

Inhaltsverzeichnis

1	Einleitung	1
1.1	Über dieses Buch	1
1.1.1	Motivation	1
1.1.2	Was leistet dieses Buch und was nicht?	2
1.1.3	Wie und was soll mithilfe des Buchs gelernt werden?	2
1.1.4	Wer sollte dieses Buch lesen?	4
1.2	Aufbau des Buchs	4
1.2.1	Gliederung des Buchs	5
1.2.2	Kapitelübersicht	7
1.3	Konventionen	12
1.4	Danksagungen	14

I Java-Grundlagen, Analyse und Design **17**

2	Professionelle Arbeitsumgebung	19
2.1	Vorteile von IDEs am Beispiel von Eclipse	19
2.2	Projektorganisation	20
2.3	Einsatz von Versionsverwaltungen	22
2.3.1	Arbeiten mit Versionsverwaltungen	25
2.3.2	Tagging und Branching	26
2.3.3	CVS und SVN im Vergleich	29
2.4	Einsatz eines Unit-Test-Frameworks	30
2.4.1	Das JUnit-Framework	31
2.4.2	Schreiben und Ausführen von Tests	33
2.5	Debugging	36
2.5.1	Fehlersuche mit einem Debugger	37
2.5.2	Remote Debugging	41
2.6	Einsatz eines IDE-unabhängigen Build-Prozesses	46
2.6.1	Motivation für ein IDE-unabhängiges Build-System	46
2.6.2	Ant-Build	47
2.7	Weiterführende Literatur	56

3	Objektorientiertes Design	57
3.1	OO-Grundlagen	58
3.1.1	Grundbegriffe	58
3.1.2	Beispielentwurf: Ein Zähler	69
3.1.3	Diskussion der OO-Grundgedanken	76
3.1.4	Wissenswertes zum Objektzustand	80
3.2	Grundlegende OO-Techniken	89
3.2.1	Abstrakte Basisklassen	89
3.2.2	Schnittstellen (Interfaces)	91
3.2.3	Interfaces und abstrakte Basisklassen	93
3.3	Vom imperativen zum objektorientierten Entwurf	94
3.4	Fortgeschrittenere OO-Techniken	99
3.4.1	Read-only-Interface	99
3.4.2	Immutable-Klasse	105
3.4.3	Probleme durch Vererbung	110
3.4.4	Delegation statt Vererbung	115
3.4.5	Marker-Interface	119
3.4.6	Konstantensammlungen, Aufzählungen und das Enum-Muster	120
3.4.7	Value Object	126
3.5	Formen der Varianz	128
3.5.1	Grundlagen der Varianz	128
3.5.2	Kovariante Rückgabewerte	132
3.6	Generische Typen (Generics)	134
3.6.1	Einführung	134
3.6.2	Generics und Auswirkungen der Type Erasure	139
3.6.3	Containerklassen: Generics und Varianz	146
3.7	Weiterführende Literatur	158
4	Java-Grundlagen	159
4.1	Die Klasse <code>Object</code>	159
4.1.1	Die Methode <code>toString()</code>	161
4.1.2	Die Methode <code>equals()</code>	165
4.2	Primitive Datentypen und Wrapper-Klassen	177
4.2.1	Konvertierung von Werten	178
4.2.2	Ausgabe und Verarbeitung von Zahlen	184
4.3	Stringverarbeitung	187
4.3.1	Die Klasse <code>String</code>	188
4.3.2	Die Klassen <code>StringBuffer</code> und <code>StringBuilder</code>	192
4.3.3	Ausgaben mit <code>format()</code> und <code>printf()</code>	196
4.3.4	Die Klasse <code>StringTokenizer</code>	197
4.3.5	Die Methode <code>split()</code> und das 1x1 der regulären Ausdrücke	198

4.4	Datumsverarbeitung	203
4.4.1	Fallstricke der Datums-APIs	204
4.4.2	Das <code>Date</code> -API	205
4.4.3	Das <code>Calendar</code> -API	208
4.5	Interfaces und innere Klassen	210
4.5.1	Interfaces	210
4.5.2	Varianten innerer Klassen	211
4.5.3	Designbeispiel mit inneren Klassen und Interfaces	214
4.5.4	Lokal definierte Klassen und Interfaces	216
4.6	Ein- und Ausgabe (I/O)	219
4.6.1	Dateibehandlung und die Klasse <code>File</code>	219
4.6.2	Ein- und Ausgabestreams	226
4.6.3	Speichern und Laden von Daten und Objekten	233
4.6.4	Grundlagen der Netzwerkprogrammierung	240
4.7	Fehlerbehandlung	245
4.7.1	Exception Handling	247
4.7.2	Assertions	253
4.8	Weiterführende Literatur	256

