technische universität
dortmund

Fakultät für Informatik
Lehrstuhl 14
Lars Hildebrand

Praktikum zu

Einführung in die Informatik für LogWilngs und WiMas

Wintersemester 2017/18

Übungsblatt 5

Besprechung:
20.–24.11.2017
(KW 47)

Vorbereitende Aufgaben

Aufgabe 5.1: Schleifentypen

Auf diesem Übungsblatt sollen Sie sich hauptsächlich mit Schleifen beschäftigen.

Geben Sie die grundlegende Struktur der drei in Java verfügbaren Schleifen an:

- **while**-Schleifen

- **do-while**-Schleifen

- **for**-Schleifen (Zählschleifen)

Aufgabe 5.2: Finde die passende Schleife

Formulieren Sie zu folgenden Sätzen den Kopf bzw. Fuß einer dazu passenden Schleife:

- „Führe folgendes mindestens einmal aus, solange x größer als 0 ist: “

- „Für jedes i zwischen 3 und 15, führe folgendes aus: “

- „Solange x kleiner ist als 20, führe folgendes aus: “

Präsenzaufgaben

Aufgabe 5.3: while-Schleifen

In dieser Aufgabe sollen Sie sich mit der Implementierung einer Wiederholung mit Hilfe einer **while**-Schleife vertraut machen.

Die Collatz-Folge einer Zahl n wird nach einer einfachen Regel gebildet:

- Ist n gerade, setze n auf $n/2$.
- Ist n ungerade, setze n auf $n \cdot 3 + 1$.
- Wiederhole dies mit dem neuen n.

Es scheint, als würde diese Folge für jeden Startwert mit dem Zyklus $4, 2, 1$ enden. Ob dies wirklich so ist, ist bis heute ein ungelöstes Problem der Mathematik.

a) Berechnen Sie – per Hand – die Collatz-Folge dieser Zahlen:

   - 12

   - 13

   - 64

   - 9

   ---

b) Wir wollen die Berechnung der Collatz-Folge nun programmieren:

   - Erstellen Sie eine neue Klasse mit dem Namen **Collatz** im Paket **blatt05**.
   - Ergänzen Sie den Rumpf der Klasse um eine **main**-Methode.
   - Deklarieren und initialisieren Sie eine **int**-Variable mit dem Namen **collatz** mit einem beliebigen, positiven Wert als Startwert der Collatz-Folge.
   - Implementieren Sie folgende Anweisungen innerhalb einer **while**-Schleife. Die Schleife soll abbrechen, wenn die Variable **collatz** den Wert 1 erreicht.
     - Geben Sie den Inhalt der Variable **collatz** aus.
     - Implementieren Sie eine if-Anweisung, die in Abhängigkeit des aktuellen Wertes der Variablen **collatz** den Wert des nächsten Folgegliedes berechnet und in der Variablen **collatz** vermerkt.
   - Geben Sie nach Abbruch der Schleife den Wert der Variable **collatz** noch einmal aus.

Aufgabe 5.4: do-while-Schleifen

In dieser Aufgabe sollen Sie sich mit der wiederholten Ausführung von Anweisungen mit Hilfe einer **do-while**-Schleife vertraut machen.

Die Fibonacci-Folge ist eine bekannte Folge von Zahlen, die auch häufig in der Natur anzutreffen ist. Die bekanntesten Beispiele finden Sie in Radien von Schneckenhäusern, Ebenen von Blütenringen oder Hasenpopulationen wieder.

Diese Folge wird wie folgt berechnet:

- Die ersten zwei Elemente sind 1.
- Jedes weitere Element ist die Summe der beiden vorhergegangenen Elemente.

a) Berechnen Sie – per Hand – die ersten 10 Fibonacci-Zahlen:

---

b) Implementieren Sie nun die Fibonacci-Folge.

