Ludwig-Maximilians-Universität München WS 2018/19
Institut für Informatik Übungsblatt 3
Prof. Dr. D. Sabel, Dr. P. Wendler

Übungen zu Einführung in die Informatik: Programmierung und Software-Entwicklung:

Lösungsvorschlag

Aufgabe 3-1 Überprüfen von Ausdrücken *Präsenz*

In dieser Aufgabe sollen Sie Ausdrücke auf Korrektheit untersuchen. Als Grundlage für Ausdrücke in Java verwenden wir folgende vereinfachte EBNF-Grammatik. Die Nichtterminale `Variable` und `Value` sind wie in der Vorlesung definiert.

```
Expression = Variable |
             Value |
             Expression BinOp Expression |
             UnOp Expression |
             "(" Expression ")"

BinOp = "&" | "|" | "&&" | "||" | "+" | "-" | "*" | "/" | "%" |
        "==" | "!=" | ">" | ">=" | "<" | "<="
UnOp = "!" | "(" Type ")" | "-" | "+"
```

Gegeben seien folgende Ausdrücke:

1. `1 + 2`
2. `1 &&`
3. `(3 == 7) && (0 < 1)`
4. `3 == 7 && 0 < 1`
5. `3 == (7 && 0) < 1`
6. `3 != 3 | !false`
7. `* 1 == 0`
8. `-1 == 0`
9. `(1 < 2) < 3`
10. `true + false`

Geben Sie für jeden dieser Ausdrücke an, ob er syntaktisch korrekt ist oder nicht (mit Begründung) und ob er typkorrekt ist oder nicht (mit Begründung). Für typkorrekte Ausdrücke ist außerdem deren Typ anzugeben.

Lösung:

1. `1 + 2` ist syntaktisch korrekt und typkorrekt und vom Typ `int`.
2. `1 &&` ist syntaktisch nicht korrekt, da `&&` eine zweistellige Operation ist, die hier nur einstellig verwendet wird.
3. `(3 == 7) && (0 < 1)` ist syntaktisch korrekt und typkorrekt und vom Typ `boolean`.
4. `3 == 7 && 0 < 1` ist syntaktisch korrekt und typkorrekt und vom Typ `boolean`, denn: Nach vollständiger Klammerung erhält man `(3 == 7) && (0 < 1)` wie in 3.
5. `3 == (7 && 0) < 1` ist nicht typkorrekt, da `&&` eine boolesche Operation ist, die hier auf Ausdrücke vom Typ `int` angewendet wird.
6. `3 != 3 | !false`: Nach vollständiger Klammerung erhält man `(3 != 3) | (!false)`. Dieser Ausdruck ist syntaktisch korrekt und typkorrekt und vom Typ `boolean`.
7. `* 1 == 0` ist syntaktisch nicht korrekt, da `*` eine zweistellige Operation ist, die hier nur einstellig verwendet wird.
8. `-1 == 0`: Nach vollständiger Klammerung erhält man `(-1) == 0`. Dieser Ausdruck ist syntaktisch korrekt und typkorrekt und vom Typ `boolean`.
9. `(1 < 2) < 3` ist nicht typkorrekt, da der erste Teil des Ausdruck `(1 < 2)` den Typ `boolean` hat, der zweite Teil `3` den Typ `int` und `<` eine Operation ist, die auf beiden Argumenten einen numerischen Typ erwartet.
10. `true + false` ist nicht typkorrekt, da `+` eine zweistellige Operation mit numerischen Argumenttypen ist, die hier auf Ausdrücke vom Typ `boolean` angewendet wird.

Aufgabe 3-2 **Überprüfen von Ausdrücken** *Hausaufgabe*

In dieser Aufgabe sollen Sie Ausdrücke auf Korrektheit untersuchen. Als Grundlage für Ausdrücke in Java verwenden wir die in Aufgabe 3-1 gegebene EBNF-Grammatik.

Gegeben seien folgende Ausdrücke:

1. `!(3 > 5)`
2. `!3 < 5`
3. `5 ! <= 5`
4. `14 < 3 || 1/5 == 4567`
5. `false && (false || true)`
6. `false & -1`
7. `5 < 7 == !`

Geben Sie für jeden dieser Ausdrücke an, ob er syntaktisch korrekt ist oder nicht (mit Begründung) und ob er typkorrekt ist oder nicht (mit Begründung). Für typkorrekte Ausdrücke ist außerdem deren Typ anzugeben.

