# Präzedenz von Operatoren

Die Präzedenz von Operatoren bestimmt die **Struktur von Ausdrücken**. Ein Operator höherer Präzedenz bindet die Operanden stärker als ein Operator geringerer Präzedenz.
Mit der Tabelle unten gilt z. B.

**`a + b * c`** ist gleichwertig zu **`a + (b * c)`**.

| | |
|---|---|
| unäre Operatoren **`~ ! -- ++ + -`** cast | hohe Präzedenz |
| **`* / %`** | |
| **`+ -`** | |
| **`>> << >>>`** | |
| **`< > <= >= instanceof`** | |
| **`== !=`** | |
| **`&`** | |
| **`^`** | |
| **`\|`** | |
| **`&&`** | |
| **`\|\|`** | |
| bedingter Ausdruck **`b ? e1: e2`** | |
| Zuweisung | geringe Präzedenz |

Bei gleicher Präzedenz bindet der linke Operator stärker als der rechte (außer bei Zuweisungen und bedingten Ausdrücken), z. B.

**`a - b + c`** ist gleichwertig zu **`(a - b) + c`**



## Vorlesung Software-Entwicklung / Folie 30

**Ziele:**

Vorrangregeln für Operatoren

**in der Vorlesung:**

- Beispiele dazu
- vollständig geklammerte Ausdrücke

**nachlesen:**

Judy Bishop: Java lernen, 3.Aufl., Abschnitt 3.6

**Übungsaufgaben:**

Ausdrücke vollständig klammern

**Verständnisfragen:**

- Was hat die Reihenfolge, in der die Operatoren ausgewertet werden, mit der Struktur eines Ausdrucks zu tun?
- Die Reihenfolge, in der die Operanden ausgewertet werden, ist unabhängig von der Struktur des Ausdrucks. Begründung?

# Elementare Anweisungsformen

- **Zuweisung**: Variable = Ausdruck;
- **Alternative**: **`if (`**Bedingung**`)`** Anweisung **`else`** Anweisung
  Die Bedingung wird ausgewertet; ergibt sie wahr,
  so wird die erste sonst die zweite Anweisung ausgeführt.
  **`if (`**Bedingung**`)`** Anweisung
  Die Anweisung wird nur ausgeführt, wenn die Bedingung wahr liefert.
- **Schleife**: **`while (`**Bedingung**`)`** Anweisung
  Solange die Auswertung der Bedingung wahr ergibt,
  wird die Anweisung ausgeführt.
  **`do`** Anweisung **`while (`**Bedingung**`);`**
  Die Anweisung wird ausgeführt und solange wiederholt,
  wie die Bedingung wahr liefert.
- **Folge (Block)**: **`{`** Anweisung ... Anweisung **`}`**
  Die Anweisungen (und Deklarationen) werden nacheinander ausgeführt. Der Block kann für eine Anweisung stehen.
- **Aufruf**: Funktionsname **`(`**Parameterausdrücke**`);`**
  Die Ausdrücke werden ausgewertet und
  die Funktion mit den Parameterwerten aufgerufen.

Beispiel:
Größter gemeinsamer Teiler

```
{ int a, b;
  a = Text.ReadInt(in);
  b = Text.ReadInt(in);

  while (a != b)
  {
    if (a > b)
         a = a - b;
    else b = b - a;
  }

  System.out.println(a);
}
```



## Vorlesung Software-Entwicklung / Folie 31

**Ziele:**

Algorithmische Grundformen und ihre Notation in Java kennenlernen

**in der Vorlesung:**

Erläuterungen und Beispiele dazu

**nachlesen:**

Judy Bishop: Java lernen, 3.Aufl., Abschnitt 4.4, 5.2

**Übungsaufgaben:**

**Verständnisfragen:**

Wie transformiert man eine beliebige for-Schleife in eine while-Schleife, die dieselbe Wirkung hat?

# Funktionen (Methoden)

**Übersicht:**

- Funktionen dienen zur **Gliederung von Programmen**.
  Eine Funktion soll eine **übersichtliche Teilaufgabe** erledigen.
- Eine Funktion kann **Parameter** verwenden, deren **Werte erst im Aufruf** angegeben werden.
- Eine Funktion **berechnet ein Ergebnis** und/oder **verändert den Zustand** in ihrer Umgebung.
- **Methoden** sind Funktionen, die auf den Variablen von Objekten (oder Klassen) operieren.