   - Erstellen Sie eine neue Klasse mit dem Namen **Fibonacci** im Paket **blatt05**.
   - Ergänzen Sie den Rumpf der Klasse um eine **main**-Methode.
   - Deklarieren Sie zwei **int**-Variablen mit den Namen **fibLast** und **fibCurrent** und initialisieren Sie sie mit den Werten 0 und 1.
   - Deklarieren und initialisieren Sie eine **int**-Variable mit dem Namen **count** und einem Wert, der repräsentiert wie viele Elemente der Fibonacci-Folge ausgegeben werden sollen.
   - Implementieren Sie eine **do-while**-Schleife, die folgende Anweisungen wiederholen soll, solange **count** größer als 1 ist:
     - Geben Sie den Wert der Variable **fibCurrent** aus.
     - Setzen Sie den Wert der Variable **fibLast** auf den Wert von **fibCurrent** und den Wert von **fibCurrent** auf die Summe des **ehemaligen** Wertes von **fibLast** und **fibCurrent**. Verwenden Sie eine Hilfsvariable zum Zwischenspeichern der benötigten Werte.
     - Reduzieren Sie den Wert von **count** um 1.

c) Welches Problem ergibt sich bei der Implementierung dieses Programmes mit einer **do-while**-Schleife, wenn der Wert von **count** kleiner ist als 1?

Wie kann man das Problem beheben?

Aufgabe 5.5: for-Schleifen

In dieser Aufgabe sollen Sie sich mit der Implementierung einer Wiederholung mit Hilfe einer **for**-Schleife (Zählschleife) vertraut machen.

Die Fakultät einer natürlichen Zahl n, geschrieben $n!$, ist definiert als das Produkt aller natürlichen Zahlen von 1 bis n. Die Fakultät von 0 (0!) ist dabei per Definition 1.

- Erstellen Sie eine neue Klasse mit dem Namen **Factorial** im Paket **blatt05**.
- Ergänzen Sie den Rumpf der Klasse um eine **main**-Methode.
- Deklarieren und initialisieren Sie eine **long**-Variable namens **factorial** mit dem Wert 1.
- Deklarieren und initialisieren Sie eine **int**-Variable namens **n** mit einem beliebigen Wert, für den Sie die Fakultät berechnen wollen.
- Implementieren Sie eine **for**-Schleife, die von 1 bis n zählt und den Wert der Zählvariablen mit **factorial** multipliziert und so den neuen Wert für **factorial** bildet.
- Geben Sie den Wert von **factorial** aus.

Ergänzende Aufgaben

Aufgabe 5.6: Verschachtelte Schleifen

In dieser Aufgabe sollen Sie sich mit der Verwendung zweier verschachtelter Schleifen vertraut machen.

Ein Tupel (a, b) ist eine Kombination zweier mathematischer Objekte zu einem einzelnen.

So ist z. B. das Tupel $(1, 2)$ das Tupel mit den Zahlen 1 und 2.

a) Schreiben Sie ein Programm namens **Tuples** im Paket **blatt05**, in dem Sie zwei Variablen n und m deklarieren und initialisieren und alle Tupel **unterschiedlicher** Zahlenpaare zwischen 1 und n für das erste Element des Tupels und zwischen 1 und m des zweiten Elementes ausgeben. Verwenden Sie dazu zwei ineinander verschachtelte Schleifen.

   Die Ausgabe *kann* für $n = 2$ und $m = 4$ z. B. folgendermaßen aussehen:

```
(1,2)
(1,3)
(1,4)
(2,1)
(2,3)
(2,4)
```

b) Schreiben Sie ihr Programm so um, dass es nun ein Zahlenpaar nur genau einmal ausgibt, also anstatt z. B. die Tupel $(1, 2)$ und $(2, 1)$ auszugeben nur eines von beiden ausgibt.

   Hierfür können Sie annehmen, dass $m \geq n$ gilt.

   Die Ausgabe *kann* für $n = 2$ und $m = 4$ z. B. folgendermaßen aussehen:

```
(1,2)
(1,3)
(1,4)
(2,3)
(2,4)
```