

Inhaltsverzeichnis

| | | |
|----------|---|-----------|
| 1 | Einleitung | 1 |
| I | Neuerungen in Java 9 bis 11 | 5 |
| 2 | Syntaxerweiterungen in JDK 9 bis 11 | 7 |
| 2.1 | Anonyme innere Klassen und der Diamond Operator | 7 |
| 2.2 | Erweiterung der <code>@Deprecated</code> -Annotation | 8 |
| 2.3 | Private Methoden in Interfaces | 9 |
| 2.4 | Verbotener Bezeichner <code>'_'</code> | 10 |
| 2.5 | Syntaxerweiterung <code>var</code> (JDK 10 und 11) | 11 |
| 3 | Neues und Änderungen in JDK 9 | 17 |
| 3.1 | Neue und erweiterte APIs | 17 |
| 3.1.1 | Das neue Process-API | 17 |
| 3.1.2 | Collection-Factory-Methoden | 21 |
| 3.1.3 | Reactive Streams und die Klasse <code>Flow</code> | 24 |
| 3.1.4 | Erweiterungen in der Klasse <code>InputStream</code> | 34 |
| 3.1.5 | Erweiterungen rund um die Klasse <code>Optional<T></code> | 35 |
| 3.1.6 | Erweiterungen im Stream-API | 40 |
| 3.1.7 | Neue Kollektoren im Stream-API | 44 |
| 3.1.8 | Erweiterungen in der Datumsverarbeitung | 48 |
| 3.1.9 | Erweiterungen in der Klasse <code>Arrays</code> | 51 |
| 3.1.10 | Erweiterungen in der Klasse <code>Objects</code> | 55 |
| 3.1.11 | Erweiterungen in der Klasse <code>CompletableFuture<T></code> | 56 |
| 3.2 | Sonstige Änderungen | 60 |
| 3.2.1 | Optimierung bei Strings | 60 |
| 3.2.2 | Deprecation diverser Typen und Methoden im JDK | 61 |
| 4 | Neues und Änderungen in Java 10 | 63 |
| 4.1 | Neue und erweiterte APIs | 63 |
| 4.1.1 | Unveränderliche Kopien von Collections | 63 |
| 4.1.2 | Immutable Collections aus Streams erzeugen | 65 |

| | | |
|--|--|------------|
| 4.1.3 | Erweiterung in der Klasse <code>Optional<T></code> | 67 |
| 4.1.4 | Modifikationen in der Versionierung | 68 |
| 4.1.5 | Erweiterung in <code>Reader</code> | 68 |
| 5 | Neues und Änderungen in Java 11 | 69 |
| 5.1 | Neue und erweiterte APIs | 70 |
| 5.1.1 | Hilfsmethoden in der Klasse <code>String</code> | 70 |
| 5.1.2 | Hilfsmethoden in der Utility-Klasse <code>Files</code> | 73 |
| 5.1.3 | Erweiterung in der Klasse <code>Optional<T></code> | 74 |
| 5.1.4 | Erweiterung im Interface <code>Predicate<T></code> | 75 |
| 5.1.5 | HTTP/2-API | 76 |
| 5.2 | Deprecations und Entfernungen im JDK | 80 |
| 5.2.1 | Aufräumarbeiten in der Klasse <code>Thread</code> | 80 |
| 5.2.2 | Deprecation der JavaScript-Unterstützung | 80 |
| 5.2.3 | Ausgliederung von JavaFX | 80 |
| 5.2.4 | Ausgliederung von Java EE und CORBA | 81 |
| 6 | JVM-Änderungen in JDK 9 bis 11 | 83 |
| 6.1 | Änderung des Versionsschemas | 83 |
| 6.2 | Java + REPL => <code>jshell</code> | 85 |
| 6.3 | HTML5 Javadoc | 89 |
| 6.4 | Epsilon Garbage Collector (JDK 11) | 90 |
| 6.5 | Launch Single-File Source-Code Programs (JDK 11) | 90 |
| 7 | Übungen zu den Neuerungen in JDK 9 bis 11 | 91 |
| II Neuerungen in Java 12 bis 14 | | 105 |
| 8 | Neues und Änderungen in Java 12 | 107 |
| 8.1 | Microbenchmark Suite | 107 |
| 8.1.1 | Eigene Microbenchmarks und Varianten davon | 108 |
| 8.1.2 | Microbenchmarks mit JMH | 110 |
| 8.1.3 | Fazit zu JMH | 115 |
| 8.2 | API-Neuerungen | 116 |
| 8.2.1 | Neue Methoden in der Klasse <code>String</code> | 116 |
| 8.2.2 | Neue Utility-Klasse <code>CompactNumberFormat</code> | 118 |
| 8.2.3 | Neue Hilfsmethode in der Utility-Klasse <code>Files</code> | 120 |
| 8.2.4 | Der <code>teeing()</code> -Kollektor | 122 |

