

Tafeln und Übersichten für das Fach Informatik in der gymnasialen Oberstufe

Mecklenburg-Vorpommern

Version 2029/2030



CC BY SA 4.0

Aufgabenkommission Informatik MV

Dieses Dokument liegt als Entwurfsfassung vor und kann Fehler enthalten. Über Inhalt und Umfang wurde noch keine endgültige Entscheidung getroffen.

Ein Einsatz im Rahmen von Leistungsüberprüfungen ist aktuell ausgeschlossen.

Rückmeldungen bitte an t.hempel_02@iq.bm.mv-regierung.de

1	Daten	3
1.1	Einheiten und Präfixe	3
1.2	Datentypen	3
1.3	Zahlsysteme	4
1.4	Datencodierung	5
1.5	Datenschutz und Datensicherheit	8
2	Diagramme	9
2.1	Ablaufdiagramm	9
2.2	Entity-Relationship-Diagramm	9
2.3	Klassendiagramm	10
2.4	Schaltplan	10
2.5	Sequenzdiagramm	11
2.6	Struktogramm	12
2.7	Netzstrukturdiagramm	13
2.8	Syntaxdiagramm	13
2.9	Zustandsübergangsdiagramm	13
3	Datenbanksysteme	14
3.1	Entity-Relationship-Modell	14
3.2	Relationales Datenbankmodell	14
3.3	Datenbanksprache SQL	16
3.4	SQLite	18
4	Rechnerarchitektur	19
4.1	Von-Neumann- und Harvard-Architektur	19
4.2	Maschinennahe Programmierung	20
4.3	Logikgatter, Logikfunktionen und Schaltalgebra	21
5	Vernetzte Systeme	23
5.1	Schichtenmodell und Protokolle	23
5.2	Netzstrukturen und Kommunikationsabläufe	23
5.3	Sichere Kommunikation	23
6	Sprachen und Automaten	25
6.1	Automatenmodelle	25
6.2	Formale Sprachen und formale Grammatiken	26
7	Softwareentwicklung	27
7.1	Algorithmen	27
7.2	Objektorientierte Modellierung	29
8	Programmierhilfen für Java und Python	30
8.1	Algorithmische Strukturen	30
8.2	Kommentare	31
8.3	Datentypen und Variablen	31
8.4	Datentypenumwandlung	32
8.5	Ein- und Ausgaben	32
8.6	Operatoren	33
8.7	Mathematische Konstanten, Funktionen und Zufallszahlen	34
8.8	Zeichenketten	35
8.9	Listen	36
8.10	Objektorientierte Programmierung	37
9	Aufgabenoperatoren im Fach Informatik	38

1 Daten

1.1 Einheiten und Präfixe

Bit	kleinste Informationseinheit kleinster Speicherplatz hat genau einen von zwei logischen Werten typischerweise 0 oder 1
Byte	8 Bit

Binärpräfix		
Name	Präfix	Wert
Kibi	Ki	$2^{10} = 1024$
Mebi	Mi	$2^{20} = 1048576$
Gibi	Gi	2^{30}
Tebi	Ti	2^{40}
Pebi	Pi	2^{50}
Exbi	Ei	2^{60}

SI-Präfix ähnlicher Größe		
Name	Präfix	Wert
Kilo	k	$10^3 = 1000$
Mega	M	$10^6 = 1000000$
Giga	G	10^9
Tera	T	10^{12}
Peta	P	10^{15}
Exa	E	10^{18}

Beispiel: 298 MiByte = $298 \cdot 2^{20}$ Byte = 312475648 Byte $\approx$ 312 MByte.

Hinweis: Im alltäglichen Umgang werden SI- und Binärpräfix fälschlicherweise oft synonym verwendet.

1.2 Datentypen

	Größe (typisch) Wertebereich (typisch)	Beispiel	Operationen (Auszug)	Hinweis
Wahrheitswert	1 Bit WAHR, FALSCH	true false	Negation logisches ODER	
Ganzzahl	32 Bit $-2^{31} \dots 2^{31}-1$	-142 0 2098382	Addition Subtraktion Multiplikation ganzzahlige Division	↗ Seite 5
Gleitkommazahl	64 Bit mit – Vorzeichen: 1 Bit – Mantisse: 52 Bit – Exponent: 11 Bit	-0.25 0.0 98.775	Addition Subtraktion Multiplikation Division	↗ Seite 6
Zeichen	je nach Codierung beispielsweise 1 bis 4 Byte bei UTF-8	'a' '\u20AC'	Vorgänger Nachfolger	↗ Seite 7
Zeichenkette	automatisch angepasst	"Schule"	Verketten	

1.3 Zahlssysteme

Zahlen im Dezimalsystem							
Basis	10						
Ziffern	0, 1, 2, 3, 4, 5, 6, 7, 8, 9						
Stellentafel	10^7	10^6	10^5	10^4	10^3	10^2	10^1
	0	0	0	9	2	0	8
	$92081_{[10]} = 0 \cdot 10^7 + 0 \cdot 10^6 + 0 \cdot 10^5 + 9 \cdot 10^4 + 2 \cdot 10^3 + 0 \cdot 10^2 + 8 \cdot 10^1 + 1 \cdot 10^0$ $= 90000 + 2000 + 80 + 1$						

Zahlen im Binärsystem							
Basis	2						
Ziffern	0, 1						
Stellentafel	2^7	2^6	2^5	2^4	2^3	2^2	2^1
	1	0	0	1	1	0	1
	$10011011_{[2]} = 1 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$ $= 128 + 16 + 8 + 2 + 1$ $= 155$						
Additionsregeln	$0 + 0 = 0$; $0 + 1 = 1$; $1 + 0 = 1$; $1 + 1 = 0$ Übertrag 1						
Multiplikationsregeln	$0 \cdot 0 = 0$; $0 \cdot 1 = 0$; $1 \cdot 0 = 0$; $1 \cdot 1 = 1$						

Zahlen im Hexadezimalsystem							
Basis	16						
Ziffern	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F						
Stellentafel	16^7	16^6	16^5	16^4	16^3	16^2	16^1
	0	0	0	1	F	0	A
	$1F0A2_{[16]} = 1 \cdot 16^4 + 15 \cdot 16^3 + 0 \cdot 16^2 + 10 \cdot 16^1 + 2 \cdot 16^0$ $= 65536 + 61440 + 160 + 2$ $= 127138$						
Hexadezimalziffer und binäre Darstellung	Ziffer	2^3	2^2	2^1	2^0	Ziffer	2^3
	0	0	0	0	0	8	1
	1	0	0	0	1	9	1
	2	0	0	1	0	A	1
	3	0	0	1	1	B	1
	4	0	1	0	0	C	1
	5	0	1	0	1	D	1
	6	0	1	1	0	E	1
	7	0	1	1	1	F	1

1.4 Datencodierung

Codierung vorzeichenloser Ganzzahlen mit n Bit	
Struktur	Binärdarstellung
Wertebereich	$0 \dots 2^n - 1$
Beispiel	$n = 8 \rightarrow$ Wertebereich $0 \dots 255$ Umwandlung von $187_{[10]}$ in die binäre Darstellung mit 8 Bit: $ \begin{aligned} 187 : 2 &= 93 \text{ Rest } 1 \\ 93 : 2 &= 46 \text{ Rest } 1 \\ 46 : 2 &= 23 \text{ Rest } 0 \\ 23 : 2 &= 11 \text{ Rest } 1 \\ 11 : 2 &= 5 \text{ Rest } 1 \\ 5 : 2 &= 2 \text{ Rest } 1 \\ 2 : 2 &= 1 \text{ Rest } 0 \\ 1 : 2 &= 0 \text{ Rest } 1 \end{aligned} $ Ergebnis: $187_{[10]} = 10111011_{[2]}$ Umwandlung der Ganzzahl $11001011_{[2]}$ in die dezimale Darstellung: $ \begin{aligned} 11001011_{[2]} &= 1 \cdot 2^7 + 1 \cdot 2^6 + 0 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 \\ &= 128 + 64 + 8 + 2 + 1 \\ &= 203_{[10]} \end{aligned} $

Codierung vorzeichenbehafteter Ganzzahlen mit n Bit mithilfe des Zweierkomplements	
Struktur	Binärdarstellung mit Negation des Werts der höchstwertigen Stelle
Wertebereich	$-2^{n-1} \dots 2^{n-1} - 1$
Beispiel	$n = 8 \rightarrow$ Wertebereich $-128 \dots 127$ Umwandlung von $-35_{[10]}$ in die binäre Darstellung: (1) Bildung von $-35_{[10]} = 35_{[10]}$ (2) Umwandlung von $35_{[10]}$ in binäre Darstellung mit 8 Bit $\begin{aligned} 35 : 2 &= 17 \text{ Rest } 1 \\ 17 : 2 &= 8 \text{ Rest } 1 \\ 8 : 2 &= 4 \text{ Rest } 0 \\ 2 : 2 &= 1 \text{ Rest } 0 \\ 1 : 2 &= 0 \text{ Rest } 1 \end{aligned}$ Ergebnis: $35_{[10]} = 00010011_{[2]}$ (3) Bildung des Zweierkomplements von (2) durch bitweises Invertieren und anschließendes Inkrementieren: $00010011_{[2]} \rightarrow 11101101_{[2]}$ (4) Ergebnis: $11101101_{[2]} = -35_{[10]}$ Umwandlung von $11001011_{[2]}$ in die dezimale Darstellung: $ \begin{aligned} 11001011_{[2]} &= -1 \cdot 2^7 + 1 \cdot 2^6 + 0 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 \\ &= -128 + 64 + 8 + 2 + 1 \\ &= -53_{[10]} \end{aligned} $
Bildung eines Zweierkomplements	(1) Bitweises Invertieren der Binärdarstellung der Ausgangszahl (2) Bildung des Nachfolgers durch Addition von 1 (Inkrement)

Codierung vorzeichenloser Festkommazahlen mit n Bit	
Struktur	Binärdarstellung der Form $b_{k-1} \cdot 2^{k-1} + \dots + b_0 \cdot 2^0 + b_{-1} \cdot 2^{-1} \dots + b_{-m} \cdot 2^{-m}$ mit k Bit für Vorkommateil, m Bit für Nachkommateil und $n = k + m$
Wertebereich	$0 \dots 2^k - 2^{-m}$ mit der Schrittweite 2^{-m}
Beispiel	$n = 8, k = 5, m = 3 \rightarrow$ Wertebereich $0 \dots 31,875$ mit Schrittweite $0,125$ Umwandlung von $2,125_{[10]}$ in die binäre Darstellung: (1) Umwandlung des ganzen Teils der Dezimalzahl in die binäre Darstellung mit 5 Bit Teilergebnis: $2_{[10]} \rightarrow 00010_{[2]}$ (2) Umwandlung des Nachkommateils der Dezimalzahl in die binäre Darstellung mit 3 Bit $0,125 \cdot 2 = 0,25 \rightarrow$ Ganzzahl 0 $0,25 \cdot 2 = 0,50 \rightarrow$ Ganzzahl 0 $0,50 \cdot 2 = 1,00 \rightarrow$ Ganzzahl 1 Teilergebnis: $0,125_{[10]} \rightarrow 001_{[2]}$ (3) Zusammensetzen zum Ergebnis: $2,125_{[10]} = 00010001_{[2]}$ Umwandlung von $11001011_{[2]}$ in die dezimale Darstellung: $11001011_{[2]} = 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2} + 1 \cdot 2^{-3}$ $= 16 + 8 + 1 + 0,25 + 0,125$ $= 25,375_{[10]}$
Hinweise	Festkommazahlen ermöglichen ausschließlich diskrete Werte. Vorzeichenbehaftete Festkommazahlen mittels Zweierkomplement

Codierung einzelner Zeichen mit n Bit

ASCII	Zuordnung Zeichen ↔ 7 Bit 128 Zeichen, davon 32 Steuerzeichen
Unicode	Zuordnung Zeichen ↔ Codepunkt in der Form U+0000 ... U+10FFFF Verfahren zur Kodierung der Codepunkte: – UTF-8: Codierung als Folge von einem bis vier Bytes – UTF-16: Codierung als Folge von ein oder zwei 16 Bit-Einheiten

