

Java Programmierung Aufbau

Seminarunterlage

Version: 11.11



Dieses Dokument wird durch die ORDIX AG veröffentlicht.

Copyright ORDIX AG. Alle Rechte vorbehalten.

Alle Produkt- und Dienstleistungs-Bezeichnungen sind Warenzeichen oder eingetragene Warenzeichen der jeweiligen Firmen und beziehen sich auf Eintragungen in den USA oder USA-Warenzeichen.

Weitere Logos und Produkt- oder Handelsnamen sind eingetragene Warenzeichen oder Warenzeichen der jeweiligen Unternehmen.

Kein Teil dieser Dokumentation darf ohne vorherige schriftliche Genehmigung der ORDIX AG weitergegeben oder benutzt werden.

Adressen der ORDIX AG

Die ORDIX AG besitzt folgende Geschäftsstellen

ORDIX AG
Karl-Schurz-Str. 19a
D-33100 Paderborn
Tel.: (+49) 0 52 51 / 10 63 - 0
Fax.: (+49) 01 80 / 1 67 34 90

ORDIX AG
An der alten Ziegelei 5
D-48157 Münster
Tel.: (+49) 02 51 / 9 24 35 – 00
Fax.: (+49) 01 80 / 1 67 34 90

ORDIX AG
Welser Straße 9
D-86368 Gersthofen
Tel.: (+49) 08 21 / 507 492 – 0
Fax.: (+49) 01 80 / 1 67 34 90

ORDIX AG
Kreuzberger Ring 13
D-65205 Wiesbaden
Tel.: (+49) 06 11 / 7 78 40 – 00
Fax.: (+49) 01 80 / 1 67 34 90

ORDIX AG
Wikingerstraße 18-20
D-51107 Köln
Tel.: (+49) 02 21 / 8 70 61 – 0
Fax.: (+49) 01 80 / 1 67 34 90

Internet: <http://www.ordix.de>

Email: seminare@ordix.de

Inhaltsverzeichnis

1	Grundlagen	8
1.1	Was ist Java?	9
1.2	Historie	10
1.3	Java-Design Kriterien	11
1.3.1	Einfach und Objektorientiert	12
1.3.2	Verteilt, Interpretiert,	13
1.3.3	Hochleistungsfähig,	14
1.4	Java Virtual Machine (JVM)	15
1.5	*.java und *.class-Datei	16
1.6	Java-Anwendungen	17
1.7	Kompilieren und Starten	18
1.8	Inhalt	19
2	Generics	20
2.1	Java Generics	21
2.2	Collections bisher	22
2.3	Generische Collections	23
2.4	Generische Collections mit mehreren Typparametern	24
2.5	Neuerung in Java 7	25
2.6	Ober- und Unterklassen in generischen Typen	26
2.7	weak und strong typing	28
2.8	Beispiel Collection Framework	29
2.9	Eigene generische Typen	30
2.10	Bounds	33
2.11	TreeMap ohne Bounds	34
2.12	Bounds	36
2.13	TreeMap mit Bounds	37
2.14	Generische Methoden	38
2.14.1	generische Methoden Java 4	39
2.14.2	generische Methoden Java 5	41
2.15	Typ-Inferenz	43
2.16	Wildcard Instanzierung	45
2.17	Wildcard Beispiele	47
2.18	Type Erasure	50
2.19	Generische Typen – Grenzen	52
2.20	Zusammenfassung	55
3	Innere Klassen	56
3.1	Was sind Innere Klassen?	57
3.2	Beispiel: Innere Klassen	58
3.3	Typisierung von Inneren Klassen	59
3.4	Statische innere Klassen	60
3.5	Nicht-statische innere Klassen	62
3.6	Member Klassen	63
3.7	Beispiel: Member Klassen	65
3.8	Lokale Klassen	66
3.9	Beispiel: Lokale Klassen	68
3.10	Anonyme Klassen	69
3.11	Beispiel: Anonyme Klassen	71
3.12	Anwendung: Implementation Hiding	72
3.13	Anwendung: Adapterklassen	73
3.14	Zusammenfassung	74
4	Multithreading	75
4.1	Grundlagen: Concurrency& Threads	76
4.2	Multithreading	77
4.3	Thread Basics	78
4.4	Thread Status	80

