

# HANSER



## Leseprobe

zu

## Programmieren trainieren

von Luigi Lo Iacono, Stephan Wiefling und Michael Schneider

Print-ISBN: 978-3-446-47766-7

E-Book-ISBN: 978-3-446-47821-3

E-Pub-ISBN: 978-3-446-47999-9

Weitere Informationen und Bestellungen unter

<https://www.hanser-kundencenter.de/fachbuch/artikel/9783446477667>

sowie im Buchhandel

© Carl Hanser Verlag, München

# Inhalt

<b>Vorwort</b> .....	<b>XIX</b>
Neues in der 3. Auflage .....	XX
Danksagung .....	XX
<b>1 Einleitung</b> .....	<b>1</b>
1.1 Wozu sollte ich programmieren lernen? .....	1
1.2 Wie kann mir dieses Buch dabei helfen? .....	2
1.3 Was muss ich mitbringen? .....	3
1.4 Wie geht das vonstatten? .....	3
1.5 Was muss ich sonst noch wissen? .....	5
1.6 Wie kann und sollte ich ChatGPT & Co verwenden? .....	6
<b>2 Einführung in die Programmierung</b> .....	<b>9</b>
2.1 Warm-up .....	9
2.2 Workout .....	13
W.2.1 Three-Two-One – Mein erstes Programm .....	13
W.2.2 Weihnachtsbaum .....	14
W.2.3 Haus .....	15
W.2.4 Perlenkette .....	16
W.2.5 Die erste Zeichnung .....	17
W.2.6 Nachteule .....	19
W.2.7 Daumen .....	20
W.2.8 Tasse <sup>(Neu)</sup> .....	21
W.2.9 Raupe Allzeitappetit .....	22
W.2.10 Klötzchen-Kunst .....	23
W.2.11 GhettoBlaster .....	24
W.2.12 Gesichtsmaske <sup>(Neu)</sup> .....	26
W.2.13 Hallo Bello! .....	27
<b>3 Variablen, Datentypen, Operatoren und Ausdrücke</b> .....	<b>29</b>
3.1 Warm-up .....	29
3.2 Workout .....	32
W.3.1 Einfache Rechenaufgaben .....	32
W.3.2 Perlenkette 2.0 .....	33
W.3.3 Blutalkoholkonzentration .....	34
W.3.4 Stoffwechselrate .....	36

W.3.5	Baumstammvolumen.....	38
W.3.6	Körperoberfläche.....	39
W.3.7	Schuhgröße <sup>(Neu)</sup> .....	40
W.3.8	Haus mit Garage.....	41
W.3.9	RGB nach CMYK.....	42
W.3.10	Tic-Tac-Toe-Spielfeld.....	44
W.3.11	Gamecontroller <sup>(Neu)</sup> .....	45
W.3.12	Fußballtor.....	46
<b>4</b>	<b>Kontrollstrukturen.....</b>	<b>49</b>
4.1	Warm-up.....	49
4.2	Workout.....	53
W.4.1	Maximum bestimmen.....	53
W.4.2	Summe berechnen.....	54
W.4.3	Tippspiel.....	55
W.4.4	PIN-Code-Generator.....	56
W.4.5	Ladevorgang-Rädchen.....	57
W.4.6	Windrad.....	59
W.4.7	Rotierte Dreiecke.....	60
W.4.8	Moderne Kunst.....	61
W.4.9	Schachbrett.....	62
W.4.10	Ebbe und Flut berechnen.....	63
W.4.11	Dominosteine.....	65
W.4.12	Radialer Farbverlauf.....	66
W.4.13	Sinuskurve.....	67
W.4.14	Sternzeichen bestimmen <sup>(Neu)</sup> .....	68
W.4.15	Zahlen-Palindrom.....	70
W.4.16	Titschender Ball.....	71
W.4.17	Interaktiver Button.....	73
W.4.18	CAPTCHA <sup>(Neu)</sup> .....	75
<b>5</b>	<b>Funktionen.....</b>	<b>77</b>
5.1	Warm-up.....	77
5.2	Workout.....	79
W.5.1	Endliches Produkt.....	79
W.5.2	Fakultät.....	80
W.5.3	Konfektionsgröße.....	81
W.5.4	Schaltjahr Prüfung.....	82
W.5.5	Literzahlen umwandeln.....	83
W.5.6	Analoger Uhrzeiger.....	84
W.5.7	Körperoberfläche 2.0.....	86
W.5.8	Sportwetten.....	87
W.5.9	Lkw-Maut.....	89
W.5.10	Wurfparabel.....	90
W.5.11	Tortendiagramm.....	92
W.5.12	Pixelart-Generator <sup>(Neu)</sup> .....	93
W.5.13	GPS-Luftlinie.....	94

W.5.14	IBAN-Generator .....	96
W.5.15	Sanduhr .....	97
W.5.16	Der faire Würfel .....	98
W.5.17	Quadrat mit Kreisausschnitten .....	99
W.5.18	Meme-Generator <sup>(Neu)</sup> .....	101
W.5.19	Mondphase berechnen .....	103
W.5.20	Pendelanimation .....	105
W.5.21	Bogenschießen-Spiel .....	107
<b>6</b>	<b>Arrays .....</b>	<b>109</b>
6.1	Warm-up .....	109
6.2	Workout .....	112
W.6.1	Tankladung .....	112
W.6.2	Rückwärtsausgabe .....	113
W.6.3	Minimale Distanz .....	114
W.6.4	Endlose Animation .....	115
W.6.5	Wochentag .....	116
W.6.6	Onlinedating .....	118
W.6.7	Sitzplatzreservierung .....	119
W.6.8	Platztausch .....	120
W.6.9	Spiegeln .....	121
W.6.10	Reflexion .....	122
W.6.11	Greenscreen .....	124
W.6.12	Bild umdrehen und invertieren .....	125
W.6.13	Bild mit Schatten .....	126
W.6.14	Bild rotieren .....	127
W.6.15	Bildverkleinerung .....	128
W.6.16	Bildvergrößerung .....	129
W.6.17	Durchschnittstemperatur <sup>(Neu)</sup> .....	130
W.6.18	Morsecode .....	132
W.6.19	Minimale Punktdistanz .....	133
W.6.20	Glatte Kurven .....	135
W.6.21	Bildausschnitt .....	137
W.6.22	Bild mit Rahmen .....	138
W.6.23	Memory-Spielfeldgenerator .....	139
W.6.24	Geldautomat .....	140
W.6.25	Sudoku-Check .....	141
W.6.26	Postleitzahlen visualisieren .....	142
W.6.27	Medianfilter .....	144
W.6.28	Dreiecksfilter .....	146
W.6.29	Gesichtserkennung <sup>(Neu)</sup> .....	148
<b>7</b>	<b>Strings und Stringverarbeitung .....</b>	<b>151</b>
7.1	Warm-up .....	151
7.2	Workout .....	153
W.7.1	String-Kompression .....	153
W.7.2	Leetspeak <sup>(Neu)</sup> .....	154

W.7.3	Split-Funktion .....	155
W.7.4	Geldschein-Blütencheck .....	156
W.7.5	E-Mail-Check .....	157
W.7.6	Prüfen auf eine korrekte Klammerung .....	158
W.7.7	URL-Encoding .....	159
W.7.8	Webserver-Antwort verarbeiten .....	160
W.7.9	IMDB-Einträge verarbeiten .....	162
W.7.10	Sternchenmuster .....	163
W.7.11	Geheimsprache .....	164
W.7.12	Ähnlich klingende Wörter .....	165
W.7.13	Textrahmen .....	167
W.7.14	Starkes Passwort .....	169
W.7.15	Telefonbuch bearbeiten .....	171
W.7.16	JSON-Array .....	173
W.7.17	Kennzeichenverarbeitung .....	174
W.7.18	Barcode-Generator .....	176
W.7.19	Datensatz-Vorverarbeitung <sup>(Neu)</sup> .....	178
W.7.20	Stimmungsbarometer <sup>(Neu)</sup> .....	180
<b>8</b>	<b>Objektorientierung .....</b>	<b>183</b>
8.1	Warm-up .....	183
8.2	Workout .....	186
W.8.1	Schrittzähler .....	186
W.8.2	Songtextsuche .....	188
W.8.3	Body-Mass-Index .....	189
W.8.4	Druckerwarteschlange .....	191
W.8.5	Stoppuhr .....	193
W.8.6	Parteistimmen .....	194
W.8.7	Kopffitness .....	196
W.8.8	Fernbedienung .....	198
W.8.9	Laufschrift <sup>(Neu)</sup> .....	199
W.8.10	Passwortklasse .....	200
W.8.11	Tic Tac Toe .....	202
W.8.12	Verschlüsselung .....	204
W.8.13	Zwischenablage .....	206
W.8.14	Temperaturgraph .....	207
W.8.15	Ambient Light .....	209
W.8.16	Autovervollständigung <sup>(Neu)</sup> .....	212
W.8.17	Mastermind .....	215
<b>9</b>	<b>Referenzdatentypen .....</b>	<b>217</b>
9.1	Warm-up .....	217
9.2	Workout .....	219
W.9.1	Kreis-Klasse .....	219
W.9.2	Mathematischer Bruch .....	220
W.9.3	Highscore-Liste .....	221
W.9.4	Adressbuch .....	222