Codierung von Buchstaben

Zeichen	ASCII	Unicode	Zeichen	ASCII	Unicode	Zeichen	ASCII	Unicode
A	65	U+0041	U	85	U+0055	k	107	U+006B
B	66	U+0042	V	86	U+0056	l	108	U+006C
C	67	U+0043	W	87	U+0057	m	109	U+006D
D	68	U+0044	X	88	U+0058	n	110	U+006E
E	69	U+0045	Y	89	U+0059	o	111	U+006F
F	70	U+0046	Z	90	U+005A	p	112	U+0070
G	71	U+0047	Ä		U+00C4	q	113	U+0071
H	72	U+0048	Ö		U+00D6	r	114	U+0072
I	73	U+0049	Ü		U+00DC	s	115	U+0073
J	74	U+004A	ß		U+00E7	t	116	U+0074
K	75	U+004B	a	97	U+0061	u	117	U+0075
L	76	U+004C	b	98	U+0062	v	118	U+0076
M	77	U+004D	c	99	U+0063	w	119	U+0077
N	78	U+004E	d	100	U+0064	x	120	U+0078
O	79	U+004F	e	101	U+0065	y	121	U+0079
P	80	U+0050	f	102	U+0066	z	122	U+007A
Q	81	U+0051	g	103	U+0067	ä		U+00E4
R	82	U+0052	h	104	U+0068	ö		U+00F6
S	83	U+0053	i	105	U+0069	ü		U+00FC
T	84	U+0054	j	106	U+006A	ß		U+00DF

Codierung von Sonderzeichen, Satzzeichen, Ziffern

Zeichen	ASCII	Unicode	Zeichen	ASCII	Unicode	Zeichen	ASCII	Unicode
	32	U+0020	0	48	U+0030	@	64	U+0040
!	33	U+0021	1	49	U+0031	[	91	U+005B
"	34	U+0022	2	50	U+0032	\	92	U+005C
#	35	U+0023	3	51	U+0033	]	93	U+005D
\$	36	U+0024	4	52	U+0034	^	94	U+005E
%	37	U+0025	5	53	U+0035	_	95	U+005F
&	38	U+0026	6	54	U+0036	`	96	U+0060
'	39	U+0027	7	55	U+0037	{	123	U+007B
(	40	U+0028	8	56	U+0038		124	U+007C
)	41	U+0029	9	57	U+0039	}	125	U+007D
*	42	U+002A	:	58	U+003A	~	126	U+007E
+	43	U+002B	;	59	U+003B	§		U+00A7
,	44	U+002C	<	60	U+003C	°		U+00B0
-	45	U+002D	=	61	U+003D	€		U+20AC
.	46	U+002E	>	62	U+003E	≠		U+2260
/	47	U+002F	?	63	U+003F	—		U+26D4

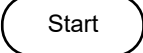
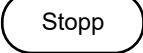
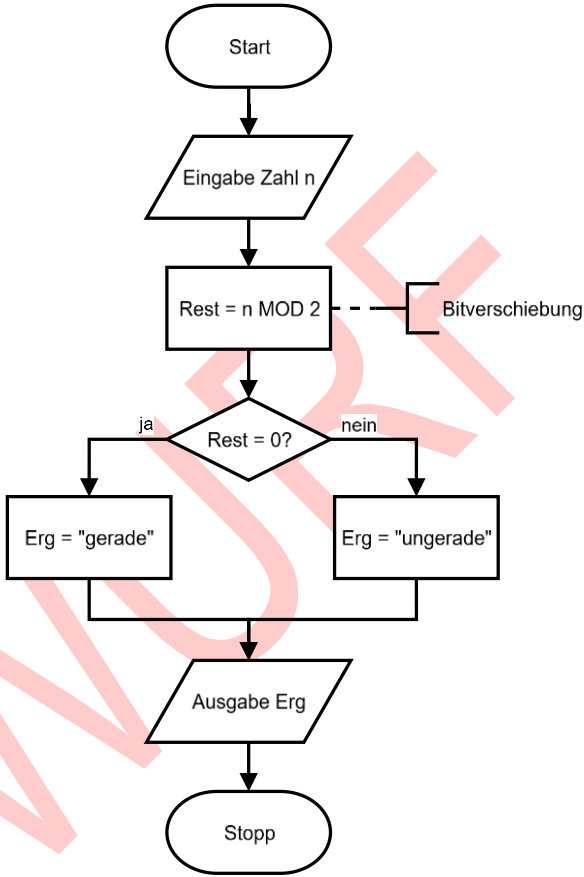
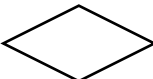
1.5 Datenschutz und Datensicherheit

Datenschutz	Rechtliche Regelungen zur Verarbeitung personenbezogener Daten mit dem Ziel der Gewährleistung von Privatsphäre und informationeller Selbstbestimmung Prinzipien: Zweckbindung, Datenminimierung, Transparenz, Datensicherheit, Betroffenenrechte
Datensicherheit	Technische Maßnahmen zum Schutz von Daten zur Sicherstellung von Verfügbarkeit, Integrität und Vertraulichkeit

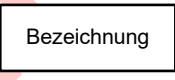
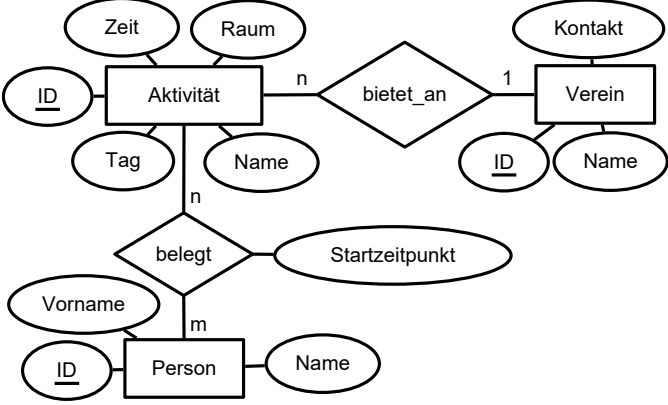
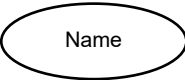
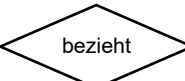
ENTWURF

2 Diagramme

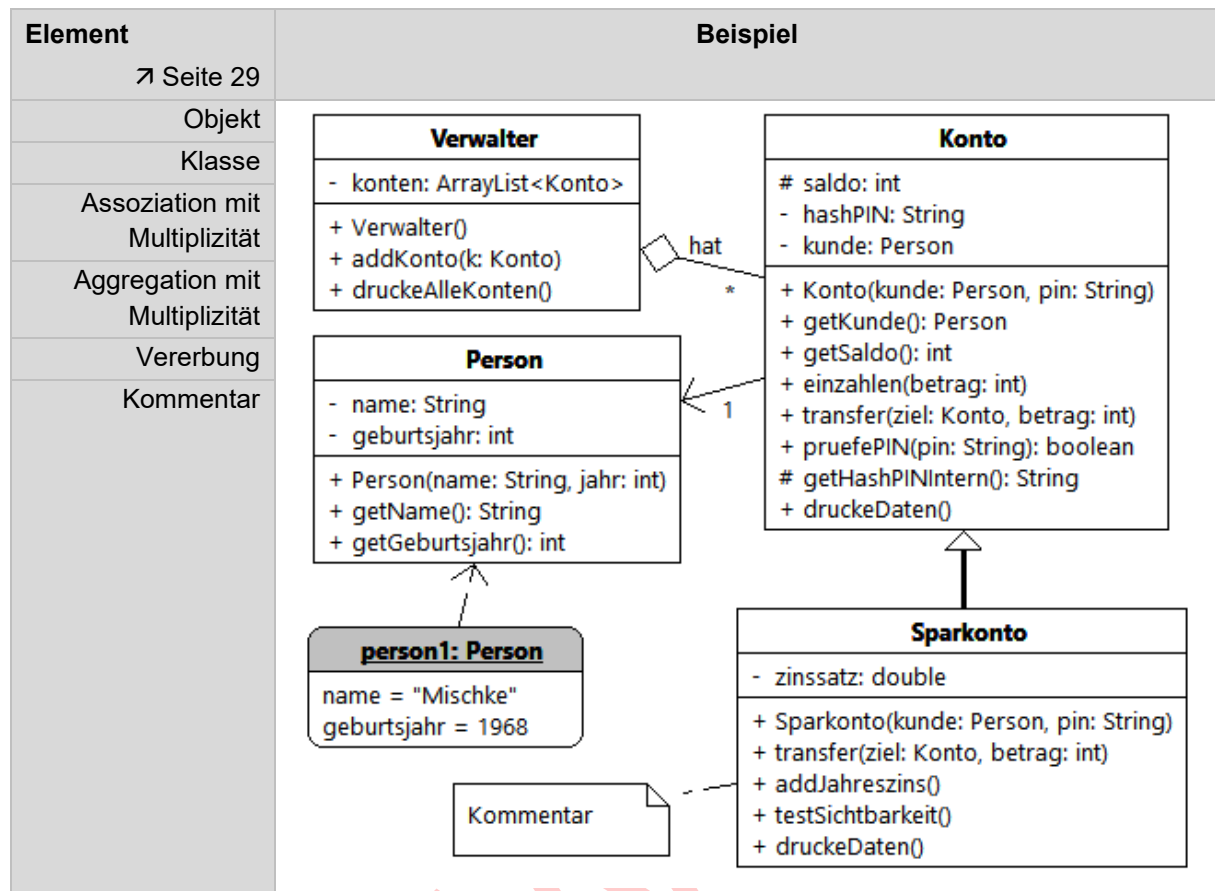
2.1 Ablaufdiagramm

Element	Symbol	Beispiel
Terminator	 	
Operation		
Unterprogramm		
Verzweigung Entscheidung		
Eingabe/Ausgabe		
Ablaufrichtung		
Kommentar		

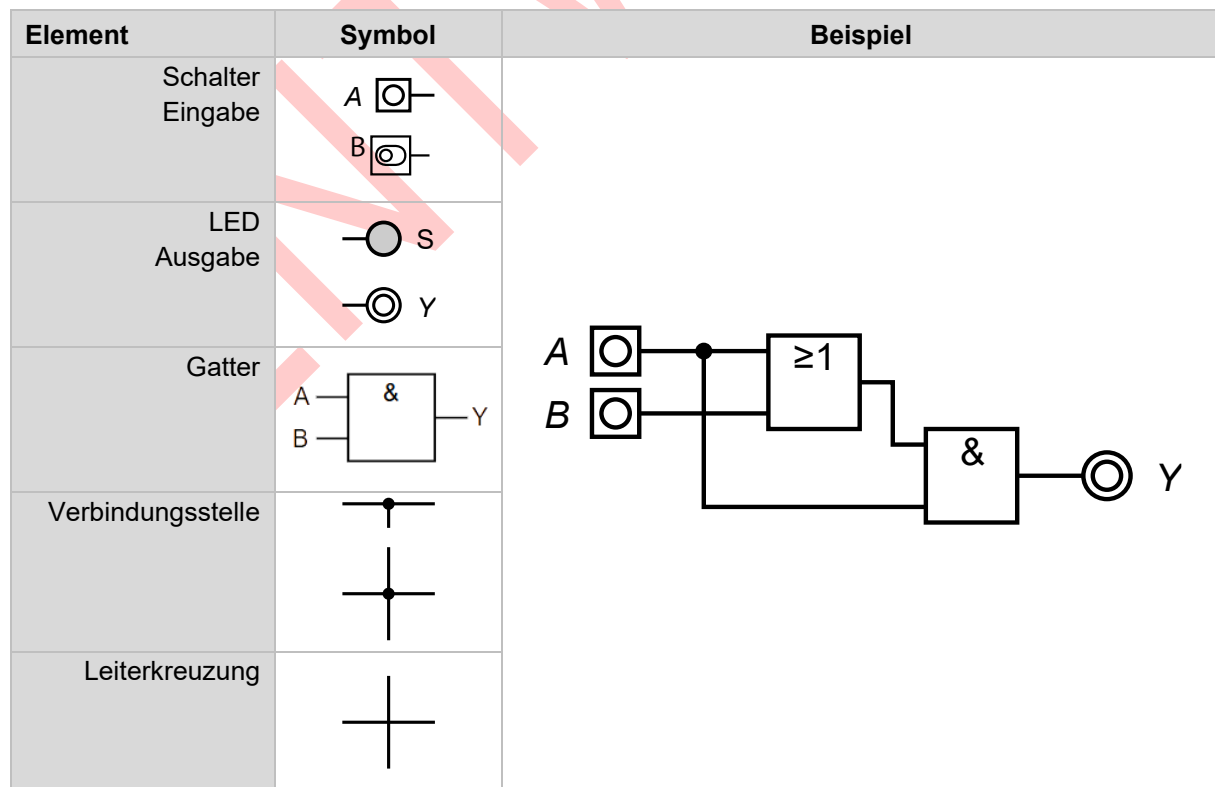
2.2 Entity-Relationship-Diagramm

Element	Symbol	Beispiel
Entitätstyp		
Attribut		
Schlüsselattribut		
Beziehungstyp		

