Java - Referenz

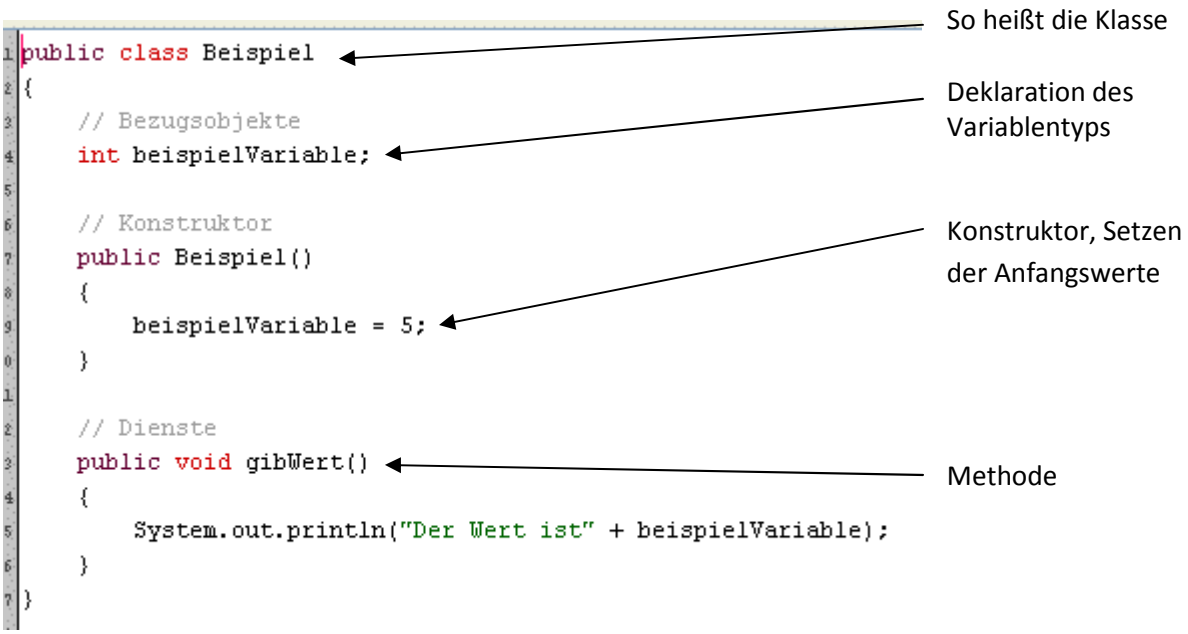
Programmaufbau

Jede **Klasse** beginnt mit der Definition, ob sie öffentlich einsehbar (public) sein soll, dann folgt der Hinweis auf die Klasse (class) und dann der Name (hier: Beispiel). Dann folgt die geschweifte Klammer, die erst am Ende der gesamten Klasse geschlossen wird.

Der nächste Schritt ist die Deklaration, mit welchen Variablen gearbeitet wird und von welchem Typ sie sind. Im Beispiel wird mit einer Integerzahl gearbeitet, die beispielVariable genannt wird.

Der **Konstruktor** heißt wie die Klasse, gefolgt von runden Klammern. Die Zuweisung von Startwerten steht in geschweiften Klammern.

Es gibt unterschiedliche **Methoden**. Auch hier wird zunächst festgelegt, ob diese einsehbar (public) sein sollen. Die Angabe void weist darauf hin, dass kein Wert zurückgegeben wird. Es folgt der Variablenname mit runden Klammern.




```
public class Beispiel
{
    // Bezugsobjekte
    int beispielVariable;

    // Konstruktor
    public Beispiel()
    {
        beispielVariable = 5;
    }

    // Dienste
    public void gibWert()
    {
        System.out.println("Der Wert ist" + beispielVariable);
    }
}
```

Beliebte Fehler:

- Vergessenes Semikolon nach Befehlen oder Methodenaufrufen
- Gesetztes Semikolon in der ersten Zeile der Methode (Deklaration)
- Vergessene geschweifte Klammer am Ende der Methode oder Klasse

Datentypen

Typ	Mögliche Werte und Größe
boolean	Nur zwei Möglichkeiten: true oder false (1 Bit)
char	Unicode-Zeichen z.B. A, a, Σ, ϕ, ä, ß… (16 Bit)
byte	Ganzzahl von -128 bis 127 (8 Bit)
short	Ganzzahl von -32768 bis 32767 (16 Bit)
int	Ganzzahl von -2147483648 bis 2147483647 (32 Bit)
float	Gleitkommazahl (32 Bit)
double	Gleitkommazahl doppelter Genauigkeit (64 Bit)

Methoden

Es gibt Methoden, die einen Wert verändern (verändernde Methoden) oder einen Wert zurückliefern (sondierende Methoden).

