

Übung 2.2.8 #1

Der folgende Quelltext soll die Werte von x und y vertauschen, wenn x größer ist als y.

Überprüfe, ob der Quelltext korrekt funktioniert. Falls das nicht der Fall sein sollte, schreibe den Quelltext so um, dass er korrekt funktioniert.

```
if (x > y)
    int swap = x;
    x = y;
    y = x;
```

```
if (x > y)
{
    int swap = x;
    x = y;
    y = swap;
}
```

Übung 2.2.8 #2

Eine Methode soll Flüssigkeitsmengen in eine leicht lesbare Form umwandeln und als String zurückgeben. Die Flüssigkeitsmenge **in Litern** wird als double-Parameter volumen übergeben.

```
public String getVolumenLesbar(double volumen)
```

Hier ein paar Beispiele für mögliche Ausgaben:

double volumen	String
4.0	"ca. 4 Liter"
4.3	"ca. 4 Liter"
3.8	"ca. 4 Liter"
0.2	"ca. 20 Zentiliter"
0.18	"ca. 18 Zentiliter"
0.0032	"ca. 3 Milliliter"
0.0017	"ca. 2 Milliliter"

```
public String getVolumenLesbar(double volumen)
{
    int vol;

    if (volumen >= 1)
    {
        vol = (int) Math.round(volumen);
        return "ca. " + vol + " Liter";
    }
    else if (volumen >= 0.1)
    {
        vol = (int) Math.round(volumen * 100);
        return "ca. " + vol + " Zentiliter";
    }
    else if (volumen >= 0.001)
    {
        vol = (int) Math.round(volumen * 1000);
        return "ca. " + vol + " Milliliter";
    }
    else
    {
        return "Volumen ist zu klein!";
    }
}
```

Übung 2.2.8 #3

Für ein Reisevergleichsportal sollen Reiserücktritt-Versicherungen von drei Versicherungs-Unternehmen verglichen werden.

- A. Prämie = 50 Euro + 4% vom Reisepreis
- B. Prämie = 5% vom Reisepreis
- C. Prämie = 100 Euro + 3,5% vom Reisepreis

Schreibt eine Methode, die berechnet, welche Versicherung für den gegebenen Reisepreis die günstigste ist. Der Reisepreis wird als `double`-Parameter übergeben, die ausgewählte Versicherung kann als `char`-Wert zurückgegeben werden oder mit `System.out.println()` direkt ausgegeben werden.

```
public char
    getGuenstigsteVersicherung(double preis)
{
    double prA = 50 + 0.04 * preis;
    double prB = 0.05 * preis;
    double prC = 100 + 0.035 * preis;

    char min = 'A';

    if (prB < prA && prB < prC)
        min = 'B';
    else if (prC < prA && prC < prB)
        min = 'C';

    return min;
}
```

Übung 2.2.8 #4 (einfache Version)

Schreibt eine Methode

`public void zeigeJahreszeit(int monat)`

die - abhängig von dem Parameter `monat` - die jeweilige Jahreszeit mit `System.out.println()` ausgibt.

Hinweis: Der Winter umfasst die Monate Dezember bis Februar, der Frühling die Monate März bis Mai, der Sommer den Juni bis August und der Herbst schließlich den September bis November.

```
// Variante 1

public void zeigeJahreszeitV1(int monat)
{
    if (monat == 12 || monat == 1 || monat == 2)
        System.out.println("Winter");
    else if (monat >= 3 && monat <= 5)
        System.out.println("Frühling");
    else if (monat >= 6 && monat <= 8)
        System.out.println("Sommer");
    else if (monat >= 9 && monat <= 11)
        System.out.println("Herbst");
    else
        System.out.println("Ungültiger Monat");
}
```

Übung 2.2.8 #4 (einfache Version)

Schreibt eine Methode

```
public void zeigeJahreszeit(int monat)
```

die - abhängig von dem Parameter `monat` - die jeweilige Jahreszeit mit `System.out.println()` ausgibt.

