

# Inhaltsverzeichnis

	<b>Einleitung</b> . . . . .	15
	Vorkenntnisse . . . . .	15
	Aufbau des Buches . . . . .	16
	Benötigte Software . . . . .	20
	Downloads . . . . .	21
	<b>Autorin</b> . . . . .	22
<b>1</b>	<b>Klassendefinition und Objektinstanziierung</b> . . . . .	<b>23</b>
1.1	Klassen und Objekte . . . . .	23
	☆ Aufgabe 1.1: Definition einer Klasse . . . . .	27
	☆ Aufgabe 1.2: Objekt (Instanz) einer Klasse erzeugen. . . . .	27
1.2	Das Überladen von Methoden . . . . .	27
	☆ Aufgabe 1.3: Eine Methode überladen . . . . .	28
1.3	Die Datenkapselung, ein Prinzip der objektorientierten Programmierung . . . . .	28
	☆ Aufgabe 1.4: Zugriffsmethoden . . . . .	29
1.4	Das »aktuelle Objekt« und die »this-Referenz« . . . . .	29
	☆☆ Aufgabe 1.5: Konstruktordefinitionen . . . . .	29
1.5	Die Wert- und Referenzübergabe in Methodenaufrufen . . . . .	30
	☆☆☆ Aufgabe 1.6: Wertübergabe in Methoden (»call by value«) . . . . .	30
1.6	Globale und lokale Referenzen. . . . .	31
	☆☆ Aufgabe 1.7: Der Umgang mit Referenzen . . . . .	31
1.7	Java-Pakete . . . . .	31
	☆ Aufgabe 1.8: Die package-Anweisung . . . . .	32
	☆ Aufgabe 1.9: Die import-Anweisung . . . . .	33
1.8	Die Modifikatoren für Felder und Methoden in Zusammenhang mit der Definition von Paketen . . . . .	33
	☆☆ Aufgabe 1.10: Pakete und die Sichtbarkeit von Mitgliedern einer Klasse . . . . .	34
1.9	Standard-Klassen von Java . . . . .	34
1.10	Die Wrapper-Klassen von Java und das Auto(un)boxing . . . . .	36
	☆☆ Aufgabe 1.11: Das Auto(un)boxing . . . . .	38
1.11	Das Paket java.lang.reflect. . . . .	38
1.12	Arrays (Reihungen) und die Klassen Array und Arrays . . . . .	40
	☆☆ Aufgabe 1.12: Der Umgang mit Array-Objekten . . . . .	41

1.13	Zeichenketten und die Klasse String . . . . .	41
	☆ Aufgabe 1.13: Der Umgang mit String-Objekten . . . . .	50
	☆☆ Aufgabe 1.14: Der Umgang mit Textblöcken . . . . .	51
1.14	Sprachmerkmale, die in den weiteren Beispielen genutzt werden . . . . .	52
	☆ Aufgabe 1.15: Methoden mit variablen Argumentenlisten . . . . .	53
1.15	Den Zugriff auf Klassen und Felder minimieren: Unveränderliche (immutable) Klassen und Objekte . . . . .	53
1.16	Die alte und neue Date&Time-API als Beispiel für veränderliche und unveränderliche Klassen . . . . .	55
	☆☆☆ Aufgabe 1.16: Die Methoden von Date/Time- Klassen . . . . .	59
1.17	Private Konstruktoren . . . . .	60
	☆ Aufgabe 1.17: Objekte mithilfe eines privaten Konstruktors erzeugen . . . . .	60
1.18	Lösungen . . . . .	61
	Lösung 1.1 . . . . .	61
	Lösung 1.2 . . . . .	61
	Lösung 1.3 . . . . .	62
	Lösung 1.4 . . . . .	63
	Lösung 1.5 . . . . .	64
	Lösung 1.6 . . . . .	66
	Lösung 1.7 . . . . .	68
	Lösung 1.8 . . . . .	71
	Lösung 1.9 . . . . .	71
	Lösung 1.10 . . . . .	72
	Lösung 1.11 . . . . .	73
	Lösung 1.12 . . . . .	76
	Lösung 1.13 . . . . .	79
	Lösung 1.14 . . . . .	82
	Lösung 1.15 . . . . .	91
	Lösung 1.16 . . . . .	93
	Lösung 1.17 . . . . .	100
2	<b>Abgeleitete Klassen und Vererbung . . . . .</b>	<b>103</b>
2.1	Abgeleitete Klassen . . . . .	103
2.2	Die Konstruktoren von abgeleiteten Klassen . . . . .	103
2.3	Abgeleitete Klassen und die Sichtbarkeit von Feldern und Methoden . . . . .	103
	☆☆ Aufgabe 2.1: Test von Sichtbarkeitsebenen im Zusammenhang mit abgeleiteten Klassen . . . . .	106

