

Inhaltsverzeichnis

	Einleitung	13
	Vorkenntnisse	13
	Aufbau des Buches	13
	Benötigte Software	17
	Autorin	18
1	Klassendefinition und Objektinstanziierung	19
1.1	Klassen und Objekte	19
	☆ Aufgabe 1.1: Definition einer Klasse	23
	☆ Aufgabe 1.2: Objekt (Instanz) einer Klasse erzeugen	23
1.2	Das Überladen von Methoden	23
	☆ Aufgabe 1.3: Eine Methode überladen	24
1.3	Die Datenkapselung, ein Prinzip der objektorientierten Programmierung	24
	☆ Aufgabe 1.4: Zugriffsmethoden	25
1.4	Das »aktuelle Objekt« und die »this-Referenz«	25
	☆☆ Aufgabe 1.5: Konstruktordefinitionen	25
1.5	Die Wert- und Referenzübergabe in Methodenaufrufen	26
	☆☆☆ Aufgabe 1.6: Wertübergabe in Methoden (»call by value«)	26
1.6	Globale und lokale Referenzen	27
	☆☆ Aufgabe 1.7: Der Umgang mit Referenzen	27
1.7	Java-Pakete	27
	☆ Aufgabe 1.8: Die package-Anweisung	28
	☆ Aufgabe 1.9: Die import-Anweisung	29
1.8	Die Modifikatoren für Felder und Methoden in Zusammenhang mit der Definition von Paketen	29
	☆☆ Aufgabe 1.10: Pakete und die Sichtbarkeit von Mitgliedern einer Klasse	30
1.9	Standard-Klassen von Java	30
1.10	Die Wrapper-Klassen von Java und das Auto(un)boxing	32
	☆☆ Aufgabe 1.11: Das Auto(un)boxing	34
1.11	Das Paket java.lang.reflect.	34
1.12	Arrays (Reihungen) und die Klassen Array und Arrays	36
	☆☆ Aufgabe 1.12: Der Umgang mit Array-Objekten	37
1.13	Zeichenketten und die Klasse String	37

	☆	Aufgabe 1.13: Der Umgang mit String-Objekten	46
	☆☆	Aufgabe 1.14: Der Umgang mit Textblöcken.	47
1.14		Sprachmerkmale, die in den weiteren Beispielen genutzt werden	48
	☆	Aufgabe 1.15: Methoden mit variablen Argumentenlisten	49
1.15		Den Zugriff auf Klassen und Felder minimieren: Unveränderliche (immutable) Klassen und Objekte.	49
1.16		Die alte und neue Date&Time-API als Beispiel für veränderliche und unveränderliche Klassen	51
	☆☆☆	Aufgabe 1.16: Die Methoden von Date/Time-Klassen	55
1.17		Private Konstruktoren	56
	☆	Aufgabe 1.17: Objekte mithilfe eines privaten Konstruktors erzeugen	56
1.18		Lösungen	57
		Lösung 1.1	57
		Lösung 1.2	57
		Lösung 1.3	58
		Lösung 1.4	59
		Lösung 1.5	60
		Lösung 1.6	62
		Lösung 1.7	64
		Lösung 1.8	67
		Lösung 1.9	67
		Lösung 1.10	68
		Lösung 1.11.	69
		Lösung 1.12	72
		Lösung 1.13	75
		Lösung 1.14	78
		Lösung 1.15	87
		Lösung 1.16	89
		Lösung 1.17	96
2		Abgeleitete Klassen und Vererbung	99
2.1		Abgeleitete Klassen	99
2.2		Die Konstruktoren von abgeleiteten Klassen	99
2.3		Abgeleitete Klassen und die Sichtbarkeit von Feldern und Methoden	99
	☆☆	Aufgabe 2.1: Test von Sichtbarkeits Ebenen im Zusammenhang mit abgeleiteten Klassen	102

