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

München, 17.11.2016

Einführung in die Programmierung

WS 2016/17

Übungsblatt 5: Ausdrücke, Substitution, Rekursion

Besprechung: 28.11./30.11./02.12.2016

Ende der Abgabefrist: Freitag, 25.11.2016 14:00 Uhr.

Aufgabe 5-1 *Ausdrücke* **6 Punkte**

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.
Abgaben, bei denen nur der Wert angegeben sind werden mit 0 Punkten bewertet.

(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`

Aufgabe 5-2 *Rekursion und Methoden in Java* **2+2+3+1+4+3+3 Punkte**

In dieser Aufgabe sollen Sie für zwei gegebene Kreise den kleinsten Kreis bestimmen, der beide Eingabekreise enthält. Anschließend soll der Umfang und die Fläche dieser „Bounding Sphere“ berechnet werden. Eine `main`-Methode wird Ihnen dabei bereits zur Verfügung gestellt. Die Entwicklung der nötigen Funktionalität behandeln wir getrennt in mehreren Teilaufgaben.
Erstellen Sie für die Abgabe eine Klasse `Kreis.java` und geben Sie Ihre Lösung für diese Aufgabe in dieser Datei ab. Schreiben Sie also sämtlichen Code in diese Datei und notieren Sie sonstige Antworten entweder als Java-Kommentar, um die Kompilierfähigkeit zu bewahren, oder in einer sinnvoll benannten weiteren Datei entsprechenden Formats.
Hinweis: Ab jetzt werden nur noch kompilierbare java-Dateien korrigiert!

(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 im Skript Seite 275. Runden Sie gegebenenfalls Zwischenergebnisse auf eine Nachkommastelle.

(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.

(c) 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.

(d) Nun widmen wir uns wieder der eigentlichen Problemstellung. Ein Kreis im zweidimensionalen Raum ist definiert durch einen Mittelpunkt $(x, y) \in \mathbb{R}^2$ und einen Radius $r \in \mathbb{R}$. Für zwei Kreise nennen wir den kleinsten Kreis, der beide Kreise vollständig umfasst, die Bounding Sphere. Betrachten Sie als Beispiel die Abbildung 1.

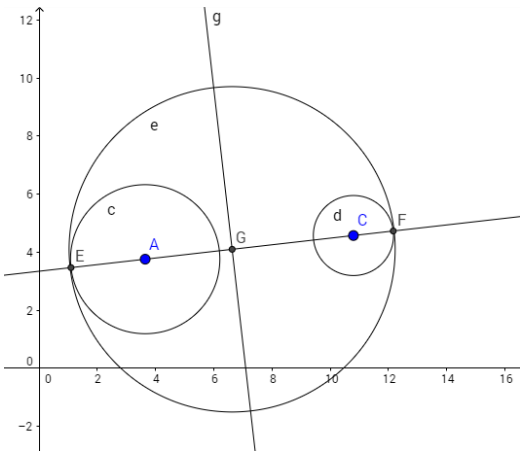

Abbildung 1: Bounding Sphere zweier Kreise c und d. Kreis e umfasst beide Kreise komplett. Sein Mittelpunkt liegt auf der Geraden durch beide Kreismittelpunkte. Bedingt durch die unterschiedlichen Radien liegt er hier näher zu c.

Optional können Sie mit der Mathe-App GeoGebra (https://www.geogebra.org/) und der auf unserer Webseite zur Verfügung gestellten Datei `geogebra-export.ggb` mit dem Szenario experimentieren.
Zum Einstieg implementieren Sie eine Methode `berechneDistanz(...)`, die die (euklidische) Distanz zwischen beiden Mittelpunkten berechnet. Hier können Sie Ihre Wurzelfunktion gerne anwenden oder Sie nutzen die bereits implementierte Methode `Math.sqrt(double x)`.

(e) Implementieren Sie dann eine Methode `berechnePole(...)`, die Ihnen für verschiedene Parameter die Koordinaten der Pole E und F berechnet. Nutzen Sie eine geometrische Herangehensweise.
Hinweis: Wenn `berechnePol(x1,y1,r1,x2,y2,r2)` *die x-Koordinate von Pol E berechnet, dann liefert* `berechnePol(y1,x1,r1,y2,x2,r2)` *seine y-Koordinate und* `berechnePol(x2,y2,r2,x1,y1,r1)` *die x-Koordinate von Pol F.*

(f) Implementieren Sie Methoden, die die drei benötigten Werte für den Mittelpunkt und den Radius der Bounding Sphere berechnen, sodass die Hauptklasse `ProgrammUe05` kompiliert und die korrekte Ausgabe liefert.

```
public static double berechneX(...)
public static double berechneY(...)
public static double berechneR(...)
```

Falls Sie die Methode `pi(int n)` nicht implementiert haben, können Sie `Math.PI` verwenden.

(g) Kommentieren Sie sämtliche implementierten Methoden gemäß Javadoc, wie sie es zum Beispiel auf der Vorlesungsfolie Seite 285 gesehen haben.