AB 3.3-1: for-Schleifen

Aufgabe / Übung 3.3 #1

Informieren Sie sich zunächst über die **switch-case-Anweisung**.

Ersetzen Sie dann die große if-else-Anweisung in der Klasse **DatumRegeln**, Methode **tageImMonat()**, durch eine solche switch-case-Anweisung. Experten versuchen sogar, hier einen switch-case-*Ausdruck* einzusetzen. Testen Sie das Projekt danach, ob weiterhin alles korrekt funktioniert.

Übung 3.3 #2

Erstellen Sie für das Projekt eine neue Klasse **DatumTools**. Schreiben Sie dann eine Methode

```
public int tageSeitJahresanfang(Datum d)
```

die Ihnen die Anzahl der Tage zurückliefert, die seit dem 1. Januar des Jahres d.getJahr() vergangen sind. Hierzu sollten Sie dann eine for-Schleife einsetzen.

Beispiel:

Datum = 13.05.2025

Tage seit Jahresbeginn = 31 (für Januar) + 28 (Februar) + 31 (März) + 30 (April) + 13 (Mai) = 133.

Übung 3.3 #3

Ergänzen Sie die Klasse **DatumTools** durch eine Methode

```
public int tageBisJahresende(Datum d)
```

die Ihnen die Zahl der Tage liefert, die bis zum Ende des Jahres noch vergehen.

Für den 01.01.2025 müssten Sie den Wert 364 erhalten, für den 31.12.2025 den Wert 0.

Übung 3.3 #4

Ergänzen Sie die Klasse **DatumTools** durch eine Methode

```
public int getTagesdifferenz(Datum d1, Datum d2)
```

Diese Methode sollte Ihnen die Anzahl der Tage zurückliefern, die zwischen den beiden Daten liegen.

Übung 3.3 #5

Jetzt noch ein kleiner Kür-Teil für die Experten unter Ihnen. Denken Sie sich selbst auch noch ein oder zwei schöne Aufgaben zum Umgang mit dem Datums-Projekt aus und lösen Sie diese in Partnerarbeit oder Gruppenarbeit. Anschließend präsentieren Sie Ihr Projekt und erklären genau, was Sie gemacht haben.

Übung 3.3 #6

Schreiben Sie eine Methode

```
public getHaeufigkeitBuchstabe(String zeichenkette, char buchstabe)
```

Diese Methode soll zählen, wie oft der angegebene Buchstabe in dem übergebenem String vorkommt. Wenn Sie sich noch nicht so gut mit der Klasse String auskennen, gehen Sie bitte auf die Seite "String" in der Abteilung "Java-Klassen" auf diese Homepage. Dort finden Sie die wichtigsten Methoden der Klasse String.

Übung 3.3 #7

Berechnen Sie die Summe der ersten n Quadratzahlen, die durch drei teilbar sind und der ersten n Kubikzahlen, die durch fünf teilbar sind.

Beispiel:

Mit n = 10 würden wir erhalten

3*3 + 5*5*5 + 6*6 + 9*9 + 10*10*10 = 1251

Aufgabe 3.3 #8

Was macht die folgende for-Schleife ?

```
for (int a = 1; a <= 10; a++)
{
   for (int b = 1; b <= 10; b++)
      System.out.printf("%6d", a*b);
   System.out.println();
}
```

Übung 3.3 #9

Die folgende Konsolenausgabe soll einen Tannenbaum darstellen:

Schreiben Sie eine Java-Methode, die einen solchen Tannenbaum in die Konsole zeichnet.

1. Ergänzung für Profis

Machen Sie Ihre Methode flexibler: man soll die maximale **Breite** des Tannenbaums als Parameter an die Methode übergeben können. Im obigen Beispiel hat der Baum eine maximale Breite von 13.

2. Ergänzung für Super-Profis

Die Methode soll noch flexibler werden: man soll auch die **Höhe** des Stammes als Parameter übergeben können. Im Beispiel ist der Stamm 5 Sterne hoch.

```
     *
    ***
   *****
  *******
 *********
***********
*************
     *
     *
     *
     *
     *
```

Übung 3.3 #10

Erzeugen Sie mithilfe von zwei geschachtelten for-Schleifen ein Schachbrettmuster aus Zeichen in der Konsole.

Übung 3.3 #11

Erstellen Sie eine Klasse Uhr mit den Instanzvariablen stunde und minute (beide vom Typ int).

Die Klasse soll folgende Methoden besitzen:

- public void **stellen**(int h, int m) - stellt die Uhrzeit auf die Stunde und Minute neu ein
- public String **getZeit**() - liefert einen String mit der Uhrzeit zurück, nach dem Muster hh:mm
- public void **anzeigen**() - zeigt den Uhrzeit-String in der Konsole an
- public void **tick**() - erhöht die Uhrzeit um genau eine Minute
- public void **run**() - lässt die Uhr in einer for- oder while-Schleife über 1500 Minuten laufen (**tick()** aufrufen). Alle 10 Minuten soll die Uhrzeit in der Konsole ausgegeben werden.

Die Konsolenausgabe sollte dann so aussehen wie rechts.

Experten erweitern diese Klasse noch um **Sekunden**. Ein Aufruf von **tick()** würde die Uhrzeit dann um eine Sekunde erhöhen.

```
00:10:00
00:20:00
00:30:00
00:40:00
00:50:00
01:00:00
01:10:00
01:20:00
01:30:00
01:40:00
01:50:00
02:00:00
02:10:00
```

Übung 3.3 #12

Schreiben Sie eine Methode `public void iteriere()`

Zunächst soll die Methode mit `double zahl = Math.random() * 10` eine Zufallszahl zwischen 0 und 10 (exklusiv) erzeugen.

Dann soll folgender Algorithmus in einer Schleife wiederholt werden:

- Falls zahl < 1, dann multipliziere Zahl mit 2
- Falls zahl >= 1, dann dekrementiere Zahl um 1.

Die Schleife soll beendet werden, sobald Zahl <= 0 ist oder mehr als 100 Durchläufe erfolgt sind.

Beispiel für zahl = 8.45

8.45 - 7.45 - 6.45 - 5.45 - 4.45 - 3.45 -2.45 - 1.45 - 0.45 - 0.9 - 1.8 - 0.8 - 1.6 - 0.6 - 1.2 - 0.2 - 0.4 - 0.8 - 1.6 - 0.6 - 1.2 - 0.2 - 0.4 - 0.8 - 1.6 - 0.6 etc.

Übung 3.3 #13

Schreiben Sie eine Methode ähnlich wie in 3.3 #12, die die Collatz-Folge ausgibt:

- Falls zahl gerade, zahl → zahl / 2
- Falls zahl ungerade: zahl → 3 * zahl + 1.

Beispiel für zahl = 7:

7 - 22 - 11 - 34 - 17 - 52 - 26 - 13 - 40 - 20 - 10 - 5 - 16 - 8 - 4 - 2 - 1

Übung 3.3 #14

In der Übung 3.2 #4 haben Sie eine Methode erstellt, die die Fakultät einer Zahl berechnet. Schreiben Sie jetzt eine Methode

`public void testeAufFakultaet(int zahl)`

die auf der Konsole ausgibt, ob die übergebene Zahl eine Fakultät ist. Wenn das der Fall sein sollte, muss natürlich auch ausgegeben werden, welche Zahl dieser Fakultät zu Grunde liegt.

Beispiele:

zahl = 362880, Ausgabe der Methode: 362880 ist 9!

zahl = 1000, Ausgabe der Methode: 1000 ist keine Fakultät.