W.9.5	Digitaler Bilderrahmen .....	224
W.9.6	Polygonzug .....	225
W.9.7	Flughafen-Check-in .....	227
W.9.8	Socialwall .....	229
W.9.9	Musikalbenanwendung .....	231
W.9.10	Koch-Website .....	233
W.9.11	Suchmaschinenranking <sup>(Neu)</sup> .....	235
W.9.12	Partygäste .....	237
W.9.13	Hotelzimmerverwaltung .....	239
W.9.14	Fototagebuch .....	241
W.9.15	Raumbelegung .....	243
W.9.16	Rotationspuzzle <sup>(Neu)</sup> .....	246
<b>10</b>	<b>Vererbung .....</b>	<b>249</b>
10.1	Warm-up .....	249
10.2	Workout .....	251
W.10.1	Lampen .....	251
W.10.2	Meeting-Protokoll .....	253
W.10.3	Online-Shop .....	255
W.10.4	Gewässer .....	257
W.10.5	Schere, Stein, Papier <sup>(Neu)</sup> .....	258
W.10.6	To-do-Liste .....	259
W.10.7	E-Book .....	261
W.10.8	Zoo .....	263
W.10.9	Audioeffekt-Player .....	264
W.10.10	Meetingplaner <sup>(Neu)</sup> .....	265
W.10.11	Tanzwettbewerb <sup>(Neu)</sup> .....	269
W.10.12	Fahrtenbuch .....	271
W.10.13	Webseitengenerator .....	272
<b>A</b>	<b>Installation Processing .....</b>	<b>275</b>
A.1	Einleitung .....	275
A.2	Windows .....	276
A.3	macOS .....	277
A.4	Linux .....	278
A.5	Aktivierung des Python Mode .....	279
<b>B</b>	<b>Howto: Buch-Aufgaben ohne Processing lösen .....</b>	<b>281</b>
B.1	Java .....	281
B.2	Python .....	281
B.3	Fazit .....	285
<b>C</b>	<b>Lösungen in Java .....</b>	<b>287</b>
C.1	Download und Verwendung der elektronischen Lösungen .....	287
C.1.1	Download von GitHub .....	287
C.1.2	Download von Hanser Plus .....	288
C.1.3	Öffnen der Programme .....	288
C.1.4	Tipp: Debugger .....	289

C.2	Einführung in die Programmierung.....	291
C.2.1	Three-Two-One – Mein erstes Programm.....	291
C.2.2	Weihnachtsbaum.....	292
C.2.3	Haus.....	293
C.2.4	Perlenkette.....	294
C.2.5	Die erste Zeichnung.....	295
C.2.6	Nachteule.....	296
C.2.7	Daumen.....	297
C.2.8	Tasse.....	298
C.2.9	Raupe Allzeitappetit.....	299
C.2.10	Klötzchen-Kunst.....	300
C.2.11	Ghettoblaster.....	301
C.2.12	Gesichtsmaske.....	303
C.2.13	Hallo Bello!.....	304
C.3	Variablen, Datentypen, Operatoren und Ausdrücke.....	306
C.3.1	Einfache Rechenaufgaben.....	306
C.3.2	Perlenkette 2.0.....	308
C.3.3	Blutalkoholkonzentration.....	309
C.3.4	Stoffwechselrate.....	310
C.3.5	Baumstammvolumen.....	311
C.3.6	Körperoberfläche.....	312
C.3.7	Schuhgröße.....	313
C.3.8	Haus mit Garage.....	314
C.3.9	RGB nach CMYK.....	315
C.3.10	Tic-Tac-Toe-Spielfeld.....	316
C.3.11	Gamecontroller.....	317
C.3.12	Fußballtor.....	318
C.4	Kontrollstrukturen.....	320
C.4.1	Maximum bestimmen.....	320
C.4.2	Summe berechnen.....	321
C.4.3	Tippspiel.....	322
C.4.4	PIN-Code-Generator.....	323
C.4.5	Ladevorgang-Rädchen.....	324
C.4.6	Windrad.....	325
C.4.7	Rotierte Dreiecke.....	326
C.4.8	Moderne Kunst.....	327
C.4.9	Schachbrett.....	328
C.4.10	Ebbe und Flut berechnen.....	329
C.4.11	Dominosteine.....	330
C.4.12	Radialer Farbverlauf.....	331
C.4.13	Sinuskurve.....	332
C.4.14	Sternzeichen bestimmen.....	333
C.4.15	Zahlen-Palindrom.....	335
C.4.16	Titschender Ball.....	337
C.4.17	Interaktiver Button.....	339
C.4.18	CAPTCHA.....	340

C.5	Funktionen .....	343
C.5.1	Endliches Produkt .....	343
C.5.2	Fakultät .....	344
C.5.3	Konfektionsgröße .....	345
C.5.4	Schaltjahr Prüfung .....	346
C.5.5	Literzahlen umwandeln .....	347
C.5.6	Analoger Uhrzeiger.....	348
C.5.7	Körperoberfläche.....	349
C.5.8	Sportwetten .....	350
C.5.9	LKW-Maut .....	352
C.5.10	Wurfparabel .....	354
C.5.11	Tortendiagramm.....	355
C.5.12	Pixelart-Generator .....	357
C.5.13	GPS-Luftlinie.....	358
C.5.14	IBAN-Generator .....	359
C.5.15	Sanduhr.....	361
C.5.16	Der faire Würfel.....	362
C.5.17	Quadrat mit Kreisabschnitten .....	364
C.5.18	Meme-Generator .....	366
C.5.19	Mondphase berechnen .....	368
C.5.20	Pendelanimation .....	369
C.5.21	Bogenschießen-Spiel .....	371
C.6	Arrays .....	375
C.6.1	Tankladung.....	375
C.6.2	Rückwärtsausgabe .....	376
C.6.3	Bestimmung minimale Distanz .....	377
C.6.4	Endlose Animation.....	378
C.6.5	Wochentag.....	379
C.6.6	Onlinedating .....	381
C.6.7	Sitzplatzreservierung .....	382
C.6.8	Platztausch .....	384
C.6.9	Spiegeln.....	385
C.6.10	Reflexion.....	387
C.6.11	Greenscreen.....	389
C.6.12	Bild umdrehen und invertieren .....	390
C.6.13	Bild mit Schatten .....	392
C.6.14	Bild rotieren .....	394
C.6.15	Bildverkleinerung .....	396
C.6.16	Bildvergrößerung.....	398
C.6.17	Durchschnittstemperatur .....	400
C.6.18	Morsecode.....	402
C.6.19	Minimale Punktdistanz.....	404
C.6.20	Glatte Kurven .....	405
C.6.21	Bildausschnitt.....	407
C.6.22	Bild mit Rahmen.....	409
C.6.23	Memory-Spielfeldgenerator .....	411
C.6.24	Geldautomat .....	413

C.6.25	Sudoku-Check .....	414
C.6.26	Postleitzahlen visualisieren .....	417
C.6.27	Medianfilter .....	419
C.6.28	Dreiecksfilter .....	421
C.6.29	Gesichtserkennung .....	423
C.7	Strings und Stringverarbeitung .....	425
C.7.1	String-Kompression .....	425
C.7.2	Leetspeak .....	426
C.7.3	Split-Funktion .....	428
C.7.4	Geldschein-Blütencheck .....	430
C.7.5	E-Mail-Check .....	433
C.7.6	Prüfen auf eine korrekte Klammerung .....	434
C.7.7	URL-Encoding .....	435
C.7.8	Webserver-Antwort verarbeiten .....	436
C.7.9	IMDB-Einträge verarbeiten .....	438
C.7.10	Sternchenmuster .....	440
C.7.11	Geheimsprache .....	442
C.7.12	Ähnlich klingende Wörter .....	443
C.7.13	Textrahmen .....	445
C.7.14	Starkes Passwort .....	446
C.7.15	Telefonbuch bearbeiten .....	448
C.7.16	JSON-Array .....	451
C.7.17	Kennzeichenverarbeitung .....	453
C.7.18	Barcode-Generator .....	455
C.7.19	Datensatz-Vorverarbeitung .....	459
C.7.20	Stimmungsbarometer .....	461
C.8	Objektorientierung .....	465
C.8.1	Schrittzähler .....	465
C.8.2	Songtextsuche .....	466
C.8.3	Body-Mass-Index .....	467
C.8.4	Druckerwarteschlange .....	468
C.8.5	Stoppuhr .....	470
C.8.6	Parteistimmen .....	472
C.8.7	Kopffitness .....	474
C.8.8	Fernbedienung .....	475
C.8.9	Laufschrift .....	477
C.8.10	Passwortklasse .....	479
C.8.11	Tic Tac Toe .....	482
C.8.12	Verschlüsselung .....	484
C.8.13	Zwischenablage .....	487
C.8.14	Temperaturgraph .....	489
C.8.15	Ambient Light .....	492
C.8.16	Autovervollständigung .....	495
C.8.17	Mastermind .....	497
C.9	Referenzdatentypen .....	500
C.9.1	Kreis-Klasse .....	500
C.9.2	Mathematischer Bruch .....	502

