

Java

Prof. Dr. Stefan Enderle
NTA Isny

5. Kontrollstrukturen

Allgemein

- Kontrollstrukturen dienen zur Steuerung des Programmablaufs.
- (Bemerkung: C, C++ und Java besitzen nahezu die selben Kontrollstrukturen.)
- Ein Programmablauf kann z.B. durch ein Flußdiagramm dargestellt werden.

Flußdiagramme

- Ein Flußdiagramm ist eine grafische Methode zur Beschreibung von Programmabläufen

Grundelemente

- Start
- Stop
- Anweisung
- Fallunterscheidung
- Ein- / Ausgabe

- Pfeile zeigen den Programmablauf an

Anweisungen

- Eine Anweisung ist in jeder Programmiersprache der grundlegende Baustein.
- Beispiele für Anweisungen:
 - `a = 5;`
 - `System.out.print(name);`

Anweisungsblöcke

- Mehrere Anweisungen können durch geschweifte Klammern zu einem Block zusammengefasst werden.

- Beispiel:

```
{  
    a = 5;  
    System.out.print(name) ;  
}
```

Fallunterscheidung: if

- Eine Fallunterscheidung dient dazu, Code in Abhängigkeit einer Bedingung auszuführen.
- Formal: `if (<bedingung>)`
`<anweisung>`
- Wenn <bedingung> true ist, wird <anweisung> ausgeführt.
- Beispiel:

```
if (a>b)  
    System.out.print(name) ;
```

Fallunterscheidung: if

- Soll die <Anweisung> aus mehr als einer Anweisung bestehen, wird ein *Anweisungsblock* benötigt.
- Beispiel:

```
if (a>b) {  
    System.out.print(name) ;  
    a = a-b;  
}
```

- Vorsicht falsch:

```
if (a>b)  
    System.out.print(name) ;  
    a = a-b;
```

Fallunterscheidung: if / else

- Wenn in beiden Fällen einer Bedingung (true und false) unterschiedlich weitergearbeitet werden soll, kann `if / else` benutzt werden:

- Formal:

```
if (<bedingung>
    <anweisung1>
else
    <anweisung2>
```

- Beispiel:

```
if (a==b)
    System.out.print("a gleich b");
else
    System.out.print("a ungleich b");
```

Fallunterscheidung: if / else

- Beispiel mit Anweisungsblöcken:

```
if (a==b) {
    System.out.print("a gleich b");
    a=0;
}
else {
    System.out.print("a ungleich b");
    a=a-b;
}
```

Schachtelung von if / else

- Beispiel:

```
if (a!=b)
    if (a>b)
        System.out.print("a größer b");
    else
        System.out.print("a kleiner b");
else
    System.out.print("a gleich b");
```

Schachtelung von if / else

- "Schöner": Schachtelung mit Block-Klammern
- Beispiel:

```
if (a!=b) {
    if (a>b)
        System.out.print("a größer b");
    else
        System.out.print("a kleiner b");
}

else {
    System.out.print("a gleich b");
}
```

Kaskadieren von if / else

- Im else-Zweig lässt sich gleich wieder eine Abfrage machen – evtl. mit neuem else-Zweig, ...
- Beispiel:

```
if (a>b)
    System.out.print("a ist größer");
else if (a<b)
    System.out.print("a ist kleiner");
else
    System.out.print("a gleich b");
```

Switch-Anweisung

- Soll eine Variable auf viele mögliche Werte geprüft werden, ist es manchmal einfacher/eleganter anstatt einer kaskadierten if-Anweisung die switch-Anweisung zu benutzen.
- Beispiel: (if-Kaskade)

```
if (a==1)
    System.out.print("a ist 1");
else if (a==2)
    System.out.print("a ist 2");
else if (a==3)
    System.out.print("a ist 3");
else if (a==4)
    System.out.print("a ist 4");
else
    System.out.print("a größer als 4");
```

Switch-Anweisung

- Eleganter mit switch:

```
switch (a)
{
    case 1:    System.out.print("a ist 1");
              break;
    case 2:    System.out.print("a ist 2");
              break;
    case 3:    System.out.print("a ist 3");
              break;
    case 4:    System.out.print("a ist 4");
              break;
    default:   System.out.print("a > 4");
}
}
```

Switch-Anweisung

- Allgemeine Syntax:

```
switch (<variable>
{
    case <konstante> :   <anweisungsliste>
                        break ;
    case <konstante> :   <anweisungsliste>
                        break ;
    case <konstante> :   <anweisungsliste>
                        break ;
    ...
    default:             <anweisungsliste>
}

```

- Der default-Teil darf fehlen.
- Achtung: 'break's nicht vergessen!!

