

Einführung in die Programmierung  
WS 2018/19

Übungsblatt 7: Typecast, Schleifen

Aufgabe 7-1 *Typecasting*

- (a) Geben Sie für jeden der folgenden Java-Ausdrücke an, ob er syntaktisch korrekt ist. Falls ja, geben Sie außerdem den Wert und den Typ des Ausdrucks an. Begründen Sie jeweils Ihre Antwort. Hierbei seien die Variablen `x`, `y` und `z` wie folgt initialisiert: `int x = 9, y = 12, z = 16;`

- |   |                                 |
|---|---------------------------------|
| (i) <code>10/4</code> .                         | (v) <code>1    0</code>         |
| (ii) <code>10/4</code>                          | (vi) <code>(byte)(127+1)</code> |
| (iii) <code>x == y ? x &lt; y : y &gt; z</code> | (vii) <code>'x' + y + z</code>  |
| (iv) <code>false &amp;&amp; true</code>         | (viii) <code>x + y + "z"</code> |

- (b) Ausdrücke in Java, die aus mehr als nur einem elementaren Datentyp zusammengesetzt sind, werden je nach Verwendung durch implizite oder explizite Typkonversion ausgewertet. Ersetzen Sie für jeden der folgenden Java-Ausdrücke schrittweise alle impliziten Typecasts durch Explizite. Geben Sie dann den Wert und den Typ des Ausdrucks an. Falls der Ausdruck dann nicht typkorrekt ist, begründen Sie den Fehler.

Hierbei seien die Variablen `x` und `y` wie folgt initialisiert: `int x = 9, y = 12;`

Bsp: `5L * 3.0`  $\Leftrightarrow ((double)(5L) * 3.0) \Leftrightarrow (5.0 * 3.0) \Leftrightarrow (15.0)$ , vom Typ `double`.

- |   |  |
|---|--|
| (i) <code>(x &gt; y) &lt; true</code>             | (iv) <code>(int) ((char) 65536)</code> |
| (ii) <code>(long)(3F/4.0) == (int)(4/5.0D)</code> | (v) <code>1L + (float) 3D</code>       |
| (iii) <code>(long)3F/4.0 == (int)4/5.0D</code>    | (vi) <code>'4' + '7'</code>            |

- (c) Gegeben sind nun folgende Variablendeklarationen:

<code>long l1 = 126L;</code>	<code>byte b1 = 100;</code>	<code>char c1 = '!';</code>
<code>int i1 = 6;</code>	<code>float f1 = 2.0f;</code>	<code>char c2 = 'c';</code>
<code>short s1 = 8;</code>	<code>double d1 = 0.125;</code>	<code>String str1 = "789";</code>

Ersetzen Sie alle impliziten Typecasts durch Explizite. Geben Sie dann an welchen Typ und welchen Wert die folgenden Ausdrücke haben.

- |   |                                  |
|---|----------------------------------|
| (i) <code>(f1 * 3.0f) / (b1 * i1)</code>    | (iv) <code>f1 / d1 + l1</code>   |
| (ii) <code>"1 + 2 + 3 = " + (i1 - 2)</code> | (v) <code>c2 / c1</code>         |
| (iii) <code>"l1 + i1 = " + 126 + 6</code>   | (vi) <code>i1 + str1 + s1</code> |

## Hinweise:

- *Zeichen* vom Typ `char` werden in zusammengesetzten arithmetischen Ausdrücken automatisch in den Typ `int` konvertiert (entsprechend ihrem ASCII-Code).
- In Java werden *Zeichenketten* mit doppelten Anführungszeichen umrahmt und sind vom Typ `String`. Die Konkatenation von Zeichenketten erfolgt durch den `+`-Operator (der Operator ist also *überladen*). Ist wenigstens einer der beiden Operanden in `a + b` ein `String`, so wird der gesamte Ausdruck als String-Konkatenation ausgeführt.
- Schauen Sie für diese Aufgabe die ASCII-Zeichen im Internet nach. Für die Klausur wird die ASCII-Tabelle angegeben, falls Sie sie benötigen.

## Aufgabe 7-2      Terminierung von Schleifen

Welche der folgenden Schleifen terminieren, welche terminieren nicht? Begründen Sie Ihre Antworten kurz.

(a) 

```
int i = 0;
int j = 0;
do {
    j = j++;
    i = j + i;
} while(i < 200);
```

(c) 

```
char i = 'a';
short j = 0;
while (i != j) {
    i++;
    j = (short) (i+i);
}
```

(b) 

```
double i = 0.5;
double j = 0;
while(j < 1) {
    j += i;
    i *= 0.5;
}
```

(d) 

```
long i = 26L;
long j = 24L;
for (long x = 0L; x < 1000L; x++) {
    i = i / 12 + 23 * (--x);
    j = (x--) + j + 5;
}
```

### Aufgabe 7-3 *Kontrollstrukturen*

Im Folgenden sind die Programmausschnitte (a), (b) und (c) gegeben. Betrachten Sie alle vorkommenden Variablen als deklariert und initialisiert.

Überlegen Sie, was in den Schleifen passiert. Wie können Sie den gleichen Effekt ohne Verwendung von Wiederholungsanweisungen erzielen? Geben Sie ggfs. ein entsprechendes Code-Fragment an oder begründen Sie, warum dies nicht möglich ist.

```
(a) int i = 0;
    int j = 42;
    while(i < j | i > j){
        i++;
        j--;
    }

(b) int i = 0;
    int j = 42;
    while(i < j | i > j){
        i++;
        j--;
    }

(c) int i = 0;
    int j = 42;
    while(i < j | i > j)
        i++;
        j--;
```

- (d) Schreiben Sie für die folgende **do-while**-Schleife jeweils eine äquivalente **while**- und **for**-Schleife. **n** sei eine deklarierte **int**-Variable, die mit einem beliebigen Wert größer 0 initialisiert wurde.

```
double x = 0.;
double y = 1.;
int i = 1;
do {
    x = x + y;
    y = y / i;
} while(i++ <= n);
```

### Aufgabe 7-4 *Switch (optional)*

Geben Sie für drei gegebenen ganzen Zahlen **x**, **y** und **z** diese Zahlen in aufsteigender Reihenfolge an. Das folgende Codefragment sollen Sie dafür als Grundgerüst verwenden. Implementieren Sie an der mit *TODO* markierten Stelle ein **Switch**-Konstrukt, sodass die Ausgabemethode aufgerufen wird und stets einen gültigen Ausdruck liefert. Ändern Sie dabei nicht die Ausgabemethode. Verwenden Sie keine **if**-Verzweigung.

Sie dürfen die Datei **Switch.java** zum Testen verwenden. Schreiben Sie dort Ihre Lösung an die betreffende Stelle und geben Sie die Datei ab.

```
public static void sortiere(int x, int y, int z){
    int n = 0;
    n += (x > y ? 1 : 0);
    n += (x > z ? 2 : 0);
    n += (y > z ? 4 : 0);
    // TODO
}

public static void ausgabe(int a, int b, int c) {
    if(a > b || a > c || b > c)
        System.out.println("Da stimmt etwas nicht!");
    System.out.println("Es gilt: " + a + " <= " + b + " <= " + c);
}
```