Informatik: Kontrollstrukturen

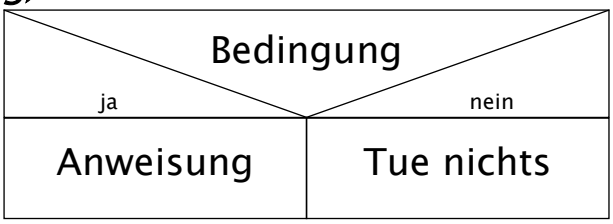
Ziele

- Die `if`-Anweisung mit ihren Eigenarten anwenden können
- Mit Blöcken und Einrückungen richtig umgehen können
- Mit dem Datentyp `boolean` umgehen können
- Auch komplexe Bedingungen auswerten können
- Die Funktionsweise der einzelnen Schleifenarten verstehen
- Die richtige Schleifenart angepasst auf die Art des Problems auswählen können
- `break` und `continue` verstehen
- Gültigkeitsbereiche von Variablen richtig einsetzen können
- Die `switch`-Anweisung anwenden können

Kontrollstrukturen erlauben die Steuerung des Programmablaufes

if-Anweisungen (bedingte Anweisung)

if (Bedingung)
    Anweisung;



Beispiel: double-Toleranz

```
double a = 1 / 3.0;
double b = 10 + a - 10;
if (a == b)
    System.out.println("gleich");
else
    System.out.println("ungleich");
```

ACHTUNG: Einrückungen!!! 💣

ACHTUNG: Ergibt ungleich!!!

BESSER: Toleranz

```
…
final double EPSILON = 1E-10;
if (Math.abs(a - b) < EPSILON)
   …
```

Beispiel: Verschachtelte if-Anweisung

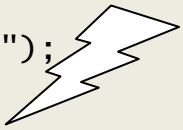
```
int a = 4;
int b = 2;
int c = 1;
int min = 0;
if (a < b)
    if (a < c)
        min = a;
    else
        min = c;
else
    if (b < c)
        min = b;
    else
        min = c;
System.out.println("Minimum: " + min);
```

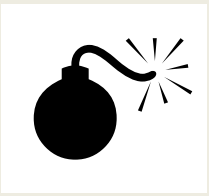
REGEL:
Ein **else** bezieht sich immer auf das letzte freie **if**

ACHTUNG: Einrückungen!!!💣

Beispiel: *Dangling Else*

```
int sichtweite = 80;
if (sichtweite < 60)
   if (sichtweite < 30)
      System.out.println("Geschwindigkeit 10");
else
   System.out.println("Freie Fahrt");
```





Blöcke

```
if (Bedingung) {
   Anweisung1;
   ...
   Anweisungn;
}
```

bündeln mehrere Anweisungen zu einem Block

{ muss neben `if` platziert werden

```
int sichtweite = 20;
if (sichtweite < 60) {
   System.out.print ("Langsam fahren");
   if (sichtweite < 30)
      System.out.println(" Geschwindigkeit 10");
} else
   System.out.println("Freie Fahrt");
```

Beispiel: Quadratische Gleichung

$$ax^2 + bx + c = 0 \qquad x_{1/2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

```
double a = 1;
double b = 3;
double c = 1;
double d = b * b - 4 * a * c;
if (d < 0)
   System.out.println("Keine Lösung");
else
   if (d == 0)
      System.out.println("Eine Lösung: " + -b / (2 * a));
   else {
      d = Math.sqrt(d);
      System.out.println("Zwei Lösungen: " +
         (-b + d) / (2 * a) + ", " +
         (-b - d) / (2 * a));
   }
```

Wahrheitswerte und der Datentyp boolean

```
double a = 5;
double b = 6;
if (a >= 5 && a <= 10 && a > b) …
     wahr      wahr      falsch
              falsch
```

! Nicht && Und || Oder

zuerst !, dann &&,
dann ||

```
boolean istOK = false;
istOK = !istOK && a > b || a < 10;
          true    false
             false          true
                    true
```

istOK **true**

Beispiel: Bedingte Auswertung

```
double d = 0;
if (d != 0 && 3 / d > 0)
   …
if (d == 0 || 7 / d > 1)
   …
```

Es tritt keine Division durch Null auf!!!

REGEL für **&&**:
Nur dann wenn linke Bedingung **true** liefert, wird rechte ausgewertet

REGEL für **||**:
Wenn linke Bedingung **true** liefert, wird rechte nicht ausgewertet

while-Schleife (abweisende)

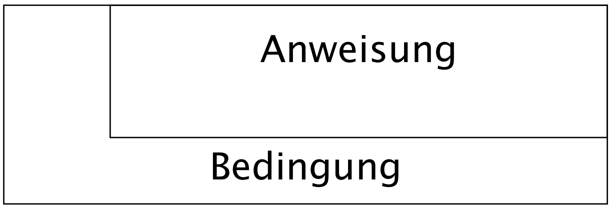
while (Bedingung**)**
    Anweisung**;**

while dtsch. **SOLANGE**

do-Schleife (nicht abweisende)

do
    Anweisung**;**
while (Bedingung**);**

do-while dtsch. **TUE SOLANGE**



Beispiel: Endlosschleife

```
double d = 2;
while (d >= 0) {
    System.out.println(d);
    d = d / 2;
}
```

ACHTUNG: Einrückungen!!! 💣

Beispiel: Verschachtelte Schleifen

```
int x = 0;
while (x < 10) {
    int y = 0;
    while (y < 10 - x) {
        System.out.print(y + " ");
        y = y + 1;
    }
    System.out.println();
    x = x + 1;
}
```

Welche Ausgabe liefert Programm?

