Vorwort ..... **XVII**

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

Teil I: Konzeptioneller Aufbau von Computern und Software **5**

2 Komponenten eines Computers **7**
2.1 Beliebige Daten als binäre Zahlen 7
2.2 Wie Zahlen in Texte, Bilder und Animationen umgewandelt werden 10
2.3 Zahlen als ausführbarer Code 11

3 Zugriff auf den Speicher **13**
3.1 Organisation des Speichers 14
3.2 Daten im Speicher und Datenverarbeitung im Prozessor 15
3.3 Heap und Stack 16
3.4 Programme als Code schreiben statt als Zahlenfolgen 16

4 Interpreter und Compiler **19**
4.1 Virtuelle Maschinen, Bytecode und Maschinencode 20
4.2 Kotlin – eine Sprache, viele Plattformen 21

5 Syntax, Semantik und Pragmatik **23**
5.1 Syntax 23
5.2 Semantik 24
5.3 Pragmatik 26

Bibliografische Informationen
http://d-nb.info/121585885X

digitalisiert durch DEUTSCHE NATIONAL BIBLIOTHEK

6 Eingabe – Verarbeitung – Ausgabe 29

7 Los geht's 31

7.1 Integrierte Entwicklungsumgebungen 32

7.2 Projekt anlegen 34

Teil II: Grundlagen des Programmierens 37

8 Anweisungen und Ausdrücke 39

8.1 Ausdrücke 40

8.1.1 Literale 41

8.1.2 Operationen 42

8.1.3 Variablen und Funktionsaufrufe 44

8.2 Evaluation von Ausdrücken 45

8.2.1 Evaluieren von Operatoren 45

8.2.2 Evaluieren von Funktionen 46

8.2.3 Evaluieren von Variablen 47

8.3 Zusammenspiel von Werten und Typen 48

8.3.1 Typprüfungen durch den Compiler 49

8.3.2 Typen als Bausteine 50

9 Basis-Datentypen 53

9.1 Numerics 54

9.2 Characters und Strings 58

9.3 Booleans 59

9.4 Arrays 60

9.5 Unit 62

9.6 Any 65

9.7 Nothing 66

9.8 Zusammenfassung 67

10 Variablen 69

10.1 Deklaration, Zuweisung und Verwendung 70

10.2 Praxisbeispiel 73

10.2.1 Relevante Informationen extrahieren 73

10.2.2 Das Problem im Code lösen 74

11 Kontrollstrukturen **77**
11.1 Fallunterscheidungen mit if 77
11.1.1 if-Anweisung 77
11.1.2 if-Ausdruck 79
11.2 Pattern-Matching mit when 81
11.2.1 Interpretieren von Werten 83
11.2.2 Typüberprüfungen 84
11.2.3 Überprüfen von Wertebereichen 86
11.2.4 Abbilden von langen if-else-Blöcken 88
11.3 Wiederholung von Code mit while-Schleifen 90
11.3.1 Zählen, wie oft etwas passiert 92
11.3.2 Gameloop 93
11.4 Iterieren über Datenstrukturen mit for-Schleifen 94
11.4.1 Iteration mit Arrays 95
11.4.2 Iteration mit Ranges 96
11.4.3 Geht das alles nicht auch mit einer while-Schleife? 97

12 Funktionen **99**
12.1 Top-Level- und Member-Functions 99
12.2 Funktionsaufrufe (Applikation) 100
12.3 Syntax 101
12.4 Funktionsdefinition (Deklaration) 103
12.5 Funktionen als Abstraktion 105
12.6 Scoping 106
12.7 Rekursive Funktionen 108
12.7.1 Endlose Rekursion 108
12.7.2 Terminierende Rekursion 109
12.7.3 Rekursion vs. Iteration 110
12.7.4 Von Iteration zur Rekursion 111
12.8 Shadowing von Variablen 112
12.9 Pure Funktionen und Funktionen mit Seiteneffekt 113
12.9.1 Das Schlechte an Seiteneffekten 114
12.9.2 Ohne kommen wir aber auch nicht aus 117
12.9.3 Was denn nun? 118
12.10 Die Ideen hinter Funktionaler Programmierung 119
12.11 Lambdas 120
12.12 Closures 123

