Übersicht

Inhalte

✓ Blöcke – Kontrollstrukturen für Sequenz

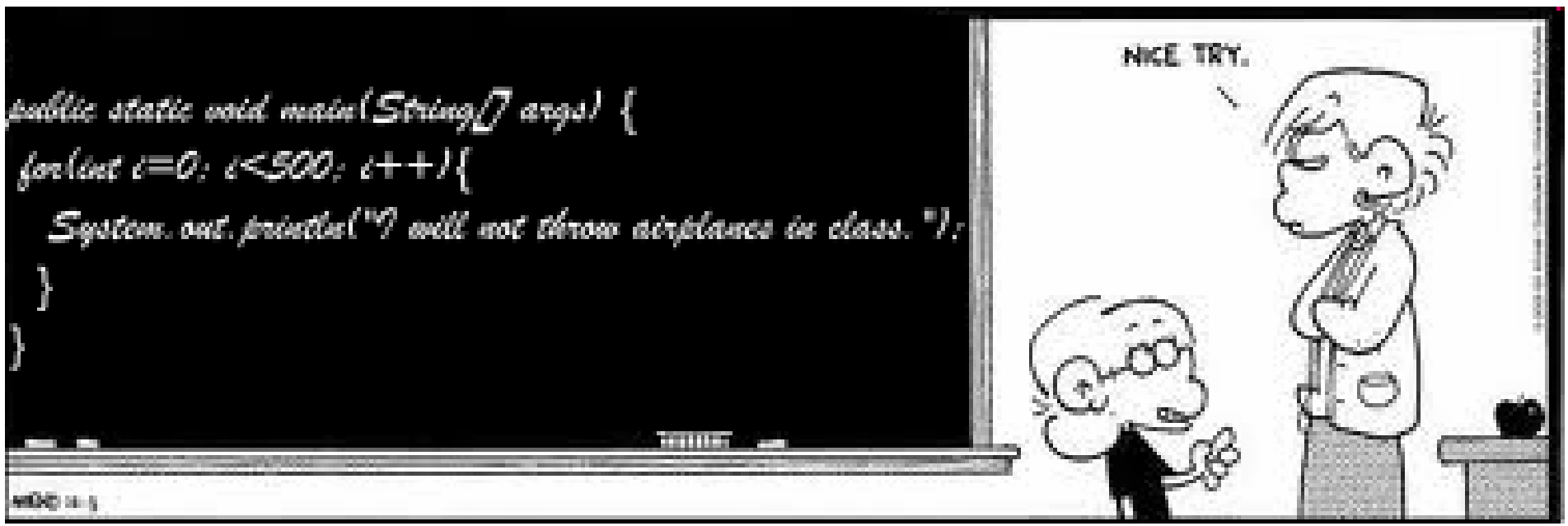
Selektion

Iteration

Sprunganweisungen

Kontrollstrukturen

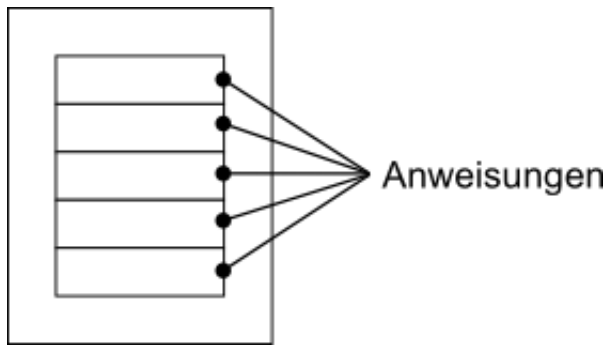
- Kontrollstrukturen definieren die Reihenfolge, in der Aktionen durchgeführt werden. Dieses Kapitel beschreibt
  - Selektion/Verzweigungen (if, else, switch/case)
  - Schleifen (while, for, do while, break, continue)



Blöcke – Kontrollstrukturen für die Sequenz

- Ein Block (eine zusammengesetzte Anweisung) kann an jeder Stelle stehen, an der eine einzelne Anweisung angeschrieben werden kann.

```
{
        Anweisung_1
        Anweisung_2
        …
        Anweisung_n;
}
```