C.9.3	Highscore-Liste .....	504
C.9.4	Adressbuch .....	506
C.9.5	Digitaler Bilderrahmen .....	510
C.9.6	Polygonzug .....	512
C.9.7	Flughafen-Check-in .....	514
C.9.8	Socialwall .....	517
C.9.9	Musikalbenanwendung .....	519
C.9.10	Koch-Website .....	522
C.9.11	Suchmaschinenranking .....	525
C.9.12	Partygäste .....	528
C.9.13	Hotelzimmerverwaltung .....	531
C.9.14	Fototagebuch .....	534
C.9.15	Raumbelegung .....	537
C.9.16	Rotationspuzzle .....	541
C.10	Vererbung .....	546
C.10.1	Lampen .....	546
C.10.2	Meeting-Protokoll .....	548
C.10.3	Online-Shop .....	551
C.10.4	Gewässer .....	555
C.10.5	Schere, Stein, Papier .....	558
C.10.6	To-do-Liste .....	560
C.10.7	E-Book .....	564
C.10.8	Zoo .....	569
C.10.9	Audioeffekt-Player .....	572
C.10.10	Meetingplaner .....	575
C.10.11	Tanzwettbewerb .....	579
C.10.12	Fahrtenbuch .....	583
C.10.13	Webseitengenerator .....	586
<b>D</b>	<b>Lösungen in Python .....</b>	<b>593</b>
D.1	Download und Verwendung der elektronischen Lösungen .....	593
D.1.1	Download von GitHub .....	593
D.1.2	Download von Hanser Plus .....	593
D.1.3	Öffnen der Programme .....	593
D.2	Einführung in die Programmierung .....	595
D.2.1	Three-Two-One – Mein erstes Programm .....	595
D.2.2	Weihnachtsbaum .....	596
D.2.3	Haus .....	597
D.2.4	Perlenkette .....	598
D.2.5	Die erste Zeichnung .....	599
D.2.6	Nachteule .....	600
D.2.7	Daumen .....	601
D.2.8	Tasse .....	602
D.2.9	Raupe Allzeitappetit .....	603
D.2.10	Klötzchenkunst .....	604
D.2.11	Ghettoblaster .....	605

D.2.12	Gesichtsmaske .....	607
D.2.13	Hallo Bello! .....	608
D.3	Variablen, Datentypen, Operatoren und Ausdrücke .....	610
D.3.1	Einfache Rechenaufgaben .....	610
D.3.2	Perlenkette 2.0 .....	612
D.3.3	Blutalkoholkonzentration .....	613
D.3.4	Stoffwechselrate .....	614
D.3.5	Baumstammvolumen .....	615
D.3.6	Körperoberfläche .....	616
D.3.7	Schuhgröße .....	617
D.3.8	Haus mit Garage .....	618
D.3.9	RGB nach CMYK .....	619
D.3.10	Tic-Tac-Toe-Spielfeld .....	620
D.3.11	Gamecontroller .....	621
D.3.12	Fußballtor .....	622
D.4	Kontrollstrukturen .....	624
D.4.1	Maximum bestimmen .....	624
D.4.2	Summe berechnen .....	625
D.4.3	Tippspiel .....	626
D.4.4	PIN-Code-Generator .....	627
D.4.5	Ladevorgang-Rädchen .....	628
D.4.6	Windrad .....	629
D.4.7	Rotierte Dreiecke .....	630
D.4.8	Moderne Kunst .....	631
D.4.9	Schachbrett .....	632
D.4.10	Ebbe und Flut berechnen .....	633
D.4.11	Dominosteine .....	634
D.4.12	Radialer Farbverlauf .....	635
D.4.13	Sinuskurve .....	636
D.4.14	Sternzeichen bestimmen .....	637
D.4.15	Zahlen-Palindrom .....	639
D.4.16	Titschender Ball .....	641
D.4.17	Interaktiver Button .....	643
D.4.18	CAPTCHA .....	644
D.5	Funktionen .....	647
D.5.1	Endliches Produkt .....	647
D.5.2	Fakultät .....	648
D.5.3	Konfektionsgröße .....	649
D.5.4	Schaltjahr Prüfung .....	650
D.5.5	Literzahlen umwandeln .....	651
D.5.6	Analoger Uhrzeiger .....	652
D.5.7	Körperoberfläche .....	653
D.5.8	Sportwetten .....	654
D.5.9	LKW-Maut .....	656
D.5.10	Wurfparabel .....	657
D.5.11	Tortendiagramm .....	658
D.5.12	Pixelart-Generator .....	660

D.5.13	GPS-Luftlinie .....	661
D.5.14	IBAN-Generator .....	663
D.5.15	Sanduhr .....	664
D.5.16	Der faire Würfel .....	665
D.5.17	Quadrat mit Kreisausschnitten .....	667
D.5.18	Meme-Generator .....	668
D.5.19	Mondphase berechnen .....	670
D.5.20	Pendelanimation .....	671
D.5.21	Bogenschießen-Spiel .....	673
D.6	Arrays .....	677
D.6.1	Tankladung .....	677
D.6.2	Rückwärtsausgabe .....	678
D.6.3	Bestimmung minimale Distanz .....	679
D.6.4	Endlose Animation .....	680
D.6.5	Wochentag .....	681
D.6.6	Onlinedating .....	683
D.6.7	Sitzplatzreservierung .....	684
D.6.8	Platztausch .....	686
D.6.9	Spiegeln .....	687
D.6.10	Reflexion .....	689
D.6.11	Greenscreen .....	691
D.6.12	Bild umdrehen und invertieren .....	692
D.6.13	Bild mit Schatten .....	694
D.6.14	Bild rotieren .....	696
D.6.15	Bildverkleinerung .....	698
D.6.16	Bildvergrößerung .....	700
D.6.17	Durchschnittstemperatur .....	702
D.6.18	Morsecode .....	704
D.6.19	Minimale Punktdistanz .....	706
D.6.20	Glatte Kurven .....	707
D.6.21	Bildausschnitt .....	709
D.6.22	Bild mit Rahmen .....	711
D.6.23	Memory-Spielfeldgenerator .....	713
D.6.24	Geldautomat .....	715
D.6.25	Sudoku-Check .....	716
D.6.26	Postleitzahlen visualisieren .....	718
D.6.27	Medianfilter .....	720
D.6.28	Dreiecksfilter .....	721
D.6.29	Gesichtserkennung .....	723
D.7	Strings und Stringverarbeitung .....	725
D.7.1	String-Kompression .....	725
D.7.2	Leetspeak .....	726
D.7.3	Split-Funktion .....	727
D.7.4	Geldschein-Blütencheck .....	728
D.7.5	E-Mail-Check .....	730
D.7.6	Prüfen auf eine korrekte Klammerung .....	731
D.7.7	URL-Encoding .....	732

D.7.8	Webserver-Antwort verarbeiten .....	733
D.7.9	IMDB-Einträge verarbeiten .....	734
D.7.10	Sternchenmuster .....	735
D.7.11	Geheimsprache .....	736
D.7.12	Ähnlich klingende Wörter .....	737
D.7.13	Textrahmen .....	739
D.7.14	Starkes Passwort .....	740
D.7.15	Telefonbuch bearbeiten .....	742
D.7.16	JSON-Array .....	744
D.7.17	Kennzeichenverarbeitung .....	745
D.7.18	Barcode-Generator .....	747
D.7.19	Datensatz-Vorverarbeitung .....	750
D.7.20	Stimmungsbarometer .....	752
D.8	Objektorientierung .....	756
D.8.1	Schrittzähler .....	756
D.8.2	Songtextsuche .....	757
D.8.3	Body-Mass-Index .....	758
D.8.4	Druckerwarteschlange .....	759
D.8.5	Stoppuhr .....	761
D.8.6	Parteistimmen .....	763
D.8.7	Kopffitness .....	765
D.8.8	Fernbedienung .....	766
D.8.9	Laufschrift .....	768
D.8.10	Passwortklasse .....	770
D.8.11	Tic Tac Toe .....	772
D.8.12	Verschlüsselung .....	774
D.8.13	Zwischenablage .....	777
D.8.14	Temperaturgraph .....	779
D.8.15	Ambient Light .....	782
D.8.16	Autovervollständigung .....	784
D.8.17	Mastermind .....	786
D.9	Referenzdatentypen .....	788
D.9.1	Kreis-Klasse .....	788
D.9.2	Mathematischer Bruch .....	789
D.9.3	Highscore-Liste .....	790
D.9.4	Adressbuch .....	792
D.9.5	Digitaler Bilderrahmen .....	795
D.9.6	Polygonzug .....	797
D.9.7	Flughafen-Check-in .....	799
D.9.8	Socialwall .....	801
D.9.9	Musikalbenanwendung .....	803
D.9.10	Koch-Website .....	806
D.9.11	Suchmaschinenranking .....	808
D.9.12	Partygäste .....	811
D.9.13	Hotelzimmerverwaltung .....	814
D.9.14	Fototagebuch .....	816

D.9.15	Raumbelegung .....	818
D.9.16	Rotationspuzzle .....	821
D.10	Vererbung .....	825
D.10.1	Lampen .....	825
D.10.2	Meeting-Protokoll .....	827
D.10.3	Online-Shop .....	830
D.10.4	Gewässer .....	833
D.10.5	Schere, Stein, Papier .....	836
D.10.6	To-do-Liste .....	838
D.10.7	E-Book .....	841
D.10.8	Zoo .....	845
D.10.9	Audioeffekt-Player .....	848
D.10.10	Meetingplaner .....	851
D.10.11	Tanzwettbewerb .....	854
D.10.12	Fahrtenbuch .....	858
D.10.13	Webseitengenerator .....	860



# Vorwort

Der Messenger auf deinem Smartphone, der Bluetooth-Lautsprecher, mit dem du deine Lieblingsmusik abspielst, der Algorithmus, der auf Netflix die neueste Serie vorschlägt – jemand hat all das programmiert. Fast nichts in der Welt kommt noch ohne Code aus. Du bist gerade auf dem besten Weg, ein Teil davon zu werden und die Zukunft mitzugestalten. Wie aufregend!

Klar, es gibt auch frustrierende Momente – wenn du nicht herausfinden kannst, wo der Fehler in deinem Code liegt, oder wenn du das Kapitel über Objektorientierung in deinem Lehrbuch nicht verstehst. Aber keine Sorge, wir alle haben diese Erfahrung gemacht. Es braucht Zeit und Übung, um wirklich gut im Programmieren zu werden.

