

Inhaltsverzeichnis

	Einleitung	25
	Autorin	28
I	Klassendefinition und Objektinstanziierung	31
1.1	Klassen und Objekte	31
	Aufgabe 1.1: Definition einer Klasse	34
	Aufgabe 1.2: Objekt (Instanz) einer Klasse erzeugen ..	35
1.2	Die Member einer Klasse: Felder und Methoden	35
	Aufgabe 1.3: Zugriff auf Felder	36
	Aufgabe 1.4: Aufruf von Methoden	36
1.3	Das Überladen von Methoden	37
	Aufgabe 1.5: Eine Methode überladen	37
1.4	Die Datenkapselung, ein Prinzip der objektorientierten Programmierung	38
	Aufgabe 1.6: Zugriffsmethoden	38
1.5	Das »aktuelle Objekt« und die »this-Referenz«	38
	Aufgabe 1.7: Konstruktordefinitionen	39
1.6	Die Wert- und Referenzübergabe in Methodenaufrufen	39
	Aufgabe 1.8: Wertübergabe in Methoden (»call by value«)	40
1.7	Globale und lokale Referenzen	40
	Aufgabe 1.9: Der Umgang mit Referenzen	40
	Aufgabe 1.10: Wiederholungsaufgabe	41
1.8	Selbstreferenzierende Klassen und Felder (»self-referential classes and fields«)	42
	Aufgabe 1.11: Der Einsatz von selbstreferenzierenden Feldern	42
1.9	Java-Pakete	43
	Aufgabe 1.12: Die package-Anweisung	44
	Aufgabe 1.13: Die import-Anweisung	45
1.10	Die Modifikatoren für Felder und Methoden in Zusammenhang mit der Definition von Paketen	45

	Aufgabe 1.14: Pakete und die Sichtbarkeit von Membem einer Klasse	45
1.11	Standard-Klassen von Java	46
	Aufgabe 1.15: Aufruf von Methoden der Klasse Math.	48
	Aufgabe 1.16: Wiederholungsaufgabe	49
1.12	Die Wrapper-Klassen von Java und das Auto(un)boxing	50
	Aufgabe 1.17: Die Felder und Methoden von Wrapper-Klassen.	51
	Aufgabe 1.18: Das Auto(un)boxing	51
1.13	Arrays (Reihungen) und die Klassen Array und Arrays.	52
	Aufgabe 1.19: Der Umgang mit Array-Objekten	53
1.14	Zeichenketten und die Klasse String	54
	Aufgabe 1.20: Der Umgang mit String-Objekten.	54
1.15	Mit der Version 5.0 eingeführte Spracherneuerungen für Arrays und Methoden	54
	Aufgabe 1.21: Einfache und erweiterte for-Schleifen	55
	Aufgabe 1.22: Methoden mit variablen Argumentenlisten.	56
1.16	Das Initialisieren von Klassen- und Instanzfeldern	56
	Aufgabe 1.23: Das Initialisieren von Instanzfeldern	57
	Aufgabe 1.24: Das Initialisieren von Klassenfeldern	57
1.17	Private Konstruktoren	57
	Aufgabe 1.25: Ein Objekt mit Hilfe eines privaten Konstruktoren erzeugen	58
	Aufgabe 1.26: Mehrere konstante Werte (Objekte) mit Hilfe eines privaten Konstruktoren erzeugen	58
1.18	Lösungen	59
	Lösung 1.1	59
	Lösung 1.2	59
	Lösung 1.3	59
	Lösung 1.4	61
	Lösung 1.5	63
	Lösung 1.6	64
	Lösung 1.7	65
	Lösung 1.8	67
	Lösung 1.9	69
	Lösung 1.10	71
	Lösung 1.11.	76

	Lösung 1.12	78
	Lösung 1.13	78
	Lösung 1.14	79
	Lösung 1.15	80
	Lösung 1.16	82
	Lösung 1.17	85
	Lösung 1.18	87
	Lösung 1.19	90
	Lösung 1.20	93
	Lösung 1.21	95
	Lösung 1.22	96
	Lösung 1.23	98
	Lösung 1.24	100
	Lösung 1.25	101
	Lösung 1.26	102
2	Abgeleitete Klassen und Vererbung	105
2.1	Abgeleitete Klassen	105
2.2	Die Konstruktoren von abgeleiteten Klassen	105
2.3	Abgeleitete Klassen und die Sichtbarkeit von Feldern und Methoden	105
	Aufgabe 2.1: Test von Sichtbarkeits Ebenen	106
2.4	Das Verdecken von Klassenmethoden und das statische Binden von Methoden.	107
	Aufgabe 2.2: Der Aufruf von verdeckten Klassenmethoden	108
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
	Aufgabe 2.7: Wiederholungsaufgabe	112
2.9	Vergrößernde und verkleinernde Konvertierung (»up- und down-casting«)	114