- Ein Block ist eine Sequenz von Anweisungen

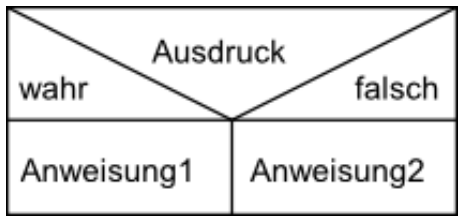
Übersicht

Inhalte

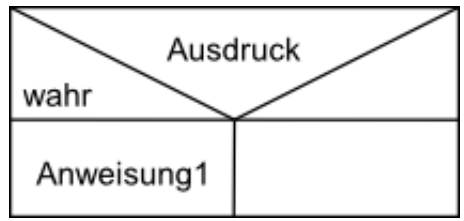
- ✓ Blöcke – Kontrollstrukturen für Sequenz
- ✓ Selektion
- Iteration
- Sprunganweisungen

Selektion If und else

- Bedingte Anweisung (die auch geschachtelt werden kann) und einfache Alternative



(*if-else*-Anweisung)



(Anweisung mit *if*)

- Für die defensive Programmierung sollten stets geschweifte Klammern verwendet werden, damit der Handlungsablauf leicht um weitere Anweisungen ergänzt werden kann

```
if (BEDINGUNG) {
    BLOCK
} else {
    BLOCK
}
```

```
int zahl=7;

if(zahl==7) {
    System.out.println("sieben");
}
```

Selektion if und else Verschachtelung

- if und else - Verschachtelung

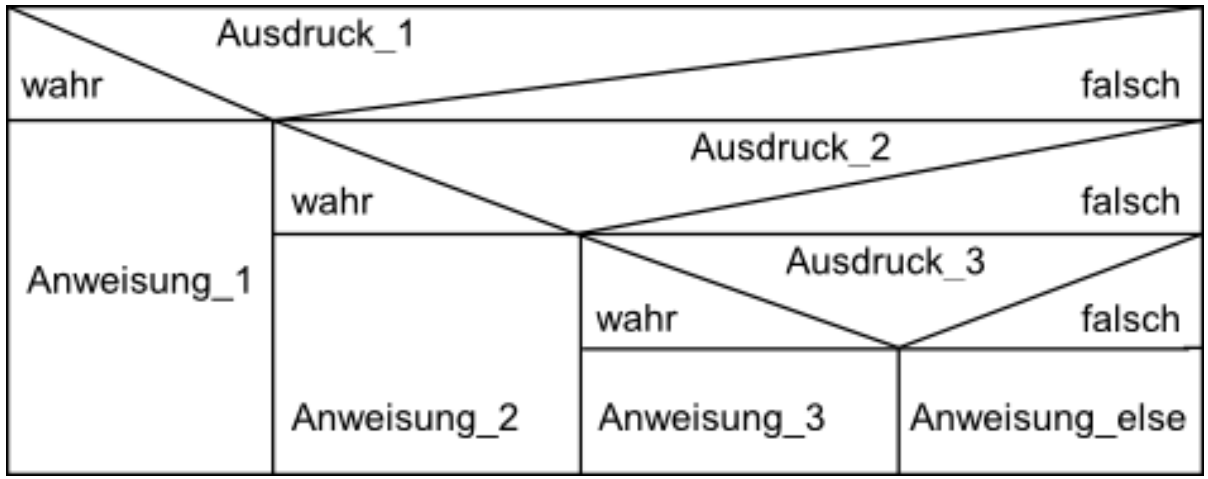
```
int zahl=8;

if(zahl==7) {
 System.out.println("sieben");
}else {
    if(zahl==8) {
        System.out.println("acht");
    }else {
        System.out.println("nicht sieben und nicht acht");
    }
}
```

```
acht
```

Selektion if und else Mehrfachauswahl

- Mehrfache Alternative: *else-if*

```
if(Ausdruck_1)
    Anweisung_1
else if(Ausdruck_2)
    Anweisung_2
…
…
else if(Ausdruck_n)
    Anweisung_n
else
    Anweisung_else
```



*(Struktogramm der else-if-*Anweisung)

Aufgaben zu if und else

Legen Sie für die folgenden Aufgaben entsprechende Klassen inkl. main-Methode in Eclipse an (z.B. Aufg_05_01.java für Aufgabe 05.01 etc.)