4.5	Thread-safe	81
4.6	Deadlock	82
4.7	Immutable Objects	83
4.8	synchronized	84
4.9	Atomic Klassen	85
4.10	Guarded Blöcke	87
4.11	Lock Objects	88
4.12	Wir kommt man zu Multithreading	89
4.13	Package java.util.concurrent	90
4.14	java.util.concurrent: Executor und Co	91
4.15	Executor und Co	92
4.16	ExecutorService	93
4.17	Datenstrukturen: Queues und Co	94
4.18	Queues und Co	95
4.19	BlockingQueue Implementierungen	96
4.20	ArrayBlockingQueue	97
4.21	PriorityBlockingQueue	98
4.22	Datenstrukturen: Concurrent Collections	99
4.23	Concurrent Collections	100
4.24	ConcurrentHashMap	101
4.25	Synchronizers	102
4.26	CountDownLatch	103
4.27	CyclicBarrier	104
4.28	Semaphore	105
4.29	Phaser	106
4.30	Callable und Future	107
4.31	Callable & Future mit ScheduledExecutorService	109
4.32	Swing - Event Dispatcher Thread	110
4.33	GUI Hilfsklasse: SwingWorker<T,V>	111
4.34	SwingWorker Beispiel	112
4.35	Fork/Join	115
4.36	Fork/Join – wichtige Klassen und Interfaces	117
4.37	Fork/Join – Threads und Tasks	119
4.38	Fork/Join Beispiel	120
4.39	Work stealing	123
5	Reflection & Annotations	124
5.1	Reflection API	125
5.2	Klassen der Reflection API	126
5.3	Class-Objekt	127
5.4	Namen und Modifizierer einer Klasse ermitteln	128
5.5	Superklasse und Interfaces ermitteln	129
5.6	Klassen- bzw. Interfaceattribute ermitteln	130
5.7	Konstruktoren ermitteln	132
5.8	Methoden samt Signatur ermitteln	133
5.9	Objekte manipulieren	134
5.10	Objekte über den Default-Konstruktor erzeugen	135
5.11	Objekterzeugung: parametrisierter Konstruktor	136
5.12	Attributwerte ermitteln & setzen	138
5.13	Methoden aufrufen	140
5.14	Was sind Annotations?	141
5.15	Anwendungsgebiete	143
	Anwendungsgebiete: EJB & Co	143
5.16	Annotation Beispiel: @Deprecated	145
5.17	Verwendung von Annotations	147
5.18	Vordefinierte Annotations	149
5.19	Standard Annotations	150
5.20	Definition eigener Annotations-Typen	152
5.21	Annotations mit Eigenschaften	153
5.22	Meta-Annotation Target	158

5.23	Meta-Annotation Retention	161
5.24	Meta-Annotation: Documented	163
5.25	Meta-Annotation Inherited.....	166
5.26	Zugriff auf Annotations über Reflection API.....	168
5.27	AnnotatedElement.....	169
5.28	Beispiel: Zugriff über Reflection	170
6	Typesafe Enums.....	172
6.1	Typesafe Enums	173
6.2	Aufzählungen bisher	174
6.3	Probleme mit int-Enumerations.....	175
6.4	Idiom für typsichere Aufzählungen.....	177
6.5	Typesafe Enums	180
6.6	Deklaration von enum-Klassen	182
6.7	Eigenschaften von enum-Klassen	184
6.8	enum Beispiel.....	187
6.9	Methoden in enum-Klassen	191
6.9.1	switch-Statement.....	193
6.9.2	abstrakte Methoden.....	195
6.10	Zwei neue Klassen für enum.....	196
6.10.1	EnumSet.....	197
6.10.2	EnumMap	201
7	Java Persistence API (JPA).....	204
7.1	Datenbanken	205
7.2	Database Connectivity	206
7.3	JDBC	207
7.4	JDBC DB-Verbindung aufbauen	208
7.5	SQL-Anweisungen - Statement	209
7.6	Auslesen einer Ergebnismenge - ResultSet	211
7.7	JDBC DB-Verbindung aufbauen.....	213
7.8	Vorbereitete Anweisungen - PreparedStatement	214
7.9	JDBC Transaktionen	215
7.10	JDBC Metadaten.....	217
7.11	Zusammenfassung JDBC	219
7.12	Was ist JPA?.....	220
7.13	Persistieren mit der JPA.....	221
7.14	Vor-/Nachteile	230
8	JAXB – XML Verarbeitung in Java.....	231
8.1	Allgemeines.....	232
8.2	Primäre Ziele	233
8.3	Klassengenerierung	234
8.4	Un-/Marshalling bei Data Binding	235
8.5	Zyklus (Round Trip).....	237
8.6	Unmarshal: XML → Java	238
8.7	Marshal: Java → XML.....	239
8.8	Annotationen	240
8.9	Annotationen in JAXB	241
8.10	@XmlElement und @XmlAttribute.....	242
Exkurs JavaBean (aus Wikipedia):.....		242
8.11	@XmlElement und @XmlAttribute, Beispiel	244
8.12	@XmlAccessorType.....	245
8.13	@XmlAccessorType.PUBLIC_MEMBER.....	247
8.14	@XmlAccessorType.PROPERTY	248
8.15	@XmlAccessorType.FIELD.....	249
8.16	@XmlAccessorType.NONE	250
8.17	@XmlTransient	251
8.18	Zusammenpiel Java-Anwendung und XML Data Binding	252
8.19	Generierung und Nutzung.....	253