Hinweis: Der Winter umfasst die Monate

Dezember bis Februar, der Frühling die Monate

März bis Mai, der Sommer den Juni bis August und

der Herbst schließlich den September bis

November.

```
// Variante 2
```

```
public void zeigeJahreszeitV2(int monat)
{
```

```
    if (monat < 1 || monat > 12)
        System.out.println("Ungültiger Monat");
```

```
    else if (monat >= 3 && monat <= 5)
        System.out.println("Frühling");
```

```
    else if (monat >= 6 && monat <= 8)
        System.out.println("Sommer");
```

```
    else if (monat >= 9 && monat <= 11)
        System.out.println("Herbst");
```

```
    else // monat = 12, 1 oder 2
        System.out.println("Winter");
```

```
}
```

Übung 2.2.8 #4 (anspruchsvollere Version)

Schreibt ähnlich wie in der einfachen Version eine Methode

```
public void  
zeigeJahreszeit  
    (int monat, int tag)
```

In dieser anspruchsvolleren Version müsst ihr auch den Parameter `tag` berücksichtigen. Der Winter beginnt nämlich am 21. Dezember, der Frühling am 21. März, der Sommer am 21. Juni und der Herbst am 21. September.

```
// Variante 3

public void zeigeJahreszeitV3(int monat, int tag)
{
    // Monat prüfen ...
    // Tag prüfen ... mit getMaxDay(monat)

    String jahreszeit;

    if (monat == 12)
        jahreszeit = (tag < 21) ? "Herbst" : "Winter";
    else if (monat == 3)
        jahreszeit = (tag < 21) ? "Winter" : "Frühling";
    else if (monat == 6)
        jahreszeit = (tag < 21) ? "Frühling" : "Sommer";
    else if (monat == 9)
        jahreszeit = (tag < 21) ? "Sommer" : "Herbst";
    else if (monat == 1 || monat == 2) jahreszeit = "Winter";
    else if (monat == 4 || monat == 5) jahreszeit = "Frühling";
    else if (monat == 7 || monat == 8) jahreszeit = "Sommer";
    else jahreszeit = "Herbst";

    System.out.println(jahreszeit);
}
```

Übung 2.2.8 #5

Erstellt eine Methode

```
public int getSchraubentyp(int  
durchmesser, int laenge)
```

die den Typ der Schraube zurückliefert.

Dabei gilt:

- Durchmesser bis zu 3 mm, Länge bis zu 20 mm = Typ 1
- Durchmesser von 4 bis 6 mm, Länge von 21 bis 30 mm = Typ 2
- Durchmesser von 7 bis 20 mm, Länge von 31 bis 50 mm = Typ 3
- andere Werte = Typ 0 (unbekannter Schraubentyp)

// Lösung ist noch nicht korrekt!

```
public int getSchraubentyp1(int durchmesser, int laenge)  
{  
    if (durchmesser <= 3 && laenge <= 20) return 1;  
    if (durchmesser <= 6 && laenge <= 30) return 2;  
    if (durchmesser <= 20 && laenge <= 50) return 3;  
    return 0;  
}
```

Übung 2.2.8 #5

Erstellt eine Methode

```
public int getSchraubenTyp(int  
durchmesser, int laenge)
```

die den Typ der Schraube zurückliefert.

Dabei gilt:

- Durchmesser bis zu 3 mm, Länge bis zu 20 mm = Typ 1
- Durchmesser von 4 bis 6 mm, Länge von 21 bis 30 mm = Typ 2
- Durchmesser von 7 bis 20 mm, Länge von 31 bis 50 mm = Typ 3
- andere Werte = Typ 0 (unbekannter Schraubentyp)

```
public int getSchraubenTyp2(int durchmesser, int laenge)  
{  
    if (durchmesser >= 1 && durchmesser <= 3 &&  
        laenge >= 1 && laenge <= 20)  
        return 1;  
  
    if (durchmesser >= 4 && durchmesser <= 6 &&  
        laenge >= 21 && laenge <= 30)  
        return 2;  
  
    if (durchmesser >= 7 && durchmesser <= 20 &&  
        laenge >= 31 && laenge <= 50)  
        return 3;  
  
    return 0; // unbekannter Typ  
}
```

Übung 2.2.8 #6

Die sogenannte **Diskriminante** D einer quadratischen Gleichung $ax^2 + bx + c$ wird folgendermaßen berechnet:

$$D = b*b - 4*a*c.$$

Mit der Diskriminante kann man die Zahl der Lösungen dieser Gleichung bestimmen:

$D > 0$: Zwei Lösungen;

$D < 0$: keine Lösung;

$D = 0$: eine Lösung.