12.13 Funktionen höherer Ordnung 124
12.13.1 Funktionen, die Funktionen als Parameter akzeptieren 125
12.13.2 Funktionen, die Funktionen zurückgeben 127
12.14 Zusammenfassung 134
12.15 Das war's 134

Teil III: Objektorientierte Programmierung 135

13 Was sind Objekte? 137

14 Klassen 141
14.1 Eigene Klassen definieren 141
14.2 Konstruktoren 143
14.2.1 Aufgaben des Konstruktors 145
14.2.2 Primärer Konstruktor 145
14.2.3 Parameter im Konstruktor verwenden 146
14.2.4 Initialisierungsblöcke 146
14.2.5 Klassen ohne expliziten Konstruktor 147
14.2.6 Zusätzliche Eigenschaften festlegen 147
14.2.7 Klassen mit sekundären Konstruktoren 148
14.2.8 Default Arguments 149
14.2.9 Named Arguments 150
14.3 Funktionen und Methoden 151
14.3.1 Objekte als Parameter 151
14.3.2 Methoden: Funktionen auf Objekten ausführen 152
14.3.3 Von Funktionen zu Methoden 154
14.4 Datenkapselung 156
14.4.1 Setter und Getter 157
14.4.2 Berechnete Eigenschaften 159
14.4.3 Methoden in Eigenschaften umwandeln 160
14.4.4 Sichtbarkeitsmodifikatoren 161
14.5 Spezielle Klassen 163
14.5.1 Daten-Klassen 163
14.5.2 Enum-Klassen 165
14.5.3 Singuläre Objekte 170
14.6 Verschachtelte Klassen 173
14.6.1 Statische Klassen 174
14.6.2 Lokale Klassen 175

14.6.3 Innere Klassen ... 176
14.6.4 Anonyme innere Objekte ... 178
14.7 Klassen und Objekte sind Abstraktionen ... 178

15 Movie Maker – Ein Simulationsspiel ... 181
15.1 Überlegungen zur Klassenstruktur ... 182
15.1.1 Eigenschaften und Methoden von Movie ... 183
15.1.2 Eigenschaften und Methoden von Director ... 184
15.1.3 Eigenschaften und Methoden von Actor ... 185
15.1.4 Genre als Enum ... 185
15.1.5 Objektstruktur ... 186
15.2 Von der Skizze zum Programm ... 187
15.2.1 Movie-Maker-Projekt anlegen ... 187
15.2.2 Genre implementieren ... 188
15.2.3 Actor und Director implementieren ... 188
15.2.4 Erfahrungszuwachs bei Fertigstellung eines Films ... 190
15.3 Komplexe Objekte zusammensetzen ... 191
15.3.1 Skills als eine Einheit zusammenfassen ... 191
15.3.2 Begleit-Objekt für Skills ... 192
15.3.3 Objektkomposition und Objektaggregation ... 193
15.3.4 Zusammensetzung der Klasse Movie ... 195
15.3.5 Film produzieren ... 197
15.3.6 Ein Objekt für Spieldaten ... 198

Teil IV: Vererbung und Polymorphie ... 201

16 Vererbung ... 203
16.1 Vererbungsbeziehung ... 204
16.2 Klassenhierarchien ... 206
16.3 Eigenschaften und Methoden vererben ... 207

17 Polymorphie ... 211
17.1 Überschreiben von Methoden ... 212
17.1.1 Eine Methode unterschiedlich überschreiben ... 212
17.1.2 Dynamische Bindung ... 213
17.1.3 Überschreiben eigener Methoden ... 214
17.1.4 Überladen von Methoden ... 216
17.2 Typen und Klassen ... 217
17.2.1 Obertypen und Untertypen ... 218

17.2.2 Generalisierung und Spezialisierung ... 219
17.2.3 Typkompatibilität ... 221
17.2.4 Upcast und Downcast ... 224
17.2.5 Vorsicht bei der Typinferenz ... 225
17.2.6 Smart Casts ... 226