	Aufgabe 2.8: Up- und Down-Casts	114
	Aufgabe 2.9: Der Unterschied zwischen »ist-ein-« und »hat-ein- Beziehungen«	115
2.10	Der Polymorphismus, ein Prinzip der objektorientierten Programmierung	116
	Aufgabe 2.10: Der »Subtyp-Polymorphismus« im Kontext einer Klassenhierarchie	117
2.11	Lösungen	118
	Lösung 2.1	118
	Lösung 2.2	121
	Lösung 2.3	123
	Lösung 2.4	125
	Lösung 2.5	127
	Lösung 2.6	128
	Lösung 2.7	131
	Lösung 2.8	135
	Lösung 2.9	139
	Lösung 2.10	142
3	Abstrakte Klassen und Interfaces	145
3.1	Abstrakte Klassen	145
3.2	Abstrakte Java-Standard-Klassen und eigene Definitionen von abstrakten Klassen	145
	Aufgabe 3.1: Die abstrakte Klasse Number und ihre Unterklassen	145
	Aufgabe 3.2: Definition einer eigenen abstrakten Klasse	146
3.3	Die Referenzen vom Typ einer abstrakten Klassen	147
	Aufgabe 3.3: Der Subtyp-Polymorphismus für Methoden im Kontext einer Klassenhierarchie mit abstrakten Klassendefinitionen	147
	Aufgabe 3.4: Der Subtyp-Polymorphismus für Felder im Kontext einer Klassenhierarchie mit abstrakten Klassendefinitionen	147
3.4	Interfaces (Schnittstellen)	148
	Aufgabe 3.5: Die Definition eines Interface	148
3.5	Die Entscheidung zwischen abstrakten Klassen und Interfaces	149
	Aufgabe 3.6: Paralleler Einsatz von Interfaces und abstrakten Klassen	150
3.6	Oberinterfaces	151
	Aufgabe 3.7: Das Ableiten von Interfaces	151

3.7	Implementieren von mehreren Interfaces für eine Klasse	152
	Aufgabe 3.8: Wiederholungsaufgabe	152
3.8	Die Vererbung an Beispielen von Java-Standard-Klassen und Standard-Interfaces	153
3.9	Das Klonen von Objekten	154
	Aufgabe 3.9: Das Klonen von Instanzen der eigenen Klasse	154
	Aufgabe 3.10: Das Klonen von Instanzen anderer Klassen	155
	Aufgabe 3.11: Das Klonen von Arrays	155
	Aufgabe 3.12: Das Überschreiben der clone()-Methode in Java 5.0	155
3.10	Die Gleichheit von Objekten	156
	Aufgabe 3.13: Die Gleichheit von geklonten Objekten	156
3.11	Das oberflächliche und das tiefe Klonen (»shallow und deep cloning«).	157
	Aufgabe 3.14: Das Klonen und der Copy-Konstruktor	157
	Aufgabe 3.15: Tiefes Klonen am Beispiel von Array-Objekten	158
	Aufgabe 3.16: Oberflächliches und tiefes Klonen für Referenztypen	158
3.12	Der Garbage Collector und das Beseitigen von Objekten	159
	Aufgabe 3.17: Das Zerstören von Instanzen	159
3.13	Lösungen	160
	Lösung 3.1	160
	Lösung 3.2	161
	Lösung 3.3	162
	Lösung 3.4	163
	Lösung 3.5	165
	Lösung 3.6	167
	Lösung 3.7	170
	Lösung 3.8	171
	Lösung 3.9	175
	Lösung 3.10	176
	Lösung 3.11	176
	Lösung 3.12	177
	Lösung 3.13	178
	Lösung 3.14	179

