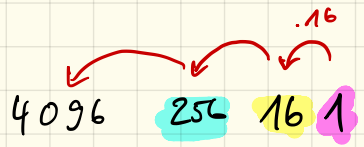
Informatik Kursstufe 2-stündig
Schuljahre 17/18 und 18/19

Hexadezimalsystem



C 1 =

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

$= 940$

Hexadez.	Dez
0	0
1	⋮
⋮	
9	9
A	10
B	11
C	12
D	13
E	14
F	15

16 8 4 2 1

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

256 16 1

A A B

$= 10 \cdot 256$

$= 11 \cdot 11$

$= 11 \cdot 1$

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) $C1_{16}$ = 12·16 + 1·1 = 193

b) $F7_{16}$ = 15·16 + 7·1 = 247

c) AAB_{16} = 10·256 + 10·16 + 11·1 = 2731

d) $1FC_{16}$ = 1·256 + 15·16 + 12·1 = 508

e) 123_{16} = 291

f) $5AB_{16}$ = 1451

g) $73A1_{16}$ = 29601

h) $13AF_{16}$ = 5039

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 = $FFFF_{16}$

0	0000	0
1	0001	1
2	0010	2
3	0011	3
⋮	⋮	⋮
14	1110	E
15	1111	F

$$
\begin{aligned}
27 &= 1 \cdot 16 + 11 \\
11 &= 1 \cdot 8 + 3 \\
3 &= 0 \cdot 4 + 3 \\
3 &= 1 \cdot 2 + 1 \\
1 &= 1 \cdot 1 + 0
\end{aligned}
$$

$$
\begin{aligned}
27 &= 1 \cdot 16 + 11 \\
11 &= B \cdot 1 + 0
\end{aligned}
$$

$$27 = 1B_{16}$$

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) 01101001_2 = 105$_{10}$ (6 9 $_{16}$)

b) 01011101_2 = 5D

c) 001001010010_2 = 252

d) 11010011101_2 = 34D

e) 11100110010_2 = 732

f) 101110101010_2 = BAA

g) 1010001001001010_2 = A24A (A 2 4 A)

h) 1110101001011111_2 = EA5F

4. Aufgabe

Wandle die folgenden Hexadezimalzahlen ins Binärsystem um:

a) $C1_{16}$ = 11000001

b) $F7_{16}$ = 1111 0111

c) AAB_{16} = 101010101011

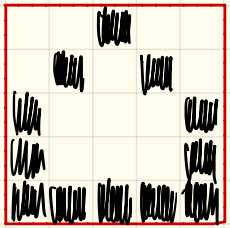
d) $1FC_{16}$ = 111111100

e) 123_{16} = 100100011

f) $5AB_{16}$ = 10110101011

g) $73A1_{16}$ = 111001110100001

h) $13AF_{16}$ = 1001110101111



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

Breite

00000101 00000101 | 00100010 | 10100011 | 00011111 | 1000 0000

Höhe

Codierung: Bilder

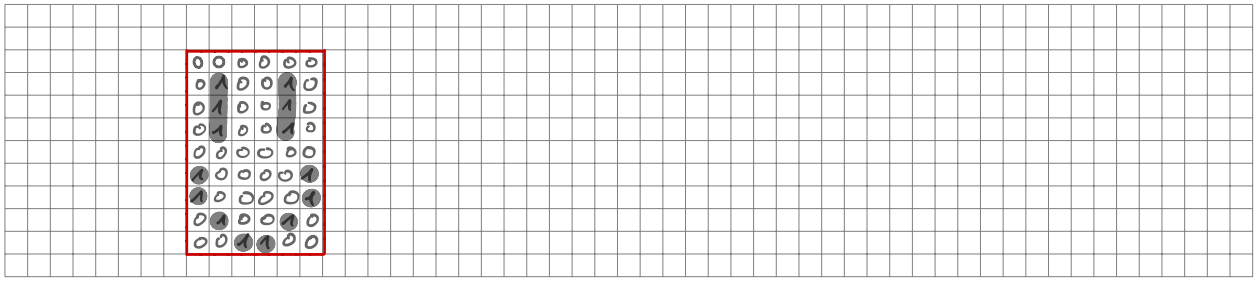
1. Schwarz-Weiß-Pixelbilder

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.

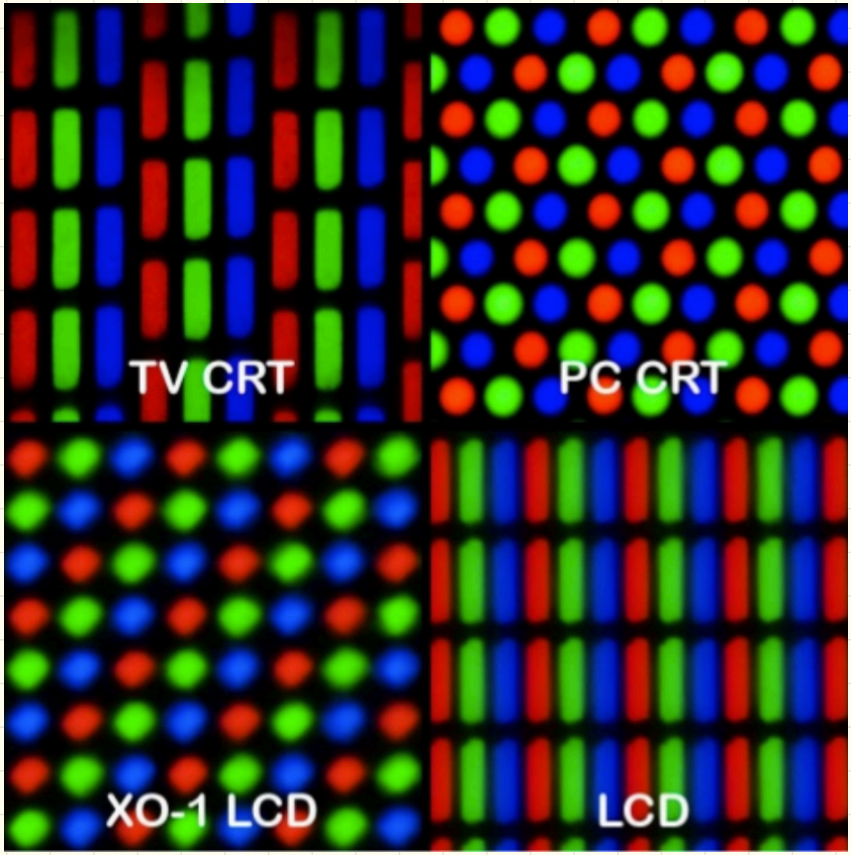
2. Aufgabe

Entschlüssle mit obigen Angaben folgendes „Bild“:
00000110 00001001 00000001 00100100 10010010 00000010 00011000 01010010 00110000
Breite: 6 Höhe: 9



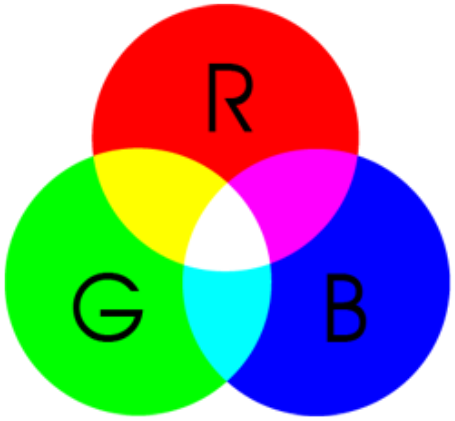
3. Aufgabe

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


TV CRT
PC CRT
XO-1 LCD
LCD

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:

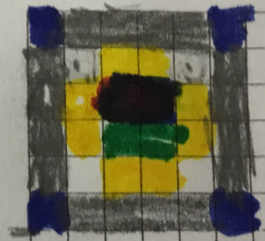
- Im ersten Byte der „Datei“ steht die Breite des Bildes
- Im zweiten Byte steht die Höhe des Bildes
- Anschließend folgen die Pixel, dabei gehören immer 3 Bits zu einem Pixel:
  - das erste Bit gibt an, ob der rote Anteil dabei angeschaltet ist
  - das zweite Bit gibt den grünen Anteil an
  - das dritte 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

5. Aufgabe

Entschlüssle mit obigen Angaben
00000110 00000110 00100000 00
10110000 00011111 01101110 000



1 Byte = 8 Bit $\hat{=}$ 256 Möglichkeiten

ASCII:

73	110	102	111	114	109	97	116	105	107	32	105	115	116	32	116	111	108	108	33
I	n	f	o	r	m	a	t	i	k		i	s	t		t	o	l	l	!

30. 11. 17

Klausur – Termin

~~(11. 1. PH / EH)~~

oder

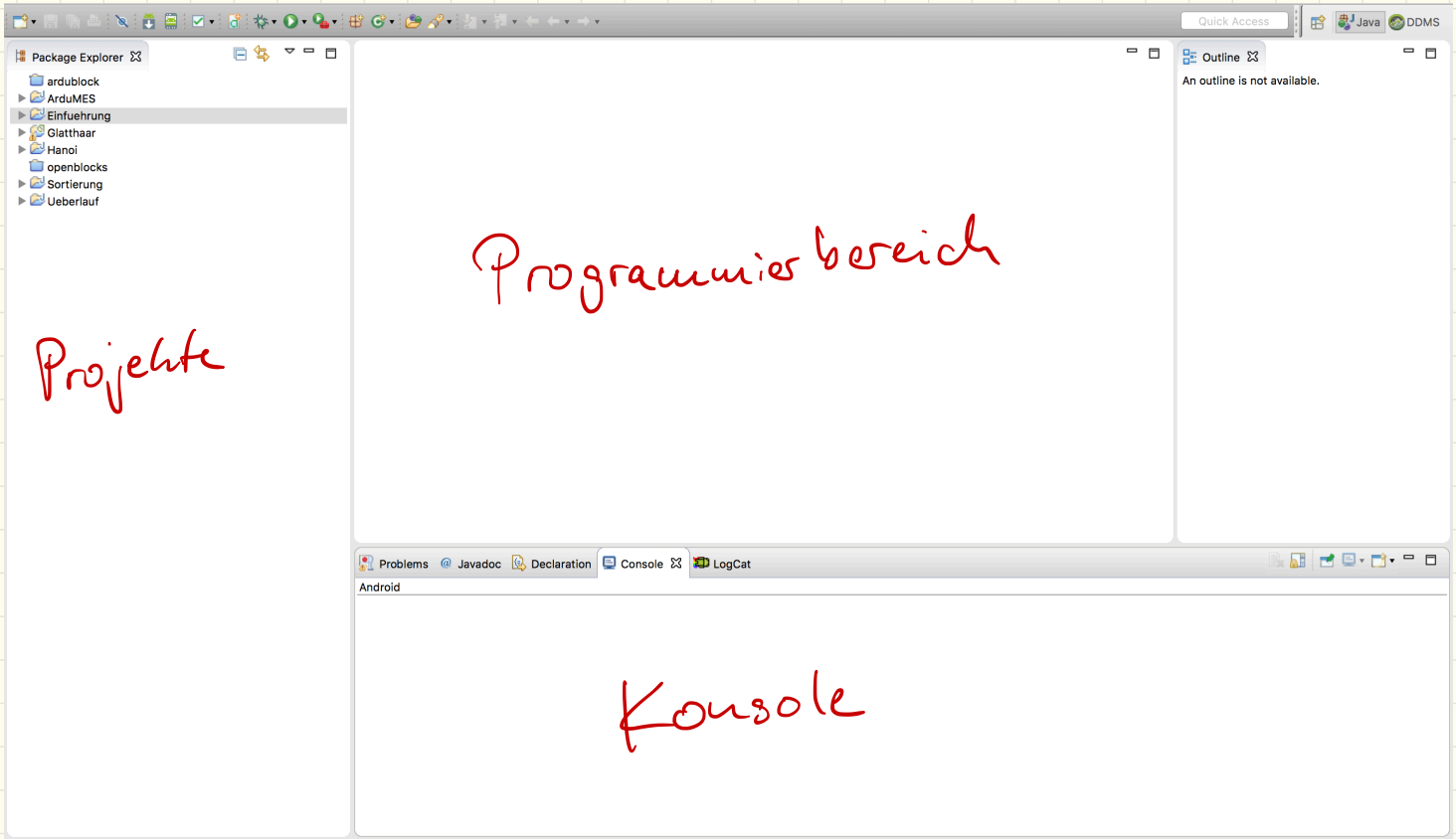
18. 1.

