Programmiervorkurs Tag 2 - Lösungen

Aufgabe 2.1

Bei dieser Aufgabe solltet ihr, falls vorhanden, immer zuerst Unäre Operatoren und Klammern auswerten und dann von links nach rechts ausrechnen.

(a) `true & false = false`

(b) `true & true = true`

(c) `false & false = false`

(d) `false | false = false`

(e) `false | true = true`

(f) `true | true = true`

(g) `false ^ false = false`

(h) `true ^ false = true`

(i) `true ^ true = false`

(j) `true & !true = true & false = false`

(k) `(true | false) & (false | (true & true & true | false)) | !false`
Hier ist schnell klar, dass durch das ' | true ' am Ende das Ganze auf jeden Fall wahr ist. Hier trotzdem die komplette Rechnung:

```
true & (false | (true & true & true | false))| true
 = true & (false | (true & true | false))| true
 = true & (false | (true | false))| true
 = true & (false | true)| true
 = true & true | true
 = true | true = true
```

(l) `1 == 2 && (3 != 4) || (1 < 2)`
Hier könnte die Bindungsstärke relevant sein, wir haben aber Glück und && bindet stärker als ||. Falls ihr soetwas schreibt bitte klammern: false && true || true = false || true = true

(m) `(7 < 5) ^ (2 == (9 / 4))  =  false ^ (2 == 2) = false ^ true = true`

Aufgabe 2.2

Welche der folgenden Aussagen sind richtig (hier ist keine Programmierung gefragt - einfach mit Stift und Papier beantworten)

Bei dieser Aufgabe kann es sinnvoll sein sich Wahrheitstabellen aufzustellen. Mit der Folgenden kann man z.B. die Aufgabe 2 ganz gut lösen:

a	**b**	**&**	**|**	**!&**	**!|**	**& | !&**
0	0	0	0	1	1	1
0	1	0	1	1	0	1
1	0	0	1	1	0	1
1	1	1	1	0	0	1

(a) Wenn `(a || b)` true ist, ist mindestens eine der Variablen true.
-> stimmt

(b) Wenn `(a && b)` true ist, sind beide Variablen true.
-> stimmt

(c) Wenn `(a && b)` false ist, sind beide Variablen false.
-> stimmt nicht

(d) Wenn `(a || b)` false ist, sind beide Variablen false.
-> stimmt

(e) Wenn `!(a && b)` false ist, ist keine der beiden Variablen true.
-> stimmt nicht

(f) Wenn `!(a || b)` false ist, sind beide Variablen true.
-> stimmt nicht

(g) `(a && b)|| !(a && b)` ist immer true.
-> stimmt

Aufgabe 2.3

Berechne 1000000000000 + 0,00001 sowie 10000 + 0,0001 mit Java.

Versuche es jeweils einmal mit double und einmal mit float als Variablentyp. Wieso kommt dieses Ergebnis zustande? (Hinweis: Erinnere dich, wie du sicherstellen kannst, dass ein Wert einen bestimmten Datentyp hat.)

Der kleinere Wert sollte immer geschluckt werden, außer bei den kleineren Zahlen mit double. Es muss darauf geachtet werden, dass ein Wert durch Anhängen von .0 zu double wird, durch das Anhängen von f wird er zum float.

Aufgabe 2.4

Gib eine Bedingung an, die für einen int-Wert x wahr wird, wenn dieser über 25 liegt und eine ungerade Zahl ist. (Lösungshinweis: Zur Prüfung, ob es sich um eine ungerade Zahl handelt, ist der Modulo-Operator % hilfreich.)

```
(x > 25) && (x % 2 != 0)
```

Aufgabe 2.5

Ein Schaltjahr ist ein Jahr, dessen Jahreszahl restlos durch vier teilbar ist. Eine Ausnahme bilden die Jahre, deren Jahreszahl durch 100 teilbar ist: Diese sind nur dann ein Schaltjahr, wenn ihre Jahreszahl auch durch 400 teilbar ist.
Gib einen booleschen Ausdruck an, der für einen Integer-Wert jahr genau dann wahr wird, wenn es sich bei dem in jahr gespeicherten Wert um die Jahreszahl eines Schaltjahrs handelt.

```
(jahr % 4 == 0 && jahr % 100 != 0) || jahr % 400 == 0
```

Aufgabe 2.6

Ändere den folgenden Programmteil so, dass es Frauen und Männer bei der Begrüßung unterscheidet:

```
bool male = true;
String name = "Peter";

Console.Writeln("Guten␣Tag,␣Herr␣" + name);
```

```
bool male = true;
String name = "Peter";

if (male) {
    Console.Writeln("Guten␣Tag,␣Herr␣" + name);
} else {
    Console.Writeln("Guten␣Tag,␣Frau␣" + name);
}
```