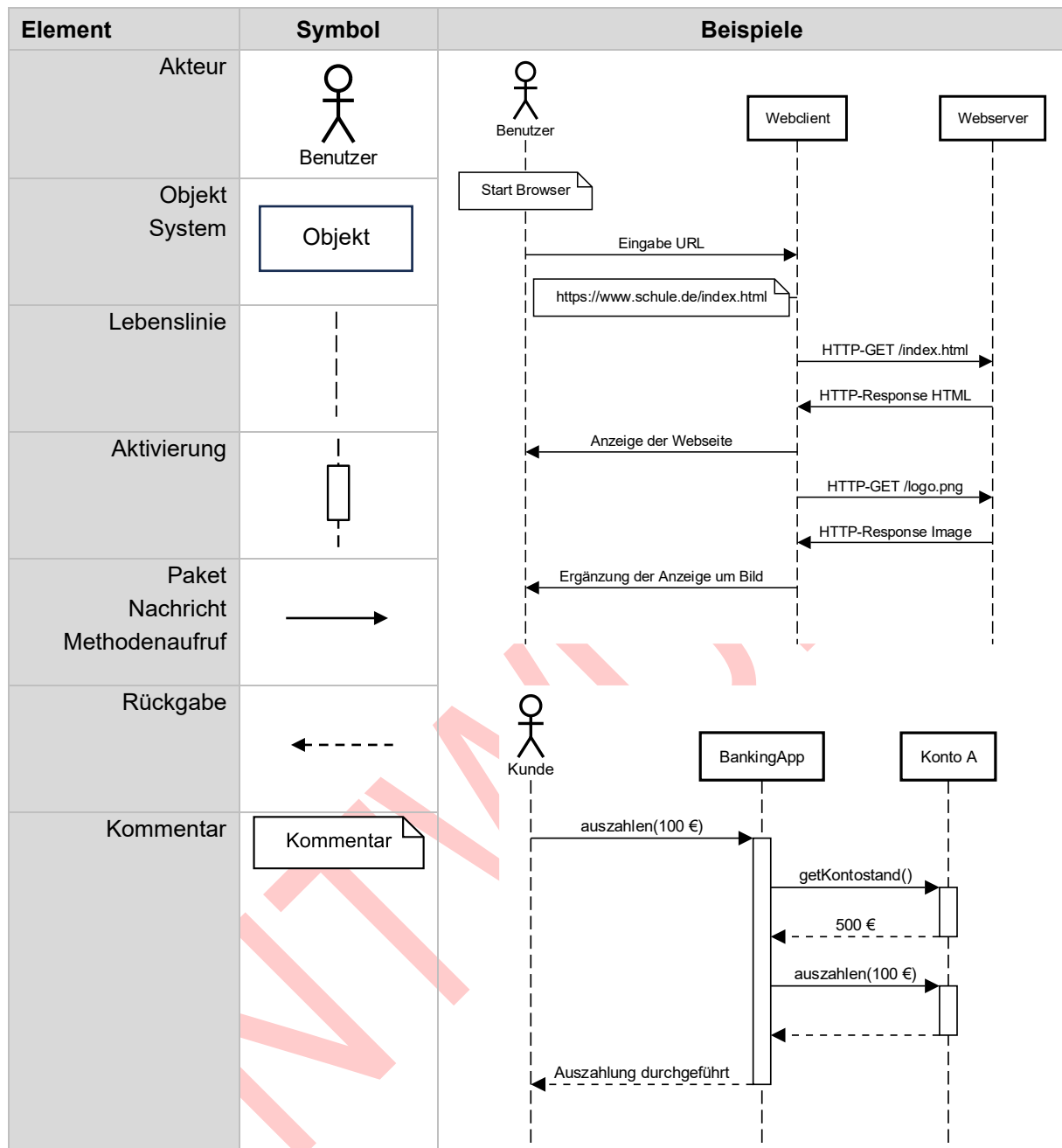
2.3 Klassendiagramm



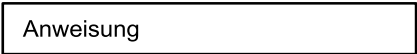
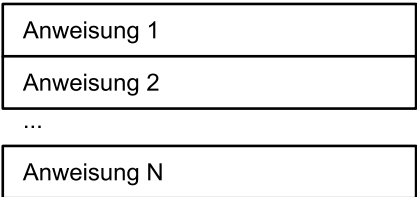
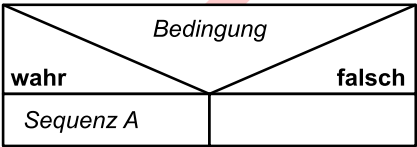
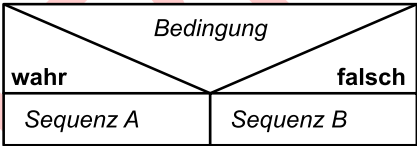
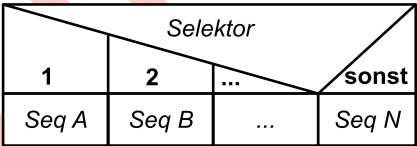
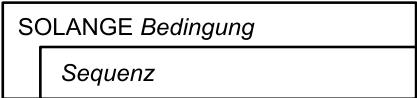
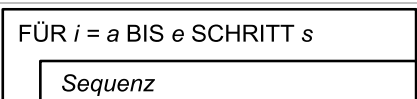
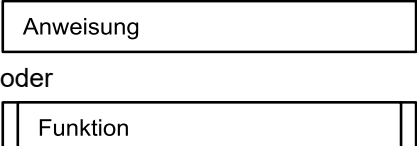
2.4 Schaltplan



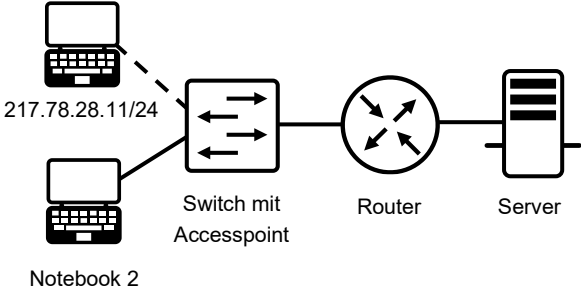
2.5 Sequenzdiagramm



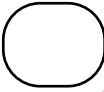
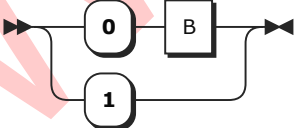
2.6 Struktogramm

Element	Verbale Formulierung	Struktogramm
Anweisung	<i>Anweisung</i>	
Sequenz	<i>Anweisung 1</i> <i>Anweisung 2</i> ... <i>Anweisung n</i>	
Auswahl		
Einseitige Auswahl	WENN <i>Bedingung</i> DANN <i>Sequenz</i>	
Zweiseitige Auswahl	WENN <i>Bedingung</i> DANN <i>Sequenz 1</i> SONST <i>Sequenz 2</i>	
Mehrseitige Auswahl	WENN <i>Selektor</i> 1: <i>Seq 1</i> 2: <i>Seq 2</i> ... SONST: <i>Seq n</i>	
Schleifen		
Bedingte Schleife	SOLANGE <i>Bedingung</i> WIEDERHOLE <i>Sequenz</i>	
Schleife mit fester Anzahl	WIEDERHOLE <i>n</i> MAL <i>Sequenz</i>	
Zählschleife	FÜR <i>i</i> VON <i>a</i> BIS <i>e</i> IN <i>s</i> -SCHRITT WIEDERHOLE <i>Sequenz</i>	
Funktion		
Funktionsaufruf	<i>Funktionsname</i>	

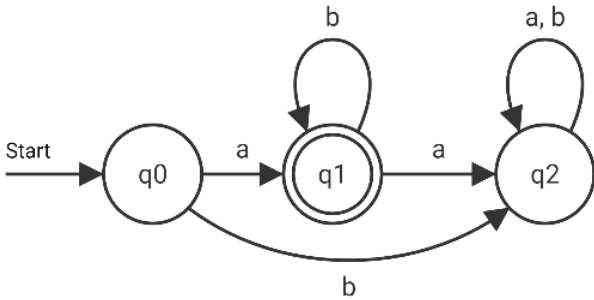
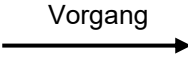
2.7 Netzstrukturdiagramm

Element	Symbolbeispiel	Beispiel
Endgerät		
Switch		
Router		
Wolke Black Box		
Kabel Funk	 	

2.8 Syntaxdiagramm

Element	Symbol	Beispiel
Terminal: Element des Alphabets		S:  B: 
Nichtterminal: Hilfssymbol, Variable		

2.9 Zustandsübergangsdiagramm

Element	Symbol	Beispiel
Zustand		
Startzustand		
Endzustand		
Zustandsübergang		

3 Datenbanksysteme

3.1 Entity-Relationship-Modell

Entitätstyp	Entitäten (Objekte) mit gleichartigen Eigenschaften (Attributen)
Beziehungstyp	Gleichartiger Zusammenhang zwischen Entitäten
Attribut	Eigenschaft der Entitäten eines Entitätstyps
Kardinalität	maximale Anzahl von Entitäten eines Entitätstyps, die mit genau einer Entität des anderen Entitätstyps in Beziehung stehen Grundformen: 1:1 1:n n:m
Darstellung im ER-Diagramm	↗ Seite 9

3.2 Relationales Datenbankmodell

Relationenschema	Primärschlüssel	Attribut	Attributwert	Fremdschlüssel		
Aktivität	ID	Name	Tag	Zeit	Raum	↑Verein.ID
	262	Fußball	Dienstag	14:30	TH1	Sp1
	285	Handball	Dienstag	16:00	TH1	Sp1
	286	Schach	Montag	14:30	2.01	Sp2
	319	Zeichnen	Mittwoch	15:00	2.01	Km5
	...	...	...	...	...	...

Relation

Notationen	
Grundnotation	Relationsname (<u>Primärschlüssel</u> , Attribut, ..., ↑Fremdschlüssel)
Hinweise	Primärschlüssel wird oft mit PK für primary key und Fremdschlüssel mit FK für foreign key abgekürzt. Fremdschlüssel werden oft in einer der drei Weisen notiert: ↑Fremdschlüssel ↑Tabelle.Schlüssel ↑Tabelle.Schlüssel AS Fremdschlüssel
Beispiel	Aktivität (<u>ID</u> , Name, Tag, Zeit, Raum, ↑Verein.ID)

Umwandlungsregeln ER-Modell → relationales Modell	
Entitätstyp	Relationsname (<u>Primärschlüssel</u> , Attribut, ...)
n:m-Beziehung	Relationsname (<u>↑PK1</u> , <u>↑PK2</u> , Attribut, ...) Es entsteht eine neue Relation. Die Primärschlüssel der beiden Entitäten werden als Fremdschlüssel übernommen. Sie bilden zusammen den Primärschlüssel. Bei mehrfach möglichen Beziehungen kann der Primärschlüssel um weitere Attribute (z. B. Zeitangaben) erweitert werden.
1:n-Beziehung	Der Primärschlüssel der 1-Seite wird als Fremdschlüssel in die Relation der n-Seite übernommen. Beziehungsattribute werden ebenfalls dort zugeordnet.
1:1-Beziehung	Der Primärschlüssel einer der beiden Relationen wird als Fremdschlüssel in die andere Relation übernommen. Beziehungsattribute werden ebenfalls dort zugeordnet.
Beispiel	Person (<u>ID</u> , Name, Vorname) Verein (<u>ID</u> , Name, Kontakt) Aktivität (<u>ID</u> , Name, Tag, Zeit, Raum, <u>↑Verein.ID</u>) Belegung (<u>↑Person.ID</u> , <u>↑Aktivität.ID</u> , Startzeitpunkt)