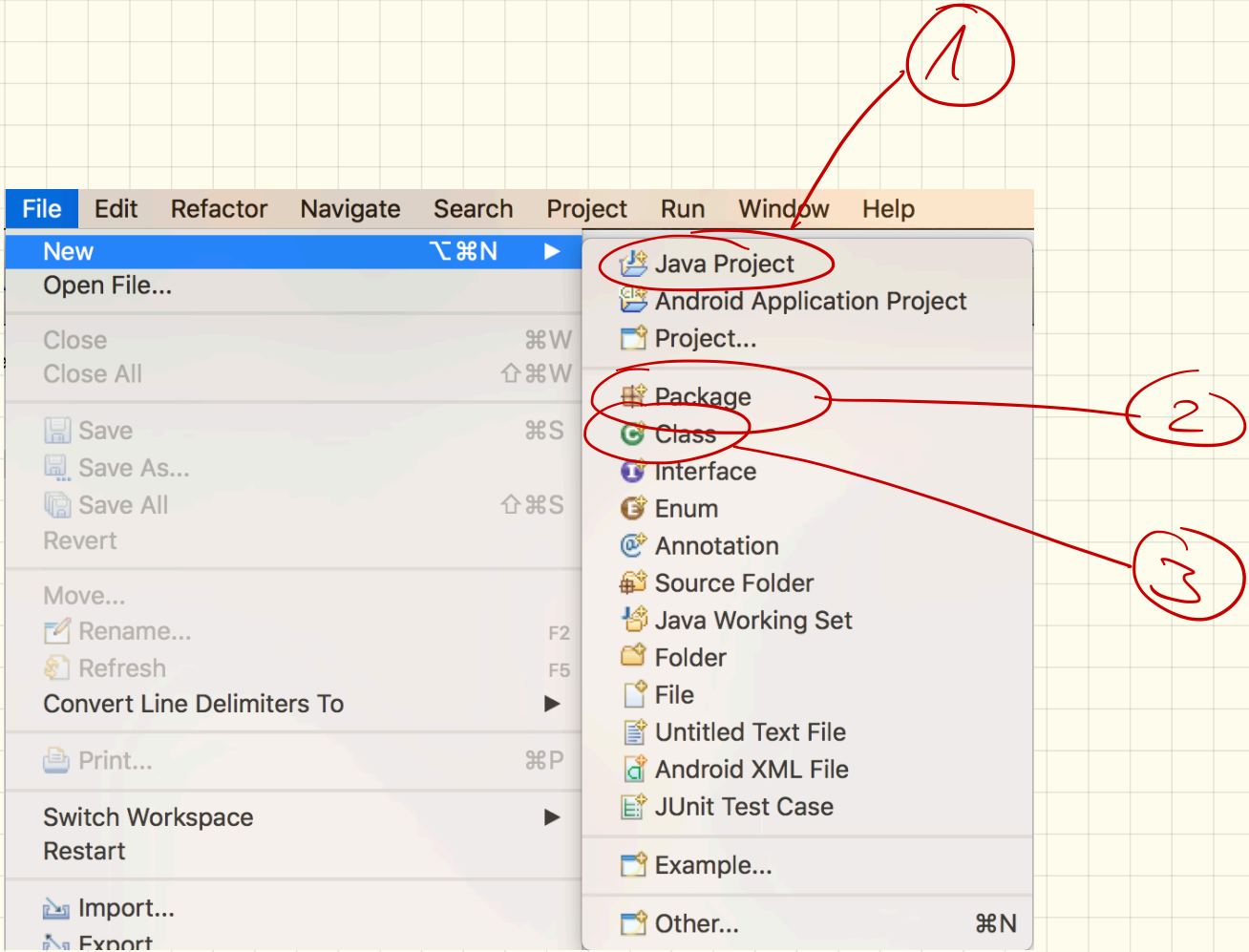
Mo Di Fr
M / GR / MV / Bio

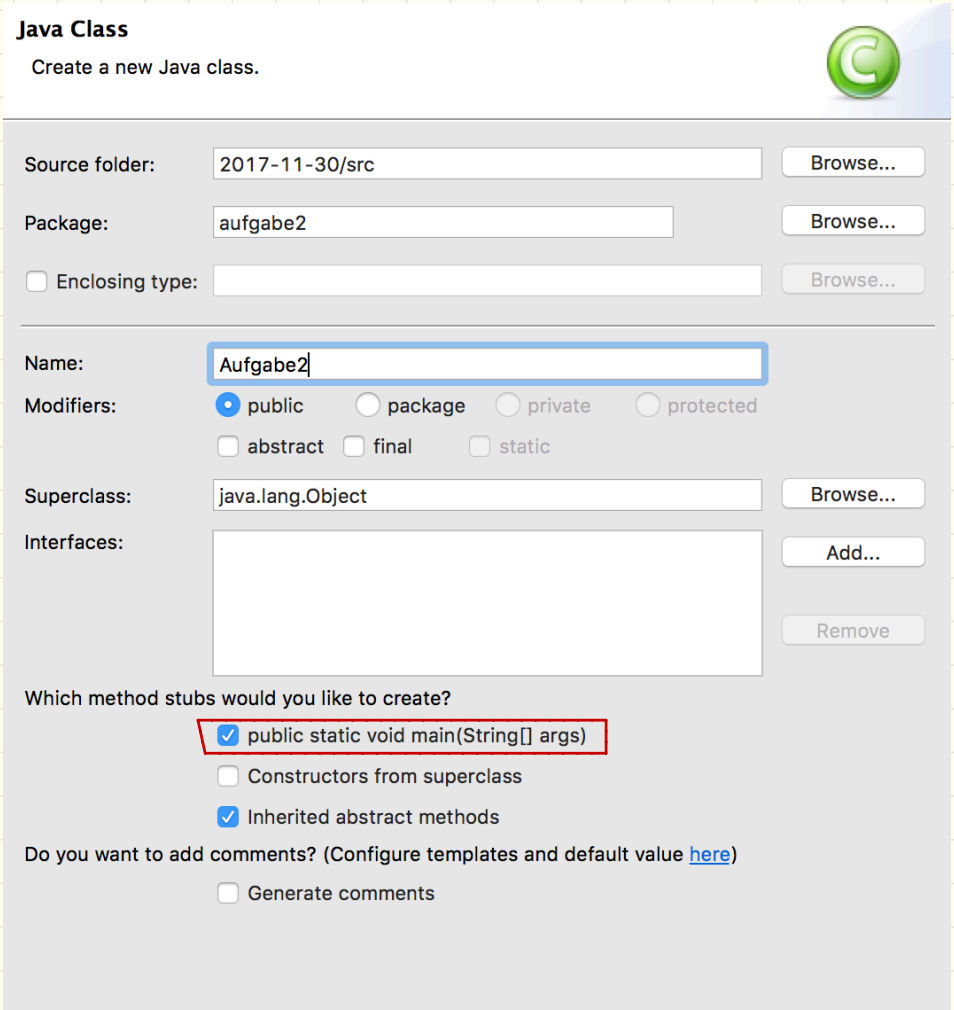
Zeugniskonferenzen (DHG) 22. – 24. 1.
(AMG) 24. – 25. 1.

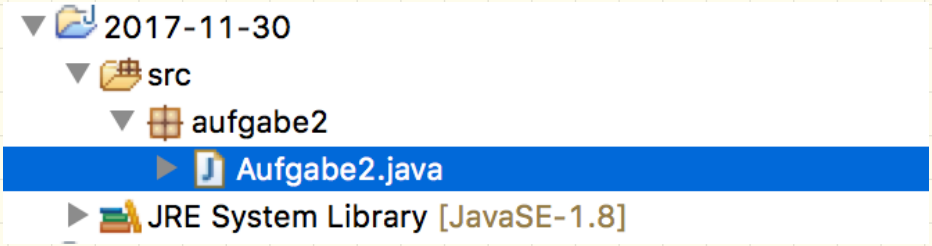
Einführung Programmierung

30.11.17




File
Edit
Refactor
Navigate
Search
Project
Run
Window
Help
New ⌥⌘N
Open File...
Close ⌘W
Close All ⇧⌘W
Save ⌘S
Save As...
Save All ⇧⌘S
Revert
Move...
Rename... F2
Refresh F5
Convert Line Delimiters To
Print... ⌘P
Switch Workspace
Restart
Import...
Export
Java Project
Android Application Project
Project...
Package
Class
Interface
Enum
Annotation
Source Folder
Java Working Set
Folder
File
Untitled Text File
Android XML File
JUnit Test Case
Example...
Other... ⌘N
1
2
3


Java Class
Create a new Java class.
Source folder:
2017-11-30/src
Browse...
Package:
aufgabe2
Browse...
Enclosing type:
Browse...
Name:
Aufgabe2
Modifiers:
public
package
private
protected
abstract
final
static
Superclass:
java.lang.Object
Browse...
Interfaces:
Add...
Remove
Which method stubs would you like to create?
public static void main(String[] args)
Constructors from superclass
Inherited abstract methods
Do you want to add comments? (Configure templates and default value here)
Generate comments




```java
package aufgabe2;

public class Aufgabe2 {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

    }

}
```

Datentypen

int	ganze Zahlen
bool	1 oder 0 / true oder false
char	einzelne Buchstaben
byte	ganze Zahlen
float	Kommazahlen
double	Kommazahlen
String	Zeichenketten

11.1.2018

Inhalte Klausur

* Codierung
  - Binärsystem
  - Hexadezimalsystem
  - Bildcodierung
  - Textcodierung
* Programmierung
  - Datentypen
  - einfachen Code lesen und beschreiben

Eingabe über die Konsole

1. `import java.util.Scanner;`

2. `Scanner sc = new Scanner(System.in);`

   Scanner = Datentyp

   `int a = sc.nextInt();`

2. Fehler – Teil 1

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

Fehler: durch Null darf man nicht teilen

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

Abfrage, ob zweite Zahl Null ist

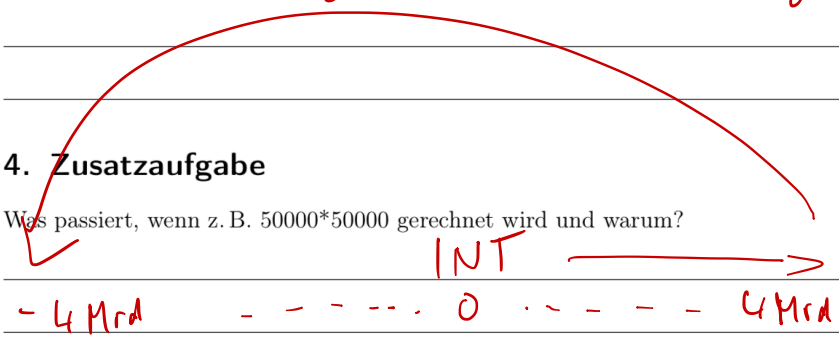
3. Fehler – Teil 2

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

Division von ganzen Zahlen (int) ergibt ganze Zahl

4. Zusatzaufgabe

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



byte 0 - 255

$255 + 1 = 0$ + Flag

```java
import java.util.Scanner;

public class Start {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int a = sc.nextInt();
        int b = sc.nextInt();
        sc.close();

        System.out.println("a+b="+(a+b));
        System.out.println("a-b="+(a-b));
        System.out.println("a*b="+(a*b));
        if( b == 0 ) {
            System.out.println("Division nicht erlaubt");
        }
        else {
        System.out.println("a/b="+(a/b));
        }


    }

}
```

if(b != 0)

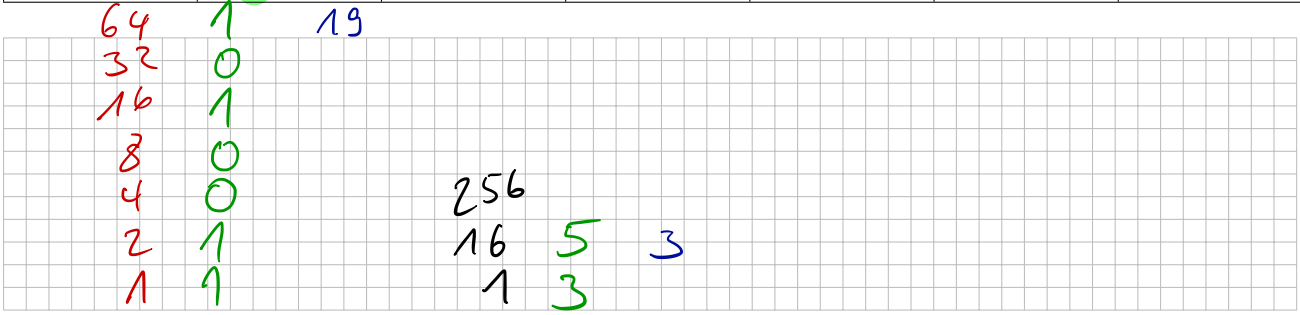
Klausur 18.01.2018

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

1. Umrechnung (12VP)

Ergänze folgende Tabelle:

	a)	b)	c)	d)	e)	f)
Dezimal	83			207		
Binär	1010011	10010111			1110011	
Hexadezimal	53		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

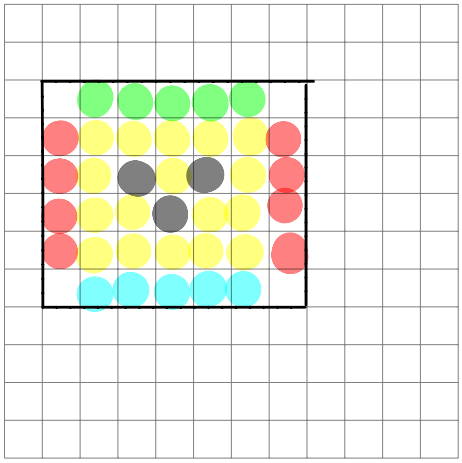
32	space	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	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` ganze Zahlen ca. 4 Mrd