Aufgabe 2.7

Mit Hilfe der p-q-Formel lassen sich quadratische Gleichungen der Form $x^2 + p \cdot x + q = 0$ *lösen. Es gilt:* $x_{1/2} = -\frac{p}{2} \pm \sqrt{\left(\frac{p}{2}\right)^2 - q}$. *Eine reelle Lösung existiert nur dann, wenn die Diskriminante (der Ausdruck unter der Wurzel) nicht negativ ist.*
Schreibe ein Programm, das das Ergebnis/die Ergebnisse für gegebene p und q berechnet und auf der Konsole ausgibt. Wenn es kein Ergebnis im Bereich der reellen Zahlen gibt, soll eine entsprechende Meldung auf der Konsole ausgegeben werden.
Um Wurzeln und Potenzen zu berechnen, kannst du die Methoden sqrt() und pow() der Klasse Math verwenden. Beide Methoden nehmen double-Werte als Parameter entgegen und geben einen double-Wert zurück, den man zum Beispiel in einer Variablen speichern kann.
Bsp.:

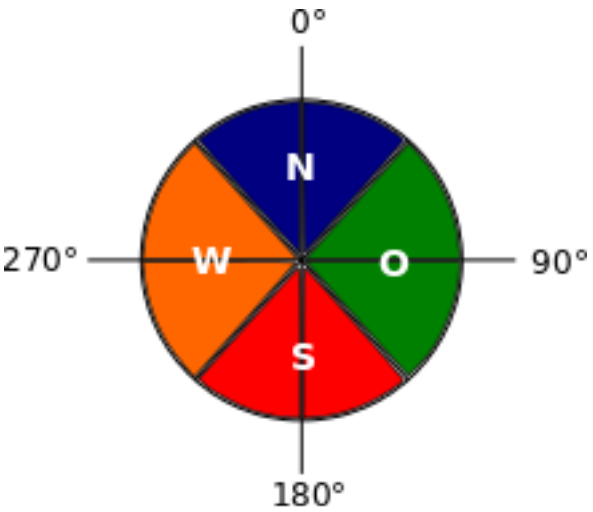
- *Math.sqrt(4.0) gibt den Wert 2.0 zurück*
- *Math.pow(2.0, 3.0) gibt den Wert 8.0 zurück*

```
double p = 0.0;
double q = 0.0;
double diskriminante = Math.pow(p * 0.5, 2.0) - q;
double x1;
double x2;

if (diskriminante < 0) {
    Console.Writeln("Für␣die␣gegebenen␣p␣und␣q␣existiert␣
        keine␣Loesung␣im␣Bereich␣der␣reellen␣Zahlen!");
} else if (diskriminante == 0){
    x1 = p * -0.5;
    Console.Writeln("x:␣" + x1);
} else {
    x1 = p * -0.5 + Math.sqrt(diskriminante);
    x2 = p * -0.5 - Math.sqrt(diskriminante);
    Console.Writeln("x1:␣" + x1);
    Console.Writeln("x2:␣" + x2);
}
```

Aufgabe 2.8

Schreibe ein Programm, das abhängig von einem Winkel (positiver Integer-Wert) eine der vier Himmelsrichtung Norden, Osten, Süden oder Westen ausgibt. Norden liegt dabei auf 0°.
Es reicht, wenn Winkel von 0° *bis* 359° *verarbeitet werden können.*




```
int winkel = 0;

if (winkel >= 0 && winkel < 45 || winkel > 315 && winkel
    < 360) {
   Console.Writeln("Norden");
} else if (winkel => 45 && winkel < 135) {
   Console.Writeln("Osten")
} else if (winkel => 135 && winkel < 225) {
   Console.Writeln("Sueden");
} else if (winkel => 225 && winkel < 315) {
   Console.Writeln("Westen");
} else {
   Console.Writeln("Ungueltiger␣Winkel!␣Bitte␣einen␣Wert␣
       zwischen␣0␣und␣359␣wählen!");
}
```

Aufgabe 2.9

Schreibe ein Programm, das das gleiche Ziel ganz ohne if-Abfragen nur mit switch/case erreicht.
Lösungshinweise: Normiere den Winkelbereich auf die Anzahl der Himmelsrichtungen. Jede Himmelsrichtung wird dann durch einen von vier Integer-Werten repräsentiert, der als Switch-Variable dient. Dabei ist die ganzzahlige Division hilfreich.
Du solltest dir außerdem überlegen, was man gegen die „unschöne“ Aufteilung der Winkelbereiche tun kann. Jeder dieser Bereiche ist zwar $90°$ *groß, aber Norden liegt zum Beispiel zwischen* $315°$ *und* $45°$*, Süden zwischen* $135°$ *und* $225°$*. Das ist zum Normieren etwas unpraktisch.*
Vier case statements sollten reichen.

```
int winkel = 0;
int normWinkel = ((winkel + 45) % 360) / 90;
switch(normWinkel){
   case 0:
      Console.Writeln("Norden");
      break;
   case 1:
      Console.Writeln("Osten");
      break;
   case 2:
      Console.Writeln("Sueden");
      break;
   case 3:
      Console.Writeln("Westen");
      break;
}
```