	Lösung 3.15	181
	Lösung 3.16	183
	Lösung 3.17	185
4	Einführung in die graphische Programmierung	187
4.1	Das AWT (Abstract Windowing Toolkit) und Swing	187
4.2	Fenster unter graphischen Oberflächen	191
4.3	Die Klassen Graphics und Graphics2D	191
4.4	Methoden zum Zeichnen	192
	Aufgabe 4.1: Eine einfache AWT-Komponente vom Typ Frame	195
	Aufgabe 4.2: Eine Einfache Swing-Komponente vom Typ JFrame	195
	Aufgabe 4.3: Ein JWindow-Fenster	196
	Aufgabe 4.4: Ein JDialog-Fenster	196
	Aufgabe 4.5: Das Neuzeichnen einer Swing- Komponente ohne Benutzung des Clip-Rectangles....	196
	Aufgabe 4.6: Das Neuzeichnen einer Swing- Komponente mit Benutzung des Clip-Rectangles	197
	Aufgabe 4.7: Das Parametrisieren der paint()-Methode	197
	Aufgabe 4.8: Die Instanzen der Klasse Color	198
4.5	Die Transparenz-Eigenschaft und der Hintergrund von Komponenten	198
4.6	Layout-Manager	199
	Aufgabe 4.9: Die vordefinierten Layout-Manager für Standard-Klassen	201
	Aufgabe 4.10: Die Layout-Manager von AWT-Komponenten	201
	Aufgabe 4.11: Die Layout-Manager von Swing-Komponenten	202
	Aufgabe 4.12: Das BorderLayout	202
	Aufgabe 4.13: Das GridBagLayout	203
	Aufgabe 4.14: Das null-Layout	204
4.7	Das Überlappen von Komponenten	204
	Aufgabe 4.15: Das Verhalten von AWT-LW- und -HW-Komponenten	205
	Aufgabe 4.16: AWT-Container mit LW- und HW-Kindkomponenten	205
	Aufgabe 4.17: Z-Order-Index für AWT-LW-Komponenten	206

	Aufgabe 4.18: Das Verhalten von Swing-Komponenten	206
	Aufgabe 4.19: Swing-Container und der Z-Order-Index für Swing-Komponenten	207
4.8	Das System als Auslöser für Zeichenoperationen	207
	Aufgabe 4.20: Resizing von AWT-Komponenten	208
	Aufgabe 4.21: Resizing von Swing-Komponenten	208
4.9	Eventbehandlung	209
4.10	Events auf niedriger Ebene	210
	Aufgabe 4.22: Das Interface WindowListener und die Klasse WindowEvent	210
	Aufgabe 4.23: Die Klasse WindowAdapter	210
4.11	Events auf höherer Ebene	211
	Aufgabe 4.24: Das Interface ActionListener und die Klasse ActionEvent	211
	Aufgabe 4.25: Das Interface KeyListener und die Klasse KeyEvent	211
4.12	Das Delegationsmodell in der Eventbehandlung	212
	Aufgabe 4.26: Die Ereignisbehandlung in einer separaten Klasse definieren	212
4.13	Lösungen	213
	Lösung 4.1	213
	Lösung 4.2	214
	Lösung 4.3	215
	Lösung 4.4	216
	Lösung 4.5	217
	Lösung 4.6	218
	Lösung 4.7	220
	Lösung 4.8	221
	Lösung 4.9	223
	Lösung 4.10	224
	Lösung 4.11	227
	Lösung 4.12	230
	Lösung 4.13	231
	Lösung 4.14	233
	Lösung 4.15	234
	Lösung 4.16	235
	Lösung 4.17	237
	Lösung 4.18	238
	Lösung 4.19	240
	Lösung 4.20	243

	Lösung 4.21	245
	Lösung 4.22	247
	Lösung 4.23	248
	Lösung 4.24	249
	Lösung 4.25	250
	Lösung 4.26	251
5	Erweiterte graphische Programmierung	253
5.1	Der RootPane-Container	253
	Aufgabe 5.1: Die LayeredPane einer Fensterkomponente	255
	Aufgabe 5.2: Eine beliebige Instanz der Klasse JLayeredPane	256
	Aufgabe 5.3: Das Positionieren von Komponenten in und innerhalb von Ebenen	257
	Aufgabe 5.4: Eine GlassPane, die Ereignisse empfängt	257
	Aufgabe 5.5: Eine benutzerdefinierte GlassPane-Komponente	258
5.2	Interne Fenster	258
	Aufgabe 5.6: Die Instanzen der Klasse JInternalFrame	259
	Aufgabe 5.7: Die Verschachtelung von internen Fenstern	259
5.3	Die Applikation als Auslöser für Zeichenoperationen	259
	Aufgabe 5.8: Aufruf der repaint()-Methode für AWT-Komponenten	260
	Aufgabe 5.9: Aufruf der repaint()-Methode für Swing-Komponenten	260
	Aufgabe 5.10: Die getGraphics()-Methode	261
	Aufgabe 5.11: Die Methoden paint(), getGraphics() und repaint() gleichzeitig nutzen.....	261
	Aufgabe 5.12: Weiterführendes Zeichnen (>incremental painting«)	261
5.4	Das Interface Shape	263
	Aufgabe 5.13: Die Methode draw() der Klasse Graphics2D	263
	Aufgabe 5.14: Wiederholungsaufgabe	264
	Aufgabe 5.15: Wiederholungsaufgabe	264
5.5	Praxisnahe Zeichenvorgänge und Eventbehandlungen	264