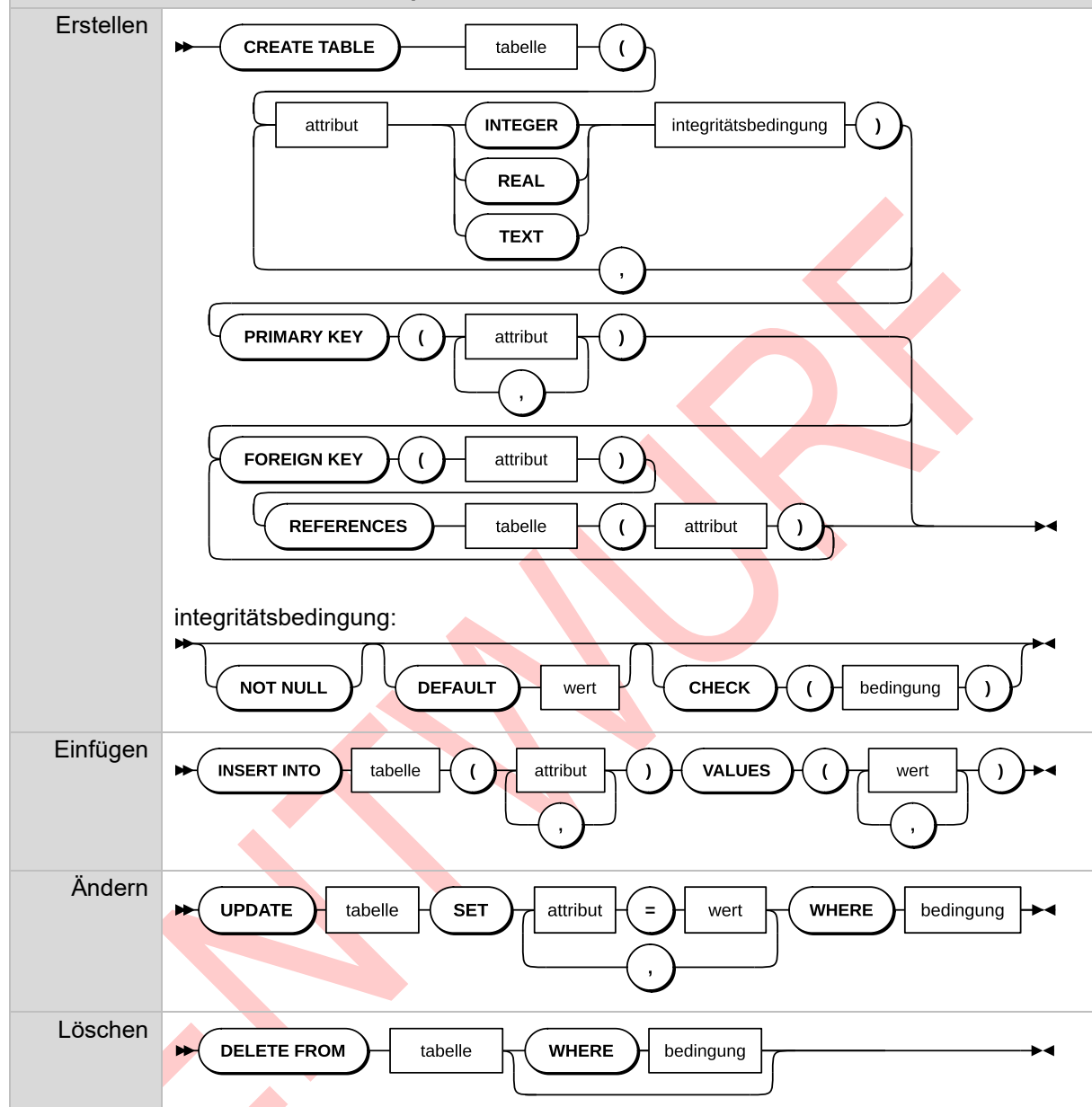
Normalisierung	
Ziele	Strukturierung von Daten Minimierung von Redundanzen Vermeidung von Änderungs-, Einfüge- und Löschanomalien
1. Normalform	Die Relation enthält ausschließlich atomare Attributwerte.
2. Normalform	Die Relation liegt in erster Normalform vor. Jedes Nichtschlüsselattribut ist vom gesamten Primärschlüssel voll funktional abhängig.
3. Normalform	Die Relation liegt in zweiter Normalform vor. Es gibt keine transitiven Abhängigkeiten vom Primärschlüssel.

Elemente der Relationenalgebra	
Selektion	Auswahl von Datensätzen einer Relation, die einer Bedingung genügen. Notation: $S_{\text{Bedingung}}(\text{Relation})$
Projektion	Reduktion einer Relation auf ausgewählte Attribute mit Entfernung von Duplikaten. Notation: $P_{\text{AttributA}, \text{AttributB}}(\text{Relation})$
Verbund (Join)	Verknüpfung zweier Relationen zu einer neuen Relation über ein gemeinsames Attribut Notationen: $J_{\text{Attribut}}(\text{RelationA}, \text{RelationB})$ $J_{\text{RelationA.AttributA} = \text{RelationB.AttributB}}(\text{RelationA}, \text{RelationB})$ $\text{RelationA} \bowtie \text{RelationB}$

3.3 Datenbanksprache SQL

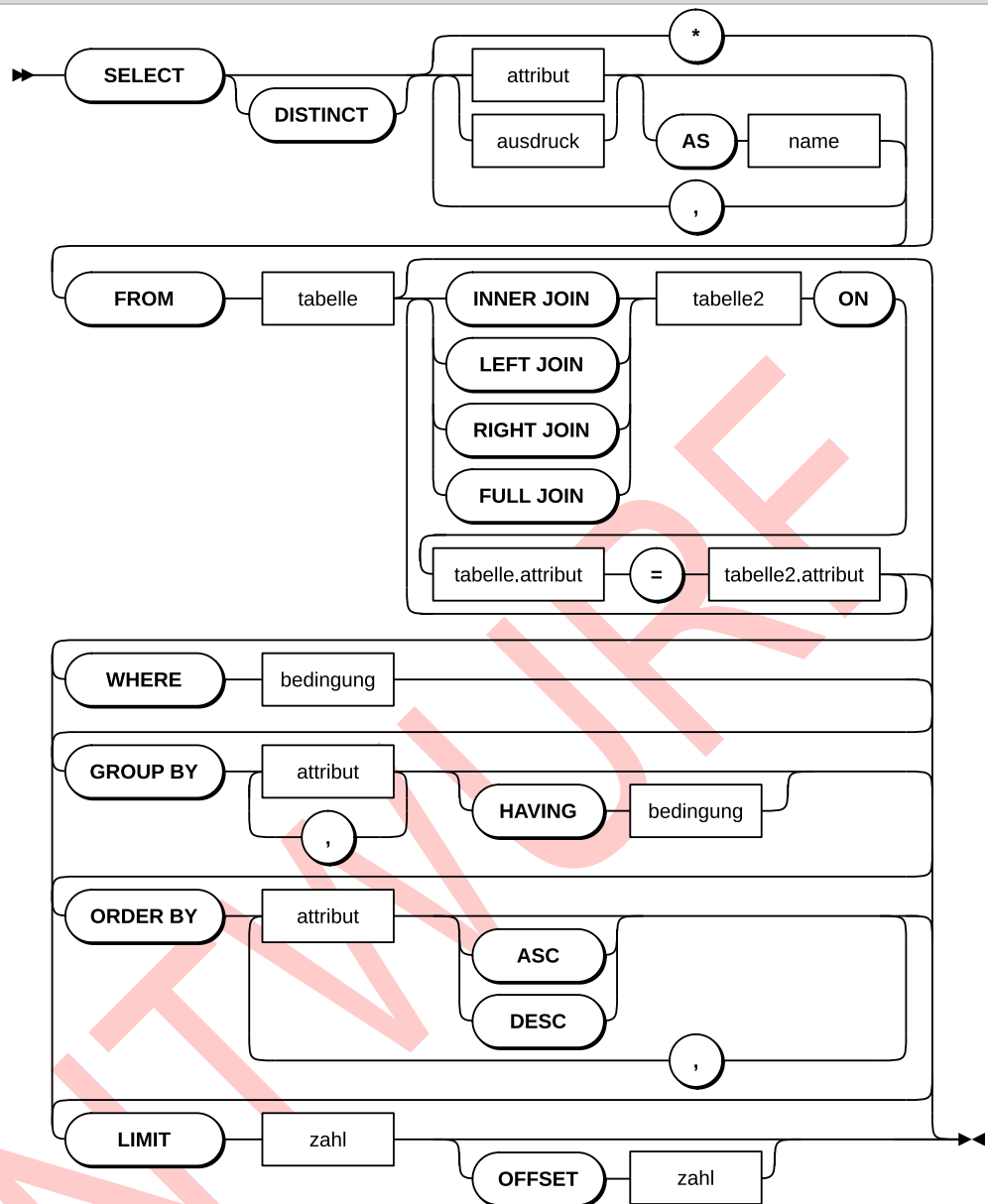
Hinweis: Die Diagramme zeigen eine inhaltlich reduzierte, nicht vollständig ausgeführte Syntax. Für eine vollständige Beschreibung wird auf die Dokumentation verwiesen.

Definition einer Relation und Manipulation von Daten



Abfragen von Daten (Überblick)

Überblick



Bedingung	Ausdruck mit dem Ergebnis WAHR oder FALSCH	
	Vergleichsoperatoren	= != < <= > >=
	Logische Operatoren	AND OR NOT
	Bereichsvergleich von x bis y	wert BETWEEN x AND y
	Aufzählungsvergleich	wert IN (a, b, c, d)
	NULL-Wert-Vergleich	wert IS NULL wert IS NOT NULL
	Mustervergleich mit Platzhalter % ... beliebige Zeichen _ ... genau ein Zeichen	wert LIKE muster Beispiel: wert LIKE 'A%'
Ausdruck	Mathematischer Term oder eine Funktion, die einen Wert bestimmt	
	Mathematische Operationen	+ - * / %
	Aggregatfunktionen	COUNT() SUM() MAX() MIN() AVG()

3.4 SQLite

Datentyp	Bezeichnung	Bemerkung
Ganzzahl	INTEGER	
Gleitkommazahl	REAL	
Zeichenkette	TEXT	
Wahrheitswert		Ersatztyp INTEGER mit 0 $\triangleq$ FALSCH, 1 $\triangleq$ WAHR
Zeitstempel		Ersatztyp TEXT im Format YYYY-MM-DD hh:mm:ss
Datum		Ersatztyp TEXT im Format YYYY-MM-DD
Uhrzeit		Ersatztyp TEXT im Format hh:mm:ss

Funktion (Auswahl)	Beschreibung
CAST(a AS typ)	a im Datentyp typ
DATE(now)	aktuelles Datum in der Form YYYY-MM-DD
TIME(now)	aktuelle Zeit in der Form hh:mm:ss
LENGTH(z)	Länge der Zeichenkette z
LOWER(z)	z in Kleinbuchstaben
UPPER(z)	z in Großbuchstaben
REPLACE(z, z1, z2)	z mit Ersetzung von z1 durch z2
SUBSTR(z, p, e)	Teilzeichenkette von z ab Position p mit e Elementen
TRIM(z)	z ohne Leerzeichen am Anfang und Ende
ABS(x)	Betrag von x
ROUND(x)	x kaufmännisch gerundet
RANDOM()	Zufallszahl

4 Rechnerarchitektur

4.1 Von-Neumann- und Harvard-Architektur

Rechner/Computer	
Universalität	Eine Vielzahl von Problemen ist mithilfe geeigneter Programme und Daten lösbar.
Programm	Ein Programm ist eine Folge von Anweisungen, die entsprechend ihrer Abarbeitungsfolge hintereinander im Speicher steht. Sprungbefehle erlauben Abweichungen von der sequenziellen Abarbeitung der Befehle.
von-Neumann-Architektur mit Aufbau und Arbeitsprinzip	
Aufbau	<pre> graph TD subgraph Prozessor direction TB R[Rechenwerk] S[Steuerwerk] end E[Eingabewerk] <--> Bus P[Prozessor] P <--> Bus A[Ausgabewerk] P <--> Bus Sp[Speicherwerk] </pre>
Speicher	Auf gleichgroße und fortlaufend nummerierte Zellen besteht über die Zellnummer (Adresse) lesender und schreibender Zugriff. Die Zellwerte sind binär codiert und enthalten Programmcode, Daten oder Ergebnisse.
Zyklus	<pre> graph LR 1[1 Fetch] --> 2[2 Decode] 2 --> 3[3 Fetch Operands] 3 --> 4[4 Execute] 4 --> 5[5 Write Back] 5 --> 1 </pre>
Elemente der Harvard-Architektur	
Aufbau	<pre> graph TD subgraph Prozessor direction TB R[Rechenwerk] S[Steuerwerk] end D[Datenspeicher] <--> P[Prozessor] Pr[Programmspeicher] <--> P E[Eingabewerk] <--> P A[Ausgabewerk] <--> P </pre>
Speicher	Programm- und Datenspeicher sind streng getrennt. Der Zugriff erfolgt über je einen eigenen Bus. Auf gleichgroße und fortlaufend nummerierte Zellen besteht über die Zellnummer (Adresse) lesender bzw. schreibender Zugriff. Die Zellwerte sind binär codiert.

