Übungsaufgaben Extra

Zählschleife / Kopfgesteuerte & Fussgesteuerte Schleife

For-Schleife (Kopfgesteuerte Schleife)

Aufgabe 26

Erstellen Sie ein ausführbares Programm mit dem Namen **ForSchleife.java.** Das Programm gibt die Zahlen von 1 bis 10 durch Komma getrennt auf dem Bildschirm aus:

1, 2, 3, 4, 5, 6, 7, 8, 9, 10,

Hinweis: Mit Schleifen können Quelltextzeilen wiederholt ausgeführt werden. Wenn mehrere Zeilen ausgeführt werden sollen, müssen diese in einem Block { } eingebunden werden. Wenn lediglich eine Quelltextzeile wiederholt werden soll, kann auf die geschweiften Klammern verzichtet werden.

Aufgabe 27

Erstellen Sie ein ausführbares Programm mit dem Namen **QuadratZahlen.java.** Das Programm berechnet die Quadratzahlen von 1 bis 40, ermittelt zusätzlich die Summe der Quadratzahlen und gibt diese auf dem Bildschirm aus.

1, 4, 9, 16, 25, 36, 49, 64, 81, 100,
121, 144, 169, 196, 225, 256, 289,
324, 361, 400, 441, 484, 529, 576,
625, 676, 729, 784, 841, 900, 961,
1024, 1089, 1156, 1225, 1296,
1369, 1444, 1521, 1600,
Summe = 22140

Hinweis: Um zwischengespeicherte Werte zu summieren eignet sich die Formel:
summe = summe + quadratZahl ;

Aufgabe 28

Erstellen Sie ein ausführbares Programm mit dem Namen **PotenzenTabelle.java.** Das Programm soll unten stehende Bildschirmausgabeerzeugen:

```
X     x²   x³
     --------------------
1     1
2     4    8
3     9    27
4     16   64
5     25   125
6     36   216
7     49   343
```

Hinweis: Sie könnten durch eine weitere Schleife auch die 19 Bindestriche erzeugen.

Aufgabe 29

Erstellen Sie ein ausführbares Programm mit dem Namen **LetztesKommaWeg.java.** Das Programm gibt die Zahlen von 1 bis 10 durch Komma getrennt auf dem Bildschirm aus. (Wie in Aufgabe 26) Das letzte Komma soll nicht erscheinen!

1, 2, 3, 4, 5, 6, 7, 8, 9, 10

Hinweis: Es gibt 2 Lösungsvorschläge:
1. Lösungsvorschlag: mit einer Bedingungsanweisung
2. Lösungsvorschlag: den letzten Wert außerhalb der Schleife ausgeben

Bemerkung: Sehen Sie sich den Lösungsvorschlag 2 genauer an und vergleichen Sie ihn mit dem Lösungsvorschlag 1:
Die Deklaration der Integer Variable i erfolgt außerhalb der Schleife, damit unterhalb der Schleife auf den letzten Wert (10) zugegriffen werden kann. Beachten Sie auch, dass die Schleifenbedingung led. bis <10 erfüllt ist. Beim letzten Schleifendurchlauf wird noch mit i++ die Variable i auf 10 hochgesetzt. Damit ist die Schleifenbedingung <10 nicht mehr erfüllt. Die Variable i hat aber zu diesem Zeitpunkt den Wert 10, der dann auch außerhalb der Schleife (ohne Komma) ausgegeben wird.

Aufgabe 30

Erstellen Sie ein ausführbares Programm mit dem Namen **WuerfelFor.java.** In Ihrem Programm soll solange gewürfelt werden, bis zum ersten Mal die Zahl 6 gewürfelt wird. Natürlich ist es auch möglich, dass mehr als dreimal gewürfelt werden muss. Im besten Fall muss led. einmal gewürfelt werden.

Wurf 1: 4
Wurf 2: 3
Wurf 3: 6

Hinweis: Bei einer for-Schleife können auch Initialisierung, Bedingungsanweisung und Inkrementation (hochzählen) bzw. Dekrementation (runterzählen) weggelassen werden. Die kürzeste for-Schleife könnte so aussehen: for(;;). Das wäre dann eine Endlosschleife, wenn nicht im Ausführungsteil bzw. Ausführungsblock eine Abbruchbedingung erfolgt.

