# Polymorphismus

(am Beispiel mehrerer max()-Methoden)

Wir haben mehrere Methoden implementiert, die wir

* max\_2\_int()
* max\_3\_int()
* max\_4\_int()
* max\_2\_double()
* max\_3\_double()
* max\_4\_double()

genannt haben.

Sie nehmen 2, 3 oder 4 Ganz- oder Fließkommazahlen entgegen.

Beim Aufruf müssen wir aufpassen,

die richtige Methode zu der Anzahl u. dem Typ übergebener Parameter herauszusuchen.

Vertun wir uns dabei, gibt es eine Fehlermeldung,

wenn wir z.B. "max\_2\_int()" mit 3 int-Parametern aufrufen.

Auch macht es uns Arbeit,

die Methoden unterschiedlich zu benennen,

je nach Anzahl u. Typ.

Überlegung:

Bei einem Methodenaufruf sucht der Compiler unter den vorhandenen Methoden,

wo sich die mit dem gesuchten Namen befindet.

Wunsch:

Wäre es nicht denkbar,

dass wir alle Methoden gleich benennen,

u. sich der Compiler zur Laufzeit die richtige heraussucht,

entsprechend der aktuellen Parameterliste?

Er soll also nicht nur nach 1. dem Namen suchen,

sondern, sofern mehrere mit gleichem Namen vorhanden sind,

auch 2. nach der passenden Parameterliste.

Würde das Funktionieren,

hätte das die beiden Vorteile:

1. Wir müssten keinen Aufwand betreiben,  
   die Methoden unterschiedlich zu benennen  
   u. den Typ u. die Anzahl der Parameter im Namen codieren.
2. Es bestünde keine Verwechslungsgefahr  
   zwischen dem Methodennamen u. den übergebenen Parametern.

Aussage des Polymorphismus:

Es darf mehrere Methoden gleichen Namens parallel geben,

sofern sie sich in der Parameterliste unterscheiden!

Der Compiler sucht zur Laufzeit die richtige heraus,

so dass nicht nur der Name,

sondern auch die Parameterliste passt.

Gibt es mehrere Methoden gleichen Namens,

nennt man die Methode auch "überladen".

Man sagt: Eine Methode ist überladen.

Unterscheidungskriterien für Parameterlisten:

1. Länge der Liste
2. Typ
3. Reihenfolge

Der Name des Parameters ist kein gültiges Unterscheidungskriterium.

Beispiel:

public class Polymorphismus {

public static void main(String[] args) {

m();

m(17);

m(true);

m(23, false);

m(true, 42);

}

public static void m() {

System.out.println("Hier m ohne Parameter.");

}

public static void m(int i) {

System.out.println("Hier m mit einem int-Parameter.");

}

public static void m(boolean b) {

System.out.println("Hier m mit einem bool-Parameter.");

}

public static void m(int i, boolean b) {

System.out.println

("Hier m mit erst einem int- dann einem bool-Parameter.");

}

public static void m(boolean b, int i) {

System.out.println

("Hier m mit erst einem bool- dann einem int-Parameter.");

}

// public static void m(int anderer\_Name) {

// System.out.println("Hier m mit einem int-Parameter.");

// }

}