`float` Dezimalzahlen $3{,}4 \cdot 10^{38}$

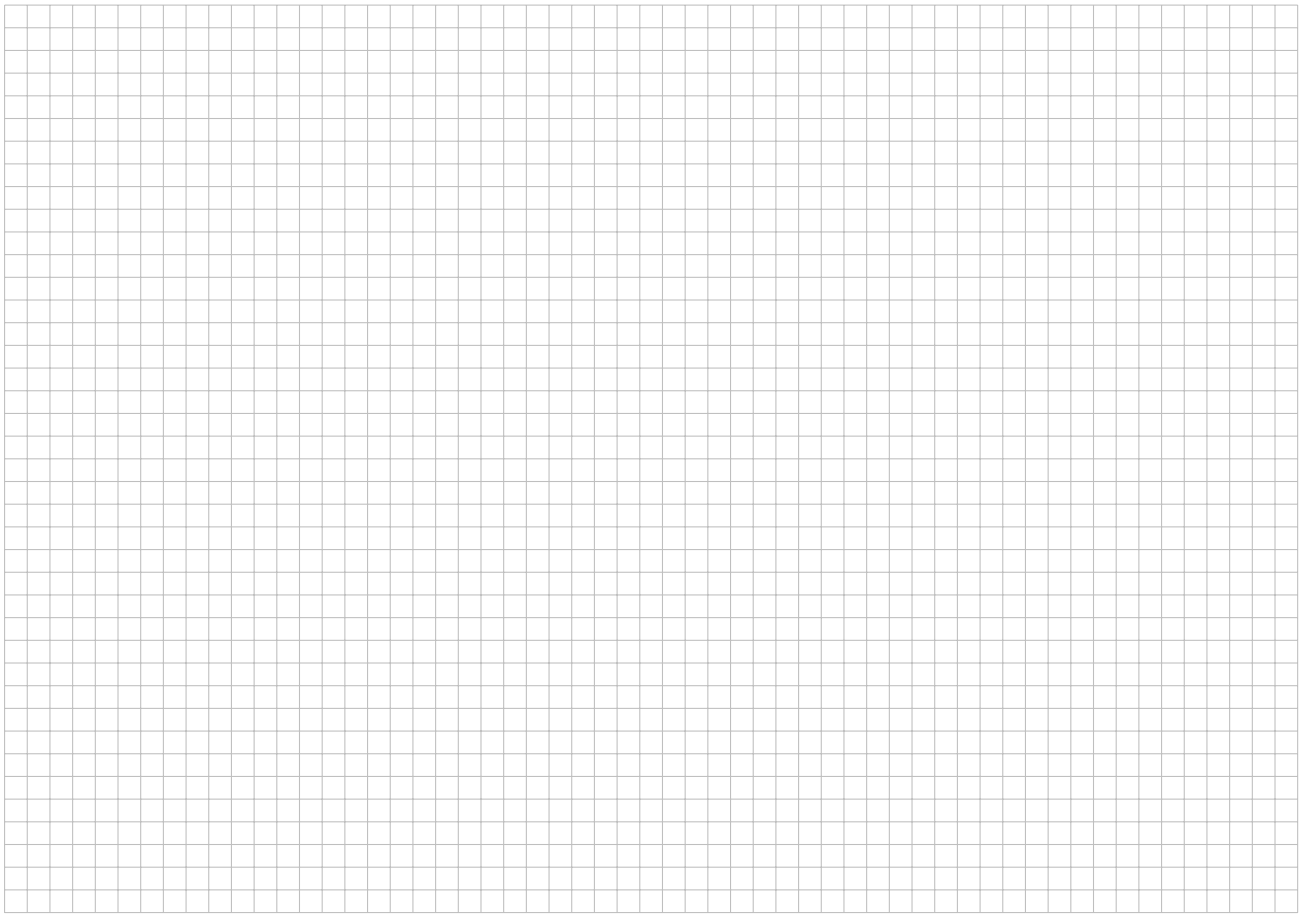
`char` einzelnes Zeichen

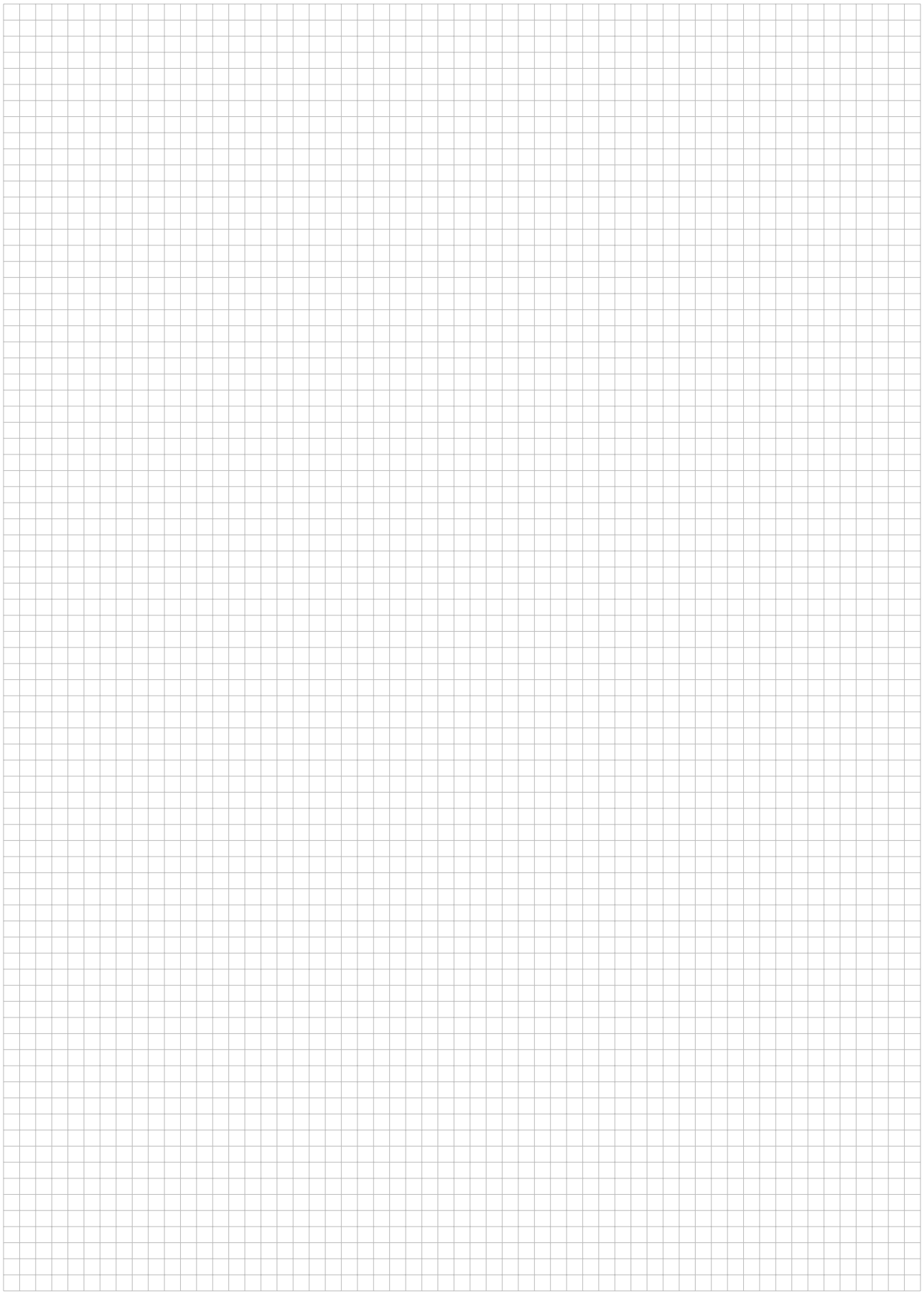
`String` Zeichenkette

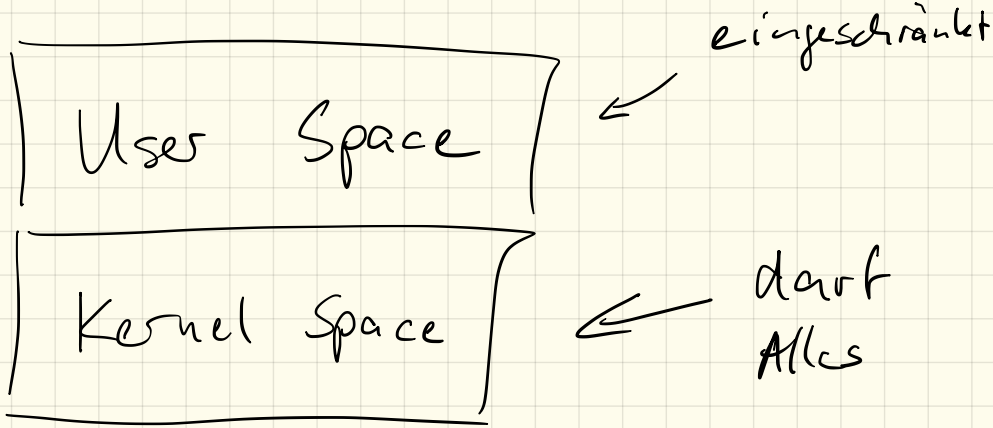
5. Programmierung (8VP)

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                                  System.in
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 }
```






User Space
eingeschränkt
Kernel Space
darf Alles

```
int i = 0;
while( i < 10 ) {
    System.out.println("Hallo");
    i++;
}
System.out.println("Ende");
```

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

können, und diese bei der Ausgabe auch in eingegebener Reihenfolge ausgegeben werden.

1.4 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

Schleifen

1. Aufgabe: Schleifen

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

Wichtig: beide Schleifentypen lassen sich auch mit einer Schleife vom anderen Typ programmieren!

1.1 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“

1.2 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
```

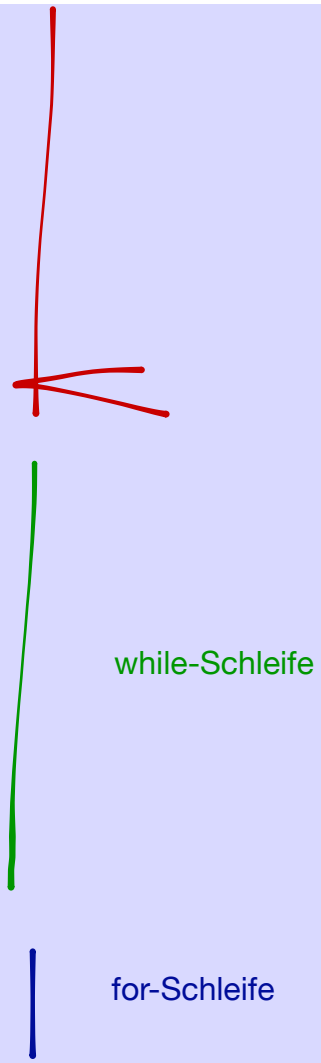
Listing 3: Rückgeld

1.3 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.

1.4 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

```java
public class demo {

    public static void main(String[] args) {

        System.out.println("Das Quadrat von 7 ist "+(7*7));
        System.out.println("Das Quadrat von 5 ist "+(5*5));
        System.out.println("Das Quadrat von 49 ist "+(49*49));

        quadrat(44);

    }

    public static void quadrat(int a) {
        System.out.println("Das Quadrat von "+a+" ist "+(a*a));
    }

}
```

```java
public static void main(String[] args) {
    Pot(2.5 , 3);
    Pot(1.3 , 5);
    Pot(0.829 , 17);
}
```

```java
public static void Pot(double basis, int exponent) {
    double ergebnis = basis;

    for(int i=1 ; i<exponent; i++) {
        ergebnis = ergebnis*basis;
    }

    System.out.println("Die Potenz von "+basis+" hoch "+exponent+" ist "+ergebnis);
}
```

```java
public static void main(String[] args) {
    double p = Pot(9, 3);
    System.out.println("Das Doppelte davon ist: "+(p*2));
}

public static double Pot(double basis, int exponent) {
    double ergebnis = basis;

    for(int i=1 ; i<exponent; i++) {
        ergebnis = ergebnis*basis;
    }

    System.out.println("Die Potenz von "+basis+" hoch "+ex

    return ergebnis;
}
```

729

9

3

ergebnis = 729

$2{,}5^3 = 1 \cdot 2{,}5 \cdot 2{,}5 \cdot 2{,}5$

ergebnis = 1

```
for (3x) {
    ergebnis * 2,5
}
```

$$Pot(2.5, 3) = Pot(2.5, 2) \cdot 2,5$$

```
pub. st. double Pot (double b, int exp) {
        if(exp == 0) return 1;
        return b · Pot(b, exp - 1);
}
```

$$\begin{aligned} Pot(2.5, 3) &= b \cdot Pot(2.5, 2) \\ &= b \cdot b \cdot Pot(2.5, 1) \\ &= b \cdot b \cdot b \cdot \underbrace{Pot(2.5, 0)} \\ &= 2,5 \cdot 2,5 \cdot 2,5 \cdot 1 \end{aligned}$$

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. 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.

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.

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 ; Semikolon 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 Zeilenende

`/* [...] */` kann über mehrere Zeilen gehen

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 -2^{31} bis $+2^{31}-1$

`float` Dezimalzahlen $-3{,}4 \cdot 10^{38}$ bis $+3{,}4 \cdot 10^{38}$

`double` Dezimalzahlen $-1{,}7 \cdot 10^{308}$ bis $+1{,}7 \cdot 10^{308}$

`boolean` Wahrheitswerte 1/0 bzw true/false

`char` einzelne Zeichen

`String` Zeichenketten

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 2018 zu:

```
int jahr;
jahr = 2018;
```

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:

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

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` Gleichheit von a und b

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

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

`a != b` Ungleich

5.2 Beispiel

Ergänze das Beispiel:

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

`istSchaltjahr` Methodenname

`(int jahr)` „Eingabewert", Parameter

`{ [...] }` *Methodenrumpf* Auszuführende Befehle

`return [...];` Abbruch der Methode mit Ergebnis

7. Schleifen

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

mit einer Schleife kann eine Befehlsfolge mehrfach durchgeführt werden.

