Tagesprogramm

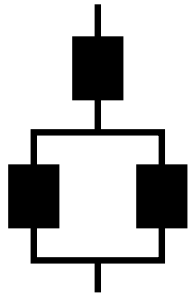
Ausdrücke und Anweisungen (inkl. Bedingungen)

Sequenz, Auswahl, Wiederholung

Blockstruktur und Scope

Daten und Algorithmen

If-Anweisung



```
/**
 * ist die Zahl gerade?
 */
public class GeradeZahl {
    public static void main(String[] args) {

        int zahl = 27;

        if ((zahl % 2) == 0) {  // wenn Divisionsrest ist 0
                                // führe ersten Zweig aus
            System.out.println("Zahl ist gerade");
        }
        else {                  // sonst
                                // führe Alternativzweig aus
            System.out.println("Zahl ist ungerade");
        }
    }
}
```

Ausdruck versus Anweisung

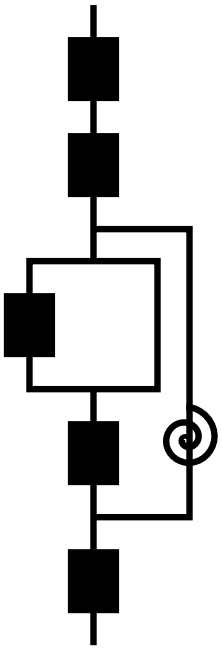
Ausdruck liefert bei Berechnung einen Wert als Ergebnis:

`2`	Literal, Ergebnis ist 2
`zahl`	Lesen des Wertes in der Variablen
`zahl % 2`	Anwendung eines Operators auf zwei Werte
`(zahl % 2) == 0`	geschachtelte Ausdrücke

Anweisung wird ausgeführt, enthält oft Ausdrücke, am Ende `;` oder `{...}`:

`int zahl;`	Variablendeklaration
`zahl = 27;`	Zuweisung
`int zahl = 27;`	Variablendeklaration mit Initialisierung
`while (...) {...}`	Kontrollstruktur (Schleife)
`if (...) {...} else {...}`	Kontrollstruktur (bedingte Anweisung)
`System.out.println(...);`	Methodenaufruf als Anweisung

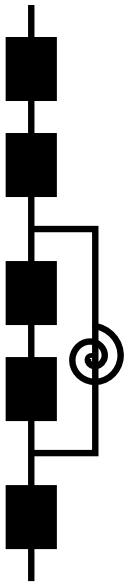
Geschachtelte Kontrollstrukturen



```
/* Summe nicht durch 3 teilbarer Zahlen von 1 bis 100 */
public class KrummeSumme {
    public static void main(String[] args) {

        int sum = 0;
        int i = 1;

        while (i <= 100) {
            if ((i % 3) != 0) {
                sum = sum + i;
            }
            else {}          /* leeren else-Zweig weglassen */
            i = i + 1;
        }
        System.out.println(sum);
    }
}
```

Bedingung im Ausdruck

```
/* Summe nicht durch 3 teilbarer Zahlen von 1 bis 100 */
public class KrummeSummeMitBedingtemAusdruck {
    public static void main(String[] args) {

        int sum = 0;
        int i = 1;

        while (i <= 100) {
            sum = sum + (((i % 3) != 0) ? i : 0);
            i = i + 1;
        }
        System.out.println(sum);
    }
}
```



Aufgabe: Anweisung erkennen

Angenommen, wir finden im Programmtext ; oder { . . . }.
Handelt es sich dabei um das Ende einer Anweisung?

A ja, in jedem Fall

B bei ; ja, bei { . . . } möglicherweise nicht

C bei { . . . } ja, bei ; möglicherweise nicht

D an ; und { . . . } kann man das Ende nicht erkennen

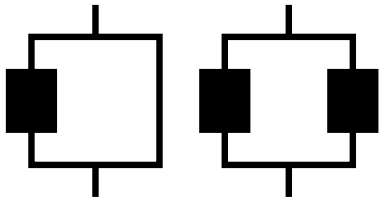
Strukturierte Programmierung

Sequenz (Hintereinanderausführung)

```
while (...) {...} System.out.println(...);
(i % 3) == 0
```

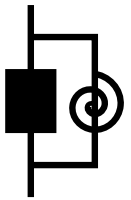
Auswahl

```
if (...) {...} else {...}
if (...) {...}
... ? ... : ...
```



Wiederholung

```
while (...) {...}
```



Aufgabe: Kombinationsmöglichkeiten

Lösen Sie in Gruppen zu 2 bis 3 Personen folgende Aufgabe:

Zählen Sie Möglichkeiten auf, wie man in einem Programm Sequenz, Auswahl und Wiederholung miteinander kombinieren kann.

Zeit: 3 Minuten

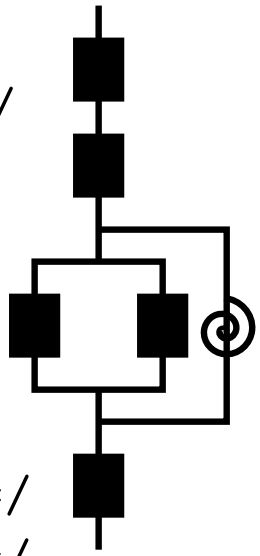
Blockstruktur

```
{
    {                          Einrückungstiefe
        {
            {                  innerer versus äußerer Block
            }
            {                  Scope: Bereich bis Blockende
                {
                }
            }
        }
        {
        }
    }
}
```

Scope (Gültigkeitsbereich) lokaler Variablen

```
{
    {                          Variablen nur im Scope zugreifbar
        { int i;
            { int j;
            }
            { int j;
                { int k;
                }
            }
        }
        { int i;
        }
    }
}
```

Größter gemeinsamer Teiler



```
/** berechne größten gemeinsamen Teiler von 1027 und 395 **/
public class GroessterGemeinsamerTeiler {
    public static void main(String[] args) {

        int m = 1027;
        int n = 395;

        while (m != n) {        /* solange m und n ungleich */
            if (m > n) {        /* wenn m größer n          */
                m = m - n;      /* ziehe n von m ab         */
            }
            else {              /* sonst (n größer m)       */
                n = n - m;      /* ziehe m von n ab         */
            }
        }                       /* m und n sind nun gleich  */
        System.out.println(m);
    }
}
```

Daten und Algorithmen

Daten: Inhalte aller Variablen, z.B. `m` und `n`

Algorithmus: Was das Programm macht, z.B.

> solange zwei natürliche Zahlen ungleich sind,
> ziehe die kleinere von der größeren ab;
> danach gib die resultierende Zahl aus

Unsere Interpretation kennt der Computer nicht:

> die derart errechnete Zahl ist der größte gemeinsame Teiler
> der ursprünglichen Zahlen (und aller Zwischenergebnisse)

Aufgabe: Programm vereinfachen

Finden Sie in Gruppen zu 2 bis 3 Personen heraus, was dieses Programm ausgibt (auch mit anderen Initialisierungen von x und y) und vereinfachen Sie es:

```
public class VereinfacheMich {
    public static void main(String[] args) {
        int x = 911;  int y = 29;     /* Zahlen änderbar */
        int z = 0;

        if (x >= 0) {
            if (y >= 1) {
                while (y <= x) {
                    x = x - y;
                    z = z + 1;
                }
                System.out.println(z);
                System.out.println(x);
            }
        }
    }
}
```