2.4	Das Verdecken von Klassenmethoden und das statische Binden von Methoden	103
	☆☆ Aufgabe 2.2: Der Aufruf von verdeckten Klassenmethoden	103
2.5	Das Überschreiben von Instanzmethoden und das dynamische Binden von Methoden	104
	☆ Aufgabe 2.3: Das dynamische Binden von Methoden	105
2.6	Vererbung und Komposition	105
	☆ Aufgabe 2.4: Die Komposition	106
	☆ Aufgabe 2.5: Die Vererbung	106
2.7	Kovariante Rückgabetypen in Methoden	107
	☆☆ Aufgabe 2.6: Die Benutzung von kovarianten Rückgabetypen	107
2.8	Verdeckte Felder	108
2.9	Vergrößernde und verkleinernde Konvertierung (»up- und down-casting«)	108
2.10	Der Polymorphismus, ein Prinzip der objektorientierten Programmierung	108
	☆☆ Aufgabe 2.7: Der »Subtyp-Polymorphismus« im Kontext einer Klassenhierarchie	109
2.11	Die Methoden der Klassen java.lang.Object und java.util.Objects	110
	☆ Aufgabe 2.8: Die equals()- und hashCode()-Methoden von Object	117
	☆☆☆ Aufgabe 2.9: Die equals()-Methode und die Vererbung	118
2.12	Das Klonen und die Gleichheit von geklonten Objekten	122
	☆ Aufgabe 2.10: Das Klonen von Instanzen der eigenen Klasse	122
	☆ Aufgabe 2.11: Das Klonen von Instanzen anderer Klassen	122
	☆ Aufgabe 2.12: Das Klonen und der Copy-Konstruktor	123
2.13	Der Garbage Collector und das Beseitigen von Objekten	123
2.14	Lösungen	124
	Lösung 2.1	124
	Lösung 2.2	128
	Lösung 2.3	130
	Lösung 2.4	132
	Lösung 2.5	134
	Lösung 2.6	136
	Lösung 2.7	138

	Lösung 2.8	141
	Lösung 2.9	147
	Lösung 2.10	154
	Lösung 2.11	155
	Lösung 2.12	156
3	Die Definition von abstrakten Klassen, Interfaces und Annotationen	159
3.1	Abstrakte Klassen	159
3.2	Abstrakte Java-Standard-Klassen und eigene Definitionen von abstrakten Klassen	159
	☆ Aufgabe 3.1: Die abstrakte Klasse Number und ihre Unterklassen	159
	☆ Aufgabe 3.2: Definition einer eigenen abstrakten Klasse	160
3.3	Interfaces (Schnittstellen)	161
	☆☆ Aufgabe 3.3: Die Definition eines Interface	161
3.4	Die Entscheidung zwischen abstrakten Klassen und Interfaces ...	162
	☆☆ Aufgabe 3.4: Paralleler Einsatz von Interfaces und abstrakten Klassen	163
3.5	Implementieren mehrerer Interfaces für eine Klasse	164
	☆☆ Aufgabe 3.5: Das Ableiten von Interfaces	164
3.6	Die Definition von inneren Klassen	165
	☆ Aufgabe 3.6: Ein Beispiel mit anonymer Klasse ...	167
3.7	Erweiterungen in der Definition von Interfaces	168
	☆☆☆ Aufgabe 3.7: Private Interface-Methoden	169
3.8	Die Definition von Annotationen	171
3.9	Vordefinierte Annotationstypen	174
	☆ Aufgabe 3.8: Annotationen an Methoden und Parameter von Methoden anheften	175
	☆ Aufgabe 3.9: Eine Klasse annotieren	176
	☆☆ Aufgabe 3.10: Die @Override- und @Inherited-Annotation	176
3.10	Lösungen	177
	Lösung 3.1	177
	Lösung 3.2	179
	Lösung 3.3	180
	Lösung 3.4	183
	Lösung 3.5	186
	Lösung 3.6	188
	Lösung 3.7	189
	Lösung 3.8	192

