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

WS 2017/18
27. Oktober 2017

Konzepte der Programmierung

Übungsblatt 1

Lösungsskizzen

Aufgabe 1.1 (Quadratwurzel als Java-Programm)

(a) Zuordnung der 8 unterstrichenen Quelltextfragmente

- Z. 3: **Kommentare** sind keine lexikalischen Einheiten. Sie sind für den Compiler genauso unsichtbar wie Zeilenumbrüche, Tabstops und Leerzeichen.
- Z. 5: Das Semikolon ist ein **Begrenzer**, der aufeinanderfolgende Anweisungen voneinander trennt.
- Z. 6: `In` ist der **Bezeichner** einer verwendeten Klasse.
- Z. 12: Im Quelltext notierte Gleitpunktzahlen sind **Literale**. Sie repräsentieren den Datentyp `double`.
- Z. 14: Zeichenketten innerhalb `" "` sind **Literale**. Sie repräsentieren den Datentyp `String`.
- Z. 16: Das **Schlüsselwort** `int` kennzeichnet Variablen vom gleichnamigen Datentyp, der Ganzzahlen repräsentiert. Analog steht `double` für Kommazahlen (auch: Gleitpunktzahlen).
- Z. 17 `while` ist ein **Schlüsselwort**. Es kennzeichnet den Beginn einer Schleife. Direkt darauf folgt die Schleifenbedingung in runden Klammern. Die Schleife wird solange ausgeführt bis die Bedingung nicht mehr erfüllt ist. Der Inhalt der Schleife steht innerhalb der darauffolgenden geschweiften Klammern.
- Z. 22: Die geschweifte Klammer ist ein **Begrenzer**, zusammen mit der öffnenden Klammer in Z. 17 umschließt sie den *Rumpf* der `while`-Schleife, analog umschließen z.B. die Klammern in Zeile 4 und 26 den Rumpf der `main`-Methode.

(b) Diese Ersetzung hat keine Auswirkungen, da sich lediglich der **Bezeichner** einer Variable ändert. Für das ausgeführte Programm sind Variablenbezeichner „Schall und Rauch“.

(c) Die Variable `i` wird nicht mehr mit dem `int`-Literal `1`, sondern mit `0` initialisiert. Entsprechend ändert sich die Programmausgabe: Wie in der Informatik üblich, wird bei Null zu zählen begonnen.

(d) Das Ändern eines **Literals** kann zu unterschiedlichem Laufzeitverhalten führen. In diesem speziellen Fall betrifft das nicht die Berechnung selbst, sondern nur die Ausgabe: Statt dem Text „`Iteration`“ steht im modifizierten Programm „`Durchlauf`“.

(e) Die Anweisung `i = i + 1` wird nicht mehr innerhalb der `while`-Schleife, sondern danach ausgeführt. Während die Schleife durchlaufen wird, bleibt der Wert der Variable `i` unverändert und es wird jedes Mal „`Durchlauf 0`“ ausgegeben.

(f) Das Laufzeitverhalten des Programms ändert sich nicht: Der Umkehr der Operanden bei einer Subtraktion führt zum Vorzeichenwechsel des Ergebnisses. Das Vorzeichen ist jedoch durch die Berechnung des Absolutbetrags (`Math.abs`) irrelevant.

(g) Das Ändern eines **Datentyps** kann das Laufzeitverhalten in unterschiedlicher Weise beeinflussen. Hier wird mit geringerer Genauigkeit (`float` anstatt `double`) gerechnet, was zu einem weniger exakten Ergebnis führt. Mit `float` lässt sich dafür schneller rechnen, was sich in unserem kleinen Programm aber nicht bemerkbar macht. Wir werden bei Gleitpunktzahlen im Normalfall `double` verwenden.

(h) Der Fall, dass eine negative Zahl oder Null eingegeben wird, wird nicht mehr abgefangen. Das Programm terminiert für negative Eingaben nicht. In Fall $x = 0$ wird als Ergebnis 0 ausgegeben.

Aufgabe 1.2 (Flussdiagramm)
Das Flussdiagramm zur Berechnung eines Näherungswertes für π ist in Abbildung 1 dargestellt.

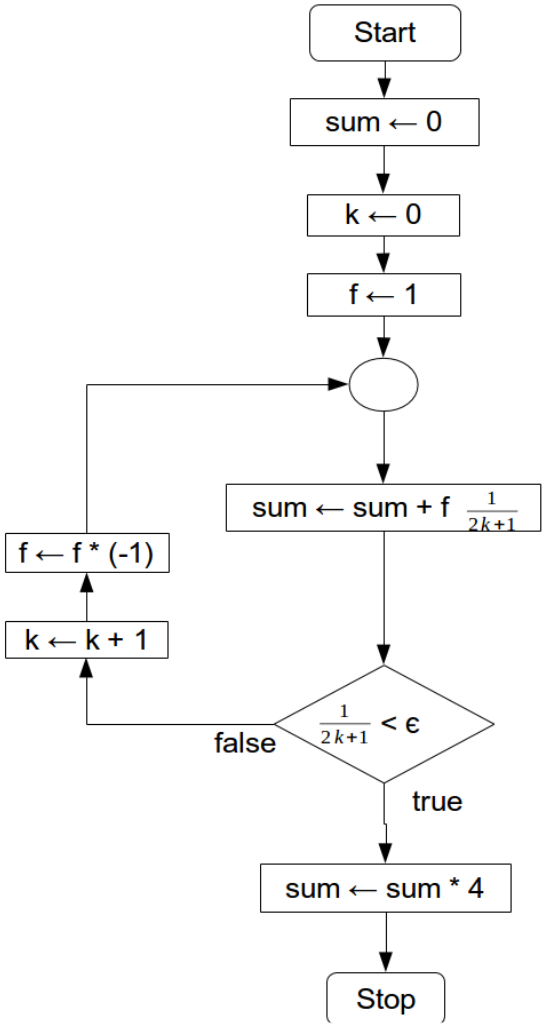

Abbildung 1: Flussdiagramm

Aufgabe 1.3 (Summenberechnung mit BlueJ)

(a) Wenn die Methode fak mit einer negativen ganzen Zahl aufgerufen wird, liefert sie das Ergebnis 1. Das liegt daran, dass res mit 1 initialisiert wird. Da die Schleifenbedingung nicht erfüllt ist, wird die Schleife nicht durchlaufen und res unverändert zurückgegeben.

(b) Als Kontrollstruktur wird eine while-Schleife verwendet.

(c) ff. siehe jar