Javakurs für Anfänger

Einheit 07: Weitere Kontrollstrukturen

Stefan Langer
Lehrstuhl für Mobile und Verteilte Systeme

Heutige Agenda

1. Teil: Weitere Kontrollstrukturen

- Wiederholungsstrukturen (Schleifen)
  - `While-`,`do-`,`for`-Schleifen
- Sprunganweisungen
  - `break` und `continue`

2. Teil: Übungen

- Die Klasse Hund erweitern
- ZahlenDreieck
- Zahlenraten
  - Die Klasse Random für Zufallszahlen

Lernziele

- Weitere Kontrollstrukturen kennenlernen
- Umgang mit Kontrollstrukturen einüben
- Weitere Programme schreiben

Heutige Agenda

1. Teil: Weitere Kontrollstrukturen

- Wiederholungsstrukturen (Schleifen)
  - `While-`,`do-`,`for`-Schleifen
- Sprunganweisungen
  - `break` und `continue`

2. Teil: Übungen

- Die Klasse Hund erweitern
- ZahlenDreieck
- Zahlenraten
  - Die Klasse Random für Zufallszahlen

Lernziele

- Weitere Kontrollstrukturen kennenlernen
- Umgang mit Kontrollstrukturen einüben
- Weitere Programme schreiben

Überblick über Kontrollstrukturen

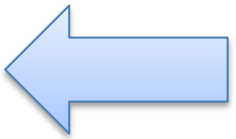
Ein Algorithmus lässt sich durch 3 Grundstrukturen beschreiben:

- Anweisungsfolge bzw. Sequenz
  - Schrittweise Abarbeitung von Befehlen von oben nach unten
- Auswahlstruktur bzw. Selektion
  - Ermöglicht die bedingte Ausführung von Anweisungen
  - Bsp.: `if, else, switch-case`
- Wiederholungsstruktur bzw. Iteration oder Schleife
  - Mehrmalige Ausführung der gleichen Anweisungen
  - Beispiele: `while-, do-, for-`Schleifen

Überblick über Kontrollstrukturen

Ein Algorithmus lässt sich durch 3 Grundstrukturen beschreiben:

- Anweisungsfolge bzw. Sequenz
  - Schrittweise Abarbeitung von Befehlen von oben nach unten
- Auswahlstruktur bzw. Selektion
  - Ermöglicht die bedingte Ausführung von Anweisungen
  - Bsp.: `if, else, switch-case`
- **Wiederholungsstruktur bzw. Iteration oder Schleife**
  - Mehrmalige Ausführung der gleichen Anweisungen
  - Beispiele: `while-, do-, for-`Schleifen



Heute

Motivation

- Bestimmte Anweisungsblöcke sollen mehrfach ausgeführt werden (ohne diese mehrfach schreiben zu müssen)
- Wiederholung soll von einer **Bedingung** abhängen

2 Arten:

- Abweisende bzw. kopfgesteuerte Schleifen
  - Prüft die Bedingung **vor** der ersten Ausführung eines Anweisungsblocks
    - Es kann sein, dass Anweisungsblock nie ausgeführt wird
    - Bsp.: `while`-Schleife, `for`-Schleife

- Nicht abweisende bzw. fußgesteuerte Schleifen
  - Durchläuft den Anweisungsblock auf jeden Fall 1 mal und prüft **danach** die Bedingung
    - Erneutes Ausführen des Anweisungsblocks, falls Bedingung true
    - Bsp.: `do`-Schleife

Wiederholungsstrukturen: Die `while`-Schleife

Die `while`-Schleife führt die Anweisungen in ihrem Block solange aus, bis die definierte Bedingung verletzt ist.

- Falls die Bedingung von Anfang an nicht erfüllt ist, werden die Anweisungen nie ausgeführt => abweisende Schleife

```
// Führe die folgenden Anweisungen
aus, bis die Bedingung verletzt ist

while(Bedingung){

   anweisung1;
   anweisungXY;

   update der Bedingung;
}
```

```
// Konkretes Beispiel
// Schreibt 10 mal „Hallo“
int i = 0;
while(i<10){

   System.out.println(„Hallo“);


   i++; //Erhöht i um 1 auch: i = i+1;
}
```

Wiederholungsstrukturen: Die do-Schleife

Die do-Schleife führt die Anweisungen in ihrem Block aus und wiederholt diese solange die Bedingung am Ende erfüllt ist.

