Subklassen-Konstruktor ruft mit `super()` explizit einen Konstruktor der Superklasse auf.
- **Regel:** `super()` muss immer die **allererste Anweisung** im Subklassen-Konstruktor sein.

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

- **Unsichtbares `super()`:** Wenn in einem Konstruktor weder `this()` noch `super()` explizit aufgerufen wird, fügt der Compiler **automatisch** ein `super();` (Aufruf des parameterlosen Super-Konstruktors) als erste Zeile ein. Hat die Superklasse keinen solchen parameterlosen Konstruktor, führt dies zu einem Compiler-Fehler.

- `this` **vs.** `this()`**:** `this` ist eine Referenz auf das Objekt. `this()` ist ein Aufruf an einen anderen Konstruktor derselben Klasse.
- `super` **vs.** `super()`**:** `super` ist eine Referenz auf das Superklassen-Objekt (z.B. für `super.methode()`). `super()` ist ein Aufruf an einen Superklassen-Konstruktor.
- **Ausführungsreihenfolge bei** `this()`**:** Bei einem Aufruf wie `new Email("Betreff")` wird zuerst der **aufgerufene** Konstruktor (`Email(String, String)`) vollständig ausgeführt, und erst danach der Rest des **aufrufenden** Konstruktors (`Email(String)`).