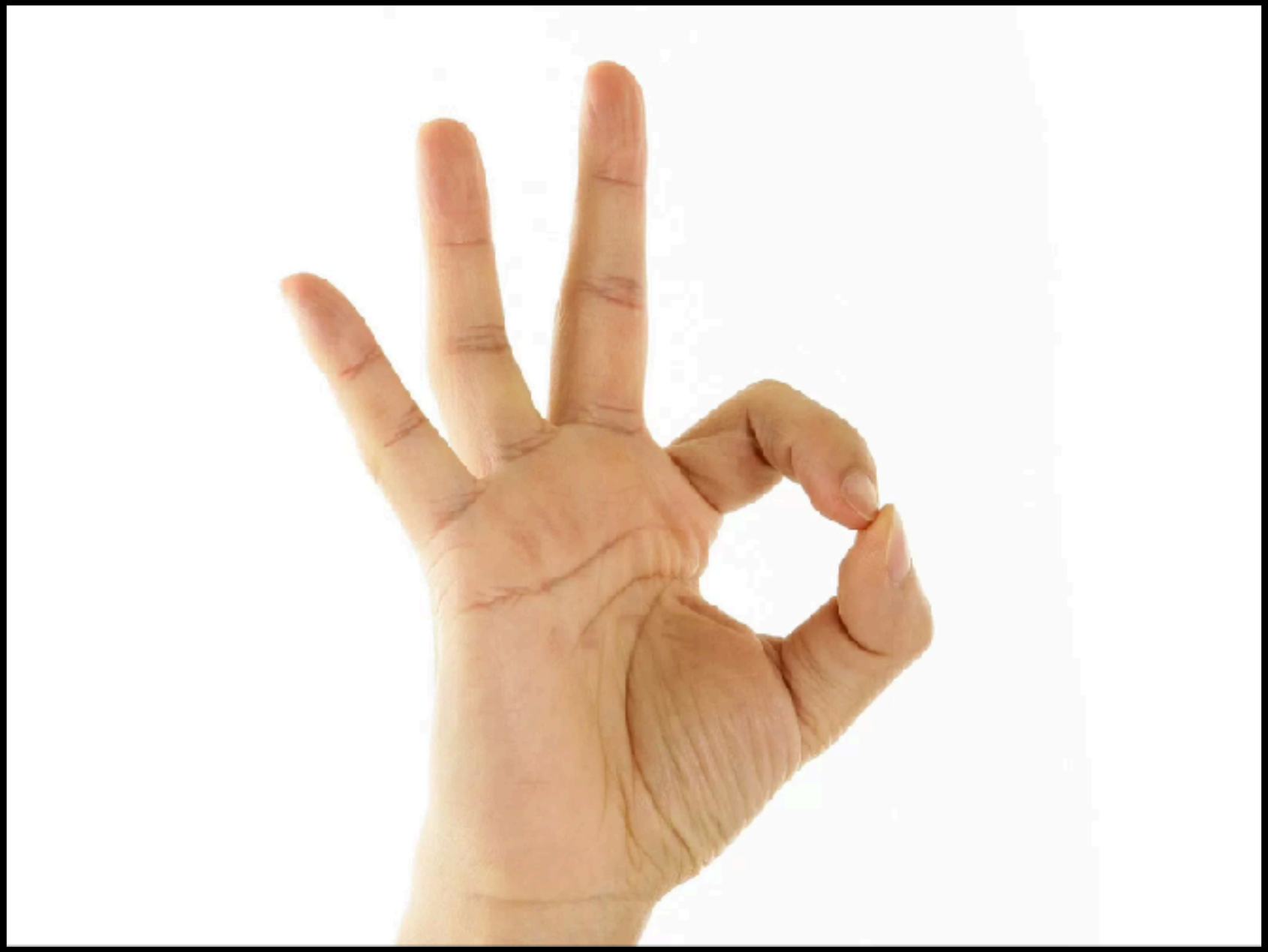
Informatik Kursstufe 2-stündig

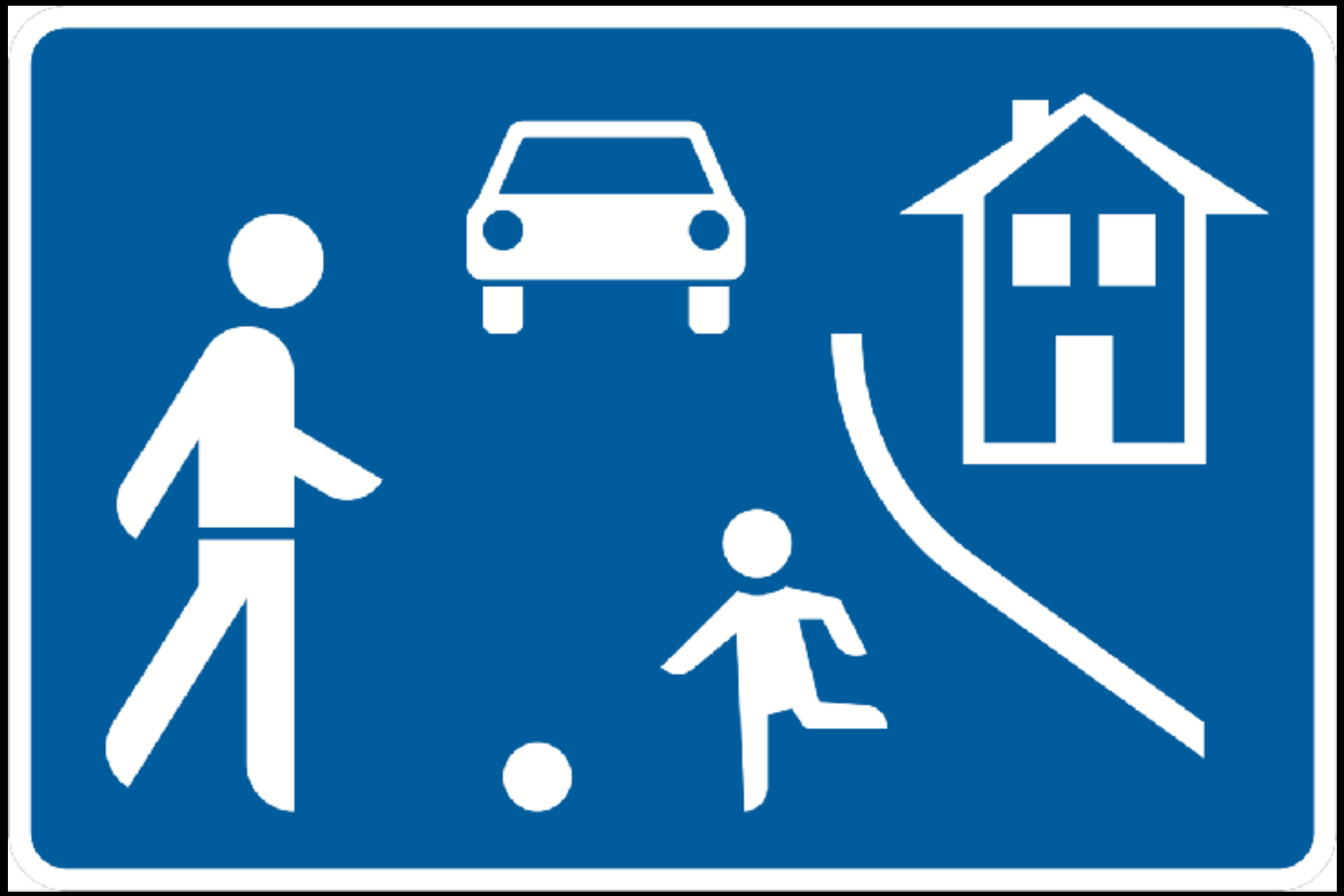
Schuljahr 18/19

10.9.18

17.9.18


8
www.MOTOR-TALK.de/blogs/andy/rx





MMXVIII

17.10.2018

09/17/2018

2018-09-17

# 1. Information

Informationen erhält man aus Wahrnehmungen, also von dem was wir sehen, hören, riechen, schmecken und fühlen. Wann erhält man eine Information?
**Beispiel:** Du fragst einen Freund, wie das Wetter morgen wird. Du hoffst auf eine Antwort wie: *„Es wird schön!“* oder *„Es wird regnerisch!“* Stattdessen erhältst du aber die Antwort: *„39“*.
Die Antwort ist für dich eigentlich keine Information.
Hättest du gefragt: *„Wie alt ist deine Mutti?“*, wäre *„39“* korrekt, denn hier enthält die Antwort die gewünschte Information.
Der Informationsgehalt ist also stets von der Frage abhängig.

# 2. Daten

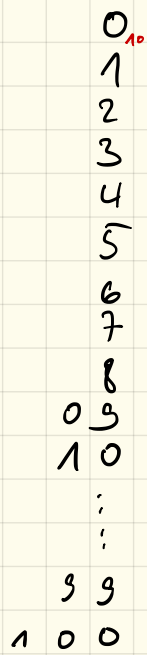
Nachrichten sind Angaben über einen Sachverhalt. Sie sind für uns stets erkennbar (in Wörtern, Bildern, Zahlen). Nachrichten enthalten Informationen und Informationen werden im Computer durch Daten dargestellt.
Daten sind demnach die computergerechte Form von Nachrichten. Daten sind also Träger von Informationen.
In der Elektronischen Datenverarbeitung (EDV) wird alles als Daten bezeichnet, was man für den Computer erkennbar speichern und darstellen kann.

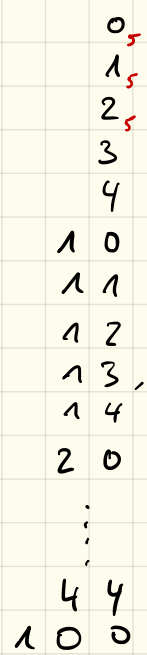
# 3. Codierung

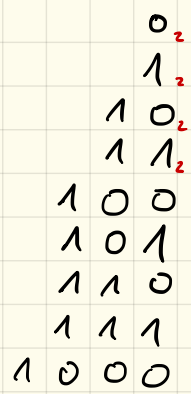
*Wie werden Daten dargestellt?*
Zwischen uns Menschen werden Informationen oft durch Zeichen übertragen (Schreiben, Lesen). Dazu wird das Alphabet benötigt, da alle Wörter aus Buchstaben zusammengesetzt werden.
Es gibt auch andere Möglichkeiten zur Informationsübertragung. Zum Beispiel die Farben Rot, Gelb und Grün bei der Verkehrsampel enthalten eine bestimmte Information.
Der Computer kann unsere Zeichenfolgen nicht lesen. Wir müssen unsere Sprache in eine Sprache übersetzen, die der Computer versteht. Dieses Übersetzen nennt man Codierung.
Der Code ist dabei der Schlüssel für die Lesbarkeit (Übersetzungsschlüssel). Und sowie die Quelle als auch der Empfänger müssen im Besitz dieses Schlüssels sein, damit die übermittelten Daten gelesen werden können.

# 4. Aufgabe

Überlege dir, welche Informationen es in deinem Alltag gibt und wie diese codiert werden.


0 10
1
2
3
4
5
6
7
8
09
10
99
100


0 5
1 5
2 5
3
4
10
11
12
13
14
20
44
100


0 2
1 2
10 2
11 2
100
101
110
111
1000

## Dezimalsystem

Das Dezimalsystem – oftmals auch Zehnersystem genannt – ist ein Stellenwertsystem zur Darstellung von Zahlen. Es verwendet die Basis 10. Das Dezimalsystem ist heute das weltweit verbreitetste Zahlensystem. Vermutlich hat das Dezimalsystem seinen Ursprung dem Umstand zu verdanken, dass der Mensch zehn Finger hat, welche man zum Zählen einsetzen kann. Im Zehnersystem kommen 10 Ziffern zum Einsatz: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Hat man von 0 bis 9 gezählt und möchte dies fortsetzen, dann beginnt man die folgenden Zahlen zusammen zu setzen. So folgt nach der 9 dann die 10, die 11, die 12 usw.
Haben wir beim Zählen beispielsweise an der Einerstelle schon die höchste Ziffer erreicht, so erhöhen wir die nächstgrößere Zehnerstelle und beginnen bei der Einerstelle wieder bei Null.
Wollen wir beispielsweise von der 99 auf die nächstgrößere Zahl, so sehen wir, dass an der Einerstelle bereits die höchste Ziffer steht, somit beginnen wir an dieser Stelle wieder bei 0 und erhöhen dafür die Zehnerstelle. Da an der Zehnerstelle jedoch auch bereits eine 9 steht, so beginnen wir auch hier wieder bei der 0 und erhöhen dafür die Hunderterstelle von 0 auf 1. Damit erhalten wir als Ergebnis die 100.

## Binärsystem

Ein Computer rechnet mit Strom. hierbei kann er jedoch nur zwischen *Strom an* und *Strom aus* entscheiden bzw. zwischen 1 und 0. Ein Computer kann also lediglich mit 2 Ziffern anstatt mit 10 Ziffern rechnen!
Die Zählweise funktioniert aber ansonsten genau gleich wie im Dezimalsystem!

| **Dezimalsystem** | **Binärsystem** |
|---|---|
| 0 | $0_2$ |
| 1 | $1_2$ |
| 2 | $10_2$ |
| 3 | $11_2$ |
| 4 | $100_2$ |
| 5 | $101_2$ |

← im Binärsystem ist das bereits die höchste Ziffer...
← ...deshalb bekommt die Zahl bereits hier eine zweite Stelle!