	Aufgabe 5.16: Wiederholungsaufgabe	264
5.6	Benutzerdefinierte Event-Objekte und Event-Listener	265
	Aufgabe 5.17: Das Erweitern der Klasse EventObject und des Interface EventListener	266
5.7	Lösungen	267
	Lösung 5.1	267
	Lösung 5.2	268
	Lösung 5.3	270
	Lösung 5.4	272
	Lösung 5.5	274
	Lösung 5.6	276
	Lösung 5.7	277
	Lösung 5.8	279
	Lösung 5.9	280
	Lösung 5.10	281
	Lösung 5.11	282
	Lösung 5.12	284
	Lösung 5.13	286
	Lösung 5.14	288
	Lösung 5.15	289
	Lösung 5.16	291
	Lösung 5.17	295
6	Das Erscheinungsbild einer Anwendung mit graphischer Oberfläche	301
6.1	Die Architektur Model View Controller (MVC) von Swing-Komponenten.	301
6.2	Benutzerdefinierte Modelle, die Standard-Model-Interfaces implementieren.	301
	Aufgabe 6.1: Die AWT-Klasse List und ein DefaultComboBox-Modell	302
	Aufgabe 6.2: Benutzerdefiniertes ComboBox- Modell ohne Eventbehandlung	303
	Aufgabe 6.3: Benutzerdefiniertes ComboBox- Modell mit Eventbehandlung	305
	Aufgabe 6.4: Benutzung des ComboBox-Modells für eine Viewer-Komponente vom Typ JComboBox ...	305
	Aufgabe 6.5: Die Vorlage für ein benutzerdefiniertes Modell als Erweiterung der Klasse PlainDocument definieren	306
6.3	Standard-Modelle am Beispiel der Klasse JTree	307

	Aufgabe 6.6: Die Klassen JTree und DefaultMutableTreeNode	307
	Aufgabe 6.7: Die Klassen DefaultTreeModel und DefaultTreeSelectionModel	307
	Aufgabe 6.8: Benutzerdefiniertes Tree-Modell mit Eventbehandlung	308
6.4	Die UI-Delegationsklassen und ihre Instanzen, der UI-Delegate	309
6.5	Java-Standard-User-Interface-Delegationsklassen	310
	Aufgabe 6.9: Der Standard-UI-Delegate	310
	Aufgabe 6.10: Das Setzen der bevorzugten Größe für Komponenten	310
	Aufgabe 6.11: Das Setzen einer minimalen Größe für Komponenten in Abhängigkeit von der Größe eines zu zeichnenden Bildes	311
	Aufgabe 6.12: Das Setzen einer minimalen Größe für Komponenten in Abhängigkeit von der Größe der beim Zeichnen verwendeten Schrift	311
6.6	Benutzerdefinierte User-Interface-Delegaten	312
	Aufgabe 6.13: Ein benutzerdefinierter UI-Delegate und das Setzen der bevorzugten Größe von Komponenten	312
	Aufgabe 6.14: Ein benutzerdefinierter UI-Delegate für eine benutzerdefinierte Komponente	313
	Aufgabe 6.15: Wiederholungsaufgabe	314
6.7	Die Klassen LookAndFeel, UIDefaults, UIManager und das Interface UIResource	315
	Aufgabe 6.16: Lesen von Werten der UIDefaults-Tabelle	316
	Aufgabe 6.17: Setzen von Werten der UIDefaults-Tabelle	317
	Aufgabe 6.18: Setzen des UI-Delegationsobjektes mit der Methode put() der UIManager-Klasse	317
6.8	Standard-LookAndFeel-Komponenten	318
	Aufgabe 6.19: LookAndFeel-spezifische Wiedergabe von Komponenten	319
	Aufgabe 6.20: Wechseln zwischen allen installierten LookAndFeel-Komponenten	320
6.9	Das Erweitern der LookAndFeel-Klasse	320
	Aufgabe 6.21: Benutzerdefinierte LookAndFeel-Komponenten	320