4.2 Maschinennahe Programmierung

Johnny			
Hinweise	Der Speicher ist über die Adressen 000 bis 999 erreichbar. Die Daten in einer Speicherzelle bestehen aus den zwei Teilen Hi und Lo. Es sind Werte von 00 000 bis 19 999 zulässig. Befehle werden auf dem Hi-Teil mit einem Wert von 01 bis 10 codiert. Der Operand steht dann im Lo-Teil und gibt eine Adresse an.		
Befehlssatz	Befehl	Code	Funktion
	TAKE <i>adr</i>	01 <i>adr</i>	Lade den Wert von <i>adr</i> in den Akkumulator
	ADD <i>adr</i>	02 <i>adr</i>	Addiere den Wert von <i>adr</i> zum Akkumulator
	SUB <i>adr</i>	03 <i>adr</i>	Subtrahiere den Wert von <i>adr</i> vom Akkumulator
	SAVE <i>adr</i>	04 <i>adr</i>	Speichere den Akkumulatorwert auf <i>adr</i>
	JMP <i>adr</i>	05 <i>adr</i>	Springe zu <i>adr</i>
	TST <i>adr</i>	06 <i>adr</i>	Überspringe nächsten Befehl, falls Wert auf <i>adr</i> Null
	INC <i>adr</i>	07 <i>adr</i>	Erhöhe den Wert von <i>adr</i> um Eins
	DEC <i>adr</i>	08 <i>adr</i>	Vermindere den Wert von <i>adr</i> um Eins
	NULL <i>adr</i>	09 <i>adr</i>	Speichere den Wert 00 000 auf <i>adr</i>
	HLT	10	Beende das Programm
MOPS			
Hinweise	Adressen bestehen aus dem Zeichen \$ gefolgt von der Nummer der Speicherzelle. Daten können ausschließlich auf den Adressen \$64 bis \$71 gespeichert werden. Das Ziel eines Sprungbefehls wird entweder durch die Adresse oder durch eine Zielmarke der Form „MNummer“ angegeben. Die Zielmarke wird hinter dem Befehl auf der Zieladresse in der Form „:MNummer“ definiert.		
Befehlssatz	Befehl	Code	Funktion
	ld <i>adr</i>	10 <i>adr</i>	Lade den Wert von <i>adr</i> in Akkumulator
	ld <i>val</i>	11 <i>val</i>	Lade den Wert <i>val</i> in Akkumulator
	st <i>adr</i>	12 <i>adr</i>	Speichere den Akkumulatorwert auf <i>adr</i>
	in <i>adr</i>	20 <i>val</i>	Schreibe den Wert des Eingaberegisters auf <i>adr</i>
	out <i>adr</i>	22 <i>adr</i>	Schreibe den Wert von <i>adr</i> ins Ausgaberegister
	out <i>val</i>	23 <i>val</i>	Schreibe den Wert <i>val</i> ins Ausgaberegister
	add <i>adr</i>	30 <i>adr</i>	Addiere den Wert von <i>adr</i> zum Akkumulator
	add <i>val</i>	31 <i>val</i>	Addiere den Wert <i>val</i> zum Akkumulator
	sub <i>adr</i>	32 <i>adr</i>	Subtrahiere den Wert von <i>adr</i> vom Akkumulator
	sub <i>val</i>	33 <i>val</i>	Subtrahiere den Wert <i>val</i> vom Akkumulator
	mul <i>adr</i>	34 <i>adr</i>	Multipliziere den Wert von <i>adr</i> mit Akkumulator
	mul <i>val</i>	35 <i>val</i>	Multipliziere den Wert <i>val</i> mit Akkumulator
	div <i>adr</i>	36 <i>adr</i>	Dividiere ganzzahlig den Akkumulator durch den Wert auf <i>adr</i>
	div <i>val</i>	37 <i>val</i>	Dividiere ganzzahlig den Akkumulator durch den Wert <i>val</i>
	mod <i>adr</i>	38 <i>adr</i>	Berechne Divisionsrest von Akkumulator durch Wert auf <i>adr</i>
	mod <i>val</i>	39 <i>val</i>	Berechne Divisionsrest von Akkumulator durch Wert <i>val</i>
	cmp <i>adr</i>	40 <i>adr</i>	Vergleiche Akkumulator mit Wert auf <i>adr</i>
	cmp <i>val</i>	41 <i>val</i>	Vergleiche Akkumulator mit Wert <i>val</i>
	jmp <i>tar</i>	50 <i>tar</i>	Springe zu <i>tar</i>
	jlt <i>tar</i>	52 <i>tar</i>	Springe zu <i>tar</i> , falls Akkumulator bei vorherigem CMP kleiner
	jeq <i>tar</i>	54 <i>tar</i>	Springe zu <i>tar</i> , falls Akkumulator bei vorherigem CMP gleich
	jgt <i>tar</i>	56 <i>tar</i>	Springe zu <i>tar</i> , falls Akkumulator bei vorherigem CMP größer
	end	60 00	Beende Programm

4.3 Logikgatter, Logikfunktionen und Schaltalgebra

Element	NOT nicht	AND und	OR oder																																				
Symbol																																							
Funktion	$Y = \neg A$	$Y = A \wedge B$	$Y = A \vee B$																																				
Wertetabelle	<table><tr><th>A</th><th>Y</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	A	Y	0	1	1	0	<table><tr><th>A</th><th>B</th><th>Y</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	Y	0	0	0	0	1	0	1	0	0	1	1	1	<table><tr><th>A</th><th>B</th><th>Y</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	Y	0	0	0	0	1	1	1	0	1	1	1	1
A	Y																																						
0	1																																						
1	0																																						
A	B	Y																																					
0	0	0																																					
0	1	0																																					
1	0	0																																					
1	1	1																																					
A	B	Y																																					
0	0	0																																					
0	1	1																																					
1	0	1																																					
1	1	1																																					

Element	NAND nicht und	NOR nicht oder	XOR exklusiv oder	XNOR Äquivalenz																																																												
Symbol																																																																
Funktion	$Y = \neg(A \wedge B)$	$Y = \neg(A \vee B)$	$Y = A \oplus B$	$Y = \neg(A \oplus B)$																																																												
Wertetabelle	<table><tr><th>A</th><th>B</th><th>Y</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	Y	0	0	1	0	1	1	1	0	1	1	1	0	<table><tr><th>A</th><th>B</th><th>Y</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	Y	0	0	1	0	1	0	1	0	0	1	1	0	<table><tr><th>A</th><th>B</th><th>Y</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	Y	0	0	0	0	1	1	1	0	1	1	1	0	<table><tr><th>A</th><th>B</th><th>Y</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	Y	0	0	1	0	1	0	1	0	0	1	1	1
A	B	Y																																																														
0	0	1																																																														
0	1	1																																																														
1	0	1																																																														
1	1	0																																																														
A	B	Y																																																														
0	0	1																																																														
0	1	0																																																														
1	0	0																																																														
1	1	0																																																														
A	B	Y																																																														
0	0	0																																																														
0	1	1																																																														
1	0	1																																																														
1	1	0																																																														
A	B	Y																																																														
0	0	1																																																														
0	1	0																																																														
1	0	0																																																														
1	1	1																																																														

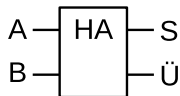
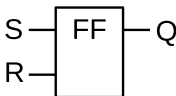
Bildungsvorschrift
für XOR

$$A \oplus B = (A \wedge \neg B) \vee (\neg A \wedge B)$$

$$A \oplus B = (A \vee B) \wedge \neg(A \wedge B)$$

Regeln der Schaltalgebra für NOT, AND und OR

Reihenfolge	Klammern → NOT → AND → OR	
Doppelte Negation	$A = \neg(\neg A)$	
Kommutativgesetz	$A \wedge B = B \wedge A$	$A \vee B = B \vee A$
Assoziativgesetz	$A \wedge (B \wedge C) = (A \wedge B) \wedge C$	$A \vee (B \vee C) = (A \vee B) \vee C$
Distributivgesetz	$A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C)$	$A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C)$
De-Morgan-Regeln	$\neg(A \wedge B) = \neg A \vee \neg B$	$\neg(A \vee B) = \neg A \wedge \neg B$
Neutralität	$A \wedge 1 = A$	$A \vee 0 = A$
Dominanzgesetz	$A \wedge 0 = 0$	$A \vee 1 = 1$
Selbstverknüpfung	$A \wedge A = A$	$A \vee A = A$
Komplementgesetz	$A \wedge \neg A = 0$	$A \vee \neg A = 1$
Absorptionsgesetz	$A \wedge (A \vee B) = A$	$A \vee (A \wedge B) = A$
Reduktionsregel	$A \wedge (\neg A \vee B) = A \wedge B$	$A \vee (\neg A \wedge B) = A \vee B$

Rechnen und Speichern																																						
Element	Halbaddierer		RS-Flipflop																																			
Zweck	binäre Addition zweier Bits		Speichern eines Bits																																			
Symbol																																						
Funktion	$S = A \oplus B, \ddot{U} = A \wedge B$		$Q_{(neu)} = f(S, R, Q_{(alt)})$																																			
Wertetabelle	<table><tr><th>A</th><th>B</th><th>S</th><th>Ü</th></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td><td>1</td></tr></table>		A	B	S	Ü	0	0	0	0	0	1	1	0	1	0	1	0	1	1	0	1	<table><tr><th>R</th><th>S</th><th>Q_(neu)</th></tr><tr><td>0</td><td>0</td><td>Q_(alt)</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>nicht sinnvoll</td></tr></table>	R	S	Q _(neu)	0	0	Q _(alt)	0	1	1	1	0	0	1	1	nicht sinnvoll
A	B	S	Ü																																			
0	0	0	0																																			
0	1	1	0																																			
1	0	1	0																																			
1	1	0	1																																			
R	S	Q _(neu)																																				
0	0	Q _(alt)																																				
0	1	1																																				
1	0	0																																				
1	1	nicht sinnvoll																																				

Darstellung	
Schaltplan	↗ Seite 10

5 Vernetzte Systeme

5.1 Schichtenmodell und Protokolle

TCP/IP-Schicht	OSI-Schicht	Gerät Applikation	Protokoll
Anwendung	Anwendung	Client- und Server-Software	HTTP Hypertext Transfer Protocol
	Darstellung		SMTP Simple Mail Transfer Protocol
	Sitzung		POP3 Post Office Protocol
Transport	Transport	Firewall	IMAP Internet Message Access Protocol
			DNS Domain Name System
			DHCP Dynamic Host Configuration Protocol
Internet	Vermittlung	Router Firewall	TCP Transmission Control Protocol
Netzzugang	Sicherung	Switch WLAN-AP Modem Netzkarte Netzkabel	UDP User Datagram Protocol
			IP Internet Protocol
			ICMP Internet Control Message Protocol
Netzzugang	Sicherung	Switch WLAN-AP Modem Netzkarte Netzkabel	ARP Address Resolution Protocol
			MAC Medium Access Control
			DSL Digital Subscriber Line
Netzzugang	Bitübertragung	Switch WLAN-AP Modem Netzkarte Netzkabel	IEEE 802.11 .. Funknetzwerkprotokoll
			IEEE 802.3 Ethernetprotokoll

