

Inhaltsverzeichnis

1	Einleitung	1
I	Sprach- und API-Erweiterungen	3
2	Syntaxerweiterungen	5
2.1	Anonyme innere Klassen und der Diamond Operator	5
2.2	Nutzung von »effectively final«-Variablen im ARM	6
2.3	Neuerung bei der @SafeVarargs-Annotation	8
2.4	Erweiterung der @Deprecated-Annotation	11
2.5	Private Methoden in Interfaces	12
2.6	Verbotener Bezeichner '_'	14
3	Neues und Änderungen im JDK	15
3.1	Neue APIs	15
3.1.1	Das neue Process-API	15
3.1.2	Collection-Factory-Methoden	21
3.1.3	Reactive Streams und die Klasse Flow	27
3.1.4	Taskbar-Support	37
3.1.5	Verarbeitung von Stackframes und die Klasse StackWalker	40
3.1.6	HTTP/2-Support	42
3.2	Erweiterte APIs	47
3.2.1	Erweiterungen in der Klasse InputStream	48
3.2.2	Erweiterungen rund um die Klasse Optional	50
3.2.3	Erweiterungen im Stream-API	55
3.2.4	Erweiterungen in der Klasse LocalDate	59
3.2.5	Support von UTF-8 in ResourceBundle	60
3.2.6	Erweiterungen in der Klasse Arrays	65
3.2.7	Erweiterungen in der Klasse Objects	67
3.2.8	Erweiterungen in der Klasse CompletableFuture	67
3.2.9	Erweiterungen in Class<T>	71
3.2.10	Die Klasse MethodHandle	73
3.2.11	Die Klasse VarHandle	76

II Modularisierung 119

II Modularisierung 119

6	Modularisierung mit Project Jigsaw	121
6.1	Grundlagen	122
6.1.1	Bisherige Varianten der Modularisierung	123
6.1.2	Warum wir Modularisierung brauchen	125
6.2	Modularisierung im Überblick	126
6.2.1	Grundlagen zu Project Jigsaw	126
6.2.2	Einführendes Beispiel	134
6.2.3	Komplexeres Beispiel	137
6.2.4	Packaging	146
6.2.5	Linking	147
6.2.6	Abhängigkeiten und Modulgraphen	151
6.2.7	Module des JDKs einbinden	153
6.2.8	Arten von Modulen	160
6.3	Sichtbarkeiten und Zugriffsschutz	161
6.3.1	Sichtbarkeiten	161
6.3.2	Zugriffsschutz an Beispielen	163
6.3.3	Transitive Abhängigkeiten (Implied Readability)	168
6.4	Zusammenfassung	173

7	Weiterführende Themen zur Modularisierung	175
7.1	Modularisierung und Services	176
7.1.1	Begrifflichkeiten: API, SPI und Service Provider	176
7.1.2	Service-Ansatz in Java seit JDK 6	177
7.1.3	Services im Bereich der Modularisierung	180
7.1.4	Definition eines Service Interface	181
7.1.5	Realisierung eines Service Provider	182
7.1.6	Realisierung eines Service Consumer	184
7.1.7	Kontrolle der Abhängigkeiten	186
7.1.8	Fazit	187
7.2	Modularisierung und Reflection	188
7.2.1	Verarbeitung von Modulen mit Reflection	188
7.2.2	Tool zur Ermittlung von Modulen zu Klassen	190
7.2.3	Konvertierungstool für <code>import</code> zu <code>requires</code>	192
7.2.4	Besonderheiten bei Reflection	195
7.3	Kompatibilität und Migration	201
7.3.1	Kompatibilitätsmodus	201
7.3.2	Migrationsszenarien	204
7.3.3	Fallstrick bei der Bottom-up-Migration	208
7.3.4	Beispiel: Migration mit Automatic Modules	209
7.3.5	Beispiel: Automatic und Unnamed Module	211
7.3.6	Abwandlung mit zwei Automatic Modules	214
7.3.7	Mögliche Schwierigkeiten bei Migrationen	216
7.3.8	Fazit	216
8	Übungen zur Modularisierung	217
III	Verschiedenes	227
9	Build-Tools und IDEs	229
9.1	Nicht modularisierte Applikationen	229
9.1.1	Gradle	231
9.1.2	Maven	233
9.1.3	Eclipse	235
9.1.4	IntelliJ IDEA	235
9.1.5	NetBeans	235
9.2	Nicht modularisierte Applikationen mit HTTP/2-API	236
9.2.1	Gradle	236
9.2.2	Maven	238
9.2.3	Eclipse	240
9.2.4	IntelliJ IDEA	241
9.2.5	NetBeans	243

9.3	Modularisierte Applikationen	244
9.3.1	Gradle	245
9.3.2	Maven	250
9.3.3	Eclipse	255
9.3.4	IntelliJ IDEA	257
9.3.5	NetBeans	261
9.4	Besonderheiten beim Unit-Testen	264
9.4.1	Gradle	265
9.4.2	Maven	266
9.4.3	Eclipse	267
9.4.4	IntelliJ IDEA	268
9.4.5	NetBeans	268
9.4.6	Kommandozeile	268
9.5	Kompatibilitätsmodus	275
9.5.1	Gradle	276
9.5.2	Maven	277
9.5.3	Eclipse	278
9.5.4	IntelliJ IDEA	278
9.5.5	NetBeans	278
9.5.6	Kommandozeile	278
9.6	Fazit	279
10	Zusammenfassung	281

IV Anhang **285**

A	Schnelleinstieg in Java 8	287
A.1	Einstieg in Lambdas	287
A.1.1	Lambdas am Beispiel	287
A.1.2	Functional Interfaces und SAM-Typen	288
A.1.3	Type Inference und Kurzformen der Syntax	291
A.1.4	Methodenreferenzen	292
A.2	Streams im Überblick	293
A.2.1	Streams erzeugen – Create Operations	294
A.2.2	Intermediate und Terminal Operations im Überblick	296
A.2.3	Zustandslose Intermediate Operations	298
A.2.4	Zustandsbehaftete Intermediate Operations	301
A.2.5	Terminal Operations	303
A.3	Neuerungen in der Datumsverarbeitung	307
A.3.1	Neue Aufzählungen, Klassen und Interfaces	308
A.3.2	Die Klasse <code>Instant</code>	310
A.3.3	Die Klasse <code>Duration</code>	311

A.3.4	Die Klassen <code>LocalDate</code> , <code>LocalTime</code> und <code>LocalDateTime</code>	312
A.3.5	Die Klasse <code>Period</code>	313
A.3.6	Datumsarithmetik mit <code>TemporalAdjusters</code>	315
A.4	Diverse Erweiterungen	317
A.4.1	Erweiterungen im Interface <code>Comparator<T></code>	317
A.4.2	Die Klasse <code>Optional<T></code>	319
A.4.3	Die Klasse <code>CompletableFuture</code>	325
B	Einführung Gradle	329
B.1	Projektstruktur für Maven und Gradle	329
B.2	Builds mit Gradle	331
C	Einführung Maven	341
C.1	Maven im Überblick	341
C.2	Maven am Beispiel	344
	Literaturverzeichnis	347
	Index	349