2.4	Das Verdecken von Klassenmethoden und das statische Binden von Methoden . . . . .	107
	☆☆ Aufgabe 2.2: Der Aufruf von verdeckten Klassenmethoden . . . . .	107
2.5	Das Überschreiben von Instanzmethoden und das dynamische Binden von Methoden . . . . .	108
	☆ Aufgabe 2.3: Das dynamische Binden von Methoden . . . . .	109
2.6	Vererbung und Komposition . . . . .	109
	☆ Aufgabe 2.4: Die Komposition . . . . .	110
	☆ Aufgabe 2.5: Die Vererbung . . . . .	110
2.7	Kovariante Rückgabetypen in Methoden . . . . .	111
	☆☆ Aufgabe 2.6: Die Benutzung von kovarianten Rückgabetypen . . . . .	111
2.8	Verdeckte Felder . . . . .	112
2.9	Vergrößernde und verkleinernde Konvertierung (»up- und down-casting«) . . . . .	112
2.10	Der Polymorphismus, ein Prinzip der objektorientierten Programmierung . . . . .	112
	☆☆ Aufgabe 2.7: Der »Subtyp-Polymorphismus« im Kontext einer Klassenhierarchie . . . . .	113
2.11	Die Methoden der Klassen <code>java.lang.Object</code> und <code>java.util.Objects</code> . . . . .	114
	☆ Aufgabe 2.8: Die <code>equals()</code> - und <code>hashCode()</code> -Methoden von <code>Object</code> . . . . .	121
	☆☆☆ Aufgabe 2.9: Die <code>equals()</code> -Methode und die Vererbung . . . . .	122
2.12	Das Klonen und die Gleichheit von geklonten Objekten . . . . .	126
	☆ Aufgabe 2.10: Das Klonen von Instanzen der eigenen Klasse . . . . .	126
	☆ Aufgabe 2.11: Das Klonen von Instanzen anderer Klassen . . . . .	126
	☆ Aufgabe 2.12: Das Klonen und der <code>Copy-Konstruktor</code> . . . . .	127
2.13	Der Garbage Collector und das Beseitigen von Objekten . . . . .	127
2.14	Lösungen . . . . .	128
	Lösung 2.1 . . . . .	128
	Lösung 2.2 . . . . .	132
	Lösung 2.3 . . . . .	134
	Lösung 2.4 . . . . .	136
	Lösung 2.5 . . . . .	138
	Lösung 2.6 . . . . .	140
	Lösung 2.7 . . . . .	142

	Lösung 2.8 .....	145
	Lösung 2.9 .....	151
	Lösung 2.10 .....	158
	Lösung 2.11 .....	159
	Lösung 2.12 .....	160
<b>3</b>	<b>Die Definition von abstrakten Klassen, Interfaces und Annotationen .....</b>	<b>163</b>
3.1	Abstrakte Klassen .....	163
3.2	Abstrakte Java-Standard-Klassen und eigene Definitionen von abstrakten Klassen .....	163
	☆ Aufgabe 3.1: Die abstrakte Klasse Number und ihre Unterklassen .....	163
	☆ Aufgabe 3.2: Definition einer eigenen abstrakten Klasse .....	164
3.3	Interfaces (Schnittstellen) .....	165
	☆☆ Aufgabe 3.3: Die Definition eines Interface. ....	165
3.4	Die Entscheidung zwischen abstrakten Klassen und Interfaces ...	166
	☆☆ Aufgabe 3.4: Paralleler Einsatz von Interfaces und abstrakten Klassen .....	167
3.5	Implementieren mehrerer Interfaces für eine Klasse .....	168
	☆☆ Aufgabe 3.5: Das Ableiten von Interfaces .....	168
3.6	Die Definition von inneren Klassen .....	169
	☆ Aufgabe 3.6: Ein Beispiel mit anonymer Klasse ...	171
3.7	Erweiterungen in der Definition von Interfaces .....	172
	☆☆☆ Aufgabe 3.7: Private Interface-Methoden. ....	173
3.8	Die Definition von Annotationen .....	175
3.9	Vordefinierte Annotationstypen .....	178
	☆ Aufgabe 3.8: Annotationen an Methoden und Parameter von Methoden anheften .....	179
	☆ Aufgabe 3.9: Eine Klasse annotieren .....	180
	☆☆ Aufgabe 3.10: Die @Override- und @Inherited-Annotation. ....	180
3.10	Lösungen .....	181
	Lösung 3.1 .....	181
	Lösung 3.2 .....	183
	Lösung 3.3 .....	184
	Lösung 3.4 .....	187
	Lösung 3.5 .....	190
	Lösung 3.6 .....	192
	Lösung 3.7 .....	193
	Lösung 3.8 .....	196