*Hinweis: Um Verwechslungen zu vermeiden schreibt man Zahlen im Binärsystem mit einer kleinen angehängten 2.*

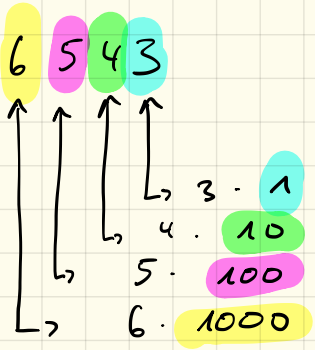
## 1. Aufgabe

Vervollständige die Tabelle:

| **Dezimalsystem** | **Binärsystem** |
|---|---|
| 0 | $0_2$ |
| 1 | $1_2$ |
| 2 | $10_2$ |
| 3 | $11_2$ |
| 4 | $100_2$ |
| 5 | $101_2$ |
| 6 | 110 |
| 7 | 111 |

| **Dezimalsystem** | **Binärsystem** |
|---|---|
| 8 | 1000 |
| 9 | 1001 |
| 10 | 1010 |
| 11 | 1011 |
| 12 | 1100 |
| 13 | 1101 |
| 14 | 1110 |
| 15 | 1111 |

# Dezimalsystem




# Binärsystem

$1_2 = 1$

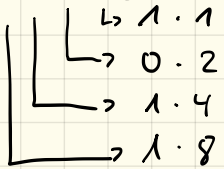
$10_2 = 2$

$100_2 = 4$

$1000_2 = 8$

$10000_2 = 16$

$1101_2 = 13$



64 32 16 8 4 2 1

$1011010_2 = 90_{10}$

| | 16 | 8 | 4 | 2 | 1 | |
|---|---|---|---|---|---|---|
| $17_{10} =$ | 1 | 0 | 0 | 0 | 1 | $_2$ |

$$24_{10} = 0 \cdot 32 + 24$$
$$24 = 1 \cdot 16 + 8$$
$$8 = 1 \cdot 8 + 0$$
$$0 = 0 \cdot 4 + 0$$
$$0 = 0 \cdot 2 + 0$$
$$0 = 0 \cdot 1 + 0$$

$$24_{10} = 11000_2$$

$$28 = 1 \cdot 16 + 12$$
$$12 = 1 \cdot 8 + 4$$
$$4 = 1 \cdot 4 + 0$$
$$0 = 0 \cdot 2 + 0$$
$$0 = 0 \cdot 1 + 0$$

$$28_{10} = 11100_2$$

## 2. Aufgabe

Überlegt euch ein Rechenverfahren um auch ohne die obige Tabelle eine Zahl aus dem Binärsystem in das Dezimalsystem umzurechnen. Beschreibt euer Verfahren und testet es mit folgenden Zahlen: $10_2$, $101_2$, $1101_2$, $10101_2$, $100001_2$ und $101111_2$

$10_2 = 2_{10}$ ✓

$101_2 = 6_{10}$ ✓

$1101_2 = 13_{10}$ ✓

$10101_2 = 21_{10}$ ✓

$100001_2 = 33_{10}$ ✓

$101111_2 = \cancel{42}\ 47_{10}$ ✓

## 3. Aufgabe

Überlegt euch ein Rechenverfahren für den „Rückweg“ vom Dezimal- in das Binärsystem. Beschreibt euer Verfahren und testet es mit folgenden Zahlen: 17, 24, 28, 157 und 332

$17_{10} = 10001_2$

$24_{10} = 11000_2$

$28_{10} = 11100_2$

$157_{10} = 1 \cdot 128 + 29$
$29 = 0 \cdot 64 + 29$
$29 = 0 \cdot 32 + 29$
$29 = 1 \cdot 16 + 13$
$13 = 1 \cdot 8 + 5$
$5 = 1 \cdot 4 + 1$
$1 = 0 \cdot 2 + 1$
$1 = 1 \cdot 1 + 0$
$157_{10} = 10011101_2$

1001 1101

$332_{10} = 1 \cdot 256 + 76$
$76 = 0 \cdot 128 + 76$
$76 = 1 \cdot 64 + 12$
$12 = 0 \cdot 32 + 12$
$12 = 0 \cdot 16 + 12$
$12 = 1 \cdot 8 + 4$
$4 = 1 \cdot 4 + 0$
$0 = 0 \cdot 2 + 0$
$0 = 0 \cdot 1 + 0$

$332_{10} = 101001100_2$

1 0100 1100

## 4. Aufgabe

Rechne in das jeweils andere Stellenwertsystem um.

a) $1101\ 1110_2$ $222_{10}$
b) $0011\ 1111_2$ $63_{10}$
c) $1111\ 1101_2$ 253
d) $0101\ 1010_2$ 90
e) $13_{10}$ 1101
f) $96_{10}$ 110 0000
g) $254_{10}$ 1111 1110
h) $127_{10}$ 111 1111

## 5. Aufgabe

Jede Stelle im Binärsystem wird durch ein Bit repräsentiert. Wie viele Bits benötigt man um folgende Zahlen darstellen zu können? Rechne die Zahlen ins Binärsystem um und versuche, einen Gesetzmäßigkeit für die Bitlänge zu finden.

a) $1 = 1$ → 1 Bit
b) $7 = 111$ → 3 Bits
c) $8 = 1000$ → 4 Bits
d) $31 = 11111$ → 5 Bits
e) $32 = 100000$ → 6 Bits
f) $120 = 111\ 1000$ → 7 Bits
g) $260 = 100000100$ → 9 Bits
h) $4703 = 1\ 0010\ 0101\ 1111$ → 13 Bits

Welche Zahlen können mit 8 Bit = 1 Byte, 2 Byte, 4 Byte und 8 Byte dargestellt werden?

8 Bit = 1 Byte ≙ 0 – 255

16 Bit = 2 Byte ≙ von 0 bis $2^{16} - 1 = 65535$

32 Bit = 4 Byte ≙ von 0 bis $2^{32} - 1 = 4\,294\,967\,295$

64 Bit = 8 Byte ≙ von 0 bis $2^{64} - 1 = 1{,}84 \cdot 10^{19}$

3 Bit ≙ 0 – 7 also von 0 bis $2^3 - 1$

5 Bit ≙ 0 bis $2^5 - 1 = 31$

24.9.18

| Hexadezimal | | Dezimal |
|---|---|---|
| 0 | ≙ | 0 |
| 1 | ≙ | 1 |
| 2 | ≙ | 2 |
| ⋮ | | |
| 9 | ≙ | 9 |
| A | ≙ | 10 |
| B | ≙ | 11 |
| C | ≙ | 12 |
| D | ≙ | 13 |
| E | ≙ | 14 |
| F | ≙ | 15 |

256 16 1

$3\ A\ C_{16} = 940$

$3 \cdot 256 + 10 \cdot 16 + 12 \cdot 1$

8 4 2 1

$1\ 0\ 0\ 1_2 = 9$

$1 \cdot 8 \quad + 1 \cdot 1$

# Umrechnung Dezimal -> Binär

| | | | | |
|---|---|---|---|---|
| 372 = | 1 · | 256 | + | 116 |
| 116 = | 0 · | 128 | + | 116 |
| 116 = | 1 · | 64 | + | 52 |
| 52 = | 1 · | 32 | + | 20 |
| 20 = | 1 · | 16 | + | 4 |
| 4 = | 0 · | 8 | + | 4 |
| 4 = | 1 · | 4 | + | 0 |
| | 0 | 2 | | |
| | 0 | 1 | | |

# Umrechnung Dezimal -> Hexadezimal

$372 = 1 \cdot 256 + 116 = 174_{16}$

$116 = 7 \cdot 16 + 4$

$4 = 4 \cdot 1 + 0$

---

$539 = 2 \cdot 256 + 27 = 21B_{16}$

$27 = 1 \cdot 16 + 11$

$11 = 11 \cdot 1 + 0$

## Hexadezimalsystem

Im Hexadezimalsystem gibt es 16 Ziffern. Da wir in unserem „normalen“ Dezimalsystem jedoch nur 10 Ziffern kennen müssen wir weitere 6 „erfinden“. Hierfür nehmen wir Buchstaben:

- A $\widehat{=}$ 10
- B $\widehat{=}$ 11
- C $\widehat{=}$ 12
- D $\widehat{=}$ 13
- E $\widehat{=}$ 14
- F $\widehat{=}$ 15

Die Stellenwerte sind dabei:

| | | | |
|---|---|---|---|
| $16^3 = 4096$ | $16^2 = 256$ | $16^1 = 16$ | $16^0 = 1$ |

Die Hexadezimale Zahl $3AC_{16}$ entspricht also der dezimalen Zahl $3 \cdot 256 + \underbrace{10}_{\text{A}} \cdot 16 + \underbrace{12}_{\text{C}} = 940$

## 1. Aufgabe

Rechne die folgenden hexadezimalen Zahlen ins Dezimalsystem um:

a) $C1_{16}$ 193

b) $F7_{16}$ 247

c) $AAB_{16}$ 2.731

d) $1FC_{16}$ 508

e) $123_{16}$ 291

f) $5AB_{16}$ 1.451

g) $73A1_{16}$ 29.601

h) $13AF_{16}$ 5.039

## 2. Aufgabe

Rechne die folgenden dezimalen Zahlen ins Hexadezimalsystem um:

a) 14 $E_{16}$

b) 27 $1B_{16}$

c) 63 $3F_{16}$

d) 139 $8B_{16}$

e) 532 $214_{16}$

f) 1024 $400_{16}$

g) 52012 $CB2C_{16}$

h) 65535 $FFFF_{16}$

# Umrechnung Binär <-> Hexadezimal

| Dez | Bin | Hex |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 1 | 1 |
| 2 | 10 | 2 |
| 3 | 11 | 3 |
| 4 | 100 | 4 |
| 5 | 101 | 5 |
| 6 | 110 | 6 |
| 7 | 111 | 7 |
| 8 | 1000 | 8 |
| 9 | 1001 | 9 |
| 10 | 1010 | A |
| 11 | 1011 | B |
| 12 | 1100 | C |
| 13 | 1101 | D |
| 14 | 1110 | E |
| 15 | 1111 | F |
| 16 | 10000 | 10 |

101 0111 1100

5 7 C

## Umrechnung Binärsystem ↔ Hexadezimalsystem

Das Hexadezimalsystem wird besonders dazu verwendet, um binäre Zahlen und Ausdrücke kürzer darzustellen. Hierbei werden imm 4 binäre Stellen zu einer hexadezimalen Ziffer zusammengefasst:

- $0000_2 = 0_{16}$
- $0001_2 = 1_{16}$
- $0010_2 = 2_{16}$
- $0011_2 = 3_{16}$
- $0100_2 = 4_{16}$
- $0101_2 = 5_{16}$
- $0110_2 = 6_{16}$
- $0111_2 = 7_{16}$
- $1000_2 = 8_{16}$
- $1001_2 = 9_{16}$
- $1010_2 = A_{16}$
- $1011_2 = B_{16}$
- $1100_2 = C_{16}$
- $1101_2 = D_{16}$
- $1110_2 = E_{16}$
- $1111_2 = F_{16}$

## 3. Aufgabe

Wandle damit die folgenden Binärzahlen in Hexadezimalzahlen um:

a) $\overbrace{01101001}^{6\ \ 9}_2 = 69_{16}$

b) $\overbrace{01011101}^{5\ \ 13}_2 = 5D_{16}$

c) $\overbrace{10010100010}^{2\ \ 5\ \ 2}_2 = 252_{16}$

d) $\overbrace{11010011101}^{3\ \ 4\ \ 13}_2 = 34D_{16}$

e) $\overbrace{11100110010}^{7\ \ 3\ \ 2}_2 = 732_{16}$

f) $\overbrace{101110101010}^{11\ \ 10\ \ 10}_2 = BAA_{16}$

g) $\overbrace{1010001001001010}^{10\ \ 2\ \ 4\ \ 10}_2 = A24A_{16}$

h) $\overbrace{1110101001011111}^{14\ \ 10\ \ 5\ \ 15}_2 = EA5F_{16}$

## 4. Aufgabe

Wandle die folgenden Hexadezimalzahlen ins Binärsystem um:

a) $C1_{16} = 1100\ 0001_2$

b) $F7_{16} = 1111\ 0111_2$

c) $AAB_{16} = 1010\ 1010\ 1011_2$

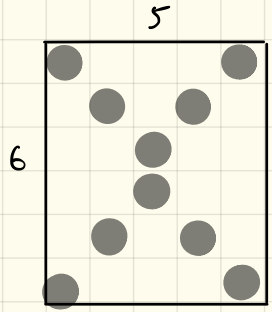
d) $1FC_{16} = 1\ 1111\ 1100_2$

e) $123_{16} = 1\ 0010\ 0011_2$

f) $5AB_{16} = 101\ 1010\ 1011_2$

g) $73A1_{16} = 111\ 0011\ 1010\ 0001_2$

h) $13AF_{16}\ 1\ 0011\ 1010\ 1111_2$




0 0 0 0 0 1 0 1
0 0 0 0 0 1 1 0
1 0 0 0 1 0 1 0
1 0 0 0 1 0 0 0
0 1 0 0 0 1 0 1
0 1 0 0 0 1 0 0

# Codierung: Bilder

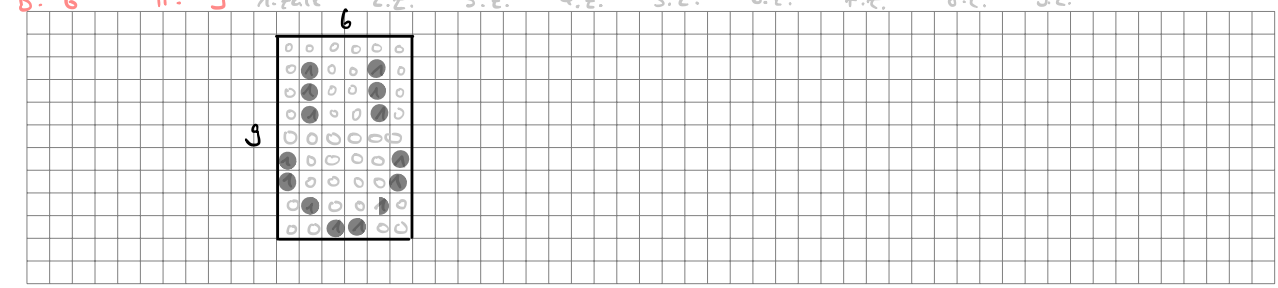
## 1. Schwarz-Weiß-Pixelbilder

Einfache schwarz-weiß-Pixelbilder lassen sich sehr einfach darstellen. Hierbei muss lediglich angegeben werden, welche Pixel eingeschaltet (weiß) bzw. ausgeschaltet (schwarz) sind. Ein Bild kann so als lange Bit-Kette geschrieben werden.
Damit der Computer jedoch weiß, wann eine neue Zeile beginnt, muss zunächst die Breite des Bildes angegeben werden. Damit ein Bild richtig abgespeichert werden kann und auch wieder richtig gelesen werden kann müssen wir uns ein eigenes Dateiformat definieren:

- Im ersten Byte der „Datei“ steht die Breite des Bildes
- Im zweiten Byte steht die Höhe des Bildes
- Anschließend folgen die Bits, die angeben, ob ein Pixel an oder aus ist.
- Ist die Länge der Bitkette kein Vielfaches von 8, so werden die restlichen Bits mit Nullen aufgefüllt.

## 2. Aufgabe

Entschlüssle mit obigen Angaben folgendes „Bild“:
00000110 00001001 00000001 00100100 10010010 00000010 00011000 01010010 00110000

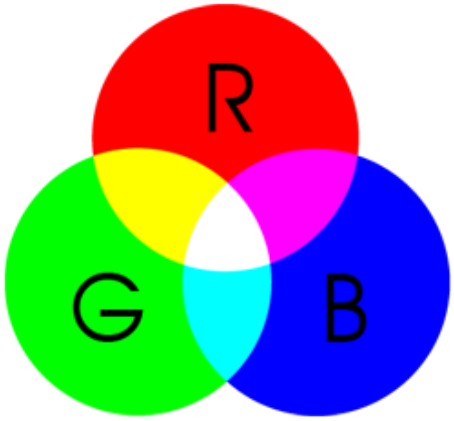



## 3. Aufgabe

Erstelle selbst ein Bild, codiere es in eine Bitfolge und gib es deinem Nachbarn zur Entschlüsselung.

