

Inhaltsverzeichnis

| | | |
|----------|---|-------------|
| | Vorwort | xiii |
| 1 | Einleitung | 1 |
| 1.1 | Elementare Konzepte und Begriffe | 5 |
| 1.2 | Funktionale Programmierung in Java | 10 |
| 2 | Sprachliche Grundlagen | 11 |
| 2.1 | Java Generics | 11 |
| 2.1.1 | Typparameter | 11 |
| 2.1.2 | Typconstraints | 12 |
| 2.1.3 | Ko- und Kontravarianz | 13 |
| 2.1.4 | Typinferenz bei Generics | 16 |
| 2.1.5 | Schwachstellen der Generics in Java | 18 |
| 2.2 | Default-Methoden | 19 |
| 2.3 | Lambda-Ausdrücke | 21 |
| 2.3.1 | Formen von Lambda-Ausdrücken | 24 |
| 2.3.2 | Typ eines Lambda-Ausdrucks | 24 |
| 2.3.3 | Ausnahmen bei Lambda-Ausdrücken | 25 |
| 2.3.4 | Closures | 27 |
| 2.4 | Funktionale Interfaces | 29 |
| 2.5 | Methodenreferenzen | 32 |
| 2.6 | Zusammenfassung | 33 |
| 3 | Programmieren ohne Seiteneffekte | 35 |
| 3.1 | Reine Funktionen | 35 |
| 3.1.1 | Iteration vs. Rekursion | 37 |
| 3.1.2 | Referentielle Transparenz und Ersetzungsprinzip | 38 |
| 3.1.3 | Funktionen mit Gedächtnis | 39 |
| 3.2 | Funktionale Ausnahmebehandlung mit Optional | 41 |

| | | |
|----------|---|-----------|
| 3.3 | Funktionale Listen | 43 |
| 3.3.1 | Beispielanwendung | 49 |
| 3.4 | Paare und Tupel | 51 |
| 3.5 | Zusammenfassung | 52 |
| 4 | Programmieren mit Funktionsparametern | 55 |
| 4.1 | Listenverarbeitung mit Funktionen höherer Ordnung | 56 |
| 4.2 | Flexible Programmschnittstellen | 59 |
| 4.3 | Algorithmen | 63 |
| 4.3.1 | Tiefensuche | 63 |
| 4.3.2 | Verallgemeinerung der Suche | 66 |
| 4.4 | Entwurfsmuster | 69 |
| 4.4.1 | Strategie | 69 |
| 4.4.2 | Kommando | 70 |
| 4.4.3 | Besucher | 73 |
| 4.5 | Eingebettete und bedingte Ausführung | 78 |
| 4.5.1 | Eingebetteter Code | 78 |
| 4.5.2 | Bedingte Ausführung | 80 |
| 4.5.3 | Fallunterscheidungen | 82 |
| 4.5.4 | Typtests | 85 |
| 4.6 | Auswertung nach Bedarf | 88 |
| 4.6.1 | Faule Iteratoren | 88 |
| 4.6.2 | Unendliche Folgen | 90 |
| 4.6.3 | Faule Iteration über die Knoten eines Graphen | 91 |
| 4.7 | Zusammenfassung | 92 |
| 5 | Kombination von Funktionen | 95 |
| 5.1 | Flüssige Schnittstellen | 95 |
| 5.2 | Funktionskomposition | 96 |
| 5.2.1 | Aufrufketten beim funktionalen Interface Function | 96 |
| 5.2.2 | Logische Verknüpfungen bei Predicate | 98 |
| 5.2.3 | Bilden von Vergleichsketten mit Comparator | 99 |
| 5.2.4 | Beispiel-Workflows | 102 |
| 5.3 | Kombinator-Parser | 105 |
| 5.3.1 | Parser und Parser-Ergebnisse | 105 |
| 5.3.2 | Kombinationsoperatoren | 109 |
| 5.3.3 | Parser für Boolesche Ausdrücke | 112 |
| 5.4 | Domänen-spezifische Sprachen | 116 |
| 5.4.1 | Fallbeispiel Zustandsmaschinen | 117 |
| 5.5 | Zusammenfassung | 122 |

| | | |
|----------|--|------------|
| 6 | Funktoren, Monoide und Monaden | 123 |
| 6.1 | Funktoren | 124 |
| 6.1.1 | Funktor Optional | 125 |
| 6.1.2 | Gesetze und Eigenschaften | 126 |
| 6.2 | Monoide und Reduktion | 128 |
| 6.2.1 | Monoide | 128 |
| 6.2.2 | Reduktion | 129 |
| 6.2.3 | Monoide in Java | 130 |
| 6.2.4 | Reduzierbare Strukturen | 132 |
| 6.2.5 | Anwendungsbeispiele zur Reduktion mit Monoiden | 136 |
| 6.3 | Monaden | 139 |
| 6.3.1 | Monade Optional | 142 |
| 6.3.2 | Monade Parser | 143 |
| 6.3.3 | Gesetze | 144 |
| 6.3.4 | Bedeutung von Monaden | 145 |
| 6.3.5 | MonadPlus: Monade mit monoider Kombination | 153 |
| 6.4 | Zusammenfassung | 154 |
| 7 | Streams | 155 |
| 7.1 | Grundlagen von Streams | 155 |
| 7.1.1 | Ein erstes Beispiel | 156 |
| 7.1.2 | Externe vs. interne Iteration | 156 |
| 7.1.3 | Bedarfsauswertung | 157 |
| 7.2 | Klassen von Streams | 160 |
| 7.3 | Stream-Operationen | 162 |
| 7.3.1 | Erzeuger-Operationen | 163 |
| 7.3.2 | Zwischenoperationen | 166 |
| 7.3.3 | Terminal-Operationen | 168 |
| 7.4 | Collectors | 172 |
| 7.4.1 | Interface Collector | 172 |
| 7.4.2 | Vordefinierte Collectors | 174 |
| 7.4.3 | Downstream Collectors | 176 |
| 7.4.4 | Eine eigene Collector-Implementierung | 178 |
| 7.5 | Anwendungsbeispiele | 180 |
| 7.5.1 | Ergebnisauswertung mit Streams | 181 |
| 7.5.2 | Wortindex zu einem Text | 184 |
| 7.6 | Hinweise | 187 |
| 7.6.1 | Einmal-Iteration | 187 |
| 7.6.2 | Begrenzung von unendlichen Streams | 188 |
| 7.6.3 | Zustandslose und zustandsbehaftete Operationen | 188 |
| 7.6.4 | Reihenfolge von Operationen | 189 |