5.2 Netzstrukturen und Kommunikationsabläufe

Diagramme	
Netzstruktur	↗ Seite 13
Kommunikation	↗ Seite 11

5.3 Sichere Kommunikation

Sicherheitsziele	Vertraulichkeit, Integrität, Authentizität, Verbindlichkeit
Prinzip von Kerckhoffs	Das kryptografische Verfahren muss im Wesentlichen unentzifferbar sein und darf keine Geheimhaltung erfordern. Die Schlüssel bedürfen weiterhin der Geheimhaltung.
Schlüsselprinzip	symmetrisch gleicher Schlüssel zum Ver- und Entschlüsseln asymmetrisch verschiedene Schlüssel zum Ver- und Entschlüsseln
Klassische symmetrische Verfahren	Substitution Zeichen werden ersetzt monoalphabetisch genau ein Geheimentalphabet polyalphabetisch mehrere Geheimentalphabete Transposition Zeichen werden umgeordnet
Moderne Verfahren	Algorithmen symmetrisch, asymmetrisch Protokolle Schlüsseltausch, Signatur, hybride Systeme Infrastruktur Zertifikate, Trust Center

Vigenère-Quadrat

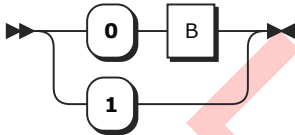
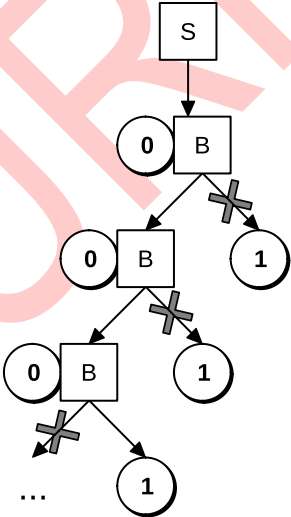
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

6 Sprachen und Automaten

6.1 Automatenmodelle

Formale Notation	Zustandsübergangsdiagramm ↗ Seite 13
Mealy-Automat $MA = (X, Y, Z, \delta, \lambda, q_0)$	
X Eingabealphabet (nichtleer, endlich) Y Ausgabealphabet (nichtleer, endlich) Z Zustandsmenge (nichtleer, endlich) δ totale Überföhrungsfunktion $Z \times X \rightarrow Z$ λ totale Ausgabefunktion $Z \times X \rightarrow Y$ q_0 Startzustand $q_0 \in Z$	$MA = (\{10, 20\}, \{-, Tee\}, \{q_0, q_1\}, \delta, \lambda, q_0)$
Deterministischer endlicher Automat – Akzeptor $A = (X, Z, \delta, q_0, Z_E)$	
X Eingabealphabet (nichtleer, endlich) Z Zustandsmenge (nichtleer, endlich) δ totale Überföhrungsfunktion $Z \times X \rightarrow Z$ q_0 Startzustand $q_0 \in Z$ Z_E Endzustandsmenge $Z_E \subseteq Z$	$A = (\{a, b\}, \{q_0, q_1, q_2\}, \delta, q_0, \{q_1\})$ $L(A) = \{ab^n, n \geq 0\}$
Deterministische Turingmaschine $TM = (X, Z, \Gamma, \delta, q_0, \\$, Z_E)$	
X Eingabealphabet (nichtleer, endlich) Z Zustandsmenge (nichtleer, endlich) Γ Bandalphabet $X \subset \Gamma$ δ partielle Überföhrungsfunktion $Z \times \Gamma \rightarrow Z \times \Gamma \times \{L, N, R\}$ q_0 Startzustand $q_0 \in Z$ $\$$ Bandvorbelegungszeichen $\$ \in \Gamma$ und $\$ \notin X$ Z_E Endzustandsmenge $Z_E \subseteq Z$	$TM = (\{0, 1\}, \{q_0, q_1, q_2\}, \{\$, 0, 1\}, \delta, q_0, \$, \{q_2\})$ $L(TM) = \{w \mid w \text{ endet auf } 1\}$
Eine Turingmaschine akzeptiert das Eingabewort, falls der aktuelle Zustand ein Endzustand und mit dem aktuellen Eingabezeichen kein Übergang mehr möglich ist. Das Halteproblem („Hält eine TM für eine gegebene Eingabe?“) ist nicht entscheidbar, da keine allgemeine Methode existiert, die dies für alle Programme bestimmt.	
Deterministischer/Nichtdeterministischer Kellerautomat $KA = (X, Z, \Gamma, \delta, z_0, k_0, Z_E)$	
X Eingabealphabet (nichtleer, endlich) Z Zustandsmenge (nichtleer, endlich) Γ Kelleralphabet δ partielle Überföhrungsfunktion: deterministisch $Z \times (X \cup \{\epsilon\}) \times \Gamma \rightarrow Z \times \Gamma^*$ nichtdeterministisch $Z \times (X \cup \{\epsilon\}) \times \Gamma \rightarrow \mathcal{P}(Z \times \Gamma^*)$ z_0 Startzustand $z_0 \in Z$ k_0 Kellerleerzeichen $k_0 \in \Gamma$ und $k_0 \notin X$ Z_E Endzustandsmenge $Z_E \subseteq Z$	$KA = (\{a, b\}, \{q_0, q_1, q_2\}, \{\#, A\}, \delta, q_0, \#, \{q_2\})$ $L(KA) = \{a^n b^n, n \geq 1\}$
Ein Kellerautomat akzeptiert das Eingabewort, falls er sich nach Abarbeitung des Wortes im Endzustand befindet und der Keller leer ist. <i>Hinweis: Beim Lesen des Kellerzeichens wird dieses aus dem Keller gelöscht. Soll es erhalten bleiben, muss es wieder in den Keller geschrieben werden.</i>	

6.2 Formale Sprachen und formale Grammatiken

Formale Notation	Grafische Notation anhand eines Beispiels
Grammatik $G = (N, T, P, S)$	
N Menge der Nichtterminale (nichtleer, endlich) T Menge der Terminale (nichtleer, endlich) P Menge der Produktionsregeln S Startsymbol $S \in N$	Syntaxdiagramm ↗ Seite 13 $G = (\{S, B\}, \{0, 1\}, \{S \rightarrow 0 B, B \rightarrow 0 B \mid 1\}, S)$: S:  B: 
Die formale Sprache $L(G)$ ist die Menge aller von einer Grammatik G erzeugten Worte.	$L(G) = \{0^n 1 \text{ mit } n > 0\}$. Ableitungsbaum für das Wort $w = 0001 \in L(G)$ 

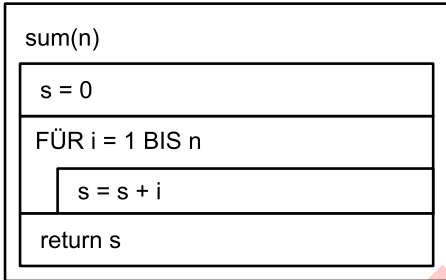
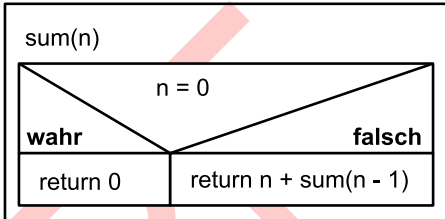
Chomsky-Hierarchie			
Typ	Grammatik	Produktionsregeln	Hinweis
0	allgemein	$\alpha \rightarrow \beta, \alpha \neq \epsilon$	A, B Variable α, β, γ Terminal- und Nichtterminalfolgen ϵ..... leeres Terminal a Terminal Verknüpfung „oder“
1	kontextsensitiv	$\alpha A \beta \rightarrow \alpha \gamma \beta, \gamma \neq \epsilon$	
2	kontextfrei	$A \rightarrow \alpha$	
3	regulär	entweder $A \rightarrow \epsilon \mid a \mid a B$ oder $A \rightarrow \epsilon \mid a \mid B a$	

Zusammenhang zwischen der Chomsky-Hierarchie und erkennenden Automaten		
Typ	Grammatik	Erkennende Automaten
0	allgemein	Turingmaschine
1	kontextsensitiv	Turingmaschine
2	kontextfrei	Nichtdeterministischer Kellerautomat, Turingmaschine
3	regulär	Akzeptor, Kellerautomat, Turingmaschine

7 Softwareentwicklung

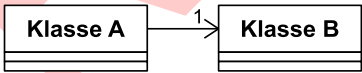
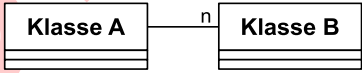
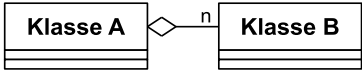
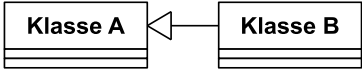
7.1 Algorithmen

Beschreibung	Ein Algorithmus ist eine eindeutige Handlungsvorschrift zur Lösung einer Klasse von Problemen. Algorithmen bestehen aus einer endlichen und wohldefinierten Abfolge von Anweisungen, die systematisch ausgeführt werden. Durch den Algorithmus werden Eingabewerte schrittweise verarbeitet und in Ausgabewerte umgewandelt. Nach der Church-Turing-These ist die Menge der algorithmisch berechenbaren Funktionen identisch mit der Menge der von Turingmaschinen berechenbaren Funktionen.													
Parameter	Parameter ermöglichen flexible Eingangswerte.													
Variable	Eine Variable ermöglicht das Speichern eines Datums (Singular von Daten) während der Abarbeitung des Algorithmus.													
Eigenschaften	allgemeingültig	Ein Algorithmus gilt für eine Klasse gleichartiger Probleme. Die Spezifizierung erfolgt mithilfe von Parametern.												
	ausführbar	Alle Anweisungen sind verständlich formuliert und ausführbar.												
	endlich	Die Beschreibung erfolgt in einem endlich langen Text.												
	eindeutig	Mit jeder Anweisung ist auch die nächstfolgende festgelegt. Gleiche Eingabewerte werden bei wiederholter Abarbeitung auf dieselben Ausgangswerte abgebildet.												
	terminiert	Nach endlich vielen Schritten ist eine Lösung erreicht.												
Darstellungen	Struktogramm ↗ Seite 12 Programmablaufplan ↗ Seite 9 Zustandsübergang Turingmaschine ↗ Seite 25 Pseudocode ↗ Seite 12 Quelltext ↗ Seite 30													
Zeitkomplexität	Beschreibung des Laufzeitverhaltens eines Algorithmus in Abhängigkeit von der Eingabegröße													
O-Notation	obere Schranke der Zeitkomplexität Beispiele für vergleichsbasierte Sortierverfahren: <table border="1"> <thead> <tr> <th>Sortierverfahren</th><th>Best Case</th><th>Worst Case</th></tr> </thead> <tbody> <tr> <td>Bubble Sort</td><td>$O(n)$</td><td>$O(n^2)$</td></tr> <tr> <td>Insertion Sort</td><td>$O(n)$</td><td>$O(n^2)$</td></tr> <tr> <td>Merge Sort</td><td>$O(n \log n)$</td><td>$O(n \log n)$</td></tr> </tbody> </table>		Sortierverfahren	Best Case	Worst Case	Bubble Sort	$O(n)$	$O(n^2)$	Insertion Sort	$O(n)$	$O(n^2)$	Merge Sort	$O(n \log n)$	$O(n \log n)$
Sortierverfahren	Best Case	Worst Case												
Bubble Sort	$O(n)$	$O(n^2)$												
Insertion Sort	$O(n)$	$O(n^2)$												
Merge Sort	$O(n \log n)$	$O(n \log n)$												

Iterative und rekursive Algorithmen		
	Iteration	Rekursion
Beschreibung	Wiederholung einer Sequenz entsprechend einer Bedingung	Aufruf einer parameterbehafteten Funktion aus sich selbst heraus, bis eine Abbruchbedingung erfüllt ist. Zerlegung des Problems in Teilprobleme (Teile und Herrsche)
Beispiel	Algorithmus zur Berechnung der Summe der natürlichen Zahl bis n	
Formal	$\text{sum}(n) = 0 + 1 + 2 + \dots + n$	$\text{sum}(0) = 0$ $\text{sum}(n) = n + \text{sum}(n-1)$
Struktogramm	 <pre> graph TD subgraph sum_n [sum(n)] s0[s = 0] loop[FÜR i = 1 BIS n] subgraph loop_body [] splus[s = s + i] end return_s[return s] s0 --> loop loop --> loop_body loop_body --> return_s end </pre>	 <pre> graph TD subgraph sum_n [sum(n)] n0{n = 0} return0[return 0] return_rec[return n + sum(n - 1)] n0 -- wahr --> return0 n0 -- falsch --> return_rec end </pre>

7.2 Objektorientierte Modellierung

Element	Symbol
Objekt	bezeichner1: KlasseA attribut1 = wert1 attribut2 = wert2 bezeichner2: KlasseB attribut1 = wert1 attribut2 = bezeichner1
Klasse	Klassenname Attribute Methoden <i>Hinweis: Vor Attributen und Methoden wird oft ihre Sichtbarkeit angegeben.</i>
Sichtbarkeit von Attribut und Methode	+ öffentlich, public # geschützt, protected - privat, private

Beziehungen	Beschreibung	Symbol
Assoziation	Gerichtet: Ein Objekt der Klasse A steht mit mindestens einem Objekt der Klasse B in Beziehung. Ungerichtet: Die Beziehung besteht wechselseitig.	 
Aggregation	Ein Objekt der Klasse A beinhaltet mindestens ein Objekt der Klasse B.	
Multiplizität bei Assoziation bzw. Aggregation	maximale Anzahl von Objekten einer Klasse, die mit einem Objekt der anderen Klasse in Beziehung stehen <i>Hinweis:</i> <i>Statt des Begriffs Multiplizität wird teilweise auch der Begriff Kardinalität verwendet.</i>	1 genau ein Objekt * beliebig viele Objekte n beliebig viele Objekte
Vererbung	Ein Unterklasse B ist eine Spezialisierung der Oberklasse A. B enthält von A alle vererbten Attribute und Methoden. B kann eigene Attribute und Methoden ergänzen oder vererbte Methoden überschreiben. Vererbbar sind Attribute und Methoden mit der Sichtbarkeit <i>öffentlich</i> oder <i>geschützt</i> .	