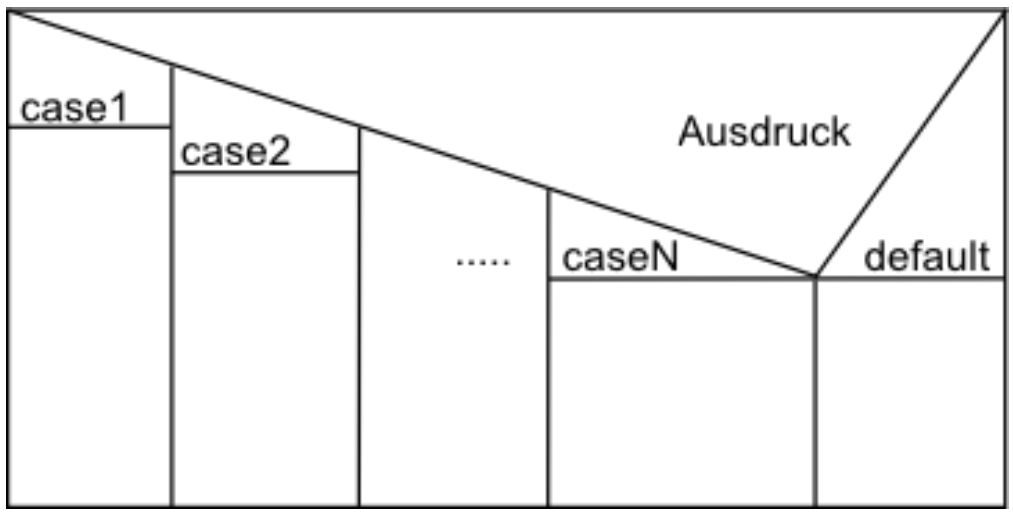
- **Aufg. 05.01**: Lesen Sie das Alter eines Menschen ein. Geben Sie dann in Abhängigkeit vom Alter die entsprechende Zeichenkette aus: Kind/Jugendlicher (unter 18), Erwachsener (18 bis unter 65), Rentner (ab 65).

- **Aufg. 05.02**: Lesen Sie eine int-Variable für eine Punktzahl (max. 10) ein. Daraus abgeleitet soll die Note als Text ausgegeben werden: 10 – sehr gut, 9 – gut, 8 – befriedigend, 7 – ausreichend, <7 – ungenügend

- **Aufg. 05.03**: Lassen Sie den Benutzer drei Zahlen eingeben. Geben Sie auf dem Bildschirm die Zahl aus, die am kleinsten ist.

Selektion switch I

- Mehrfache Alternative: switch
  - switch für eine Selektion unter mehreren Alternativen
    - switch(Ausdruck)
    - case X: für die einzelnen Fälle, nach dem Doppelpunkt folgen die Anweisungen
    - break schließt einen case-Block ab
    - default-Block: wenn kein Fall erreicht wird, wird der default-Block ausgeführt



(Struktogramm der switch-Anweisung)

Selektion switch II

- Mehrfache Alternative: switch
- der auszuwertende Ausdruck muss bei switch einer der folgenden Datentypen sein

char, byte, short, int, Character, Byte, Short, Integer (seit Java 7 auch String)

```
int a=2;
switch(a) {
    case 1: System.out.println("a ist eins\n");
    break;
    case 2: System.out.println("a ist zwei\n");
    break;
    case 3: System.out.println("a ist drei\n");
    break;
    default: System.out.println("a ist irgendwas\n");
    break;
}
```

```
a ist zwei
```