Lösung:

1. `!(3 > 5)` ist syntaktisch korrekt und typkorrekt und vom Typ `boolean`.
2. `!3 < 5`: Nach vollständiger Klammerung erhält man `(!3) < 5`. Dieser Ausdruck ist nicht typkorrekt, da `!` eine boolesche Operation ist, die hier auf einen Ausdruck vom Typ `int` angewendet wird.
3. `5 ! <= 5` ist syntaktisch nicht korrekt, da nicht zwei Operationen (hier `!` und `<=`) direkt hintereinander verwendet werden dürfen.
4. `14 < 3 || 1/5 == 4567`: Nach vollständiger Klammerung erhält man `(14 < 3) || ((1/5) == 4567)`. Dieser Ausdruck ist syntaktisch korrekt und typkorrekt und vom Typ `boolean`.
5. `false && (false || true)` ist syntaktisch korrekt und typkorrekt und vom Typ `boolean`.
6. `false & -1`: Nach vollständiger Klammerung erhält man `false & (-1)`. Dieser Ausdruck ist nicht typkorrekt, da `&` eine boolesche Operation ist, die hier auf einen Ausdruck vom Typ `int` im zweiten Argument angewendet wird.
7. `5 < 7 == !` ist syntaktisch nicht korrekt, da nicht zwei Operationen (hier `==` und `!`) direkt hintereinander verwendet werden dürfen.

Aufgabe 3-3 **Auswertung von Ausdrücken in Java** *Präsenz*

In dieser Aufgabe sollen Sie sich mit der Auswertung von Ausdrücken in Java vertraut machen.

a) Gegeben seien folgende Variablendeklarationen:

```
1 int zahl = 17;
2 int teiler = 0;
```

Welcher Zustand σ wird durch diese Deklarationen beschrieben?

Lösung:
$\sigma = [(\texttt{zahl}, 17), (\texttt{teiler}, 0)]$

b) Werten Sie folgende Ausdrücke bezüglich des Zustands σ aus Teilaufgabe a) aus:

```
1 1-1-1-1-zahl
2 1-1-1-1*zahl
3 zahl == teiler
4 teiler != 0 & zahl/teiler > 1
5 teiler != 0 && zahl/teiler > 1
```

Lösung:

Vollständig geklammert:
`(((1-1)-1)-1)-zahl` $=_\sigma$
`((0-1)-1)-zahl` $=_\sigma$
`(-1-1)-zahl` $=_\sigma$
`-2-zahl` $=_\sigma$
`-2-17` $=_\sigma$
`-19`

Vollständig geklammert:
`((1-1)-1)-(1*zahl)` $=_\sigma$
`(0-1)-(1*zahl)` $=_\sigma$
`-1-(1*zahl)` $=_\sigma$
`-1-(1*17)` $=_\sigma$
`-1-17` $=_\sigma$
`-18`

Schon vollständig geklammert:
`zahl == teiler` $=_\sigma$
`17 == 0` $=_\sigma$
`false`

Vollständig geklammert:
`(teiler != 0) & ((zahl/teiler) > 1)` $=_\sigma$
`(0 != 0) & ((zahl/teiler) > 1)` $=_\sigma$
`(0 != 0) & ((zahl/teiler) > 1)` $=_\sigma$
`false & ((zahl/teiler) > 1)` $=_\sigma$ (wegen striktem "und")
`false & ((17/0) > 1)` $=_\sigma$
$\longrightarrow$ Laufzeitfehler

Vollständig geklammert:
`(teiler != 0) && ((zahl/teiler) > 1)` $=_\sigma$
`(0 != 0) && ((zahl/teiler) > 1)` $=_\sigma$
`false && ((zahl/teiler) > 1)` $=_\sigma$ (wegen sequentiellem "und")
`false`

c) Schreiben Sie ein Java-Programm, das die Ausdrücke aus Teilaufgabe b) auswertet und das Ergebnis am Bildschirm zeigt.