Ich selbst habe angefangen, Informatik zu studieren, ohne jemals auch nur eine Zeile Code gesehen zu haben. Und oh boy, war das eine Herausforderung. Am Anfang fiel es mir echt schwer. Aber irgendwas hat mich gepackt, und mit jeder Übungsaufgabe wurde es ein bisschen einfacher. Und je mehr ich gelernt habe, desto mehr habe ich gemerkt, wie viel Spaß programmieren macht. Es ist einfach das beste Gefühl, wenn man den Fehler im Code dann irgendwann doch findet und das Programm endlich genau so funktioniert, wie man es sich vorgestellt hast.

Und irgendwann siehst du dann, was du mit diesem Handwerk alles anstellen und wie kreativ du mit Code werden kannst. In meiner Arbeit als Spieleprogrammiererin kann ich mit nur wenigen Zeilen Code die Steuerung für eine Spielfigur festlegen, eine Unterhaltung zwischen zwei Charakteren auf dem Bildschirm anzeigen oder ein ganzes Level automatisch generieren lassen. Die Möglichkeiten sind fast endlos! Und das Beste: Es gibt selten nur eine richtige Lösung für ein Problem. Das gibt uns beim Programmieren die Chance, unsere eigenen Ideen und Visionen umzusetzen. Und nein, wir sitzen nicht den ganzen Tag allein im Keller vor dem Bildschirm, wie Kriminelle beim Hacken in einem Hollywoodfilm. Wir arbeiten im Team an spannenden Projekten, um gemeinsam ein Ziel zu erreichen.

Mit den Übungsaufgaben in diesem Buch (in dieser Auflage übrigens mit 20 brandneuen Aufgaben) kannst du deine Fähigkeiten im Programmieren Schritt für Schritt ausbauen. Damit auch du dich bald an eigene und komplexere Projekte wagen kannst und mit deinem Code die Zukunft gestaltest!

*Kathrin Radtke* (Spellgarden Games), im Januar 2023

## ■ Neues in der 3. Auflage

Im Vergleich zu den ersten beiden Auflagen hat sich in der 3. Auflage erneut einiges geändert. Wie du dir vorstellen kannst, sind **neue Aufgaben** hinzugekommen. Dieses Mal sind es zwanzig Aufgaben mehr im Vergleich zur 2. Auflage. Wir haben die neuen Aufgaben gekennzeichnet, damit du sie schnell finden kannst. Sowohl im Inhaltsverzeichnis als auch in der Aufgabenüberschrift befindet sich der Hinweis „Neu“, an dem du die neuen Aufgaben direkt erkennen kannst. Ansonsten sind die Aufgaben in den Kapiteln nach unserer subjektiven Einschätzung der Schwierigkeit, des Zeitaufwands und der erforderlichen Kreativität aufsteigend sortiert. Du kannst dich also auch gut von vorne nach hinten durcharbeiten.

Außerdem haben wir **Fehler korrigiert**, die uns von Leserinnen und Lesern zuletzt zur 2. Auflage zurückgemeldet wurden (siehe auch Danksagung). Es versteht sich von selbst, dass wir zudem alles Erforderliche **auf den aktuellen Stand** gebracht haben. Dies betrifft insbesondere alle Kapitel, in denen die dem Buch zugrunde gelegte Programmierumgebung „Processing“ behandelt wird. Auch hier haben wir für die 3. Auflage das Feedback unserer Leserschaft aufgegriffen und genauer beschrieben, wie die Beispiellösungen auch mit einer anderen Programmierumgebung verwendet werden können. Für die Aufgaben, für die die Verwendung außerhalb von Processing ohne größere Änderungen an der Musterlösung möglich ist, haben wir eine Kennzeichnung eingeführt, die dies ersichtlich macht. Das Icon, das an jeder Aufgabe zur visuellen Unterstützung steht, ist dann eingerahmt und mit dem Text „Auch ohne Processing lösbar“ versehen.

Eine weitere wichtige Neuerung der 3. Auflage ist, dass wir uns entschlossen haben, die Beispiellösungen nicht mehr im Buch mit abzudrucken. Der Grund dafür ist, dass wir wertvolles **Papier einsparen** möchten. Wie du es vermutlich schon von Hanser-Büchern und natürlich auch von unseren ersten beiden Auflagen gewohnt bist, hast du viele verschiedene Zugänge zu den Beispiellösungen. Diese sind z. B. im E-Book enthalten oder können als Programmtexte von GitHub oder direkt von Hanser heruntergeladen werden. Also wundere dich nicht, wenn bei den Anhängen C und D die Lösungen, die im E-Book enthalten sind, im gedruckten Buch fehlen. Jetzt weißt Du, warum dies der Fall ist.

Darüber hinaus erhältst du als Käufer:in des gedruckten Buches einen Code, mit dem du das komplette E-Book herunterladen kannst. Diesen Code findest du ganz vorne im Buch unter der Überschrift „E-Book inklusive“.

## ■ Danksagung

Ein Buchprojekt ist harte Arbeit. Ohne die Unterstützung vieler helfender Hände geht es nicht. Wir können uns gar nicht genug bei euch allen bedanken, versuchen es aber dennoch, so gut wir können.

Zeit ist wohl das Kostbarste, was wir haben. Darum bin ich umso dankbarer, dass meine liebe Familie mir diese für derartige und andere Projekte einräumt. Danke, Barbara, Giuliana, Antonio und Fabio.

Danke an Brigitte, Frank, Christian und alle anderen wundervollen Menschen, die mich bei der Arbeit an diesem Buch unterstützt haben. Außerdem danke ich allen Förderinnen und Förderern, besonders meinem damaligen Informatiklehrer „Herr Schepanek“.

Ich danke allen Kolleginnen und Kollegen, die mich im Laufe der Jahre begleitet haben. Besonders möchte ich mich bei Katrin für ihre Geduld und ihren Support sowie bei Juli und Lana für ihre Motivation bedanken.

Gemeinsam möchten wir uns bei Christian Ullenboom bedanken. Er hat das Buch kritisch durchgearbeitet und uns viele wertvolle Anmerkungen dazu gegeben. Gleiches gilt für Dirk Louis, der uns zudem freundlicherweise das Vorwort zur Erstauflage des Buches geschrieben hat. Bei Kathrin Radtke und Patrick Stenzel möchten wir uns für die Vorworte zur dritten bzw. zweiten Auflage bedanken. Dem Hanser Verlag und insbesondere Sylvia Hasselbach möchten wir für die viele Geduld mit uns und die ungebrochene Unterstützung bedanken. Ein derartiger Rückhalt ist unverzichtbar.

Außerdem bedanken wir uns bei Remo Lötscher, Lothar Massing, Lars Mühlbauer, Jens Schönbohm und Lars Wildeshaus, die uns auf kleine Fehler in den vorherigen Auflagen aufmerksam gemacht haben, welche wir in dieser Auflage korrigieren konnten.

Schließlich wollen wir uns an dieser Stelle bei den vielen Freiwilligen da draußen bedanken, die ihre wertvolle Zeit dafür aufwenden, um der Allgemeinheit viele nützliche Dinge kosten- und diskriminierungsfrei zur Verfügung zu stellen. Unser Buch bedient sich viel aus der Public Domain, wofür wir uns gerne durch Benennung der wesentlichen Bausteine bedanken wollen. Zunächst sind hierzu die beiden zugrunde gelegten Programmiersprachen zu nennen. **Java** (<https://www.java.com/de/>) wird von der Oracle Corporation und **Python** (<https://www.python.org/>) von der Python Software Foundation bereitgestellt. Beide gehören aktuell zu den am meisten eingesetzten Sprachen und können auf vielfältigste Weise verwendet werden. Um die Hürden gerade für den (fachfremden) Einstieg in die Programmierung weitestgehend zu eliminieren, stellt die Processing Foundation die gleichnamige Entwicklungsumgebung zur Verfügung (<https://processing.org/>). **Processing** basiert dabei von Hause aus auf Java. Im Laufe der Zeit sind weitere Programmiersprachen hinzugekommen, darunter neben JavaScript auch Python (<https://py.processing.org/>). Wir verwenden im Buch zudem **Piktogramme**, um die Aufgaben durch kleine Icons visuell zu unterstreichen. Iconify.it stellt eine Sammlung von 650 freien Glyphicons bereit, aus der wir uns hierzu bedienen haben. Schließlich verwenden wir in einigen Programmieraufgaben Bilder, die durch das Programm verarbeitet werden sollen. Hier haben wir auf die Public-Domain-Cliparts von clker.com (<https://clker.com/>) zurückgegriffen. Ebenso haben wir das Public-Domain-Bild „squirrel“ von Lola Williams in zwei Aufgaben verwendet.

Last, but not least wollen wir uns **bei dir bedanken**. Wir freuen uns, dass wir dein Interesse geweckt und es schon mal bis in deine Hände geschafft haben. Jetzt bleibt uns nur noch, dir beim Programmierentrainieren viel Erfolg und auch Spaß zu wünschen.

*Luigi Lo Iacono, Stephan Wiefeling und Michael Schneider*

August 2023

## ■ Vorwort zur zweiten Auflage

*„Jede hinreichend fortschrittliche Technologie ist von Magie nicht zu unterscheiden.“*

— Arthur C. Clarke

Wir sind Programmierer. Wir sind Magier. Das MIT ist unser Hogwarts, der Google Campus ist unsere unsichtbare Universität, Cupertino ist unser Narnia. Steve Jobs ist unser David Copperfield, Larry Page und Sergey Brin sind unsere Ehrlich Brother, und Frank Thelen ist immerhin vielleicht noch so was wie unser Vincent Raven. Wir sind Siegfried und Roy, und aus Versehen programmierte Endlosschleifen sind unsere weißen Tig. . . strapazieren wir die Allegorie mal nicht über. Jedenfalls: Wir sind Magier.

