



Praktikum zu  
**Einführung in die Informatik für  
LogWings, WiMas und MedPhys**  
Wintersemester 2024/25

**Übungsblatt 5**  
Besprechung:  
25.11.–29.11.2024  
(KW 48)

## Vorbereitende Aufgaben

### Aufgabe 5.1: Schleifentypen

Auf diesem Übungsblatt wollen wir uns hauptsächlich mit Schleifen beschäftigen. Geben Sie die Syntaxdiagramme der drei in Java verfügbaren Schleifen an:

a) **for**-Schleifen (Zählschleifen)

b) **while**-Schleifen

c) **do-while**-Schleifen

# Präsenzaufgaben

## Aufgabe 5.2: Schleifenquiz

### Quiz

Welche Schleife eignet sich für folgende Probleme am besten?

- i) Schälen von Kartoffeln aus einem großen Kartoffelsack
  - a) for-Schleife
  - b) while-Schleife
  - c) do-while-Schleife
- ii) Passieren der Sicherheitskontrolle am Flughafen
  - a) for-Schleife
  - b) while-Schleife
  - c) do-while-Schleife
- iii) Berechnen der Durchschnittsnote einer Klausur in einer Schulklasse
  - a) for-Schleife
  - b) while-Schleife
  - c) do-while-Schleife

## Aufgabe 5.3: Finde die passende Schleife

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

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

---

- b) „Für jedes  $i$  ab 3 bis 15, führe folgendes aus: “

---

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

---

## Aufgabe 5.4: for-Schleifen

In dieser Aufgabe wollen wir uns 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 ist dabei per Definition 1.

Erstellen Sie zuerst eine neue Klasse mit dem Namen **Factorial** und ergänzen Sie den Rumpf der Klasse um eine **main**-Methode.

Deklariert und initialisiert Sie eine **long**-Variable namens **factorial** mit dem Wert 1 und 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, den Wert der Zählvariablen mit **factorial** multipliziert und so den neuen Wert für **factorial** bildet. Geben Sie abschließend das Ergebnis der Fakultät aus.

### Aufgabe 5.5: while-Schleifen

In dieser Aufgabe wollen wir uns mit der Implementierung einer Wiederholung mit Hilfe einer **while**-Schleife vertraut machen. Dafür betrachten wir die Collatz-Folge. Die Collatz-Folge einer Zahl  $n$  wird nach folgender 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:

i) 12

---

ii) 13

---

iii) 9

---

b) Wir wollen die Berechnung der Collatz-Folge nun programmieren: Erstellen Sie zunächst eine neue Klasse mit dem Namen **Collatz**. 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. In der Schleife sollen Sie den Inhalt der Variable **collatz** ausgeben. Implementieren Sie zudem eine if-Anweisung, die in Abhängigkeit des aktuellen Wertes der Variablen **collatz** den Wert des nächsten Folgliedes berechnet und den alten Wert von **collatz** überschreibt.

Geben Sie abschließend nach Schleifenabbruch den Wert der Variable **collatz** noch einmal aus.

### Aufgabe 5.6: Muster

In dieser Aufgabe beschäftigen wir uns weiter mit Schleifen. Versuchen Sie mithilfe von zwei sinnvoll ineinander verschachtelten **for**-Schleifen folgende Muster nachzubilden. Programmieren Sie ihre Lösung.

a) Ausgabe:

```
11111
22222
33333
44444
55555
```

b) Ausgabe:

```
12345
12345
12345
12345
12345
```

c) Ausgabe:

```
1
12
123
1234
12345
```

## Ergänzende Aufgaben

### Aufgabe 5.7: Weitere Muster

In Aufgabe 6 hatten Sie bereits einige Muster implementiert. Überlegen Sie, wie Sie folgende Ausgaben mithilfe von zwei sinnvoll ineinander verschachtelten for-Schleifen erzeugen können und implementieren Sie dies.

a) Ausgabe:

```
5
.4
..3
...2
....1
```

b) Ausgabe:

```
1.3.5
1.3.5
1.3.5
1.3.5
1.3.5
```

### Aufgabe 5.8: do-while-Schleifen

In dieser Aufgabe wollen wir uns mit der wiederholten Ausführung von Anweisungen mit Hilfe von **do-while**-Schleifen vertraut machen.

Die Fibonacci-Folge ist eine bekannte Folge von Zahlen, die auch häufig in der Natur anzutreffen ist. Die bekanntesten Beispiele hierfür sind Radien von Schneckenhäusern, Ebenen von Blütenringen oder Hasenpopulationen. Die Fibonacci-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**. Ergänzen Sie den Rumpf der Klasse um eine **main**-Methode.

Deklarieren Sie zwei **int**-Variablen mit den Namen **fibLast** und **fibCurrent** und initialisieren Sie diese mit den Werten 0 und 1. Deklarieren Sie eine **int**-Variable mit dem Namen **count** und initialisieren Sie diese mit 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 0 ist: Geben Sie den Wert der Variable **fibCurrent** aus. Setzen Sie den Wert der Variable **fibLast** auf den Wert von **fibCurrent**. Setzen Sie dagegen 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 initial der Wert von **count** kleiner ist als 1?

---

Wie kann man das Problem beheben?

---