| | | |
|-----------|---|------------|
| 9 | Neues und Änderungen in Java 13 und 14 | 125 |
| 9.1 | Switch Expressions | 126 |
| 9.1.1 | Einführendes Beispiel | 126 |
| 9.1.2 | Weitere Gründe für die Neuerung | 128 |
| 9.1.3 | <code>yield</code> mit Rückgabewert | 130 |
| 9.2 | Verbesserung bei <code>NullPointerException</code> s | 132 |
| 9.3 | Preview-Features | 135 |
| 9.3.1 | Text Blocks | 135 |
| 9.3.2 | Records | 140 |
| 9.3.3 | Pattern Matching bei <code>instanceof</code> | 146 |
| 9.4 | Java 14 – notwendige Anpassungen für Build-Tools und IDEs | 147 |
| 9.4.1 | Java 14 mit Gradle | 148 |
| 9.4.2 | Java 14 mit Maven | 149 |
| 9.4.3 | Java 14 mit Eclipse | 150 |
| 9.4.4 | Java 14 mit IntelliJ | 151 |
| 9.4.5 | Java 14 mit JShell oder der Kommandozeile | 152 |
| 9.5 | Fazit | 152 |
| 10 | Übungen zu den Neuerungen in JDK 12 bis 14 | 153 |

| | | |
|------------|------------------------|------------|
| III | Modularisierung | 163 |
|------------|------------------------|------------|

| | | |
|-----------|---|------------|
| 11 | Modularisierung mit Project Jigsaw | 165 |
| 11.1 | Grundlagen | 166 |
| 11.1.1 | Bisherige Varianten der Modularisierung | 167 |
| 11.1.2 | Warum Modularisierung wünschenswert ist | 169 |
| 11.2 | Modularisierung im Überblick | 170 |
| 11.2.1 | Grundlagen zu Project Jigsaw | 170 |
| 11.2.2 | Einführendes Beispiel mit zwei Modulen | 178 |
| 11.2.3 | Packaging | 187 |
| 11.2.4 | Linking | 189 |
| 11.2.5 | Abhängigkeiten und Modulgraphen | 193 |
| 11.2.6 | Module des JDKs einbinden | 195 |
| 11.2.7 | Arten von Modulen | 200 |
| 11.3 | Sichtbarkeiten und Zugriffsschutz | 202 |
| 11.3.1 | Sichtbarkeiten | 202 |
| 11.3.2 | Zugriffsschutz an Beispielen | 204 |
| 11.3.3 | Transitive Abhängigkeiten (Implied Readability) | 209 |
| 11.4 | Zusammenfassung | 214 |