Oder wenigstens wirken wir für unser Umfeld so. Der Onkel dritten Grades, der im Atomkraftwerk arbeitet, würde uns selbst dann anrufen, wenn AssetWise mal hakt, weil wir eben Informatiker sind und uns dementsprechend mit allem auskennen, was irgendwie mit Computern zusammenhängt. Dabei ist es auch egal, wie komplex oder unterkomplex die Aufgabe ist. Wir werden angerufen, wenn ein Teilchenbeschleuniger angesteuert werden muss, aber auch, wenn es im Fachgeschäft für Strickzubehör „Woll im Leben“ einer Freundin des Freundes deiner Tante väterlicherseits in der alten Fußgängerzone der Kleinstadt, in der du geboren wurdest, nicht mehr ganz so gut läuft und sie jetzt „mal eben“ einen Shop braucht, um den Globalisierungseffekt besser für sich zu nutzen und das Wollgeschäft im Sturm zu erobern. Gestrickt wird ja wohl überall, und sie ist sogar bereit, dir für die drei Wochen Arbeit noch 100 € in die Hand zu drücken. Dafür müsstest du dann aber auch die nächsten drei Jahre zu jeder Tages- und Nachtzeit für Support zur Verfügung stehen.

Doch wir steuern nicht nur die Stromversorgung und das weltweite Woll-Business. Wir halten Banken am Laufen, das Transportwesen und die Kommunikation, ohne uns läuft gar nichts mehr heutzutage. Wir können Welten erschaffen, und wir können sie auch zerstören, je nachdem, ob wir Harry Potter oder Lord Voldemort sein wollen.

Welchen Weg du einschlagen willst – und jeder, der schon mal programmiert hat und behauptet, niemals auf die dunkle Seite geschaut zu haben, lügt – entscheidest du selbst, und auf dem Weg zu deiner Entscheidung ist dieses Buch dein Hogwarts Express, und du musst nicht mal gegen eine Mauer rennen, um hineinzugelangen. Du hast die erste Seite aufgeschlagen und das Vorwort gelesen und die Richtung, in der du weiterblättest, ist deine rote und deine blaue Pille.

Schlag es wieder zu – dann endet die Geschichte hier, du wachst auf in deinem Bett und glaubst, was immer du glauben möchtest. Blätter weiter, bleib im Wunderland, und das Buch zeigt dir, wie tief der Kaninchenbau geht. Nerd today, boss tomorrow.

*Patrick Stenzel (@rock\_galore), im Januar 2020*

## ■ Vorwort zur ersten Auflage

Nerds sind in. Diese liebenswerten Zeitgenossen mit dem vielen Spezialwissen und den kindlichen Vorlieben für Superhelden werden lange nicht mehr nur komisch beäugt. Im Gegenteil. Sie selbst sind nunmehr Stars in vielen Fernsehserien, und ihr modischer Stil ist allgemein akzeptiert. Diese Entwicklung kommt auch der Programmierung zugute. Lange Zeit galt diese Fertigkeit als ein Gebiet, das den Nerds vorbehalten ist. Dem ist nicht so! Es muss nur der Mut aufgebracht werden, sich damit auseinanderzusetzen. Dann wird schnell klar, was mit der Programmierung alles umgesetzt werden kann. Die Bandbreite ist groß und wird durch aktuelle Trends stetig befeuert. Insbesondere durch die Digitalisierung und Vernetzung vieler Alltagsgegenstände finden sich Softwareprogramme vermehrt jenseits gängiger Anwendungsfälle im betrieblichen Kontext von Unternehmen wieder. Also, keine Scheu und ran ans Programmieren!

Mir selbst bereitet das Programmieren viel Freude. Zudem ist es mir eine Herzensangelegenheit, mein Programmier-Know-how und meine Erfahrung an andere weiterzugeben. Ich weiß aus vielen Schulungen sehr genau, was es für Hürden und Stolpersteine beim Programmierenlernen gibt und wie diesen zu begegnen ist. **Gutem Trainingsmaterial kommt dabei eine zentrale Rolle zu.**

Die Autoren Lo Iacono, Wiefling und Schneider schließen hier eine wichtige Lücke. Sie versorgen dich mit vielen Trainingsaufgaben, die dir helfen werden, die wesentlichen Programmierkonzepte wirklich zu verstehen. Und mehr noch. Du kannst und solltest so lange mit den vielen Aufgaben trainieren, bis der Groschen tatsächlich gefallen ist. Das ist wichtig. Denn erst dann wirst du in der Lage sein, mit dem erlernten Handwerkszeug auch selbstständig Entwicklungsaufgaben bewältigen und lösen zu können. Genau da sollst du hin. Viele Lehrformate gehen hier nicht weit genug. Die falsche Annahme ist dabei häufig, dass ein Beispiel und eine Übungsaufgabe zum Verständnis ausreichen. Weit gefehlt. Es fängt schon damit an, dass nicht jeder mit dem gegebenen Beispiel oder der gestellten Übungsaufgabe etwas anfangen kann. Hier schafft das vorliegende Buch Abhilfe, und es gehört damit in die „Einstieg in die Programmierung“-Ecke deines Bücherregals.

*Dirk Louis*, im Januar 2018



# 2

## Einführung in die Programmierung

### ■ 2.1 Warm-up

Dein Training beginnt in diesem Kapitel mit ersten einfachen Programmen. Dazu musst du wissen, wie der grundlegende Aufbau eines Programms sowie der Aufbau der Anweisungen in einer bestimmten Programmiersprache sind. Letzteres gehört zur sogenannten **Syntax** einer Programmiersprache. So wie z. B. die Syntax einer natürlichen Sprache Prinzipien und Regeln des Wort- und Satzbaus festlegt, so legt die Syntax einer Programmiersprache das Vokabular und den Aufbau von Anweisungen fest.

Für die allerersten Programme, die du entwickeln sollst, genügt zunächst die allereinfachste Struktur überhaupt. Hierbei werden Programme als eine lineare Abfolge von Anweisungen angegeben. Anweisungen verfügen immer über einen Namen und eine Liste von Parametern, die die Anweisung verarbeiten soll. Um den Anweisungsnamen von der Parameterliste unterscheiden zu können, werden die Parameter häufig eingeklammert und dem Anweisungsnamen nachgestellt.

```
nameAnweisung(parameter);
```

Verfügt die Parameterliste über mehrere Einträge, so werden diese mit Komma (,) voneinander getrennt.

```
nameAnweisung(parameter1, parameter2);
```

Parameterlose Anweisungen sind durch ein leeres Klammernpaar gekennzeichnet.

```
nameAnweisung();
```

Um mehrere Anweisungen voneinander unterscheiden zu können, wird dafür ein Trennzeichen in der Syntax einer Programmiersprache festgelegt. In Java ist das das Semikolon (;). Das folgende Beispiel zeigt ein abstraktes Programm, das sich aus sieben Anweisungen zusammensetzt, die in der angegebenen Reihenfolge ausgeführt werden. Die lineare Programmabfolge führt die programmierten Anweisungen zeilenweise von links nach rechts beginnend mit der obersten Zeile aus.

```
Anweisung1(); Anweisung2(); Anweisung3(); Anweisung4(); Anweisung5();  
Anweisung6(); Anweisung7();
```

Durch diese Syntaxregel können die einzelnen Anweisungen separiert werden, unabhängig davon, wie du diese in die Quelltextdatei schreibst. Zur besseren Lesbarkeit empfehlen wir dir aber, dich auf eine Anweisung pro Zeile zu beschränken und die Anweisungen untereinander zu schreiben.

```
Anweisung1();  
Anweisung2();  
Anweisung3();  
Anweisung4();  
Anweisung5();  
Anweisung6();  
Anweisung7();
```

Die Programmiersprache Python legt in ihrer Syntax als Trennzeichen von Anweisungen den Zeilenumbruch fest. Ein Zeilenumbruch kann je nach Betriebssystem aus einem oder zwei Zeichen bestehen ('\n', '\r' oder '\r\n').

In der Programmierliteratur hat sich das "Hello World!"-Programm als einführendes Beispiel zur Darstellung der grundlegenden Syntax eines einfachen Programms in einer bestimmten Programmiersprache etabliert. Das Hello-World-Programm gibt in der Konsole einen einfachen Text aus, nämlich Hello World!. Wir wollen es zur Konkretisierung der einführenden Erläuterungen verwenden.

Java:

```
print("Hallo_Welt!");
```

Python:

```
print("Hallo_Welt!")
```

Die `print()`-Anweisung bekommt einen Parameter übergeben. Dieser enthält den Text, den die Anweisung in der Konsole ausgeben soll. Um den Text eingrenzen zu können, wird dieser von doppelten Anführungszeichen (") eingerahmt.

Die Aufgaben dieses Kapitels drehen sich um derartige Programme. Deine Aufgabe wird es sein, die zur Lösung der Aufgabenstellung benötigten Anweisungen zu identifizieren und diese dann in einer geeigneten Abfolge zu platzieren. Welche Anweisungen eine Programmiersprache im Standardumfang bereitstellt, sind in der Referenzdokumentation aufgeführt. Die Referenz der von Processing bereitgestellten Anweisungen kann im Internet eingesehen werden:

- <https://processing.org/reference/> (Java)
- <http://py.processing.org/reference/> (Python)

Referenzen sind sehr umfangreich. Dies gilt auch für die von Processing. Es kann daher etwas dauern, bist du dich darin zurechtfindest. Für die in diesem Kapitel bereitgestellten Trainingsaufgaben sind insbesondere Funktionen zur Ausgabe von Texten in der Konsole und Funktionen zur Ausgabe elementarer geometrischer Formen im grafischen Ausgabefenster wichtig. Um dir das Auffinden dieser Anweisungen zu erleichtern, führen wir dir in der nachfolgenden Auflistung die relevanten auf.

