

Java Programmierung Aufbau

Seminarunterlage

Version: 11.12



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

ORDIX AG
Südwestpark 67/2
D-90449 Nürnberg
Tel.: (+49) 0 52 51 / 10 63 - 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	aGenerics	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	Ober- und Unterklassen in generischen Typen	25
2.6	weak und strong typing	27
2.7	Beispiel Collection Framework	28
2.8	Eigene generische Typen	29
2.9	Bounds	32
2.10	TreeMap ohne Bounds	33
2.11	Bounds	35
2.12	TreeMap mit Bounds	36
2.13	Generische Methoden	37
2.13.1	generische Methoden Java 4	38
2.13.2	generische Methoden Java 5	40
2.14	Typ-Inferenz	42
2.15	Wildcard Instanziierung	44
2.16	Wildcard Beispiele	46
2.17	Type Erasure	49
2.18	Generische Typen – Grenzen	51
2.19	Zusammenfassung	54
3	Innere Klassen	55
3.1	Was sind Innere Klassen?	56
3.2	Beispiel: Innere Klassen	57
3.3	Typisierung von Inneren Klassen	58
3.4	Statische innere Klassen	59
3.5	Nicht-statische innere Klassen	61
3.6	Member Klassen	62
3.7	Beispiel: Member Klassen	64
3.8	Lokale Klassen	65
3.9	Beispiel: Lokale Klassen	67
3.10	Anonyme Klassen	68
3.11	Beispiel: Anonyme Klassen	70
3.12	Anwendung: Implementation Hiding	71
3.13	Anwendung: Adapterklassen	72
3.14	Zusammenfassung	73
4	Multithreading	74
4.1	Grundlagen: Concurrency& Threads	75
4.2	Multithreading	76
4.3	Thread Basics	77
4.4	Thread Status	79
4.5	Thread-safe	80

4.6	Deadlock	81
4.7	Immutable Objects	82
4.8	synchronized	83
4.9	Atomic Klassen.....	84
4.10	Guarded Blöcke	86
4.11	Lock Objects.....	87
4.12	Wir kommt man zu Multithreading.....	88
4.13	Package java.util.concurrent	89
4.14	java.util.concurrent: Executor und Co	90
4.15	Executor und Co	91
4.16	ExecutorService	92
4.17	Datenstrukturen: Queues und Co	93
4.18	Queues und Co	94
4.19	BlockingQueue Implementierungen.....	95
4.20	ArrayBlockingQueue	96
4.21	PriorityBlockingQueue.....	97
4.22	Datenstrukturen: Concurrent Collections	98
4.23	Concurrent Collections	99
4.24	ConcurrentHashMap	100
4.25	Synchronizers.....	101
4.26	CountDownLatch.....	102
4.27	CyclicBarrier	103
4.28	Semaphore.....	104
4.29	Phaser	105
4.30	Callable und Future	106
4.31	Callable & Future mit ScheduledExecutorService	108
4.32	Swing - Event Dispatcher Thread	109
4.33	GUI Hilfsklasse: SwingWorker<T,V>	110
4.34	SwingWorker Beispiel	111
4.35	Fork/Join.....	114
4.36	Fork/Join – wichtige Klassen und Interfaces.....	116
4.37	Fork/Join – Threads und Tasks.....	118
4.38	Fork/Join Beispiel.....	119
4.39	Work stealing.....	122
5	Reflection & Annotations	123
5.1	Reflection API.....	124
5.2	Klassen der Reflection API	125
5.3	Class-Objekt.....	126
5.4	Namen und Modifizierer einer Klasse ermitteln	127
5.5	Superklasse und Interfaces ermitteln.....	128
5.6	Klassen- bzw. Interfaceattribute ermitteln	129
5.7	Konstruktoren ermitteln	131
5.8	Methoden samt Signatur ermitteln	132
5.9	Objekte manipulieren	133
5.10	Objekte über den Default-Konstruktor erzeugen	134
5.11	Objekterzeugung: parametrisierter Konstruktor	135
5.12	Attributwerte ermitteln& setzen	137
5.13	Methoden aufrufen	139
5.14	Was sind Annotations?	140
5.15	Anwendungsgebiete.....	142
	Anwendungsgebiete: EJB & Co	142
5.16	Annotation Beispiel: @Deprecated	144
5.17	Verwendung von Annotations	146
5.18	Vordefinierte Annotations.....	148
5.19	Standard Annotations.....	149
5.20	Definition eigener Annotations-Typen.....	151
5.21	Annotations mit Eigenschaften.....	152
5.22	Meta-Annotation Target	156
5.23	Meta-Annotation Retention	159