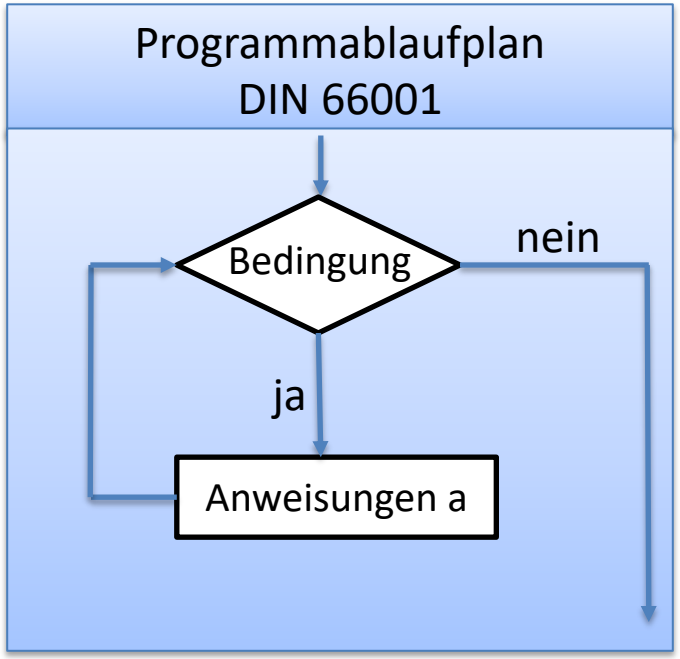
- Falls die Bedingung von Anfang an nicht erfüllt ist, werden die Anweisungen einmal ausgeführt => nicht abweisende Schleife

```
// Führe die folgenden Anweisungen
aus und wiederhole gem. der Bedingung

do{

   anweisung1;
   anweisungXY;

   update der Bedingung;

} while (Bedingung);
```

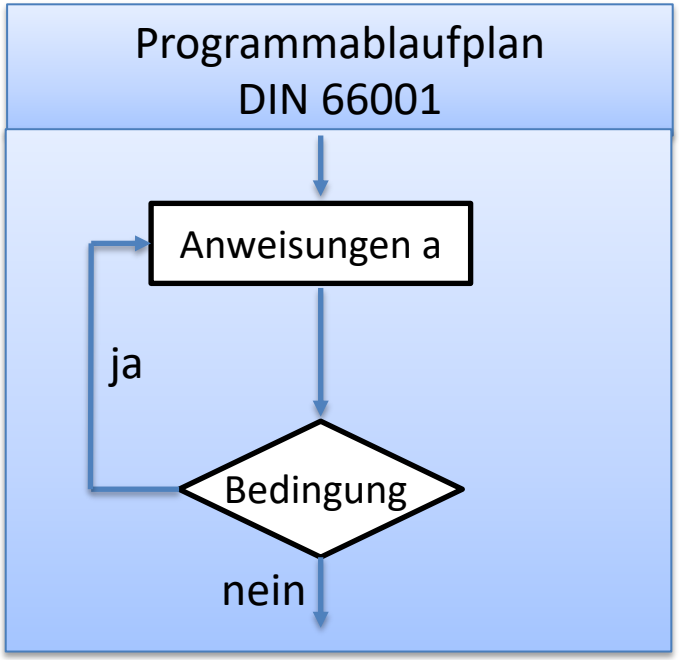
```
// Konkretes Beispiel
// Schreibt 1 mal „Hallo“
int i = 10;

do{

   System.out.println(„Hallo“);

   i++; //Erhöht i um 1 auch: i = i+1;

} while(i<10);
```

Unterschied zwischen `while` und `do`

Der Unterschied zwischen `while` und `do` Schleifen wird auch im jeweiligen Programmablaufplan sichtbar:



Abweisende Schleife: `while, for`



Nicht abweisende Schleife: `do`

Wiederholungsstrukturen: Die `for`-Schleife

Die `for`-Schleife ist wie schon `while` eine abweisende Schleife.