- Konsolenausgabe
  - [https://processing.org/reference/print\\_.html](https://processing.org/reference/print_.html) (Java)
  - <http://py.processing.org/reference/print.html> (Python)
- Linie
  - [https://processing.org/reference/line\\_.html](https://processing.org/reference/line_.html) (Java)
  - <http://py.processing.org/reference/line.html> (Python)

- Dreieck
  - [https://processing.org/reference/triangle\\_.html](https://processing.org/reference/triangle_.html) (Java)
  - <http://py.processing.org/reference/triangle.html> (Python)
- Rechteck
  - [https://processing.org/reference/rect\\_.html](https://processing.org/reference/rect_.html) (Java)
  - <http://py.processing.org/reference/rect.html> (Python)
- Viereck
  - [https://processing.org/reference/quad\\_.html](https://processing.org/reference/quad_.html) (Java)
  - <http://py.processing.org/reference/quad.html> (Python)
- Ellipse
  - [https://processing.org/reference/ellipse\\_.html](https://processing.org/reference/ellipse_.html) (Java)
  - <http://py.processing.org/reference/ellipse.html> (Python)
- Kreisausschnitt
  - [https://processing.org/reference/arc\\_.html](https://processing.org/reference/arc_.html) (Java)
  - <http://py.processing.org/reference/arc.html> (Python)

Um sich mit der Funktionsweise der Anweisungen vertraut zu machen, empfehlen wir dir, die Beschreibung in der Referenz aufmerksam zu lesen. Dies ist eine wichtige Grundfertigkeit, die zum Programmieren dazugehört.

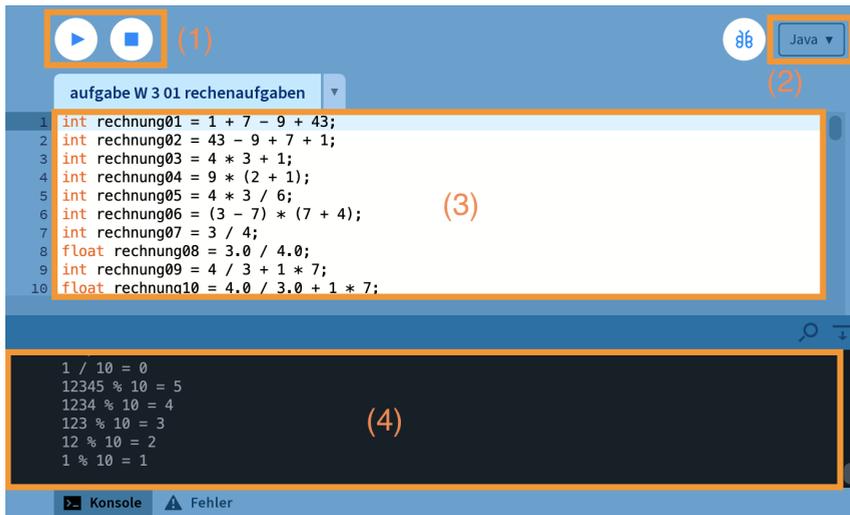
Verwendet werden wir in diesem Buch die Entwicklungsumgebung Processing. Hiermit können wir Programme sowohl in Java als auch in Python schreiben. Processing bietet nicht nur den Vorteil der einfachen Installation auf nahezu allen Betriebssystemen. Wir können damit auch sehr einfach (grafische) Programme auf Basis von Anweisungen schreiben. Aber auch höherwertige Konzepte, wie wir sie in den späteren Kapiteln umsetzen werden, sind in Processing möglich; perfekte Voraussetzungen also zum Trainieren deiner Programmieretechniken mit diesem Buch.

Alle Installationsschritte von Processing findest du in [Abschnitt A.1](#). Wie du an die digitalen Quelltexte unserer Lösungsvorschläge zu einzelnen Aufgaben kommst und wie du sie in Processing öffnest, steht im [Anhang C.1.1](#) für Java und im [Anhang D.1.1](#) für Python.

Dateien mit Quelltext können wir in Processing mit Klick auf *Datei* → *Öffnen* ... laden. In [Bild 2.1](#) haben wir zum Beispiel eine solche Datei geöffnet. Dort können wir gut die grafische Bedienoberfläche von Processing erkennen:

- Mit dem Start- und Stopp-Button (1) kannst du deinen Java- bzw. Python-Code ausführen.
- Um vom Java- auf den Python-Modus zu wechseln, kannst du den Modus-Auswahlreiter (2) verwenden. Wie du den Python-Modus in Processing installierst, steht in [Anhang A.5](#). Links neben diesem Button ist der integrierte Debugger, den du zur Analyse von Java-Code verwenden kannst. Mehr dazu findest du in [Anhang C.1.4](#).
- In der Mitte der Bedienoberfläche (3) steht der eigentliche Quelltext. In diesen Bereich kannst du deinen Java- bzw. Python-Code hineinschreiben.
- Entsprechende Ausgaben in der Konsole findest du im darunterliegenden Bereich (4). Hier werden auch auftretende Fehler im Code angezeigt, sofern es welche gibt.

Nach der Einrichtung von Processing und dem Lesen der Einführung solltest du für dieses Kapitel ausgerüstet sein. In dem Sinne: Viel Spaß bei den ersten Aufgaben, und ran an die Workouts!



**Bild 2.1** So sieht die grafische Bedienoberfläche von Processing aus.

## ■ 2.2 Workout

### W.2.1 Three-Two-One – Mein erstes Programm

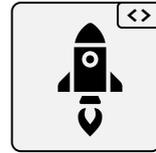
Schwierigkeit



Zeitaufwand



Kreativität



Auch ohne  
Processing lösbar

#### Themen

Mit dieser Aufgabe wollen wir folgende Dinge trainieren:

- Struktur eines einfachen Programms
- Aufbau von Programmanweisungen
- Ausgabe in der Konsole

#### Beschreibung

Wir wollen ein erstes Programm schreiben. Der Klassiker hierfür ist die Ausgabe eines Texts – meist der Text `Hello World` – in der Konsole. Dazu braucht es in der Regel nur eine einzige Anweisung. An dieser kannst du aber bereits den Aufbau von Anweisungen und einfachen Programmen nachvollziehen und trainieren. Los geht's!

#### Aufgabenstellung

Schreibe ein Programm, das den Text `Three-Two-One - Takeoff!` in der Konsole ausgibt. Wenn dein Programm funktioniert, solltest du den angegebenen Text in der Konsole lesen können, so wie nachfolgend exemplarisch zu sehen ist:

```
Three-Two-One - Takeoff!
```

Wenn das geklappt hat, dann mach doch einfach weiter und modifiziere dein erstes Programm nach deinen Wünschen. Ändere z. B. den Text oder füge weitere Anweisungen zur Textausgabe hinzu. Reflektiere dabei, wie dein Programm auf die Änderungen reagiert. Wenn du das Resultat hast kommen sehen, und es ist nichts Unerwartetes bei der Ausführung deines Programms passiert, hast du es im Griff und verstanden, wie Anweisungen und einfache Programme aufgebaut sind.

#### Testfälle

Zum Testen deines Programms brauchst du in diesem Fall noch keine Testdaten. Starte dein Programm und prüfe, ob die geforderte Ausgabe in der Konsole ausgegeben wird.

#### (Algorithmische) Tipps

Wenn du stockst und nicht weiterweißt, dann versuch mal Folgendes:

- Gib nicht auf. Du solltest es so lange probieren, bis es klappt. Das nennt man *Trial and Error* (Versuch und Irrtum). Versuch es weiter! Vermutlich bist du schon nah dran an der Lösung, denn der Fehler liegt sehr häufig im Detail.
- Wir benötigen eine passende Anweisung, die uns die Programmiersprache zur Ausgabe von Daten in der Konsole bereitstellt. Wie lautet diese?
- Anweisungen folgen einem festgelegten Aufbau. Hier schleichen sich schon mal Tippfehler ein. Was sagen denn die Fehlermeldungen, wenn du versuchst, dein Programm zu starten?

## W.2.2 Weihnachtsbaum

Schwierigkeit



Zeitaufwand



Kreativität



Auch ohne  
Processing lösbar

### Themen

Mit dieser Aufgabe wollen wir folgende Dinge trainieren:

- Struktur eines einfachen Programms
- Aufbau und Abfolge von Programmanweisungen
- Ausgabe in der Konsole

### Beschreibung

Wir wollen jetzt ein erstes Muster in die Konsole schreiben. Dafür werden wir bestimmte Zeichen so oft hinter- und untereinander schreiben, bis sich daraus eine Form ergibt. Diese Form des „Malens“ ist bei vielen Konsolenprogrammen üblich und wird auch heute noch verwendet.

### Aufgabenstellung

Schreibe ein Programm, das das folgende Muster in der Konsole ausgibt:

```

*
***
*****
*****
*****
*****
*****
*****
***

```

### Testfälle

Wenn die Tanne wie angegeben in der Konsole ausgegeben wird, dann hast du alles richtig gemacht und diese Aufgaben erfolgreich bearbeitet. Gesetzt den Fall, dass du noch weitere Programme dieses Typs erstellen willst, geben wir dir hier noch weitere Anregungen (du kannst dir aber auch gerne selbst was überlegen!):

```

Sanduhr:  ****          Pizzastück:  *****          Diamant:  **
          ***              * *              * *
          *                * *              * *
          ***              * *              * *
          ****             **              **

```

Für diese zusätzlichen Trainingseinheiten bieten wir dir keine Lösungsvorschläge mehr an. Wir sind fest davon überzeugt, dass du das selbst hinbekommst und unsere Hilfe hierfür nicht mehr benötigst.