Aufgabe 31

10 x 10 – X-Block (Schleifen) Erstellen Sie ein ausführbares Programm mit dem Namen **SchleifenBlock.java.** Ihr Quelltext darf led. einmal den Buchstaben X enthalten. Das Programm erzeugt untenstehende Bildschirmausgabe:

```
x x x x x x x x x x
x x x x x x x x x x
x x x x x x x x x x
x x x x x x x x x x
x x x x x x x x x x
x x x x x x x x x x
x x x x x x x x x x
x x x x x x x x x x
x x x x x x x x x x
x x x x x x x x x x
```

Hinweis: Es gibt 2 Lösungsvorschläge:
1. Lösungsvorschlag: 2 Schleifen (eine innere für die Spalte und eine äußere für die Zeile.
2. Lösungsvorschlag: Mit led. einer Schleife. Der Zeilenumbruch wird mit einer Bedingungsanweisung und dem Modulo Operator vorgenommen.

Aufgabe 32

Erstellen Sie ein ausführbares Programm mit dem Namen **ZahlenDreieck.java.** Verwenden Sie in Ihrem Programm zwei Schleifen, welche aufgrund der Deklaration

```
zeilen = 8 ;
```

folgende Bildschirmausgabe erzeugt:

```
1: 1
2: 2 1
3: 3 2 1
4: 4 3 2 1
5: 5 4 3 2 1
6: 6 5 4 3 2 1
7: 7 6 5 4 3 2 1
8: 8 7 6 5 4 3 2 1
```

Aufgabe 33

Erstellen Sie ein ausführbares Programm mit dem Namen **ASCIITabelle.java.** Das Programm gibt die ASCII-Zeichen von 1 bis 255 auf dem Bildschirm aus.
Aus Übersichtlichkeitsgründen soll nach maximal 6 Zeichen ein Zeilenumbruch erfolgen:

32 =	33 = !	34 = "	35 = #	36 = $	37 = %
38 = &	39 = '	40 = (	41 =)	42 = *	43 = +
44 = ,	45 = -	46 = .	47 = /	48 = 0	49 = 1
50 = 2	51 = 3	52 = 4	53 = 5	54 = 6	55 = 7
56 = 8	57 = 9	58 = :	59 = ;	60 = <	61 = =
62 = >	63 = ?	64 = @	65 = A	66 = B	67 = C
68 = D	69 = E	70 = F	71 = G	72 = H	73 = I
74 = J	75 = K	76 = L	77 = M	78 = N	79 = O
80 = P	81 = Q	82 = R	83 = S	84 = T	85 = U
86 = V	87 = W	88 = X	89 = Y	90 = Z	91 = [
92 = \	93 =]	94 = ^	95 = _	96 = `	97 = a
98 = b	99 = c	100 = d	101 = e	102 = f	103 = g
104 = h	105 = i	106 = j	107 = k	108 = l	109 = m
110 = n	111 = o	112 = p	113 = q	114 = r	115 = s
116 = t	117 = u	118 = v	119 = w	120 = x	121 = y
122 = z	123 = {	124 = |	125 = }	126 = ~	127 = □
128 = ?	129 = ?	130 = ?	131 = ?	132 = ?	133 = ?
134 = ?	135 = ?	136 = ?	137 = ?	138 = ?	139 = ?
140 = ?	141 = ?	142 = ?	143 = ?	144 = ?	145 = ?
146 = ?	147 = ?	148 = ?	149 = ?	150 = ?	151 = ?
152 = ?	153 = ?	154 = ?	155 = ?	156 = ?	157 = ?
158 = ?	159 = ?	160 =	161 = ¡	162 = ¢	163 = £
164 = ¤	165 = ¥	166 = ¦	167 = §	168 = ¨	169 = ©
170 = ª	171 = «	172 = ¬	173 =	174 = ®	175 = ¯
176 = °	177 = ±	178 = ²	179 = ³	180 = ´	181 = µ
182 = ¶	183 = ·	184 = ¸	185 = ¹	186 = º	187 = »
188 = ¼	189 = ½	190 = ¾	191 = ¿	192 = À	193 = Á
194 = Â	195 = Ã	196 = Ä	197 = Å	198 = Æ	199 = Ç
200 = È	201 = É	202 = Ê	203 = Ë	204 = Ì	205 = Í
206 = Î	207 = Ï	208 = Ð	209 = Ñ	210 = Ò	211 = Ó
212 = Ô	213 = Õ	214 = Ö	215 = ×	216 = Ø	217 = Ù
218 = Ú	219 = Û	220 = Ü	221 = Ý	222 = Þ	223 = ß
224 = à	225 = á	226 = â	227 = ã	228 = ä	229 = å
230 = æ	231 = ç	232 = è	233 = é	234 = ê	235 = ë
236 = ì	237 = í	238 = î	239 = ï	240 = ð	241 = ñ
242 = ò	243 = ó	244 = ô	245 = õ	246 = ö	247 = ÷
248 = ø	249 = ù	250 = ú	251 = û	252 = ü	253 = ý
254 = þ	255 = ÿ				

