Ludwig-Maximilians-Universität München
Institut für Informatik
Prof. Dr. Peer Kröger
Michael Fromm, Florian Richter

Einführung in die Programmierung

WS 2018/19

Übungsblatt 5: Ausdrücke, Substitution, Rekursion

Besprechung: 26.11 - 30.11.2018

Aufgabe 5-1 *Ausdrücke*

Geben Sie für jedes der folgenden Literale an, ob es ein syntaktisch korrekter Java-Ausdruck ist. Falls ja, geben Sie außerdem den Wert und den Typ des Ausdrucks an. Falls nein, geben Sie eine kurze Begründung an, warum der Ausdruck fehlerhaft ist.

(a) `- -1`

(b) `-(-1)`

(c) `-+2`

(d) `--2`

(e) `'FALSE'`

(f) `17*11(-16)`

(g) `(17)*(11)-16`

(h) `5%3=1`

(i) `8%3==2`

(j) `18%21%7%5`

(k) `- - - 2 + - + 5`

(l) `TRUE`

(m) `3d`

(n) `0x1a`

(o) `0x2g`

(p) `057`

(q) `41L`

(r) `2/0`

Lösungsvorschlag:

(a) `- -1`
syntaktisch korrekt, Wert `1` vom Typ `int`

(b) `-(-1)`
syntaktisch korrekt, Wert `1` vom Typ `int`

(c) `-+2`
syntaktisch korrekt, Wert `-2` vom Typ `int`

(d) `--2`
syntaktisch inkorrekt, der Dekrementoperator `--` darf nur auf eine Variable angewandt werden

(e) `'FALSE'`
syntaktisch inkorrekt, gemeint ist wohl die Zeichenkette `"FALSE"`

(f) `17*11(-16)`
syntaktisch inkorrekt, zwischen `11` und `(-16)` fehlt ein Operator

(g) `(17)*(11)-16`
syntaktisch korrekt, Wert `171` vom Typ `int`

Lösungsvorschlag:

h) `5%3=1`
syntaktisch inkorrekt, der Zuweisungsoperator `=` verlangt links eine Variable.
Achtung, Verwechslungsgefahr mit dem Vergleichsoperator:

	Java
Zuweisungsoperator	`=`
Vergleichsoperator	`==`

i) `8%3==2`
syntaktisch korrekt, Wert `true` vom Typ `boolean`

j) `18%21%7%5`
syntaktisch korrekt, Wert `4` vom Typ `int` (linksassoziativ, bei Rechtsassozitivität wäre der Wert `1`)

k) `- - - 2 + - + 5`
syntaktisch korrekt, Wert `-7` vom Typ `int`

l) `TRUE`
syntaktisch inkorrekt, gemeint ist wohl der Boole'sche Wert `true`

m) `3d`
syntaktisch korrekt, Wert `3.0` vom Typ `double`

n) `0x1a`
syntaktisch korrekt, Wert `26` vom Typ `int` (Hexadezimalform)

o) `0x2g`
syntaktisch inkorrekt, hexadezimale Literale dürfen nur aus den Buchstaben `a/A` bis `f/F` bestehen

p) `057`
syntaktisch korrekt, Wert `47` vom Typ `int` (Oktalform)

q) `41L`
syntaktisch korrekt, Wert `41` vom Typ `long`

r) `2/0`
syntaktisch korrekt, kein Wert, sondern ArithmeticException

Aufgabe 5-2 *Rekursion und Methoden in Java I*

(a) Wir benötigen später eine Methode, die die Quadratwurzel einer Gleitkommazahl x berechnet. Für diese Aufgabe darf keine andere Klasse (insbesondere nicht `Math`) benutzt werden. Um die Wurzel anzunähern, bedienen wir uns einem Annäherungsverfahren aus der Numerik, das auch als Heronverfahren oder babylonisches Wurzelziehen bekannt ist. Die induktiv definierte Folge

$$x_0 = \frac{x+1}{2}$$

$$x_n = \frac{1}{2}\left(x_{n-1} + \frac{x}{x_{n-1}}\right)$$

konvergiert gegen den Wert $\sqrt{x}$. Es ist hier nicht nötig, dass sie verstehen, warum dieses Verfahren korrekt arbeitet. Die rekursive Funktion, die die Wurzel berechnet, lässt sich damit wie folgt als Pseudocode programmieren:

```
function:  quadratwurzel(Reelle Zahl x, Ganze Zahl n)
 output:   Quadratwurzel von x
 pre:      x >= 0, n >= 0
 body:
  Wenn n == 0
   Dann (x + 1)/2
   Sonst 0.5*(quadratwurzel(x,n-1)+x/quadratwurzel(x,n-1))
```