18 Abstrakte Klassen und Schnittstellen ... 227
18.1 Abstrakte Klassen ... 227
18.2 Schnittstellen ... 229
18.2.1 Schnittstellen definieren ... 230
18.2.2 Schnittstellen implementieren ... 230
18.2.3 Schnittstellen für polymorphes Verhalten ... 231
18.2.4 Standardverhalten für Interfaces ... 234
18.2.5 Mehrere Interfaces implementieren ... 235
18.3 Alles sind Typen ... 236

Teil V: Robustheit ... 239

19 Nullfähigkeit ... 241
19.1 Nullfähige Typen ... 241
19.1.1 Typen nullfähig machen ... 242
19.1.2 Optional ist ein eigener Typ ... 242
19.2 Sicherer Zugriff auf nullfähige Typen ... 243
19.2.1 Überprüfen auf null ... 244
19.2.2 Safe Calls ... 244
19.2.3 Verkettung von Safe Calls ... 245
19.3 Nullfähige Typen auflösen ... 246
19.3.1 Überprüfen mit if-else ... 246
19.3.2 Der Elvis-Operator rockt ... 247
19.3.3 Erzwungenes Auflösen ... 247

20 Exceptions ... 249
20.1 Sowohl Konzept als auch eine Klasse ... 249
20.2 Beispiele für Exceptions ... 250
20.2.1 ArrayIndexOutOfBoundsException ... 250
20.2.2 ArithmeticException ... 251
20.3 Exceptions aus der Java-Bibliothek ... 252
20.4 Exceptions auffangen und behandeln ... 253
20.4.1 Schreiben in eine Datei ... 253
20.4.2 Metapher: Balancieren über ein Drahtseil ... 254

20.5 Exceptions werfen ... 256
20.6 Exceptions umwandeln ... 257
20.6.1 Von Exception zu Optional ... 258
20.6.2 Von Optional zu Exception ... 259
20.6.3 Exceptions vs. Optionals ... 259
20.7 Exceptions weiter werfen ... 260
20.8 Sinn und Zweck von Exceptions ... 263

21 Movie Maker als Konsolenspiel umsetzen ... 265
21.1 Die Gameloop ... 265
21.2 Einen neuen Film produzieren ... 267
21.3 Statistik anzeigen ... 270

22 Entwurfsmuster ... 271
22.1 Das Strategiemuster ... 272
22.1.1 Im Code verstreute Fallunterscheidungen mit when ... 272
22.1.2 Probleme des aktuellen Ansatzes ... 274
22.1.3 Unterschiedliche Strategien für die Ausgabe ... 275
22.1.4 Nutzen der Strategie ... 278
22.2 Das Dekorierermuster ... 278
22.2.1 Probleme des gewählten Ansatzes ... 280
22.2.2 Dekorierer für Komponenten ... 281
22.2.3 Umsetzung des Dekorierers ... 283
22.2.4 Nutzen des Dekorierers ... 285
22.3 Weitere Entwurfsmuster ... 286

23 Debugger ... 287

Teil VI: Datensammlungen und Collections ... 291

24 Überblick ... 293
24.1 Pair und Triple ... 295
24.1.1 Verwendung ... 295
24.1.2 Syntaktischer Zucker ... 296
24.1.3 Destructuring ... 296
24.1.4 Einsatzgebiete ... 296
24.2 Arrays ... 297
24.2.1 Direkter Datenzugriff ... 297
24.2.2 Arrays mit null-Referenzen ... 298

24.2.3 Arrays mit primitiven Daten 300
24.2.4 Arrays vs. Listen 300
24.3 Listen 301
24.3.1 Unveränderliche Listen 301
24.3.2 Veränderliche Listen 302
24.3.3 List und MutableList sind verwandte Schnittstellen 302
24.4 Sets 303
24.4.1 Sets verwenden 303
24.4.2 Mengen-Operationen 304
24.5 Maps 305
24.5.1 Maps erzeugen 305
24.5.2 Arbeiten mit Maps 306
24.5.3 Maps durchlaufen 307