## 4. Farbige Bilder

Auf Dauer werden Schwarz-Weiß-Bilder recht langweilig und es muss etwas Farbe ins Spiel kommen. Hier nutzen wir das RGB-System: Jeder Pixel besteht dabei aus 3 Farben rot, grün und blau und können folgendermaßen gemischt werden:



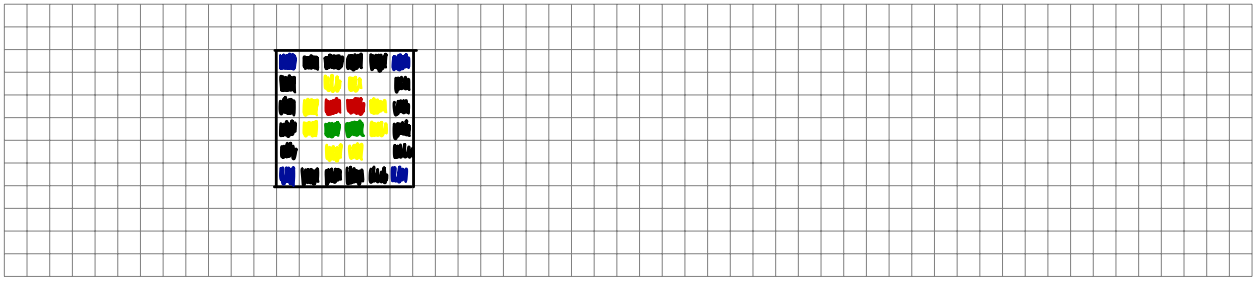

- Rot
- Grün
- Blau
- Rot + Grün = Gelb
- Rot + Blau = Magenta
- Blau + Grün = Cyan
- Rot + Grün + Blau = Weiß

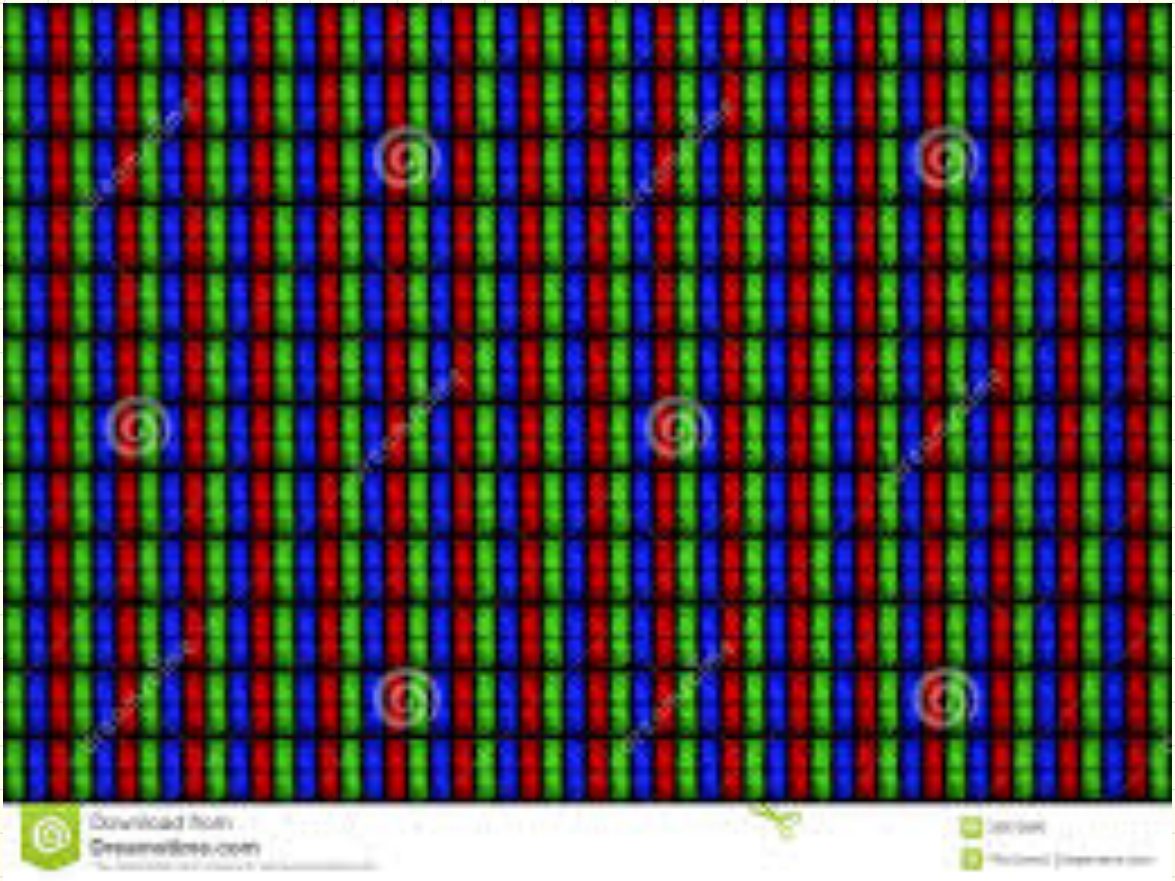
Damit definieren wir ein neues Dateiformat:

- Im ersten Byte der „Datei“ steht die Breite des Bildes
- Im zweiten Byte steht die Höhe des Bildes
- Anschließend folgen die Pixel, dabei gehören immer 3 Bits zu einem Pixel:
  - das erste Bit gibt an, ob der rote Anteil dabei angeschaltet ist
  - das zweite Bit gibt den grünen Anteil an
  - das dritte Bit gibt den blauen Anteil an
  - z. B., eine Bitfolge 000 bedeutet schwarz, 001 bedeutet blau, 110 bedeutet gelb, 111 bedeutet weiß.
- Ist die Länge der Bitkette kein Vielfaches von 8, so werden die restlichen Bits mit Nullen aufgefüllt.

## 5. Aufgabe

Entschlüssle mit obigen Angaben folgendes „Bild“:
00000110 00000110 00100000 00000000 01000111 11011011 10000001 10100100 11000000 01100100 10110000 00011111 01101110 00001000 00000000 00010000




Download from
Dreamstime.com

# Codierung: Text und Zeichen

## 1. Einführung: ASCII

Der **American Standard Code for Information Interchange** (ASCII) ist eine 7-Bit-Zeichencodierung und dient als Grundlage für spätere, auf mehr Bits basierende Kodierungen für Zeichensätze. Die druckbaren Zeichen umfassen das lateinische Alphabet in Groß- und Kleinschreibung, die zehn arabischen Ziffern sowie einige Interpunktions- und andere Sonderzeichen. Der Zeichenvorrat entspricht weitgehend dem einer Tastatur oder Schreibmaschine für die *englische Sprache*.
Um weitere Zeichen darstellen zu können, wurden von verschiedenen Gremien Erweiterungen für das Standardpaket entwickelt, welche dann 8 Bit verwendet und damit 256 Zeichen darstellen können. Die ersten 128 Zeichen sind dabei identisch zum Standard-Zeichensatz:

| ASCII control characters | | | ASCII printable characters | | | | | | Extended ASCII characters | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00 | NULL | (Null character) | 32 | space | 64 | @ | 96 | ` | 128 | Ç | 160 | á | 192 | └ | 224 | Ó |
| 01 | SOH | (Start of Header) | 33 | ! | 65 | A | 97 | a | 129 | ü | 161 | í | 193 | ┴ | 225 | ß |
| 02 | STX | (Start of Text) | 34 | " | 66 | B | 98 | b | 130 | é | 162 | ó | 194 | ┬ | 226 | Ô |
| 03 | ETX | (End of Text) | 35 | # | 67 | C | 99 | c | 131 | â | 163 | ú | 195 | ├ | 227 | Ò |
| 04 | EOT | (End of Trans.) | 36 | $ | 68 | D | 100 | d | 132 | ä | 164 | ñ | 196 | ─ | 228 | õ |
| 05 | ENQ | (Enquiry) | 37 | % | 69 | E | 101 | e | 133 | à | 165 | Ñ | 197 | ┼ | 229 | Õ |
| 06 | ACK | (Acknowledgement) | 38 | & | 70 | F | 102 | f | 134 | å | 166 | ª | 198 | ã | 230 | µ |
| 07 | BEL | (Bell) | 39 | ' | 71 | G | 103 | g | 135 | ç | 167 | º | 199 | Ã | 231 | þ |
| 08 | BS | (Backspace) | 40 | ( | 72 | H | 104 | h | 136 | ê | 168 | ¿ | 200 | ╚ | 232 | Þ |
| 09 | HT | (Horizontal Tab) | 41 | ) | 73 | I | 105 | i | 137 | ë | 169 | ® | 201 | ╔ | 233 | Ú |
| 10 | LF | (Line feed) | 42 | * | 74 | J | 106 | j | 138 | è | 170 | ¬ | 202 | ╩ | 234 | Û |
| 11 | VT | (Vertical Tab) | 43 | + | 75 | K | 107 | k | 139 | ï | 171 | ½ | 203 | ╦ | 235 | Ù |
| 12 | FF | (Form feed) | 44 | , | 76 | L | 108 | l | 140 | î | 172 | ¼ | 204 | ╠ | 236 | ý |
| 13 | CR | (Carriage return) | 45 | - | 77 | M | 109 | m | 141 | ì | 173 | ¡ | 205 | ═ | 237 | Ý |
| 14 | SO | (Shift Out) | 46 | . | 78 | N | 110 | n | 142 | Ä | 174 | « | 206 | ╬ | 238 | ¯ |
| 15 | SI | (Shift In) | 47 | / | 79 | O | 111 | o | 143 | Å | 175 | » | 207 | ¤ | 239 | ´ |
| 16 | DLE | (Data link escape) | 48 | 0 | 80 | P | 112 | p | 144 | É | 176 | ░ | 208 | ð | 240 | ≡ |
| 17 | DC1 | (Device control 1) | 49 | 1 | 81 | Q | 113 | q | 145 | æ | 177 | ▒ | 209 | Ð | 241 | ± |
| 18 | DC2 | (Device control 2) | 50 | 2 | 82 | R | 114 | r | 146 | Æ | 178 | ▓ | 210 | Ê | 242 | ‗ |
| 19 | DC3 | (Device control 3) | 51 | 3 | 83 | S | 115 | s | 147 | ô | 179 | │ | 211 | Ë | 243 | ¾ |
| 20 | DC4 | (Device control 4) | 52 | 4 | 84 | T | 116 | t | 148 | ö | 180 | ┤ | 212 | È | 244 | ¶ |
| 21 | NAK | (Negative acknowl.) | 53 | 5 | 85 | U | 117 | u | 149 | ò | 181 | Á | 213 | ı | 245 | § |
| 22 | SYN | (Synchronous idle) | 54 | 6 | 86 | V | 118 | v | 150 | û | 182 | Â | 214 | Í | 246 | ÷ |
| 23 | ETB | (End of trans. block) | 55 | 7 | 87 | W | 119 | w | 151 | ù | 183 | À | 215 | Î | 247 | ¸ |
| 24 | CAN | (Cancel) | 56 | 8 | 88 | X | 120 | x | 152 | ÿ | 184 | © | 216 | Ï | 248 | ° |
| 25 | EM | (End of medium) | 57 | 9 | 89 | Y | 121 | y | 153 | Ö | 185 | ╣ | 217 | ┘ | 249 | ¨ |
| 26 | SUB | (Substitute) | 58 | : | 90 | Z | 122 | z | 154 | Ü | 186 | ║ | 218 | ┌ | 250 | · |
| 27 | ESC | (Escape) | 59 | ; | 91 | [ | 123 | { | 155 | ø | 187 | ╗ | 219 | █ | 251 | ¹ |
| 28 | FS | (File separator) | 60 | < | 92 | \ | 124 | \| | 156 | £ | 188 | ╝ | 220 | ▄ | 252 | ³ |
| 29 | GS | (Group separator) | 61 | = | 93 | ] | 125 | } | 157 | Ø | 189 | ¢ | 221 | ¦ | 253 | ² |
| 30 | RS | (Record separator) | 62 | > | 94 | ^ | 126 | ~ | 158 | × | 190 | ¥ | 222 | Ì | 254 | ■ |
| 31 | US | (Unit separator) | 63 | ? | 95 | _ | | | 159 | ƒ | 191 | ┐ | 223 | ▀ | 255 | nbsp |
| 127 | DEL | (Delete) | | | | | | | | | | | | | | |

## 2. Aufgabe

Entschlüssele folgende Nachricht:
73 110 102 111 114 109 97 116 105 107 32 105 115 116 32 116 111 108 108 33

## 3. Aufgabe

Schreibe deinem Sitznachbarn eine (kurze) Nachricht, indem du lediglich die ASCII-Codes verwendest.

## 4. Zusatzaufgabe

Warum gab es immer wieder Fehler in der Darstellung von Webseiten? Wie wurde das Problem umgangen?
*Hinweis: Informiere dich hierzu ggf. über die Norm **ISO 8859** und vergleiche diese mit dem oben abgedruckten **extended ASCII**.*

5.11.18

# Datentypen in JAVA

# Einführung

## 1. Einführung

Ziel der Aufgabe ist es, Eclipse etwas kennen zu lernen und zu wissen, wie man ein neues Java-Projekt in Eclipse anlegt.

### 1.1 Information: Neues Projekt anlegen

- File → New → Java Project (oder: File → New → Project... → Java → Java Project)
- Projektname eingeben und mit *Finish* bestätigen
- Paket anlegen mit File → New → Package
- Paketname eingeben und mit *Finish* bestätigen
- Neue Klasse in Paket anlegen mit File → New → Class
- Klassenname eingeben
- Bei *Which method stubs would you like to create?* die `main()`-Methode auswählen und mit *Finish* bestätigen.

### 1.2 Information: Kompilieren und Ausführen

- Run → Run As → Java Application
- oder direkt in der Menüleiste den grünen Button benutzen

### 1.3 Konventionen für den Unterricht

- **Workspace auf Netzlaufwerk „H:“!**
- Wir legen für die ersten Wochen ein Projekt `Einfuehrung` an.
- In diesem Projekt werden einzelne Pakete für die Arbeitsblätter angelegt. (`21Einfuehrung`)
- In diesem Paket sollen die Aufgaben als einzelne Klassen angelegt werden (z. B. `Aufgabe1a`)
- (Jeder Quellcode soll auch kommentiert werden, das hilft beim späteren Durchlesen dem Verständnis!)

