Vorwort ..... **XVII**

1 Einführung **1**
1.1 Eine Sprache für viele Plattformen 2
1.2 Deshalb ist Kotlin so besonders 3
1.3 Darauf dürfen Sie sich freuen 4

Teil I: Konzeptioneller Aufbau von Computern und Software **7**

2 Komponenten eines Computers **9**
2.1 Beliebige Daten als binäre Zahlen 9
2.2 Wie Zahlen in Texte, Bilder und Animationen umgewandelt werden 12
2.3 Zahlen als ausführbarer Code 13

3 Zugriff auf den Speicher **15**
3.1 Organisation des Speichers 16
3.2 Daten im Speicher und Datenverarbeitung im Prozessor 17
3.3 Heap und Stack 18
3.4 Programme als Code schreiben statt als Zahlenfolgen 18

4 Interpreter und Compiler **21**
4.1 Virtuelle Maschinen, Bytecode und Maschinencode 22
4.2 Kotlin – eine Sprache, viele Plattformen 23

5 Syntax, Semantik und Pragmatik **25**
5.1 Syntax 25
5.2 Semantik 26
5.3 Pragmatik 28

6 Eingabe – Verarbeitung – Ausgabe **31**

Bibliografische Informationen
https://d-nb.info/1302388576

digitalisiert durch DEUTSCHE NATIONAL BIBLIOTHEK

7 Los geht's ... 33
7.1 Integrierte Entwicklungsumgebung ... 34
7.2 Projekt anlegen ... 36

Teil II: Grundlagen des Programmierens ... 39

8 Anweisungen und Ausdrücke ... 41
8.1 Ausdrücke ... 42
8.1.1 Literale ... 43
8.1.2 Operationen ... 44
8.1.3 Variablen und Funktionsaufrufe ... 46
8.2 Evaluation von Ausdrücken ... 47
8.2.1 Evaluieren von Operatoren ... 47
8.2.2 Evaluieren von Funktionen ... 48
8.2.3 Evaluieren von Variablen ... 49
8.3 Zusammenspiel von Werten und Typen ... 50
8.3.1 Typprüfungen durch den Compiler ... 51
8.3.2 Typen als Bausteine ... 52

9 Basis-Datentypen ... 55
9.1 Numerics ... 56
9.2 Characters und Strings ... 60
9.3 Booleans ... 61
9.4 Arrays ... 62
9.5 Unit ... 64
9.6 Any ... 67
9.7 Nothing ... 68
9.8 Zusammenfassung ... 69

10 Variablen ... 71
10.1 Deklaration, Zuweisung und Verwendung ... 72
10.2 Praxisbeispiel ... 75
10.2.1 Relevante Informationen extrahieren ... 75
10.2.2 Das Problem im Code lösen ... 76
10.2.3 Zusammenfassung ... 78

11 Kontrollstrukturen ... 79
11.1 Fallunterscheidungen mit if ... 79
11.1.1 if-Anweisung ... 79
11.1.2 if-Ausdruck ... 81

11.2 Pattern-Matching mit when ... 83
11.2.1 Interpretieren von Werten ... 85
11.2.2 Typüberprüfungen ... 86
11.2.3 Überprüfen von Wertebereichen ... 88
11.2.4 Abbilden von langen if-else-Blöcken ... 90
11.3 Wiederholung von Code mit while-Schleifen ... 92
11.3.1 Zählen, wie oft etwas passiert ... 93
11.3.2 Gameloop ... 94
11.4 Iterieren über Datenstrukturen mit for-Schleifen ... 96
11.4.1 Iteration mit Arrays ... 97
11.4.2 Iteration mit Ranges ... 98
11.4.3 Geht das alles nicht auch mit einer while-Schleife? ... 99
11.5 Zusammenfassung ... 100

12 Funktionen ... 101
12.1 Top-Level- und Member-Functions ... 101
12.2 Funktionsaufrufe (Applikation) ... 102
12.3 Syntax ... 103
12.4 Funktionsdefinition (Deklaration) ... 105
12.5 Funktionen als Abstraktion ... 106
12.6 Scoping ... 108
12.7 Rekursive Funktionen ... 110
12.7.1 Endlose Rekursion ... 110
12.7.2 Terminierende Rekursion ... 110
12.7.3 Rekursion vs. Iteration ... 112
12.7.4 Von Iteration zur Rekursion ... 113
12.8 Shadowing von Variablen ... 114
12.9 Inline-Funktionen ... 115
12.10 Pure Funktionen und Funktionen mit Seiteneffekt ... 116
12.10.1 Das Schlechte an Seiteneffekten ... 117
12.10.2 Ohne kommen wir aber auch nicht aus ... 120
12.10.3 Was denn nun? ... 121
12.11 Die Ideen hinter Funktionaler Programmierung ... 122
12.12 Lambdas ... 123
12.13 Closures ... 126
12.14 Funktionen höherer Ordnung ... 128
12.14.1 Funktionen, die Funktionen als Parameter akzeptieren ... 128
12.14.2 Funktionen, die Funktionen zurückgeben ... 130
12.15 Zusammenfassung ... 137
12.16 Das war's ... 137