II Bausteine stabiler Java-Applikationen 257

5	Das Collections-Framework	259
5.1	Datenstrukturen	259
5.1.1	Wahl einer geeigneten Datenstruktur	260
5.1.2	Arrays	262
5.1.3	Das Interface <code>Collection</code>	264
5.1.4	Listen und das Interface <code>List</code>	269
5.1.5	Mengen und das Interface <code>Set</code>	275
5.1.6	Grundlagen von hashbasierten Containern	277
5.1.7	Grundlagen automatisch sortierender Container	286
5.1.8	Die Methoden <code>equals()</code> , <code>hashCode()</code> und <code>compareTo()</code> im Zusammenspiel	293
5.1.9	Konkrete Realisierungen von Mengen	295
5.1.10	Schlüssel-Wert-Abbildungen und das Interface <code>Map</code>	298
5.1.11	Erweiterungen am Beispiel der Klasse <code>HashMap</code>	305
5.1.12	Entscheidungshilfe zur Wahl von Datenstrukturen	308
5.2	Suchen, Sortieren und Filtern	309
5.2.1	Suchen	310
5.2.2	Sortieren mit Komparatoren	312
5.2.3	Filtern von Collections	318
5.3	Utility-Klassen und Hilfsmethoden	324
5.3.1	Nützliche Hilfsmethoden	325

5.3.2	Dekorierer <code>synchronized</code> , <code>unmodifiable</code> und <code>checked</code>	326
5.3.3	Vordefinierte Algorithmen	331
5.3.4	Design eines Zugriffsinterface	334
5.4	Probleme im Collections-Framework	338
5.4.1	Merkwürdigkeiten in Arrays	338
5.4.2	Probleme von <code>Stack</code> , <code>Queue</code> und <code>Deque</code>	340
5.5	Weiterführende Literatur	343
6	Applikationsbausteine	345
6.1	Einsatz von Bibliotheken am Beispiel	345
6.2	Wertebereichs- und Parameterprüfungen	350
6.2.1	Prüfung einfacher Wertebereiche und Wertemengen	350
6.2.2	Prüfung komplexerer Wertebereiche	354
6.3	Logging-Frameworks	362
6.3.1	Apache <code>log4j</code>	363
6.3.2	Tipps und Tricks zum Einsatz von Logging mit <code>log4j</code>	370
6.4	Utility-Klassen zur Dateibehandlung	378
6.4.1	Die Klasse <code>FileUtils</code>	378
6.4.2	Die Klasse <code>StreamUtils</code>	380
6.4.3	Implementierung von <code>StringStreams</code>	385
6.5	Konfigurationsparameter und -dateien	387
6.5.1	Einlesen von Kommandozeilenparametern	387
6.5.2	Verarbeitung von <code>Properties</code>	396
6.5.3	Die Klasse <code>Preferences</code>	402
6.5.4	Weitere Möglichkeiten zur Konfigurationsverwaltung	404
7	Multithreading	411
7.1	Threads und <code>Runnable</code> s	413
7.1.1	Definition der auszuführenden Aufgabe	413
7.1.2	Start, Ausführung und Ende von Threads	414
7.1.3	Lebenszyklus von Threads und Thread-Zustände	418
7.2	Zusammenarbeit von Threads	423
7.2.1	Konkurrierende Datenzugriffe	423
7.2.2	Locks, Monitore und kritische Bereiche	425
7.2.3	Deadlocks und Starvation	429
7.2.4	Kritische Bereiche und das Interface <code>Lock</code>	431
7.3	Kommunikation von Threads	435
7.3.1	Kommunikation mit Synchronisation	436
7.3.2	Kommunikation über die Methoden <code>wait()</code> , <code>notify()</code> und <code>notifyAll()</code>	439
7.3.3	Abstimmung von Threads	448
7.3.4	Unerwartete <code>IllegalMonitorStateException</code> s	452
7.4	Das Java-Memory-Modell	453

