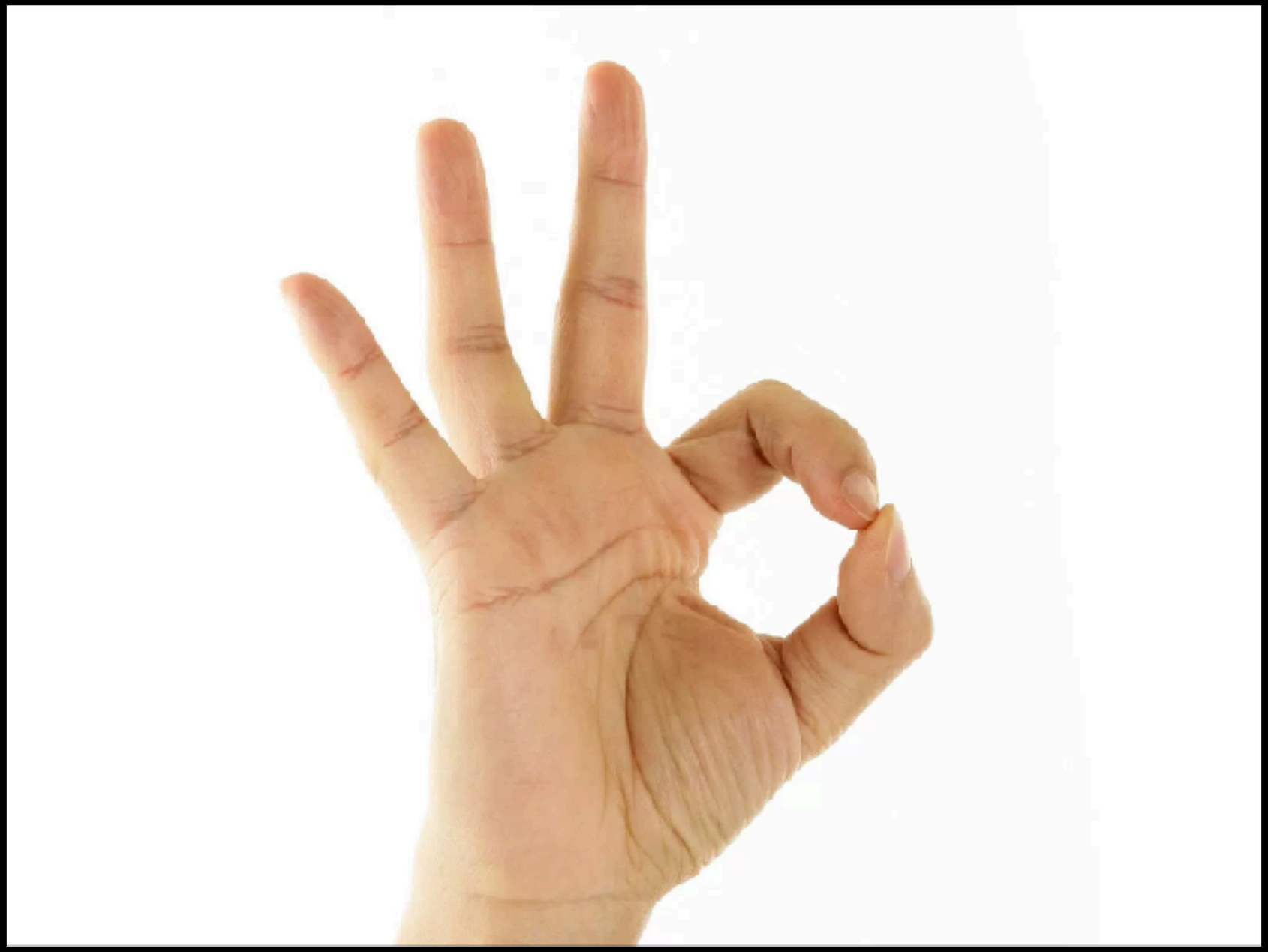
Informatik Kursstufe 2-stündig

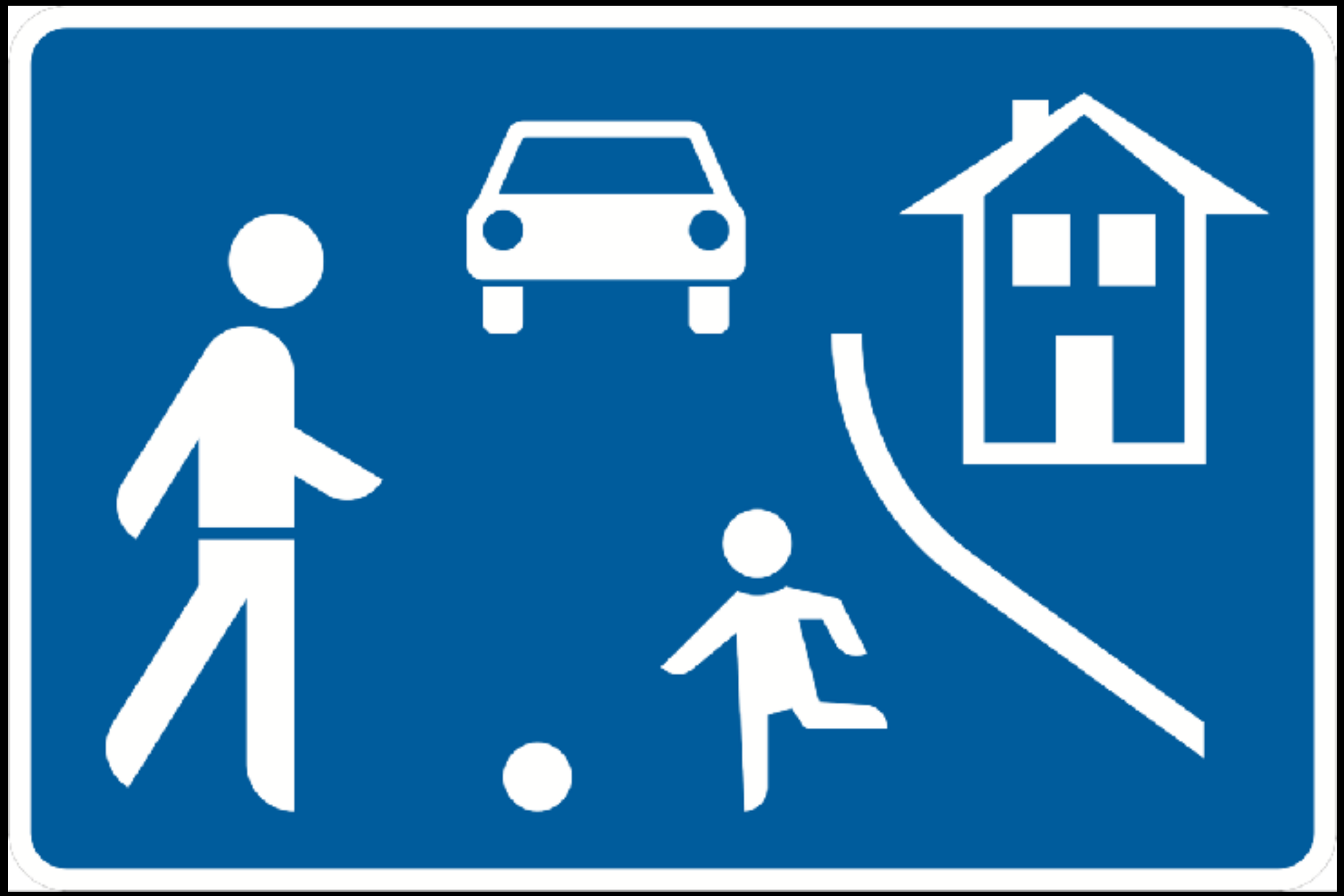
Schuljahr 18/19

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

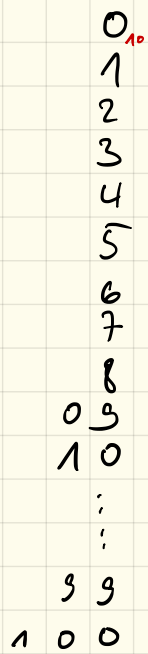
Nachrichten sind Angaben über einen Sachverhalt. Sie sind für uns stets erkennbar (in Wörtern, Bildern, Zahlen). Nachrichten enthalten Informationen und Informationen werden im Computer durch Daten dargestellt.
Daten sind demnach die computergerechte Form von Nachrichten. Daten sind also Träger von Informationen.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