## 2. Ein erstes Programm

Die `main`-Methode wird beim Programmstart aufgerufen, d. h. die Befehle in dieser Methode werden nacheinander abgearbeitet.
Mit dem Befehl

```
System.out.println("Hallo Welt");
```

Listing 1: Ausgeben von Daten auf der Konsole

wird die Zeichenkette `Hallo Welt` auf der Konsole ausgegeben.
Lege also wie oben beschrieben ein neues Projekt `Einfuehrung` mit einem Paket `21Einfuehrung` an und erstelle darin eine Klasse `Aufgabe2`. Achte bei der Erzeugung der Klasse darauf, dass diese eine `main`-Methode enthält!
Schreibe also zunächst diesen Befehl in die `main`-Methode (d. h. innerhalb der geschweiften Klammern) und führe das Programm aus.
Neben dem Befehl `System.out.println();` gibt es auch noch den Befehl `System.out.print();` Beschreibe kurz den Unterschied der beiden Befehle:

## 3. Variablen

Mithilfe von sogenannten Variablen können wir Werte zwischenspeichern. Eine Variable `a` können wir mithilfe des Befehls

```
int ersteVariable;
```

erzeugen. Um dieser Variablen beispielsweise den Wert `5` zuzuweisen schreiben wir

```
ersteVariable = 5;
```

Dieser Variablen haben wir dabei den Namen `ersteVariable` gegeben. Nachdem wir diese Variable erzeugt und einen Wert festgelegt haben können wir diese danach im Quelltext verwenden und schreiben dabei statt dem Wert den Naben dieser Variablen. Wollen wir beispielweise den *Wert dieser Variablen* auf der Konsole ausgeben schreiben wir den Befehl

```
System.out.println(ersteVariable);
```

Das Wort `int` gibt dabei den sogenannten Datentyp an, d. h. welche Art von Werten können in der Variable. `int` bedeutet dabei, dass wir in der Variablen `ersteVariable` *ganze Zahlen* speichern können.
Recherchiere im Internet, welche anderen Datentypen es noch gibt:

int: ganze Zahlen von -2Mrd. bis +2Mrd.
short: ganze Zahlen von -32T bis +32T
long: ganze Zahlen von $-2^{63}$ bis $+2^{63}$
float: Kommazahlen $1{,}4 \cdot 10^{-45}$ bis $3{,}4 \cdot 10^{38}$
double: mehr Genauigkeit $4{,}9 \cdot 10^{-324}$ bis $1{,}7 \cdot 10^{308}$
byte: ganze Zahlen von -128 bis +127
char: einzelne Zeichen
String: Zeichenketten / Texte
boolean: wahr / falsch
true / false

# Eingabe

## Eingabe von Zahlen über die Konsole

Um Daten über die Konsole eingeben zu können musst du die Funktion dazu in deinem Programm "nachrüsten". Binde deshalb die entsprechenden Funktionen mit dem Befehl

```
import java.util.Scanner;
```

Listing 1: Einbinden von java.util.Scanner

in dein Programm ein. Dieser Befehl gehört direkt nach die Zeile `package ab2;`
Um nun ganze Zahlen über die Konsole in eine Variable einzulesen nutzen wir den Befehl

```
Scanner sc = new Scanner(System.in);
int a = sc.nextInt();
```

Listing 2: Einlesen von Daten mit java.util.Scanner

*Hinweis: um mehrere Werte nacheinander einlesen zu können genügt es, die zweite Zeile davon zu wiederholen, das Scanner-Objekt in Zeile 1 muss nur ein einziges Mal angelegt werden!*

## 1. Rechner

Lege innerhalb des Projektes ein neues Paket `22Eingabe` an mit einer Klasse `Aufgabe1`. Achte bei der Erzeugung der Klasse darauf, dass diese wieder eine `main`-Methode enthält!
Lasse nun mithilfe des `Scanner`s zwei Werte einlesen und gib davon die Summe, die Differenz, das Produkt und den Quotienten aus.
Mögliche Ausgabe:

```
Erste Zahl eingeben: 12
Zweite Zahl eingeben: 3
Summe: 15
Differenz: 9
Produkt: 36
Quotient: 4
```

Listing 3: Beispielausgabe

## 2. Fehler – Teil 1

Was passiert, wenn als zweite Zahl eine `0` eingegeben wird und warum?

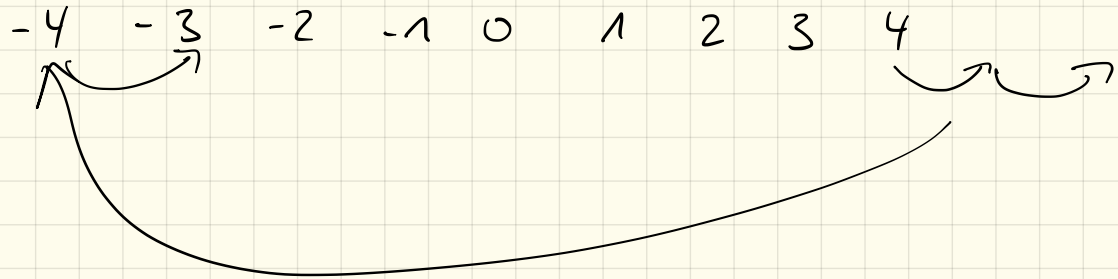
Wie könnten wir verhindern, dass das Programm mit einem Fehler abstürzt?

## 3. Fehler – Teil 2

Bei welchen Rechnungen liefert das Programm „falsche“ Ergebnisse und warum?

## 4. Zusatzaufgabe

Was passiert, wenn z. B. 50000*50000 gerechnet wird und warum?


-4 -3 -2 -1 0 1 2 3 4

# Bedingungen/Verzweigungen

## Einführung

Bedingungen bzw. Verzweigungen dienen dazu, dass bestimmte Befehlsblöcke nur dann ausgeführt werden, wenn eine bestimmte Voraussetzung gegeben ist.
Diese Bedingungen werden in Java mit dem `if`-Befehl verwirklicht:

```
if (BEDINGUNG) {
   [...]
}
```

Listing 1: `if`-Bedingung

Die Befehle innerhalb des mit geschweiften Klammern eingeschlossenen Blocks werden nur ausgeführt, wenn die Bedingung wahr ist. Hierbei können verschiedene Vergleiche benutzt werden:

| | |
|---|---|
| `<` kleiner als | `>=` größer oder gleich |
| `>` größer als | `==` genau gleich (nicht nur bei Zahlen) |
| `<=` kleiner oder gleich | `!=` ungleich (nicht nur bei Zahlen) |

## 1. Aufgabe: Absturzzicherer Rechner

Kopiere den Rechner von letztem Arbeitsblatt und ergänze diesen so, dass er bei einer Eingabe von „0“ als zweiter Zahl nicht abstürzt sondern eine eigene Fehlermeldung anzeigt.

## if-else

Erweitern lässt sich der `if`-Block noch um einen `else`-Block. Dessen Befehle werden nur ausgeführt, wenn die Bedingung *nicht* zutrifft.

```
if(x>5) {
  System.out.println("x ist größer als 5");
}
else {
  System.out.println("x ist kleiner oder gleich 5");
}
```

Listing 2: Beispiel zu `if-else`

Ebenso lassen sich mehrere Blöcke auch kombinieren:

```
if(alter<16) {
  System.out.println("du darfst kein Roller und kein Auto fahren");
}
else if(alter<18) {
  System.out.println("du darfst Roller, aber kein Auto fahren");
}
else {
  System.out.println("du darfst Roller und Auto fahren");
}
```

Listing 3: Beispiel zu `if-else`

# Klausur 26. 11.

- Codierung
  - Binärsystem
  - Hexadezimalsystem
  - Text / ASCII
  - Bilder
- Grundlagen Programmierung
  - Ausgabe
  - Eingabe
  - Variablen + Datentypen
  - Bedingungen
  - Schleifen

19.11.2018

```
3  public class Start {
4
5      public static void main(String[] args) {
6
7          System.out.println(1);
8          System.out.println(4);
9          System.out.println(9);
10         System.out.println(16);
11         System.out.println(25);
12         System.out.println(36);
13         System.out.println(49);
14         System.out.println(64);
15
16     }
17
18 }
```

2 * 2

3 * 3

Initialisierung

Bedingung

```
for (int i=0; i<10; i=i+1) {
    System.out.println(i*i);
}
```

```
int i = 0;
while ( i < 10 ) {

    i++;
}
```

```
if (BEDINGUNG) {

}
```

# Schleifen

## Einführung

Schleifen können dann genutzt werden, wenn Codeabschnitte mehrfach durchgeführt werden sollen. Man unterscheidet zwischen der `while`-Schleife und der `for`-Schleife.
`while`-Schleifen werden benutzt um Codeabschnitte so oft auszuführen, solange eine Bedingung zutrifft (vgl. `if`-Bedingung):

```
while(BEDINGUNG) {
  // mache irgendwas
}
```

Listing 1: `while`-Schleife

`for`-Schleifen werden auch als *Zählschleifen* bezeichnet. Hierbei wird – für gewöhnlich – eine Variable angelegt und diese hochgezählt:

```
for(int i=0 ; i<10 ; i=i+1) {
  // mache irgendwas
}
```

Listing 2: `for`-Schleife

## 1. Quadratzahlen

Lasse mithilfe einer Schleife die Quadratzahlen von 1 bis 20 auf der Konsole ausgeben.
Das Ergebnis könnte beispielsweise so aussehen: `Die Quadratzahl von 7 ist 49`

## 2. Programmierung „Getränkeautomat“

Ziel ist es, den gezeigten „Getränkeautomat“ nachzuprogrammieren. Dieser soll folgendes können:

1. Ausgabe der verfügbaren Getränke und Preise
2. Eingabe des gewünschten Getränks
3. Eingabe der gewünschten Menge
4. Bezahlvorgang: Anzeige des fehlenden Restbetrages und „Einwurf“ der Münzen
5. Wenn komplett bezahlt dann „Ausgabe der Getränke“

## 3. Zusatzaufgabe 1: Rückzahlung

Erweitere den Automat so, dass bei Überbezahlung der Restbetrag in möglichst wenigen Münzen wieder zurückbezahlt wird. Bildschirmausgabe z. B.

```
Rückgeld: 0.45
gebe zurück: 0.20
Rückgeld: 0.25
gebe zurück: 0.20
Rückgeld: 0.05
gebe zurück: 0.05
```

Listing 3: Rückgeld

# 4. Zusatzaufgabe 2: mehrere Getränke

Erweitere den Automat so, dass auch mehrere unterschiedliche Getränke gleichzeitig bestellt werden können, und diese bei der Ausgabe auch in eingegebener Reihenfolge ausgegeben werden.

## 4.1 Beispielausgabe des Automates

```
Getränke Automat v0.3

Wählen sie ihr Getränk aus:
1) Wasser (0,50 Euro)
2) Limonade (1,00 Euro)
3) Bier (2,00 Euro)

Geben sie 1, 2 oder 3 ein: 3

Geben sie die gewünschte Menge ein: 2
--- Bezahlvorgang ---

Es fehlen noch 4.00 Euro.
Bitte werfen sie ein Geldstück ein: 2

Es fehlen noch 2.00 Euro.
Bitte werfen sie ein Geldstück ein: 1

Es fehlen noch 1.00 Euro.
Bitte werfen sie ein Geldstück ein: 0.5

Es fehlen noch 0.50 Euro.
Bitte werfen sie ein Geldstück ein: 0.5

--- Getränkeausgabe ---

Flasche 1 von 2 wurde ausgegeben.
Flasche 2 von 2 wurde ausgegeben.

Vielen Dank, bitte entnehmen sie ihre Getränke.
```

Ausgabe
Scanner(art)
Scanner(menge)
Gesamtpreis: 4 €
While-Schleife
for-Schleife

```
double ges = 0;
if(art==1){
    ges= 0.5*menge;
}
else if(art==2)
    ges = 1*menge;
}
else if (art==3){
    ges = 2*menge
}
System.out.print(ges);
```

Listing 4: Ausgabe Getränkeautomat

# Klausur 26.11.2018

Name: _____ VP: _____/32P NP: _____ mündlich: _____

## 1. Umrechnung (12VP)

Ergänze folgende Tabelle:

| | a) | b) | c) | d) | e) | f) |
|---|---|---|---|---|---|---|
| Dezimal | 83 | | | 207 | | |
| Binär | | 10010111 | | | 1110011 | |
| Hexadezimal | | | 6C | | | B4 |

## 2. Textcodierung (2VP)

**ASCII control characters**

| | | |
|---|---|---|
| 00 | NULL | (Null character) |
| 01 | SOH | (Start of Header) |
| 02 | STX | (Start of Text) |
| 03 | ETX | (End of Text) |
| 04 | EOT | (End of Trans.) |
| 05 | ENQ | (Enquiry) |
| 06 | ACK | (Acknowledgement) |
| 07 | BEL | (Bell) |
| 08 | BS | (Backspace) |
| 09 | HT | (Horizontal Tab) |
| 10 | LF | (Line feed) |
| 11 | VT | (Vertical Tab) |
| 12 | FF | (Form feed) |
| 13 | CR | (Carriage return) |
| 14 | SO | (Shift Out) |
| 15 | SI | (Shift In) |
| 16 | DLE | (Data link escape) |
| 17 | DC1 | (Device control 1) |
| 18 | DC2 | (Device control 2) |
| 19 | DC3 | (Device control 3) |
| 20 | DC4 | (Device control 4) |
| 21 | NAK | (Negative acknowl.) |
| 22 | SYN | (Synchronous idle) |
| 23 | ETB | (End of trans. block) |
| 24 | CAN | (Cancel) |
| 25 | EM | (End of medium) |
| 26 | SUB | (Substitute) |
| 27 | ESC | (Escape) |
| 28 | FS | (File separator) |
| 29 | GS | (Group separator) |
| 30 | RS | (Record separator) |
| 31 | US | (Unit separator) |
| 127 | DEL | (Delete) |

