Kapitel 7 – Rekursion: Zusammenfassung + Übungen

In diesem Kapitel lernst du das Konzept der Rekursion kennen. Eine Methode ruft sich dabei selbst auf, um ein Problem schrittweise zu lösen. Rekursion ist ein wichtiges Werkzeug – besonders bei Aufgaben wie Zahlenreihen, Bäumen oder Zerlegungen.

Themenübersicht

1. Was ist Rekursion?

Eine Methode ruft sich selbst auf, mit veränderten Parametern – bis eine Abbruchbedingung erfüllt ist.

2. Basisfall und Rekursionsfall

Jede Rekursion braucht:
- Einen Basisfall, der die Rekursion stoppt
- Einen Rekursionsfall, der weiterruft

3. Beispiel: Fakultät

```
int fak(int n) {
  if (n == 0) return 1;
  return n * fak(n - 1);
}
```

4. Call-Stack

Java merkt sich die Rücksprungstellen jeder Methode im sogenannten Callstack.

5. Lineare vs. Kaskadierende Rekursion

Linear: ein Aufruf pro Schritt (z. B. fakultät)
Kaskadierend: mehrere Aufrufe pro Schritt (z. B. Fibonacci)

6. Gefahren der Rekursion

Kein Basisfall → Endlosschleife bzw. Stack Overflow.
Jeder Aufruf braucht Speicher → ineffizient bei großen Problemen.

Übungsaufgaben

1. 1. Schreibe eine Methode `fakultaet(int n)`, die n! rekursiv berechnet.
2. 2. Implementiere eine Methode `int summe(int n)`, die 1 + 2 + ... + n rekursiv addiert.

3. 3. Schreibe eine Methode `int fibonacci(int n)`, die die n-te Fibonacci-Zahl rekursiv berechnet.
4. 4. Erkläre den Unterschied zwischen dem Basisfall und dem Rekursionsfall.
5. 5. Baue absichtlich eine rekursive Methode ohne Basisfall. Was passiert beim Aufruf?
6. 6. Erstelle eine rekursive Methode zur Berechnung der Potenz: `power(x, n)` = x^n.
7. 7. Baue eine rekursive Methode, die die Ziffern einer Zahl rückwärts ausgibt (z. B. 123 → 321).
8. 8. Warum ist die rekursive Fibonacci-Methode ineffizient? Was wäre besser?