

Java Programmierung Aufbau

Seminarunterlage

Version: 11.10



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	Die „andere Seite“	29
2.9	Eigene generische Typen	31
2.10	Bounds	34
2.11	TreeMap ohne Bounds	35
2.12	Bounds	37
2.13	TreeMap mit Bounds	38
2.14	Generische Methoden	39
2.14.1	generische Methoden Java 4	40
2.14.2	generische Methoden Java 5	42
2.15	Typ-Inferenz	44
2.16	Wildcard Instanziierung	46
2.17	Wildcard Beispiele	48
2.18	Type Erasure	51
2.19	Generische Typen – Grenzen	53
2.20	Zusammenfassung	56
3	Innere Klassen	57
3.1	Was sind Innere Klassen?	58
3.2	„Normale“ Toplevel Klassen	59
3.3	Beispiel: Innere Klassen	60
3.4	Typisierung von Inneren Klassen	61
3.5	Statische innere Klassen	62
3.6	Nicht-statische innere Klassen	64
3.7	Member Klassen	65
3.8	Beispiel: Member Klassen	67
3.9	Lokale Klassen	68
3.10	Beispiel: Lokale Klassen	70
3.11	Anonyme Klassen	71
3.12	Beispiel: Anonyme Klassen	73
3.13	Anwendung: Implementation Hiding	74
3.14	Anwendung: Adapterklassen	75
3.15	Zusammenfassung	76
4	Multithreading	77
4.1	Grundlagen: Concurrency& Threads	78
4.2	Multithreading	79
4.3	Thread Basics	80

4.4	Thread Status.....	82
4.5	Thread-safe	83
4.6	Deadlock	84
4.7	Immutable Objects	85
4.8	synchronized	86
4.9	Atomic Klassen.....	87
4.10	Guarded Blöcke	89
4.11	Lock Objects.....	90
4.12	Wir kommt man zu Multithreading.....	91
4.13	Package java.util.concurrent	92
4.14	java.util.concurrent: Executor und Co	93
4.15	Executor und Co	94
4.16	ExecutorService	95
4.17	Datenstrukturen: Queues und Co	96
4.18	Queues und Co	97
4.19	BlockingQueue Implementierungen.....	98
4.20	ArrayBlockingQueue	99
4.21	PriorityBlockingQueue.....	100
4.22	Datenstrukturen: Concurrent Collections	101
4.23	Concurrent Collections	102
4.24	ConcurrentHashMap	103
4.25	Synchronizers.....	104
4.26	CountDownLatch.....	105
4.27	CyclicBarrier	106
4.28	Semaphore.....	107
4.29	Phaser	108
4.30	Callable und Future	109
4.31	Callable & Future mit ScheduledExecutorService	111
4.32	Swing - Event Dispatcher Thread	112
4.33	GUI Hilfsklasse: SwingWorker<T,V>	113
4.34	SwingWorker Beispiel	114
4.35	Fork/Join.....	117
4.36	Fork/Join – wichtige Klassen und Interfaces.....	119
4.37	Fork/Join – Threads und Tasks.....	121
4.38	Fork/Join Beispiel.....	122
4.39	Work stealing.....	125
5	Reflection & Annotations	126
5.1	Reflection API.....	127
5.2	Klassen der Reflection API	128
5.3	Class-Objekt.....	129
5.4	Namen und Modifizierer einer Klasse ermitteln	130
5.5	Superklasse und Interfaces ermitteln.....	131
5.6	Klassen- bzw. Interfaceattribute ermitteln.....	132
5.7	Konstruktoren ermitteln	134
5.8	Methoden samt Signatur ermitteln	135
5.9	Objekte manipulieren	136
5.10	Objekte über den Default-Konstruktor erzeugen	137
5.11	Objekterzeugung: parametrisierter Konstruktor	138
5.12	Attributwerte ermitteln& setzen	140
5.13	Methoden aufrufen	142
5.14	Was sind Annotations?	143
5.15	Anwendungsgebiete.....	145
	Anwendungsgebiete: EJB & Co	145
5.16	Annotation Beispiel: @Deprecated	147
5.17	Verwendung von Annotations	149
5.18	Vordefinierte Annotations.....	151
5.19	Standard Annotations.....	152
5.20	Definition eigener Annotations-Typen.....	154
5.21	Annotations mit Eigenschaften.....	155