Selektion switch III

- Mehrfache Alternative: switch case mit Aufzählerkonstanten

```
public class Richtungsweiser
{
   public enum Richtung {LINKS, RECHTS}

   public static void main (String[] args)
   {
      Richtung ref = Richtung.RECHTS;
      switch (ref)
      {
        case LINKS:
            System.out.println ("LINKS");
            break;
          case RECHTS:
            System.out.println ("RECHTS");
            break;
      }
   }
}
```

Selektion IV

- Mehrfache Alternative: switch case

```
public class ZeichenTester{
   public void testeZeichen (char c){
      switch (c){
         case '\t':
         case '\n':
         case '\r':
            System.out.println ("Steuerzeichen");
            break;
         default:
            System.out.println ("Kein Steuerzeichen: " + c);
      }
   }
   public static void main (String[] args){
      ZeichenTester pars = new ZeichenTester();
      pars.testeZeichen ('\t');
      pars.testeZeichen ('A');
      pars.testeZeichen ('\r');
   }
}
```

```
Steuerzeichen
Kein Steuerzeichen: A
Steuerzeichen
```

- Fehlt die *break*-Anweisung, so werden die Anweisungen nach der nächsten *case*-Marke abgearbeitet. Dies geht so lange weiter, bis ein *break* gefunden wird oder bis das Ende der *switch*-Anweisung erreicht ist.

Aufgaben switch

Mehrfache Legen Sie für die folgenden Aufgaben entsprechende Klassen inkl. main-Methode in Eclipse an (z.B. Aufg_05_04.java für Aufgabe 05.04 etc.)

- **Aufg. 05.04**: Lesen Sie eine int-Variable für eine Punktzahl (max. 10) ein. Daraus abgeleitet soll die Note als Text ausgegeben werden: 10 – sehr gut, 9 – gut, 8 – befriedigend, 7 – ausreichend, <7 – ungenügend

- **Aufg. 05.05**: Ein Benutzer soll eine Zahl als Kennzeichen für einen Wochentag angeben (1=Montag, 2=Dienstag etc.). Auf dem Bildschirm soll daraufhin der Name des Wochentags erscheinen.

- **Aufg. 05.06**: Abwandlung 05.05: Geben Sie für Montag bis Mittwoch „erste Wochenhälfte“, für Donnerstag und Freitag „zweite Wochenhälfte“ und für Samstag und Sonntag „Wochenende“ aus.