	Lösung 3.9	194
	Lösung 3.10	196
4	Generics	201
4.1	Die Generizität	201
4.2	Generische Klassen und Interfaces	202
	☆ Aufgabe 4.1: Generischer Datentyp als Behälter für die Instanzen vom Typ des Klassenparameters	203
	☆ Aufgabe 4.2: Generischer Datentyp als »Über-Typ« für die Instanzen vom Typ des Klassenparameters ..	203
4.3	Wildcardtypen	204
	☆☆ Aufgabe 4.3: Ungebundene Wildcardtypen	204
	☆☆ Aufgabe 4.4: Obere und untere Schranken für Wildcardtypen	204
4.4	Legacy Code, Erasure und Raw-Typen	205
	☆☆ Aufgabe 4.5: Raw-Typen am Beispiel einer generischen Klasse mit zwei Typparametern	206
	☆☆ Aufgabe 4.6: Brückenmethoden (»bridge methods«)	207
4.5	Generische Arrays	208
	☆☆ Aufgabe 4.7: Erzeugen von generischen Arrays	208
4.6	Generische Methoden	209
	☆☆ Aufgabe 4.8: Generische Methodendefinitionen	209
4.7	Generische Standard-Klassen und -Interfaces	209
4.8	for-each-Schleifen für Collections	212
	☆ Aufgabe 4.9: Das Interface List<E> und die Klasse ArrayList<E>	213
	☆☆ Aufgabe 4.10: Das Interface Collection<E> und die Klasse Vector<E>	213
	☆☆ Aufgabe 4.11: Das Interface Map<K,V> und die Klasse TreeMap<K,V>	214
4.9	Factory-Methoden in Collections	215
	☆ Aufgabe 4.12: Factory-Methoden für List, Set und Map	216
4.10	Die Interfaces Enumeration<E>, Iterable<T> und Iterator<E>	216
4.11	Enumerationen und die generische Klasse Enum<E> extends Enum<E>	217
	☆☆ Aufgabe 4.13: Die Definition von Enumerationen ..	218
4.12	Die Interfaces Comparable<T> und Comparator<T> und das Sortieren von Objekten	218
	☆☆☆☆ Aufgabe 4.14: Das Comparable<T>-Interface	221
	☆☆☆☆ Aufgabe 4.15: Comparable<T> versus Comparator<T>	224

4.13	Typinferenz für Methoden	227
4.14	Typinferenz beim Erzeugen von Instanzen eines generischen Typs	227
	☆☆ Aufgabe 4.16: Typinferenz beim Instanzieren von generischen Klassen	230
	☆☆ Aufgabe 4.17: Der Diamond-Operator in Java 9	231
4.15	Lösungen	233
	Lösung 4.1	233
	Lösung 4.2	234
	Lösung 4.3	235
	Lösung 4.4	236
	Lösung 4.5	238
	Lösung 4.6	240
	Lösung 4.7	242
	Lösung 4.8	243
	Lösung 4.9	245
	Lösung 4.10	246
	Lösung 4.11	247
	Lösung 4.12	249
	Lösung 4.13	251
	Lösung 4.14	253
	Lösung 4.15	261
	Lösung 4.16	266
	Lösung 4.17	270
5	Exceptions und Errors	275
5.1	Ausnahmen auslösen	275
5.2	Ausnahmen abfangen oder weitergeben	276
	☆ Aufgabe 5.1: Unbehandelte RuntimeExceptions	276
	☆ Aufgabe 5.2: Behandelte RuntimeExceptions	276
5.3	Das Verwenden von finally in der Ausnahmebehandlung	277
	☆ Aufgabe 5.3: Der finally-Block	277
5.4	Ausnahmen manuell auslösen	278
5.5	Exception-Unterklassen erzeugen	278
	☆ Aufgabe 5.4: Benutzerdefinierte Ausnahmen manuell auslösen	278
5.6	Multi-catch-Klausel und verbesserte Typprüfung beim Rethrowing von Exceptions	279
	☆☆ Aufgabe 5.5: Disjunction-Typ für Exceptions	280
	☆☆ Aufgabe 5.6: Typprüfung beim Rethrowing von Exceptions	281