	Aufgabe 6.22: Das Zusammenspiel zwischen den drei MVC-Komponenten Model, View und Controller	321
6.10	Lösungen	322
	Lösung 6.1.....	322
	Lösung 6.2.....	324
	Lösung 6.3.....	326
	Lösung 6.4	329
	Lösung 6.5.....	331
	Lösung 6.6	334
	Lösung 6.7.....	335
	Lösung 6.8	338
	Lösung 6.9	343
	Lösung 6.10.....	344
	Lösung 6.11.....	345
	Lösung 6.12.....	347
	Lösung 6.13.....	349
	Lösung 6.14.....	352
	Lösung 6.15.....	355
	Lösung 6.16.....	358
	Lösung 6.17.....	360
	Lösung 6.18.....	361
	Lösung 6.19.....	362
	Lösung 6.20	367
	Lösung 6.21.....	371
	Lösung 6.22	375
7	Innere Klassen	381
7.1	Die Definition von inneren Klassen und deren Instanzen	381
	Aufgabe 7.1: Instantiieren von Member-Klassen innerhalb der umgebenden Klasse	382
	Aufgabe 7.2: Instantiieren von Member-Klassen außerhalb der umgebenden Klasse.....	383
	Aufgabe 7.3: Instantiieren von Static-Member- Klassen innerhalb der umgebenden Klasse	383
	Aufgabe 7.4: Instantiieren von Static-Member- Klassen außerhalb der umgebenden Klasse.....	384
	Aufgabe 7.5: Lokale Klassen.....	384
	Aufgabe 7.6: Anonyme Klassen.....	385

	Aufgabe 7.7: Member-Interfaces	385
	Aufgabe 7.8: Member-Interface mittels einer anonymen Klasse implementieren	386
	Aufgabe 7.9: Member-Interface von einer anderen Klasse implementieren.....	386
	Aufgabe 7.10: Member-Interfaces und Member- Klassen.....	387
7.2	Innere Klassen am Beispiel von Event-Listener und Event-Adapter.....	387
	Aufgabe 7.11: Den WindowAdapter als Member-Klasse definieren.....	387
	Aufgabe 7.12: Den WindowAdapter mittels einer anonymen Klasse implementieren	388
	Aufgabe 7.13: Den ActionListener mittels einer anonymen Klasse implementieren	388
	Aufgabe 7.14: Benutzdefinierte Event-Objekte und Event-Listener für JButton- und JTextField- Komponenten	388
	Aufgabe 7.15: Benutzdefinierte Event-Objekte und Event-Listener für JRadioButton-Komponenten	389
	Aufgabe 7.16: Benutzdefinierte Event-Objekte und Event-Listener für JLabel-Komponenten	389
7.3	Weitere Beispiele mit inneren Klassendefinitionen	390
	Aufgabe 7.17: Die Methoden der Klasse JOptionPane aus einer anonymen Klasse aufrufen	391
	Aufgabe 7.18: Einen Farbauswahldialog und die main()-Methode innerhalb von inneren Klassen definieren	391
	Aufgabe 7.19: Wiederholungsaufgabe	392
7.4	Lösungen:.....	393
	Lösung 7.1	393
	Lösung 7.2	395
	Lösung 7.3	398
	Lösung 7.4	399
	Lösung 7.5	402
	Lösung 7.6.....	404
	Lösung 7.7	405
	Lösung 7.8	406
	Lösung 7.9.....	407
	Lösung 7.10	409

