

SE1, Grundlegende Kontrollstrukturen in Java

Softwareentwicklung I – Wintersemester 2016/2017

1. Verzweigung (**if**)

Eine Verzweigung mit dem Schlüsselwort **if** prüft den booleschen Ausdruck in den runden Klammern und führt den direkt folgenden Block nur aus, wenn dieser Ausdruck **true** liefert. Eine if-Verzweigung kann keine oder beliebig viele **else if**-Blöcke und maximal einen **else**-Block haben. Sollte keiner der booleschen Ausdrücke **true** liefern, wird (wenn vorhanden) der **else**-Block ausgeführt.

Der boolesche Ausdruck kann z.B. ein Methodenaufruf einer Methode mit dem Rückgabetypp **boolean** oder ein Vergleich mit den Operatoren: **==**, **<=**, **>=**, **<**, **>**, **!=** sein.

Syntax (Pseudocode)	Beispiel
<pre>if (<boolescher Ausdruck>) { <Block 1> } else if (<boolescher Ausdruck>) { <Block 2> } else { <Block 3> }</pre>	<pre>int a = 4; String ausgabe; if (a == 5) { ausgabe = "1 ausgeführt. "; } else if (a <= 15) { ausgabe = "2 ausgeführt. "; } else if (a == 4) { ausgabe = "3 ausgeführt. "; } else { ausgabe = "4 ausgeführt. "; }</pre>
Bei if -Verzweigungen sind else if -Blöcke und der else -Block optional.	<p>In diesem Beispiel hat die Variable <i>ausgabe</i> den Wert "2 ausgeführt. ".</p> <p>Zwar trifft auch der Ausdruck des zweiten else if-Blocks zu, doch es wird immer der erste Block, bei dem die Prüfung zutrifft, ausgeführt.</p>

2. Mehrfachverzweigung (switch)

Die Mehrfachverzweigung mit dem Schlüsselwort **switch** führt aus einer Reihe an Blöcken mit zugeordnetem **case**-Label denjenigen aus, dessen zugewiesener konstanter **case**-Labelwert gleich der dem **int**-Wert aus den runden Klammern ist. Eine Mehrfachverzweigung funktioniert mit allen primitiven Datentypen (Java 6), deren Wertebereich kleiner als der von **int** ist, also **byte**, **short** und **char**.

Ein **case**-Label beginnt mit dem Schlüsselwort **case** gefolgt von einem Labelwert und einem Doppelpunkt **:** und endet mit dem auszuführenden Block. Der Labelwert wird in der Regel durch einen Zahlenliteral dargestellt, beispielsweise **2**. Er kann aber auch durch eine konstante Variable repräsentiert sein, die mit **final** deklariert wurde.

Switch-Mehrfachverzweigungen haben die Eigenschaft, dass sie alle Blöcke ausführen, die unter dem ersten zutreffenden **case**-Label stehen, bis die Verzweigung endet oder mit dem Schlüsselwort **break** abgebrochen wird.

Sollte kein **case**-Label zutreffen, wird, wenn vorhanden, das **default**-Label ausgeführt.

Syntax (Pseudocode)	Beispiele
<pre>switch (<int-Wert>) { case: <Labelwert>: <Block1> break; default: <Block2> }</pre>	<pre>char buchstabe = 'a'; String ausgabe; switch (buchstabe) { case 'a': ausgabe = "1 ausgeführt. "; break; case 'u': ausgabe = "2 ausgeführt. "; case 'e': ausgabe = "3 ausgeführt. "; break; }</pre>
<p>Eine switch-Mehrfachverzweigung kann beliebig viele case-Label, jedoch maximal ein default-Label haben. In jedem Block kann ein break stehen.</p>	<p>In diesem Beispiel hat die Variable <i>ausgabe</i> nach der Ausführung den Wert "1 ausgeführt.". Das erste Label mit dem Wert 'a' trifft zu. Die break-Anweisung sorgt dafür, dass die Mehrfachverzweigung an dieser Stelle verlassen wird.</p>
	<pre>int zahl = 15; String ausgabe; switch (buchstabe) { case 1: ausgabe = "1 ausgeführt. "; break; case 2: ausgabe = "2 ausgeführt. "; case 3: ausgabe = "3 ausgeführt. "; break; default: ausgabe = "4 ausgeführt. "; }</pre>
	<p>In diesem Beispiel hat die Variable <i>ausgabe</i> nach der Ausführung den Wert "4 ausgeführt.". Da keines der case-Labels zutrifft wird, wenn vorhanden, der Block des default-Labels ausgeführt.</p>