5.22	Meta-Annotation Target	160
5.23	Meta-Annotation Retention	163
5.24	Meta-Annotation: Documented	165
5.25	Meta-Annotation Inherited.....	168
5.26	Zugriff auf Annotations über Reflection API.....	170
5.27	AnnotatedElement.....	171
5.28	Beispiel: Zugriff über Reflection	172
6	Typesafe Enums	174
6.1	Typesafe Enums	175
6.2	Aufzählungen bisher	176
6.3	Probleme mit int-Enumerations.....	177
6.4	Idiom für typsichere Aufzählungen.....	179
6.5	Typesafe Enums	182
6.6	Deklaration von enum-Klassen	184
6.7	Eigenschaften von enum-Klassen.....	186
6.8	enum Beispiel.....	189
6.9	Methoden in enum-Klassen	193
6.9.1	switch-Statement.....	195
6.9.2	abstrakte Methoden.....	197
6.10	Zwei neue Klassen für enum.....	198
6.10.1	EnumSet.....	199
6.10.2	EnumMap	203
7	Java Persistence API (JPA).....	206
7.1	Datenbanken	207
7.2	Database Connectivity	208
7.3	JDBC	209
7.4	JDBC DB-Verbindung aufbauen	210
7.5	SQL-Anweisungen - Statement.....	211
7.6	Auslesen einer Ergebnismenge - ResultSet	213
7.7	JDBC DB-Verbindung aufbauen	215
7.8	Vorbereitete Anweisungen - PreparedStatement	216
7.9	JDBC Transaktionen	217
7.10	JDBC Metadaten.....	219
7.11	Zusammenfassung JDBC	221
7.12	Was ist JPA?	222
7.13	Persistieren mit der JPA.....	223
7.14	Vor-/Nachteile.....	232
8	JAXB – XML Verarbeitung in Java.....	233
8.1	Allgemeines	234
8.2	Primäre Ziele	235
8.3	Klassengenerierung	236
8.4	Un-/Marshalling bei Data Binding.....	237
8.5	Zyklus (Round Trip).....	239
8.6	Unmarshal: XML → Java	240
8.7	Marshal: Java → XML	241
8.8	Annotationen	242
8.9	Annotationen in JAXB	243
8.10	@XmlElement und @XmlAttribute.....	244
Exkurs JavaBean (aus Wikipedia):.....		244
8.11	@XmlElement und @XmlAttribute, Beispiel	246
8.12	@XmlAccessorType.....	247
8.13	@XmlAccessType.PUBLIC_MEMBER.....	249
8.14	@XmlAccessType.PROPERTY	250
8.15	@XmlAccessType.FIELD.....	251
8.16	@XmlAccessType.NONE	252
8.17	@XmlTransient	253
8.18	Zusammenpiel Java-Anwendung und XML Data Binding	254

8.19	Generierung und Nutzung.....	255
8.20	Tools.....	256
9	Lambda-Ausdrücke.....	260
9.1	Funktionale Programmierung.....	261
9.2	OOP vs. Funktionale Programmierung.....	262
9.3	Neues Sprachkonstrukt.....	263
9.4	Lambda-Ausdruck (λ).....	264
9.4.1	Beispiel 1: Lambda (λ).....	265
9.4.2	Beispiel 2: Lambda (λ).....	266
9.5	Funktionale Interfaces.....	267
9.6	Verwendung von Lambda-Ausdrücken.....	268
9.7	Beispiel: Verwendung von Lambdas.....	269
9.8	Zieltyp eines Lambda-Ausdrucks.....	270
9.9	Beispiel: Zieltyp (<i>Target-Typing</i>).....	271
9.10	Scoping und Variablenbindung.....	272
9.11	Funktionen in Java 8 API - <i>java.util.Function</i>	273
9.12	Beispiel: Vordefinierte Funktionen in Java 8.....	274
9.13	Methodenreferenzen.....	275
9.13.1	Methodenreferenzen Definition.....	276
9.13.2	Beispiel: Methodenreferenzen.....	277
9.14	Default-Methoden.....	279
10	Streams und Bulk-Operationen.....	280
10.1	Streams, Motivation.....	281
10.2	Streams.....	282
10.3	Stream – Management.....	284
10.4	Streaming API – Lambda.....	285
10.5	Stream API – Erzeugung.....	286
10.5.1	Beispiel: Streams erzeugen.....	287
10.6	Zwischenoperationen.....	288
10.7	Terminaloperationen.....	289
10.8	Pipelines.....	290
10.9	Parallelisierung.....	291
10.10	Laziness.....	292
10.11	Streams in Aktion: filter.....	293
10.12	Stream Transformation: map.....	294
10.13	Umformung Kaskadierung: <i>flatMap</i>	295
10.14	Einzelergebnisse aus Streams: <i>reduce</i>	296
10.15	Beispiele: <i>reduce</i>	297
10.16	Strukturergebnisse aus Streams: <i>collect</i>	298
10.17	Beispiel: <i>collect</i>	299
10.18	Elemente in eine Map packen: <i>collect</i>	300
10.19	Kollisionen behandeln.....	301
10.20	Gruppierung durchführen: <i>groupingBy</i>	302
10.21	Partitionierung: <i>partitioningBy</i>	303
11	Garbage Collection.....	304
11.1	Was ist die Garbage Collection (GC)?.....	305
11.2	<i>finalize()</i> -Methode.....	306
11.3	Standard-Anordnung des Speichers.....	307
11.4	„Teilweise“ und „Vollständige“ GC.....	308
11.5	„Teilweise“ und „Vollständige“ GC.....	309
11.6	Young Generation.....	310
11.7	GC in Young Generation.....	311
11.8	GC in Old Generation / Permanent Generation.....	312
11.9	Unterschiedlichste Parametereinstellungen.....	313
11.10	Anpassung der initialen und max. Speichergröße.....	314
11.11	Verfolgen der Garbage Collection Aktivität.....	315
11.12	„Generational Virtual Machine“.....	316

11.13	Anpassen der „nursery“-Speichergröße.....	317
11.14	„Incremental Mode“	318
12	Nützliche Bibliotheken	319
12.1	Allgemeines	320
12.2	Logging.....	321
12.2.1	Logging Level	322
12.2.2	Logging Beispiel	323
12.2.3	Logging Handler	324
12.2.4	Logging Einstellungen	325
12.3	Apache Commons.....	326
12.3.1	Beispiel (einfach) – commons-lang	327
12.3.2	Beispiel (komplex) – commons -ang	328
12.3.3	Beispiel – commons-beanutils	329
12.3.4	Packages von Apache Commons.....	330
12.4	Weitere Apache Projekte	332
12.5	Guava libraries	333
12.5.1	Beispiel: Using and avoiding null	334
12.5.2	Eigene Caches mit Guava.....	335
12.5.3	Beispiel: eigene Cache Implementierung.....	336