Lehrstuhl für Angewandte Informatik I
Prof. Dr. Bernhard Westfechtel
Sandra Greiner

WS 2017/18
18. Oktober 2017

Konzepte der Programmierung

Übung 1 – Abgabe: 26.10.2017

Achten Sie ab dieser Übung auf
die Einhaltung der Quellcodekonventionen!

Aufgabe 1.1 (Quadratwurzel als Java-Programmen)
Erstellen Sie in BlueJ eine Klasse `Quadratwurzel` und füllen Sie sie mit folgendem Quelltext[1]:

```
1   public class Quadratwurzel {
2
3       // Berechnung der Quadratwurzel
4       public static void main(String[] args) {
5           Out.println("Bitte geben Sie eine positive reelle Zahl ein.") ;
6           double zahl = In.readDouble(); // Einlesen einer Kommazahl
7           if (zahl <= 0.0) {
8               Out.println("Negative Wurzeln kennen wir nicht.");
9               return;
10          }
11
12          double wurzel = zahl / 2.0;
13          double testwert = zahl / wurzel;
14          Out.println("Initialisierung. wurzel=" + wurzel + ", testwert=" +
    testwert);
15
16          int iteration = 1;  // Initialisieren des Schleifenzählers
17          while (Math.abs(testwert - wurzel) >= 0.00001) {
18              wurzel = (wurzel + testwert) / 2.0;
19              testwert = zahl / wurzel;
20              Out.println("Iteration " + iteration + ": wurzel=" + wurzel + ",
    testwert=" + testwert);
21              iteration = iteration + 1;
22          }
23
24          Out.println("Die Quadratwurzel von " + zahl + " ist " + wurzel + ".");
25
26       }
27  }
```

Übersetzen Sie die Klasse und führen Sie danach das `main`-Programm einige Male auf der Konsole aus und geben Sie jeweils unterschiedliche Zahlenwerte ein. Beachten Sie dabei, dass beim Einlesen einer Dezimalzahl der Punkt (`.`) und nicht das Komma die Nachkommastellen abtrennt.

[1] Hinweis zu Zeile 17: Ein Aufruf von `Math.abs(x)` liefert den *Absolutbetrag* einer Dezimalzahl `x`.

(a) Ordnen Sie den acht unterstrichenen Quelltextfragmenten unter Angabe der Quelltextzeile die entsprechenden Kategorien für *lexikalische Einheiten* zu (schlagen Sie an geeigneter Stelle im Skript nach) oder begründen Sie, warum es sich dabei um keine lexikalische Einheit handelt.

Im Folgenden werden Sie einige Änderungen am Quelltext vornehmen. **Beschreiben Sie die jeweiligen Auswirkungen auf die Semantik des Programmes oder begründen Sie, warum diese unverändert bleibt.** Übersetzen Sie Ihr Programm nach jeder Teilaufgabe erneut und führen Sie es mit demselben Eingabewert aus. Behalten Sie die jeweiligen Änderungen am Programm nach jeder Teilaufgabe bei.

(b) Ersetzen Sie sämtliche Vorkommen des Textes „`iteration`“ (mit kleinem Anfangsbuchstaben) durch den Text „`i`“.

(c) Ändern Sie Zeile 16 wie folgt ab:

```
int i = 0;
```

(d) Ersetzen Sie sämtliche Vorkommen des Textes „`Iteration`“ (mit großem Anfangsbuchstaben) durch den Text „`Durchlauf`“.

(e) Tauschen Sie Zeilen 21 und 22.

(f) Ändern Sie Zeile 17 wie folgt ab:

```
while (Math.abs(wurzel - testwert) >= 0.00001) {
```

(g) Ersetzen Sie sämtliche Vorkommen des Textes „`double`“ durch „`float`“ sowie „`readDouble`“ durch „`readFloat`“. Hängen Sie außerdem an jedes Gleitpunkt-Literal ein ‚`f`‘ an, z.B. „`2.0f`“.

(h) Entfernen Sie die Zeilen 7 bis einschließlich 10.

(4+7 Punkte)

Aufgabe 1.2 (Flussdiagramm)
Von James Gregory (1638 – 1675) und Gottfried Wilhelm von Leibniz (1646 – 1716) ist die folgende Näherungsformel für die Berechnung der reellen Zahl π überliefert:

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \dots$$

Um π auf diese Weise berechnen zu können, benötigt man ein Kriterium, nach wie vielen Summanden man mit der Berechnung aufhören soll. Dazu könnte man z.B. die Berechung beenden, wenn der aktuelle Summand $\pm\frac{1}{n}$ ($n \in \mathbb{N}$) betragsmäßig vernachlässigbar klein ist, d.h.

$$\frac{1}{n} < \epsilon \quad (\epsilon > 0 \text{ vorgegeben}).$$

Erstellen Sie ein Flussdiagramm, welches das hier beschriebene Verfahren zur Berechnung eines Näherungswertes von π (!) beschreibt!
Hinweis: Sie können davon ausgehen, dass ϵ bereits gegeben ist. Führen Sie außerdem geeignete Variablen ein.

(5 Punkte)

Aufgabe 1.3 (Summenberechnung mit BlueJ)
In der Vorlesung haben Sie bereits eine Klasse $Fakultaet$ mit einer Methode fak kennengelernt, die zu einer natürlichen Zahl n deren Fakultät berechnet (siehe Kapitel Programme, Folie 18).

(a) Legen Sie in BlueJ ebenfalls ein Projekt mit einer Klasse $Fakultaet$ an und übernehmen Sie den Quellcode aus dem Vorlesungsskript. Testen Sie die Methode fak, indem Sie die Klasse kompilieren und die Methode zunächst mit einer natürlichen Zahl aufrufen. Was passiert, wenn Sie die Methode mit einer negativen ganzen Zahl aufrufen? Wie ist dieses Ergebnis zu erklären?

(b) Schreiben Sie in BlueJ eine Klasse $Summe$ mit einer Methode $sum1$. Darin sollen alle natürlichen Zahlen von 1 bis zu einer übergebenen natürlichen Zahl n summiert werden. Welches Konzept verwenden Sie zur Umsetzung?

(c) Fügen Sie zu Ihrer selbstgeschriebenen Klasse $Summe$ eine weitere Methode $sum2$ hinzu, welche die Summe aller Zahlen von $1...n$ über die Formel $n * (n + 1)/2$ berechnet.

(d) Schreiben Sie eine weitere Methode $sumGerade$, die die Summe aller GERADEN Zahlen von 2 bis zu einer übergebenen Zahl berechnet.

Testen Sie Ihre Methoden, bevor Sie die Lösung abgeben. Die Methoden $sum1$ und $sum2$ sollten für die gleiche natürliche Zahl das gleiche Ergebnis liefern.

(1+2+1+2 Punkte)