	Lösung 3.9 . . . . .	198
	Lösung 3.10 . . . . .	200
4	<b>Generics</b> . . . . .	205
4.1	Die Generizität . . . . .	205
4.2	Generische Klassen und Interfaces . . . . .	206
	☆ Aufgabe 4.1: Generischer Datentyp als Behälter für die Instanzen vom Typ des Klassenparameters . . . . .	207
	☆ Aufgabe 4.2: Generischer Datentyp als »Über-Typ« für die Instanzen vom Typ des Klassenparameters . . . . .	207
4.3	Wildcardtypen . . . . .	208
	☆☆ Aufgabe 4.3: Ungebundene Wildcardtypen . . . . .	208
	☆☆ Aufgabe 4.4: Obere und untere Schranken für Wildcardtypen . . . . .	208
4.4	Legacy Code, Erasure und Raw-Typen . . . . .	209
	☆☆ Aufgabe 4.5: Raw-Typen am Beispiel einer generischen Klasse mit zwei Typparametern . . . . .	210
	☆☆ Aufgabe 4.6: Brückenmethoden (»bridge methods«) . . . . .	211
4.5	Generische Arrays . . . . .	212
	☆☆ Aufgabe 4.7: Erzeugen von generischen Arrays . . . . .	212
4.6	Generische Methoden . . . . .	213
	☆☆ Aufgabe 4.8: Generische Methodendefinitionen . . . . .	213
4.7	Generische Standard-Klassen und -Interfaces. . . . .	213
4.8	for-each-Schleifen für Collections . . . . .	216
	☆ Aufgabe 4.9: Das Interface List<E> und die Klasse ArrayList<E> . . . . .	217
	☆☆ Aufgabe 4.10: Das Interface Collection<E> und die Klasse Vector<E>. . . . .	217
	☆☆ Aufgabe 4.11: Das Interface Map<K,V> und die Klasse TreeMap<K,V>. . . . .	218
4.9	Factory-Methoden in Collections . . . . .	219
	☆ Aufgabe 4.12: Factory-Methoden für List, Set und Map. . . . .	220
4.10	Die Interfaces Enumeration<E>, Iterable<T> und Iterator<E> . . . . .	220
4.11	Enumerationen und die generische Klasse Enum<E extends Enum<E>>. . . . .	221
	☆☆ Aufgabe 4.13: Die Definition von Enumerationen . . . . .	222
4.12	Die Interfaces Comparable<T> und Comparator<T> und das Sortieren von Objekten. . . . .	222
	☆☆☆ Aufgabe 4.14: Das Comparable<T>-Interface . . . . .	225
	☆☆☆ Aufgabe 4.15: Comparable<T> versus Comparator<T>. . . . .	228

