# Parameter bei Methoden

Wir suchen eine Möglichkeit,

Informationen an eine Methode bei Aufruf übergeben zu können,

um nicht nur identische, sondern auch ähnliche Aufgabe ausführen zu können.

Im Beispiel:

Es sollen nicht nur genau 15 Sterne geschrieben werden können,

sondern eine beliebige Anzahl.

Im Vergleich mit dem "Black Box-Prinzip" suchen wir die Möglichkeit einer Eingabe.

Dazu betrachten wir uns die Deklaration einer Methode:

public static void <Methodenname> ( ) {

<Block>

}

Eingabemöglichkeit: Parameterliste

Bisher haben wir argumentiert,

die (bis dahin) leeren runden Klammern dienen zur Unterscheidung,

ob es sich um eine Variable oder eine Methode handelt.

Hier bekommt das runde Klammerpaar eine weitere Bedeutung,

nämlich als Schnittstelle zur Übergabe von Informationen beim Aufrufen.

Hier würde,

um uns auf das Beispiel von oben zu beziehen,

die zu schreibende Anzahl von Sternen übergeben werden.

Die Methode bisher, die genau 15 Sterne geschrieben hat:

public static void schreibe\_Sterne ( ) {

for (int i=1; i<=15; i++)

System.out.print ("\*");

System.out.println ();

}

Methode mit einem int-Parameter "anzahl":

public static void schreibe\_Sterne (**int** **anzahl**) {

**anzahl**

for (int i=1; i<=15; i++)

System.out.print ("\*");

System.out.println ();

}

Nun kann ich beim Aufrufen festlegen,

wie viele Sterne geschreiben werden sollen:

schreibe\_Sterne(**10**);

würden 10 Sterne geschrieben werden.

Ein paar Vokabeln:

In der Deklaration nennt man die Auflistung der zu übergebenden Werte

"formale Parameterlist",

die beim Aufruf übergebenen Werte, z.B. Zahlen, nennt man

"aktuelle Parameterliste".

**formale** Parameterliste

public static void schreibe\_Sterne (**int** **anzahl**) {

for (int i=1; i<=**anzahl**; i++)

System.out.print ("\*");

System.out.println ();

}

**aktuelle** Parameterliste

schreibe\_Sterne(**10**);

Der Parameter ist wie eine lokale Variable.

Er "lebt" nur während der Laufzeit der Methode.

Bei Verlassen wird er wieder frei gegeben.

Hier habe ich lesend auf den Parameter zugegriffen.

Kann ich auch schreibend zugreifen?

Ja, aber es ist (zumindest in dem Kontext hier) schlechter Stil:

So wie wir bei einer for-Schleife auch schreibend auf die Zählvariable zugreifen können,

es aber aus stilistischen Erwägungen heraus besser unterlassen sollten,

so können wir hier auch schreibend auf den Parameter zugreifen,

was aber stilistisch unschön ist.

Wir sollten die übergebene Information "behalten",

z.B. für den Fall, dass wir erneut zugreifen wollen.

Welche Vorteile kann es bringen,

schreibend auf den Parameter zuzugreifen:

Ursprüngliche Variante von "schreibe\_Sterne()":

public static void schreibe\_Sterne (**int** **anzahl**) {

Hilfsvariable i zum Zählen

for (**int i**=1; i<=**anzahl**; i++)

System.out.print ("\*");

System.out.println ();

}

Wenn ich den Parameter benutze u. schreibend zugreife,

kann ich auf die Hilfsvariable verzichten

u. spare damit Platz für die Variable u. Zeit für die Deklartion u. Freigabe:

public static void schreibe\_Sterne (**int** **anzahl**) {

Keine Hilfsvariable schreibender Zugriff auf den Parameter

for ( ; **anzahl**>=1; **anzahl**--)

System.out.print ("\*");

System.out.println ();

}

Vergleich der beiden Varianten:

Variante 2 mit schreibendem Zugriff auf den Parameter ist

* platzsparender, weil auf die Zählvariable "i" verzichtet wird
* zeitsparender, weil Zählvariable nicht in Schleife angelegt-  
  und bei Verlassen wieder freigegeben werden muss.
* Quelltext (minimal) kürzer
* stilistisch weniger schön wegen schreibendem Zugriff auf Parameter.

Dies bezieht sich nur auf den Kontext hier,

wenn wir einfache Datentypen übergeben,

wie hier einen int.

Bei zusammengesetzten Datentypen,

wie wir sie als Felder u. Objekte kennen lernen werden,

sieht es anders aus.

(Stichworte: "call by value" u. "call by reference", siehe später)

Durch das Parameterkonzept haben wir das Methodenkonzept verallgemeintert:

Wir können dadurch Methoden nicht nur mehrfach identische Aufgaben ausführen lassen,

wie hier das Schreiben von genau 15 Sternen,

sondern auch ähnliche,

wie hier das Schreiben einer beliebigen Anzahl von Sternen.