Führen Sie formal einen Funktionsaufruf dieser Funktion mit der Variablenbelegung $\sigma = [x/3.0, n/1]$ durch, wie in der Vorlesung behandelt. Runden Sie gegebenenfalls Zwischenergebnisse auf eine Nachkommastelle.

Lösungsvorschlag:

Es ist $\sigma = [x/3.0, n/1]$. Wir rufen `quadratwurzel(x,n)` auf.

$$W_{[x/3.0,n/1]}\text{quadratwurzel}(x,n) = \text{quadratwurzel}(x,n)[x/3.0],[n/1] = \text{quadratwurzel}(3.0,1)$$

Setzen wir nun den quadratwurzel Methoden Rumpf ein folgt:

$$\textbf{if } (n == 0) \textbf{ then } (x+1)/2 \textbf{ else } 0.5 * (\text{quadratwurzel}(x, n-1) + x/\text{quadratwurzel}(x, n-1))$$

Wir werten zuerst $(n == 0)$ aus $(n == 0)[n/1] = (1 == 0) = \mathit{False}$ Somit behandeln wir nun dem else Zweig:

$$W_{[x/3.0,n/0]}0.5 * (\text{quadratwurzel}(x, n-1) + x/\text{quadratwurzel}(x, n-1))$$

$$= 0.5 * (\text{quadratwurzel}(x, n-1)) + x/\text{quadratwurzel}(x, n-1))[x/3.0],[n/1]$$

$$= 0.5 * (\text{quadratwurzel}(x[x/3.0], n[n/1]-1) + x[x/3.0]/\text{quadratwurzel}(x[x/3.0], n[n/1]-1))$$

$$= 0.5 * (\text{quadratwurzel}(3.0, 0) + 3.0/\text{quadratwurzel}(3.0, 0))$$

Setzen wir nun den quadratwurzel Methoden Rumpf erneut ein folgt:

$$= 0.5 * (\textbf{if } n = 0 \textbf{ then } (x+1)/2 \textbf{ else } \{\dots\} + 3.0/(\textbf{if } n = 0 \textbf{ then } (x+1)/2 \textbf{ else } \{\dots\})$$

$$W_{[x/3.0,n/1]} \rightarrow W_{[x/3.0,n/0]}$$

Da $(n == 0) = W_{[x/3.0,n/0]}(n == 0) = \mathit{True}$ folgen wir dem ersten Zweig(if Zweig):

$$= 0.5 * (W_{[x/3.0,n/1]}((x+1)/2) + 3.0/W_{[x/3.0,n/1]}((x+1)/2))$$

$$= 0.5 * ((x[x/3.0]+1)/2) + 3.0/((x[x/3.0]+1)/2)) = 0.5 * ((3.0+1)/2) + 3.0/((3.0+1)/2))$$

$$= 0.5 * (2.0 + 1.5)) = 1.75$$

(b) Implementieren Sie eine rekursive Wurzelfunktion gemäß voriger Aufgabe in Java:

```
public static double quadratwurzel(double x, int n) {...}
```

Nutzen Sie dazu keine Methoden außer den Ihnen bereits bekannten Basisoperatoren (`+`,`-`,`*`,`/`). Sie dürfen annehmen, dass alle Eingabewerte positiv sind. Vergessen Sie nicht, Ihre Methode zu testen.