Hinweis: ASCII steht für American Standard Code für Information Interchange

Aufgabe 34

Erstellen Sie ein ausführbares Programm mit dem Namen **Permutation.java**. Das Programm soll alle möglichen Kombinationen der Zahlen von 1 bis 4 simulieren. Dabei darf keine Zahl doppelt in der 4-stelligen Kombination vorkommen. Das Programm erzeugt folgende Bildschirmausgabe:

1 2 *3 4*
1 2 *4 3*
1 3 *2 4*
1 3 *4 2*
1 4 *2 3*
1 4 *3 2*
2 1 3 4
2 1 4 3
2 3 1 4
2 3 4 1
2 4 1 3
2 4 3 1
3 1 2 4
3 1 4 2
3 2 1 4
3 2 4 1
3 4 1 2
3 4 2 1
4 1 2 3
4 1 3 2
4 2 1 3
4 2 3 1
4 3 1 2
4 3 2 1
24 Möglichkeiten

While-Schleife (Kopfgesteuerte Schleife)

Aufgabe 33

Erstellen Sie ein ausführbares Programm mit dem Namen **WhileSchleife.java**. Das Programm gibt die Zahlen von 1 bis 10 durch Komma getrennt auf dem Bildschirm aus:

1, 2, 3, 4, 5, 6, 7, 8, 9, 10,

Hinweis: Die While-Schleife benötigt die geschweiften Klammern { } für einen Block, da neben der Ausgabe auch hochgezählt werden muss.

Aufgabe 34

Erstellen Sie ein ausführbares Programm mit dem Namen **Dual2Dez.java**. Das Programm rechnet eine Dualzahl in eine Dezimalzahl um und gibt diese auf dem Bildschirm aus. Bei der fett und kursiv dargestellten Zahl handelt es sich um eine Benutzereingabe.

Bitte geben Sie die Dualzahl ein: ***110101101***
Die Dualzahl 110101101 hat den Dezimalwert: 429

Aufgabe 35

Erstellen Sie ein ausführbares Programm mit dem Namen **ZahlStellenZaehlen.java**. Das Programm zählt die Ziffern der eingegebenen Zahl und gibt diese auf dem Bildschirm aus. Bei der fett und kursiv dargestellten Zahl handelt es sich um eine Benutzereingabe.

Von welcher Zahl sollen die Anzahl der Stellen gezählt werden: ***21355746***
Die Zahl hat 8 Stellen.

Hinweis: Es bietet sich eine While- oder Do-While-Schleife an, weil zu Beginn des Programms nicht bekannt ist, wie oft die Schleife wiederholt werden muss.

Aufgabe 36

Erstellen Sie ein ausführbares Programm mit dem Namen **ZahlUmdrehen.java**. Das Programm dreht eine Ganzzahl um und gibt diese auf dem Bildschirm aus. Bei der fett und kursiv dargestellten Zahl handelt es sich um eine Benutzereingabe.