7.4.1	Sichtbarkeit	454
7.4.2	Atomarität	455
7.4.3	Reorderings	457
7.5	Besonderheiten bei Threads	460
7.5.1	Verschiedene Arten von Threads	460
7.5.2	Exceptions in Threads	462
7.5.3	Sicheres Beenden von Threads	463
7.5.4	Zeitgesteuerte Ausführung	467
7.6	Die Concurrency Utilities	470
7.6.1	Concurrent Collections	471
7.6.2	Das Executor-Framework	480
7.7	Weiterführende Literatur	491
8	Fortgeschrittene Java-Themen	493
8.1	Crashkurs Reflection	493
8.1.1	Grundlagen	495
8.1.2	Zugriff auf Methoden und Attribute	498
8.1.3	Spezialfälle	502
8.2	Annotations	506
8.2.1	Einführung in Annotations	506
8.2.2	Standard-Annotations des JDKs	508
8.2.3	Definition eigener Annotations	510
8.2.4	Annotation zur Laufzeit auslesen	512
8.3	Serialisierung	514
8.3.1	Implementieren der Serialisierung	514
8.3.2	Die Serialisierung anpassen	518
8.3.3	Versionsverwaltung der Serialisierung	522
8.3.4	Optimierung der Serialisierung	525
8.4	Objekt kopien und das Interface <code>Cloneable</code>	530
8.4.1	Das Interface <code>Cloneable</code>	531
8.4.2	Alternativen zur Methode <code>clone()</code>	540
8.5	Internationalisierung	542
8.5.1	Grundlagen	543
8.5.2	Die Klasse <code>Locale</code>	544
8.5.3	Die Klasse <code>PropertyResourceBundle</code>	547
8.5.4	Formatierte Ein- und Ausgabe	551
8.5.5	Zahlen und die Klasse <code>NumberFormat</code>	552
8.5.6	Datumswerte und die Klasse <code>DateFormat</code>	555
8.5.7	Textmeldungen und die Klasse <code>MessageFormat</code>	560
8.5.8	Stringvergleiche mit der Klasse <code>Collator</code>	562
8.6	Programmbausteine zur Internationalisierung	567
8.6.1	Unterstützung mehrerer Datumsformate	568
8.6.2	Nutzung mehrerer Sprachdateien	572

8.7	Garbage Collection	582
8.7.1	Einflussfaktoren auf die Garbage Collection	582
8.7.2	Algorithmen zur Garbage Collection	586
8.7.3	Optimierungen der Garbage Collection	588
8.7.4	Memory Leaks: Gibt es die auch in Java?!	589
8.7.5	Objektzerstörung und <code>finalize()</code>	591
8.8	Weiterführende Literatur	594
9	Programmierung grafischer Benutzeroberflächen	595
9.1	Grundlagen zu grafischen Oberflächen	596
9.1.1	Überblick: Bedienelemente und Container	600
9.1.2	Einführung in das Layoutmanagement	603
9.1.3	Komplexere Layouts durch Kombination von Layoutmanagern	608
9.1.4	Grundlagen zur Ereignisbehandlung	613
9.1.5	Gebräuchliche Event Listener	619
9.1.6	Varianten der Ereignisverarbeitung	628
9.2	Multithreading und Swing	633
9.2.1	Crashkurs Event Handling in Swing	634
9.2.2	Ausführen von Aktionen	635
9.2.3	Die Klasse <code>SwingWorker</code>	639
9.3	Zeichnen in GUI-Komponenten	643
9.3.1	Generelles zum Zeichnen in GUI-Komponenten	643
9.3.2	<code>JTextField</code> mit Markierungslinien	652
9.3.3	Einführung in Java 2D	660
9.3.4	Bedienelemente mit Java 2D selbst erstellen	668
9.4	Komplexe Bedienelemente	676
9.4.1	Grundlagen	676
9.4.2	Die Klasse <code>JList</code>	685
9.4.3	Die Klasse <code>JTable</code>	700
9.4.4	Die Klasse <code>JTree</code>	723
9.5	Weiterführende Literatur	734
10	Einführung in Persistenz und Datenbanken	737
10.1	Grundlagen zur Persistenz	738
10.1.1	Beschränkungen einfacher Persistenzlösungen	739
10.1.2	Modelle zur Persistierung von Objekten	741
10.1.3	Wissenswertes zur Speicherung von Daten in Tabellen relationaler Datenbanken	742
10.1.4	Abbildung zwischen Objekt- und Datenbank-Modell	750
10.1.5	Die Datenbanksysteme Java DB und HSQLDB	761
10.1.6	SQL-Grundlagen	765
10.2	Datenbankzugriffe per JDBC	778