Teil III: Objektorientierte Programmierung **139**

13 Was sind Objekte? **141**

14 Klassen **145**

14.1 Eigene Klassen definieren 145

14.2 Konstruktoren 147

14.2.1 Aufgaben des Konstruktors 149

14.2.2 Primärer Konstruktor 149

14.2.3 Parameter im Konstruktor verwenden 150

14.2.4 Initialisierungsblöcke 150

14.2.5 Klassen ohne expliziten Konstruktor 151

14.2.6 Zusätzliche Eigenschaften festlegen 151

14.2.7 Klassen mit sekundären Konstruktoren 152

14.2.8 Default Arguments 153

14.2.9 Named Arguments 154

14.3 Funktionen und Methoden 155

14.3.1 Objekte als Parameter 155

14.3.2 Methoden: Funktionen auf Objekten ausführen 156

14.3.3 Von Funktionen zu Methoden 158

14.4 Datenkapselung 160

14.4.1 Setter und Getter 161

14.4.2 Berechnete Eigenschaften 163

14.4.3 Methoden in Eigenschaften umwandeln 163

14.4.4 Sichtbarkeitsmodifikatoren 165

14.5 Spezielle Klassen 167

14.5.1 Daten-Klassen 167

14.5.2 Enum-Klassen 172

14.5.3 Singuläre Objekte 176

14.5.4 Daten-Objekte 179

14.6 Verschachtelte Klassen 180

14.6.1 Statische Klassen 181

14.6.2 Innere Klassen 182

14.6.3 Lokale innere Klassen 184

14.6.4 Anonyme innere Objekte 184

14.7 Inline-Value-Klassen 185

14.8 Klassen und Objekte sind Abstraktionen 188

14.9 Zusammenfassung 189

15 Movie Maker – Ein Simulationsspiel ... 191
15.1 Überlegungen zur Klassenstruktur ... 192
15.1.1 Eigenschaften und Methoden von Movie ... 193
15.1.2 Eigenschaften und Methoden von Director ... 194
15.1.3 Eigenschaften und Methoden von Actor ... 195
15.1.4 Genre als Enum ... 195
15.1.5 Objektstruktur ... 196
15.2 Von der Skizze zum Programm ... 197
15.2.1 Movie-Maker-Projekt anlegen ... 197
15.2.2 Genre implementieren ... 198
15.2.3 Actor und Director implementieren ... 198
15.2.4 Erfahrungszuwachs bei Fertigstellung eines Films ... 200
15.3 Komplexe Objekte zusammensetzen ... 201
15.3.1 Skills als eine Einheit zusammenfassen ... 201
15.3.2 Begleit-Objekt für Skills ... 202
15.3.3 Objektkomposition und Objektaggregation ... 203
15.3.4 Zusammensetzung der Klasse Movie ... 205
15.3.5 Film produzieren ... 207
15.3.6 Ein Objekt für Spieldaten ... 208
15.3.7 Code zum Projekt ... 209

Teil IV: Vererbung und Polymorphie ... 211

16 Vererbung ... 213
16.1 Vererbungsbeziehung ... 214
16.2 Klassenhierarchien ... 216
16.3 Eigenschaften und Methoden vererben ... 217
16.4 Zusammenfassung ... 219

17 Polymorphie ... 221
17.1 Überschreiben von Methoden ... 222
17.1.1 Eine Methode unterschiedlich überschreiben ... 222
17.1.2 Dynamische Bindung ... 223
17.1.3 Überschreiben eigener Methoden ... 224
17.1.4 Überladen von Methoden ... 226
17.2 Typen und Klassen ... 227
17.2.1 Obertypen und Untertypen ... 228
17.2.2 Generalisierung und Spezialisierung ... 229
17.2.3 Typkompatibilität ... 231
17.2.4 Upcast und Downcast ... 234
17.2.5 Vorsicht bei der Typinferenz ... 235
17.2.6 Smart Casts ... 236