**ASCII printable characters**

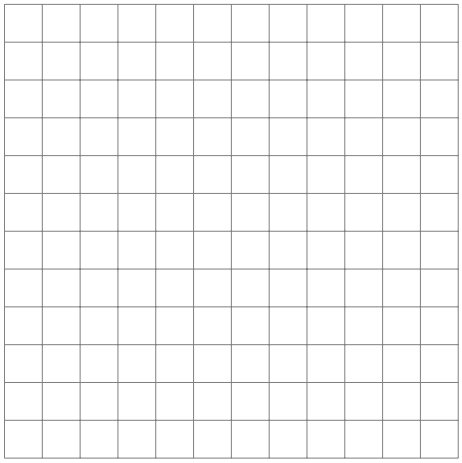
| | | | | | |
|---|---|---|---|---|---|
| 32 | space | 64 | @ | 96 | ` |
| 33 | ! | 65 | A | 97 | a |
| 34 | " | 66 | B | 98 | b |
| 35 | # | 67 | C | 99 | c |
| 36 | $ | 68 | D | 100 | d |
| 37 | % | 69 | E | 101 | e |
| 38 | & | 70 | F | 102 | f |
| 39 | ' | 71 | G | 103 | g |
| 40 | ( | 72 | H | 104 | h |
| 41 | ) | 73 | I | 105 | i |
| 42 | * | 74 | J | 106 | j |
| 43 | + | 75 | K | 107 | k |
| 44 | , | 76 | L | 108 | l |
| 45 | - | 77 | M | 109 | m |
| 46 | . | 78 | N | 110 | n |
| 47 | / | 79 | O | 111 | o |
| 48 | 0 | 80 | P | 112 | p |
| 49 | 1 | 81 | Q | 113 | q |
| 50 | 2 | 82 | R | 114 | r |
| 51 | 3 | 83 | S | 115 | s |
| 52 | 4 | 84 | T | 116 | t |
| 53 | 5 | 85 | U | 117 | u |
| 54 | 6 | 86 | V | 118 | v |
| 55 | 7 | 87 | W | 119 | w |
| 56 | 8 | 88 | X | 120 | x |
| 57 | 9 | 89 | Y | 121 | y |
| 58 | : | 90 | Z | 122 | z |
| 59 | ; | 91 | [ | 123 | { |
| 60 | < | 92 | \ | 124 | \| |
| 61 | = | 93 | ] | 125 | } |
| 62 | > | 94 | ^ | 126 | ~ |
| 63 | ? | 95 | _ | | |

**Extended ASCII characters**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 128 | Ç | 160 | á | 192 | └ | 224 | Ó |
| 129 | ü | 161 | í | 193 | ┴ | 225 | ß |
| 130 | é | 162 | ó | 194 | ┬ | 226 | Ô |
| 131 | â | 163 | ú | 195 | ├ | 227 | Ò |
| 132 | ä | 164 | ñ | 196 | ─ | 228 | õ |
| 133 | à | 165 | Ñ | 197 | ┼ | 229 | Õ |
| 134 | å | 166 | ª | 198 | ã | 230 | µ |
| 135 | ç | 167 | º | 199 | Ã | 231 | þ |
| 136 | ê | 168 | ¿ | 200 | ╚ | 232 | Þ |
| 137 | ë | 169 | ® | 201 | ╔ | 233 | Ú |
| 138 | è | 170 | ¬ | 202 | ╩ | 234 | Û |
| 139 | ï | 171 | ½ | 203 | ╦ | 235 | Ù |
| 140 | î | 172 | ¼ | 204 | ╠ | 236 | ý |
| 141 | ì | 173 | ¡ | 205 | ═ | 237 | Ý |
| 142 | Ä | 174 | « | 206 | ╬ | 238 | ¯ |
| 143 | Å | 175 | » | 207 | ¤ | 239 | ´ |
| 144 | É | 176 | ▒ | 208 | ð | 240 | ≡ |
| 145 | æ | 177 | ▓ | 209 | Ð | 241 | ± |
| 146 | Æ | 178 | ▓ | 210 | Ê | 242 | ‗ |
| 147 | ô | 179 | │ | 211 | Ë | 243 | ¾ |
| 148 | ö | 180 | ┤ | 212 | È | 244 | ¶ |
| 149 | ò | 181 | Á | 213 | ı | 245 | § |
| 150 | û | 182 | Â | 214 | Í | 246 | ÷ |
| 151 | ù | 183 | À | 215 | Î | 247 | ¸ |
| 152 | ÿ | 184 | © | 216 | Ï | 248 | ° |
| 153 | Ö | 185 | ╣ | 217 | ┘ | 249 | ¨ |
| 154 | Ü | 186 | ║ | 218 | ┌ | 250 | · |
| 155 | ø | 187 | ╗ | 219 | █ | 251 | ¹ |
| 156 | £ | 188 | ╝ | 220 | ▄ | 252 | ³ |
| 157 | Ø | 189 | ¢ | 221 | ¦ | 253 | ² |
| 158 | × | 190 | ¥ | 222 | Ì | 254 | ■ |
| 159 | ƒ | 191 | ┐ | 223 | ▀ | 255 | nbsp |

„Übersetze“ folgenden ASCII-codierten Text:
68 97 115 32 104 97 115 116 32 100 117 32 103 117 116 32 103 101 109 97 99 104 116 33

## 3. Bildcodierung (8VP)



Entschlüssele folgende Bildcodierung und zeichne das Bild

- Im ersten Byte der „Datei“ steht die Breite des Bildes
- Im zweiten Byte steht die Höhe des Bildes
- Anschließend folgen die Pixel, dabei gehören immer 3 Bits zu einem Pixel:
  - das erste Bit gibt an, ob der rote Anteil dabei angeschaltet ist
  - das zweite Bit gibt den grünen Anteil an
  - das dritte Bit gibt den blauen Anteil an
- Ist die Länge der Bitkette kein Vielfaches von 8, so werden die restlichen Bits mit Nullen aufgefüllt.

Entschlüssle mit diesen Angaben folgendes „Bild“:
00000111 00000110 11101001 00100100 10111100 11011011 01101101 00100110 00011000 01101001 00110110 00011011 01001001 10110110 11011010 01110010 01001001 00111100
*Hinweis: falls du keine Farben hast kannst du auch Buchstaben in die Kästchen schreiben. Gib dann aber auch eine Legende an!*

## 4. Datentypen bei Java (2VP+1VP)

Gib an, welche Arten von Daten Variablen von folgenden Datentypen speichern können. *Zusatzaufgabe: gib bei Zahlen an, wie groß die gespeicherten Zahlen maximal (ungefähr) sein dürfen.*

`int` _____

_____

`float` _____

_____

`char` _____

_____

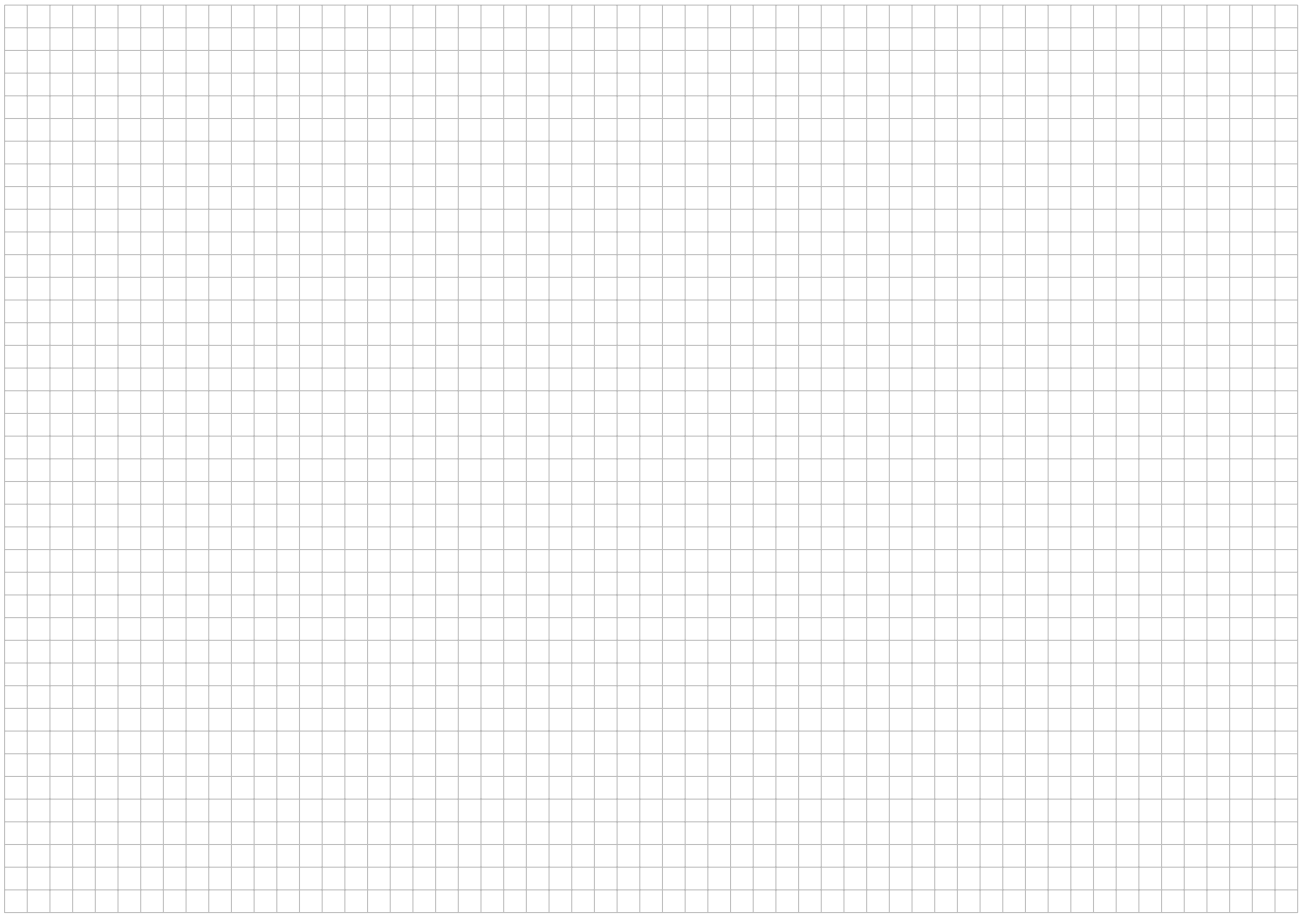
`String` _____

_____

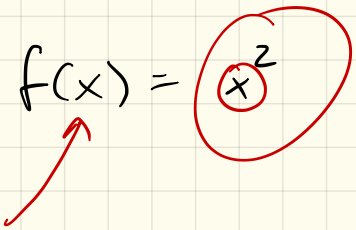
## 5. Programmierung (8VP)

Beschreibe kurz, was folgendes Programm macht. Es haben sich außerdem einige Fehler eingeschlichen. Korrigiere diese.

```
1  import java.util.Scanner;
2
3  public class Start {
4      public static void main(String[] args) {
5      int heute = 2018;
6
7          Scanner sc = new Scanner();
8      int float geburtstag = sc.nextInt();
9
10         int alter = heute - geburtstag;
11
12         if(alter => 18) {
13             System.out.println("Du bist volljährig");
14         }else{
15             System.out.println("Du bist noch nicht volljährig");
16         }
17     }
18 }
```



17.17.18




Name

Parameter

```
public static void main(String[] args){
    quadrat(5);
}
public static void quadrat(double x){
    System.out.println(x*x);
}
```

# Methoden

## Einführung

Methoden dienen dazu, wiederkehrende Befehle bzw. Befehlsabfolgen zu *kapseln* um diese nicht mehrfach programmieren zu müssen.
Man kann dadurch die eigentliche Hauptmethode `main` übersichtlicher halten und die Fehlersuche wird vereinfacht.
Eigene Methoden erstellt man **innerhalb** der Klasse und **vor** der `main`-Methode. (*Anmerkung: Prinzipiell lassen sich Methoden auch nach der `main`-Methode anlegen, in der Schule einigen wir uns darauf, dass eigene Methoden davor programmiert werden.*)
Eine Methode hat folgenden Aufbau:

```
public static void NAME(DATENTYP1 PARAMETER1, DT2 P2 [,...]) {
    ...     // Befehle
}
```

Listing 1: Aufbau einer Methode

`public static` gehört zu Beginn immer dazu. Die genaue Bedeutung davon wird erst bei der *objektorientierten Programmierung* wichtig.
`void` ist der *Rückgabetyp*, näheres dazu auf dem nächsten Arbeitsblatt.
Der Name der Methode ist frei wählbar, muss aber innerhalb einer Klasse eindeutig sein.
Eine Methode kann außerdem (beliebig viele) Parameter annehmen. Auch hier gilt wiederum: jeder Parameter ist innerhalb der Methode identisch zu einer Variablen. Deshalb müssen wir für jeden Parameter ebenfalls dessen Datentyp mit angeben.
Soll eine solche Methode nun aufgerufen werden, so geschieht das mit dem Namen der Methode. Für jeden Parameter den die Methode erwartet müssen wir den *Wert* angeben. Ein Beispiel steht unter Aufgabe 1.

## 1. Aufgabe: Potenz

Erstelle ein neues Paket `ab3` mit einer Klasse `Methoden`.
Programmiert werden soll eine Methode `Pot`, die die Potenz berechnet. Diese muss natürlich zwei Parameter annehmen:

- die Basis vom Typ `double`
- den Exponent vom Typ `int`

Nach der Berechnung der Potenz soll – ebenfalls in der Methode – das Ergebnis in der Konsole ausgegeben werden.

```
Pot( 2.5 , 3 );
```

Listing 2: Beispielaufruf der Methode

Dieser Aufruf soll folgende Ausgabe erzeugen:
`Die Potenz von 2.5 hoch 3 ist 15.625`
Rufe zunächst die Methode mit fest in den Quellcode einprogrammierten Zahlen auf. Wenn alles funktioniert erweitere das Programm so, dass die Basis und der Exponent vom Benutzer über die Konsole eingegeben werden können.

2,5 3

```
public static void Pot(double basis, int exponent) {

    double ergebnis = 1;

    for(int i=0; i<exponent; i++) {

        ergebnis = ergebnis * basis;

    }

    System.out.println(ergebnis);

}
```

1

1 · 2,5 · 2,5 · 2,5

## 2. Zusatzaufgabe: Rückgabewert

Bisher wird das Ergebnis der Rechnung lediglich auf der Konsole ausgegeben, wir können damit innerhalb unseres Programms noch nicht weiterrechnen.
Informiere dich über den *Rückgabewert* einer Methode, d. h. wie ein bestimmter Wert als Ergebnis der Methode zur weiteren Rechnung/Bearbeitung genutzt werden kann.
Beschreibe alle Änderungen am bisherigen Code so, dass du es allen deinen Mitschülern erklären könntest.

Rückgabetyp angeben: void -> keine Rückgabe
int / double / ...

Wert zurückgeben: mit return

## 3. Zusatzaufgabe: Rekursion

Informiere dich, was *Rekursion* bedeutet und beschreibe:

Auch die Potenzberechnung lässt sich rekursiv durchführen. Ändere deine Methode dahingehend ab dass die Potenz rekursiv berechnet wird.