Lösung:

```
1  public class Calculation {
2    public static void main(String[] args){
3      int zahl = 17;
4      int teiler = 0;
5
6      int calculation1 = 1 - 1 - 1 - 1 - zahl;
7      System.out.println("Ergebnis Berechnung 1: " + calculation1);
8
9      int calculation2 = 1 - 1 - 1 - 1 * zahl;
10     System.out.println("Ergebnis Berechnung 2: " + calculation2);
11
12     boolean test1 = zahl == teiler;
13     System.out.println("Ergebnis Test 1: " + test1);
14
15     boolean test3 = teiler != 0 && zahl / teiler > 1;
16     System.out.println("Ergebnis Test 3: " + test3);
17
18     boolean test2 =  teiler != 0 & zahl/teiler > 1;
19     System.out.println("Ergebnis Test 2: " + test2);
20   }
21 }
```

Aufgabe 3-4 **Auswertung von Ausdrücken in Java** *Hausaufgabe*

In dieser Aufgabe sollen Sie sich mit der Auswertung von Ausdrücken in Java vertraut machen.

a) Gegeben seien folgende Variablendeklarationen:

```
1 int tday = 26; //today
2 int tmonth = 10; //this month
3 int bday = 27; //my Grandma's birthday is on the 27th
4 int bmonth = 10; //my Grandma's birthday is in October
```

Welcher Zustand σ wird durch diese Deklarationen beschrieben?

Lösung:
$\sigma = [(\texttt{tday}, 26), (\texttt{tmonth}, 10), (\texttt{bday}, 27), (\texttt{bmonth}, 10)]$

b) Werten Sie folgende Ausdrücke bezüglich des Zustands σ aus Teilaufgabe a) aus:

```
1 bday - tday
2 bday - tday / 30
3 tday == bday && tmonth == bmonth
```

Lösung:

`bday - tday` $=_\sigma$
`27 - 26` $=_\sigma$
`1`

Vollständig geklammert:
`bday - (tday / 30)` $=_\sigma$
`27 - (tday / 30)` $=_\sigma$
`27 - (26 / 30)` $=_\sigma$
`27 - 0` $=_\sigma$
`27`

Vollständig geklammert:
`(tday == bday) && (tmonth == bmonth)` $=_\sigma$
`(26 == 27) && (tmonth == bmonth)` $=_\sigma$
`false && (tmonth == bmonth)` $=_\sigma$
`false`

c) Schreiben Sie ein Java-Programm, das die Ausdrücke aus Teilaufgabe b) auswertet und das Ergebnis am Bildschirm zeigt.

Lösung:

```
1  public class Birthday {
2    public static void main(String[] args){
3      int tday = 26; //today
4      int tmonth = 10; //this month
5      int bday = 27; //my Grandma's birthday is on October 27th
6      int bmonth = 10;
7
8      int daysleft = bday - tday;
9      System.out.println("There are still " + daysleft +
10     " days left until Grandma's birthday.");
```

```
11
12      int monthsleft = bday - tday / 30; //semantisch nicht korrekt
13      System.out.println("There are still " + monthsleft +
14      " months left until Grandma's birthday.");
15
16      boolean happybirthday = tday == bday && tmonth == bmonth;
17      System.out.println("Today is Grandma's birthday is " +
18      happybirthday + ".");
19    }
20 }
```

Besprechung der Präsenzaufgaben in den Übungen ab 02.11.2018. Abgabe der Hausaufgaben bis Mittwoch, 14.11.2018, 14:00 Uhr über UniworX (siehe Begrüßungsfolien).

- *Der Java-Code in ihrer Abgabe muss als separate* `.java`*-Datei abgegeben werden. Wir benötigen* ***nicht*** *Ihre* `.class`*-Dateien. Öffnen Sie dazu im Explorer den Ordner, den Sie als Eclipse-Workspace ausgewählt haben und navigieren Sie in den entsprechenden Unterordner anhand ihrer Projektstruktur.*
- *Kopieren Sie alle Dateien Ihrer Abgabe in einen eigenen Ordner (für jedes Übungsblatt ein eigener Ordner) und geben Sie den gesamten Ordner als ZIP-Archiv ab. In dem Ordner dürfen nur Dateien mit der Endung* `.java`*,* `.pdf`*,* `.jpg` *oder* `.txt` *enthalten sein.* ***Word-Dokumente werden nicht korrigiert!*** *Geben Sie keine Screenshots ab!*
- *Unter Windows kann ein ZIP-Archiv wie folgt erstellt werden: rechter Mausklick auf den Ordner, Auswahl von* `Senden an -> ZIP-komprimierter Ordner`*. Unter Mac OS hingehen: rechter Mausklick auf den Ordner, Auswahl von* `Komprimieren/Compress`*.*