5.7	Lösungen	283
	Lösung 5.1	283
	Lösung 5.2	284
	Lösung 5.3	286
	Lösung 5.4	287
	Lösung 5.5	289
	Lösung 5.6	292
6	Lambdas und Streams	297
6.1	Mittels anonymer Klassen Code an Methoden übergeben	297
6.2	Funktionale Interfaces	297
6.3	Syntax und Deklaration von Lambda-Ausdrücken	298
	☆ Aufgabe 6.1: Lambda-Ausdruck ohne Parameter versus anonymer Klasse	300
	☆ Aufgabe 6.2: Lambda-Ausdruck mit Parameter versus anonymer Klasse	301
6.4	Scoping und Variable Capture	302
	☆☆ Aufgabe 6.3: Die Umgebung von Lambda-Ausdrücken	303
6.5	Methoden- und Konstruktor-Referenzen	304
	☆ Aufgabe 6.4: Methoden-Referenzen in Zuweisungen	306
	☆☆ Aufgabe 6.5: Konstruktor-Referenzen und die neuen funktionalen Interfaces <code>Supplier<T></code> und <code>Function<T,R></code>	307
6.6	Default- und statische Methoden in Interfaces	308
6.7	Das neue Interface <code>Stream</code>	310
6.8	Die <code>forEach</code> -Methoden von <code>Iterator</code> , <code>Iterable</code> und <code>Stream</code>	313
	☆ Aufgabe 6.6: Die funktionalen Interfaces <code>BiConsumer<T,U></code> , <code>BiPredicate<T,U></code> und <code>BiFunction<T,U,R></code>	315
	☆☆ Aufgabe 6.7: Die Methoden des Interface <code>Stream</code> und die Behandlung von <code>Exceptions</code> in <code>Lambda-Ausdrücken</code>	316
6.9	Das Interface <code>Collector</code> und die Klasse <code>Collectors</code> : Reduktion mittels Methoden von <code>Streams</code> und Kollektoren	317
	☆☆ 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>	319
	☆☆☆ Aufgabe 6.9: Das Interface <code>Collector</code> und die Klasse <code>Collectors</code>	322
6.10	Parallele <code>Streams</code>	323
	☆☆☆ Aufgabe 6.10: Parallele <code>Streams</code>	325

6.11	Die map()- und flatMap()-Methoden von Stream und Optional	327
	☆☆ Aufgabe 6.11: map() versus flatMap()	329
6.12	Spracherweiterungen mit Java 10, Java 11, Java 12 und Java 13	331
	☆☆ Aufgabe 6.12: Typinferenz für lokale Variablen in Java 10 und Java 11.	336
	☆☆ Aufgabe 6.13: Switch-Statements und Switch-Expressions	337
6.13	Lösungen	339
	Lösung 6.1	339
	Lösung 6.2	340
	Lösung 6.3	342
	Lösung 6.4	347
	Lösung 6.5	348
	Lösung 6.6	353
	Lösung 6.7	357
	Lösung 6.8	363
	Lösung 6.9	381
	Lösung 6.10	391
	Lösung 6.11	400
	Lösung 6.12	405
	Lösung 6.13	412
7	Die Modularität von Java	419
7.1	Das Java-Modulsystem.	419
	☆☆ Aufgabe 7.1: Eine einfache Modul-Definition	426
	☆☆ Aufgabe 7.2: Eine Applikation mit mehreren Modulen	428
	☆☆ Aufgabe 7.3: Implizites Lesen von Modulen	430
	☆☆ Aufgabe 7.4: Eine modulbasierte Service-Implementierung	432
7.2	Lösungen	433
	Lösung 7.1	433
	Lösung 7.2	435
	Lösung 7.3	441
	Lösung 7.4	447
	Stichwortverzeichnis	453