Übungen Kapitel 2

Hinweis: Bei allen Aufgaben, bei denen Benutzereingaben erforderlich sind, verwenden Sie die mitgelieferte Klasse `Scanner`. (`Codebeispiele\ScannerExample`).

2.1 ( ) Variable

Wie kann der Wert einer Variablen verändert werden?

2.2 (*) Zahlensysteme *(hexadezimal, octal, dual, dezimal Umwandlung)*

Schreiben Sie ein Konsolenprogramm **Zahlensysteme**, das jeweils einen Zahlenwert in andere Zahlensysteme konvertiert. Das folgende Beispiel zeigt die Benutzung anhand eines Beispiels:

```
Bitte geben Sie eine Zahl dezimal ein
   11
   Hex: b
   Binär:1011
   Bitte geben Sie eine Zahl hexadezimal ein
   FF
   Dezimal: 255
   Binär: 11111111
   Bitte geben Sie eine Zahl binär ein
   101
   Dezimal: 5
   Hex: 5
   Bitte geben Sie eine Zahl octal ein
   77
   Dezimal: 63
   Hex: 3f
```

Lösungshinweis: Verwenden Sie hierzu die `Integer.parseInt(String s, int base)` Methode. Machen Sie sich außerdem mit den Funktionen `Integer.toBinaryString` sowie `Integer.toHexString` vertraut.
Dazu lesen Sie die entsprechende javadoc Dokumentation durch: z.B.

https://docs.oracle.com/en/java/javase/13/docs/api/java.base/java/lang/Integer.html

2.3 ( ) Variablentypen *(Java Datentypen)*

Legen Sie für die folgenden Variablen mit der zugeordneten Bedeutung die Datentypen fest und schreiben eine entsprechende Java **Deklaration mit Initialisierung** und begründen Ihre Entscheidung:

a. `einePostLeitzahl`, die alle möglichen Postleitzahlen speichern soll
b. `eineKarteikarte`, die für einen Karteikasten einen Buchstaben des Alphabets speichert. Ziffern sind nicht erlaubt.
c. `PI`, die die Zahl Pi repräsentiert
d. `einStrassenname`, der einen Straßennamen mit Straßennummer enthalten soll
e. `dieRaumNummer`, das für ein Gebäude mit 110 Räumen die Raumnummer speichert
f. `einLichtSchalter`, der angibt, ob ein Schalter an oder aus ist.
g. `dieAussentemperatur`, die die aktuelle Temperatur speichert.
h. `dasStockwerk`, die eine aktuelle Nummer des Stockwerks für einen Aufzug enthält
i. `eineISBNNummer`, die eine ISBN Nummer eines Buches speichern soll
j. `derAutor`, das den Autor eines Buches speichert

k. `dieAnzahlPCs`, die im Rahmen einer statistischen Untersuchung die Anzahl der PC's weltweit speichert.

2.4 (*) Wochentag *(arithmetische Operatoren, ganzzahlig)*

Der Geistliche Christoph Zeller hat 1885 eine Formel aufgestellt, die für ein gegebenes Kalenderdatum den Wochentag liefert. Die Zeller'sche Formel lautet:

$$w = (d + \frac{26*(m+1)}{10} + \frac{5*y}{4} + \frac{c}{4} + 5*c - 1) \bmod 7$$

wobei:

- **d** Tag im Monat (1 <= d <= 31)
- **m** Monat (3 <= m <= 14) In die Formel müssen **Januar und Februar als Monate 13 und 14 des Vorjahres** eingesetzt werden
- **y** Jahr im Jahrhundert (0 <= d <= 99)
- **c** Jahrhundert
- **w** Index des Wochentages, gezählt ab 0 = Sonntag bis 6= Samstag

Alle Divisionen sind ganzzahlig.

Beispiel: Für das Jahr 1847 muss also **c** den Wert 18 und **y** den Wert 47 haben!

Schreiben Sie ein Programm Zeller, das drei Zahlen für Tag, Monat und Jahr von der Konsole als Zahlen einliest und den Wochentag durch die angegebene Umrechnung ermittelt und ausgibt.