1. Schleifenart: for - Schleife / Zählschleife

```
for (int i=0; i<10; i++) {

}
```

Initialisierung vor 1. Durchlauf

Bedingung (vor jedem Durchlauf)

Befehl nach jedem Durchlauf

2. Schleifenart: while - Schleife

```
while (i<10) {

}
```

Bedingung

Aufgabe:

Programmiere ein Programm, das mehrere Dreiecke ausgibt. Das Programm soll als erstes fragen wie viele Dreiecke es ausgeben sollen.

Anschließend soll gefragt werden, wie hoch jedes Dreieck sein soll.

Nachfolgend ein Programm-Ablauf-Beispiel zum besseren Verständnis:

```
Wieviele Dreiecke wollen Sie ausgeben? 2
Wie hoch soll jedes Dreieck sein? 3
*
**
***
*
**
***
```

Programmiere die Ausgabe eines Dreiecks als Methode

```
public static void dreieck(int hoehe) {
    […]
}
```

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:

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

$5! = 4! \cdot 5$

$3! \cdot 4$

$2! \cdot 3$

$1! \cdot 2$

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

$fak(4) = 4 \cdot fak(3)$

$fak(3) = 3 \cdot fak(2)$

$fak(2) = 2 \cdot fak(1)$ 1

$fak(1) = 1$

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

fak(n) → bei n = 1

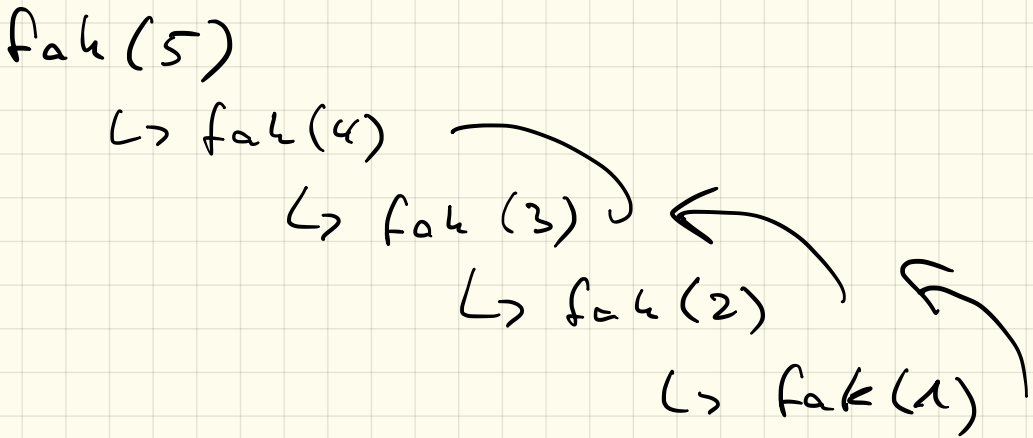
c) Programmiere die Methode `fak` in Java. (*Hinweis: diese soll wieder direkt programmiert werden, ohne ein Objekt anzulegen, also direkt unter die `main`-Methode.*)

d) Wie muss der Methodenkopf in Java aussehen?

Rückgabetyp, Parameter

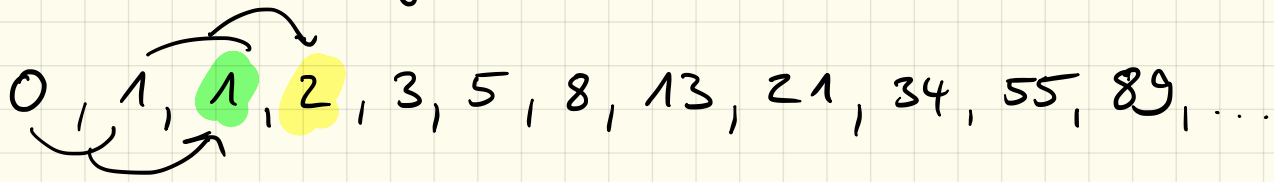
```
public static int fak(int n)
```

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)

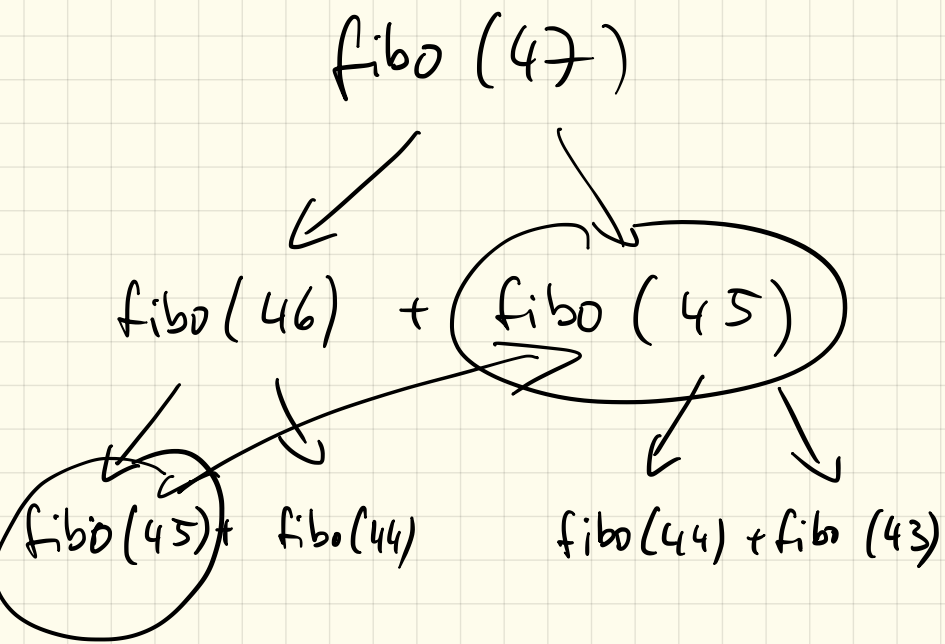

fak(5)
↳ fak(4)
↳ fak(3)
↳ fak(2)
↳ fak(1)

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

    if(n == 1) {
        return 1;
    }
    else {
        return n * fak(n-1);
    }

}
```

Fibonacci-Folge




```
try {
    fibo(-5);
}
catch(Exception e){
    ||
}
public static int fibo(int n) {
    if(n<0) throw new Exception("Fehler");
}
```


fibo (47)
fibo(46) + fibo (45)
fibo(45) + fibo(44)
fibo(44) + fibo (43)

2. Fibonacci-Folge

Die Fibonacci-Folge

$$0,1,1,2,3,5,8,13,21,34,55,\ldots$$

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 $n = 5$:

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

c) Wie muss der Methodenkopf in Java aussehen?

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

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:

$$
\begin{array}{ccccccccccc}
 & & & & & 1 & & & & & \\
 & & & & 1 & & 1 & & & & \\
 & & & 1 & & 2 & & 1 & & & \\
 & & 1 & & 3 & & 3 & & 1 & & \\
 & 1 & & 4 & & 6 & & 4 & & 1 & \\
1 & & 5 & & 10 & & 10 & & 5 & & 1 \\
 & 6 & & 15 & & 20 & & 15 & & 6 & \\
\ddots & & & & \vdots & & & & & & \ddots
\end{array}
$$

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.

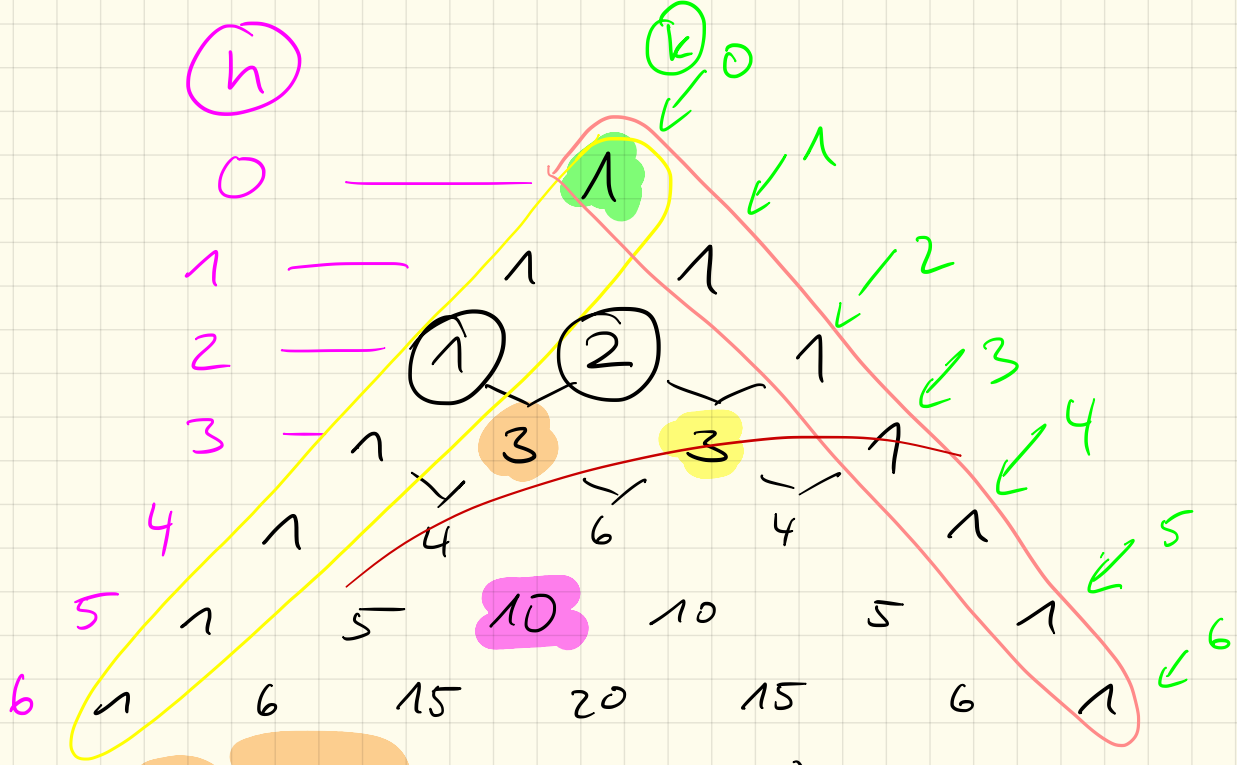
b) Notiere dir die rekursiven Methodenaufrufe und das Aufrufschema für $n = 3$ und $k = 2$:

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

d) Wie muss der Methodenkopf in Java aussehen?

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)



binom(3,1) = binom(2,0) + binom(2,1)

binom(n,k) = binom(n-1,k-1) + binom(n-1,k)

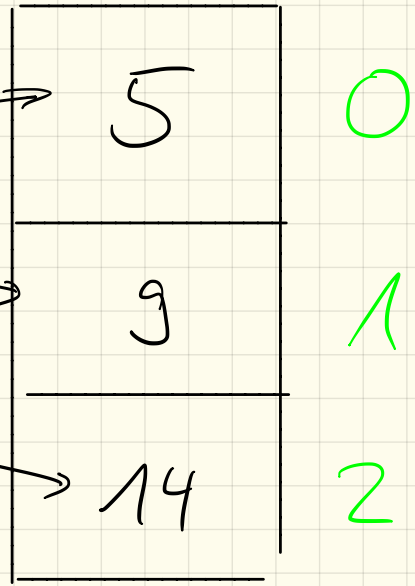
binom(n,0) = 1 / binom(n,n) = 1

19.4.18

Klausur: ~~3.5.~~ 5.7.

Array

```
int zahl1 = 5;
int zahl2 = 9;
int zahl3 = 14;
```



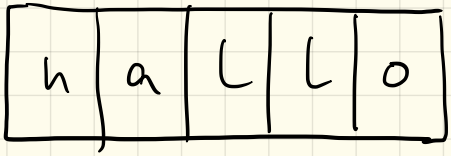

```
int[] zahlen = new int[3];
```

```
zahlen[0] = 5;
zahlen[1] = 9;
```

zahlen[3]

Index Out Of Bound Exception

String test = "hallo";



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.

SelectionSort – Teil 1

1. sortierte Ausgabe

Erstelle in deinem Projekt `Sortierung` ein Paket `selectionsort`. Lege in diesem eine Klasse `Ausgabe` (inklusive `main`-Methode) an.
Implementiere ein Programm, welches:

- Ein Array mit 20 Zufallszahlen (zwischen 0 und 50) füllt.
- Dieses Array soll zunächst unsortiert ausgegeben werden.
- Mithilfe der Minimumsuche sollen wie Werte sortiert auf der Konsole ausgegeben werden.

Hinweis: du darfst natürlich den Code von letztem Mal nutzen und in die neue Klasse kopieren!

Beispielausgabe auf der Konsole:

```
Unsortiert:
20,6,30,34,5,11,0,34,28,12,4,26,11,15,44,28,40,7,20,7
Sortiert:
0
4
5
6
7
7
...
```

2. Abspeicherung

Erstelle im Paket `selectionsort` eine Klasse `OutOfPlace` mit einer `main`-Methode.
Programmiert werden soll ein Programm, welches wie oben ein zufällig befülltes Array generiert. Anstatt die Werte direkt auszugeben, sollen diese nun in einem *neuen* Array gespeichert werden, damit diese Werte später im Programm wieder weiterverwendet werden können.
Gib zur Kontrolle nach der Sortierung mithilfe einer Schleife das sortierte Array auf der Konsole aus.

Beispielausgabe auf der Konsole:

```
Unsortiert:
20,6,30,34,5,11,0,34,28,12,4,26,11,15,44,28,40,7,20,7
Sortiert:
0,4,5,6,7,7,11,11,12,15,20,20,26,28,28,30,34,34,40,44
```

3. Zusatzaufgabe

Informiere dich über die Begriffe *out-of-place* und *in-place*. Was bedeuten diese im Hinblick auf Sortieralgorithmen?
Hinweis: soll hier noch nicht schriftlich festgehalten werden!

7.6.2018

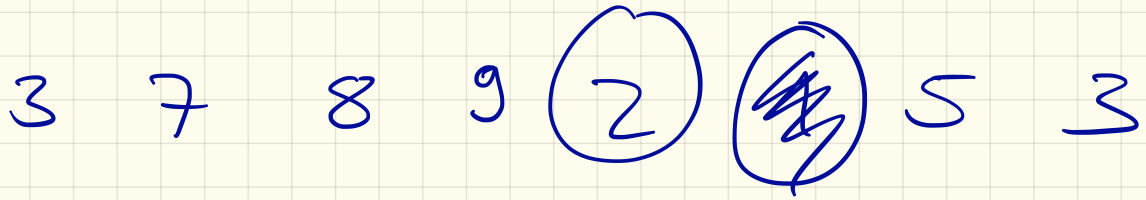
* ~~SelectionSort~~
* InsertionSort
* MergeSort
* BubbleSort
* SwapSort
* HeapSort
* QuickSort
* TimSort
* ...

- Verfahren vorstellen + zeigen
  Vor-/Nachteile
- Geschwindigkeit / Operationen im Vergleich mit anderen Verfahren
- Präsentation (~5 min)
- Handout (A4)

Recherche: 7.6.

Präsentation: 14.6.

Quellen



Anzahl: n

Durchgänge: n

1 2

Insgesamt $n \cdot n$ Operationen

$O(n^2)$

Klausur 5.7.

- Rekursion
- Sortierverfahren
  - SelectionSort
  - InsertionSort
  - Bubble Sort
  - Heap Sort
  - QuickSort
  - ~~Merge Sort~~

- Arrays
  - in-place / out-of-place
  - stabil

QuickSort 5 7 19 3 6 2

```
public static int[] quicksort(int[] unsortiert)
{   if (unsortiert.length <= 1) -> Abbruch
    int pivot = unsortiert[0];
    int[] links = new int[unsortiert.length-1];   int l = 0;
    int[] rechts = new int[unsortiert.length-1];  int r = 0;
    for(int i = 1 ; i<unsortiert.length; i++){
        if (unsortiert[i] < pivot){
            links[l] = unsortiert[i];
            l++;
        }
        else {
            rechts[r] = unsortiert[i];
            r++;
        }
    }
    // links + rechts kürzen
    links = quicksort(links);
    rechts = quicksort(rechts);
                                          // sortiert = links + pivot + rechts
                                      }
```

QuickSort

Das QuickSort-Verfahren basiert darauf, zuerst ein Pivot-Element auszuwählen, anschließend die Liste anhand diesem aufzuteilen und die Teillisten wieder per QuickSort rekursiv zu sortieren. Anschließend werden die Teillisten wieder zusammengesetzt.

```
public static int[] quicksort(int[] unsortiert) {
    // Abbruchbedingung wenn unsortiert.length<=1
    // Pivot-Element auswählen (erstes Element des Arrays)
    // linke und rechte Teilliste anlegen
    // Liste anhand des Pivot-Elements aufteilen
    // linke und rechte Teilliste "kürzen"
    // linke und rechte Teilliste rekursiv sortieren
    // Gesamtliste sortiert zusammenfügen
}
```

Listing 1: Überblick QuickSort

Liste „kürzen“ bzw. zusammenfügen

Ein Array in JAVA hat immer eine konstante Länge. Um ein kürzeres Array zu erhalten, müssen wir dieses neu anlegen und die Elemente kopieren. Hierzu gibt es die Methode