ACHTUNG: Einrückungen!!! 💣

Programm umschreiben so dass nicht abweisende Schleifen verwendet werden

Beispiel: Schleifenabbruch und -kurzschluss mit break und continue

```
int n = 0;
while (true) {
   n = n + 1;
   if (n == 5)
      break;
   if (n == 3)
      continue;
   System.out.print(n + " ");
}
```

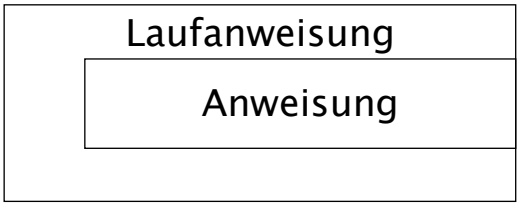
break sofort Schleifenabbruch

continue sofort nächster Bedingungstest

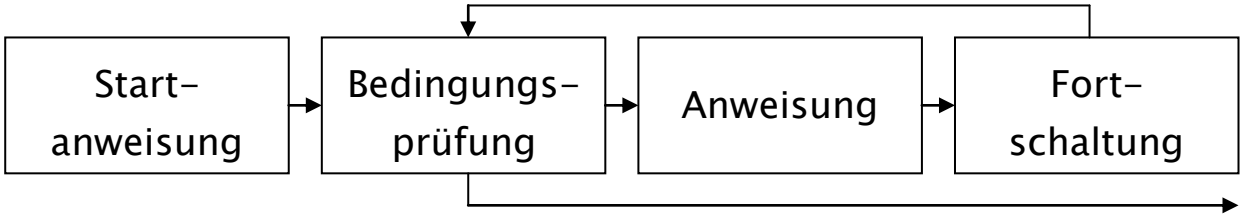
Welche Ausgabe liefert dieses Programm?

for-Schleife

for (Startanweisung; Bedingung; Fortschaltanweisung)
    Anweisung;




```
for (int i = 3; i < 5; i = i + 1)
    System.out.println(i);
```



Startanweisung, Bedingung, Fortschaltanweisung können entfallen

```
for (;;) {
    …
}
```

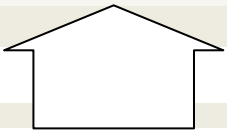
Beispiel: Eine Aufgabe, drei Schleifen

```
int sum = 0;          int sum = 0;       int sum = 0;
int i = 1;            int i = 1;         for (int i = 1; i < 4; i = i + 1)
while (i < 4) {       do {                  sum = sum + i;
   sum = sum + i;        sum = sum + i;
   i = i + 1;            i = i + 1;
}                     } while (i < 4);
```

Gültigkeitsbereiche

```
{
   int a = 3;
   …
   {
      int b = a + 4;
      …
   }
}
```

```
{
   int a = 3;
   …
   {
      int b = a + 4;
      …
   }
}
```



```
{
   int a = 3;
   {
      int b = a + 4;
   }
   a = a + 3;
   b = a;
   c = 4;
   {
      int b = 7;
      int a = 0;
   }
   int c = 3;
}
```

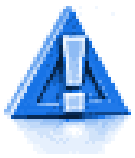
Eine Variablendefinition gilt ab dem Ort der Definition bis zum Ende des Blockes

Gültigkeitsbereich engl. **scope**

Lebensdauer einer Variablen

Wird Block mehrmals zur Laufzeit durchlaufen, so wird jedes Mal eine neue **Inkarnation** der Variable angelegt

Programmierregeln



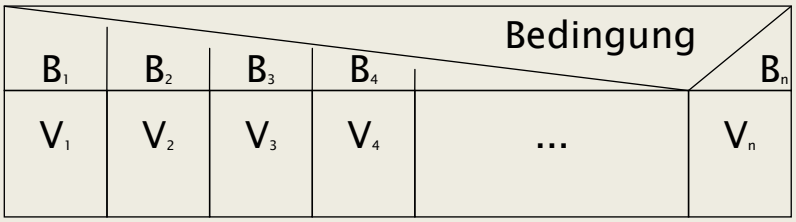
- **break** und **continue** wenn möglich nicht verwenden
- **for**-Schleife dann einsetzen wenn vor Durchlauf der Schleife bereits bekannt ist, wie oft diese durchlaufen werden soll
- In einer **for**-Schleife darf die Laufvariable nie händisch verändert werden:

```
for (int i = 0; i < 10; i = i + 1)
   if (i == 3)
       i = 10;
```

- Als Laufvariablenname für **for**-Schleifen werden meist **i**, **j** verwendet
- Variablen nur in jenen Blöcken deklarieren in denen sie benötigt werden (Prinzip der **Lokalität**)

switch-Anweisung (Mehrfachauswahl)

```
int i = 3;
switch (i) {
    case 1: {
        anz1 = anz1 + 1;
        break;
    }
    case 2: {
        anz2 = anz2 + 1;
        break;
    }
    default: {
        System.out.println(i);
    }
}
```



ACHTUNG:

- Doppelte Einrückungen!!!
- **break**