7.1.19

# Rekursion

Fakultät: $5! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5$

$5! = 5 \cdot 4!$

$4! = 4 \cdot 3!$

$3! = 3 \cdot 2!$

$2! = 2 \cdot 1!$

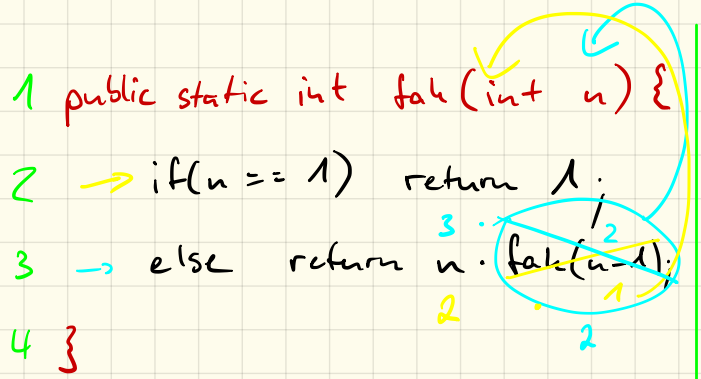
$1! = 1$

$n! = n \cdot (n-1)!$

$fak(5) = 5 \cdot fak(4)$

```
public static int fak(int n) {
    if(n == 1) return 1;
    else return n * fak(n-1);
}
```

$fak(n) = n \cdot fak(n-1)$




```
int a = fak(3);
System.out.println(a);
```

# Rekursion

*„Wer Rekursion verstehen will, muss vorher Rekursion verstehen."*

## Einführung

Unter einer *rekursiven Methode* versteht man eine Methode, die sich selbst wieder aufruft.
Damit diese Aufrufe nicht *unendlich* weitergehen und das Programm zu einem Ergebnis kommt, brauchen wir eine *Abbruchbedingung*.

## 1. Fakultät

Eine wichtige mathematische Funktion ist die Fakultät:

$$n! = 1 \cdot 2 \cdot 3 \cdots (n-2) \cdot (n-1) \cdot n$$

$$5! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 = 120$$

Diese kann mit einer rekursiven Methode `fak` berechnet werden.

a) Notiere dir zunächst die rekursiven Methodenaufrufe und ein konkretes Zahlenbeispiel (für $n = 5$) dafür:

| | |
|---|---|
| $5! = 5 \cdot 4!$ | fak(5) = 5 · fak(4) |
| $4! = 4 \cdot 3!$ | fak(4) = 4 · fak(3) |
| $3! = 3 \cdot 2!$ | fak(3) = 3 · fak(2) |
| $2! = 2 \cdot 1!$ | fak(2) = 2 · fak(1) |
| $1! = 1$ | fak(1) = 1 |

b) Wann bricht die Rekursion ab, d. h. wann wird nicht mehr erneut die Methode `fak` aufgerufen?

bei n = 1

c) Programmiere die Methode `fak` in Java.

d) Wie muss der Methodenkopf in Java aussehen?

```
public static int fak(int n)
```

e) Rufe deine Methode `fak` aus der `main`-Methode auf mit folgenden Parametern: `fak(1);` (Ergebnis: 1), `fak(3);` (Ergebnis: 6), `fak(5);` (Ergebnis: 120), `fak(11);` (Ergebnis: 39 916 800)

## 2. Fibonacci-Folge

Die Fibonacci-Folge

0 1 2 3 4 5
0,1,1,2,3,5,8,13,21,34,55,...

ist folgendermaßen definiert:

- $F(0) = 0$
- $F(1) = 1$
- $F(2) = F(0) + F(1)$
- $F(3) = F(1) + F(2)$
- oder allgemein: $F(n) = F(n-2) + F(n-1)$

Diese soll nun mit einer Methode `fibo` umgesetzt werden.

a) Notiere dir zunächst die rekursiven Methodenaufrufe dafür und das Aufrufschema für $n = 5$:

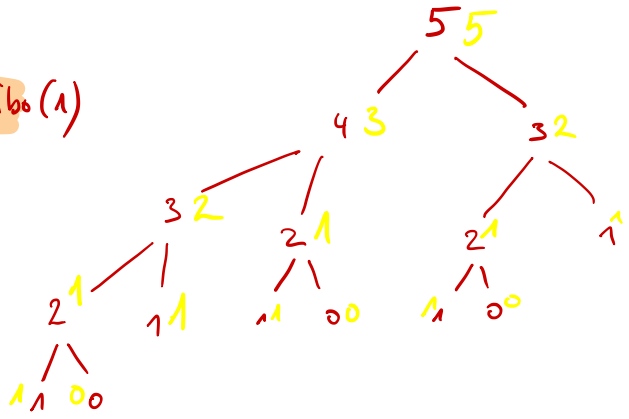
fibo(5) = fibo(4) + fibo(3)

fibo(4) = fibo(3) + fibo(2)

fibo(3) = fibo(2) + fibo(1)

fibo(2) = fibo(1) + fibo(0)

fibo(1)=1 fibo(0)=0




b) Wann bricht die Rekursion ab, d. h. wann wird nicht mehr erneut die Methode `fibo` aufgerufen?

```
if (n == 1)   return 1;
if (n == 0)   return 0;
return fibo(n-1) + fibo(n-2);
```

c) Wie muss der Methodenkopf in Java aussehen?

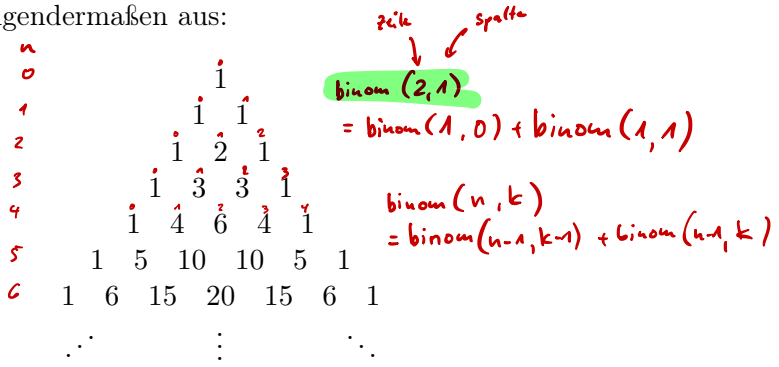
```
public static int fibo(int n)
```

d) Programmiere die Methode `fibo` in Java.

e) Rufe deine Methode `fibo` aus der `main`-Methode auf mit folgenden Parametern: `fibo(1);` (Ergebnis: 1), `fibo(3);` (Ergebnis: 2), `fibo(5);` (Ergebnis: 5), `fibo(11);` (Ergebnis: 89), `fibo(23);` (Ergebnis: 28 657)

## 3. Pascal'sches Dreieck

Das PASCALsche Dreieck sieht folgendermaßen aus:




Die Zahlen darin sollen mit der Methode `binom(n,k);` berechnet werden, wobei $n$ die Zeile und $k$ die Spalte (innerhalb dieser Zeile) angibt. (Beispiele s. unten, *Hinweis: die Zeilen- und Spaltennummerierung beginnt bei 0!*)

a) Recherchiere im Internet, wie das PASCALsche Dreieck bzw. dessen Inhalte *rekursiv* berechnet werden können.

b) Notiere dir die rekursiven Methodenaufrufe und das Aufrufschema für $n = 3$ und $k = 2$:

binom (3, 2)

binom (2, 2) + binom (2, 1)

= 1 (rechter Rand)

binom (1, 0) + binom (1, 1)

= 1 (linker Rand) = 1 (rechter Rand)

c) Wann bricht die Rekursion ab, d. h. wann wird nicht mehr erneut die Methode ~~binom~~ binom aufgerufen?

wenn ganz am Rand

wenn n == k → return 1 (rechter Rand)

wenn k == 0 → return 1 (linker Rand)

d) Wie muss der Methodenkopf in Java aussehen?

public static int binom (int n, int k)

e) Programmiere die Methode `binom` in Java. (*Hinweis: diese soll wieder direkt programmiert werden, ohne ein Objekt anzulegen, also direkt unter die `main`-Methode.*)

f) Rufe deine Methode `binom` aus der `main`-Methode auf mit folgenden Parametern: `binom(0,0);` (Ergebnis: 1), `binom(2,1);` (Ergebnis: 2), `binom(5,3);` (Ergebnis: 10), `binom(9,4);` (Ergebnis: 126), `binom(20,7);` (Ergebnis: 77 520)


6 -> durch 2 teilbar ? -> ja

3

4

5

modulo

$9 : 2 = 4$ Rest $1$ — $9 \,\%\, 2 = 1$

$9 : 3 = 3$ Rest $0$ — $9 \,\%\, 3 = 0$

$9 : 4 = 2$ Rest $1$

$9 : 5 = 1$ Rest $4$

$9 : 6 = 1$ Rest $3$

$9 : 7 = 1$ Rest $2$

$9 : 8 = 1$ Rest $1$

n ist teilbar durch t, wenn $n \,\%\, t == 0$

# Übungen zu Methoden

## 1. Summe

Programmiere eine Methode `summe`, die zwei `int`-Paramter annimmt, deren Summe berechnet und anschließend das Ergebnis der Rechnung wieder zurückgibt.

## 2. Primzahlen

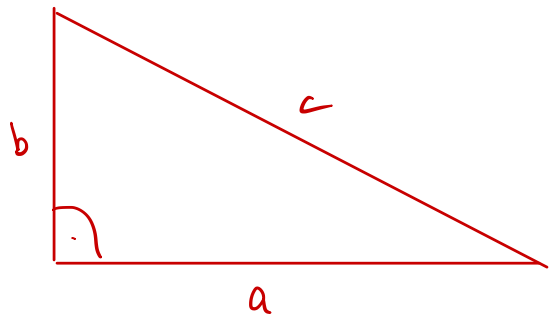
Programmiere eine Methode `istPrim`, die einen `int`-Paramter `n` annimmt. Die Methode soll überprüfen, ob der Paramter eine Primzahl ist und das Ergebnis als `boolean`, d. h. entweder `true` oder `false` zurückgeben.

## 3. Primzahl-Doubletten

Eine Primzahl-Doublette besteht aus zwei Primzahlen, deren Differenz gleich 2 ist (z. B. 3 und 5 oder 11 und 13 oder 1019 und 1021 etc.)
Programmiere eine Methode `primDoublette`, die einen `int`-Parameter `min` annnimmt. Von diesem Wert `min` soll aufsteigend nach der nächsten Primzahl-Doublette gesucht werden.
*Hinweis: es soll dabei die Methode aus Aufgabe 2 weiterverwendet werden!*

## 4. Pythagoräische Tripel

Ein pythagoräisches Tripel besteht aus drei ganzen Zahlen $a$, $b$ und $c$, die zusammen die Bedingung $a^2 + b^2 = c^2$ erfüllen, wie z. B. (3,4,5)
Programmiere eine Methode ohne Parameter, die sämtliche pythagoräischen Tripel auf der Konsole ausgibt ($a$,$b$,$c \leq 100$)




$a^2 + b^2 = c^2$
$3^2 + 4^2 = 5^2$

```
for(int a = 1; a <= 100; a++){
    for (b
        for (c
            if(...  )
    }
}
}
```

# Zusammenfassung

## 1. JAVA: Aufbau eines Programms und Syntax

### 1.1 Grundlegender Aufbau eines JAVA-Programms

Ein JAVA-Programm besteht aus (mindestens) einer Klasse. Innerhalb dieser Klasse brauchen wir die `main()`-Methode. Diese wird beim Starten des Programms aufgerufen.

```
class Name_Der_Klasse {
    public static void main(String[] args) {
        [...]
    }
}
```

Listing 1: Aufbau einer JAVA-Klasse

### 1.2 Befehle

Jeder auszuführende Befehl muss in JAVA mit einem ; abgeschlossen werden.
Dagegen werden bei Verzweigungen, Schleifen und Methoden mehrere Befehle in Blöcken, welche mit { } eingeschlossen werden, zusammengefasst.

### 1.3 import

JAVA ist grundsätzlich modular in Paketen bzw. *packages* aufgebaut. Wollen wir Befehle und Objekte nutzen, die nicht im „Standardpaket“ von JAVA sind, so müssen wir die entsprechenden packages mit `import` einbinden. (s. beispielsweise *Eingabe über die Konsole*)
*Wichtig:* diese `import`-Befehle müssen **vor** der Klasse (`class Name { [...] }`) geschrieben werden!

### 1.4 Kommentare

Um Code verständlicher zu machen können in JAVA *Kommentare* benutzt werden. Diese werden von JAVA komplett ignoriert und können vom Programmierer dazu genutzt werden, Methoden und Befehle zu beschreiben.
Es gibt zwei Arten von Kommentaren. Beschreibe kurz den Unterschied:

`// [...]` bis zum Zeilenende

`/* [...] */` von /* bis */ wird alles als Kommentar angesehen

# 2. Variablen

Eine Variable ist ein Speicherplatz für Daten, die im Programm verwendet werden können. Bei der *Deklaration* gibt man der Variable einen Typ und einen Namen. Der *Datentyp* gibt an, welche Art von Werten in der Variable gespeichert werden über den der Variablenwert später wieder abgerufen und verändert werden kann.

## 2.1 Variablentypen

Beschreibe zunächst die verschiedenen Variablentypen und gib – sofern bekannt – bei den Datentypen für Zahlen den Zahlbereich an den diese Typen aufnehmen können:

`int` ganze Zahlen von ca. -2 Mrd bis ca. +2 Mrd.

`float` Kommazahlen von $1{,}4 \cdot 10^{-45}$ bis $3{,}4 \cdot 10^{38}$

`double` Kommazahlen von $4{,}9 \cdot 10^{-324}$ bis $1{,}7 \cdot 10^{308}$

`boolean` Wahrheitswerte true / false

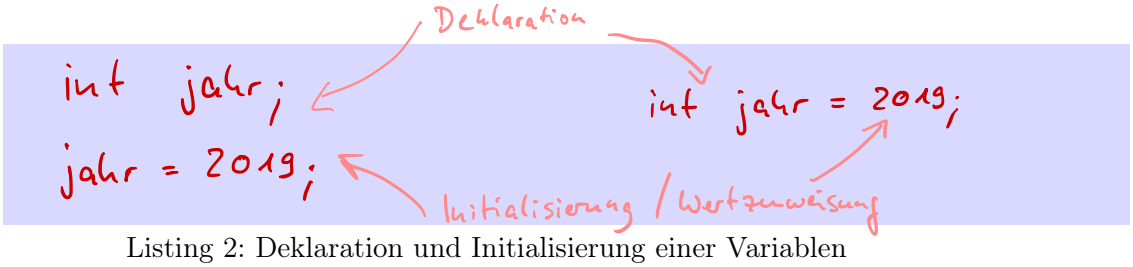
`char` einzelnes Zeichen

`String` Zeichenketten / Texte

*Hinweis:* Neben diesen Standarddatentypen gibt es noch viele weitere und wir können uns auch selbst neue Datentypen erstellen.