```
System.arraycopy(Quelle, Start-Index, Ziele, Start-Index, Länge);
```

Listing 2: Inhalt eines Arrays kopieren

1. Programmierung

Fülle zunächst oben stehenden Code aus. Programmiere anschließend die Methode.

2. Test

Lasse anschließend ein Array mit 10 (100, 1000,...) Einträgen zufällig befüllen und ausgeben. Rufe dann deine Methode auf und lasse die sortierte Liste wieder ausgeben.

3. Laufzeit

Um die Laufzeit des Verfahrens zu messen, können wir die Methode `System.currentTimeMillis();` verwenden. Diese gibt die vergangenen Millisekunden seit 1. Januar 1970 an. (Datentyp `long`)
Lassen wir diese vor und nach der Sortierung ausgeben können wir daraus die benötigte Zeit berechnen. Erzeuge (nacheinander) verschieden große Arrays und messe damit die Laufzeiten. Vergleiche die verschieden großen Arrays.

Schuljahr 18/19

27.9.2018

Organisation:

1 Klausur pro HJ

(50%+20%) schriftlich + ggf. Projekt
30% mündlich

E-Mail: schule@lehrer-kimmig.de

wiki.lehrer-kimmig.de
ab.lehrer-kimmig.de

GFS noch möglich

Inhalte:

* Netzwerke (bis ca. Herbstferien)
  * Begriffe
  * Adressen
  * Protokolle (HTTP)
* Webprogrammierung (bis ca. Weihnachtsferien)
  * HTML
  * CSS
  * JavaScript
* Datenbanken (bis ca. Fasnet)
* Kryptografie (bis ca. Ostern)

—Abitur—

* objektorientierte Programmierung

Netzwerke: Begriffe und Grundlagen

1. Begriffe – „Netzwerk“

Beschreibe die folgenden Begriffe und gib, sofern möglich, Einsatzzwecke davon an:

- Netzwerk (allgemein):
  Verbund mehrerer Rechner(gruppen) zur Kommunikation um Ressourcen gemeinsam zu verwenden
- LAN:
  (Wireless) Local Area Network
- WAN:
  Wide Area Network
- ARPANET:
  Advanced Research Projects Agency Network
- Internet:
  öffentlich abrufbare Rechner

2. Begriffe – „Netzwerktopologie“

Beschreibe die folgenden Begriffe:

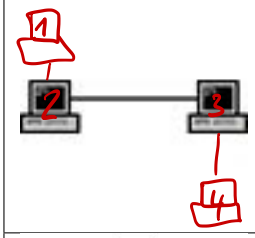
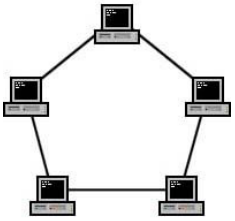
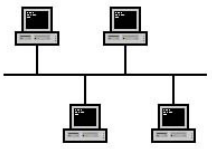
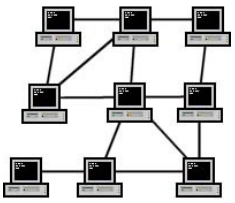
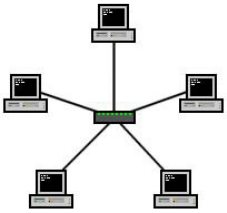
- Netzwerktopologie (allgemein):
  Anordnung / Verkettungsarten
- physische Topologie:
  verkabelte Topologie

- logische Topologie:

wie die Daten im Netzwerk fließen

3. Topologiearten

Benenne die folgenden Topologiearten und gib einige Vor- und Nachteile an:

	P2P / Linie - leicht verständlich - hoher Aufwand - nicht ausfallsicher
	Ring - etwas ausfallsicherer
	Bus - ausfallsicher (relativ) - nur 1 gleichzeitiger Sender - abhörbar
	vermascht - sehr ausfallsicher (Routing) - komplexe Verkabelung
	Stern - ausfallsicher - leicht erweiterbar - alles abhängig von zentralem Knoten

Bildquellen: Wikipedia: Topologie, https://de.wikipedia.org/wiki/Topologie_(Rechnernetz), abgerufen am 26.5.2016

4. Begriffe – Hardware und Software

Beschreibe die folgenden Begriffe und deren Einsatzzweck:

- Hub:

  Verteilzentrum, Daten von mehreren Geräten kommen an und werden an alle anderen Geräte weitergeleitet

- Switch:

  Leitet Datenpakete nur an den richtigen Empfänger weiter

- Modem:

  zur Datenübertragung über eine Nicht-Datenleitung

  MOdulator - Demodulator

- Router:

  verbindet mehrere getrennte Rechnernetze miteinander

- MAC-Adresse:

  Hardware-Adresse, eindeutig weltweit

  physisch

- IP-Adresse:

  Adresse des Endgerätes

  logisch

- Client-Server-Modell:

  Server = Dienstleister

  Client stellt Anfragen

- Peer-to-Peer-Modell (P2P):

  alle Rechner gleichwertig

- Port:

  kennzeichnet verschiedene Dienste auf dem Server

- Domain:

- DNS:

MAC-Adresse 00:11:22:33:44:55

IP-Adresse 10.18.30.179

Server: 10.16.1.1.

10.16.x.y.

10.17.x.y

10.18.x.y

10.10.x.y

10.11.x y

IP-Adr. 192.145.96.201 = 11000000.10010001.01100000.11001001

Subnetz 255.255.255.240 = 11111111.11111111.11111111.11110000

Netzwerkteil 192.145.96.192 = 11000000.10010001.01100000.11000000

Geräteteil 0.0.0.9 = 0.0.0.00001001

kleinste nutzbare Adresse = | Netzwerkteil | 0001

größte nutzbare Adresse = | Netzwerkteil | 1110

Netzwerke: Adressierung

1. IP-Adressen

Zur Kommunitation innerhalb eines Netzwerkes wird jedem Rechner eine eindeutige IP-Adresse zugewiesen. Diese sind in der noch gebräuchlichsten Version (IPv4) in 4 Blöcke mit je 8 Bit aufgeteilt. Es sind also theoretisch die Adressen von 0.0.0.0 bis 255.255.255.255 möglich.
Zum besseren Verständnis ist eine Darstellung im Binärsystem hilfreich.

2. Übung: Umrechnung Dezimalsystem ↔ Binärsystem

Wiederhole die Umrechung zwischen dem Binärsystem und dem Dezimalsystem. Wandle anschließend in das jeweils andere Stellenwertsystem um.

a) $1101\ 1110_2$ 222 c) $1111\ 1101_2$ 253 e) 13_{10} 1101 g) 254_{10} 1111 1110

b) $0011\ 1111_2$ 63 d) $0101\ 1010_2$ 90 f) 96_{10} 1100000 h) 127_{10} 0111 1111

3. Subnetzmaske

Eine *Subnetzmaske* ist in IPv4 ebenfalls eine 32-Bit-Zahl, welche eine IP-Adresse in *Netzwerkteil* und *Geräteteil* trennt.
Die Subnetzmaske gibt quasi an, welche Geräte direkt miteinander kommunizieren können, bzw. welche Geräte in einem *logischen* Netz verbunden sind.
Durch UND-Verknüpfung der IP mit der Subnetzmaske erhält man den Netzwerkteil. Alle Geräte mit dem selben Netzwerkteil gehören zum selben logischen Netzwerk.
Durch UND-Verknüpfung mit der invertierten Subnetzmaske erhält man den Geräteteil.
Mit dem Netzwerkteil ergibt sich dann auch die kleinstmögliche und größtmögliche IP-Adresse. Diese beiden Adressen dürfen nicht an Geräte vergeben werden, sie sind für die *Netzwerkadresse* (identisch zum Netzwerkteil) bzw. die *Broadcastadresse* reserviert.
Beispiel:

IP-Adresse:	192.145.96.201	=	11000000.10010001.01100000.11001001
Subnetzmaske:	255.255.255.~~240~~ 28	=	11111111.11111111.11111111.11110000
Netzwerkteil:	192.145.96.192	=	11000000.10010001.01100000.11000000
invertierte Subnetzmaske:	0.0.0.15	=	00000000.00000000.00000000.00001111
Geräteteil:	0.0.0.9	=	00000000.00000000.00000000.00001001
Netzwerkadresse:	192.145.96.192	=	11000000.10010001.01100000.11000000
Broadcastadresse:	192.145.96.207	=	11000000.10010001.01100000.11001111
kleinste nutzbare Adresse:	192.145.96.193	=	11000000.10010001.01100000.11000001
größte Nutzbare Adresse:	192.145.96.206	=	11000000.10010001.01100000.11001110

Vervollständige die Tabelle:

IP	Subnetzmaske	Netzwerkteil	Geräteteil	Broadcast	1. Adresse	letzte Adresse
192.168.213.15	255.255.255.192	192.168.213.0	0.0.0.15	192.168.213.63	192.168.213.1	192.168.213.62
172.16.5.254	255.255.255.0	172.16.5.0	0.0.0.254	172.16.5.255	172.16.5.1	172.16.5.254
172.254.13.8	255.255.248.0	172.254.8.0	0.0.5.8	172.254.15.255	172.254.8.1	172.254.15.254
10.18.51.5	255.240.0.0	10.16.0.0	0.2.51.5	10.31.255.255	10.16.1.1	10.31.255.254
10.0.0.15	255.0.0.0	10.0.0.0	0.0.0.15	10.255.255.255	10.0.0.1	10.255.255.254

192.168.213.15 = 11000000 . 10101000 . 11010101 . 00001111

255.255.255.192 = 11111111 . 11111111 . 11111111 . 11000000

Netzwerkteil = 11000000 . 10101000 . 11010101 . 00 000000 = 192.168.213.0

Geräteteil: = 0 . 0 . 0 . 00 001111 = 0.0.0.15

Broadcast: = 11000000 . 10101000 . 11010101 . 00 111111 = 192.168.213.63

erste Adresse = Netzwerkteil + 1 = 192.168.213.1

Letzte Adresse = Broadcast − 1 = 192.168.213.62

4. Übungen

a) Eine Nachricht wird im Netzwerk mit der Subnetzmaske 255.255.248.0 von einem Rechner mit der IP 192.168.203.15 an einen Rechner mit der IP 192.168.200.65 geschickt. Bleibt die Nachricht im Netzwerk oder muss sie über das Internet verschickt werden?

b) Eine Nachricht wird im Netzwerk mit der Subnetzmaske 255.240.0.0 von einem Rechner mit der IP 10.32.100.12 an einen Rechner mit der IP 10.16.1.1 geschickt. Bleibt die Nachricht im Netzwerk oder muss sie geroutet werden?

a) Netzwerkteil: 192.168.200.0 → kein Router nötig

b) 10.32.100.12 → Netzwerkteil 10.32.0.0
10.16.1.1 → Netzwerkteil 10.16.0.0
} Router nötig!

Netzwerke: Filius

Filius

Filius ist eine kostenlose Lernsoftware um Den Aufbau von Netzwerken zu simulieren. Sie kann kostenlos unter `www.lernsoftware-filius.de` heruntergeladen werden.

1. Verbindung zwischen zwei Rechnern

Erstelle ein Netzwerk mit zwei verbundenen Computern. Diese sollen die IPs `192.168.0.10` und `192.168.0.11` haben. Durch die Subnetzmaske `255.255.255.0` stellt ihr sicher, dass beide PCs im selben Netz liegen und gegenseitig erreichbar sind.

2. Verbindungstest

„Installiere“ im Aktionsmodus auf dem ersten Rechner die *Befehlszeile* und teste die Verbindung indem du den Befehl
`ping 192.168.0.11`
ausführst. Beobachte dabei den Datenaustausch auf dem ersten Rechner.

3. Erweiterung des Netzes

Erweitere das Netzwerk um einen dritten PC mit der IP `192.168.0.12`
Da der dritte PC nicht direkt an die beiden anderen angeschlossen werden kann musst du einen Switch dazwischen bauen.

4. Echo-Server

Installiere (im Aktionsmodus) auf dem dritten PC einen *Echo-Server* und starte ihn. Dieser nimmt Anfragen vom *Echo-Client* entgegen und antwortet daraufhin.
Installiere dazu auf einem anderen PC den *Echo-Client* und verbinde diesen mit dem Echo-Server indem du dessen IP-Adresse eingibst.
Sende dann Nachrichten vom Echo-Client aus und beobachte dabei den Datenaustausch.

5. Verbindung zweier Netze

5.1 zweites Netz

Erstelle zunächst ein zweites **vorerst unabhängiges** Netz mit 3 PCs mit den IPs `192.168.1.10`, `192.168.1.11` und `192.168.1.12`.

5.2 Vermittlungsrechner

Um die Kommunikation zwischen diesen beiden Netzen zu ermöglichen benötigen wir einen *Vermittlungsrechner* der zwischen den beiden Netzen vermittelt. Dieser Rechner ist Teil von beiden Netzen und hat deshalb auch zwei IP-Adressen: `192.168.0.1` und `192.168.1.1` (eine IP-Adresse pro Netzwerk).
Verbinde die beiden Netze mit einem Vermittlungsrechner und weise diesem die oben stehenden IPs zu. Teste mit einem `ping`-Befehl ob du von einem Rechner aus dem ersten Netz einen PC aus dem zweiten Netz erreichen kannst.

5.3 Gateway

Damit die Rechner aus dem ersten Netz mit den PCs im zweiten Netz kommunizieren können müssen wir die IP des Vermittlingsrechners als *Gateway* eintragen.

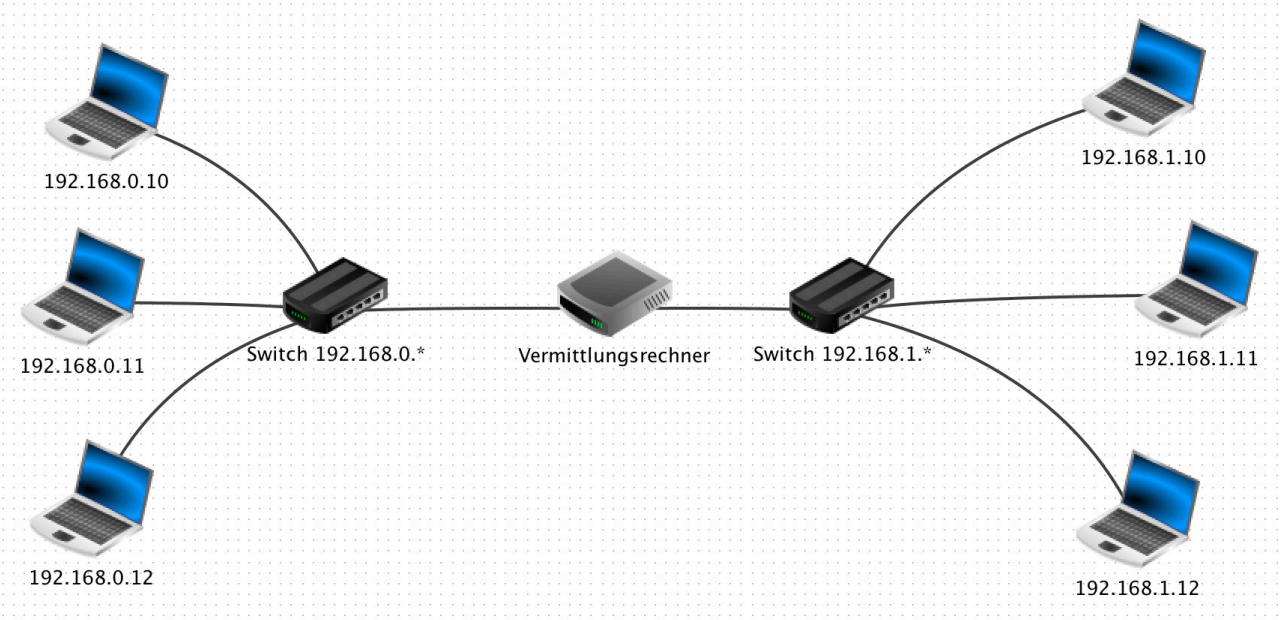
Netzwerke: Filius

Simulation des Word Wide Webs

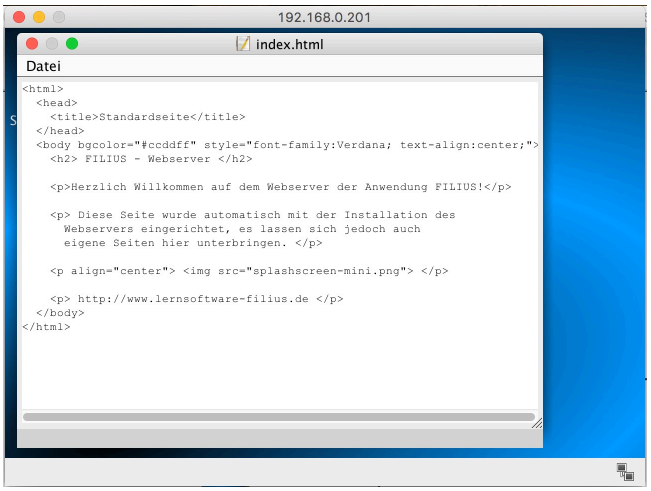
Eine der wichtigsten Aufgaben des heutigen Internets ist zweifelsohne das World Wide Web. Mithilfe von FILIUS kann man die grundlegenden Abläufe bei der Kommunikation zwischen einem Webbrowser und einem irgendwo anders befindlichen Webserver simulieren und analysieren.

1. Webserver

Erstelle zunächst wieder den Aufbau von letztem Mal:



Denke dabei daran, die IP-Adressen und das Gateway richtig einzustellen.
Schließe danach einen Rechner mit der IP `192.168.0.201` an.
Installiere anschließend auf diesem Rechner einen *Webserver*, und starte diesen.



Installiere auf einem (beliebigen) anderen PC einen *Webbrowser* und rufe mit diesem die Adresse `http://192.168.0.201` auf, hier sollte die standardmäßig eingestellt Webseite angezeigt werden.

2. Domain Name System

Informiere dich, was das *Domain Name System* (DNS) ist, wie es funktioniert und welche Voraussetzungen dafür gegeben sein müssen.

2.1 Schwachstellen

Informiere dich über mögliche Schwachstellen und Angriffspunkte des DNS.

3. DNS in Filius

Erweitere deinen Vermittlungsrechner um eine weitere Netzwerkschnittstelle (mit dem Button *Verbindungen verwalten*) und gib dieser die IP-Adresse `192.168.2.1`.
Schließe daran einen weiteren Server an mit der IP `192.168.2.201`.
Installiere darauf einen *DNS-Server* mit einem Adresseintrag z. B. `meinserver.de` auf die IP-Adresse `192.168.0.201`.
Versuche danach mit einem Webbrowser auf die Seite `http://meinserver.de` zuzugreifen. Es funktioniert nicht – warum? Korrigiere die Einstellungen so, dass es funktioniert!
Beobachte anschließend (mit den blinkend dargestellten Kabel), wie die Kommunikation abläuft. *Tipp: Die Geschwindigkeit lässt sich in FILIUS mit dem Schieberegler in der Funktionsleiste oben einstellen!*

3.1 Direkte Abfrage der Ziel-IP

Der DNS-Server kann auch direkt über die *Befehlszeile* abgefragt werden. Hierfür wird der Befehl `host meinserver.de` verwendet.

4. Dynamic Host Configuration Protocol

Speziell in großen Netzwerken wird es sehr aufwändig, alle IP-Adressen, Gateway, DNS-Adressen,... richtig zu konfigurieren bzw. bei einer Änderung aktuell zu halten.
Informiere dich, wie das *Dynamic Host Configuration Protocol* DHCP grundlegend funktioniert.
Richte auf dem `192.168.0.201` (in den Einstellungen im Entwurfsmodus) einen DHCP-Server ein und lasse die IP-Adressen von `192.168.0.10` bis `192.168.0.20` automatisch vergeben.
Damit die automatischen Einstellungen auf den Computern übernommen werden, musst du die manuellen IP-Adressen durch die automatische Konfiguration ersetzen. Wähle dazu die Option *DHCP zur Konfiguration verwenden* aus.
Beobachte, welche Kommunikation stattfindet, wenn du in den Aktionsmodus wechselst.

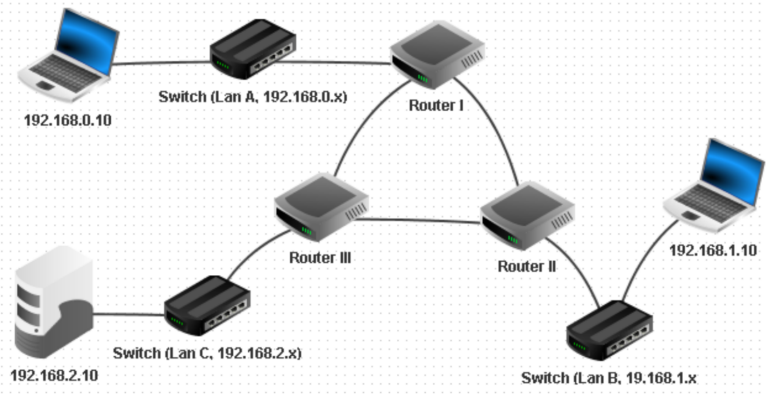
Netzwerke: Filius

Routing

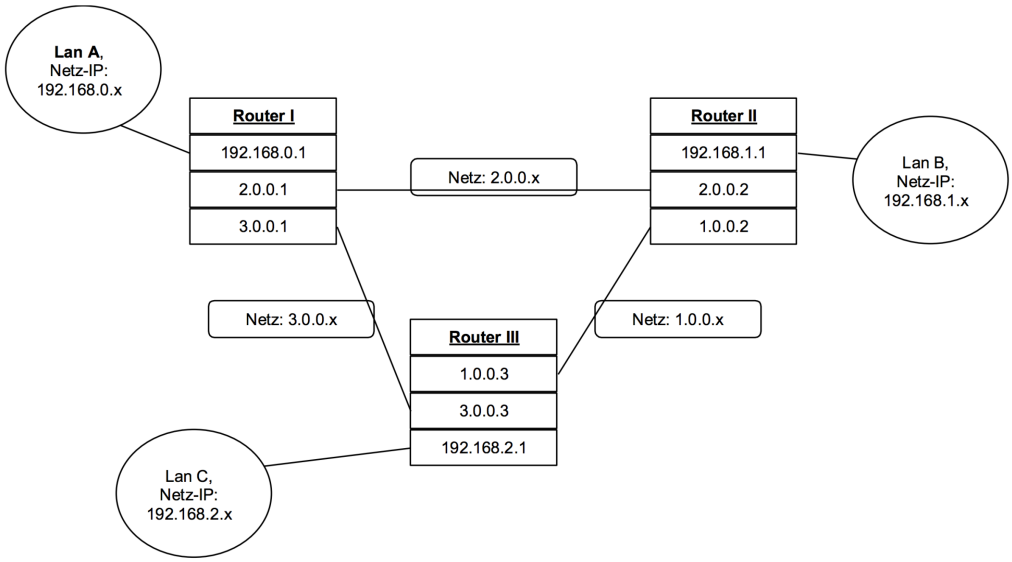
Bisher haben wir uns darauf beschränkt, über einen Vermittlungsrechner zwei Netze miteinander zu verbinden. In der Realität sieht es allerdings so aus, dass zwischen zwei Netzen oft sehr viele Vermittlungsrechner sind die auch oft in Form eines vermaschten Netzes verbunden werden.
Hierbei müssen die Router jeweils wissen, welche Ziel-IP-Adressen wie geroutet werden sollen.

1. Routing

Erstelle zunächstfolgenden Aufbau:



Die Router müssen dabei uber verschiedene Netze mit eigenständigen IP-Adressbereichen verbunden sein. Schematisch ergibt sich folgendes Bild:



Stelle dann folgende Routing-Tabellen in den Vermittlungsrechnern ein (automatisches Routing deaktivieren!)

Router I:

Ziel	Netzmaske	nächstes Gateway	über Schnittstelle
192.168.1.0	255.255.255.0	2.0.0.2	2.0.0.1
192.168.2.0	255.255.255.0	3.0.0.3	3.0.0.1

Router II:

Ziel	Netzmaske	nächstes Gateway	über Schnittstelle
192.168.0.0	255.255.255.0	2.0.0.1	2.0.0.2
192.168.2.0	255.255.255.0	1.0.0.3	1.0.0.2

Router III:

Ziel	Netzmaske	nächstes Gateway	über Schnittstelle
192.168.0.0	255.255.255.0	3.0.0.1	3.0.0.3
192.168.1.0	255.255.255.0	1.0.0.2	1.0.0.3

2. Netzwerkverkehr

Installiere auf dem Server `192.168.2.10` einen *Echo-Server* und auf dem Computer `192.168.0.10` einen *Einfachen Client*. Verbinde den Client mit dem Server und beobachte den Netzwerkverkehr.

3. Netzwerkverkehr II

Ändere die Routingtabellen so ab, dass Nachrichten von LAN A nach LAN C (und umgekehrt) den „Umweg“ über Router II nehmen.

4. Sicherheit

Welche Sicherheitsprobleme könnte es hierbei geben?

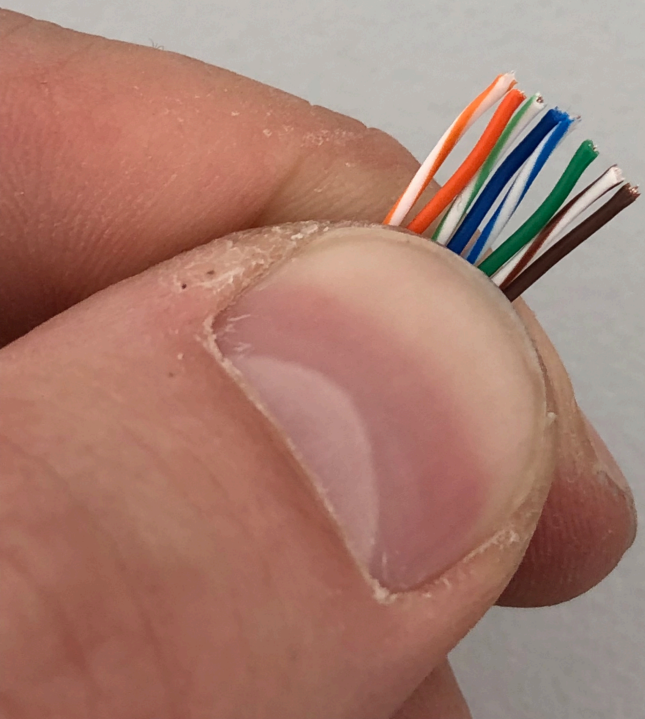
0. Abdeckkappe auf Kabel fädeln
1. Abisolieren

2. Drahtgeflecht nach hinten schlagen

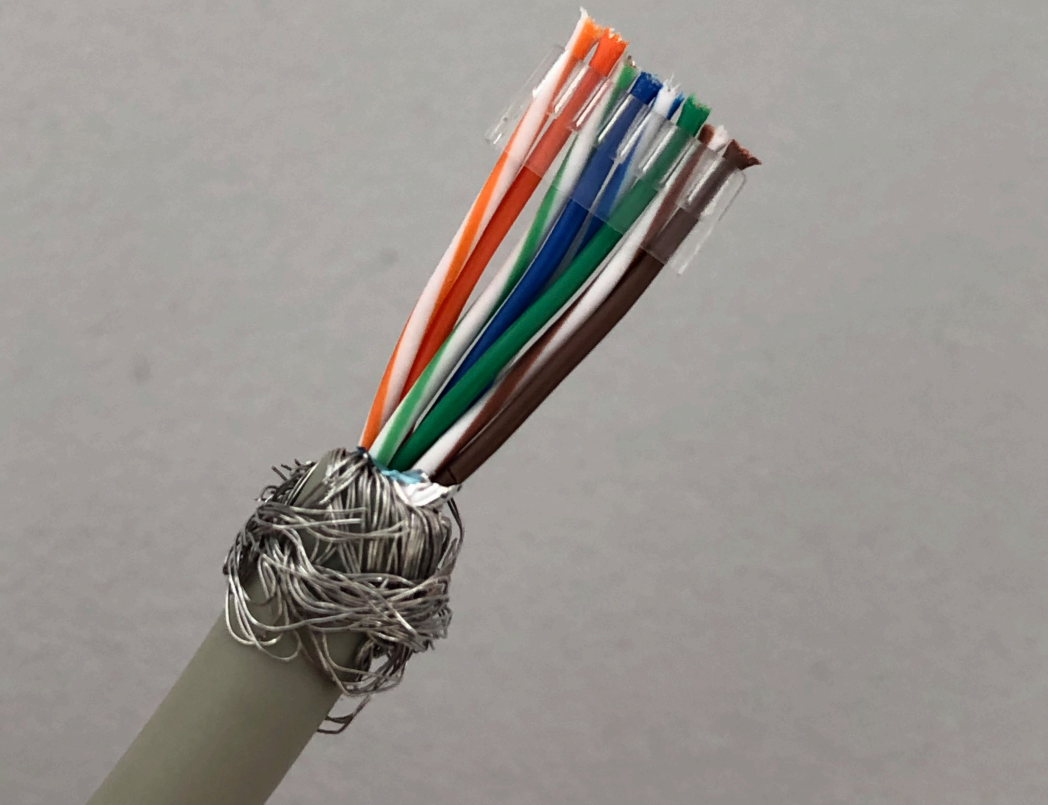
3. Alufolie abschneiden

4. Adern sortieren

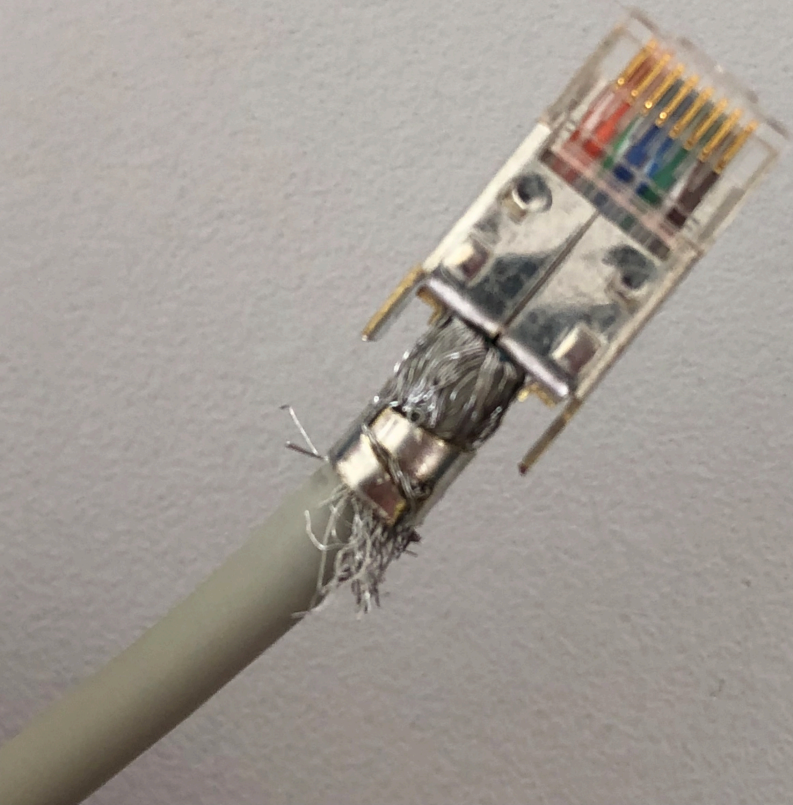
5. Adern in Einschubhilfe fädeln
und Reihenfolge kontrollieren!



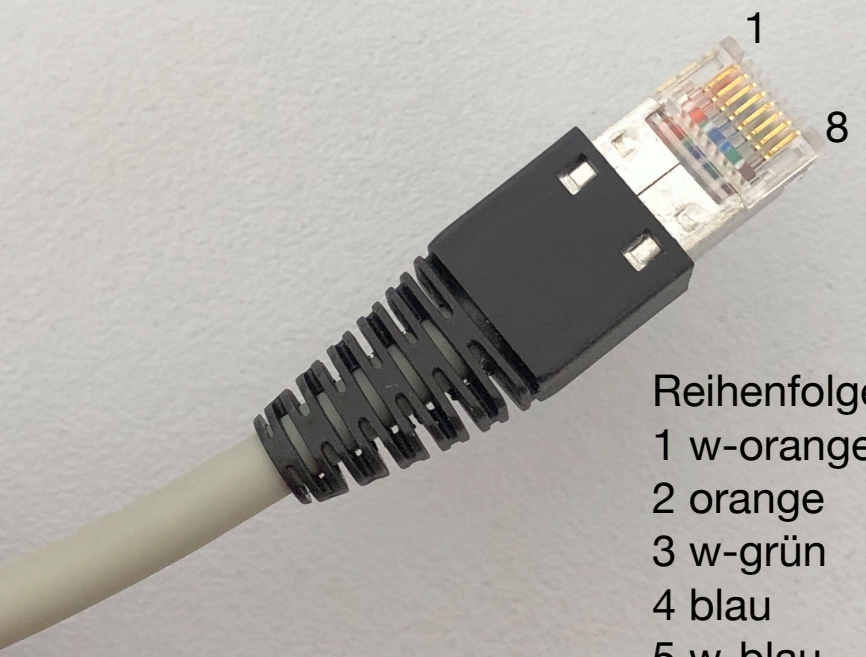
6. Kabel bündig abschneiden
und in Stecker fädeln



7. mit Crimpzange kontaktieren



8. Knickschutzkappe überziehen



Reihenfolge Adern:
1 w-orange
2 orange
3 w-grün
4 blau
5 w-blau
6 grün
7 w-braun
8 braun

8.11.18

HTML

Hyper Text Markup Language