4.13	Typinferenz für Methoden .....	231
4.14	Typinferenz beim Erzeugen von Instanzen eines generischen Typs .....	231
	☆☆ Aufgabe 4.16: Typinferenz beim Instanzieren von generischen Klassen .....	234
	☆☆ Aufgabe 4.17: Der Diamond-Operator in Java 9 .....	235
4.15	Lösungen .....	237
	Lösung 4.1 .....	237
	Lösung 4.2 .....	238
	Lösung 4.3 .....	239
	Lösung 4.4 .....	240
	Lösung 4.5 .....	242
	Lösung 4.6 .....	244
	Lösung 4.7 .....	246
	Lösung 4.8 .....	247
	Lösung 4.9 .....	249
	Lösung 4.10 .....	250
	Lösung 4.11 .....	251
	Lösung 4.12 .....	253
	Lösung 4.13 .....	255
	Lösung 4.14 .....	257
	Lösung 4.15 .....	265
	Lösung 4.16 .....	270
	Lösung 4.17 .....	274
<b>5</b>	<b>Exceptions und Errors .....</b>	<b>279</b>
5.1	Ausnahmen auslösen .....	279
5.2	Ausnahmen abfangen oder weitergeben .....	280
	☆☆ Aufgabe 5.1: Unbehandelte RuntimeExceptions .....	280
	☆☆ Aufgabe 5.2: Behandelte RuntimeExceptions .....	280
5.3	Das Verwenden von finally in der Ausnahmebehandlung .....	281
	☆☆ Aufgabe 5.3: Der finally-Block .....	281
5.4	Ausnahmen manuell auslösen .....	282
5.5	Exception-Unterklassen erzeugen .....	282
	☆☆ Aufgabe 5.4: Benutzerdefinierte Ausnahmen manuell auslösen .....	282
5.6	Multi-catch-Klausel und verbesserte Typprüfung beim Rethrowing von Exceptions .....	283
	☆☆ Aufgabe 5.5: Disjunction-Typ für Exceptions .....	284
	☆☆ Aufgabe 5.6: Typprüfung beim Rethrowing von Exceptions .....	285

5.7	Lösungen .....	287
	Lösung 5.1 .....	287
	Lösung 5.2 .....	288
	Lösung 5.3 .....	290
	Lösung 5.4 .....	291
	Lösung 5.5 .....	293
	Lösung 5.6 .....	296
<b>6</b>	<b>Lambdas und Streams .....</b>	<b>301</b>
6.1	Mittels anonymer Klassen Code an Methoden übergeben .....	301
6.2	Funktionale Interfaces .....	301
6.3	Syntax und Deklaration von Lambda-Ausdrücken .....	302
	★ Aufgabe 6.1: Lambda-Ausdruck ohne Parameter versus anonymer Klasse .....	304
	★ Aufgabe 6.2: Lambda-Ausdruck mit Parameter versus anonymer Klasse .....	305
6.4	Scoping und Variable Capture .....	306
	☆☆ Aufgabe 6.3: Die Umgebung von Lambda-Ausdrücken .....	307
6.5	Methoden- und Konstruktor-Referenzen .....	308
	★ Aufgabe 6.4: Methoden-Referenzen in Zuweisungen .....	310
	☆☆ Aufgabe 6.5: Konstruktor-Referenzen und die neuen funktionalen Interfaces <code>Supplier&lt;T&gt;</code> und <code>Function&lt;T,R&gt;</code> .....	311
6.6	Default- und statische Methoden in Interfaces .....	312
6.7	Das neue Interface <code>Stream</code> .....	314
6.8	Die <code>forEach</code> -Methoden von <code>Iterator</code> , <code>Iterable</code> und <code>Stream</code> .....	317
	★ Aufgabe 6.6: Die funktionalen Interfaces <code>BiConsumer&lt;T,U&gt;</code> , <code>BiPredicate&lt;T,U&gt;</code> und <code>BiFunction&lt;T,U,R&gt;</code> .....	319
	☆☆ Aufgabe 6.7: Die Methoden des Interface <code>Stream</code> und die Behandlung von <code>Exceptions</code> in Lambda-Ausdrücken .....	320
6.9	Das Interface <code>Collector</code> und die Klasse <code>Collectors</code> : Reduktion mittels Methoden von <code>Streams</code> und <code>Kollektoren</code> .....	321
	☆☆ Aufgabe 6.8: Weitere Methoden des Interface <code>Stream</code> : <code>limit()</code> , <code>count()</code> , <code>max()</code> , <code>min()</code> , <code>skip()</code> , <code>reduce()</code> und <code>collect()</code> .....	323
	☆☆☆☆ Aufgabe 6.9: Das Interface <code>Collector</code> und die Klasse <code>Collectors</code> .....	326
6.10	Parallele <code>Streams</code> .....	327
	☆☆☆☆ Aufgabe 6.10: Parallele <code>Streams</code> .....	329