Eine Ausgabe für den 14. Februar 2000 sieht dann wie folgt aus:

```
Tag eingeben
14
Monat eingeben
14
Jahr eingeben
1999
1
```

Wer möchte kann das Programm auch so umbauen, dass für die Monate Januar und Februar 1 resp. 2 eingegeben wird und der Wochentag im Klartext ausgegeben wird:

```
Tag eingeben
14
Monat eingeben
2
Jahr eingeben
2000
Montag
```

2.5 (*) Formeln *(Reelle Arithmetik)*

Zum Testen der Fließkommaarithmetik unter Java schreiben Sie folgendes Programm:
Es definiert drei double- Variable x, y, z. Es fordert den Bediener zur Eingabe von zwei Fliesskommazahlen auf und liest diese in die Variablen x und y ein. Es berechnet der Reihe nach mit möglichst wenigen Operationen die folgenden in mathematischer Notation geschriebenen Ausdrücke:

a) $z = x^2y^2 - 4xy + 4$

b) $z = \frac{(1+xy)^2}{1+(1+xy)^4}$

c) $z = xy + (3-x)y - y$

d) $z = 2y$

Eine einfache Umformung zeigt, dass die Ausdrücke c) und d) übereinstimmen. Beispiel:

```
Bitte x eingeben
35,81
Bitte y eingeben
2e10
1) 5.126559999971359E23
2) 1.950625760738598E-24
3) 4.0E10
4) 4.0E10
```

Probieren Sie aus, welche Werte Sie für diese Ausdrücke erhalten, wenn Sie `x = 1,5e30` und `y = 2e10` eingeben. Wie erklärt sich die auffallende Diskrepanz zwischen Ergebnissen von c und d in diesem Fall, obwohl x und y und sämtliche Zwischenergebnisse hier noch weit innerhalb der double-Gültigkeit liegen?

2.6 ( ) Wahr oder falsch *(Bezeichner)*

a. Java Operatoren werden von links nach rechts ausgewertet.
b. Das sind gültige Variablennamen: `under_bar , m9281245, t5, j7, her_sales$, his_account$,_total, a, b$, c, z, z2`
c. Das sind ungültige Variablennamen: `3g, 87, 67h2, h22, 2h`
d. Ein gültiger Java Ausdruck ohne Klammern wird von links nach rechts ausgewertet.

2.7 ( ) Vervollständigen Sie die folgenden Statements *(Variable, Präzedenz)*

a. Welche arithmetischen Operationen haben dieselbe Präzedenz wie Multiplikation? ____
b. Eine Adresse im Computerspeicher, der verschiedene Werte zu verschiedenen Zeitpunkten speichern kann, heißt _____

[1] ACHTUNG: Bei der Eingabe einer reellen Zahl muss `,` statt `.` verwendet werden!!

2.8 ( ) Wahr oder falsch *(Operator Präzedenz)*

1. Gegeben ist folgende Gleichung: $y = ax_3 + 7$. Welche der folgenden Anweisungen ist korrekt?
   a. y = a * x *x *x +7;
   b. y = a * x *x *(x +7);
   c. y = (a * x) *x *(x +7);
   d. y = (a * x) *x *x +7;
   e. y = a * (x *x *x) +7;
   f. y = a * x * (x *x +7);

2. Benennen Sie, in welcher Reihenfolge die Operatoren ausgewertet werden und berechnen Sie den Wert von x:

   a. x = 7 + 3 * 6 / 2 – 1;
   b. x = 2 % 2 + 2 * 2 – 2 / 2;
   c. x = (3 * 9 * (3 + (9* 3 / (3))));

2.9 ( ) Blöcke *(Lebensdauer, Sichtbarkeit)*

Gegeben sind die Variablen `x, y, z` vom Typ `integer` und die Variable `f, g, h` vom Typ `float`. Deklarieren Sie drei Blöcke A, B, C wie folgt: `x, y` werden im äußersten Block A mit 5 initialisiert. Der äußere Block enthält einen inneren Block B, der die Variablen `f, g` deklariert und initialisiert. (Den Initialwert können Sie selber bestimmen). Weisen Sie im Block B der Variablen `x` den Wert 20 zu. Der Block B enthält einen inneren Block C, in dem die Variablen `z` und `h` deklariert und initialisiert werden. Am Ende von Block A weisen Sie `h` den Wert `5.0f` zu. Am Ende von Block B weisen Sie der Variablen y den Wert 3000 zu. Am Ende von Block A weisen Sie der Variablen z den Wert `4.0f` zu.
a. Setzen Sie die Beschreibung gemäß Java Syntax um
b. Korrigieren Sie die Fehler und erklären an dem Beispiel die Lebensdauer und Sichtbarkeit jeder einzelnen Variablen.