## 2.2 Deklaration und Initialisierung

Erzeuge eine Variable `jahr`, die einen ganzzahligen Wert speichern kann und weise ihr den Wert `2019` zu:




Listing 2: Deklaration und Initialisierung einer Variablen

# 3. Ausgabe auf der Konsole

Damit Ergebnisse auch angezeigt werden, müssen wir diese natürlich ausgeben lassen. Die einfachste Ausgabe ist auf der Konsole, hierzu gibt es den Befehl:


Listing 3: Ausgabe auf der Konsole

# 4. Eingabe über die Konsole

Die einfachste Möglichkeit, wie wir Benutzereingaben von der Konsole einlesen können, ist mit einem `Scanner`-Objekt.
Hierfür müssen wir zunächst das Paket java.util.Scanner – wie im Punkt 1.3 beschrieben – importieren.
Anschließend erzeugen wir uns ein Objekt vom Typ `Scanner` und können dann die Benutzereingabe in eine Variable einlesen:

```
// Scanner-Objekt erstellen
  Scanner sc = new Scanner(System.in);
// Einlesen einer ganzen Zahl
  int a = sc.nextInt();
```

Listing 4: Eingabe über die Konsole

*Hinweis:* neben ganzen Zahlen können auch andere Datentypen eingelesen werden. Hierfür gibt es jeweils eine entsprechende Methode des `Scanner`-Objektes.

# 5. Verzweigungen

Verzweigungen diesen dazu, bestimmte Befehle bzw. Programmteile nur unter bestimmten Voraussetzungen auszuführen.
Wir haben bisher die `if-else`-Verzweigungen kennengelernt. Diese stellt eine einfache wenn-dann-sonst-Beziehung her: **Wenn** eine Bedingung erfüllt ist **dann** wird der erste Block ausgeführt **sonst** wird der zweite Block ausgeführt.

## 5.1 Bedingungen

Bedingungen können hierbei z. B. einfache numerische Vergleiche sein:

`a == b` a genau gleich b

`a < b` **bzw.** `a > b` a kleiner als b bzw a größer als b

`a <= b` **bzw.** `a >= b` a kleiner oder gleich b bzw a größer oder gleich b

`a != b` a ungleich b

## 5.2 Beispiel

Ergänze das Beispiel:

```
// Wenn Variable "jahr" größer oder gleich 2018
   if( jahr >= 2018) {
// dann Ausgabe "Zukunft"
        System.out.print( „Zukunft“ );
   }
// sonst Ausgabe "Vergangenheit"
   else {
        System.out.print( „Vergangenheit“ );
   }
```

Listing 5: `if-else`-Verzweigung

# 6. Methoden

Methoden sind vergleichbar mit mathematischen Funktionen: beim Aufruf übergibt man ihnen bestimmte *Parameter* und die Methoden lösen dann definierte Teilaufgaben. Das Ergebnis dieser Teilaufgaben kann beispielsweise eine Bildschirmausgabe oder ein *Rückgabewert* sein.
Eine Methode wird dazu benutzt, um wiederkehrenden Code zu *kapseln* und den Programmaufbau damit logisch zu strukturieren. Die Algorithmen müssen so nur ein einziges Mal programmiert werden und können immer wieder verwendet werden.

## 6.1 Aufbau einer Methode

Eine Methode sieht im Allgemeinen wie folgt aus:

```
public static boolean istSchaltjahr(int jahr) {
  if(((jahr%4)==0)&&(((jahr%100)!=0)||((jahr%400)==0))) {
    return true;
  }
  else {
    return false;
  }
}
```

Listing 6: Aufbau einer Methode

Erkäre kurz die folgenden Teile davon:

`public`: die Methode ist von überall aufrufbar (genaueres später bei der Objektorientierung)

`static` die Methode ist ohne Objekt (genaueres später bei der Objektorientierung)

`boolean` Rückgabetyp, kann in diesem Fall „true“ oder „false“ sein
falls keine Rückgabe: void

`istSchaltjahr` Name der Methode

`(int jahr)` „Variablen“ der Methode, bzw. „Parameter“

`{ [...] }` *Methodenrumpf* Befehlsabfolge, die ausgeführt werden soll

`return [...];` Der Wert wird zurückgegeben, Methode wird abgebrochen

# 7. Schleifen

Beschreibe, wozu man Schleifen verwendet und welche 2 Schleifenarten wir benutzt haben:

Schleifen können benutzt werden um Codeabschnitte mehrfach auszuführen

**1. Schleifenart:** for-Schleife

wenn man davor schon weiß, wie oft die Schleife laufen soll

```
      Initialisierung    Bedingung    Aktualisierung
for(     int i = 0     ;     i < 10     ;     i++     )
```

**2. Schleifenart:** while-Schleife

wenn man davor noch nicht weiß, wie oft die Schleife durchlaufen soll

```
            Bedingung
while(      rest > 0       )
```

# 8. Arrays

Unter einem Array in Java versteht man einen „Container“, vergleichbar mit einem Schubladenschrank, der in der Lage ist, mehrere Objekte _____ aufzunehmen und zu verwalten.
Wir haben zwei unterschiedliche Möglichkeiten kennengelernt um Arrays zu erzeugen:

## 8.1 Deklaration und direkte Initialisierung

Vervollständige die Zeile um ein Array mit 5 Elementen zu erzeugen, das die Zahlen 1 bis 5 enthält:

```
int meinArray =
```

Listing 7: Arrays: direkte Initialisierung

Der Nachteil an dieser Möglichkeit ist, _____

_____

_____

## 8.2 Deklaration ohne Initialisierung

Um den oben genannten Nachteil zu umgehen können wir ein Array auch ohne Initialisierung deklarieren, beispielsweise mit Platz für 100 Werte:

```
int meinArray =
```

Listing 8: Arrays: Deklaration ohne Initialisierung

## 8.3 Arbeiten mit Arrays

Mit dem Ausdruck _____ können wir dann auf das dritte Element des Arrays zugreifen.
**Achtung:** der *Index* beginnt bei _____!
Die Länge eines Arrays `meinArray` können wir mit dem Ausdruck _____ bestimmen.

209 Sekunden $\hat{=}$ 3 Min + 29 Sek

Modulo

209 % 60 = 29 sek

209 / 60 = 3 min

7 9 33 sek % 60 = 13 sek → Ausgabe

7 9 33 / 60 = 132

132 % 60 = 12 min → Ausgabe

132 / 60 = 2

43 429 741 % 60 = 1 sek

43 429 741 / 60 = 723 829

723 829 % 60 = 49 min

723 829 / 60 = 12063

12063 % 24 = 15 h

12063 / 24 = 502

502 % 365 = 137 d

502 / 365 = 1 a

# Verschiedene Programmieraufgaben

## Hinweise zu Zeichenketten

Legt man eine Variable vom Datentyp `String` an, so kann mit der Methode `.length()` die Länge, bzw. mit `.chatAt(index)` der Buchstabe an Position `index` abgerufen werden. (**Achtung: Zählung beginnt bei 0!**)

```
String test = "Hello World!";

System.out.println( test.length() ); // Ausgabe: "12"
System.out.println( test.charAt(1) ); // Ausgabe: "e"
```

Listing 1: Beispiel

## 1. Aufgabe

Entwickle ein Programm, welches Strings „spiegelt“. Aus `Hello World!` soll dabei `!dlroW olleH` werden.

## 2. Aufgabe

Programmiere ein Programm, bei dem du eine Sekundenzahl eingeben kannst und daraus Jahre, Tage, Stunden, Minuten und Sekunden berechnet werden.

```
158036522 Sekunden entsprechen:
5 Jahren,
4 Tagen,
3 Stunden,
2 Minuten und
2 Sekunden.
```

Listing 2: Beispiel

*Hinweis: Du musst keine Schaltjahre berücksichtigen und kannst daher mit 365 Tagen für ein Jahr rechnen.*

## 3. Aufgabe

Gib das (kleine) Einmaleins aus.

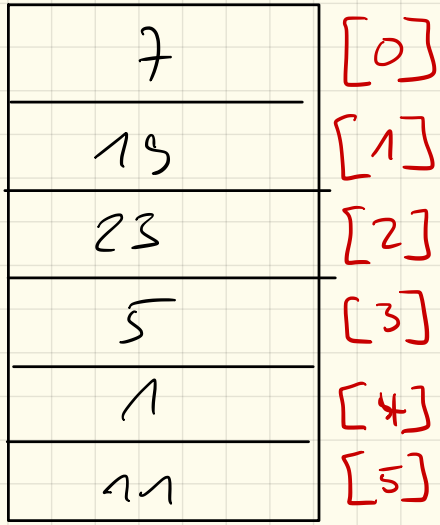
```
1  2  3  4  5  6  7  8  9  10
2  4  6  8  10 12 14 16 18 20
3  6  9  12 15 18 21 24 27 30
4  8  12 16 20 24 28 32 36 40
5  10 15 20 25 30 35 40 45 50
6  12 18 24 30 36 42 48 54 60
7  14 21 28 35 42 49 56 63 70
8  16 24 32 40 48 56 64 72 80
9  18 27 36 45 54 63 72 81 90
10 20 30 40 50 60 70 80 90 100
```

```
for(int zeile = 1; zeile <= 10; zeile++){
    for(int spalte = 1; spalte <= 10; spalte++){
        System.out.print(zeile*spalte + " ");
    }
    System.out.println();
}
```

Listing 3: Beispiel

18.2.19

# Array




```
int[] a = {7, 19, 23, 5, 1, 11};
int[] b = new int[6];
System.out.print(a[2]);
```

# Arrays

## Einführung

Ein Array ist ein Objekt, in dem mehrere Werte des gleichen Typs gespeichert werden. Man kann es also auffassen als „gruppierte Variable“:

```
int[] arr = new int[3]; // Anlegen des Arrays

arr[0] = 5; // erster Wert festlegen
arr[1] = 9; // zweiter Wert festlegen
arr[2] = arr[0] + arr[1]; // dritter Wert festlegen
```

Listing 1: Anlegen eines Arrays

Zu beachten ist dabei, dass die Zählung hierbei bei Null beginnt, ein Array mit 3 Einträgen hat also die Indices von 0 bis 2!
Das obige Array hat also folgenden Inhalt:

| | | | |
|---|---|---|---|
| **Wert:** | 5 | 9 | 14 |
| **Index:** | [0] | [1] | [2] |

*Hinweis: `Strings` können beispielsweise als solche Arrays vom Datentyp `char` angesehen werden! Der wichtigste Unterschied ist, dass bei einem String nicht bereits im Vorfeld die Länge angegeben werden muss!*
Achtung: ein Array kann nicht im Gesamten auf der Konsole ausgegeben werden! Es muss jedes Element einzeln – z. B. mit einer `for`-Schleife – ausgegeben werden.

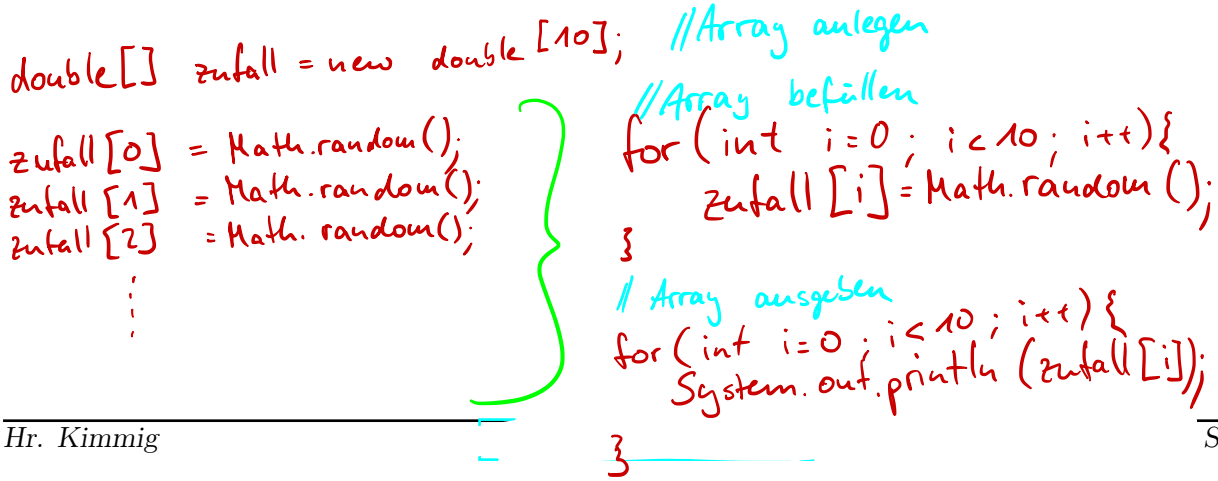
## 1. Aufgabe

Mit der Methode `Math.random();` kann man eine Zufallszahl zwischen 0 und 1 (Datentyp `double`) erzeugen.
Lasse 10 solcher Zufallszahlen erzeugen und speichere diese in einem Array. Anschließend:

a) berechne die Summe dieser 10 Zahlen

b) bestimme das Minimum und Maximum

und gib das Array, die Summe sowie Minimum und Maximum auf der Konsole aus.
*Tipp: Zur Minimum/Maximum-Bestimmung benötigst du jeweils eine weitere Variable!*



# Minimumsuche

## 1. Pseudocode

Schreibe in *Pseudocode* (d. h. in „freier Sprache“), wie die Minumumsuche funktioniert, also wie in einem gegebenen, unsortierten Array, der kleinste Wert herausgefunden werden kann:

## Einführung Zufallszahlen

Es bietet sich an, nicht mit einem vordefinierten Array zu arbeiten, sondern die Zahlen im Array *zufällig* zu erzeugen.
Java bietet hierfür die Möglichkeit Zufallszahlen[1] zu erzeugen.
Für eine bessere Übersichtlichkeit rechnen wir hier vorerst nur mit ganzen Zahlen. Mit dem Befehl `Math.random();` erzeugen wir jedoch lediglich Fließkommazahlen zwischen 0 und 1.
Für ganzzahlige Zufallszahlen müssen wir zunächst das Paket `java.util.Random` importieren. Anschließend können wir in unserem Programmablauf einen Zufallszahlengenerator erzeugen:

[1] Anmerkung: es können mit einem herkömmlichen Computer keine *echten Zufallszahlen* erzeugt werden, diese werden mit einem aufwändigen Algorithmus *berechnet*. Je besser dieser Algorithmus ist, desto zufälliger sehen die Zahlen aus. Wir sprechen deshalb auch von *Pseudozufallszahlen*.

```java
import java.util.Random;

class Zufallszahl {
    public static void main(String[] args) {
        // erzeuge Zufallszahlengenerator
        Random rand = new Random();

        //erzeuge Zufallszahl zwischen >=0 und < 50
        rand.nextInt(50);
    }
}
```

Listing 1: Erzeugung von Zufallszahlen

Der Parameter der `rand.nextInt`-Methode gibt dabei die obere Grenze der Zufallszahlen an. Im obigen Beispiel liegen die erzeugten Zufallszahlen also immer zwischen *inklusive* 0 und *exklusive* 50.