Übersicht

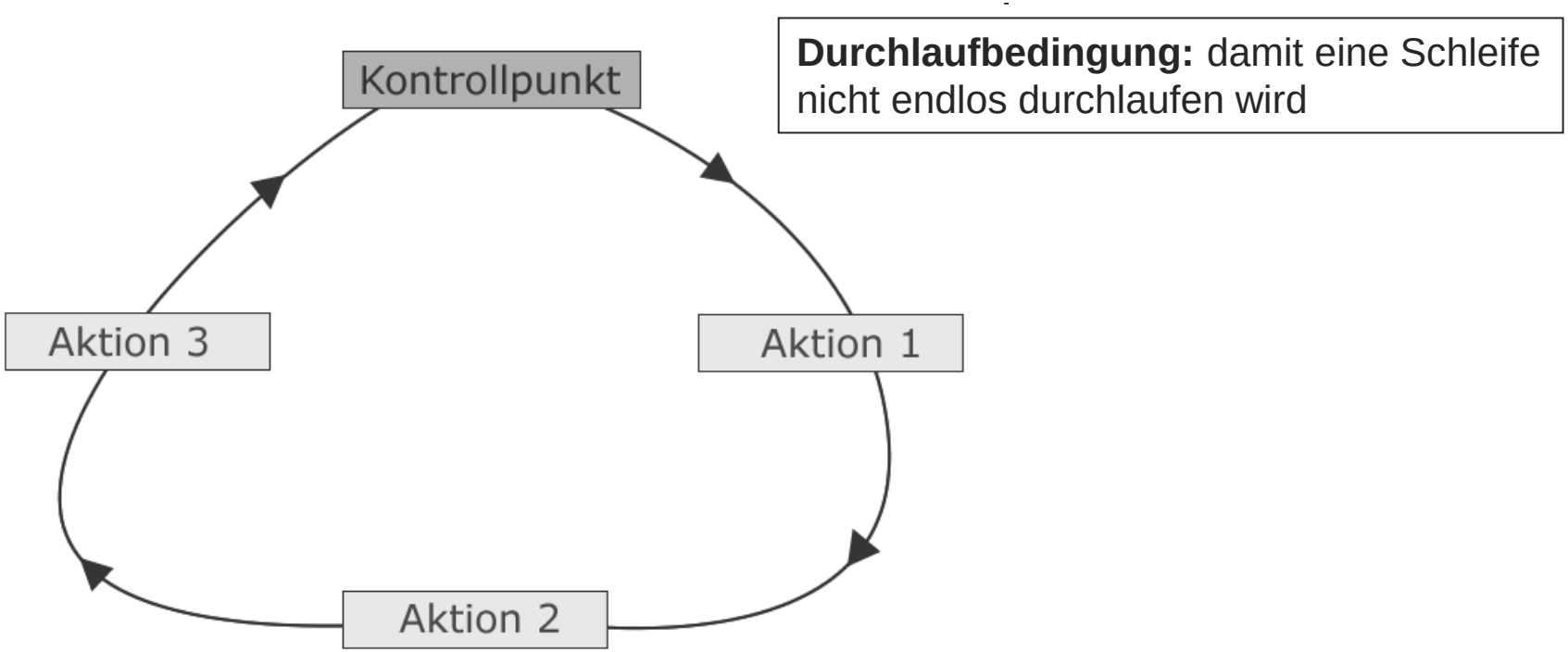
Inhalte

- ✓ Blöcke – Kontrollstrukturen für Sequenz
- ✓ Selektion
- ✓ Iteration

Sprunganweisungen

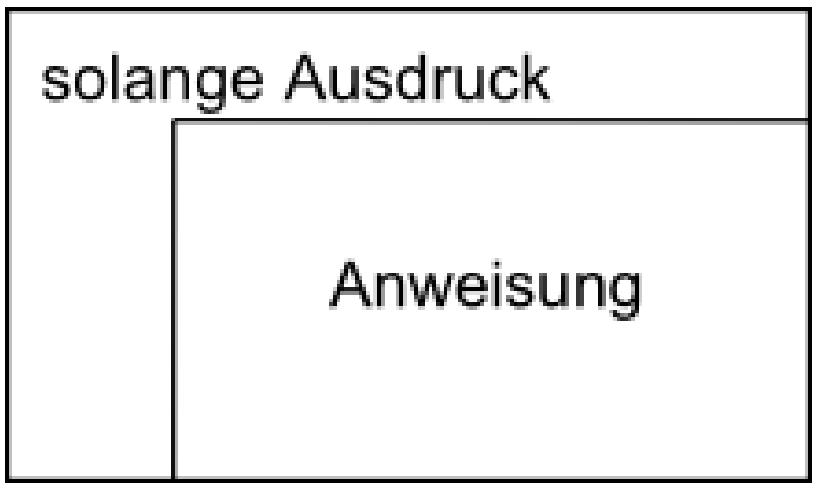
Iteration I

- Schleifen
  - werden verwendet, um Wiederholungen im Programm zu realisieren
  - auch Wiederholungsstrukturen oder Iterationen genannt
  - Programm läuft nicht von oben nach unten, sondern springt zurück und wiederholt einen Programmteil mehrmals



Iteration while

- Abweisende Schleife mit *while*
  - bei jedem Schleifendurchlauf wird der Ausdruck bewertet
  - erst wenn dieser Ausdruck nicht mehr wahr ist, wird die Schleife verlassen
  - es kann natürlich sein, dass der Ausdruck von Anfang an nicht wahr ist (dann wird die Schleife kein einziges Mal durchlaufen)
  - man spricht auch von einer abweisenden oder Kopf-gesteuerten Schleife