2.10 (*) Inkrements *(Inkrement Anweisungen)*

1. Schreiben Sie vier verschiedene Java Anweisungen auf, die zur `integer` Variabel `x` 1 aufaddieren.
2. Schreiben Sie Java Anweisungen für folgende Aufgaben:
   a. Weisen Sie die Summe von `x` und `y z` zu und erhöhen den Wert von `x` um 1 nach der Berechnung. Benutzen Sie nur eine Anweisung.
   b. Testen Sie, ob der Wert von `count` größer als 10 ist. Falls dies der Fall ist, geben Sie auf der Console „`Count is greater than 10`“ aus.
   c. Erniedrigen Sie die Variable `x` um 1 und subtrahieren Sie anschließend von der Variable `total`.
   d. Berechnen Sie den Rest der Division `q` geteilt durch `divisor` und weisen das Resultat `q` zu. Schreiben Sie eine Anweisung auf zwei verschiedenen Arten.

3.Bestimmen Sie den Wert jeder Variablen nach Ausführung aller Anweisungen.
Annahme: Zu Beginn der Ausführung haben alle Variablen den Wert 5:

```
a. product *= x++;
b. quotient /= ++x;
```

2.11 ( ) Double/Float Casts *(Reelle Datentyp Konvertierung, Genauigkeit)*

Schreiben Sie ein Java Programm, das zwei `integer` Variablen `zaehler` und `nenner` deklariert und intialisieren `zaehler` mit 14 und `nenner` mit 4. Deklarieren Sie eine Variable `quotient` vom Typ `float`.
a. Berechnen Sie sich den Wert von `quotient` bei ganzzahliger Division von `zaehler` und `nenner` und lassen sich das Ergebnis ausgeben.
b. Berechnen Sie sich den Wert von `quotient,` wenn Sie das Ergebnis der ganzzahligen Division von `zaehler` und `nenner` nach float casten und lassen sich das Ergebnis ausgeben.
c. Berechnen Sie sich den Wert von `quotient,` wenn Sie einen der beiden Operanden `zaehler` oder `nenner` nach `float` casten und lassen sich das Ergebnis ausgeben.

2.12 (*) Lesbarkeit eines Programmes *(Code Conventions)*

Gegeben ist folgendes Programm:

```
public class Raetsel{public static void main(String[] args){
double a; int d; a=5; int b=9; d=Integer.parseInt(args[0]);
double c; System.out.println( "Input: " + d ); c = a/b;
int h1=32; a=d-h1; a=a*c; System.out.println("Output: "+a);}}
```

Unglücklicherweise ist der Autor dieses verunglückten - dafür aber korrekt rechts- und linksbündig gesetzten ;-) - Programms unbekannt verzogen.

Sie sollen herausfinden, wofür dieses Programm gedacht ist. Formatieren und kommentieren Sie dazu den Programmcode neu und machen sie sich die berechnete Funktion (formal) klar. Vereinfachen Sie das Programm und versehen Sie es mit Dokumentation, so dass aus dieser und dem Programmtext selbst die berechnete Funktion und die korrekte Funktionsweise offensichtlich wird.

2.13 (**) Zusatzaufgabe: Mathematische Funktionen *(Math Funktionen)*

Schreiben Sie ein Java Programm, das folgende Variablen definiert: `n, m` vom Typ `int`, `x, y` vom Typ `double; a0, a1, a2, a3, a4, a5, a6` vom Typ `double`, die mit den Werten 1.0, 2.5, 0.1, 0.0, 0.8, 0.0, 1.5

Berechnen Sie mit möglichst wenigen Rechenoperationen der Reihe nach die folgenden in mathematischer Notation geschriebenen Ausdrücke für y und geben das Ergebnis auf der Konsole aus (Hinweis: Hilfsvariable dürfen nach Bedarf definiert werden):

Ausdruck 1: $y = \log(|n - m| x)$

Ausdruck 2: $y = \sin(\frac{n+m}{2} x) - \cos(\frac{n-m}{2} x)$

Ausdruck 3: $y = \frac{1}{2} \frac{n-m}{n+m} x\, e^{-(n+m)x^2}$

Ausdruck 4: $y = A_0 - A_1 x + A_2 x^2 - A_3 x^3 + A_4 x^4 - A_5 x^5 + A_6 x^6$ (Die Berechnung soll mit dem Horner- Schema erfolgen)

Ausdruck 5: $y = \frac{(e^x + x)^2}{\frac{1}{2}x + e^x} - e^x$

Zwei Beispiele:

```
>Bitte n eingeben:
8
Bitte m eingeben:
99
Bitte x eingeben:
34
1) y= 8.037220031133012
2) y= -0.250094446176858
3) y= -0.0
4) y= 2.3182757244E9
5) y= 51.0


>Bitte n eingeben:
2
Bitte m eingeben:
5
Bitte x eingeben:
7
1) y= 3.044522437723423
2) y= -0.11582060186913179
3) y= -1.633367711723573E-149
4) y= 178382.7
5) y= 10.511135015708078
```

Lösungshinweis:

Für die einzelnen mathematischen Funktionen verwenden Sie das Paket `Math`. Z.B.

```
double d = 150.0;
double ergebnis = Math.cos(d);
```

liefert den Cosinus Wert von 150.0 und weist der Variablen `ergebnis` das Resultat zu.