	Lösung 7.11	411
	Lösung 7.12	412
	Lösung 7.13	412
	Lösung 7.14	413
	Lösung 7.15	418
	Lösung 7.16	421
	Lösung 7.17	425
	Lösung 7.18	427
	Lösung 7.19	429
8	Generics	433
8.1	Die Generizität	433
8.2	Generische Klassen und Interfaces	434
	Aufgabe 8.1: Generischer Datentyp als Behälter für die Instanzen vom Typ des Klassenparameters	435
	Aufgabe 8.2: Generischer Datentyp als »Über-Typ« für die Instanzen vom Typ des Klassenparameters	435
	Aufgabe 8.3: Generischer Stack	436
8.3	Wildcardtypen	436
	Aufgabe 8.4: Ungebundene Wildcardtypen	437
	Aufgabe 8.5: Obere Schranke (»upper bound wildcard«) für Wildcardtypen	437
	Aufgabe 8.6: Untere Schranke (»lower bound wildcard«) für Wildcardtypen	438
8.4	Legacy Code, Erasure und Raw-Typen	439
	Aufgabe 8.7: Raw-Typen am Beispiel einer generischen Klasse mit zwei Typparametern	440
	Aufgabe 8.8: Generische Interfaces	441
	Aufgabe 8.9: Brückenmethoden (»bridge methods«)	441
8.5	Generische Arrays	442
	Aufgabe 8.10: Erzeugen von generischen Arrays	442
8.6	Generische Methoden	443
	Aufgabe 8.11: Generische Methodendefinitionen	443
8.7	for-each-Schleifen für Collections	444
	Aufgabe 8.12: Generische Arrays in generischen Methodendefinitionen	444
8.8	Generische Standard-Klassen und -Interfaces	445
	Aufgabe 8.13: Die Klasse ArrayList<E> und die Schnittstelle List<E>	446

	Aufgabe 8.14: Die Klasse <code>Vector<E></code> und die Schnittstelle <code>Collection<E></code>	446
	Aufgabe 8.15: Die Klasse <code>TreeMap<K,V></code>	447
	Aufgabe 8.16: Wiederholungsaufgabe	448
8.9	Enumerationen und die generische Klasse <code>Enum<E extends Enum<E>></code>	448
	Aufgabe 8.17: Die Definition von Enumerationen	449
	Aufgabe 8.18: Konstruktoren und Methoden von <code>enum</code> -Klassen	449
8.10	Die Interfaces <code>Enumeration<E></code> , <code>Iterable<T></code> und <code>Iterator<E></code> sowie <code>Map<K,V></code> und <code>Set<E></code>	450
	Aufgabe 8.19: Weitere generische Schnittstellen	450
8.11	Die Einträge der <code>UIDefaults</code> -Tabelle als Instanz der Klasse <code>Hashtable<Object, Object></code>	451
	Aufgabe 8.20: Das Ändern der <code>font</code> -Eigenschaft von <code>Swing</code> -Komponenten	451
8.12	Die generischen Klassen <code>Class<T></code> und <code>Constructor<T></code> und das »dynamische« Erzeugen von Objekten	452
8.13	Das <code>Reflection</code> -API	452
	Aufgabe 8.21: Die Klasse <code>Class<T></code>	456
	Aufgabe 8.22: Die Klasse <code>Constructor<T></code>	457
	Aufgabe 8.23: Erzeugen von generischen Arrays mit Hilfe eines <code>Class</code> -Objekts	458
	Aufgabe 8.24: Mit <code>Reflection</code> Informationen zu Klassen, Oberklassen und Interfaces holen	459
	Aufgabe 8.25: Das Interface <code>GenericDeclaration</code> und die Unterinterfaces von <code>Type</code>	460
8.14	Definition von benutzerdefinierten Modellen, die generische Klassen und generische Interfaces benutzen	462
	Aufgabe 8.26: Ein benutzerdefiniertes <code>UIDefaults</code> - <code>Tree</code> -Modell, das die Eigenschaften von <code>JButton</code> -, <code>JList</code> - und <code>JTree</code> -Komponenten speichert	462
	Aufgabe 8.27: Die Syntax der <code>firexxx</code> -Methodensignaturen von <code>TreeModel</code> -Klassen	463
	Aufgabe 8.28: Ein <code>UIDefaults</code> - <code>Tree</code> -Modell, das die <code>String</code> -Eigenschaften von <code>Swing</code> -Komponenten speichert	464
	Aufgabe 8.29: Benutzerdefiniertes <code>List</code> -Modell ohne <code>Event</code> -Behandlung	465
	Aufgabe 8.30: Benutzung des <code>List</code> -Modells für eine <code>Viewer</code> -Komponente vom Typ <code>JList</code>	466

	Aufgabe 8.31: Benutzerdefiniertes List-Modell mit Eventbehandlung	466
	Aufgabe 8.32: Benutzung des List-Modells für eine Viewer-Komponente vom Typ JList	467
	Aufgabe 8.33: Vervollständigung der Lösung für ein benutzerdefiniertes List-Modell	467
8.15	Lösungen	468
	Lösung 8.1	468
	Lösung 8.2	469
	Lösung 8.3	470
	Lösung 8.4	472
	Lösung 8.5	473
	Lösung 8.6	475
	Lösung 8.7	476
	Lösung 8.8	480
	Lösung 8.9	481
	Lösung 8.10	482
	Lösung 8.11	483
	Lösung 8.12	485
	Lösung 8.13	486
	Lösung 8.14	487
	Lösung 8.15	489
	Lösung 8.16	490
	Lösung 8.17	492
	Lösung 8.18	493
	Lösung 8.19	496
	Lösung 8.20	499
	Lösung 8.21	500
	Lösung 8.22	502
	Lösung 8.23	505
	Lösung 8.24	506
	Lösung 8.25	511
	Lösung 8.26	518
	Lösung 8.27	522
	Lösung 8.28	526
	Lösung 8.29	530
	Lösung 8.30	532
	Lösung 8.31	534
	Lösung 8.32	537