### Algorithmische Tipps

Wenn du stockst und nicht weiterweißt, dann versuch mal Folgendes:

- Schau' dir die allererste Aufgabe dieses Buches doch noch einmal an und überlege dir, wie die Ausgabe für jede Zeile von oben nach unten aussehen muss.
- In Processing für Java gibt es zwei Befehle, mit denen du Text in die Konsole schreiben kannst. Der eine fügt eine neue Zeile hinzu, der andere hingegen nicht.
- Das Sternchen- und das Leerzeichen führen zum Ziel!

# 9

## Referenzdatentypen

### ■ 9.1 Warm-up

In allen Übungsaufgaben, mit denen wir bisher in Berührung gekommen sind, haben wir Variablen benötigt, um Werte zu berechnen, zu speichern oder abzurufen. Wir wissen, dass jede Variable einem bestimmten Datentyp zugeordnet wird. Der Datentyp ist eine Menge von Werten und Operationen, die auf diesen Werten definiert sind.

Wenn wir mit Zahlen umgehen, haben wir es mit Ganzzahl- oder Fließkommadatentypen zu tun. Dabei werden die Datentypen in vielen Programmiersprachen noch mal nach ihrem Wertebereich (bei Integer-Werten) bzw. nach ihrer Genauigkeit (bei Fließkommazahlen) in verschiedene Datentypen unterteilt. Dazu kommt dann meistens noch der boolesche Datentyp, der die Werte `true` und `false` verarbeiten kann. Gegebenenfalls gibt es auch noch einen Datentypen für die Repräsentation eines Zeichens, wie zum Beispiel den Datentyp `char` in Java. Bei all diesen Datentypen handelt es sich um **Werttypen**. Je nach verwendeter Programmiersprache werden sie auch als *primitive Datentypen* oder *elementare Datentypen* bezeichnet. Egal, wie sie nun genannt werden, sie haben eine Sache gemein: Sie operieren auf einfachen Zahlen und Zeichen.

Neben diesen elementaren Datentypen existiert aber noch eine zweite Art von Datentypen, zu denen beispielsweise Strings, Arrays oder Klassen gehören. Sie werden als **Referenzdatentypen** oder auch als **zusammengesetzte Datentypen** bezeichnet. Ein Array ist bekanntlich eine endliche Menge von Elementen eines anderen Datentyps, zum Beispiel ein Array von Integer-Werten. Ein String ist eine endliche Menge von Zeichen und definiert beispielsweise Operationen wie Konkatenation, Länge, Teilstring, Vergleich usw. Zuletzt betrachten wir noch den im vorigen Kapitel besprochenen Datentyp der Klasse. Hier definieren wir selbst, aus welchen Attributen und Methoden sie besteht, und somit auch die verwendeten Daten und Datentypen.

Der entscheidende Unterschied liegt darin, wo die Werte eines Datentyps abgelegt werden: entweder auf dem **Stack**, bei dem es sich um einen kleineren Speicherbereich mit sehr performantem Zugriff handelt, oder dem **Heap**, einem wesentlich größeren Speicherbereich, der aber mit weniger Performance arbeitet. Die elementaren Datentypen werden auf dem Stack abgelegt und enthalten direkt die eigentlichen Daten. Demgegenüber werden die Daten von Referenzdatentypen auf dem Heap gespeichert. Wenn wir beispielsweise die Instanz einer Klasse anlegen, wird zunächst Speicher auf dem Heap allokiert. Wie viel Speicher das System reserviert, hängt von den in der Klasse verwendeten Datentypen ab.

Denn für jeden Datentyp ist in einem Programmiersystem festgehalten, wie viel Speicher er benötigt. Insofern ist die Addition der Werte hier richtungsweisend.

Wenn also der Speicher reserviert wurde, erhalten wir in der Variablen die sogenannte **Objektreferenz**. In Programmiersprachen wie C wird hier auch von einem **Zeiger** (engl. *pointer*) gesprochen. Diese Objektreferenz ist also die Startadresse des Speicherbereichs im Heap, ab der die Daten abgelegt sind. Hierbei ist von der **Objektidentität** die Rede, die dann alle notwendigen Daten zu unserem Objekt hält.

Es kann problematisch sein, mit Referenzdatentypen umzugehen. Dann nämlich, wenn mehrere Variablen auf dieselben Adressen verweisen. Hier führt die Änderung von Werten auf jeder Seite zu Änderungen an denselben Daten! Da dies in den seltensten Fällen tatsächlich gewünscht ist, sollten wir die Daten kapseln. Dies erreichen wir dadurch, dass wir eine Kopie der Daten anlegen, um die Unveränderlichkeit der Daten zu gewährleisten. Unter Umständen müssen wir dies auch in Methoden und für Rückgabewerte berücksichtigen!

Und nun – ran an die Workouts und viel Erfolg beim Training!

## ■ 9.2 Workout

### W.9.1 Kreis-Klasse

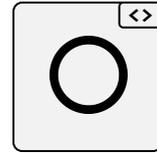
Schwierigkeit



Zeitaufwand



Kreativität



Auch ohne  
Processing lösbar

#### Themen

Mit dieser Aufgabe wollen wir folgende Dinge trainieren:

- Referenzdatentypen

#### Beschreibung

Zur besseren Charakterisierung von Kreisen wollen wir eine Kreis-Klasse schreiben, die alle Informationen zu einem Kreis in einem Objekt bündelt.

Ein Kreis kann z. B. mit den Parametern der Position (x,y) sowie dem Radius (r) beschrieben werden.

#### Aufgabenstellung

Schreibe eine Klasse `Coordinate` mit Konstruktor, welche die Informationen zu den Koordinaten beinhaltet. Schreibe dort die Funktion `toString()`, welche die Koordinaten in der Form `(x, y)` als String zurückgibt.

Schreibe eine Klasse `Circle` mit Konstruktor, welche die Informationen zu einem Kreis enthält (Position, Radius). Schreibe dort die Funktion `area()` zur Rückgabe des Flächeninhaltes sowie die Funktion `toConsole()` zur Ausgabe der Kreisinformationen in der Konsole.

Die Ausgabe in der Konsole könnte zum Beispiel so aussehen:

```
Ich stehe bei (50, 100) und bin 1017.87604809 groß.
```

Prüfe anschließend außerhalb der Klassen die `Circle`-Funktion und erzeuge eine Konsolenausgabe mit Testdaten.

#### Testfälle

- x: 10, y: 43, r: 4
  - Ich stehe bei (10, 43) und bin 50.2654838562 groß.

#### Algorithmische Tipps

Wenn du stockst und nicht weiterweißt, dann versuch mal Folgendes:

- Der Flächeninhalt eines Kreises kann mit folgender Formel berechnet werden:

$$A = \pi * r^2$$

- Vielleicht hilft es, zunächst die `Coordinate`-Klasse zu testen und anschließend erst die `Circle`-Klasse zu programmieren.
- Besonders in Python ist es wichtig, Zahlen zunächst in Strings umzuwandeln. Andernfalls wird bei der Ausgabe in der Konsole ein Fehler auftauchen.

## W.9.2 Mathematischer Bruch

Schwierigkeit



Zeitaufwand



Kreativität



Auch ohne  
Processing lösbar

### Themen

Mit dieser Aufgabe wollen wir folgende Dinge trainieren:

- Referenzdatentypen

### Beschreibung

Zur Darstellung und Berechnung von mathematischen Brüchen wollen wir unseren eigenen Datentyp schreiben. Dieser soll neben der Bruchdarstellung auch Brüche voneinander addieren oder subtrahieren können.

Zur Erinnerung:

- Addition von Brüchen:

$$\frac{a}{b} + \frac{c}{d} = \frac{(a * d) + (c * b)}{b * d}$$

- Multiplikation von Brüchen:

$$\frac{a}{b} * \frac{c}{d} = \frac{a * c}{b * d}$$

### Aufgabenstellung

Schreibe die Klasse `Fraction`, die einen Bruch repräsentiert, und gib einen Konstruktor und Implementierungen für die Methoden `add()` und `multiply()` an. Getter- und Setter-Methoden sowie eine `toString()`-Methode sollten ebenfalls eingebaut werden.

Teste die Klasse mit Beispielrechnungen in der Konsole.

### Testfälle

- $\frac{1}{2} + \frac{1}{4} = \frac{(1 * 4) + (1 * 2)}{2 * 4} = \frac{6}{8}$
- $\frac{1}{2} * \frac{1}{4} = \frac{1 * 1}{2 * 4} = \frac{1}{8}$

### Algorithmische Tipps

Wenn du stockst und nicht weiterweißt, dann versuch mal Folgendes:

- Natürlich kannst und solltest du deinen Datentyp zur Berechnung mit anderen Brüchen weiterverwenden.
- Als Datentyp bieten sich ganze Zahlen (`Integer`) an, da sich in Brüchen keine Kommazahlen befinden dürfen.
- Wenn du Bruch 1 mit Bruch 2 addierst oder multiplizierst, sollte sich weder der Wert von Bruch 1 noch von Bruch 2 verändern.

## W.9.3 Highscore-Liste

Schwierigkeit



Zeitaufwand



Kreativität



Auch ohne  
Processing lösbar

### Themen

Mit dieser Aufgabe wollen wir folgende Dinge trainieren:

- Referenzdatentypen

### Beschreibung

Von klassischen Computerspielen kennen wir vielleicht noch die sogenannte Highscore-Liste. Diese stellen die höchsten in dem entsprechenden Spiel erreichten Punktzahlen in Tabellenform absteigend dar. Eine solche wollen wir mithilfe der folgenden Eigenschaften realisieren:

*„Eine Highscore-Liste für ein Computerspiel verwaltet mehrere Einträge, die jeweils durch einen Spitznamen der spielenden Person repräsentiert sind. Die Highscore-Liste verfügt zudem über gängige Methoden, die das Hinzufügen von Platzierungen und die Ausgabe der gesamten Liste ermöglichen.“*

### Aufgabenstellung

Überlege dir ein UML-Klassendiagramm für die beschriebenen Eigenschaften.