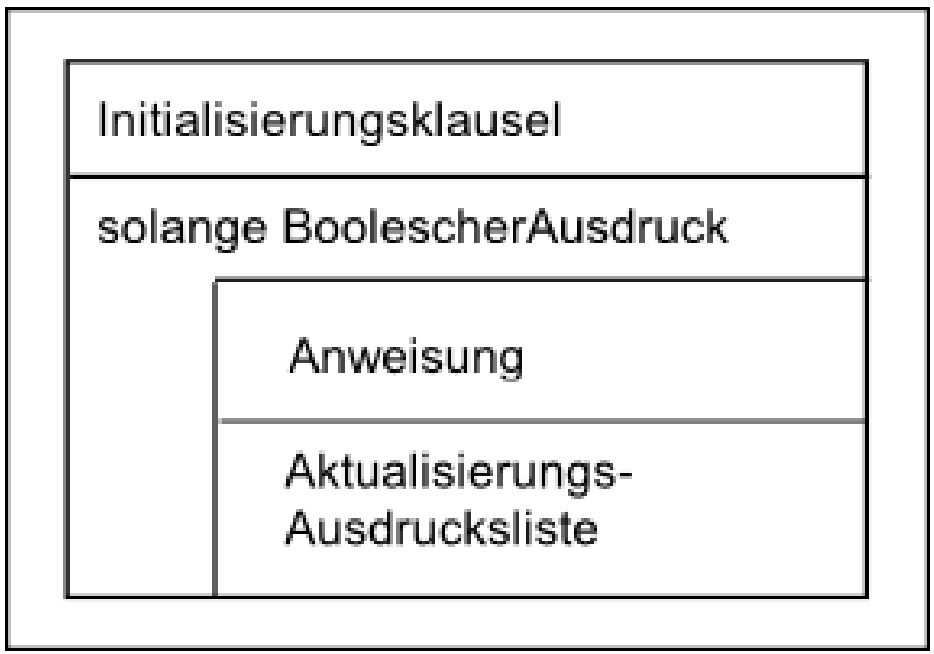
(Struktogramm der *while*-Schleife)

```
int i = 1;
while (i <= 100){
    System.out.println(i+". Zeile");
    i++;
}
```

Iteration for

- Abweisende Schleife mit *for*
  - *ebenfalls eine abweisenden bzw. Kopf-gesteuerte Schleife*
  - *Syntax*

    for(Initialisierungsklausel,BoolescherAusdruck,Aktualisierungs-Ausdruckliste)
    Anweisung




```
for(int i=1; i<=100; i++){
    System.out.println(i+". Zeile");
}
```

(Struktogramm der zur *for*-Anweisung äquivalenten *while*-Schleife)

Iteration for Verschachtelung

- for Schleife Verschachtelung
  - Schleifen können beliebig verschachtelt werden
  - Beispiel:
  - äußere Schleife: es sollen zehn Zeilen ausgegeben werden
  - innere Schleife: es sollen in jeder Zeile Sternchen ausgegeben werden, wobei die Anzahl der * der jeweiligen Zeilennummer entspricht (z.B. Zeile 2 hat zwei Sternchen)

```
int i, j;

// Schleife fuer die Zeilen
for(i=0; i<10; i++) {
   System.out.println("Zeile "i+1+":");

    // Schleife fuer die Spalten
    for(j=0; j<=i; j++) {
        System.out.println("*");
    }
}
```

```
Zeile 1: *
Zeile 2: **
Zeile 3: ***
Zeile 4: ****
Zeile 5: *****
Zeile 6: ******
Zeile 7: *******
Zeile 8: ********
Zeile 9: *********
Zeile 10: **********
```