18 Abstrakte Klassen und Schnittstellen ... 237
18.1 Abstrakte Klassen ... 237
18.2 Schnittstellen ... 239
18.2.1 Schnittstellen definieren ... 240
18.2.2 Schnittstellen implementieren ... 240
18.2.3 Schnittstellen für polymorphes Verhalten ... 241
18.2.4 Standardverhalten für Interfaces ... 244
18.2.5 SAM-Interfaces ... 245
18.2.6 Mehrere Interfaces implementieren ... 249
18.3 Alles sind Typen ... 250
18.4 Zusammenfassung ... 252

Teil V: Robustheit ... 253

19 Nullfähigkeit ... 255
19.1 Nullfähige Typen ... 255
19.1.1 Typen nullfähig machen ... 256
19.1.2 Optional ist ein eigener Typ ... 256
19.2 Sicherer Zugriff auf nullfähige Typen ... 257
19.2.1 Überprüfen auf null ... 258
19.2.2 Safe Calls ... 258
19.2.3 Verkettung von Safe Calls ... 259
19.3 Nullfähige Typen auflösen ... 260
19.3.1 Überprüfen mit if-else ... 260
19.3.2 Der Elvis-Operator rockt ... 261
19.3.3 Erzwungenes Auflösen ... 261

20 Exceptions ... 263
20.1 Sowohl Konzept als auch eine Klasse ... 263
20.2 Beispiele für Exceptions ... 264
20.2.1 ArrayIndexOutOfBoundsException ... 264
20.2.2 ArithmeticException ... 265
20.3 Exceptions aus der Java-Bibliothek ... 266
20.4 Exceptions auffangen und behandeln ... 267
20.4.1 Schreiben in eine Datei ... 267
20.4.2 Metapher: Balancieren über ein Drahtseil ... 268
20.5 Exceptions werfen ... 270

20.6 Exceptions umwandeln ... 271
20.6.1 Von Exception zu Optional ... 272
20.6.2 Von Optional zu Exception ... 273
20.6.3 Exceptions vs. Optionals ... 273
20.7 Exceptions weiter werfen ... 274
20.8 Sinn und Zweck von Exceptions ... 277

21 Movie Maker als Konsolenspiel umsetzen ... 279
21.1 Die Gameloop ... 279
21.2 Einen neuen Film produzieren ... 281
21.3 Statistik anzeigen ... 284

22 Entwurfsmuster ... 285
22.1 Das Strategiemuster ... 286
22.1.1 Im Code verstreute Fallunterscheidungen mit when ... 286
22.1.2 Probleme des aktuellen Ansatzes ... 288
22.1.3 Unterschiedliche Strategien für die Ausgabe ... 289
22.1.4 Nutzen der Strategie ... 292
22.2 Das Dekorierermuster ... 292
22.2.1 Probleme des gewählten Ansatzes ... 294
22.2.2 Dekorierer für Komponenten ... 295
22.2.3 Umsetzung des Dekorierers ... 297
22.2.4 Nutzen des Dekorierers ... 299
22.3 Weitere Entwurfsmuster ... 300

23 Debugger ... 301

Teil VI: Datensammlungen und Collections ... 305

24 Überblick ... 307
24.1 Pair und Triple ... 309
24.1.1 Verwendung ... 309
24.1.2 Syntaktischer Zucker ... 310
24.1.3 Destructuring ... 310
24.1.4 Einsatzgebiete ... 310
24.2 Arrays ... 311
24.2.1 Direkter Datenzugriff ... 311
24.2.2 Arrays mit null-Referenzen ... 312
24.2.3 Arrays mit primitiven Daten ... 314
24.2.4 Arrays vs. Listen ... 314

24.3 Listen ... 315
24.3.1 Unveränderliche Listen ... 315
24.3.2 Veränderliche Listen ... 316
24.3.3 List und MutableList sind verwandte Schnittstellen ... 316
24.4 Sets ... 317
24.4.1 Sets verwenden ... 317
24.4.2 Mengen-Operationen ... 318
24.5 Maps ... 319
24.5.1 Maps erzeugen ... 319
24.5.2 Arbeiten mit Maps ... 320
24.5.3 Maps durchlaufen ... 321