Die **verändernde Methode** ist nach folgendem Schema aufgebaut:

```
public void setzeWert(int neuerWert)
```

public definiert die Methode als öffentlich einsehbar. Spielt erst später eine Rolle
void macht deutlich, dass kein Wert zurückerwartet wird
setzeWert ist der Methodenname. Auf ihn müssen runde Klammern folgen. Sind die Klammern leer, wird nur ein Wert berechnet oder gesetzt. Soll der Nutzer einen Wert eingeben können, folgt zunächst der Typ der Variable (im Beispiel int) und dann der Variablenname, der den vom Nutzer eingegebenen Wert speichert (hier neuerWert).

Die **sondierende Methode** ist nach dem folgenden Schema aufgebaut:

```
public int gibWert()
{
    return beispielVariable;
}
```

public definiert die Methode als öffentlich einsehbar. Spielt erst später eine Rolle
int gibt den Datentyp des Wertes an, der zurückgegeben werden soll
gibWert() ist der Methodenname
return gibt den Wert der folgenden Variable (hier beispielVariable) zurück; die Variable muss vom im Methodenkopf definierten Typ sein.

Auch die sondierende Methode erlaubt Nutzerangaben nach dem Beispiel der ersten Methode.

Die Rückgabe eines Wertes ist über die Konsole möglich, dann wird der Wert aber nur angezeigt und dient nicht der Weiterverarbeitung. Soll mit dem ermittelten Wert weitergearbeitet werden, bedient man sich der sondierenden Methode.

Der Name einer Methode sollte ihre Funktion deutlich machen.

Kontrollstrukturen

Bedingte Anweisung

if (Bedingung)
{
  Anweisung(en);
}
	Wenn die Bedingung gilt, werden die Anweisungen in den geschweiften Klammern ausgeführt. „Bedingung“ kann eine boolesche Variable if (gutesWetter)… oder eine Berechnung sein if (x < 3)…, deren Ergebnis true oder false liefert

if (Bedingung)
{
  Anweisung(en);
}
else
{
  Anweisung(en);
}
	Wenn die Bedingung gilt, werden die Anweisungen in den geschweiften Klammern ausgeführt. Wenn sie nicht gilt, werden die Anweisungen nach else ausgeführt

Schleifen

while (Bedingung)
{
  Anweisung(en);
}
	Solange die Bedingung gilt, werden die Anweisungen in den geschweiften Klammern ausgeführt. Gilt die Bedingung nicht, wird die Anweisung nicht ausgeführt.

do { Anweisung(en); } while (Bedingung);	Die Anweisungen in den geschweiften Klammern werden ausgeführt, solange die Bedingung gilt. Unterschied zu while: Die Schleife wird mindestens einmal durchlaufen, auch wenn die Bedingung schon beim ersten Mal nicht gilt. Aus diesem Grund wird eher die while-Schleife verwendet.

for
(Startwert der Zählvariable;
Abbruchbedingung;
Zählschritt der Variable)
{
  Anweisung(en);
}
	Die FOR-Schleife ist eine Zählschleife. Sie zählt
	  vom ersten in den Klammern stehenden Wert
	  bis die als zweite stehende Bedingung nicht mehr gilt
	  in den als letztes angegebenen Zählschritten, beispielsweise
	for (i=0; i<10; i=i+1)
	{
	  System.out.println(„Ich schreibe mal was…“);
	}

Vergleichsoperatoren, häufig in Verbindung mit der if-Abfrage benutzt

<	Kleiner als	>	Größer als
<=	Kleiner oder gleich	>=	Größer oder gleich
==	Ist gleich	!=	Nicht gleich
&&	Logisches UND, alle Bedingungen müssen erfüllt sein	||	Logisches ODER, mindestens eine Bedingung muss erfüllt sein

Rechenoperatoren

Addition:	+	Multiplikation:	*
Subtraktion:	-	Division:	/