10.2.1	Schritte zur Abfrage von Datenbanken	780
10.2.2	Umgang mit Fehlern	787
10.2.3	Besonderheiten von <code>ResultSet</code>	792
10.2.4	Abfrage von Metadaten	797
10.2.5	Probleme bei der Ausführung von <code>Statements</code>	806
10.2.6	Das Interface <code>PreparedStatement</code>	809
10.2.7	Transaktionen in JDBC	812
10.3	Grundlagen zum ORM mit JDBC	815
10.3.1	Rekonstruktion von Objekten	815
10.3.2	Zugriffe mit einem Data Access Object (DAO)	820
10.4	Datenbanken und JPA	822
10.4.1	Grundlagen zum ORM und zum Java Persistence API ...	823
10.4.2	Persistente Klassen	825
10.4.3	Datenbankzugriffe per JPA	829
10.4.4	DAO-Funktionalität und die Klasse <code>EntityManager</code>	833
10.4.5	Spezialfälle des ORMs per JPA	837
10.5	Weiterführende Literatur	840
11	Neuerungen in JDK 7	841
11.1	Erweiterungen der Sprache selbst	841
11.2	Erweiterungen des NIO in JDK 7	849
11.2.1	Dateibehandlung in JDK 7	850
11.2.2	Asynchronous I/O	855
11.3	Multithreading	856
11.4	Neuerungen in AWT und Swing	859
11.5	Collections	861
11.6	Der Garbage Collector »G1«	862

III Fallstricke und Lösungen im Praxisalltag 863

12	Bad Smells	865
12.1	Programmdesign	867
12.1.1	Bad Smell: Verwenden von Magic Numbers	867
12.1.2	Bad Smell: Konstanten in Interfaces definieren	868
12.1.3	Bad Smell: <code>System.exit()</code> mitten im Programm	871
12.1.4	Bad Smell: Zusammengehörende Konstanten nicht als Typ definiert	872
12.1.5	Bad Smell: Programmcode im Logging-Code	874
12.1.6	Bad Smell: Unvollständige Betrachtung aller Alternativen .	875
12.1.7	Bad Smell: Unvollständige Änderungen nach Copy-Paste	876
12.1.8	Bad Smell: Casts auf unbekannte Subtypen	878
12.1.9	Bad Smell: Pre-/Post-Increment in komplexeren Statements	880

12.1.10	Bad Smell: Keine Klammern um Blöcke	882
12.1.11	Bad Smell: Variablendeklaration nicht im kleinstmöglichen Sichtbarkeitsbereich	884
12.1.12	Bad Smell: Mehrere aufeinanderfolgende Parameter gleichen Typs	885
12.1.13	Bad Smell: Grundloser Einsatz von Reflection	886
12.2	Klassendesign	888
12.2.1	Bad Smell: Unnötigerweise veränderliche Attribute	888
12.2.2	Bad Smell: Aufruf abstrakter Methoden im Konstruktor ...	890
12.2.3	Bad Smell: Herausgabe von <code>this</code> im Konstruktor	894
12.2.4	Bad Smell: Referenzierung von Subklassen in Basisklassen	895
12.2.5	Bad Smell: Mix abstrakter und konkreter Basisklassen ...	897
12.2.6	Bad Smell: Öffentlicher Defaultkonstruktor lediglich zum Zugriff auf Hilfsmethoden.....	899
12.3	Fehlerbehandlung und Exception Handling	901
12.3.1	Bad Smell: Unbehandelte Exception	901
12.3.2	Bad Smell: Unpassender Exception-Typ	902
12.3.3	Bad Smell: Exceptions zur Steuerung des Kontrollflusses	904
12.3.4	Bad Smell: Fangen der allgemeinsten Exception	905
12.3.5	Bad Smell: Rückgabe von <code>null</code> statt Exception im Fehlerfall	907
12.3.6	Bad Smell: Unbedachte Rückgabe von <code>null</code>	909
12.3.7	Bad Smell: Sonderbehandlung von Randfällen	911
12.3.8	Bad Smell: Keine Gültigkeitsprüfung von Eingabeparametern	912
12.3.9	Bad Smell: Fehlerhafte Fehlerbehandlung	914
12.3.10	Bad Smell: I/O ohne <code>finally</code> bzw. <code>finalize()</code>	916
12.3.11	Bad Smell: Resource Leaks durch Exceptions im Konstruktor	918
12.4	Häufige Fallstricke.....	922
12.5	Weiterführende Literatur	930
13	Refactorings.....	931
13.1	Das Standardvorgehen	939
13.2	Der Refactoring-Katalog	942
13.2.1	Reduziere die Sichtbarkeit von Attributen	942
13.2.2	Minimiere veränderliche Attribute	945
13.2.3	Reduziere die Sichtbarkeit von Methoden	949
13.2.4	Ersetze Mutator- durch Business-Methode	950
13.2.5	Minimiere Zustandsänderungen (Refactoring-Kombination)	951
13.2.6	Führe ein Interface ein	951
13.2.7	Aufspalten eines Interface	952
13.2.8	Einführen eines Read-only-Interface	953