3. Präfix- und Postfixoperatoren (++ , --)

Präfix- bzw. Postfixoperatoren sind das Doppelminus -- und das Doppelplus ++ sind Operatoren in Java, die den Wert einer Variable um eins erhöhen bzw. verringern.

Dabei wird der Präfixoperator (aus lat. *præ*, „vor“) wie z.B. ++i vor einer Zuweisung ausgeführt, während es bei dem Postfixoperator (aus lat. *post*, „nach“) wie z.B. i++ erst danach ausgeführt wird. Allgemein gesagt, geschieht beim Präfix- bzw. Postfixoperator der Seiteneffekt vor bzw. nach Auswertung der Variablen. Wobei Seiteneffekt die Erhöhung bzw. Verringerung des Wertes ist.

Folgende Beispiele sollten das Verhalten verdeutlichen.

Prefix-Operator ++i		Postfix-Operator i++	
<code>int i = 0;</code>	<code>i = 0</code>	<code>int i = 0;</code>	<code>i = 0</code>
<code>int j = ++i;</code>	<code>j = 1 i = 1</code>	<code>int j = i++;</code>	<code>j = 0 i = 1</code>
Prefix-Operator --i		Postfix-Operator i--	
<code>int i = 1;</code>	<code>i = 1</code>	<code>int i = 1;</code>	<code>i = 1</code>
<code>int j = --i;</code>	<code>j = 0 i = 0</code>	<code>int j = i--;</code>	<code>j = 1 i = 0</code>

4. Verhalten des Zuweisungsoperators (=)

Die Zuweisung (engl. *assignment*) wird in Java mit = realisiert. (Hier kommt es häufig zu Verwechslungen mit dem booleschen Vergleichsoperator ==.) Zuweisungen sind rechts-assoziativ. Das bedeutet, sie werden von rechts nach links ausgewertet. Ein Beispiel: `a=b=c;` bedeutet `a=(b=c)`; . Es wird der Wert von c auf b und dieser Wert auf a zugewiesen. Einige weitere Beispiele:

<code>int i = 0;</code>	Variablendeklaration mit Zuweisung eines Literals.
<code>i = 3;</code>	Zuweisung eines Wertes, der alte Wert wird überschrieben.
<code>int j = i++;</code>	Variablendeklaration mit Zuweisung des Wertes einer anderen Variablen und anschließender Erhöhung des Wertes der anderen Variablen. (Siehe 3: Präfix- und Postfixoperatoren (++ , --))

Neben den einfachen Zuweisungen gibt es weitere Zuweisungsausdrücke. Folgende Zuweisungsbeispiele haben letztendlich alle den gleichen Effekt. Annahme für jede Zeile: `int i = 0;`

Anweisung	Bezeichnung	Wert von i nach Ausführung
<code>i = 1;</code>	Normale Zuweisung eines Literals	<code>i = 1</code>
<code>i = i + 1;</code>	Normale Zuweisung, zuerst wird die rechte Seite ausgewertet, dann zugewiesen	<code>i = 1</code>
<code>i = i++;</code>	Postfixoperator	<code>i = 1</code>
<code>i = ++i;</code>	Präfixoperator	<code>i = 1</code>
<code>i += 1;</code>	Kombinierter Zuweisungsoperator, möglich für alle binären Operatoren (<code>+=</code> , <code>-=</code> , <code>*=</code> , <code>/=</code> , <code>%=</code>)	<code>i = 1</code>