25 Funktionen höherer Ordnung für Datensammlungen ... 325
25.1 Unterschiedliche Verarbeitung von Listen ... 325
25.1.1 Imperative Verarbeitung von Listen ... 325
25.1.2 Funktionale Verarbeitung von Listen ... 327
25.1.3 Funktionen als kombinierbare Arbeitsanleitungen ... 328
25.1.4 Aufbau von Funktionen höherer Ordnung am Beispiel von map ... 329
25.2 Hilfreiche Funktionen für Datensammlungen ... 331
25.3 Anwendungsbeispiele für Funktionen höherer Ordnung ... 333
25.4 Sequenzen ... 339
25.4.1 Eager Evaluation – viel zu fleißig ... 340
25.4.2 Lazy Evaluation – Daten bei Bedarf verarbeiten ... 340
25.4.3 Sequenzen verändern die Reihenfolge ... 342
25.4.4 Fleißig oder faul – was ist besser? ... 343

26 Invarianz, Kovarianz und Kontravarianz ... 345
26.1 Typsicherheit durch Typ-Parameter ... 345
26.1.1 Invarianz ... 346
26.1.2 Die Grenzen von Invarianz ... 347
26.1.3 Kovarianz ... 347
26.1.4 Kontravarianz ... 349
26.2 Invarianz, Kovarianz und Kontravarianz im Vergleich ... 351

27 Listen selbst implementieren ... 355
27.1 Was ist eine Liste? ... 355
27.1.1 Unterschiedliche Listen als konkrete Formen ... 356
27.1.2 Eine Schnittstelle für alle möglichen Listen ... 356
27.1.3 Typ-Parameter selbst definieren (Generics) ... 357
27.1.4 Verschiedene Implementierungen derselben Schnittstelle ... 358
27.2 Implementierung der SimpleList durch Delegation ... 359

27.3 Implementierung der SimpleList mit Arrays ... 360
27.3.1 Datenstruktur ... 360
27.3.2 Direkte Abbildung der Listen-Operationen auf ein Array ... 360
27.3.3 Listen-Operationen mit aufwendiger Laufzeit bei Arrays ... 361

28 Verkettete Listen ... 365

28.1 Basisstruktur der verketteten Liste ... 366
28.2 Implementierung der verketteten Liste ... 368
28.3 Umsetzung der Funktionen ... 368
28.3.1 Einfügen am Anfang einer verketteten Liste ... 368
28.3.2 Zugriff auf das erste Element der verketteten Liste ... 370
28.3.3 Zugriff auf das letzte Element der verketteten Liste ... 370
28.3.4 Allgemeines Schema zum Durchlaufen einer verketteten Liste ... 372
28.3.5 Elemente der verketteten Liste zählen ... 372
28.3.6 Zugriff auf das n-te Element ... 373
28.3.7 Die verbleibenden Methoden implementieren ... 373
28.4 Über alle Listenelemente iterieren ... 374
28.4.1 Die Schnittstelle Iterable ... 375
28.4.2 Iterator implementieren ... 375
28.4.3 Iterator verwenden ... 376
28.4.4 Interne Iteration ... 377

29 Testen und Optimieren ... 379

29.1 Korrektheit von Programmen ... 379
29.2 Testfälle in JUnit schreiben ... 380
29.2.1 Assertions ... 381
29.2.2 Implementierung der Liste testen ... 381
29.3 Teste zuerst ... 382
29.4 Klasseninvariante ... 384
29.4.1 Alternative Implementierung von size() für die verkettete Liste ... 384
29.4.2 Gewährleistung eines gültigen Zustands ... 385

30 Optimierung und Laufzeiteffizienz ... 387

30.1 Laufzeit empirisch ermitteln ... 387
30.2 Laufzeit theoretisch einschätzen ... 388
30.3 Die O-Notation ... 389
30.4 Praktische Beispiele für die O-Notation ... 390

31 Unveränderliche verkettete Liste ... 391
31.1 Datenstruktur für die unveränderliche Liste ... 392
31.1.1 Fallunterscheidung durch dynamische Bindung ... 393
31.1.2 Explizite Fallunterscheidung innerhalb der Funktion ... 394
31.1.3 Neue Listen erzeugen statt Liste verändern ... 394
31.1.4 Hilfsfunktionen über Companion-Objekt bereitstellen ... 396
31.2 Rekursive Implementierungen ... 397
31.2.1 map und fold als rekursive Implementierung ... 397
31.2.2 forEach und Endrekursion ... 397

Teil VII: Android ... 399

32 Android Studio ... 401
32.1 Erstellen eines Projekts ... 402
32.2 Aufbau von Android Studio ... 404
32.3 Funktionsweise einer Android-App ... 405
32.3.1 MainActivity ... 406
32.3.2 Context ... 407
32.3.3 Manifest und Gradle-Skripte ... 408
32.4 Projektstruktur einer Android-App ... 408
32.5 Theming ... 409
32.6 Preview ... 411