Welche Zahl soll umgedreht werden: ***153536758***
Die umgedrehte Zahl lautet: 857635351

Aufgabe 37

Erstellen Sie ein ausführbares Programm mit dem Namen **WuerfelWhile.java**. In Ihrem Programm soll solange gewürfelt werden, bis zum ersten Mal die Zahl 6 gewürfelt wird. Natürlich ist es auch möglich, dass mehr als dreimal gewürfelt werden muss. Im besten Fall muss led. einmal gewürfelt werden.

Wurf 1: 4
Wurf 2: 3
Wurf 3: 6

Aufgabe 38

Erstellen Sie ein ausführbares Programm mit dem Namen **GeldAutomat.java**. Das Programm erzeugt untenstehende Bildschirmausgabe. Bei der fett und kursiv dargestellten Zahl handelt es sich um eine Benutzereingabe. Ihr Programm ermittelt anhand der Benutzereingabe die kleinstmögliche Anzahl an Geldscheinen.

Bitte geben sie den auszuzahlenden Geldbetrag ein: ***995***
*1*500*
*2*200*
*0*100*
*1*50*
*2*20*
*0*10*
*1*5*

Hinweis: Orientieren Sie sich an den aktuell erhältlichen Eurogeldscheinen. Und bedenken sie, das die Zahl durch 5 teilbar sein sollte.

Aufgabe 39

Erstellen Sie ein ausführbares Programm mit dem Namen **GroessterGemeinsamerTeiler.java**. Das Programm erzeugt untenstehende Bildschirmausgabe. Bei den fett und kursiv dargestellten Zahlen handelt es sich um eine Benutzereingabe.

*Zähler:**200***
*Nenner: **80***
Der größte gemeinsame Teiler ist: 40

Hinweis: Den größten gemeinsamen Teiler benötigt man zum Kürzen von Brüchen. Beispielsweise könnte der Bruch 80/200 jeweils mit 40 gekürzt werden. Das Ergebnis ist dann 2/5. Diesen Bruch kann man nicht weiter kürzen.

Do-While-Schleife (Fußgesteuerte Schleife)

Aufgabe 40

Erstellen Sie ein ausführbares Programm mit dem Namen **DoWhileSchleife.java**. Das Programm gibt die Zahlen von 1 bis 10 durch Komma getrennt auf dem Bildschirm aus:

1, 2, 3, 4, 5, 6, 7, 8, 9, 10,

Hinweis: Die Do-While-Schleife benötigt die geschweiften Klammern { } für einen Block, da neben der Ausgabe auch hochgezählt werden muss.

Aufgabe 41

Erstellen Sie ein ausführbares Programm mit dem Namen **NotenDurchschnitt.java**. Schreiben ein Programm, das vom Benutzer die Eingabe mehrerer Noten (von 1-6) verlangt. Bei Eingabe einer ungültigen Note (außerhalb von 1- 6) soll eine Fehlermeldung ausgegeben werden. Bei Eingabe von 0.0 gibt das Programm die Summe der eingegebenen Werte auf dem Bildschirm aus und den Notendurchschnitt.

*Note eingeben: **3,1***
*Note eingeben: **1,7***
*Note eingeben: **1,5***
*Note eingeben: **2,1***
*Note eingeben: **0,0***
Notensumme = 8.4,
Notendurchschnitt = 2.1

Aufgabe 42

Erstellen Sie ein ausführbares Programm mit dem Namen **WuerfelDo.java**. In Ihrem Programm soll solange gewürfelt werden, bis zum ersten Mal die Zahl 6 gewürfelt wird. Natürlich ist es auch möglich, dass mehr als dreimal gewürfelt werden muss. Im besten Fall muss led. einmal gewürfelt werden.

Wurf 1: 4
Wurf 2: 3
Wurf 3: 6

Aufgabe 43

Erstellen Sie ein ausführbares Programm mit dem Namen **ZahlenRaten.java**. Das Programm erzeugt untenstehende Bildschirmausgabe. Bei den kursiv und fett dargestellten Werten (65, 30, 48 und 40) handelt es sich um Eingaben. Ziel des Spiels ist es, eine Zahl zu erraten, die ihr Programm generiert hat.
Verwenden Sie zum Erzeugen einer Zufallszahl die Methode random() der Klasse Math.