| | | |
|-----------|---|------------|
| 12 | Weiterführende Themen zur Modularisierung | 215 |
| 12.1 | Empfehlenswertes Verzeichnislayout für Module | 216 |
| 12.2 | Modularisierung und Services | 218 |
| 12.2.1 | Begrifflichkeiten: API, SPI und Service Provider | 218 |
| 12.2.2 | Service-Ansatz in Java seit JDK 6 | 219 |
| 12.2.3 | Services im Bereich der Modularisierung | 222 |
| 12.2.4 | Definition eines Service Interface | 223 |
| 12.2.5 | Realisierung eines Service Provider | 225 |
| 12.2.6 | Realisierung eines Service Consumer | 227 |
| 12.2.7 | Kontrolle der Abhängigkeiten | 229 |
| 12.2.8 | Fazit | 230 |
| 12.3 | Modularisierung und Reflection | 231 |
| 12.3.1 | Verarbeitung von Modulen mit Reflection | 231 |
| 12.3.2 | Tool zur Ermittlung von Modulen zu Klassen | 233 |
| 12.3.3 | Besonderheiten bei Reflection | 235 |
| 12.4 | Kompatibilität und Migration | 241 |
| 12.4.1 | Kompatibilitätsmodus | 241 |
| 12.4.2 | Migrationsszenarien | 244 |
| 12.4.3 | Fallstrick bei der Bottom-up-Migration | 248 |
| 12.4.4 | Beispiel: Migration mit Automatic Modules | 249 |
| 12.4.5 | Beispiel: Automatic und Unnamed Module | 251 |
| 12.4.6 | Beispiel: Abwandlung mit zwei Automatic Modules | 254 |
| 12.4.7 | Fazit | 256 |
| 12.5 | Build-Management für modularisierte Applikationen | 257 |
| 12.5.1 | Gradle | 258 |
| 12.5.2 | Maven | 262 |
| 12.5.3 | Eclipse | 267 |
| 12.5.4 | IntelliJ IDEA | 269 |
| 12.5.5 | Fazit | 271 |
| 13 | Übungen zur Modularisierung | 273 |
| IV | Schlussgedanken | 283 |
| 14 | Zusammenfassung | 285 |

| | | |
|----------|--|------------|
| V | Anhang | 289 |
| A | Schnelleinstieg in Java 8 | 291 |
| A.1 | Einstieg in Lambdas | 291 |
| A.1.1 | Lambdas am Beispiel | 291 |
| A.1.2 | Functional Interfaces und SAM-Typen | 292 |
| A.1.3 | Type Inference und Kurzformen der Syntax | 295 |
| A.1.4 | Methodenreferenzen | 296 |
| A.2 | Streams im Überblick | 297 |
| A.2.1 | Streams erzeugen – Create Operations | 298 |
| A.2.2 | Intermediate und Terminal Operations im Überblick | 299 |
| A.2.3 | Zustandslose Intermediate Operations | 301 |
| A.2.4 | Zustandsbehaftete Intermediate Operations | 303 |
| A.2.5 | Terminal Operations | 304 |
| A.3 | Neuerungen in der Datumsverarbeitung | 307 |
| A.3.1 | Die Klasse <code>Instant</code> | 308 |
| A.3.2 | Die Klassen <code>LocalDate</code> , <code>LocalTime</code> und <code>LocalDateTime</code> | 308 |
| A.3.3 | Die Klasse <code>Duration</code> | 310 |
| A.3.4 | Die Klasse <code>Period</code> | 311 |
| A.3.5 | Datumsarithmetik mit <code>TemporalAdjusters</code> | 312 |
| A.4 | Diverse Erweiterungen | 314 |
| A.4.1 | Erweiterungen im Interface <code>Comparator<T></code> | 314 |
| A.4.2 | Die Klasse <code>Optional<T></code> | 316 |
| A.4.3 | Die Klasse <code>CompletableFuture<T></code> | 319 |
| B | Einführung Gradle | 323 |
| B.1 | Projektstruktur für Maven und Gradle | 323 |
| B.2 | Builds mit Gradle | 325 |
| C | Einführung Maven | 335 |
| C.1 | Maven im Überblick | 335 |
| C.2 | Maven am Beispiel | 339 |
| | Literaturverzeichnis | 343 |
| | Index | 345 |