33 Jetpack Compose ... 413
33.1 Deklarative UI-Entwicklung ... 413
33.2 Composable-Functions ... 416
33.3 Layout ... 418
33.4 State-Management ... 420
33.4.1 MutableState ... 422
33.4.2 Die remember-Funktion ... 423
33.4.3 State-Hoisting ... 425
33.5 Modifier ... 427
33.6 App-Architektur ... 430
33.6.1 UI-Layer ... 430
33.6.2 Data-Layer ... 432
33.6.3 Unidirectional-Data-Flow ... 433
33.6.4 Lokaler State ... 434
33.6.5 Observable-Types ... 435
33.7 Composition und Recomposition ... 435
33.7.1 Composition-Phase ... 436

33.7.2 Layout-Phase ... 436
33.7.3 Drawing-Phase ... 438
33.8 Persistenz ... 439

34 Entwicklung der Movie-Maker-App ... 443
34.1 Setup ... 444
34.2 ViewModel und DataStore ... 445
34.3 Start-Screen ... 446
34.3.1 Scaffold ... 447
34.3.2 Budget-Screen ... 448
34.3.3 Top-Bar ... 450
34.4 Produce-Movie-Screen ... 451
34.4.1 TitleTextfield ... 452
34.4.2 Actor-Pager ... 454
34.4.3 Budget-Slider ... 459
34.4.4 State-Hoisting ... 461
34.4.5 Produce-Movie-Button ... 462
34.5 Movie-Produced-Screen ... 464
34.6 Movie-Production-Error-Screen ... 469
34.7 Navigation ... 470

Teil VIII: Nebenläufigkeit ... 475

35 Grundlagen ... 477
35.1 Threads ... 481
35.1.1 Nicht-determinierter Ablauf ... 482
35.1.2 Schwergewichtige Threads ... 483
35.2 Koroutinen (Coroutines) ... 483
35.2.1 Koroutine vs. Subroutine ... 484
35.2.2 Coroutines vs. Threads ... 485
35.3 Zusammenfassung der Konzepte ... 487

36 Coroutines verwenden ... 489
36.1 Nebenläufige Begrüßung ... 490
36.1.1 Koroutine im Global Scope starten ... 490
36.1.2 Mehrere Koroutinen nebenläufig starten ... 491
36.1.3 Künstliche Wartezeit einbauen mit sleep ... 492
36.1.4 Informationen über den aktuellen Thread ... 493
36.2 Blockieren und Unterbrechen ... 493
36.2.1 Mehrere Koroutinen innerhalb von runBlocking starten ... 494
36.2.2 Zusammenspiel von Threads ... 496

36.3 Arbeit auf Threads verteilen ... 496
36.4 Jobs ... 499
36.5 Nebenläufigkeit auf dem main-Thread ... 500
36.5.1 Zusammenspiel von blockierenden und unterbrechenden Abschnitten 501
36.5.2 Abwechselnde Ausführung ... 502
36.6 Strukturierte Nebenläufigkeit mit Coroutine Scopes ... 503
36.7 runBlocking für main ... 505
36.8 Suspending Functions ... 506
36.8.1 Unterbrechen und Fortsetzen – Behind the scenes ... 506
36.8.2 Eigene Suspending Functions schreiben ... 506
36.8.3 Async ... 508
36.8.4 Strukturierte Nebenläufigkeit mit Async ... 509
36.8.5 Auslagern langläufiger Berechnungen ... 510
36.9 Dispatcher ... 511
36.9.1 Dispatcher festlegen ... 511
36.9.2 Wichtige Dispatcher für Android ... 512

37 Wettlaufbedingungen ... 515
37.1 Beispiel: Bankkonto ... 515
37.1.1 Auftreten einer Wettlaufbedingung ... 517
37.1.2 Unplanbare Wechsel zwischen Threads ... 517
37.2 Vermeidung von Wettlaufbedingungen ... 518
37.2.1 Threadsichere Datenstrukturen ... 518
37.2.2 Thread-Confinement ... 519
37.2.3 Kritische Abschnitte ... 522

38 Deadlocks ... 525

39 Aktoren ... 529

40 Da geht noch mehr ... 533
40.1 Infix-Notation ... 533
40.2 Operatoren überladen ... 534
40.3 Scope-Funktionen ... 536
40.3.1 apply-Funktion ... 536
40.3.2 let-Funktion ... 537
40.3.3 also-Funktion ... 538
40.3.4 Unterschiede der Scope-Funktionen ... 538
40.3.5 with-Funktion ... 539
40.4 Extension Functions ... 539
40.5 Weitere Informationsquellen ... 540

Stichwortverzeichnis ... 543