Implementiere die Klasse `HighscoreEntry`, mit der wir Einträge für die Highscore-Liste erzeugen können. Diese Klasse soll eine geeignete `toString()`-Methode beinhalten.

Schreibe die Klasse `HighscoreTable`, die die Highscore-Liste realisiert. Sorge dafür, dass die `HighscoreEntry`-Listeneinträge sinnvoll initialisiert werden. Beachte außerdem, dass neue Listeneinträge immer an der entsprechenden Position in der Highscore-Liste hinzugefügt werden müssen. Bereits vorhandene Einträge müssen gegebenenfalls um eine Position verschoben werden.

### Testfälle

- Befülle die Highscore-Liste zunächst mit durchnummerierten Namen. Zum Beispiel erst mit `Name0`, dann `Name1` usw. Wenn du danach einen neuen Eintrag in der Mitte der Highscore-Liste hinzufügst, darf kein Eintrag in der neuen Liste fehlen.
- Ebenfalls dürfen keine neuen Einträge doppelt in der Liste vorhanden sein. Wenn dies der Fall sein sollte, hast du vermutlich unabsichtlich die gleiche Variablenreferenz verwendet.

### Algorithmische Tipps

Wenn du stockst und nicht weiterweißt, dann versuch mal Folgendes:

- Referenzdatentypen sollten am besten in einer Schleife `Element für Element` kopiert werden. Sonst besteht die Gefahr, dass die gleiche Referenz im neuen Element verwendet wird, also keine richtige Kopie angelegt wurde.
- Lege ein Array für alle Einträge an und speichere darin entsprechend `HighscoreEntry`-Elemente.
- Wenn deine `HighscoreEntry`-Klasse eine `toString()`-Methode hat, kannst du jeden Eintrag mit einfacher Angabe des Variablennamens ausgeben. Das erleichtert die Programmierarbeit in großen Quelltexten.

## W.9.4 Adressbuch

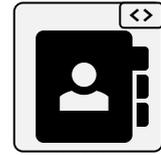
Schwierigkeit



Zeitaufwand



Kreativität



Auch ohne  
Processing lösbar

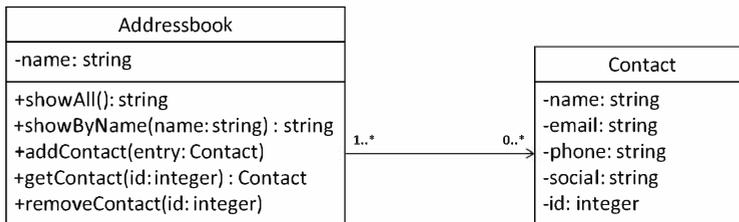
### Themen

Mit dieser Aufgabe wollen wir folgende Dinge trainieren:

- Referenzdatentypen

### Beschreibung

Wir wollen zur Verwaltung unserer Kontakte ein Adressbuch programmieren. Bereits vorgegeben ist dieses UML-Klassendiagramm, welches zur Implementierung entwickelt wurde:



Die Contact-Klasse enthält nur Getter- und Setter-Methoden. Diese sind im Diagramm nicht explizit angegeben.

### Aufgabenstellung

Implementiere das dargestellte UML-Klassendiagramm.

Schreibe ein Programm, das ein Adressbuchverwaltungsprogramm realisiert. Instanziiere dazu mithilfe der folgenden Angaben die entsprechenden Objekte:

#### Adressbuch: Privat

ID	Name	Email	Phone	Social Media
1	Ken Tern	ken.tern@mail.de	+49 221 3982781	@kentern
2	Bill Iger	bill.iger@gmx.de	+49 211 9821348	@billiger
3	Flo Kati	flo.kati@web.de	+49 251 9346441	@flokati
4	Ingeborg Mirwas	inge.mirwas@post.de	+49 228 4663289	@borgmirwas
5	Ann Schweigen	ann.schweigen@gmx.de	+49 231 6740921	@anschweigen
6	Mark Ens Schuh	mark.enschuh@gmail.com	+49 234 4565657	@markenschuh
7	Lee Köhr	lee.koehr@mail.de	+49 561 8976761	@leekoehr
8	Pit Schnass	pit.schnass@post.de	+49 721 4545754	@pitschnass

**Adressbuch: Arbeit**

ID	Name	Email	Phone	Social Media
1	Phil Tertüte	phil.tertuete@company.de	+49 177 1786756	@philtertuete
2	Flo Kati	flo.kati@laden.com	+49 161 2336541	@ibm.kati
3	Andreas Kreuz	andreas.kreuz@bazaar.de	+49 163 3442889	@asbazaar
4	Erkan Alles	erkan.alles@solver.de	+49 171 1442553	@easolver
5	Mark Reelee	mark.reelee@media.de	+49 151 5345612	@mrmedia
6	Roy Bär	roy.baer@media.de	+49 151 5477889	@rbmedia
7	Mario Nette	mario.nette@media.de	+49 151 5113341	@mnmedia
8	Klaus Uhr	klaus.uhr@media.de	+49 151 6743431	@kumedia

Teste anschließend die im UML-Klassendiagramm angegebenen Funktionen und erzeuge entsprechende Testausgaben.

**Hinweis:** Deine Freundin Flo Kati arbeitet zusammen mit dir. Dies bitte beachten, wenn der Kontakt in die Adressbücher „Privat“ und „Arbeit“ eingefügt wird.

**Testfälle**

- Zeige zunächst das komplette Adressbuch an, lösche einen Eintrag in der Mitte und zeige das Adressbuch erneut an. Sind die restlichen Einträge dann immer noch vorhanden?
- Lasse dir den Kontakt „Roy Bär“ mithilfe der Funktion `showByName()` ausgeben. Kommt der korrekte Inhalt zurück?
- Wird der Kontakt mit der ID 1 mit `getContact(1)` korrekt zurückgegeben?

**Algorithmische Tipps**

Wenn du stockst und nicht weiterweißt, dann versuch mal Folgendes:

- Die `Contact`-Klasse könnte eine `toString()`-Methode enthalten. Das würde zumindest den Code übersichtlicher halten.
- Bei den Suchfunktionen könntest du alle Elemente durchgehen und dann prüfen, ob ein Element mit dem gesuchten Wert übereinstimmt. Diesen kannst du dann zurückgeben.
- In ähnlicher Weise könntest du auch bei der Löschmethode herangehen. Unterschied ist hier natürlich, dass der zu löschende Wert einer neuen Liste nicht hinzugefügt wird. Hier gäbe es natürlich auch andere Varianten, das Problem zu lösen.

## W.9.5 Digitaler Bilderrahmen

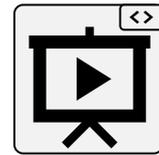
Schwierigkeit



Zeitaufwand



Kreativität



Auch ohne  
Processing lösbar

### Themen

Mit dieser Aufgabe wollen wir folgende Dinge trainieren:

- Referenzdatentypen

### Beschreibung

Digitale Bilderrahmen

Ein Bilderrahmen hat eine feste Anzahl an Speicherplätzen für Bilder. Diese sind beginnend mit 0 durchgehend nummeriert. Bilder werden mit der Klasse `Picture` repräsentiert, diese soll allerdings nur den Namen des Bildes in dieser Aufgabe speichern. Die umzusetzende `DigitalPictureFrame`-Klasse soll folgende Operationen unterstützen:

- Hinzufügen eines Bildes zum Bilderrahmen (`addPicture`). Wenn alle Speicherplätze bereits belegt sind, hat der Methodenaufruf keine Auswirkungen.
- Entfernen eines Bildes aus dem Bilderrahmen (`removePicture`). Welches Bild gelöscht werden soll, wird über die Position im Speicher angegeben. Die nachfolgenden Bilder rücken im Speicher um eine Position auf. Sind alle Speicherplätze leer, hat der Methodenaufruf keine Auswirkungen.
- Auslesen des nächsten Bildes aus dem Bilderrahmen (`getNext`). Beim ersten Aufruf wird bei der Speicherposition 0 gestartet. Wurde das letzte gespeicherte Bild durch diesen Methodenaufruf ausgelesen, beginnt es wieder von vorne bei der Speicherposition 0.
- Auslesen eines zufälligen Bildes (`getNextRandom`). Hierzu kann die Funktion `random()` verwendet werden.

### Aufgabenstellung

Schreibe eine Klasse mit dem Namen `DigitalPictureFrame`, die einen digitalen Bilderrahmen repräsentieren soll, mit Instanzvariablen, geeignetem Konstruktor und den beschriebenen Methoden. Implementiere darüber hinaus die Klasse `Picture`, welche in der anderen Klasse verwendet werden soll.

### Testfälle

- Füge drei Bilder hinzu und lösche das vierte Bild.
- Füge drei Bilder hinzu und lösche Bild 2. Bei Ausgabe von `getNextRandom()` sollten nur Bild 1 und Bild 3 angezeigt werden.

### Algorithmische Tipps

Wenn du stockst und nicht weiterweißt, dann versuch mal Folgendes:

- Wenn du ein Bild hinzufügst, speichere das Bild in aktueller Array-Position und erhöhe diese um 1.
- Sollte die Array-Position über die erlaubte Menge gehen, muss entsprechend im Code reagiert werden. Echte Profis schaffen den notwendigen Code ohne If-Else-Anweisungen. Kleiner Tipp: Modulo-Operator.
- Beim Löschen eines Bildes könntest du ab der Löschposition loszählen und alle folgenden Bilder um eine Stelle nach vorne kopieren.