8.20	Tools.....	254
9	Deklarative Programmierung / Lambda	258
9.1	Programmierparadigma	259
9.2	Deklarative Programmierung	260
9.3	Funktionale Programmierung.....	261
9.4	OOP vs. Funktionale Programmierung	262
9.5	Neues Sprachkonstrukt.....	263
9.6	Beispiel Java Comparator	264
9.7	Lambda-Ausdruck (λ).....	265
9.7.1	Lambda-Ausdruck - Abstrakt.....	266
9.7.2	Beispiel 1: Lambda (λ).....	267
9.7.3	Beispiel 2: Lambda (λ).....	268
9.8	Funktionale Interfaces.....	269
9.9	Verwendung von Lambda-Ausdrücken.....	270
9.10	Beispiel: Verwendung von Lambdas.....	271
9.11	Zieltyp eines Lambda-Ausdrucks.....	272
9.12	Beispiel: Zieltyp (<i>Target-Typing</i>).....	273
9.13	Scoping und Variablenbindung	274
9.14	Funktionen in Java 8 API - <i>java.util.Function</i>	275
9.15	Beispiel: Vordefinierte Funktionen in Java 8.....	276
9.16	Methodenreferenzen	277
9.16.1	Methodenreferenzen Definition	278
9.16.2	Beispiel: Methodenreferenzen	279
9.17	Default-Methoden.....	281
10	Optional, Streams und Bulk-Operationen.....	283
10.1	Optional - null-Referenz	284
10.2	Optional.....	286
10.3	Zugriff auf Optionals.....	287
10.4	Beispiel Verhinderung NPE.....	288
10.5	Zugriff auf Optionals mit Funktionaler Programmierung / Lambda	289
10.6	Zugriff auf Optionals mit Funktionaler Programmierung / Lambda	290
10.7	Beispiel Verhinderung NPE.....	291
10.8	Streams, Motivation	292
10.9	Streams.....	293
10.10	Stream – Management.....	295
10.11	Streaming API – Lambda.....	296
10.12	Stream API – Erzeugung	297
10.12.1	Beispiel: Streams erzeugen	298
10.13	Zwischenoperationen	299
10.14	Terminaloperationen	300
10.15	Pipelines.....	301
10.16	Parallelisierung.....	302
10.17	Laziness	303
10.18	Laziness – Beispiel (1/2).....	305
10.19	Streams in Aktion: filter	306
10.20	Stream Transformation: map	307
10.21	Umformung Kaskadierung: <i>flatMap</i>	308
10.22	Einzelergebnisse aus Streams: <i>reduce</i>	309
10.23	Beispiele: <i>reduce</i>	310
10.24	Strukturergebnisse aus Streams: <i>collect</i>	311
10.25	Beispiel: <i>collect</i>	312
10.26	Elemente in eine Map packen: <i>collect</i>	313
10.27	Kollisionen behandeln	314
10.28	Gruppierung durchführen: <i>groupingBy</i>	315
10.29	Partitionierung: <i>partitioningBy</i>	316
11	Garbage Collection	317
11.1	Was ist die Garbage Collection (GC)?.....	318

11.2	finalize()-Methode	319
11.3	Standard-Anordnung des Speichers	320
11.4	„Teilweise“ und „Vollständige“ GC	321
11.5	„Teilweise“ und „Vollständige“ GC	322
11.6	Young Generation	323
11.7	GC in Young Generation	324
11.8	GC in Old Generation / Permanent Generation	325
11.9	Unterschiedlichste Parametereinstellungen	326
11.10	Anpassung der initialen und max. Speichergröße	327
11.11	Verfolgen der Garbage Collection Aktivität	328
11.12	„Generational Virtual Machine“	329
11.13	Anpassen der „nursery“-Speichergröße	330
11.14	„Incremental Mode“	331
12	Nützliche Bibliotheken	332
12.1	Allgemeines	333
12.2	Logging	334
12.2.1	Logging Level	335
12.2.2	Logging Beispiel	336
12.2.3	Logging Handler	337
12.2.4	Logging Einstellungen	338
12.3	Apache Commons	339
12.3.1	Beispiel (einfach) – commons-lang	340
12.3.2	Beispiel (komplex) – commons -ang	341
12.3.3	Beispiel – commons-beanutils	342
12.3.4	Packages von Apache Commons	343
12.4	Weitere Apache Projekte	345
12.5	Guava libraries	346
12.5.1	Beispiel: Using and avoiding null	347
12.5.2	Eigene Caches mit Guava	348
12.5.3	Beispiel: eigene Cache Implementierung	349
13	Jigsaw - Modularisierung	350
13.1	Jigsaw – Modularisierung in Java	351
13.2	Jigsaw – woraus es sich zusammensetzt	352
13.3	Module – eine zusätzliche Hierarchiestufe	353
13.4	Was ist JPMS?	354
13.5	Ordner-Struktur eines Moduls	355
13.6	module-info.java – Modul-Name	356
13.7	module-info.java – andere Module lesen	357
13.8	module-info.java – Packages exportieren	358
13.9	Umgang mit Services	359
13.9.1	Umgang mit Services – das Service Interface	360
13.9.2	Umgang mit Services – die Service Implementierung	361
13.9.3	Umgang mit Services – die Service Factory	362
13.9.4	Umgang mit Services – die Service Factory ff.	363
13.10	module-info.java – Pakete öffnen	364
13.11	Modul oder nicht-Modul – Modul-Kategorien	365
13.12	Automatic Module	366