13.2.9	Einführen eines Read-Write-Interface	953
13.2.10	Einführen von Convenience-Methoden	954
13.2.11	Einführen einer Zustandsprüfung	956
13.2.12	Überprüfung von Eingabeparametern	958
13.2.13	Trenne Informationsbeschaffung und -verarbeitung	962
13.2.14	Konstantensammlung in <code>enum</code> umwandeln	967
13.2.15	Entferne Exceptions zur Steuerung des Kontrollflusses... ..	970
13.2.16	Umwandlung in Utility-Klasse mit statischen Hilfsmethoden ..	973
13.3	Weiterführende Literatur	976
14	Entwurfsmuster	977
14.1	Erzeugungsmuster	980
14.1.1	Erzeugungsmethode	980
14.1.2	Fabrikmethode (Factory method)	983
14.1.3	Erbauer (Builder)	986
14.1.4	Singleton	989
14.1.5	Prototyp (Prototype)	993
14.2	Strukturmuster	997
14.2.1	Fassade (Façade)	998
14.2.2	Adapter	1000
14.2.3	Dekorierer (Decorator)	1002
14.2.4	Kompositum (Composite)	1005
14.3	Verhaltensmuster	1010
14.3.1	Iterator	1010
14.3.2	Null-Objekt (Null Object)	1012
14.3.3	Schablonenmethode (Template method)	1015
14.3.4	Strategie (Strategy)	1019
14.3.5	Befehl (Command)	1027
14.3.6	Proxy	1034
14.3.7	Zuständigkeitskette (Chain of Responsibility)	1036
14.3.8	Beobachter (Observer)	1038
14.3.9	MVC-Architektur	1047
14.4	Weiterführende Literatur	1048

IV Qualitätssicherungsmaßnahmen	1051
--	-------------

15	Programmierstil und Coding Conventions	1053
15.1	Grundregeln eines guten Programmierstils	1053
15.1.1	Keep It Human-Readable	1054
15.1.2	Keep It Simple And Short	1054
15.1.3	Keep It Natural	1054
15.1.4	Keep It Clean	1054

15.2	Die Psychologie beim Sourcecode-Layout	1055
15.2.1	Faktor der Ähnlichkeit	1055
15.2.2	Faktor der Nähe	1056
15.3	Coding Conventions	1058
15.3.1	Grundlegende Namens- und Formatierungsregeln	1059
15.3.2	Namensgebung	1062
15.3.3	Dokumentation	1064
15.3.4	Programmdesign	1066
15.3.5	Klassendesign	1071
15.3.6	Parameterlisten	1074
15.3.7	Logik und Kontrollfluss	1076
15.4	Sourcecode-Überprüfung mit Tools	1078
15.4.1	Metriken	1079
15.4.2	Sourcecode-Überprüfung im Build-Prozess	1083
16	Unit Tests	1091
16.1	Überblick	1091
16.1.1	Arten von Tests	1091
16.1.2	Äußere vs. innere Qualität	1094
16.1.3	Auswirkungen von Unit Tests auf die Qualität	1095
16.2	Motivation für Unit Tests aus der Praxis	1097
16.2.1	Unit Tests für Neuentwicklungen	1097
16.2.2	Unit Tests und Legacy-Code	1104
16.3	Fortgeschrittene Unit-Test-Techniken	1115
16.3.1	Testen mit Stubs	1115
16.3.2	Testen mit Mocks	1117
16.3.3	Unit Tests von privaten Methoden	1120
16.4	Unit Tests mit Threads und Timing	1121
16.5	Nützliche Tools für Unit Tests	1126
16.5.1	Hamcrest	1126
16.5.2	Infinittest	1130
16.5.3	Cobertura	1131
16.6	Weiterführende Literatur	1137
17	Codereviews	1139
17.1	Definition	1139
17.2	Probleme und Tipps zur Durchführung	1141
17.3	Vorteile von Codereviews	1143
17.4	Codereview-Tools	1146
17.5	Codereview-Checkliste	1148