8 Programmierhilfen für Java und Python

8.1 Algorithmische Strukturen

Struktogramm	Java	Python
	Anweisung;	Anweisung
	Anweisung 1; Anweisung 2; ... Anweisung N;	Anweisung 1 Anweisung 2 ... Anweisung N
	if (Bedingung) { Sequenz }	if Bedingung: Sequenz
	if (Bedingung) { Sequenz A } else { Sequenz B }	if Bedingung: Sequenz A else: Sequenz B
	switch (Selektor) { case 1: Seq A break; case 2: Seq B break; ... default: Seq N; }	if Selektor == 1: Seq A elif Selektor == 2: Seq B ... else: Seq N
	while (Bedingung) { Sequenz }	while Bedingung: Sequenz
	for (int i=1; i<=n; i++) { Sequenz }	for i in range(n): Sequenz
	for (int i=a; i<=e; i+=s) { Sequenz }	for i in range(a, e+1, s): Sequenz
	for (int i=a; i>=e; i-=s) { Sequenz }	for i in range(a, e-1, -s): Sequenz
	for (typ element: liste) { Sequenz }	for element in liste: Sequenz

8.2 Kommentare

Art	Java	Python
Einzeiliger Kommentar	// Bemerkung	# Bemerkung
Mehrzeiliger Kommentar	/* Bemerkung über mehrere Zeilen */	""" Bemerkung über mehrere Zeilen """
Dokumentation einer Klasse	/** Name/Beschreibung der Klasse * @author Autor * @version Version */	""" Name/Beschreibung der Klasse mehrere Zeilen """
Dokumentation einer Methode	/** Name/Beschreibung der Methode * @param Name Beschreibung * @return Rückgabebeschreibung */	""" Name/Beschreibung der Klasse Parameters: Beschreibung Returns: Beschreibung """

8.3 Datentypen und Variablen

Variable	Java	Python
Typisierung	statisch	dynamisch
Deklaration	<i>typ variablenname</i>	
Deklaration mit Initialisierung	<i>typ variablenname = wert</i>	<i>variablenname = wert</i>
Gültigkeit	innerhalb der Struktur, in der sie deklariert wurde	innerhalb der Funktion/Methode oder der Klasse, in der sie deklariert wurde

Datentyp	Java			Python	
	Art	Bezeichner	Wrapperklasse	Art	Bezeichner
Ganze Zahl	primitiver Typ	int	Integer	Objekt	int
Gleitkommazahl	primitiver Typ	double	Double	Objekt	float
Wahrheitswert	primitiver Typ	boolean	Boolean	Objekt	bool
Zeichen	primitiver Typ	char	Character		
Zeichenkette	Objektyp	String		Objekt	str
Liste	Objektyp	ArrayList		Objekt	list

Eingabe ganzer Zahlen in verschiedenen Zahlssystemen

Zahlensystem	Basis	Präfix	Beispiel
Dezimalsystem	10		a = 26
Binärsystem	2	0b	a = 0b00011010
Hexadezimalsystem	16	0x	a = 0x1A

Ausgabe ganzer Zahlen in verschiedenen Zahlssystemen als Zeichenkette mit Präfix

Zahlsystem	Java	Python
Dezimalsystem	Integer.toString(a)	str(a)
Binärsystem	"0b" + Integer.toHexString(a)	bin(a)
Hexadezimalsystem	"0x" + Integer.toHexString(a)	hex(a)

8.4 Datentypenumwandlung

Aufgabe	Java	Python
Casting: Umwandlung der Variablen a in ...		
<i>Hinweis: nur falls möglich und dann in der Regel mit Informationsverlust</i>		
eine ganze Zahl	(int) a	int(a)
eine Gleitkommazahl	(double) a	float(a)
ein Zeichen	(char) a	
eine Zeichenkette	String.valueOf(a)	str(a)
ein Objekt der Klasse K	(K) a	

Aufgabe	Java	Python
Parsing: Umwandeln einer Zeichenkette z in ...		
eine ganze Zahl	Integer.parseInt(z)	int(z)
eine Gleitkommazahl	Double.parseDouble(z)	float(z)
einen Wahrheitswert	Boolean.parseBoolean(z)	bool(z)
ein Zeichen	z.charAt(0)	

Aufgabe	Java	Python
Bestimmung des Datentyps/der Klasse des Objekts o		
	o.getClass()	type(o)
	o instanceof KLASSE Für KLASSE muss der Name der Klasse eingetragen werden. Das Ergebnis ist ein Wahrheitswert.	

8.5 Ein- und Ausgaben

	Java	Python
Voraussetzung	vor der Klassendefinition bei Verwendung von Dialogfenster: import javax.swing.*	
Eingabe einer Zeichenkette in Variable a mit Ausgabe eines Hinweistexts z		
Dialogfenster	a = JOptionPane.showInputDialog(null, z)	
Konsole		a = input(z)
Ausgabe einer Zeichenkette z		
Dialogfenster	JOptionPane.showMessageDialog(null, z)	
Konsole	System.out.println(z) System.out.print(z)	print(z) print(z, end="")
Ausgabe einer Liste		
Besonderheit	Bei der Ausgabe einer Liste werden die Elemente kommasepariert und geklammert als Zeichenkette dargestellt.	

8.6 Operatoren

Beschreibung	Java	Python
Zuweisungsoperatoren		
Zuweisung	=	=
Erhöhung um	+=	+=
Verminderung um	-=	-=
Vervielfachung um	*=	*=
Inkrement (Erhöhung um 1)	var++	
Dekrement (Verminderung um 1)	var--	
Arithmetische Operatoren		
Addition/Vorzeichen	+	+
Subtraktion/Vorzeichen	-	-
Multiplikation	*	*
Division	/	/
Potenzieren	Math.pow(b, e)	**
ganzzahliger Teil bei Division zweier ganzer Zahlen	/	//
ganzzahliger Rest bei Division zweier ganzer Zahlen (Modulo)	%	%
Vergleichsoperatoren		
identisch	==	==
ungleich	!=	!=
kleiner als	<	<
größer als	>	>
kleiner gleich	<=	<=
größer gleich	>=	>=
Logische Operatoren		
logisch NICHT	!	not
logisch UND	&&	and
logisch ODER		or
logisch EXKLUSIV ODER	^	^
Bitorientierte Operatoren		
bitweise NICHT	~	~
bitweises UND	&	&
bitweises ODER		
bitweises EXKLUSIV ODER	^	^
bitweises Linksschieben	<<	<<
bitweises Rechtsschieben	>>	>>

8.7 Mathematische Konstanten, Funktionen und Zufallszahlen

	Java	Python
Voraussetzung		vor Klassendefinition: import math import random
Konstanten		
π	Math.PI	math.pi
e	Math.E	math.e
Funktionen		
$ x $	Math.abs(x)	abs(x)
$\sqrt{x}$	Math.sqrt(x)	math.sqrt(x)
x^a	Math.pow(x, a)	math.pow(x, a)
e^x	Math.exp(x)	math.exp(x)
$x!$		math.factorial(x)
$\ln(x)$	Math.log(x)	math.log(x)
$\sin(x)$	Math.sin(x)	math.sin(x)
$\cos(x)$	Math.cos(x)	math.cos(x)
$\tan(x)$	Math.tan(x)	math.tan(x)
$\arcsin(x)$	Math.asin(x)	math.asin(x)
$\arccos(x)$	Math.acos(x)	math.acos(x)
$\arctan(x)$	Math.atan(x)	math.atan(x)
$\text{ggT}(x,y)$		math.gcd(x,y)
Umrechnung		
Bogenmaß in Gradmaß	Math.toDegree(x)	math.degrees(x)
Gradmaß in Bogenmaß	Math.toRadians(x)	math.radians(x)
Runden		
kaufmännisch ¹	Math.round(x)	round(x) round(x, d)
aufunden	Math.ceil(x)	math.ceil(x)
abrunden	Math.floor(x)	math.floor(x)
Zufallszahlen		
Gleitkommazahl aus [0, 1)	Math.random();	random.random()
Ganze Zahl aus [a, b]	(int) ((b-a+1)*Math.random()+a)	random.randint(a, b)

¹ „Kaufmännisches Runden“ ist in den beiden Programmiersprachen nicht einheitlich umgesetzt.

8.8 Zeichenketten

Beschreibung	Java	Python
Besonderheiten		Hochkomma statt Anführungszeichen möglich
Zeichenkette z erzeugen und initialisieren		
leere Zeichenkette	<code>z = ""</code>	<code>z = ""</code>
Zeichenkette mit vordefiniertem Wert	<code>z = "Hallo Welt"</code>	<code>z = "Hallo Welt"</code>
Umwandeln von Zeichenketten		
Rückgabe: z in Kleinbuchstaben	<code>z.toLowerCase()</code>	<code>z.lower()</code>
Rückgabe: z in Großbuchstaben	<code>z.toUpperCase()</code>	<code>z.upper()</code>
Rückgabe: z mit Ersetzung von z1 durch z2	<code>z.replaceAll(z1, z2)</code>	<code>z.replace(z1, z2)</code>
Zugriff auf Zeichen in der Zeichenkette z		
Rückgabe: z an Stelle i	<code>z.substring(i, i+1)</code>	<code>z[i]</code>
Rückgabe: z vom Typ char/int an Stelle i	<code>z.charAt(i)</code>	
Rückgabe: z von Stelle m bis n	<code>z.substring(m, n+1)</code>	<code>z[m:n+1]</code>
Zeichenkette z prüfen		
nur WAHR, falls z1 identisch mit z2	<code>z1.equals(z2)</code>	<code>z1 == z2</code>
nur WAHR, falls z1 in z enthalten	<code>z.contains(z1)</code>	<code>z1 in z</code>
nur WAHR, falls z mit z1 beginnt	<code>z.startsWith(z1)</code>	<code>z.startswith(z1)</code>
nur WAHR, falls z mit z1 endet	<code>z.endsWith(z1)</code>	<code>z.endswith(z1)</code>
Rückgabe: Stelle des ersten Auftretens von z1 in z, sonst -1	<code>z.indexOf(z1)</code>	<code>z.find(z1)</code>
Rückgabe: Stelle des letzten Auftretens von z1 in z, sonst -1	<code>z.lastIndexOf(z1)</code>	<code>z.rfind(z1)</code>
nur WAHR, falls z1 lexikographisch kleiner als z2		<code>z1 < z2</code>
< 0, falls z lexikographisch kleiner als z1 = 0, falls z lexikographisch gleich z1 > 0, falls z lexikographisch größer als z1	<code>z.compareTo(z1)</code>	
sonstiges		
Zeichenketten z1 und z2 verknüpfen	<code>z1 + z2</code>	<code>z1 + z2</code>
Rückgabe: Länge von z	<code>z.length()</code>	<code>len(z)</code>
Rückgabe: Liste mit den durch z1 getrennten Teilzeichenketten aus z	<code>ArrayList<String> li li = new ArrayList<>(List.of(z.split(z1)))</code>	<code>li = z.split(z1)</code>

