

# Einführung in die Programmierung für NF

Zuweisungen, main-Methode und  
Kommentare

---

# Wiederholung: Deklaration lokaler Variablen

- Eine **Deklaration einer lokalen Variablen** (*Declaration*) hat die Form

```
Type VarName = Expression;
```

- Beispiele:

```
- int total = -5;
```

```
- int quadrat = total * total;
```

```
- boolean aussage = false;
```

# Deklaration lokaler Konstanten

- Eine Konstante wird durch Angabe des „Modifiers“ **final** deklariert.
- Beispiel:

```
    final int total = 100;
```
- Der Compiler stellt sicher, dass Konstanten nicht verändert werden.
- Konstanten sollten (wie auch Variablen) „sprechende“ Namen besitzen.
- **Nie „Magic Numbers“ verwenden:**
  - Anstelle von 365 im Programm für „Anzahl der Tage im Jahr“ verwende man besser **final int** tageProJahr = 365;
  - Für die mathematischen Größen  $\pi$  und  $e$  verwende man anstelle von 3.14159 und 2.7182 besser `Math.PI` bzw. `Math.E`
  - Gründe: Tippfehler, Dokumentation, Anpassung veränderbarer Konstanten

# ZUWEISUNGEN

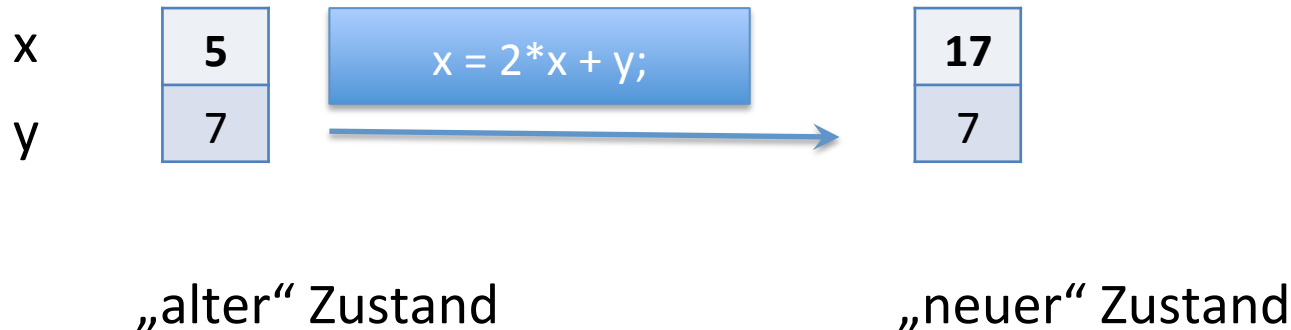
# Zuweisung

- Bei der **Zuweisung** (*Assignment*)

*VarName = Expression;*

wird der Wert der *Expression* im „alten“ Zustand berechnet. Dieser Wert wird im Nachfolgezustand der Variablen *VarName* als neuer Wert zugewiesen.

- Beispiel:



# Zuweisung: Textuelle Darstellung

Beispiel textuell:

**„alter“ Zustand**  $s1 = [(x, 5), (y, 7), (b, true)]$

**Zuweisung**  $x = 2 * x + y;$

**„neuer“ Zustand**  $s2 = [(x, 17), (y, 7), (b, true)]$

# Zuweisung: Abkürzende Schreibweisen

- Abkürzungen

- `x++;` steht für `x = x + 1;`
- `x--;` steht für `x = x - 1;`
- `x op= <Ausdruck>;` steht für `x = x op <Ausdruck>;`

- Beispiele

- `x += y;` steht für `x = x + y;`
- `b &= c;` steht für `b = b & c;`
- `x += 3 * y;` steht für `x = x + 3 * y;`

# Sequentielle Komposition

- **Sequentielle Komposition** wird durch Hintereinanderschreiben ausgedrückt.

- BNF-Regel:

*Statements = [Statement] | Statement Statements*

- Beispiel:

```
int total = 100;  
total = total + 100;
```

- Beachte: Die sequentielle Komposition ist selbst keine Anweisung, sondern besteht aus Anweisungen.



# BLÖCKE UND GÜLTIGKEITSBEREICHE

# Block

- Ein **Block** fügt mehrere Anweisungen durch geschweifte Klammern zu einer einzigen Anweisung zusammen. Er hat die Form:

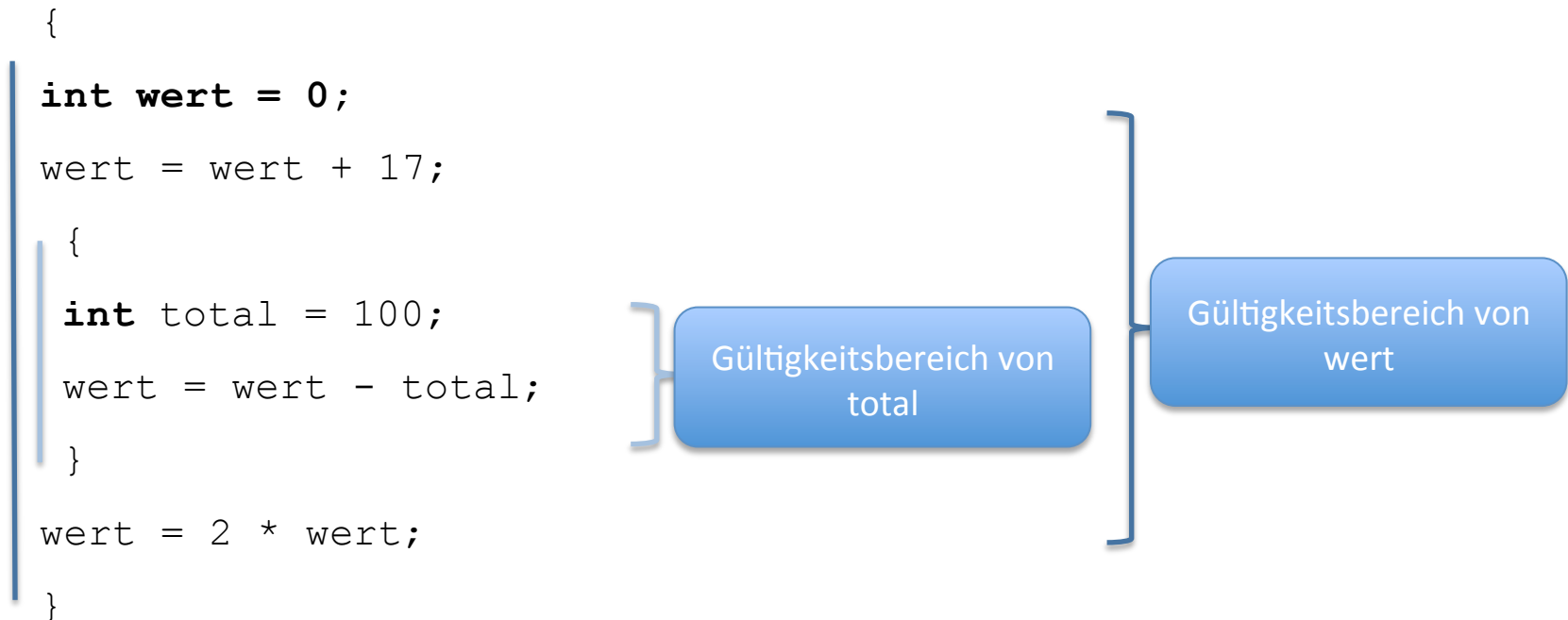
*{ Statements }*

- Durch einen Block werden **Sichtbarkeits-** und **Gültigkeitsbereich** von Variablen begrenzt:
  - Lokale Variablen sind nur innerhalb des umfassenden Blocks gültig und sichtbar.
  - Durch Methodenaufrufe können Programmpunkte erreicht werden, an denen eine Variable gültig aber nicht sichtbar ist. (dazu später mehr)

# Gültigkeitsbereich

- Der **Gültigkeitsbereich** einer lokalen Variablen oder Konstante ist der **die Deklaration umfassende Block**.
- Außerhalb dieses Blocks *existiert die Variable nicht*.

## Beispiel:



# Beispiel I

```
{  
    int m = 13, n = 2, z = 0;  
    m = m + n;  
    {  
        double x;  
        x = m / n;  
        int i = (int) x;  
        z = i - n;  
    }  
    n++;  
    int h = n + 1;  
}
```

# Beispiel II

```
{  
    double a = 2.0, b = 3.4;  
    {  
        a = a + b;  
        int c = (int) a;  
        a = c;  
    }  
    a--;  
    double d = a + b;  
    boolean test = a != d;  
}
```

# MAIN-METHODE

# Main-Methode

- Jedes JAVA-Programm benötigt eine Main-Methode
- Sie wird aufgerufen, wenn das Programm gestartet wird

# Main-Methode

- Die Main-Methode hat einen festgelegten Methodenkopf

```
public static void main (String[] args) {  
    Anweisungen;  
}
```



# KOMMENTARE – JAVADOC

# Javadoc

- Durch

```
// bla, bla}
```

wird eine Zeile oder der Rest einer Zeile zum Kommentar.

- Mehrere Zeilen können folgendermaßen auskommentiert werden:

```
/* bla
```

```
bla */
```

- Zur Erzeugung von Kommentaren zu Klassen und Methoden wird die spezielle Form verwendet:

```
/** und */
```

# Javadoc Beispiel

```
/**
 *Diese Klasse dient nur zum Anzeigen des
 *Strings "Hallo, Welt!" auf den Bildschirm
 */
public class HalloDoc {
    /**
     * Die Methode main druckt "Hallo, Welt!"
     */
    public static void main(String[] args) {
        System.out.println("Hallo, Welt!");
    }
}
```

# Spezielle Anweisungen Javadoc

- @see für Verweise
- @author für Namen des Autors
- @version für die Version
- @param für die Methodenparameter

# Beispiel II

```
/**
 * Diese Klasse dient zur Berechnung des Quadrats.
 *
 * @author Martin Wirsing
 * @version 1.1
 */
public class Square {
    /**
     * Diese Methode dient nur zur Illustration der Parameterbehandlung durch javadoc.
     *
     * @param value ein formaler Parameter vom Typ int
     * @return das Quadrat von value
     */
    public static int square(int value) {
        return value * value;
    }
}
```

# Beispiel II

```
/**  
 * Diese Klasse dient nur zum Test von Square  
 */  
public class SquareTest {  
    /**  
     * Die Methode main druckt einen Testfall von square  
     */  
    public static void main(String[] args) {  
        int wert = 17;  
        System.out.println("Das Quadrat von " + wert +  
            " ist " + Square.square(wert));  
    }  
}
```

Vielen Dank für Ihre Aufmerksamkeit