| | | |
|-----------|---|------------|
| 7.6.5 | Kombinationen von Operationen | 190 |
| 7.7 | Interne Implementierung | 191 |
| 7.7.1 | Beispiel | 193 |
| 7.8 | Zusammenfassung | 194 |
| 8 | Parallele Streams | 195 |
| 8.1 | Erzeugen von parallelen Streams | 195 |
| 8.2 | Parallele Ausführung | 196 |
| 8.2.1 | Splitterators | 199 |
| 8.2.2 | Ausführung durch Fork/Join-Pool | 201 |
| 8.2.3 | Konfiguration des Fork/Join-Thread-Pools | 204 |
| 8.3 | Bedingungen bei paralleler Ausführung | 205 |
| 8.3.1 | Parallele Ausführung und Seiteneffekte | 205 |
| 8.3.2 | Parallele Ausführung und zustandsbehaftete Berechnungen | 206 |
| 8.3.3 | Eigenschaften der Parameter von reduce | 207 |
| 8.3.4 | Paralleles Sammeln | 208 |
| 8.4 | Laufzeit | 210 |
| 8.5 | Zusammenfassung | 212 |
| 9 | Asynchrone Funktionsketten | 213 |
| 9.1 | Eine Lösung mit parallelen Streams | 214 |
| 9.2 | Asynchrone Lösung mit Futures | 216 |
| 9.3 | CompletableFuture | 218 |
| 9.4 | Asynchrone Programmschnittstellen | 219 |
| 9.5 | CompletableFuture als Promise | 219 |
| 9.6 | Kombination von CompletableFuture | 221 |
| 9.6.1 | Beispiel | 223 |
| 9.7 | Zusammenfassung | 225 |
| 10 | Reaktive Streams | 227 |
| 10.1 | Grundlagen | 228 |
| 10.1.1 | Kontrakt von Observable | 233 |
| 10.1.2 | Erzeugen von Observables | 234 |
| 10.1.3 | Anmelden und Abmelden von Observer | 236 |
| 10.2 | Varianten | 238 |
| 10.2.1 | Single | 238 |
| 10.2.2 | Completable | 240 |
| 10.2.3 | Maybe | 241 |

| | | |
|-----------|--|------------|
| 10.3 | Hot und Cold Observables | 243 |
| 10.3.1 | ConnectableObservable | 244 |
| 10.3.2 | Beispiel Echtzeitdaten | 245 |
| 10.4 | Operationen | 247 |
| 10.4.1 | Abbildungen | 249 |
| 10.4.2 | Filtern und Teilmengen | 250 |
| 10.4.3 | Reduktion | 251 |
| 10.4.4 | Sammeln | 252 |
| 10.4.5 | Operationen mit Zeit | 255 |
| 10.4.6 | Kombinationen | 257 |
| 10.4.7 | Konvertierungen | 258 |
| 10.4.8 | Seiteneffekte | 258 |
| 10.5 | Nebenläufigkeit | 259 |
| 10.5.1 | Serialisierung von nebenläufigen Ereignissen | 259 |
| 10.5.2 | subscribeOn und Scheduler | 260 |
| 10.5.3 | observeOn | 261 |
| 10.6 | Fehlerbehandlung | 263 |
| 10.6.1 | Fehlerereignisse auslösen | 263 |
| 10.6.2 | Auf Fehler reagieren | 264 |
| 10.7 | Rückstau und Flusskontrolle | 266 |
| 10.7.1 | Reduktion der Menge der Ereignisse | 266 |
| 10.7.2 | Flowables | 267 |
| 10.8 | Testen reaktiver Streams | 270 |
| 10.9 | Zusammenfassung | 272 |
| 11 | Testen mit und von Funktionen | 273 |
| 11.1 | Funktionsparameter bei JUnit 5 | 273 |
| 11.2 | AssertJ: Eine DSL für Unit-Tests | 276 |
| 11.3 | Eigenschaftsbasiertes Testen nach QuickCheck | 277 |
| 11.3.1 | Generatoren von Zufallswerten | 279 |
| 11.3.2 | Tests | 281 |
| 11.3.3 | Shrinken der Werte | 284 |
| 11.4 | Zusammenfassung | 287 |
| 12 | Weiterführende Konzepte | 289 |
| A | Bibliografie | 299 |
| B | Laufzeitexperimente Parallele Streams | 305 |
| | Index | 317 |