## 2. Zufallszahlengenerator

Erzeuge ein neues Java-Projekt `Sortierung`. An diesem Projekt werden wir die nächsten Wochen arbeiten.
Lege darin ein Paket `minimumsuche` mit einer Klasse `Minimum` (inklusive `main`-Methode) an.
Programmiere hier zunächst einen Zufallszahlengenerator:

a) Erzeuge zunächst ein Array, welches 20 Ganzzahlen speichern kann.

b) Befülle dieses Array mit 20 zufälligen Zahlen (zwischen 0 und 50).

c) Lasse die Werte dieses Arrays auf der Konsole kommagetrennt ausgeben.
Beispiel: `20,6,30,34,5,11,0,34,28,12,4,26,11,15,44,28,40,7,20,7`

## 3. Minimumsuche

Erweitere den Programmablauf aus Aufgabe 2 so, dass im zufällig befüllten Array der Minimale Wert und der zugehörige Index gesucht und ausgegeben wird. Verwende dazu den Ablauf aus Aufgabe 1.
Beispielausgabe: `Index: 6, Wert: 0`

## 4. Zusatzaufgabe: Minimumsuche als Methode

Um die Minimumsuche nicht jedes Mal erneut programmieren zu müssen soll nun eine passende Methode programmiert werden. Das zu durchsuchende Array soll dabei als *Parameter* an die Methode übergeben werden.
Überlege dir zunächst, was als Ergebnis der Methode zurückgegeben werden soll und programmiere anschließend diese Methode in die `Minimum`-Klasse.

## 5. Zusatzaufgabe 2: Sortierte Ausgabe

Überlege dir, wie man diese Minimumssuche dazu verwenden könnte, alle Einträge des Arrays sortiert auf der Konsole auszugeben.

| | | | | | |
|---|---|---|---|---|---|
| 7. | 11.02.19 | 12.02.19 | 13.02.19 | 14.02.19 | 15.02.19 |
| 8. | 18.02.19 | 19.02.19 | 20.02.19 | 21.02.19 | 22.02.19 |
| 9. | 25.02.19 | 26.02.19 | 27.02.19<br>D (3 Std.) | 28.02.19<br>Schmotziger | 01.03.19<br>Bewegl. Ferientag |
| 10. | 04.03.19<br>Fasnachtsmontag | 05.03.19<br>Fasnachtsdienstag | 06.03.19<br>Bewegl. Ferientag | 07.03.19<br>Bewegl. Ferientag | 08.03.19<br>Bewegl. Ferientag |
| 11. | 11.03.19 | 12.03.19 | 13.03.19<br>Hochschultag KN | 14.03.19 | 15.03.19<br>F |
| 12. | 18.03.19<br>BIO 2 / ~~ph~~ | 19.03.19<br>PH / CH 1 | 20.03.19 | 21.03.19 | 22.03.19<br>E |
| 13. | 25.03.19<br>M | 26.03.19 | 27.03.19<br>sp | 28.03.19<br>BIO 1 / CH 2 / L | 29.03.19<br>Festakt 75Jahre Abi |
| 14. | 01.04.19<br>BK / GK / GEO / G / INF / MU / REL / SP | 02.04.19<br>g 1 / mu 2 | 03.04.19 | 04.04.19<br>rel / eth | 05.04.19 |
| 15. | 08.04.19<br>inf | 09.04.19<br>ch 1+2 | 10.04.19 | 11.04.19<br>CH-Exkursion | 12.04.19 |
| **Osterferien vom 15.04. bis einschließlich 26. April 2019** | | | | | |
| 18. | 29.04.19 | 30.04.19<br>geo 4 | 01.05.19<br>Feiertag | 02.05.19 | 03.05.19 |
| 19. | 06.05.19<br>bio 1 / g 3 | 07.05.19<br>bio 2 / mu 1 | 08.05.19 | 09.05.19<br>BIO 2 / PH | 10.05.19 |
| 20. | 13.05.19 | 14.05.19 | 15.05.19 | 16.05.19<br>CH 1 | 17.05.19 |
| 21. | 20.05.19<br>Vma | 21.05.19 | 22.05.19<br>BK / GK / GEO / G / INF / MU / REL / SP | 23.05.19 | 24.05.19<br>F / bk 1 |
| 22. | 27.05.19<br>M | 28.05.19<br>g 2 / ph 1 | 29.05.19<br>E | 30.05.19<br>Christi Himmelfahrt | 31.05.19<br>Bewegl. Ferientag |
| 23. | 03.06.19<br>Fachprakt. Sport | 04.06.19<br>Fachprakt. Sport | 05.06.19<br>D | 06.06.19<br>BIO 1 / CH 2 / L | 07.06.19<br>geo 1-3 |
| **Pfingstferien von 11.06.19 bis einschließlich 21.06.19** | | | | | |
| 24. | 24.06.19 | 25.06.19<br>bk 2 / g 4 / ph 2 | 26.06.19 | 27.06.19 | 28.06.19 |
| 27. | 01.07.19<br>Mündl. Abitur | 02.07.19 | 03.07.19 | 04.07.19 | 05.07.19 |
| 28. | 08.07.19 | 09.07.19 | 10.07.19 | 11.07.19 | 12.07.19 |
| 29. | 15.07.19 | 16.07.19 | 17.07.19 | 18.07.19 | 19.07.19 |
| 30. | 22.07.19<br>Projekttage | 23.07.19<br>Projekttage | 24.07.19<br>Projekttage | 25.07.19<br>Projekttage | 26.07.19<br>Projekttage |
| **Sommerferien ab 29.07.19 bis einschließlich Dienstag, den 10.09.19** | | | | | |

| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| wert | 4 | 7 | 9 | 3 | 11 | 4 | 6 | 3 |

```
int min = 100;
int minindex = 0;

for ( int i = 0 ; i < 8 ; i++ ) {

    if ( a[i] < min ){

        min = a[i];
        minindex = i;
    }
}
```

```
public static int Minimum(int[] a) {
    int min = 100;
    int minindex = 0;
    for(int i=0; i<a.length; i++){
        if(a[i] < min){
            min = a[i];
            minindex = i;
        }
    }
    return min;
}
```

18.3.19

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 3 | 7 | 9 | 8 | ~~2~~ 100 | 4 | 3 | 6 |

---

min (array) → 4

Ausgabe array[4] → 2

array[4] = 100

min array → 0

Ausgabe array[0] → 3

array[0] = 100

⋮

for-Schleife:

- Minimumssuche
- Ausgabe Wert am Index
- Wert am Index = 100.

# Laufzeit

Minimumsuche bei 10 Werten

3 4 2 7 9 1 0 5 8 6

→ 10 Schritte

allgemein bei $n$ Werten → $n$ Schritte

Selection Sort bei 10 Werten → 10

allgemein bei $n$ Werten → $n$ mal Minimumsuche

---

$n$ mal Minimumsuche mit je $n$ Schritte

⟹ insgesamt $n^2$ Schritte

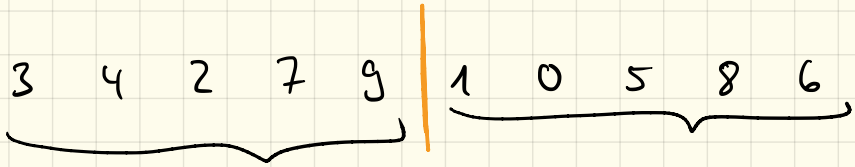
| | | | | | |
|---|---|---|---|---|---|
| 15 | 46 | 23 | 70 | 35 | 106 |
| 53 | 160 | 80 | 40 | 20 | 10 |
| 5 | 16 | 8 | 4 | 2 | 1 |

25.3.19

- Bubble Sort
- in-place <-> out-of-place
- stabiles Sortierverfahren

# Selection Sort

$n$ Elemente $\rightarrow$ $n^2$ Schritte (100)




25 + 25 Schritte

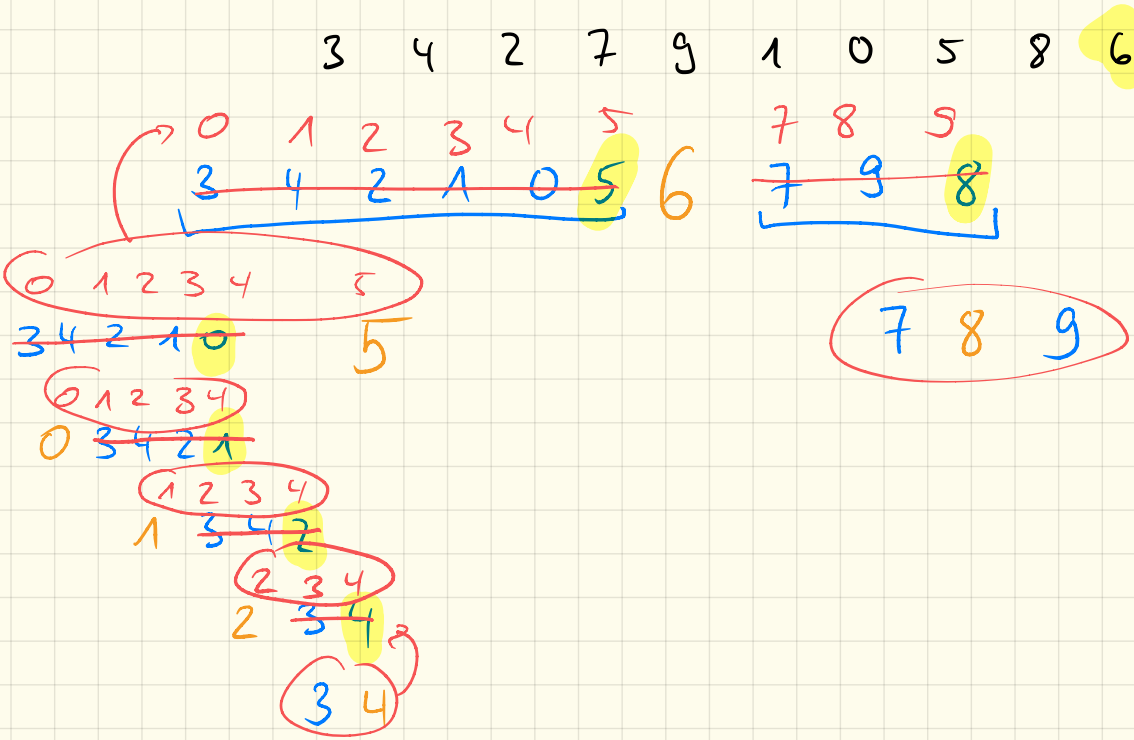
2 3 4 7 9 0 1 5 6 8

Zusammenfügen: 10 Schritte

0 1 2 3 4 5 6 7 8 9

# QuickSort

Pivot - Element



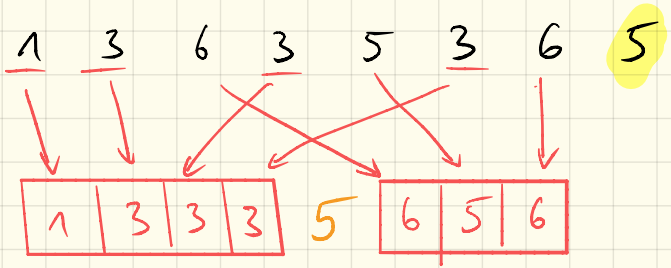
Algorithmus

1. Pivot-Element auswählen
2. anhand von Pivot-Element in Listen sortieren
3. Rekursiv für beide Teillisten durchführen
4. Zusammensetzen: Links - Pivot - Rechts

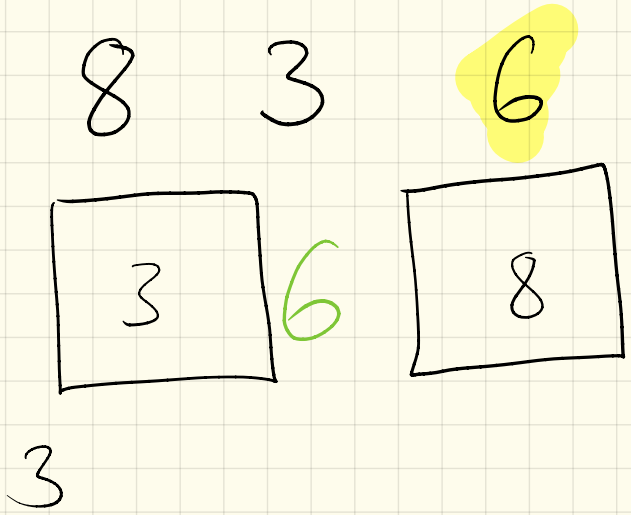
```
public static int[] quicksort(int[] arr ){
    int pivot = arr[arr.length-1];
    int größelinks: zählen, wie viele Zahlen kleiner als Pivot
    int[] links = new int[größelinks]
    int größerechts = arr.length - größelinks - 1;
    int[] rechts = new int[größerechts]
```

Schleife -> sortieren nach links und rechts

}



1.4.19




pivot = 6

links = [3]

rechts = [8]

Abbruchbedingung: wenn array aus max 1 Zahl besteht -> fertig

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 3 | 4 | 2 | 7 | 9 | 1 | 0 | 5 | 8 | 6 |

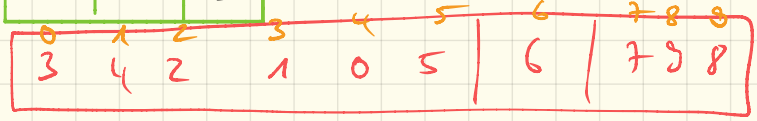
pivot: 6

links:

| 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 3 | 4 | 2 | 1 | 0 | 5 |

rechts:

| 0 | 1 | 2 |
|---|---|---|
| 7 | 9 | 8 |




```
int li = 0;
int ri = 0;

for (int i = 0; i <= arr.length - 2; i++) {
    if (arr[i] < pivot) {   // links einsortieren
        links[li] = arr[i];
        li++;
    }
```

```
        else {  // rechts einsortieren
            rechts[ri] = arr[i];
            ri++;
        }
    }
```

# Klausur 8.4.

- Rekursion
- Sortierverfahren
  - SelectionSort (Minimumsuche)
  - BubbleSort
  - QuickSort
  - Begriffe: in-place / out-of-place / stabil