```
<html lang="de">
    <head>

    </head>
    <body>
        Hier steht ein Text.
    </body>
</html>
```

Einführung in HMTL

1. Einführung

HTML bedeutet **H**yper**T**ext **M**arkup **L**anguage und ist eine sogenannte *Auszeichnungssprache*. Mit ihr wird die Struktur (und das Aussehen) einer Webseite bestimmt.
Eine einzelne Webseite besteht dabei aus vielen HTML-Elementen bzw. -Blöcken. Diese werden in sogenannten *Tags* eingeschlossen.
Der HTML-Code wird dabei als reine Textdatei geschrieben und lediglich mit der Dateiendung `.html` abgespeichert.

2. Einführung Tags

Tags dienen dazu, von ihnen eingeschlossene Texte/Inhalte zu gruppieren und zu formatieren. Tags treten dabei meist paarweise auf: ein beginnendes Tag und ein abschließendes Tag.
Ein Tag erkennt man an den eckigen Klammern, beispielsweise `<html>`. Die abschließende Version davon wird mit einem Schrägstrich geschrieben: `</html>`.
Für jedes Tag können auch noch weitere Attribute angegeben werden. Diese werden dann auch im beginnenden Tag innerhalb der spitzen Klammern geschrieben, z. B.: `<html lang="de">`.

3. Grundgerüst einer HTML-Datei

Jede HTML-Datei muss folgendes Grundgerüst enthalten:

```
1  <!DOCTYPE html>
2
3  <html lang="de">
4
5  <head>
6          <title>HTML ist toll</title>
7  </head>
8
9  <body>
10          Hallo Welt!
11 </body>
12
13 </html>
```

Listing 1: Grundgerüst einer HTML-Datei

`<!DOCTYPE html>` (Zeile 1) Die Angabe sorgt dafür, dass eine Seite im Browser korrekt verarbeitet und angezeigt wird.

`<html>...</html>` (Zeile 3-13) Das HTML-Element ist das Grundelement. In diesem Element kann als Attribut noch die Sprache des Textes auf der Seite angegeben werden. Innerhalb dieses `html`-Tags werden dann die weiteren Blöcke geschrieben.

`<head>...</head>` (Zeile 5-7) Im `head`-Element stehen Informationen über die Webseite. Es enthält sozusagen den Vorspann, der allerdings im Browser nicht erscheint. Eines der wichtigsten Elemente im head-Element ist `title`.

`<title>...</title>` (Zeile 6) Das `title`-Element enthält die Bezeichnung der jeweiligen Webseite und wird im Browserfenster angezeigt.

`<body>...</body>` (Zeile 9-11) Nach dem `head`-Element, also dem Kopf, folgt das `body`-Element. Alles, was zwischen dem Start- und End-Tag des `body`-Elements steht, wird später als sichtbarer Inhalt im Browser angezeigt.

4. Aufgabe

- Erstelle in einem Texteditor (z. B. Windows-Editor/Notepad, **nicht Word!**) eine HTML-Datei mit dem Titel *Grundgerüst einer HTML-Datei* und dem Inhalt *Informatik-AG 2017.*
- Speichere die Datei als `index.html` ab. Achte dabei auf die Dateiendung!
- Öffne die Datei in einem Browser.

5. Aufgabe

Zur Formatierung von Text können nachfolgende Tags benutzt werden. Teste diese indem du um den Text in deiner Datei zusätzlich ein Tag einfügst und beschreibe kurz die sichtbare Funktion:

`<b>...</b>`: bold — fett

`<strong>...</strong>`: fett

`<i>...</i>`: italic — kursiv

`<em>...</em>`: emphasize — kursiv

`<u>...</u>`: underline — unterstrichen

`<h1>...</h1>`: headline — 1. Überschrift

`<h2>...</h2>`: 2. —"—

`<h3>...</h3>`: 3. —"—

`<sup>...</sup>`: hochgestellt

`<sub>...</sub>`: tiefgestellt

`<del>...</del>`: durchgestrichen

6. Aufgabe

Versuche, einen Text mit einem Zeilenumbruch anzeigen zu lassen. Wie funktioniert das?

Tabellen in HTML

1. Bestandteile einer Tabelle

Eine Tabelle wird in HTML mit dem Tag `<table>...</table>` eingeschlossen. Jede Zeile wird dabei mit `<tr>...</tr>` umschlossen.
Für jede einzelne Zelle gibt es die Tags `<th>...</th>` (für Kopfzellen) bzw. `<td>...</td>` (für Inhaltszellen).

2. Aufgabe: Tabelle erstellen

Erstelle eine Webseite mit folgender Tabelle:

Augenfarbe	**Braun**	**Blau**	**Grau**	**Grün**	**Summe**
Haarfarbe					
Braun	119	84	54	29	286
Blond	7	94	10	16	127
Schwarz	68	20	15	5	108
Rot	26	17	14	14	71
Summe	220	215	93	64	592

```
<html>
<body>
<table>
  <tr>
    <th> Augenfarbe </th>
    <th> Braun </th>
    <th> Blau </th>
    <th> Grau </th>
    <th> Grün </th>
    <th> Summe </th>
  </tr>
  <tr>
    <th> Haarfarbe </th>
    <td> </td>
    <td> </td>
    <td> </td>
    <td> </td>
    <td> </td>
  </tr>
  <tr>
    <th> Braun </th>
    <td> 119 </td>
    <td> 84 </td>
```

3. Aufgabe: Tabellenattribute

Informiere dich, welche Attribute es für das `<table>`-Tag gibt, teste diese und beschreibe deren Funktion kurz:

width	Breite
height	Höhe
border	Rahmen
cellpadding	Abstand innerhalb zwischen Rahmen und Text
cellspacing	Abstand zwischen zwei Zellen
bgcolor	Hintergrundfarbe

4. Aufgabe: Abwechselnde Zeilenfarben

Zur besseren Übersicht werden die Zeilen einer Tabelle oft mit unterschiedlichen Grautönen hinterlegt. Wie lässt sich das erreichen? Ändere deine Tabelle entsprechend ab.

5. Aufgabe: Formatierte Tabelle erstellen

Erstelle eine Webseite mit folgender Tabelle:

Mannschaft	SP	S	U	N	T	GT	P
VfB Stuttgart	19	12	2	5	33	21	38
Hannover 96	19	10	5	4	33	23	35
Braunschweig	19	10	5	4	31	21	35
Union Berlin	19	9	5	5	28	21	35
Dynamo Dresden	19	8	7	4	27	21	31
SV Sandhausen	19	8	6	5	28	17	30
Heidenheim	19	8	5	6	28	19	29
Würzburger Kickers	19	7	7	5	34	33	28

Füge außerdem noch eine Überschrift **Tabelle der 1. Bundesliga** hinzu und eine kurze Beschreibung *Tabellenstand vom 10.2.2017*

6. Aufgabe: Verknüpfungen

Schön wäre es, wenn man bei einem Klick auf die Mannschaftsnamen direkt auf die Homepage der Mannschaft weitergeleitet würde. Informiere dich, wie man eine Verknüfung (bzw. einen *Link*) erzeugen kann und verknüpfe die Mannschaftsnamen in der ersten Spalte mit der jeweiligen Homepage.
Mit welchem Tag ist das möglich:

Google

Ein Nachteil der Links ist, dass sich die Mannschaftsseite in dem Fenster (bzw. in dem Tab) öffnet, in dem auch unsere Homepage offen ist. Oft wollen wir erreichen, dass bei einer Verknüpfung auf eine externe Seite diese in einem neuen Fenster/Tab geöffnet wird. Wie funktioniert das?

Attribut target="_blank"

Bilder in HTML

1. Bilder einbinden

Um die Tabelle aus letzter Woche optisch etwas ansprechender zu gestalten fügen wir am Anfang jeder Zeile noch eine Spalte ein mit dem Logo des jeweiligen Vereins.
Hierzu musst du zunächst die Logos im gleichen Ordner wie deine HTML-Datei abspeichern. Anschließend kannst du mit dem `<img>`-Tag das Bild einfügen:

```
<img src="dateiname.jpg">
```

Listing 1: Einbinden von Bildern

Neben dem `src`-Attribut, bei welchem man die einzubindende Bilddatei angibt gibt es noch nachfolgende Attribute. Beschreibe deren Funktion:

`align`: Anordnung

`alt`: Alternativer Text, falls Bild nicht geladen werden kann

`border`: Rahmen

`height`: Höhe des Bildes

`width`: Breite des Bildes

2. relative und absolute Ortsangabe

Ein wichtiger Punkt bei der Verlinkung und bei Einbindung von Bildern ist die Ortsangabe. Diese kann entweder *relativ* zur aktuellen HTML-Datei oder *absolut* angegeben werden.
Relative Pfadangaben werden meist eingesetzt, wenn eine andere Datei auf dem selben Computer bzw. Server geöffnet werden soll. Absolute Pfadangaben können hingegen dann eingesetzt werden, wenn eine externe Quelle geöffnet werden soll.
Wie funktionieren die relativen und absoluten Angaben?

Klausur 28.11.18

Netzwerke

- Begriffe
- Adressen / Netze berechnen
- DHCP / DNS

HTML

- Grundstruktur
- Tags + Attribute
- CSS

CSS

Einführung

CSS – Cascading Style Sheets – ist eine Layout- und Formatierungssprache, um strukturelle Inhalte wie HTML zu formatieren. Das Aussehen der semantisch deklarierten Abschnitte im HTML-Dokument wird damit genauer bestimmt, verändert und erweitert. Ziel davon ist es, die Struktur einer Webseite von deren Layout zu trennen.
Mit CSS geben wir sozusagen zentrale **Formatvorlagen** an, die dann einem Tag und dessen Inhalt zugewiesen werden können.
Diese Geben wir im `<head>...</head>` Bereich an:

```
<head>
    ...
    <style>
        /* hier werden die Formate definiert */
    </style>
</head>
```

Listing 1: Formatvorlagen mit CSS

1. Formatierung aller Überschriften `h1`

Formate können beispielweise für bestimmte Tags definiert werden. Hier sollen zunächst alle Überschriften `h1` formatiert werden. Dazu legen wir im `<styles>`-Bereich die Vorlage an:

```
<style>
    h1 {
        font-size: 30px;
    }
</style>
```

Listing 2: Vorlage für Überschriften

Ein Unterschied zur direkten Formatierung der Elemente ist, dass hierbei sämtliche Größenangaben mit einer Einheit angegeben werden müssen. Mögliche Einheiten sind:

`mm`: Millimeter

`cm`: Zentimeter

`px`: Pixel

`pt`: Punkt

`in`: 1 Zoll = 2,54 cm

`em`: 1 em ≙ Breite von "m"

`en`: 1 en ≙ Breite von "n"

`%`: in Relation zur nächst größeren Box

Für die Formatierung von Text können (unter anderem) folgende Eigenschaften benutzt werden:

font-size: Schriftgröße

font-family: Schriftart

font-variant: Kursiv

font-style: [illegible]

font-weight: Schriftdicke

line-height: Zeilenhöhe

color: Textfarbe

background-color: Hintergrundfarbe

letter-spacing: Zeichenabstand

text-decoration: Verzierung (unterstrichen,...)

text-shadow: Schatten

text-align: Position/Ausrichtung

Diese Eigenschaften sind zumeist selbsterklärend (bei ausreichenden Englischkenntnissen). Überlege dir zunächst, was diese Eigenschaften bewirken könnten und probiere diese anschließend aus.

2. Formatierung von Klassen

Neben der Formatierung aller h1-Tags können wir auch Formatklassen anlegen und diese dann nur bestimmten Elementen zuordnen. Formatklassen beginnen mit einem Punkt gefolgt von einem (fast) beliebigen Namen:

```
<style>
    .farbig {
        color: red;
    }
</style>
```

Listing 3: Formatklasse „farbig“

Diese Klasse muss allerdings dann noch einem Element zugewiesen werden:

```
<h1 class="farbig">Das ist eine farbige Überschrift</h1>
```

Listing 4: Formatklasse „farbig“ einem h1-Element zuweisen

Diese Klasse können wir natürlich auch mehreren Elementen zuordnen, mit dem Vorteil, dass man Änderungen dann nurnoch an einer einzigen Stelle anpassen muss.
Aufgabe: lege eine Tabelle an (oder nimm die Tabelle von letztem Mal ohne direkte Formatierungen) und erzeuge damit abwechselnd eingefärbte Zeilen. Erstelle dazu eine Formatierungsklasse mit einer Hintergrundfarbe und weise diese Klasse dann jeder zweiten Zeile zu.

Weitere CSS-Befehle

1. Hover-Effekt

Um beispielsweise Verknüpfungen besser hervorzuheben werden diese beim Überfahren mit der Maus hervorgehoben. Dieser Effekt nennt man auch **Hover**-Effekt.
Wir können mithilfe von CSS auch Formate für (fast) alle Elemente festlegen, die aktiv werden, sobald der Besucher mit der Maus darüber fährt. Hierzu schreiben wir nach dem Elementnamen `:hover` wie beispielsweise:

```
a {
    /* normale Formatierung für alle Verknüpfungen */
    color: black;
}

a:hover {
    /* Formate sobald sich die Maus über dem Link befindet */
    color: red;
}
```

Listing 1: Hover-Effekt mit CSS

Anmerkung: das funktioniert nicht nur mit Verknüpfungen, sondern beispielsweise auch mit Tabellenzeilen `tr` oder -zellen `td`.

2. abwechselnde Zeilenfarbe

Wir wissen bereits, dass wir den Zeilen unterschiedliche CSS-Klassen zuweisen können um so eine abwechselnde Zeilenfarbe zu erreichen. Was aber passiert, wenn wir eine zusätzliche Zeile irgendwo dazwischen einfügen wollen? Hier müssten wir die Klassen aller nachfolgenden Zeilen anpassen.
CSS kennt seit einiger Zeit glücklicherweise auch Befehle, um so eine abwechselnde Formatierung automatisch zuzuweisen:

```
tr {
    /* normale Formatierung für alle Zeilen */
    background-color: yellow;
}

tr:nth-child(even) {
    /* Formate für gerade Zeilen */
    background-color: red;
}
```

Listing 2: Abwechselnde Zeilenfarbe mit CSS

Hierbei sehen wir auch eine weitere Eigenschaft von CSS: wir können zuerst allen Zeilen eine Formatierung zuweisen und nachträglich diese Formate für bestimmte Zeilen (hier alle geraden Zeilen) wieder überschreiben.
Statt `even` können wir auch `odd` für alle ungeraden Zeilen oder auch eine Zahl, wenn wir nur eine bestimmte Zeile formatieren wollen. Mit `tr:nth-child(1)` könnten wir so beispielsweise die Formate für die Titelzeile festlegen.

3. Aufgabe

Erstelle folgende Webseite:

Tabelle der 1. Bundesliga

Stand: 15. 11. 2018

	Mannschaft	Spiele	Tore	Gegentore	Punkte
	Dortmund	11	33	12	27
	Gladbach	11	26	13	23
	Leipzig	11	22	9	22
	Frankfurt	11	26	13	20

Hierbei sollen eine Zeilen beim Überfahren mit der Maus eine andere Hintergrundfarbe bekommen. Außerdem sollen die Links beim Überfahren mit der Maus ebenfalls die Textfarbe ändern.

4. Weitere Befehle und Techniken für Webseiten

Informiert euch über eines der folgenden Themen und erstellt eine Beispielseite, mit der ihr die Technik erklärt. (Die Seite soll eine Anleitung und ein Beispiel dazu enthalten!)

1. **JavaScript:** Erstellen von interaktiven Webseiten (Grundlagen)
2. **Farben:** Wie werden Farben im Computer dargestellt (RGB- und CMYK-Modell), wie können eigene Farbwerte auf einer Homepage verwendet werden?
3. **Multimedia:** Einbinden von Audio- und Videodateien (nicht nur von YouTube, sondern auch einzelne Dateien!)
4. **Formulare:** Erstellen eines Formulars, verschiedene Formularelemente (ohne Interaktion)
5. **Handyoptimierung:** Mithilfe von CSS lassen sich auch unterschiedliche Formate für verschiedene Displaygrößen angeben, und so für Mobilgeräte optimierte Webseiten erstellen. (Tipp: Mit Firefox lassen sich verschiedene Bildschirmgrößen testen. Dazu im Menü Extras → Web-Entwickler die entsprechende Funktion auswählen)