25 Funktionen höherer Ordnung für Datensammlungen 311
25.1 Unterschiedliche Verarbeitung von Listen 311
25.1.1 Imperative Verarbeitung von Listen 311
25.1.2 Funktionale Verarbeitung von Listen 313
25.1.3 Funktionen als kombinierbare Arbeitsanleitungen 314
25.1.4 Aufbau von Funktionen höherer Ordnung am Beispiel von map 315
25.2 Hilfreiche Funktionen für Datensammlungen 317
25.3 Anwendungsbeispiele für Funktionen höherer Ordnung 319
25.4 Sequenzen 326
25.4.1 Eager Evaluation – viel zu fleißig 326
25.4.2 Lazy Evaluation – Daten bei Bedarf verarbeiten 326
25.4.3 Sequenzen verändern die Reihenfolge 328
25.4.4 Fleißig oder faul – was ist besser? 330

26 Invarianz, Kovarianz und Kontravarianz 331
26.1 Typsicherheit durch Typ-Parameter 331
26.1.1 Invarianz 332
26.1.2 Die Grenzen von Invarianz 333
26.1.3 Kovarianz 333
26.1.4 Kontravarianz 335
26.2 Invarianz, Kovarianz und Kontravarianz im Vergleich 337

27 Listen selbst implementieren 341
27.1 Was ist eine Liste? 341
27.1.1 Unterschiedliche Listen als konkrete Formen 342
27.1.2 Eine Schnittstelle für alle möglichen Listen 342
27.1.3 Typ-Parameter selbst definieren (Generics) 343
27.1.4 Verschiedene Implementierungen derselben Schnittstelle 344
27.2 Implementierung der SimpleList durch Delegation 345
27.3 Implementierung der SimpleList mit Arrays 346
27.3.1 Datenstruktur 346
27.3.2 Direkte Abbildung der Listen-Operationen auf ein Array 346
27.3.3 Listen-Operationen mit aufwendiger Laufzeit bei Arrays 347

28 Verkettete Listen 351
28.1 Basisstruktur der verketteten Liste 352
28.2 Implementierung der verketteten Liste 354
28.3 Umsetzung der Funktionen 354
28.3.1 Einfügen am Anfang einer verketteten Liste 354
28.3.2 Zugriff auf das erste Element der verketteten Liste 356
28.3.3 Zugriff auf das letzte Element der verketteten Liste 356
28.3.4 Allgemeines Schema zum Durchlaufen einer verketteten Liste 358
28.3.5 Elemente der verketteten Liste zählen 358
28.3.6 Zugriff auf das n-te Element 359
28.4 Über alle Listenelemente iterieren 360
28.4.1 Die Schnittstelle Iterable 361
28.4.2 Iterator implementieren 361
28.4.3 Iterator verwenden 362
28.4.4 Interne Iteration 363

29 Testen und Optimieren 365
29.1 Korrektheit von Programmen 365
29.2 Testfälle in JUnit schreiben 366
29.2.1 Assertions 367
29.2.2 Implementierung der Liste testen 367
29.3 Teste zuerst 368
29.4 Klasseninvariante 370
29.4.1 Alternative Implementierung von size() für die verkettete Liste 370
29.4.2 Gewährleistung eines gültigen Zustands 371

30 Optimierung und Laufzeiteffizienz ... 373
30.1 Laufzeit empirisch ermitteln ... 373
30.2 Laufzeit theoretisch einschätzen ... 374
30.3 Die O-Notation ... 375
30.4 Praktische Beispiele für die O-Notation ... 376

31 Unveränderliche verkettete Liste ... 377
31.1 Datenstruktur für die unveränderliche Liste ... 378
31.1.1 Fallunterscheidung durch dynamische Bindung ... 378
31.1.2 Explizite Fallunterscheidung innerhalb der Funktion ... 379
31.1.3 Neue Listen erzeugen statt Liste verändern ... 379
31.1.4 Hilfsfunktionen über Companion-Objekt bereitstellen ... 381
31.2 Rekursive Implementierungen ... 382
31.2.1 map und fold als rekursive Implementierung ... 382
31.2.2 forEach und Endrekursion ... 383