ZAHLENRATEN Errate eine Zahl zwischen 1 und 128
1. Versuch: ***65***
Meine Zahl ist kleiner!
2. Versuch: ***30***
Meine Zahl ist groesser!
3. Versuch: ***48***
Meine Zahl ist kleiner!
4. Versuch: ***40***
Du hast meine Zahl beim 4. Versuch erraten.

Hinweis:
Math.random() erzeugt eine Zufallszahl zwischen 0 und 0,9999999.
Um eine Zufallszahl zwischen 1 und 128 zu erzeugen, können Sie einen erzwungen Typecast anwenden: (int)(Math.random()*128+1);

Zunächst wird die Zufallszahl mit 128 multipliziert. Die kleinste Zahl zu diesem Zeitpunkt ist 0 und die größte 127,9999.
Danach wird 1 addiert. Nun liegt das Zufallsspektrum zwischen 1 und 128,9999.
Durch den Typecast (int) werden die Kommastallen abgeschnitten. Die Zufallszahl liegt somit zwischen 1 und 128.

Aufgabe 44

Erstellen Sie ein ausführbares Programm mit dem Namen **KennwortAbfrage1.java**. Das Programm verlangt solange eine Abfrage für ein Kennwort, bis es richtig eingegeben wurde. Bei den kursiv und fett dargestellten Worten hallo und abc handelt es sich um Eingaben:

Kennwort: ***hallo***
Kennwort: ***abc***
Kennwort ist richtig

Hinweis: Ein Textvergleich muss über die equals-Methode (eingabe.equals(kennwort)) erfolgen.

Aufgabe 45

Erstellen Sie ein ausführbares Programm mit dem Namen **KennwortAbfrage2.java.** Das Programm verlangt solange eine Abfrage für ein Kennwort, bis es richtig eingegeben wurde. Dieses Mal wird mitgezählt, wie oft eine falsche Abfrage gemacht wurde. Nach 3 Fehlversuchen soll untenstehende Bildschirmausgabe erzeugt werden. Bei den kursiv und fett dargestellten Worten bca, cba und acb handelt es sich um Eingaben:

Kennwort: ***bca***
1.)falsch
Kennwort: ***cba***
2.) falsch
Kennwort: ***acb***
3.) falsch Zugriff verweigert!

Andernfalls soll nachfolgende Ausgabe erscheinen:

Kennwort: cba
1.) falsch
Kennwort: abc
Richtig, du hast 2 Versuch(e) benötigt

Hinweis: Mit dem Befehl break können Schleifen vorzeitig verlassen werden.

Aufgabe 46

Erstellen Sie ein ausführbares Programm mit dem Namen **MatheLernprogramm.java.** Das Programm erzeugt untenstehende Bildschirmausgabe. Bei den fett und kursiv dargestellten Zahlen handelt es sich um eine Benutzereingabe. Ihr Programm generiert zufällige Zahlen zwischen 1 und 100 und prüft, ob Sie das richtige Ergebnis eintippen. Das Programm wird mit der Eingabe 0 beendet und gibt dann untenstehende Statistik auf dem Bildschirm aus.

```
11+5 = 16
richtig
65+92 = 157
richtig
79+73 = 153
falsch
48+44 = 0
Zeit: 16 Sekunden.
3 Aufgaben gelöst. 2 waren richtig. Eine Aufgabe war falsch.
```

Hinweis: Mit dem nachfolgenden Quelltextausschnitt kann die Zeit gemessen werden. Zwischen beginn und ende sollte dann der Algorithmus stehen, der zeitlich gemessen werden soll:

```
long beginn, ende;
beginn = System.currentTimeMillis();
ende = System.currentTimeMillis();
System.out.println("Zeit: " +  ((ende - beginn)/1000) + "
Sekunden.");
```

Erweitern Sie das Programm mit mehreren Leveln. Z.B. nach 10 richtigen Eingaben werden Subtraktion, Multiplikation, Division, Modulo etc abgefragt.