Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

**Programmierung 1**
WiSe 25/26
Moritz Balg
03.11.2025

# Übungsblatt 5

- Kontrollstrukturen I -

## Aufgabe 1: Java Code Conventions

Code Conventions sind (freiwillige) Konventionen zum Schönschreiben von (Java-)Programmen, die man beim Programmieren anwenden sollte. Teilweise werden diese Konventionen automatisch von eclipse (oder anderen IDEs) angewandt (zum Beispiel das Einrücken von Zeilen). Sie finden die Java Code Conventions unter: https://www.oracle.com/java/technologies/javase/codeconventions-contents.html

Lesen und verstehen Sie die Code Conventions. Notieren Sie in einer Zusammenfassung (z.B. tabellarisch) die aus ihrer Sicht wesentlichen Punkte dieser Konventionen. An einigen Stellen kommen Java-Konstrukte vor, die Sie noch nicht kennen. Ignorieren Sie diese Stellen im Moment.

**Achtung**

Ab sofort sind diese Code Conventions verbindlich für alle (!) Java Programme, die Sie schreiben.

**Info**

Der Zeitbedarf für diese Aufgabe beträgt ca. 2-3 Stunden!

## Aufgabe 2: 👾 Bitquersumme

Geben Sie ein Java-Programm in einer Klasse `BitQuersumme` an, das die Quersumme über die niederwertigsten 16 Bits einer `int`-Zahl berechnet und in einer eigenen Zeile auf dem Bildschirm ausgibt. Die Zahl wird über die Kommandozeile eingegeben.

**Beispiel**

`java BitQuersumme 3` liefert als Ergebnis 2, weil $3_{10} = 0\ldots011_2$ und davon ist die Bitquersumme 2. Nutzen Sie eine geeignete Schleifenform in ihrem Programm.

## Aufgabe 3: Leibniz-Reihe

Die Reihe $\sum_{i=0}^{\infty} \frac{(-1)^i}{2i+1} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \ldots$ konvergiert gegen $\frac{\pi}{4}$. Schreiben Sie drei Programme, die daraus eine Näherung für $\pi$ bestimmen:

1. Das erste Programm soll solange Reihenglieder berechnen und aufaddieren, bis insgesamt $n = 100000$ Reihenglieder aufaddiert wurden.
2. Das zweite Programm soll solange Reihenglieder berechnen und aufaddieren, bis der Betrag des neuen Summanden $\leq \varepsilon = 0.0000001$ ist.
3. Das dritte Programm soll solange Reihenglieder berechnen und aufaddieren, bis der berechnete Wert vom Wert `Math.PI` weniger als $0.00000001$ abweicht.

**Info**

- Der Ausdruck $(-1)^i$ bewirkt nur einen Wechsel des Vorzeichens in jeder Iteration. Dazu brauchen (sollen!) Sie keine Potenz berechnen, sondern es genügt, den Faktor 1 in jeder Iteration mit $-1$ zu multiplizieren.
- Beachten Sie, dass die Formel oben gegen $\frac{\pi}{4}$ konvergiert, Sie aber $\pi$ annäherungsweise berechnen sollen.

## Aufgabe 4: Wurfparabel

Eine Wurfparabel beschreibt die Flugbahn eines Körpers während eines Wurfs. Der allgemeine Fall ist der sog. schiefe Wurf. Schreiben Sie ein Java-Programm, das die Flugbahn eines Körpers bei einem schiefen Wurf annäherungsweise in einem zweidimensionalen System berechnet.

Der Körper wird zu einem Zeitpunkt $t_0$ von der Position $\vec{r}_0 = (x_0, y_0)$ mit einer Geschwindigkeitsbetrag von $|v_0|$ und einem Winkel $\theta$ geworfen. Simulieren Sie das Flugverhalten bis der Körper wieder auf dem Boden auftritt, d.h. $y = 0$ gilt. Gehen Sie bei der Simulation wie folgt vor. Wenn man zu einem Zeitpunkt $t$ die Position $\vec{r}_t = (x_t, y_t)$ und den Geschwindikeitsvektor $\vec{v}_t = (v_x, v_y)$ kennt und möchte die Angaben für den Zeitpunkt $t + \Delta t$ ermitteln, so kann man sich folgender Zusammenhänge bedienen.

Betrag der Geschwindigkeit $\left[\frac{m}{s}\right]$: $|v| = \sqrt{v_x^2 + v_y^2}$

Beschleunigung in x-Richtung $\left[\frac{m}{s^2}\right]$: $a_x = -C \cdot |v| \cdot v_x$

Beschleunigung in y-Richtung $\left[\frac{m}{s^2}\right]$: $a_y = -g \cdot C \cdot |v| \cdot v_y$

Neue Geschwindigkeit in x-Richtung $\left[\frac{m}{s}\right]$: $v_x = v_x + a_x \cdot \Delta t$

Neue Geschwindigkeit in y-Richtung $\left[\frac{m}{s}\right]$: $v_y = v_y + a_y \cdot \Delta t$

Neue x-Position $[m]$: $x_{t+\Delta t} = x + v_x \cdot \Delta t$

Neue y-Position $[m]$: $y_{t+\Delta t} = x + v_x \cdot \Delta t$

Die Konstante $g = 9,81 \frac{m}{s^2}$ ist die Erdbeschleunigung. C ist der Strömungswiderstandskoeffizient, den wir hier beispielhaft mit $C = 0,002$ annehmen. Als Startwerte nehmen sie für Ihr Programm $|v_0| = 80 \frac{m}{s}$ und $\theta = 60$ Grad. Geben sie für jeden Stritt die Position $\vec{r_t} = (x_t, y_t)$ aus.