Erstellt eine Methode mit den Parametern a , b und c , die dann die Anzahl der Lösungen der Gleichung ausgibt.

```
public int getSchraubenTyp2(int durchmesser, int laenge)
{
    if (durchmesser >= 1 && durchmesser <= 3 &&
        laenge >= 1 && laenge <= 20)
        return 1;

    if (durchmesser >= 4 && durchmesser <= 6 &&
        laenge >= 21 && laenge <= 30)
        return 2;

    if (durchmesser >= 7 && durchmesser <= 20 &&
        laenge >= 31 && laenge <= 50)
        return 3;

    return 0; // unbekannter Typ
}
```

Übung 2.2.8 #7

Schreibt eine Methode

`public void sortiere(int a, int b, int c)`

die die drei Zahlen a, b und c in sortierter Reihenfolge ausgibt. Ihr dürft dabei nur if- oder if-else-Anweisungen benutzen.

```
private int getMin(int a, int b)
{
    return (a < b) ? a : b;
}

private int getMax(int a, int b)
{
    return (a > b) ? a : b;
}

public void sortiere(int a, int b, int c)
{
    // kleinste Zahl bestimmen
    int min = getMin(a, getMin(b, c));

    // größte Zahl bestimmen
    int max = getMax(a, getMax(b, c));

    // mittlere Zahl bestimmen
    int mitt = a + b + c - min - max;

    System.out.println
        (min + " - " + mitt + " - " + max);
}
```

Übung 2.2.8 #8

Schreibt eine Methode

```
public boolean sortiert  
    (int a, int b, int c, int d)
```

die überprüft, ob die vier Zahlen aufsteigend sortiert sind.

```
public boolean sortiert  
    (int a, int b, int c, int d)  
{  
    return a <= b && b <= c && c <= d;  
}
```

Übung 2.2.8 #9

Schreibt eine Methode

```
public boolean istDreieck  
    (int a, int b, int c, int d)
```

die überprüft, ob die drei Längen a, b und c ein Dreieck bilden können.

Dazu muss gelten:

$$a + b > c, \quad a + c > b, \quad b + c > a$$

```
public boolean istDreieck  
    (int a, int b, int c)  
{  
    return  
        a+b > c && a+c > b && b+c > a;  
}
```

Übung 2.2.8 #10

Schreibt eine Methode

`public boolean istReihe(int a, int b, int c)`

die überprüft, ob die drei Zahlen aufeinanderfolgen – also z. B. 3 4 5 oder 10 11 12.

Die Reihenfolge muss aufsteigend sein, und der Abstand zwischen den Zahlen soll jeweils genau 1 betragen.

```
// Variante 1
```

```
public boolean istReihe  
    (int a, int b, int c)  
  
{  
    if (a + 1 == b && b + 1 == c)  
        return true;  
    else  
        return false;  
}
```

```
// Variante 2
```

```
public boolean istReihe  
    (int a, int b, int c)  
  
{  
    return (a + 1 == b && b + 1 == c)  
}
```

Übung 2.2.8 #11

Schreibt eine Methode

```
public boolean istArithmetisch  
(int a, int b, int c)
```

die überprüft, ob die drei Zahlen eine arithmetische Reihe bilden. Der Abstand zwischen den drei Zahlen muss also gleich groß sein, beispielsweise 3, 5, 7 oder 2, 6, 10.

```
public boolean istArithmetisch  
    (int a, int b, int c)  
{  
    return b - a == c - b;  
}
```

Übung 2.2.8 #12

Schreibt eine Methode

```
public String dreiecksTyp  
    (int a, int b, int c)
```

die die drei Längen a, b, c eines Dreiecks überprüft und dann den Typ des Dreiecks als String zurückgibt:

- "kein Dreieck"
- "gleichseitiges Dreieck"
- "gleichschenkliges Dreieck"
- "ungleichseitiges Dreieck"

Auf "rechtwinkliges Dreieck" muss aber nicht getestet werden, das ist zu schwer.

```
public String dreiecksTyp  
    (int a, int b, int c)  
{  
    if (!istDreieck(a,b,c))  
        return "kein Dreieck";  
  
    if (a == b && b == c)  
        return "gleichseitiges Dreieck";  
  
    if (a == b || a == c || b == c)  
        return "gleichschenkliges Dreieck";  
  
    return "ungleichseitiges Dreieck";  
}
```