6.11	Die map()- und flatMap()-Methoden von Stream und Optional. . . . .	331
	☆☆ Aufgabe 6.11: map() versus flatMap() . . . . .	333
6.12	Spracherweiterungen mit Java 10, Java 11, Java 12 und Java 13 . . . . .	335
	☆☆ Aufgabe 6.12: Typinferenz für lokale Variablen in Java 10 und Java 11. . . . .	340
	☆☆ Aufgabe 6.13: Switch-Statements und Switch-Expressions . . . . .	341
6.13	Lösungen . . . . .	343
	Lösung 6.1 . . . . .	343
	Lösung 6.2 . . . . .	344
	Lösung 6.3 . . . . .	346
	Lösung 6.4 . . . . .	351
	Lösung 6.5 . . . . .	352
	Lösung 6.6 . . . . .	357
	Lösung 6.7 . . . . .	361
	Lösung 6.8 . . . . .	367
	Lösung 6.9 . . . . .	385
	Lösung 6.10 . . . . .	395
	Lösung 6.11 . . . . .	404
	Lösung 6.12 . . . . .	409
	Lösung 6.13 . . . . .	415
7	<b>Die Modularität von Java</b> . . . . .	423
7.1	Das Java-Modulsystem. . . . .	423
	☆☆ Aufgabe 7.1: Eine einfache Modul-Definition . . . . .	430
	☆☆ Aufgabe 7.2: Eine Applikation mit mehreren Modulen . . . . .	432
	☆☆ Aufgabe 7.3: Implizites Lesen von Modulen . . . . .	434
	☆☆ Aufgabe 7.4: Eine modulbasierte Service-Implementierung . . . . .	436
7.2	Lösungen . . . . .	437
	Lösung 7.1 . . . . .	437
	Lösung 7.2 . . . . .	439
	Lösung 7.3 . . . . .	445
	Lösung 7.4 . . . . .	451
8	<b>Records, Sealed Classes und Pattern Matching</b> . . . . .	457
8.1	Das Pattern Matching für den instanceof-Operator . . . . .	457
8.2	Der neue Java-Typ Record . . . . .	459
8.3	Sealed Classes in Java . . . . .	464

8.4	Das Pattern Matching für switch . . . . .	469
	☆ Aufgabe 8.1: Die Definition von record-Klassen und das Pattern Matching für den instanceof-Operator. . .	482
	☆ Aufgabe 8.2: sealed-, final- und non-sealed-Klassen . . . . .	483
	☆ Aufgabe 8.3: sealed-Interfaces und das Pattern Matching . . . . .	485
	☆☆ Aufgabe 8.4: Algebraische Datentypen (ADTs), ein weiterer Schritt in Richtung funktionale Programmierung . . . . .	487
	☆☆ Aufgabe 8.5: Das Pattern Matching für switch . . . . .	488
8.5	Lösungen . . . . .	488
	Lösung 8.1 . . . . .	488
	Lösung 8.2 . . . . .	493
	Lösung 8.3 . . . . .	499
9	<b>JUnit-Tests</b> . . . . .	503
9.1	JUnit 5 im Überblick . . . . .	503
9.2	Tests schreiben . . . . .	504
9.3	Testen mit dem ConsoleLauncher und der JupiterEngine . . . . .	507
	☆ Aufgabe 9.1: Die Klassen App und AppTest . . . . .	508
	☆ Aufgabe 9.2: Die Klasse PublishingBookmitOrderingTest . . . . .	511
	☆ Aufgabe 9.3: Die Klassen AdditionmitMap und AdditionmitMapTest . . . . .	512
	☆☆ Aufgabe 9.4: Die Klassen MyClassTest und BuchmitEqualsTest . . . . .	513
	☆☆ Aufgabe 9.5: Die Klasse TestBeispiele . . . . .	517
	☆☆ Aufgabe 9.6: Die Klassen RechenOperationenTest und RechenOperationenParametrisierteTests . . . . .	517
	☆☆ Aufgabe 9.7: Die Klasse AssertThrowsTest . . . . .	519
9.4	JUnit-Tests mit Gradle . . . . .	520
9.5	Lösungen . . . . .	527
	Lösung 9.1 . . . . .	527
	Lösung 9.3 . . . . .	528
	Lösung 9.4 . . . . .	531
	Lösung 9.5 . . . . .	534
	<b>Stichwortverzeichnis</b> . . . . .	541

Diese Leseprobe haben Sie beim  
 **edv-buchversand.de** heruntergeladen.  
Das Buch können Sie online in unserem  
Shop bestellen.

[Hier zum Shop](#)