5.24	Meta-Annotation: Documented	161
5.25	Meta-Annotation Inherited.....	164
5.26	Zugriff auf Annotations über Reflection API.....	166
5.27	AnnotatedElement.....	167
5.28	Beispiel: Zugriff über Reflection	168
6	Typesafe Enums	170
6.1	Typesafe Enums	171
6.2	Aufzählungen bisher	172
6.3	Probleme mit int-Enumerations.....	173
6.4	Idiom für typsichere Aufzählungen.....	175
6.5	Typesafe Enums	178
6.6	Deklaration von enum-Klassen	180
6.7	Eigenschaften von enum-Klassen.....	182
6.8	enum Beispiel.....	185
6.9	Methoden in enum-Klassen	189
6.9.1	switch-Statement.....	191
6.9.2	abstrakte Methoden.....	193
6.10	Zwei neue Klassen für enum.....	194
6.10.1	EnumSet.....	195
6.10.2	EnumMap	199
7	Java Persistence API (JPA).....	202
7.1	Datenbanken	203
7.2	Database Connectivity	204
7.3	JDBC.....	205
7.4	JDBC DB-Verbindung aufbauen	206
7.5	SQL-Anweisungen - Statement.....	207
7.6	Auslesen einer Ergebnismenge - ResultSet	209
7.7	JDBC DB-Verbindung aufbauen	211
7.8	Vorbereitete Anweisungen - PreparedStatement	212
7.9	JDBC Transaktionen	213
7.10	JDBC Metadaten	215
7.11	Zusammenfassung JDBC	217
7.12	Was ist JPA?	218
7.13	Persistieren mit der JPA.....	219
7.14	Vor-/Nachteile.....	228
8	JAXB – XML Verarbeitung in Java.....	229
8.1	Allgemeines.....	230
8.2	Primäre Ziele	231
8.3	Klassengenerierung	232
8.4	Un-/Marshalling bei Data Binding.....	233
8.5	Zyklus (Round Trip).....	235
8.6	Unmarshal: XML → Java	236
8.7	Marshal: Java → XML	237
8.8	Annotationen	238
8.9	Annotationen in JAXB	239
8.10	@XmlElement und @XmlAttribute.....	240
Exkurs JavaBean (aus Wikipedia):.....		240
8.11	@XmlElement und @XmlAttribute, Beispiel	242
8.12	@XmlAccessorType.....	243
8.13	@XmlAccessorType.PUBLIC_MEMBER	245
8.14	@XmlAccessorType.PROPERTY	246
8.15	@XmlAccessorType.FIELD.....	247
8.16	@XmlAccessorType.NONE	248
8.17	@XmlTransient	249
8.18	Zusammenpiel Java-Anwendung und XML Data Binding	250
8.19	Generierung und Nutzung.....	251
8.20	Tools.....	252