Iteration for-each-schleife

- *For-each*-Schleife
  - seit dem JDK 5 dabei
  - besonders geeignet, um über Arrays bzw. Elemente von Collection-Klassen zu iterieren
  - zunächst Variable vom Typ eines Array-Elements definieren (im Beispiel: String element)
  - nach dem Doppelpunkt steht die Variable des zu durchlaufenden Arrays

```
public class ForEachTest{
   public static void main (String[] args){
      String[] testArray = new String[] {"Hallo", "for-each", "Schleife"};

      // Array mit Hilfe der erweiterten for-Schleife auslesen.
      for (String element : testArray){
         // Zugriff auf das Element des Arrays
         System.out.println (element);
      }
   }
}
```

Iteration Endlosschleife

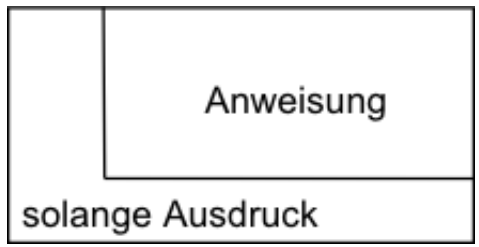
- Endlos-Schleife:

```
// Endlosschleife
for(; ;){
    …
}
```

oder

```
// Endlosschleife
while(true){
    …
}
```

- Annehmende Schleife mit *do-while*



(*Struktogramm der do-while*-Schleife)

```
public class DoWhileTest{
   public static void main (String[] args)
   {
      int i=1;
      do
      {
      System.out.println(„Geben Sie bitte Ihr Alter ein");
      alter = Tools.intEingabe();
      }
      while (alter<5 || alter>100);
   }
}
```

Übersicht

Inhalte

- ✓ Blöcke – Kontrollstrukturen für Sequenz
- ✓ Selektion
- ✓ Iteration
- ✓ Sprunganweisungen

Iteration break

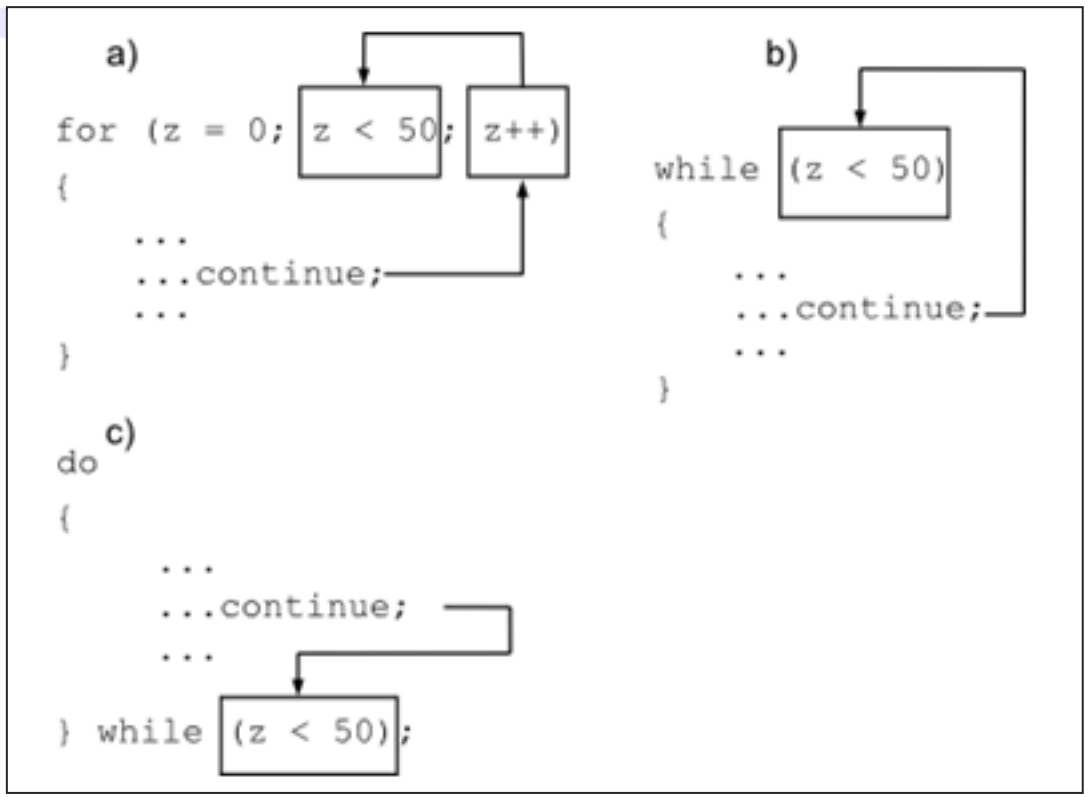
- *break*
  - mit dem Schlüsselwort break kann zu jeder Zeit eine Schleife oder eine switch-Anweisung verlassen werden
  - Beispiel:
  - Benutzer gibt Summanden zwischen 1 und 50 ein, um eine Gesamtsumme von 100 zu erreichen
  - Ist der eingegebene Summand nicht zw. 1 und 50 wird die Schleife sofort verlassen

```
int summand, summe=0;
do {
   System.out.println("Eingabe von Summand zwischen 1 und 50: ");
   summand = Tools.intEingabe();

   if(summand < 1 || summand > 50)
       break;

   summe += summand;
}while(summe < 100);

if(summe < 100)
    System.out.println("ungueltige Eingaben (100 wurde nicht erreicht)");
else
    System.out.println("Sie haben den Wert 100 erreicht.");
```