8.9 Listen

Beschreibung	Java	Python
Besonderheit	<code>ArrayList<typ></code> Elemente gleichen Objekttyps	Elemente unterschiedlichen Typs
Voraussetzung	<code>import java.util.*;</code>	
Listen primitiven Typs	über Wrapper-Klasse, z. B.: <code>ArrayList<Integer> li = ...</code>	
Liste deklarieren und initialisieren		
li deklarieren	<code>ArrayList<typ> li</code>	
li als leere Liste initialisieren	<code>li = new ArrayList<>()</code>	<code>li = []</code>
li mit Objekten initialisieren	<code>li = new ArrayList<>(List.of(o1, o2, ...))</code>	<code>li = [o1, o2, ...]</code>
<i>Beispiel Zeichenkettenliste</i>	<code>ArrayList<String> li; li = new ArrayList<>(); li.add("Ha"); li.add("se");</code>	<code>li = ["Ha", "se"]</code>
<i>Beispiel Ganzzahlenliste</i>	<code>ArrayList<Integer> li; li = new ArrayList<>(List.of(1,3));</code>	<code>li = [1,3]</code>
Hinzufügen/Austauschen von Elementen		
Objekt obj an li hinten anfügen	<code>li.add(obj)</code>	<code>li.append(obj)</code>
Objekt obj an li vorn anfügen	<code>li.add(0, obj)</code>	<code>li.insert(obj)</code>
Objekt obj an Stelle i einfügen	<code>li.add(i, obj)</code>	<code>li.insert(i, obj)</code>
Objekt obj an Stelle i ersetzen	<code>li.set(i, obj)</code>	<code>li[i] = obj</code>
Entfernen von Elementen		
Element an Stelle i entfernen	<code>li.remove(i)</code>	<code>del li[i] li.pop(i)</code>
erstes Auftreten von obj entfernen	<code>li.remove(obj)</code>	<code>li.remove(obj)</code>
Liste leeren	<code>li.clear()</code>	<code>li.clear()</code>
Zugriff auf Elemente und Listeneigenschaften		
Element an Stelle i zurückgeben	<code>li.get(i)</code>	<code>li[i]</code>
Listenlänge zurückgeben	<code>li.size()</code>	<code>len(li)</code>
Listen und Elemente prüfen		
nur WAHR, falls Liste leer	<code>li.isEmpty()</code>	<code>len(li) == 0</code>
nur WAHR, falls beide Listen identisch	<code>li1.equals(li2)</code>	<code>li1 == li2</code>
nur WAHR, falls obj in Liste enthalten	<code>li.contains(obj)</code>	<code>obj in li</code>
Stelle des ersten Auftretens von obj falls obj in Liste enthalten, sonst -1	<code>li.indexOf(obj)</code>	<code>li.index(obj)</code>
Ändern der Reihenfolge		
Liste aufsteigend sortieren	<code>Collections.sort(li)</code>	<code>li.sort()</code>
Liste umkehren	<code>Collections.reverse(li)</code>	<code>li.reverse()</code>
Liste anfügen/zerlegen/kopieren		
li2 an li1 anfügen	<code>li1.addAll(li2)</code>	<code>li1 = li1 + li2</code>
neue Liste mit Elementen von li von Stelle m bis n zurückgeben	<code>li = new ArrayList<>(li.subList(m, n+1))</code>	<code>li[m:n+1]</code>
Kopie der Liste mit referenzierten Elementen erstellen	<code>new ArrayList<>(li)</code>	<code>li.copy()</code>

8.10 Objektorientierte Programmierung

	Java	Python
Einfache Klasse		
Definition	<pre>public class Klasse { //Attribut //Konstruktor //Methoden }</pre>	<pre>class Klasse: #Konstruktor #Methoden</pre>
Attribut	<i>Sichtbarkeit Typ Attribut;</i>	Definition/Initialisierung im Konstruktor
Konstruktor	<pre>public Klasse() { ... }</pre>	<pre>def __init__(self): self.SichtbarkeitAttribut = Wert</pre>
Methode ohne Rückgabe	<pre>Sichtbarkeit void Methode() { ... }</pre>	<pre>def SichtbarkeitMethode(self): ...</pre>
Methode mit Rückgabe	<pre>Sichtbarkeit Typ Methode() { ... return ...; }</pre>	<pre>def SichtbarkeitMethode(self): ... return ...</pre>
Unterklasse		
Definition	<pre>public class Unter extends Ober { ... }</pre>	<pre>class Unter(Ober): ...</pre>
Konstruktor	<pre>public Unter(){ super(); ... }</pre>	<pre>def __init__(self): super().__init__() ...</pre>

Elemente einer Klasse		
Element	Java	Python
Sichtbarkeit öffentlich geschützt privat	public protected private	ohne Zeichen _ einzelner Unterstrich __ doppelter Unterstrich
Parameter-deklaration	<i>Typ pParameter</i>	<i>pParameter</i>
Instanziierung	<i>objektname = new Klasse();</i>	<i>objektname = Klasse()</i>
Methodenaufruf eigene Klasse Oberklasse anderes Objekt	this.methode(); super.methode(); objektname.methode();	self.methode() super().methode() objektname.methode()
Zugriff auf Attribute/Methoden der ...		
eigenen Klasse	this.attribut this.methode() <i>this kann bei Eindeutigkeit entfallen</i>	self.attribut self.methode()
Oberklasse	super.attribut; super.methode();	super().attribut super().methode()

9 Aufgabenoperatoren im Fach Informatik

Ein Operator impliziert eine Aufforderung. Er bestimmt die zu wählenden Mittel und Methoden für eine überprüfbare Bearbeitung der Aufgabenstellung.

Die nachfolgende Übersicht zeigt eine Auswahl typischer Operatoren.

Operator	Beschreibung
analysieren	eine Materialgrundlage untersuchen, Elemente identifizieren, Beziehungen zwischen den Elementen erfassen, das Untersuchungsergebnis darstellen
angeben	ohne Erläuterung, Begründung oder Lösungsweg nennen
begründen	durch rational nachvollziehbare Argumente basierend auf Regeln, Prinzipien oder kausale Zusammenhänge stützen
beschreiben	unter Verwendung der Fachsprache in eigenen Worten verständlich wiedergeben
bestimmen ermitteln	Ergebnisse unter Angabe von Annahmen, Regeln oder kausalen Zusammenhänge gewinnen
beurteilen bewerten	unter Verwendung von Fachwissen und Fachmethoden ein eigenständiges Sach- bzw. Werturteil formulieren und begründen
darstellen	in formalisierter Form grafisch oder fachsprachlich wiedergeben
entwickeln	unter Verwendung von Fachwissen und Fachmethoden nachvollziehbar herstellen oder konstruieren
erklären	einen Sachverhalt erfassen, in einen kausalen Zusammenhang einordnen und deuten
erläutern	mithilfe zusätzlicher Angaben (Beispiele, Vergleiche, ...) verständlich machen
ermitteln	Ergebnisse i. d. R. durch Rechenoperationen nachvollziehbar gewinnen
implementieren	ein informatisches Modell auf einem oder für ein Informatiksystem umsetzen
interpretieren	einen Sachverhalt erfassen und deuten
modellieren	gemäß einer Problemanalyse ein informatisches Modell anfertigen
modifizieren	unter Berücksichtigung konkreter Vorgaben anpassen und ggf. erweitern
protokollieren dokumentieren	schrittweise untersuchen und dabei in formalisierter Form fachsprachlich wiedergeben
Stellung nehmen	die eigene Meinung zu einem Problem argumentativ entwickeln und darlegen
überführen	in ein anderes informatisches Modell oder eine andere Darstellungsform bringen
vergleichen	kriteriengeleitet Gemeinsamkeiten und Unterschiede angeben

Index

A

Abfrage.....	17
Ableitungsbaum	26
Adresse.....	19
Aggregatfunktionen	17
Aggregation	29
Akzeptor	25
Algorithmus.....	27
AND	21
Anomalie	15
Antivalenz.....	21
Anweisung.....	12
Äquivalenz.....	21
ASCII	6, 7
Assoziation	29
asymmetrisch	23
Attribut.....	9, 14
Ausgabealphabet	25
Ausgabefunktion	25
Auswahl.....	12
Authentizität	23

B

Bandalphabet.....	25
Bandvorbelegungszeichen	25
Best Case	27
Beziehung.....	15
Beziehungstyp.....	9
Binärsystem.....	4
Bit	3
Byte	3

C

Chomsky-Hierarchie	26
Client	23
Codierung.....	5
CREATE TABLE	16

D

Datenbanksystem	14
Datencodierung	5

Datenminimierung	8
Datenschutz	8
Datensicherheit.....	8
Datentyp	3, 18, 31
DELETE FROM.....	16
deterministisch	25
Deterministischer endlicher Automat.....	25
Dezimalsystem	4

E

Eingabealphabet	25
Endzustand.....	13
Entitätstyp.....	9, 15
Entity-Relationship-Modell	14
ER-Modell.....	14

F

Festkommazahl	6
Firewall.....	23
Formale Sprache	26
Fremdschlüssel.....	14
Funktion	12, 18

G

Ganzzahl.....	3, 18
vorzeichenbehaftet	5
vorzeichenlos	5
Gatter	21
Gleitkommazahl	3, 18
Grammatik	26

H

Halbaddierer	22
Harvard-Architektur	19
Hexadezimalsystem	4

I

Informationelle Selbstbestimmung	8
INSERT INTO	16
Integrität	8, 23

Iteration 28

J

Johnny 20

Join 15

K

Kardinalität 14, 22, 23, 29

Kelleralphabet 25

Kellerautomat 25

Kerckhoffs 23

Klasse 10, 29

Kommentare 31

kontextfrei 26

kontextsensitiv 26

Kryptologie 23

L

Lebenslinie 10, 11

Logikgatter 21

Logischer Operator 17

M

Mealy-Automat 25

monoalphabetisch 23

MOPS 20

Multiplizität 10

Mustervergleich 17

N

NAND 21

nichtdeterministisch 25

Nichtterminal 13

NOR 21

Normalform 15

Normalisierung 15

NOT 21

NULL-Wert 17

O

Objekt 10, 29

O-Notation 27

OR 21

P

Parameter 27

polyalphabetisch 23

Präfix 3

Primärschlüssel 14

Prinzip von Kerckhoffs 23

Produktionsregeln 26

Programm 19

Projektion 15

Protokoll 23

R

Rechnerarchitektur 19

Redundanz 15

regulär 26

Rekursion 28

Relationales Datenbankmodell 14

Relationenalgebra 15

Relationenschema 14

Router 23

RS-Flipflop 22

S

Schaltalgebra 21

Schichtenmodell 23

OSI 23

TCP/IP 23

Schleife 12

Schlüsselattribut 9

SELECT 17

Selektion 15

Sequenz 12

Server 23

Sichtbarkeit 29

Speicher 19

Spezialisierung 29

SQL 16

SQLite 18

Startzustand 13, 25

Struktogramm 12, 28, 30

Switch 23

symmetrisch 23

Syntaxdiagramm 26

T

Terminal	13
Terminalmenge	26
Terminator	9
Turingmaschine	25

U

Überföhrungsfunktion	25
Unicode	6, 7
Universalitt	19
UPDATE	16
UTF-16	6
UTF-8	6

V

Variable	27, 31
Variablenmenge	26
Verbindlichkeit	23
Verbund	15
Vererbung	10, 29
Verfögbarkeit	8
Vergleichsoperator	17
Vertraulichkeit	8, 23
Vigenère-Quadrat	24
von-Neumann-Architektur	19

von-Neumann-Zyklus	19
--------------------------	----

W

Wahrheitswert	3, 18
Wertetabelle	21
Wiederholung	12
Worst Case	27

X

XNOR	21
XOR	21

Z

Zahlsystem	4
Zeichen	3
Zeichenkette	3, 18
Zelle	19
Zustand	13
Zustandsdiagramm	25
Zustandsmenge	25
Zustandsübergang	13
Zweckbindung	8
Zweierkomplement	5
Zyklus	19