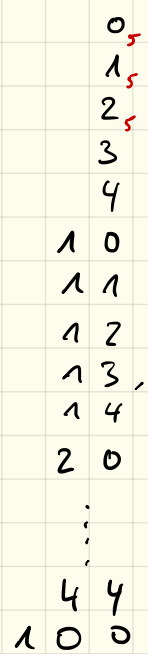
3. Codierung

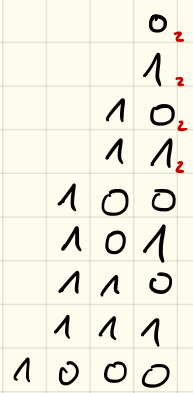
Wie werden Daten dargestellt?
Zwischen uns Menschen werden Informationen oft durch Zeichen übertragen (Schreiben, Lesen). Dazu wird das Alphabet benötigt, da alle Wörter aus Buchstaben zusammengesetzt werden.
Es gibt auch andere Möglichkeiten zur Informationsübertragung. Zum Beispiel die Farben Rot, 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.

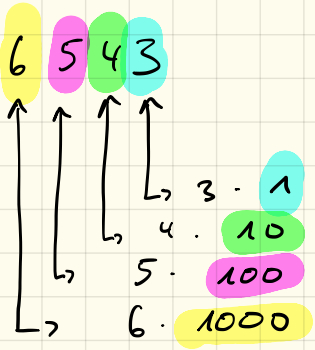
1. Aufgabe

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

$1_2 = 1$

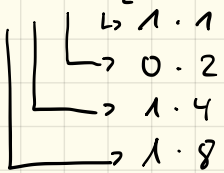
$10_2 = 2$

$100_2 = 4$

$1000_2 = 8$

$10000_2 = 16$

$1101_2 = 13$



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

$10_2 = 2_{10}$ ✓

$101_2 = 6_{10}$ ✓

$1101_2 = 13_{10}$ ✓

$10101_2 = 21_{10}$ ✓

$100001_2 = 33_{10}$ ✓

$101111_2 = \cancel{47}\ 47_{10}$ ✓

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

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

$1 \cdot 8 + 1 \cdot 1$

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}

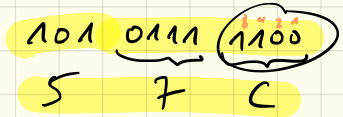
f) 1024 400_{16}

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



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) $\overbrace{0110}^{6}\overbrace{1001}^{9}_2 = 69_{16}$

b) $\overbrace{0101}^{5}\overbrace{1101}^{13}_2 = 5D_{16}$

c) $\overbrace{0010}^{2}\overbrace{0101}^{5}\overbrace{0010}^{2}_2 = 252_{16}$

d) $\overbrace{0011}^{3}\overbrace{0100}^{4}\overbrace{1101}^{13}_2 = 34D_{16}$

e) $\overbrace{0111}^{7}\overbrace{0011}^{3}\overbrace{0010}^{2}_2 = 732_{16}$

f) $\overbrace{1011}^{11}\overbrace{1010}^{10}\overbrace{1010}^{10}_2 = BAA_{16}$

g) $\overbrace{1010}^{10}\overbrace{0010}^{2}\overbrace{0100}^{4}\overbrace{1010}^{10}_2 = A24A_{16}$

h) $\overbrace{1110}^{14}\overbrace{1010}^{10}\overbrace{0101}^{5}\overbrace{1111}^{15}_2 = EA5F_{16}$

4. Aufgabe

Wandle die folgenden Hexadezimalzahlen ins Binärsystem um:

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

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

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

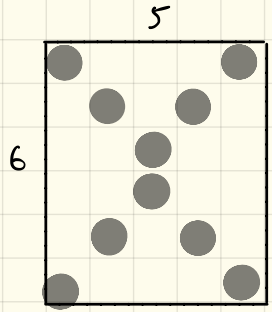
d) $1FC_{16} = 1\ 1111\ 1100_2$

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

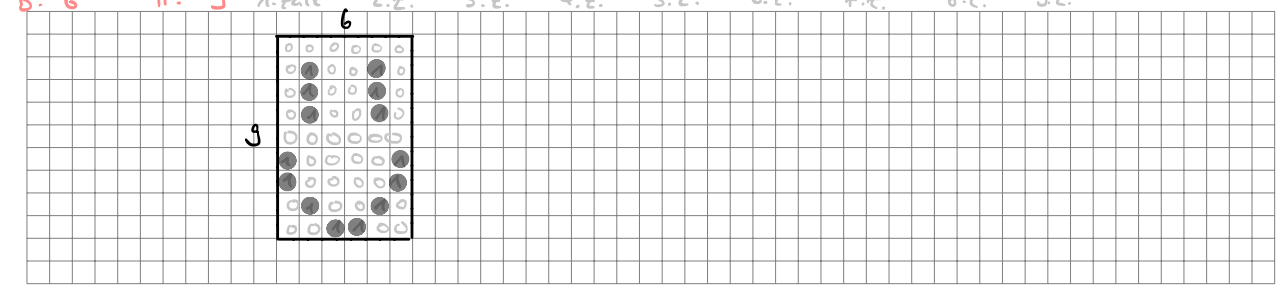
1. Schwarz-Weiß-Pixelbilder

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

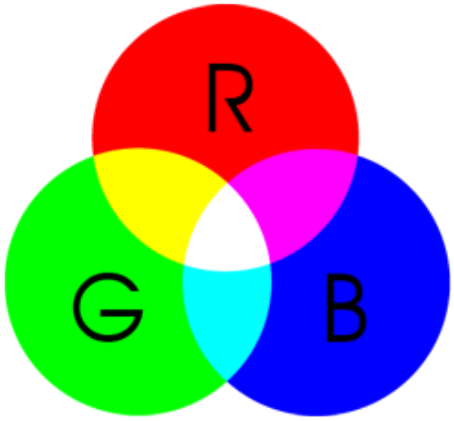


3. Aufgabe

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:



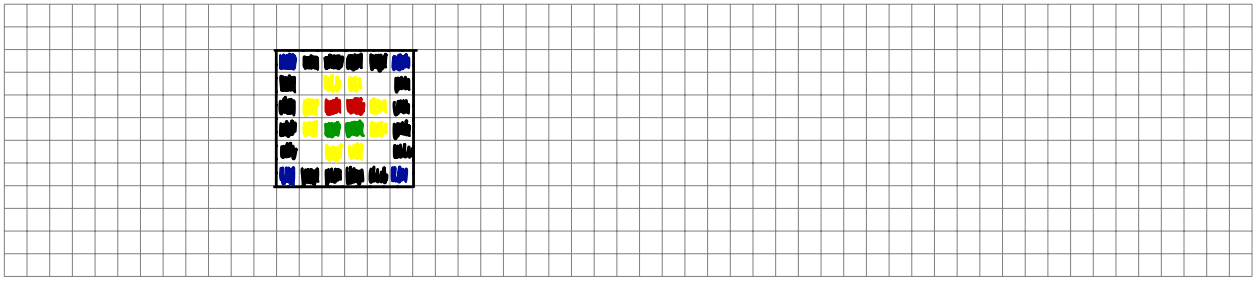
- Rot
- Grün
- Blau
- Rot + Grün = Gelb
- Rot + Blau = Magenta
- Blau + Grün = Cyan
- Rot + Grün + Blau = Weiß