Teil VII: Android ... 385

32 Grundlagen ... 387
32.1 Erstellen eines Projekts ... 388
32.2 Aufbau von Android Studio ... 391
32.3 Funktionsweise einer Android-App ... 392
32.4 Projektstruktur einer Android-App ... 397

33 Entwicklung einer Android-App ... 399
33.1 Integration der Daten ... 399
33.2 StartActivity erstellen (manuell) ... 400
33.2.1 Anlegen des Layouts ... 400
33.2.2 Erstellen der Activity ... 409
33.2.3 Registrieren der Activity ... 410
33.3 Persistenz ... 411
33.3.1 Definition der Schnittstelle ... 412
33.3.2 Implementierung mit Shared Preferences ... 413
33.3.3 Zentrales Instanziieren mit Extension Functions ... 415
33.3.4 Zugriff auf die Datenbankinstanz ... 415
33.4 CreateMovieActivity erstellen (automatisiert) ... 416
33.4.1 Erstellen der Activity ... 416
33.4.2 Starten der Activity ... 417
33.4.3 Implementieren des Layouts ... 417
33.4.4 Implementieren der Activity ... 423

33.5 Ergebnisdialog 429
33.5.1 Erstellen des Layouts 429
33.5.2 Erstellen und Anzeigen des Dialogs 431
33.6 Letzter Feinschliff 432

Teil VIII: Nebenläufigkeit 435

34 Grundlagen 437
34.1 Threads 441
34.1.1 Nicht-determinierter Ablauf 442
34.1.2 Schwergewichtige Threads 443
34.2 Koroutinen (Coroutines) 443
34.2.1 Koroutine vs. Subroutine 444
34.2.2 Coroutines vs. Threads 445
34.3 Zusammenfassung der Konzepte 447

35 Coroutines verwenden 449
35.1 Nebenläufige Begrüßung 450
35.1.1 Koroutine im Global Scope starten 450
35.1.2 Mehrere Koroutinen nebenläufig starten 451
35.1.3 Künstliche Wartezeit einbauen mit sleep 452
35.1.4 Informationen über den aktuellen Thread 453
35.2 Blockieren und Unterbrechen 453
35.2.1 Mehrere Koroutinen innerhalb von runBlocking starten 454
35.2.2 Zusammenspiel von Threads 456
35.3 Arbeit auf Threads verteilen 456
35.4 Jobs 459
35.5 Nebenläufigkeit auf dem main-Thread 460
35.5.1 Zusammenspiel von blockierenden und unterbrechenden Abschnitten 461
35.5.2 Abwechselnde Ausführung 462
35.6 Strukturierte Nebenläufigkeit mit Coroutine Scopes 463
35.7 runBlocking für main 465
35.8 Suspending Functions 466
35.8.1 Unterbrechen und Fortsetzen – Behind the scenes 466
35.8.2 Eigene Suspending Functions schreiben 466
35.8.3 Async 468
35.8.4 Strukturierte Nebenläufigkeit mit Async 470
35.8.5 Auslagern langläufiger Berechnungen 470

35.9 Dispatcher 471
35.9.1 Dispatcher festlegen 471
35.9.2 Wichtige Dispatcher für Android 473

36 Wettlaufbedingungen 475
36.1 Beispiel: Bankkonto 475
36.1.1 Auftreten einer Wettlaufbedingung 477
36.1.2 Unplanbare Wechsel zwischen Threads 477
36.2 Vermeidung von Wettlaufbedingungen 478
36.2.1 Threadsichere Datenstrukturen 478
36.2.2 Thread-Confinement 479
36.2.3 Kritische Abschnitte 482

37 Deadlocks 485

38 Aktoren 489

39 Da geht noch mehr 493
39.1 Infix-Notation 493
39.2 Operatoren überladen 494
39.3 Scope-Funktionen 496
39.3.1 apply-Funktion 496
39.3.2 let-Funktion 497
39.3.3 also-Funktion 498
39.3.4 Unterschiede der Scope-Funktionen 498
39.3.5 with-Funktion 499
39.4 Extension Functions 499
39.5 Weitere Informationsquellen 500

Stichwortverzeichnis 503