- Die Bedingung wird zu Beginn geprüft
- Zusätzlich wird durch die *Initialisierung* ein Anfangszustand definiert
- Und durch die *Aktualisierung* wird der Update-Schritt festgelegt, der nach jedem Schleifendurchlauf auszuführen ist

```
// Führe die folgenden Anweisungen aus, bis die
Bedingung verletzt ist

for (Initialisierung; Bedingung; Aktualisierung){

   anweisung1;
   anweisungXY;
}
```

```
// Konkretes Beispiel (wie vorher)
// Schreibt 10 mal „Hallo“

for (int i=0; i<10; i++){

   System.out.println(„Hallo“);

}
```

- Die `for`-Schleife bietet sich an, wenn bereits vor dem Eintritt in die Schleife die Iterationszahl feststeht!

Sprunganweisungen

Sprunganweisungen sollten selten und mit Vorsicht verwendet werden!

- Sind nicht unbedingt erforderlich, da durch `if`-Anweisungen ersetzbar
- Beispiele: `break`, `continue`

Die Anweisung `break`:

- Beendet eine `switch-`, `while-`, `do-` oder `for`-Anweisung, welche die `break`-Anweisung umgibt

Die Anweisung `continue`:

- Unterbricht den aktuellen Schleifendurchlauf einer `while-`, `do-` oder `for`-Schleife und springt an die Wiederholungsbedingung der unmittelbar umgebenden Schleife

Versuchen Sie nun die gelernten Wiederholungsstrukturen anzuwenden!

Aufgabe 1

Sie wollen in Ihrer implementierten Hundefarm bestimmen, wie oft ein Hund bellt

- Erweitern Sie also die Methode `bellen()` um einen Parameter `int anzahl`, der angibt, wie oft ein Hund hintereinander bellt
- Bsp.: Wenn Sie die Methode `bellen(3)` aufrufen, soll der Hund 3 Mal bellen
  - Hinweis: Verwenden Sie dazu eine passende Wiederholungsstruktur
- Lassen Sie nun ein paar Ihrer Hunde unterschiedlich oft bellen!

Aufgabe 2

Erstellen Sie in IntelliJ ein neues Projekt *Uebung07* und schreiben Sie ein Programm (Main-Methode) mit dem Namen *ZahlenDreieck*, welches folgende Ausgabe auf der Konsole ausgibt:

```
0
0 1
0 1 2
0 1 2 3
0 1 2 3 4
0 1 2 3 4 5
0 1 2 3 4 5 6
0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7 8
0 1 2 3 4 5 6 7 8 9
```

- Verwenden Sie dazu die `while`-Schleife
- Hinweise:
  - Es können auch mehrere `while`-Schleifen genutzt werden
  - Mit `System.out.print(String s)` wird **kein** Zeilenumbruch durchgeführt
  - Mit dem String „`\n`“ wird ein Zeilenumbruch erzwungen

Aufgabe 3

Schreiben Sie nun in Ihrem Projekt *Uebung07* ein weiteres Programm mit dem Namen *ZahlenRaten*, das folgende Aufgaben übernimmt:

- Ihr Programm denkt sich zunächst mit Hilfe der Klasse *`Random`* eine Zahl zwischen 0 und 10 aus
  - Die Klasse `Random` muss mittels `import java.util.Random` importiert werden
  - Ein Objekt der Klasse Random liefert über die Methode `nextInt(int zahl)` eine ganze Zahl zwischen [0,…,zahl[zurück
- Ihr Programm fordert nun den Nutzer immer wieder auf eine Zahl auf der Konsole einzugeben, bis der Nutzer die vom Programm ausgedachte Zahl korrekt erraten hat!
  - Gibt der Nutzer also eine falsche Zahl ein, meldet das Programm zurück, dass die eingegebene Zahl falsch ist und der Nutzer wird erneut aufgefordert eine Zahl einzugeben
  - Gibt der Nutzer hingegen die vom Programm ausgedachte Zahl ein, meldet das Programm „Herzlichen Glückwunsch“ und beendet die Ausführung

Verwenden Sie passende Kontrollstrukturen!