	Lösung 8.33	539
9	Exceptions und Errors	545
9.1	Ausnahmen auslösen	545
9.2	Ausnahmen abfangen oder weitergeben	546
	Aufgabe 9.1: Unbehandelte RuntimeExceptions	546
	Aufgabe 9. 2: Behandelte RuntimeExceptions	547
	Aufgabe 9.3: Die Weitergabe von Ausnahmen.	547
9.3	Das Verwenden von finally in der Ausnahmebehandlung.	548
	Aufgabe 9.4: Der finally-Block.	548
	Aufgabe 9.5: Geschachtelte try/catch-Blöcke	549
9.4	Ausnahmen manuell auslösen	550
	Aufgabe 9.6: Standard-Ausnahmen manuell auslösen	550
9.5	Exception-Unterklassen erzeugen	551
	Aufgabe 9.7: Benutzerdefinierte Ausnahmen manuell auslösen	551
	Aufgabe 9.8: Wiederholungsaufgabe	551
9.6	Ketten von Ausnahmen	553
	Aufgabe 9.9: Exception-Ketten	553
9.7	Die Ausnahmen bei einem Wechsel von LookAndFeel- Komponenten	554
	Aufgabe 9.10: Die LookAndFeel-spezifischen Einträge der UIDefaults-Tabelle	554
	Aufgabe 9.11: Wiederholungsaufgabe	555
9.8	Lösungen	556
	Lösung 9.1	556
	Lösung 9.2	557
	Lösung 9.3	558
	Lösung 9.4	559
	Lösung 9.5	561
	Lösung 9.6	565
	Lösung 9.7	566
	Lösung 9.8	568
	Lösung 9.9	572
	Lösung 9.10	575
	Lösung 9.11	583

10	Neue Features von Java 6.0	587
10.1	Ergänzungen im Sprachumfeld und in der Behandlung von Exceptions mit Java 6.0	587
	Aufgabe 10.1: Der GLOBAL_LOGGER_NAME, Array-Copies, Empty-Strings und throw in einem catch-Block	587
10.2	Die neuen displayName[s] Methoden der Klasse Calendar	588
	Aufgabe 10.2: Die Klassen Calendar und Date	589
10.3	Die Interfaces NavigableMaps und NavigableSets	590
	Aufgabe 10.3: Die Methoden der Interfaces NavigableMap und NavigableSet	590
10.4	Splash Screens	590
	Aufgabe 10.4: Einen Splash Screen setzen	591
	Aufgabe 10.5: Auf einen Splash Screen zeichnen	591
10.5	Der Dialog-ModalityType	592
	Aufgabe 10.6: Die Dialog.ModalityType-Option	593
10.6	Sortieren und Filtern von Tabelleneinträgen	594
	Aufgabe 10.7: Das Sortieren von Tabelleneinträgen mit Hilfe der Klasse TableRowSorter<TableModel> ...	594
	Aufgabe 10.8: Das Sortieren von Tabelleneinträgen unter Benutzung eines benutzerdefinierten Table-Modells	595
	Aufgabe 10.9: Filtern von Tabelleneinträge mit Hilfe von Standard-Filterklassen	595
	Aufgabe 10.10: Einen benutzerdefinierten Filter erzeugen	596
	Aufgabe 10.11: Wiederholungsaufgabe	597
10.7	Komponenten für die Registerkarten (Tabs) von JTabbedPane-Instanzen nutzen	597
	Aufgabe 10.12 : JTabbedPane-Registerkarten mit einer String-Beschriftung	598
	Aufgabe 10.13: JTabbedPane-Registerkarten mit Komponenten	598
10.8	Drag-and-Drop-Unterstützung in Java	599
	Aufgabe 10.14: Drag-and-Drop-Test für JTextArea- und JList-Komponenten	600
	Aufgabe 10.15: Eigenschaften für den Transfer definieren	601