**Info**

- Denken Sie daran, dass in Java Winkel nicht in Grad, sondern in Radiant angegeben werden. Überlegen Sie sich einen sinnvollen Wert für $\Delta t$.
- In der Programmierung arbeitet man häufig in interdisziplinären Teams. Es ist hier erstmal nicht Schlimm, wenn Sie die Physik dahinter nicht komplett verstehen. Sie können die Formeln trotzdem einfach in Code überführen.

## Aufgabe 5: Spezielle Realativitätstheorie

Nach der speziellen Relativitätstheorie nimmt die dynamische Masse eines Körpers zu, je schneller er sich (in einem Inertialsystem) bewegt. Die Formel dazu ist:

$$m = \frac{m_0}{\sqrt{1 - \frac{v}{c^2}}}$$

Wobei $m_0$ die Ruhemasse (das Gewicht) in kg des Körpers ist, $v$ in $\frac{m}{s}$ die Geschwindigkeit und $c = 3 \cdot 10^8 \frac{m}{s}$ die Lichtgeschwindigkeit ist. Schreiben Sie ein Java-Programm, das einen Körper ihrer Wahl (zum Beispiel ihren Körper) mit dem Gewicht $m_0$ von $v = 0 \frac{m}{s}$ Richtung Lichtgeschwindigkeit $v = 3 \cdot 10^8 \frac{m}{s}$ „beschleunigt“ (im Sinne von: zu mehreren zunehmenden Geschwindigkeiten die dynamische Masse ausrechnen). Da

es erst nahe der Lichtgeschwindigkeit interessant wird, erhöhen Sie ihre Geschwindigkeit so, dass sie zu Beginn sehr schnell zunimmt, zum Ende hin aber langsamer, also mehr Messpunkte vorhanden sind (wie kann man das in einer Schleife ausdrücken?). Geben Sie für jeden Berechnungspunkt (Geschwindigkeit) jeweils die Geschwindigkeit und die dynamische Masse aus.

## Aufgabe 6: Newton Verfahren

Programmieren Sie das Newton-Verfahren zur Berechnung einer Nullstelle einer Funktion $f : \mathbb{R} \to \mathbb{R}$ (siehe dazu auch https://de.wikipedia.org/wiki/Newtonverfahren). Gesucht wird zu einer gegebenen Funktion $f : \mathbb{R} \to \mathbb{R}$ also eine Stelle $x \in \mathbb{R}$, für die gilt, dass $f(x) = 0$ ist. Das Newton-Verfahren versucht ausgehend von einem (beliebigen) Startwert $x_0$ sukzessive weitere x-Werte zu bestimmen, mit denen man näher an eine Nullstelle kommt. Dabei funktioniert das Verfahren wie folgt. Beginnend bei einem (beliebigen) Startwert $x_0$ wird für einen Wert $x_i$ der Folgewert $x_{i+1}$ mittels der Iterationsvorschrift

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

berechnet, bis $|f(x_i)| < \varepsilon$ für eine vorgegebene Genauigkeit $\varepsilon > 0$ ist. Verwenden Sie als Beispielfunktion $f(x) = x^2 - 2$ mit dem Startwert $x_0 = 1$ und die Genauigkeit $\varepsilon = 10^{-15}$. Zur Erinnerung: Für $f(x) = x^2 - 2$ ist die Ableitung gegeben durch $f'(x) = 2x$. Verwenden Sie je eine Methode zur Berechnung von $f(x)$, von $f'(x)$ und eine für die Berechnung des Absolutbetrags. Nennen Sie diese Methoden in ihrem Java-Programm f, fstrich und abs und nutzen Sie diese Methodennamen in der obigen Berechnungsvorschrift.

- Wieviele Schleifendurchläufe benötigt Ihr Programm, bis Sie bei der vorgegebenen Funktion und Genauigkeit zu einer Lösung kommen? Wie oft wird dabei die Funktion $f$ und die Funktion $f'$ aufgerufen?
- Auch die Ableitung einer Funktion an einer Stelle $x$ kann numerisch berechnet werden, indem die Näherung $f'(x) = \frac{f(x+d)-f(x)}{d}$ für $d \to 0$ bestimmt wird.

Ändern Sie ihre Methode zur Berechnung der Ableitung so ab, dass $d$ (beginnend mit $d = 1$) solange halbiert wird, bis zwei aufeinanderfolgende Näherungswerte einen Abstand kleiner als $\varepsilon$ haben.