Switch-Anweisung

- Manchmal sind "leere Anweisungslisten" und sinnvoll.
- Beispiel:

```
switch (c)
{
    case 'a':
    case 'A':      System.out.print("a");
                  break;

    case 'b':
    case 'B':      System.out.print("b");
                  break;

    case 'c':
    case 'C':      System.out.print("c");
                  break;
}
```

Schleifen: while

- Eine Schleife dient dazu, Code in Abhängigkeit einer Bedingung mehrfach auszuführen.
- Formal: `while (<bedingung>)`
`<anweisung>`
- Solange <bedingung> true ist, wird <anweisung> ausgeführt.
- Beispiel:

```
int i = 10;
while (i>0) {
    System.out.print("i ist " + i);
    i = i-1;
}
```

Schleifen: do / while

- Es kann auch jeweils NACH Ausführung der Anweisung geprüft werden:
- Formal: **do**
 <anweisung>
 while (<bedingung>)
- Beispiel:

```
int i = 10;  
do {  
    System.out.print("i ist " + i);  
    i = i-1;  
}  
while (i>0);
```

Endlosschleifen

- Wird als Bedingung in der Schleife

```
while (<bedingung>
      <anweisung>
```

eine Bedingung angeben, die immer true ist, ergibt sich eine Endlosschleife.

- Beispiel:

```
while (1==1) { ... } // immer true
while (true) { ... } // immer true
```

Zählschleifen: for

- Eine Zählschleife (for-Schleife) wird dazu benutzt, Code eine bestimmte Anzahl mal auszuführen.
- Beispiel: $summe = 1+2+3+4+\dots+10$

mit while:

```
int summe=0;
int i=1;      // Schleifenzähler

while (i<=10) {
    summe = summe + i;
    i = i+1;
}
```

Zählschleifen: for

- Beispiel: $summe = 1+2+3+4+\dots+10$

mit for:

```
int summe=0;

for (int i=1; i<=10; i++) {
    summe = summe + i;
}
```

Zählschleifen: for

- Syntax: `for (<start>; <bedingung>; <iteration>)
 <anweisungsliste>`
- Ablauf:
 - Am Anfang wird einmal <start> ausgeführt.
 - Solange <bedingung> true ist, wird die <anweisungsliste> ausgeführt und danach der Iterationsausdruck <iteration> ausgeführt.

Zählschleifen: for

- **"Grundversion":**

n Durchläufe (von 0 bis n-1):

```
for (int i=0; i<n; i++) {  
    // tue irgend etwas  
}
```

Zählschleifen: for

- **Bemerkungen:**

- Sollen am Anfang gleich mehrere Variablen initialisiert werden, müssen diese durch Komma getrennt werden:

```
for (i=0, j=0; i<5; i++) ...
```

- Natürlich kann auch abwärts gezählt werden:

```
for (int i=10; i>=0; i--) ...
```

- Oder jede 2. Zahl:

```
for (int i=0; i<20; i=i+2) ...
```

- Theoretisch können beliebige Anweisungen in der Schleife stehen ...

break

- Mit der Anweisung `break` lässt sich nicht nur aus einer `switch`-Anweisung springen, sondern auch aus einer `while`- oder `for`-Schleife.
- Beispiel:

```
while (i<1000) {  
    // berechne etwas  
    if (zeit>10) break; // zu lange -> raus  
}
```

continue

- Die Anweisung `continue` beendet die aktuelle Iteration einer `for`- oder `while`-Schleife und startet wieder von oben.
- Beispiel:

```
for (int i=0; i<10; i++) {  
    if (i%2 != 0) continue;  
    System.out.print(i + " ist gerade");  
}
```