	Aufgabe 10.16: Drag-and-Drop-Test für JLabel-Komponenten	602
	Aufgabe 10.17: Die Klassen Clipboard, StringSelection und DataFlavor	602
	Aufgabe 10.18: Das Interface DropTargetListener und die Klasse DropTargetEvent	603
10.9	Benutzerdefinierte Transfer-Handler	605
	Aufgabe 10.19: Benutzerdefinierter Transfer-Handler für JList-Komponenten	605
	Aufgabe 10.20: Benutzerdefinierter Transfer-Handler für JTree-Komponenten	606
10.10	Die neuen Drag-and-Drop-Klassen aus der Java-Version 6.0	607
	Aufgabe 10.21: Eine neue Betrachtung der Drop-Fähigkeit für JList-Komponenten	608
	Aufgabe 10.22: Eine neue Betrachtung der Drop-Fähigkeit für JTree-Komponenten	609
10.11	Lösungen	610
	Lösung 10.1	610
	Lösung 10.2	612
	Lösung 10.3	614
	Lösung 10.4	616
	Lösung 10.5	617
	Lösung 10.6	618
	Lösung 10.7	620
	Lösung 10.8	622
	Lösung 10.9	623
	Lösung 10.10	626
	Lösung 10.11	628
	Lösung 10.12	630
	Lösung 10.13	631
	Lösung 10.14	634
	Lösung 10.15	635
	Lösung 10.16	636
	Lösung 10.17	637
	Lösung 10.18	640
	Lösung 10.19	643
	Lösung 10.20	646
	Lösung 10.21	650
	Lösung 10.22	653

II	Neue Features von Java 7	657
II.1	Strings in switch-Anweisungen und neue Literale mit Java 7	657
	Aufgabe II.1: Binäre Literale, Underscores in numerischen Literalen und Strings in switch-Anweisungen	658
II.2	Typen in Java	658
II.3	Typprüfung und Typsicherheit mittels Generics	662
II.4	Subtyping für parametrisierte Typen	668
II.5	Die extends-Klausel	669
II.6	Typinferenz für Methoden	670
II.7	Typinferenz beim Erzeugen von Instanzen eines generischen Typs	671
II.8	Heap Pollution	673
II.9	Wildcard-Capture	675
	Aufgabe II.2: Typinferenz beim Instantiieren von generischen Klassen	676
	Aufgabe II.3: Der Diamond-Operator	677
	Aufgabe II.4: Schranken für Typvariablen und Typinferenz für Methoden	678
	Aufgabe II.5: Parametrisierte Typen und Wildcardtypen	681
	Aufgabe II.6: Generische Arraytypen	685
	Aufgabe II.7: Subtyping von Referenztypen	689
II.10	Multi-catch-Klausel und verbesserte Typprüfung beim Rethrowing von Exceptions	690
	Aufgabe II.8: Disjunction-Typ für Exceptions	691
	Aufgabe II.9: Typprüfung beim Rethrowing von Exceptions	693
II.11	Transparente und nicht-rechteckige Fenster mit Java 7 erzeugen ...	695
	Aufgabe II.10: Fensterdekorationen	696
	Aufgabe II.11: Transparente und nicht-rechteckige Fenster	697
II.12	Das Überlappen von LW- und HW-Komponenten mit Java 7	699
	Aufgabe II.12: LW- und HW-Komponenten überlappen	699
II.13	Das Nimbus-LookAndFeel	700
	Aufgabe II.13: Das Nimbus-LookAndFeel und das Skinning von Komponenten	702
II.14	Swing-Komponenten mit einem JLayer dekorieren	704

	Aufgabe 11.14: Das Dekorieren von Swing-Komponenten	705
	Aufgabe 11.15: JLayer- und LayerUI-Instanzen	706
	Aufgabe 11.16: Event-Handling für JLayer- und LayerUI-Instanzen	707
11.15	Andere Erweiterungen	709
11.16	Lösungen	711
	Lösung 11.1	711
	Lösung 11.2	714
	Lösung 11.3	718
	Lösung 11.4	723
	Lösung 11.5	729
	Lösung 11.6	746
	Lösung 11.7	752
	Lösung 11.8	759
	Lösung 11.9	761
	Lösung 11.10	765
	Lösung 11.11	767
	Lösung 11.12	773
	Lösung 11.13	777
	Lösung 11.14	782
	Lösung 11.15	784
	Lösung 11.16	787
	Stichwortverzeichnis	793