9	Deklarative Programmierung / Lambda	256
9.1	Programmierparadigma	257
9.2	Deklarative Programmierung	258
9.3	Funktionale Programmierung	259
9.4	OOP vs. Funktionale Programmierung	260
9.5	Neues Sprachkonstrukt	261
9.6	Beispiel Java Comparator	262
9.7	Lambda-Ausdruck (λ)	263
9.7.1	Lambda-Ausdruck - Abstrakt	264
9.7.2	Beispiel 1: Lambda (λ)	265
9.7.3	Beispiel 2: Lambda (λ)	266
9.8	Funktionale Interfaces	267
9.9	Verwendung von Lambda-Ausdrücken	268
9.10	Beispiel: Verwendung von Lambdas	269
9.11	Zieltyp eines Lambda-Ausdrucks	270
9.12	Beispiel: Zieltyp (<i>Target-Typing</i>)	271
9.13	Scoping und Variablenbindung	272
9.14	Funktionen in Java 8 API - <i>java.util.Function</i>	273
9.15	Beispiel: Vordefinierte Funktionen in Java 8	274
9.16	Methodenreferenzen	275
9.16.1	Methodenreferenzen Definition	276
9.16.2	Beispiel: Methodenreferenzen	277
9.17	Default-Methoden	279
10	Optional, Streams und Bulk-Operationen	281
10.1	Optional - null-Referenz	282
10.2	Optional	284
10.3	Zugriff auf Optionals	285
10.4	Beispiel Verhinderung NPE	286
10.5	Bedingter Zugriff auf Optional-Daten	287
10.6	Zugriff auf Optionals mit Funktionaler Programmierung / Lambda	288
10.7	Beispiel Verhinderung NPE	289
10.8	Streams	290
10.9	Pipelines	291
10.10	Beispiele	292
10.11	Beispiele (ff.)	294
10.12	Streaming API – Lambda	295
10.13	Stream API – Erzeugung	296
10.14	Beispiel: Streams erzeugen	298
10.15	Zwischenoperationen	300
10.16	Stream Transformation: map	301
10.17	Terminaloperationen	302
10.18	Strukturergebnisse aus Streams: <i>collect</i>	303
10.19	Beispiele Collectors	304
10.20	Elemente in eine Map packen: <i>collect</i>	305
10.20.1	Kollisionen behandeln	306
10.21	Gruppierung durchführen: <i>groupingBy</i>	307
10.22	Partitionierung durchführen: <i>partitioningBy</i>	308
10.23	Collect <i>Basics</i> – ohne den Einsatz von <i>Collectors</i>	309
10.24	Einzelergebnisse aus Streams: <i>reduce</i>	310
10.25	Beispiele <i>reduce</i>	311
10.26	Umformung Kaskadierung: <i>flatMap</i>	312
10.27	Beispiel: <i>flatMap</i>	313
10.28	Parallelisierung	314
10.29	Laziness	315
10.30	Laziness – Beispiel	316
11	Garbage Collection	318
11.1	Was ist die Garbage Collection (GC)?	319

11.2	finalize()-Methode	320
11.3	Standard-Anordnung des Speichers	321
11.4	„Teilweise“ und „Vollständige“ GC	322
11.5	„Teilweise“ und „Vollständige“ GC	323
11.6	Young Generation	324
11.7	GC in Young Generation	325
11.8	GC in Old Generation / Permanent Generation	326
11.9	Unterschiedlichste Parametereinstellungen	327
11.10	Anpassung der initialen und max. Speichergröße	328
11.11	Verfolgen der Garbage Collection Aktivität	329
11.12	„Generational Virtual Machine“	330
11.13	Anpassen der „nursery“-Speichergröße	331
11.14	„Incremental Mode“	332
12	Nützliche Bibliotheken	333
12.1	Allgemeines	334
12.2	Logging	335
12.2.1	Logging Level	336
12.2.2	Logging Beispiel	337
12.2.3	Logging Handler	338
12.2.4	Logging Einstellungen	339
12.3	Apache Commons	340
12.3.1	Beispiel (einfach) – commons-lang	341
12.3.2	Beispiel (komplex) – commons -ang	342
12.3.3	Beispiel – commons-beanutils	343
12.3.4	Packages von Apache Commons	344
12.4	Weitere Apache Projekte	346
12.5	Guava libraries	347
12.5.1	Beispiel: Using and avoiding null	348
12.5.2	Eigene Caches mit Guava	349
12.5.3	Beispiel: eigene Cache Implementierung	350
13	Jigsaw - Modularisierung	351
13.1	Jigsaw – Modularisierung in Java	352
13.2	Jigsaw – woraus es sich zusammensetzt	353
13.3	Module – eine zusätzliche Hierarchiestufe	354
13.4	Was ist JPMS?	355
13.5	Ordner-Struktur eines Moduls	356
13.6	module-info.java – Modul-Name	357
13.7	module-info.java – andere Module lesen	358
13.8	module-info.java – Packages exportieren	359
13.9	Umgang mit Services	360
13.9.1	Umgang mit Services – das Service Interface	361
13.9.2	Umgang mit Services – die Service Implementierung	362
13.9.3	Umgang mit Services – die Service Factory	363
13.9.4	Umgang mit Services – die Service Factory ff.	364
13.10	module-info.java – Pakete öffnen	365
13.11	Modul oder nicht-Modul – Modul-Kategorien	366
13.12	Automatic Module	367