2.14 (*) Fakultät *(Algorithmusentwurf, while Schleife)*

Die Fakultätsfunktion einer nicht negativen Zahl (geschrieben n!) und ist definiert als n! = n * (n-1) * (n-2) * ...* 1. Z.B. ist 5! = 5 x*4 * 3 * 2 * 1 = 120. Schreiben Sie ein Java Programm, das eine Zahl n vom Bediener einliest und die Fakultät dieser Zahl n ausgibt. Testen Sie Ihr Programm auch für 20!

```
>Bitte n eingeben:
19
19! = 121645100408832000
```

2.15 (*) Infinity *(Reelle Arithmetik: Unendlich, Null, NaN)*

1. Definieren Sie die beiden Variablen `zaehler` und `nenner` vom Typ `float` und initialisieren `zaehler` mit 2,5 x 10^{27}f sowie `nenner` mit 3,4 x 10^{-12}f. Lassen Sie sich den Quotienten von beiden mit `System.out` ausgeben und studieren das Verhalten.
2. Testen Sie folgende Anweisungen und lesen die Erklärungen nach z.B. http://openbook.galileocomputing.de/javainsel/ :

```
a.   System.out.println( Math.sqrt(-4) );  //Wurzel von -4
     System.out.println( 0.0 / 0.0);       //Division von 0
b.   System.out.println(  1E300 * 1E20 );
     System.out.println( -1E300 * 1E20 );
```

2.16 (**) Boolesche Ausdrücke *(Boolesche Algebra, Logische Operatoren, if-else)*

Schreiben Sie ein Konsole- Programm "`LogicalExpressions`" , das folgendes leistet:

a) Es definiert vier Variable `a, b, c, d` vom Typ `int` sowie die Hilfsvariable `result` vom Typ `boolean`. Weitere Hilfsvariable dürfen nach Bedarf definiert werden.

b) Es lässt Werte für `a, b, c, d` über die Konsole eingeben.

c) Es berechnet der Reihe nach die Wahrheitswerte der folgenden Aussagen und legt diese in der Hilfsvariablen `result` ab. Nach jeder Berechnung wird `result` mit geeigneter Beschriftung am Bildschirm ausgegeben.

Aussage 1: Mindestens eine der Variablen a, b, c, d hat einen Wert > 1.

Aussage 2: Mindestens eine, jedoch höchstens drei der Variablen a, b, c, d haben einen Wert > 1.

Aussage 3: Genau eine der Variablen a, b, c, d hat einen Wert <0.

Aussage 4: Alle Variablen a, b, c, d, deren Werte > 0 sind, sind auch > 10.

Testbeispiele:

Eingaben a b c d	Aussage 1	Aussage 2	Aussage 3	Aussage 4
-1 -2 0 15	true	true	false	true
3 4 -1 8	true	true	true	false
-6 15 -8 11	true	true	false	true
2 15 18 20	true	false	false	false
-5 -4 -3 -2	false	false	false	true
-4 -5 3 -2	true	true	false	false
20 30 20 40	true	false	false	true
1 0 1 0	false	false	false	false
-1 0 0 0	false	false	true	true

Lösungshinweis: Für jede Aussage dürfen Sie höchstens eine `if else` Kontrollstuktur verwenden.

2.17 (*) Zusatzaufgabe: Primzahlen (Algorithmusentwurf, Schleife, modulo)

Schreiben Sie eine Anwendung, welche eine Liste aller Primzahlen von 1 bis 1000 ermittelt und diese auf der Konsole ausgibt. [Chr13]

Lösungshinweis :

- **Der Modulo-Operator („%“) ist hierbei hilfreich.**
- **Zur Lösung dieser Aufgabe müssen Sie eine Schleife verwenden. Lesen Sie dazu auch Kapitel 3.3 im Vorlesungsskript.**

[Chr13] Christian Silberbauer: *Einstieg in Java und OOP (eXamen.press) (German Edition)*, Springer 2009, 2009, 180 Seiten, ISBN 3540786155