Lösungsvorschlag:

```
/** Berechnung der zweiten Wurzel einer natuerlichen Zahl mit
* Hilfe des Annaeherungsverfahrens von Heron.
* @param    x der Radikand
* @param    n die Rekursionstiefe
* @return   die angenaeherte 2-te Wurzel von x
*/
public static double quadratwurzel(double x, int n){
        if(n == 0) {
                return 0.5*(x+1.0);
        }
        else {
                double rek = quadratwurzel(x,n-1);
                double rek2 = quadratwurzel(x,n-1);
                return 0.5*(rek+x/rek);
        }
}
```

Aufgabe 5-3 *Rekursion und Methoden in Java II*

Jetzt werden Sie eine Methode implementieren, die die Kreiszahl π annähert. Dazu nutzen Sie die folgende analytische Darstellung, die im 16. Jahrhundert von Vieta entwickelt wurde:

$$\frac{2}{\pi} = \left(\frac{1}{2}\sqrt{2}\right) \cdot \left(\frac{1}{2}\sqrt{2+\sqrt{2}}\right) \cdot \left(\frac{1}{2}\sqrt{2+\sqrt{2+\sqrt{2}}}\right) \cdot \left(\frac{1}{2}\sqrt{2+\sqrt{2+\sqrt{2+\sqrt{2}}}}\right) \cdot \ldots$$

Dafür benötigen Sie die Wurzelfunktion aus der vorherigen Aufgabe. Falls Sie diese nicht gelöst haben, dürfen Sie stattdessen auf die Funktion `Math.sqrt(x)` zurückgreifen.
Die Faktoren der obigen Darstellung lassen sich induktiv definieren:

$$a_0 = \frac{1}{2}\sqrt{2}$$

$$a_{n+1} = \frac{1}{2}\sqrt{2+2a_n}$$

Implementieren Sie in Java eine rekursive Funktion

```
public static double vietaFaktor(int n) {...}
```

Für ein beliebiges $n \geq 0$ soll `vietaFaktor(n)` das Folgeglied a_n berechnen.
Implementieren Sie anschließend eine Methode

```
public static double pi(int n) {...}
```

Diese Methode nutzt n Faktoren, um mit der Formel von Vieta die Kreiszahl π anzunähern und zurückzugeben. Hier ist es Ihnen überlassen, ob Sie dies rekursiv umsetzen.

Lösungsvorschlag:

```java
/** Berechnung eines Vieta-Faktors zur Approximation des
* Wertes von Pi.
* @param   n die Rekursionstiefe
* @return  der approximierte Wert von Pi
*/
public static double vietaFaktor(int n) {
        if(n == 0){
                return 0.5*quadratwurzel(2.0,20);
        }
        else {
                return 0.5*quadratwurzel
                (2.0+2.0*vietaFaktor(n-1),20);
        }
}

/** Approximation des Wertes von Pi mit Hilfe der analytischen
* Darstellung von Vieta.
* @param   n die Rekursionstiefe
* @return  der approximierte Wert von Pi
*/
public static double pi(int n) {
        if(n == 0)
                return 2.0/vietaFaktor(0);
        else
                return 1.0/vietaFaktor(n)*pi(n-1);
}
```

Aufgabe 5-4 *Rekursion und Methoden in Java III*

Implementieren Sie eine Methode `berechneDistanz(x1,y1,x2,y2)`, die die (euklidische) Distanz

$$d(x,y) = \sqrt[2]{((x_2 - x_2 1^2 + (y_2 - y_1)^2}$$

zwischen zwei Punkten im zweidimensonalen Raum berechnet. Hier können Sie Ihre Wurzelfunktion gerne anwenden oder Sie nutzen die bereits implementierte Methode `Math.sqrt(double x)`.

Lösungsvorschlag:

```java
/**
          * Berechnung der Distanz zwischen zwei zweidimensionalen Punkten.
          * @param   x1 die erste Koordinate des ersten Punktes
          * @param   y1 die zweite Koordinate des ersten Punktes
          * @param   x2 die erste Koordinate des zweiten Punktes
          * @param   y2 die zweite Koordinate des zweiten Punktes
          * @return  Die Distanz zwischen Punkt 1 und Punkt 2
          */
public static double berechneDistanz(double x1,double y1,double x2,
                                    double y2){
                double dx = x2-x1;
                double dy = y2-y1;
                return quadratwurzel(dx*dx+dy*dy,10);
}
```