Sprunganweisungen II

- ***Continue***
  - mit dem Schlüsselwort continue kann direkt zum Kontrollpunkt gesprungen werden
  - der restliche Code im Schleifen-Block wird dann nicht mehr ausgeführt
  - gilt nur für die for-, while- und do-while-Schleife
- Anweisung dient zum Sprung in den nächsten Schleifendurchgang

```java
// Datei: Login2.java

import java.util.Scanner;

public class Login2
{
   public static void main (String[] args)
   {
      Scanner scanner = new Scanner (System.in);
      String eingabe = null;

      while (true)
      {
         System.out.print ("Bitte geben Sie Ihr Login ein: ");
         eingabe = scanner.next();
         if (!eingabe.equalsIgnoreCase ("Anja"))
         {
            System.out.println ("Falsche Eingabe!");
            continue;
         }
         System.out.println ("Anmeldevorgang erfolgreich!");
         break;
      }
   }
}
```



Aufgaben zu Schleifen I

Legen Sie für die folgenden Aufgaben entsprechende Klassen inkl. main-Methode in Eclipse an (z.B. Aufg_05_07.java für Aufgabe 05.07 etc.)

- **Aufg. 05.07**: Schreiben Sie ein Programm, das die ganzen Zahlen von 1 bis n miteinander multipliziert. Das Programm erwartet die Eingabe von n und berechnet das Produkt von 1*2*3*…*n. In der Mathematik heißt dieses Produkt n Fakultät, geschrieben n!.
  → 4! Entspricht also 1*2*3*4 = 24

- **Aufg. 05.08**: Schreiben Sie folgendes Programm: Es sollen 5 Vornamen und Nachnamen eingelesen werden. Pro Einlesevorgang wird ein Vorname mit Nachnamen getrennt durch ein Leerzeichen eingegeben. Speichern Sie die Namen in einem zweidimensionalen Array ab (String[5][2]). Splitten Sie den vollständigen Namen beim Leerzeichen und speichern Sie den Vornamen unter Index 0 und den Nachnamen unter Index 1 in jeder der 5 Zeilen des Arrays. Geben Sie danach mit Hilfe von einer for-each-Schleife alle vollständigen Namen aus, deren Nachname mindestens 5 Zeichen lang ist und deren Vorname den Buchstaben „e“ bzw. „E“ enthält.

Aufgaben zu Schleifen II

- **Aufg. 05.08 Simple**: Schreiben Sie folgendes Programm: Es sollen 5 Namen eingelesen werden. Speichern Sie die Namen in einem eindimensionalen Array ab (String[5]). Geben Sie danach mit Hilfe von einer for-each-Schleife alle Namen aus, die mindestens 5 Zeichen lang sind und die ein „e“ bzw. „E“ enthalten.