**Funktionsdeklaration:**

Modifizierer Ergebnistyp Funktionsname (formale Parameter) Block

```
static   double   toFahrenheit  (double celsius)
                                { return celsius * 9 / 5 + 32;}
```

**Funktionsaufruf:**

Funktionsname (aktuelle Parameter)

```
double cDegree = 10; double fDegree;
fDegree = toFahrenheit (cDegree);
System.out.println (toFahrenheit (cDegree+10));
```



## Vorlesung Software-Entwicklung / Folie 32

**Ziele:**

Konzepte und Notation von Funktionen in Java

**in der Vorlesung:**

Funktionsdeklaration und -aufruf am Beispiel erläutern

**nachlesen:**

Judy Bishop: Java lernen, 3.Aufl., Abschnitt 3.3

**nachlesen:**

GdP-58 bis GdP-60

**Übungsaufgaben:**

**Verständnisfragen:**

- Beschreiben Sie die Struktur der letzten Beispielzeile.
- Beschreiben Sie die Ausführung der letzten Beispielzeile.

# Aufruf, Parameter, Ergebnis

- Zur **Ausführung eines Aufrufes** wird Speicher angelegt für die formalen Parameter, das Ergebnis und die lokalen Variablen der Funktion.
- Parameterübergabe **call-by-value**: Die **formalen Parametervariablen** werden mit den **Werten** der entsprechenden aktuellen Parameter initialisiert.
- Eine Anweisung `return` Ausdruck`;`
  beendet den Aufruf und bestimmt sein Ergebnis.

Beispiel GGT als Funktion:

```
int ggt (int a, int b)
{ while (a != b)
  {
    if (a > b)
         a = a - b;
    else b = b - a;
  }
  return a;
}
```

Aufrufe:

```
int g;
g = ggt(90, 54);
System.out.print(ggt(36, 54));
g = ggt(g, 4);
```



## Vorlesung Software-Entwicklung / Folie 33

**Ziele:**

Aufruf und Parameterübergabe verstehen

**in der Vorlesung:**

Korrekte und fehlerhafte Aufrufe an Beispielen erläutern

**nachlesen:**

Judy Bishop: Java lernen, 3.Aufl., Abschnitt 3.3

**nachlesen:**

GdP-58 bis GdP-60

**Übungsaufgaben:**

Schreiben Sie Funktionen für einfache Berechnungen, betten Sie sie in den Rahmen ein und führen Sie das Programm aus.

**Verständnisfragen:**

- Wie lange wird der Speicher für einen Aufruf gebraucht?
- Was gilt für die Existenz der Speicherbereiche von Aufrufen, wenn Funktionen aus anderen Funktionen aufgerufen werden?

# Funktionen ohne Ergebnis (Prozeduren)

Funktionen können eine **Änderung ihrer Umgebung** (Zustand, Ein-, Ausgabe) bewirken, statt ein Ergebnis zu berechnen. Sie werden dann als **Anweisungen** aufgerufen. (Zustandsändernde Funktionen mit Ergebnis sind möglich aber fragwürdiger Stil.)

Beispiel:

```
void aLine (int width, char c)
{ System.out.print ('=');
  int i = 2;
  while (i < width)
    { System.out.print (c); i = i+1; }
  System.out.println ('=');
}
void aTicket (int width, int depth, char c)
{ aLine (width, '=');
  int i = 2;
  while (i < depth)
    { aLine (width, c); i = i+1; }
  aLine (width, '=');
}
```

Aufrufe und Ausgabe:

```
aLine(7, '5');

  =55555=

aTicket(7, 6, '5');

  =======
  =55555=
  =55555=
  =55555=
  =55555=
  =======
```



## Vorlesung Software-Entwicklung / Folie 34

**Ziele:**

Zerlegung in Teilaufgaben am Beispiel

**in der Vorlesung:**

- Das Beispiel erläutern
- Kriterien für die Zerlegung und die Parameter erläutern

**Übungsaufgaben:**

**Verständnisfragen:**

- Wie ändern Sie die Funktionen, um mehrere Marken nebeneinander zu drucken?