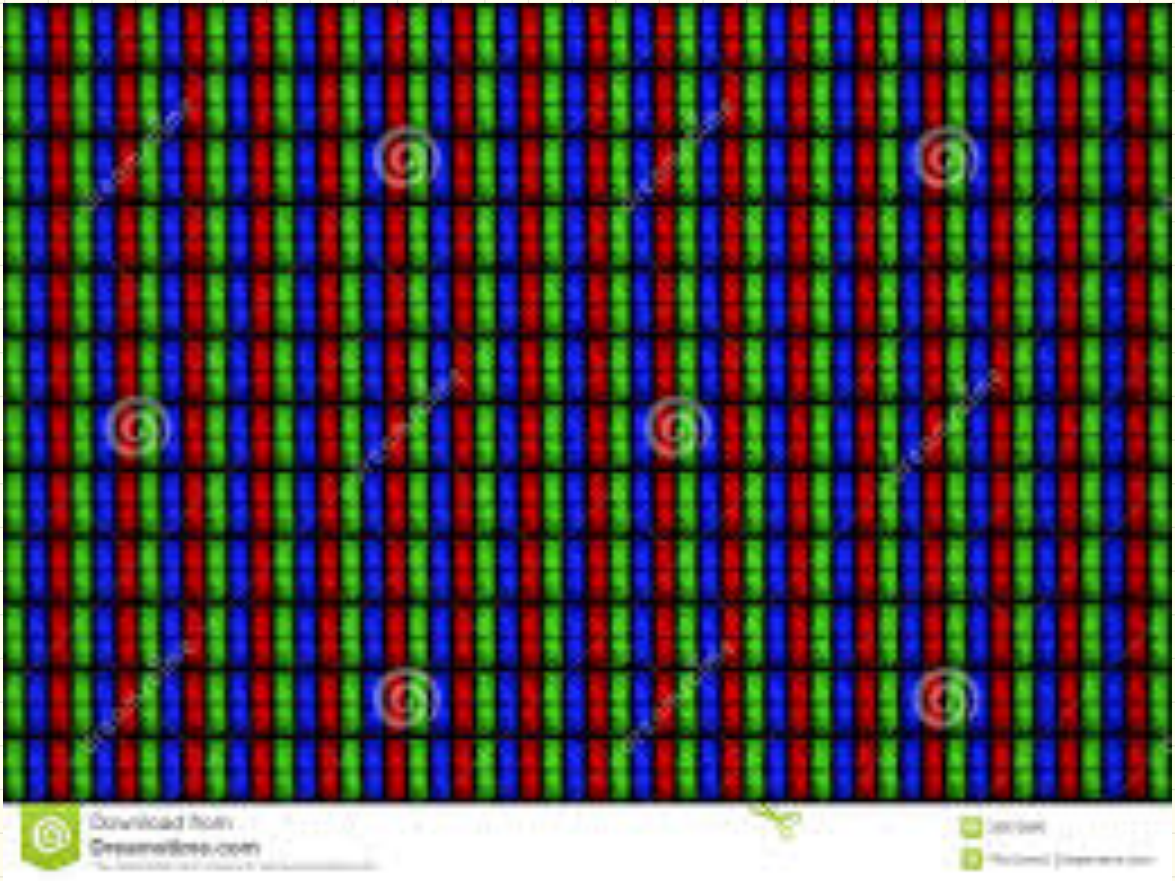
Damit definieren wir ein neues Dateiformat:

- 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

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

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?

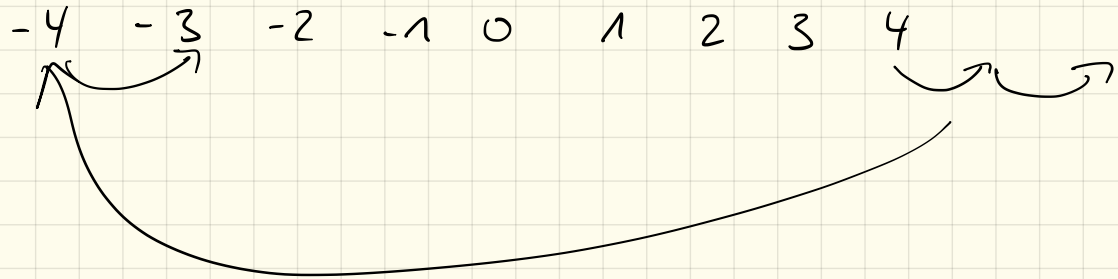
Wie könnten wir verhindern, dass das Programm mit einem Fehler abstürzt?

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.
```

Listing 4: Ausgabe Getränkeautomat