# Methodenkonzept

Erste Motivation:

Große Quelltexte sollen untergliedert werden,

um die Übersichtlichkeit zu fördern.

class … {

… main (…){

<Anw\_1>

<Anw\_2>

…

<Anw\_n-1>

<Anw\_n>

}

}

Wir wollen einen Teil der Anweisungen aus der Hauptmethode "main()" auslagern.

Diesen Block verschieben wir hinter die Hauptmethode

auf gleicher Ebene der Hauptmethode.

Wir geben dem ausgelagerten Teil einen Namen,

unter dem er aufgerufen werden kann.

Vergeben wir hier den Namen "meine\_Methode()".

Methoden u. deren Aufrufe werden immer von einem (vorerst) leeren runden Klammerpaar gefolgt,

um zwischen einer Methode u. einer Variablen zu unterscheiden:

class … {

… main (…){

<Anw\_1>

<Anw\_2> meine\_Methode();

…

<Anw\_n-1>

<Anw\_n>

}

… meine\_Methode(){

}

}

Früher hat man es "Unterprogramm" genannt.

In anderen Sprachen heißt es auch "Prozedur" (Pascal)

oder "Funktion" (C).

Der Ablauf ist vergleichbar mit einem Link in einem HTML-Dokument:

Man liest im Hauptdokument.

Dort findet man einen interessanten Verweis zu einem Zweitdokument.

Dort springt man hin u. liest den Artikel.

Ist man damit fertig,

springt man in das Hauptdokument zurück an die Stelle,

die dem Verweis folgt u. liest dort weiter.

Hier:

Findet der Compiler im Hauptprogramm den Aufruf einer Methode,

zu erkennen an dem runden Klammernpaar,

sucht er hinter dem Hauptprogramm nach der Methode mit diesem Namen.

Dort wird Programmausführung fortgesetzt. ❶

Ist die Methode abgearbeitet,

springt der Compiler an die Anweisung im Hauptprogramm zurück,

die dem Methodenaufruf folgt. ❷

class … {

… main (…){

meine\_Methode();

… ❶

<Anw\_n-1>

<Anw\_n>

}

… meine\_Methode(){

<Anw\_1>

<Anw\_2>

} ❷

}

Die Hauptmethode "main()",

die wir die ganze Zeit schon benutzt haben,

stellt den Startpunkt für den Compiler da.

Sie muss in jeder Klasse vorhanden sein,

die ausgeführt werden können soll.

Eine Methode besteht aus zwei Komponenten:

1. Deklaration der Methode,

wo ein Stück Quelltext mit einem Namen assoziiert wird,  
das bei Aufruf der Methode ausgeführt werden soll,  
zu erkennen an dem Block hinter dem Methodennamen.

1. Aufruf der Methode (ohne Block)

Beide führen nach dem Methodennamen ein (vorerst) leeres rundes Klammerpaar.

Bei der Hauptmethode stehen die Worte "public static void"   
vor dem Methodennamen.

Für selbst definierte Methoden übernehmen wir diese  
(erst einmal) unverändert:

public static void meine\_Methode (){

…

}

Konvention der Namensgebung:

Methodennamen werden

- wie auch Variablennamen - klein geschrieben,

im Gegensatz zu Klassennamen.

Unterscheidung des Konzeptes hier in Java im Gegensatz von z.B. Pascal:

In Pascal werden Methoden - dort heißen sie Funktionen oder Prozeduren -

nicht hinter das Hauptprogramm auf gleicher Ebene des Hauptprogramms ausgelagert,

sondern vor die Aufrufstelle und eine Ebene untergeordnet.

Beide Varianten haben Vor- u. Nachteile.

Methoden können geschachtelt werden,

um eine hierarchische Gliederung zu gewährleisten:

So wie das literarische Lebenswerk von 10 000 Seiten

auf erster Ebene untergliedert wird in 10 Bände á 1000 Seiten,

auf zweiter Ebenen jeder Band in 10 Kapitel von je 100 Seiten

auf dritter Ebene jedes Kapitel in 10 Unterkapitel von je 10 Seiten.

Gesamtwerk: 10 000 Seiten

Band 1: 1000 Seiten Band 2: 1000 Seiten … Band 10: 1000 Seiten

... Kapitel 1: 100 Seiten Kapitel 2: 100 Seiten … Kapitel 10: 100 Seiten …

… Unterkap. 1: 10 Seiten Unterkap. 2: 10 Seiten… Unterkap. 10: 10 Seiten …

main()

…

m1() m2() …

…

m1a() m1b() …

…

m1a\_i() m1a\_ii() …

**public** **class** Schachtel {

**public** **static** **void** main(String[] args) {

System.***out***.println("Hier beginnt die Ausfuehrung des Hauptprogramms.");

*m1*();

*m2*();

System.***out***.println("Hier ist das Hauptprogramm zu Ende.");

}

**public** **static** **void** m1() {

System.***out***.println("\tHier ist Methode \"m1\".");

*m1a*();

*m1b*();

}

**public** **static** **void** m1a() {

System.***out***.println("\t\tHier ist Unter-Methode \"m1a\".");

*m1a\_i*();

*m1a\_ii*();

}

**public** **static** **void** m1a\_i() {

System.***out***.println("\t\t\tHier ist Unter-Unter-Methode \"m1a\_i\".");

}

**public** **static** **void** m1a\_ii() {

System.***out***.println("\t\t\tHier ist Unter-Unter-Methode \"m1a\_ii\".");

}

**public** **static** **void** m1b() {

System.***out***.println("\t\tHier ist Unter-Methode \"m1b\".");

}

**public** **static** **void** m2() {

System.***out***.println("\tHier ist Methode \"m2\".");

}

}

Hier beginnt die Ausfuehrung des Hauptprogramms.

Hier ist Methode "m1".

Hier ist Unter-Methode "m1a".

Hier ist Unter-Unter-Methode "m1a\_i".

Hier ist Unter-Unter-Methode "m1a\_ii".

Hier ist Unter-Methode "m1b".

Hier ist Methode "m2".

Hier ist das Hauptprogramm zu Ende.