## Aufgabe 7: Zufallszahlen

Schreiben Sie ein Programm, das zu drei Werten $x, y, \varepsilon$ aus der Kommandozeile mit $x, y, \varepsilon \in \mathbb{R}, x < y, \varepsilon > 0, |y - x| > \varepsilon$ zwei verschiedene Zufallfszahlen aus dem Intervall $[x, y)$ erzeugt und auf dem Bildschirm ausgibt. Zwei Zahlen a, b sollen dann als verschieden gelten, wenn $|a - b| \geq \varepsilon$ .

**Info**

- Verwenden Sie die Methode `double Math.random()`. Diese liefert mit jedem Aufruf eine pseudozufällige Zahl aus dem Intervall $[0, 1)$. Überlegen Sie wie sie mit einer einfachen Berechnung diese Zahl in das angegebene Intervall $[x, y)$ überführen.
- Es ist unwahrscheinlich, aber nicht ausgeschlossen, dass zwei aufeinanderfolgende Aufrufe dieser Methode die gleiche oder fast gleiche Zufallszahl liefern. Sie müssen also solange zwei zulässige Zahlen aus dem gewünschten ntervall erzeugen, bis diese die gewünschte Eigenschaft besitzen.

## Aufgabe 8: 👾 Längste Kette

Geben Sie ein Java-Programm an, das bestimmt, was die Länge der längsten Kette gleicher natürlicher Zahlen in der Eingabe von der Tastatur ist. Die Beispieleingabe `1 1 2 2 2 3 3 3 3 4 3 3 1 5 .` liefert `4` weil vier aufeinanderfolgende Dreien vorhanden sind und keine längere Kette existiert. Die Zahlen der Eingabe werden abgeschlossen durch einen einzelnen Punkt. Sie können folgende Programmteile verwenden, um eine unbekannte Anzahl an Werten nach diesem Schema einzulesen.

```java
import java.util.*;
//...
Scanner sc = new Scanner (System.in);
while (sc.hasNextInt()){
  int i = sc.nextInt();
}
```

Die Anzahl der einzulesenden Werte wird größer als 0 sein und nach oben hin ist die Anzahl offen. Überlegen Sie sich, was mögliche Fälle sind und wie Sie diese behandeln. Überprüfen Sie ihr Verfahren / Programm vor der Abgabe erst per Hand / auf dem Papier, dann in eclipse mit verschiedenen Testwerten und geben erst dann ab, wenn dies alles korrekte Ergebnisse liefert.

**Achtung**

Wie immer: Sie dürfen nur Sprachkonstrukte verwenden, die in der Vorlesung auch bereits vorgestellt wurden!

## Aufgabe 9: 👾 Lauflängenkodierung

Schreiben Sie ein Java-Programm Lauflaengenkodierung, das eine Lauflängenkodierung für beliebig viele Eingabezahlen aus $\{0, 1, 2, ..., 9\}$ vornimmt und auf dem Bildschirm in entsprechender Form ausgibt. Alle Eingabezahlen werden über die Tastatur übergeben und sind jeweils durch Leerzeichen oder Zeilenendezeichen voneinander getrennt. Die Eingabe wird abgeschlossen durch einen einzelnen Punkt. vSiehe dazu die Vorlage zum Einlesen von Werten „Einlesen von Zahlen, deren Anzahl man vorher nicht kennt“, die Sie über unsere Java-Webseite ganz unten über den Link „Beispiele zum Einlesen von Daten“ erreichen. Eine Lauflängenkodierung geschieht dabei folgendermaßen. Für $n$ aufeinanderfolgende gleiche Werte $w$ wird diese Eingabesequenz gleicher Werte in der Ausgabe ersetzt durch die Angabe nxw . , also die Anzahl $n$ der Vorkommen, das Zeichen $x$ (kleines x), der eigentliche Wert $w$ (der n-mal vorkam) und als Schlusszeichen ein Punkt. Auch ein einzeln auftretender Wert $w$ wird nach diesem Verfahren kodiert und demzufolge als 1xw . erscheinen. Die gesamte Ausgabe wird durch ein Zeilenende abgeschlossen. Zur besseren Lesbarkeit der Beispiele wurde in den bisherigen Beispielen jeweils ein Leerzeichen zwischen den 4 Komponenten eingefügt, das in ihrer Ausgabe aber weggelassen wird (siehe Referenzausgabe unten).

**Beispiel**

- Beispiel 1: `java Lauflaengenkodierung` mit einer Eingabe von der Tastatur
3 1 1 1 7 7 1 1 0 . würde als Ergebnis liefern: `3x1.2x7.2x1.1x0.` (3 Einsen, dann 2 Sieben, dann 2 Einsen,...). Alles hintereinander, ohne Leerzeichen.
- Beispiel 2: `java Lauflaengenkodierung` mit einer Eingabe von der Tastatur
1 2 2 3 3 3 4 4 4 4 5 5 5 5 5 . würde als Ergebnis liefern: `1x1.2x2.3x3.4x4.5x5.`

**Info**

- Erweitern Sie in geeigneter Weise einen String.
- Dies ist eine alte Klausuraufgabe für ca. 15 Minuten Bearbeitungszeit.