18	Optimierungen	1149
18.1	Grundlagen	1150
18.1.1	Optimierungsebenen und Einflussfaktoren	1151
18.1.2	Optimierungstechniken	1152
18.1.3	CPU-bound-Optimierungsebenen am Beispiel	1154
18.1.4	Messungen – Erkennen kritischer Bereiche	1158
18.1.5	Abschätzungen mit der O-Notation	1165
18.2	Einsatz geeigneter Datenstrukturen	1168
18.2.1	Einfluss von Arrays und Listen	1169
18.2.2	Optimierungen für <code>Set</code> und <code>Map</code>	1173
18.2.3	API-Design <code>Collection</code> vs. <code>Iterator</code>	1175
18.3	Lazy Initialization	1176
18.3.1	Lazy Initialization am Beispiel	1177
18.3.2	Konsequenzen des Einsatzes der Lazy Initialization	1180
18.3.3	Lazy Initialization mithilfe des <code>PROXY</code> -Musters	1182
18.4	Optimierungen am Beispiel	1184
18.5	I/O-bound-Optimierungen	1192
18.5.1	Technik – Wahl passender Strategien	1192
18.5.2	Technik – Caching und Pooling	1195
18.5.3	Technik – Vermeidung unnötiger Aktionen	1196
18.6	Memory-bound-Optimierungen	1199
18.6.1	Technik – Wahl passender Strategien	1199
18.6.2	Technik – Caching und Pooling	1202
18.6.3	Optimierungen der Stringverarbeitung	1207
18.6.4	Technik – Vermeidung unnötiger Aktionen	1210
18.7	CPU-bound-Optimierungen	1212
18.7.1	Technik – Wahl passender Strategien	1213
18.7.2	Technik – Caching und Pooling	1213
18.7.3	Technik – Vermeidung unnötiger Aktionen	1214
18.8	Weiterführende Literatur	1216

V	Anhang	1217
----------	---------------	-------------

A	Einführung in die UML	1219
A.1	Die UML im Überblick	1219
A.2	Strukturdiagramme – statische Modelle	1223
A.2.1	Klassendiagramme	1223
A.2.2	Objektdiagramme	1227
A.2.3	Komponentendiagramme	1227
A.2.4	Paketdiagramme	1228
A.3	Verhaltensdiagramme – dynamische Modelle	1229
A.3.1	Anwendungsfalldiagramme	1229

A.3.2	Sequenzdiagramme	1230
A.3.3	Kommunikationsdiagramme	1234
A.3.4	Zustandsdiagramme	1235
A.3.5	Aktivitätsdiagramme	1237
A.4	Weiterführende Literatur	1238
B	Überblick über den Softwareentwicklungsprozess	1239
B.1	Vorgehensmodelle	1239
B.1.1	Aufgaben und Phasen beim Softwareentwurf	1239
B.1.2	Wasserfallmodell und V-Modell	1240
B.1.3	Extreme Programming (XP)	1243
B.1.4	Test-Driven Development (TDD)	1244
B.1.5	Diskussion	1245
C	Grundlagen zur Java Virtual Machine	1247
C.1	Wissenswertes rund um die Java Virtual Machine	1247
C.1.1	Ausführung eines Java-Programms	1247
C.1.2	Sicherheit und Speicherverwaltung	1248
C.1.3	Sicherheit und Classloading	1249
	Literaturverzeichnis	1251
	Stichwortverzeichnis	1255