Inhaltsverzeichnis

1 Einführung 1
1.1 Dimensionen der Parallelität 1
1.2 Parallelität und Nebenläufigkeit 2
1.2.1 Die Vorteile von Nebenläufigkeit 3
1.2.2 Die Nachteile von Nebenläufigkeit 3
1.2.3 Sicherer Umgang mit Nebenläufigkeit 4
1.3 Maße für die Parallelisierung 4
1.3.1 Die Gesetze von Amdahl und Gustafson 4
1.3.2 Work-Span-Analyse 6
1.4 Parallelitätsmodelle 7

I Grundlegende Konzepte 9

2 Das Thread-Konzept von Java 11
2.1 Der main-Thread 11
2.2 Erzeugung und Starten von Threads 12
2.2.1 Thread-Erzeugung durch Vererbung 13
2.2.2 Thread-Erzeugung mit Runnable-Objekten 16
2.3 Der Lebenszyklus von Threads 19
2.3.1 Beendigung eines Threads 20
2.3.2 Auf das Ende eines Threads warten 21
2.3.3 Aktives Beenden von Threads 21
2.3.4 Unterbrechung mit interrupt 24
2.3.5 Thread-Zustände 26
2.4 Weitere Eigenschaften eines Thread-Objekts 27
2.4.1 Thread-Priorität 27
2.4.2 Daemon-Eigenschaft 28
2.5 Exception-Handler 29
2.6 Zusammenfassung 31

Bibliografische Informationen
http://d-nb.info/108464701X

digitalisiert durch DEUTSCHE NATIONAL BIBLIOTHEK

3 Konkurrierende Zugriffe auf Daten 33
3.1 Ein einleitendes Beispiel 33
3.2 Java-Speichermodell 34
3.2.1 Stacks und Heap 35
3.2.2 Speicher auf der Hardwareebene 37
3.2.3 Probleme mit gemeinsam nutzbaren Daten 38
3.2.4 Sequenzielle Konsistenz 39
3.2.5 Thread-sichere Daten und unveränderliche Objekte 41
3.3 Unveränderbare Objekte 42
3.4 Volatile-Attribute 43
3.5 Final-Attributte 45
3.6 Thread-lokale Daten 46
3.7 Fallstricke 49
3.8 Zusammenfassung 50

4 Elementare Synchronisationsmechanismen 51
4.1 Schlüsselwort synchronized 51
4.1.1 Synchronized-Methoden 51
4.1.2 Synchronized-Blöcke 53
4.1.3 Beispiel: Thread-sicheres Singleton 54
4.1.4 Monitorkonzept bei Java 56
4.2 Fallstricke 57
4.3 Zusammenfassung 62

5 Grundlegende Thread-Steuerung 63
5.1 Bedingungsvariablen und Signalisieren 63
5.2 Regeln zum Umgang mit wait, notify und notifyAll 69
5.3 Zusammenfassung 72

II Weiterführende Konzepte 73

6 Threadpools 75
6.1 Das Poolkonzept und die Klasse Executors 75
6.1.1 Executors mit eigener ThreadFactory 78
6.1.2 Explizite ThreadPoolExecutor-Erzeugung 78
6.1.3 Benutzerdefinierter ThreadPoolExecutor 79
6.2 Future- und Callable-Schnittstelle 80
6.2.1 Callable, Future und FutureTask 81
6.2.2 Callable, Future und ExecutorService 81
6.3 Callable und ThreadPoolExecutor 84
6.4 Callable und ScheduledThreadPoolExecutor 88
6.5 Callable und ForkJoinPool 88

6.6 Exception-Handling ... 90
6.7 Tipps für das Arbeiten mit Threadpools ... 92
6.8 Zusammenfassung ... 93

7 Atomic-Variablen ... 95
7.1 Compare-and-Set-Operation ... 96
7.2 Umgang mit Atomic-Variablen ... 97
7.2.1 Atomic-Skalare ... 97
7.2.2 Atomic-Referenzen ... 100
7.3 Accumulator und Adder in Java 8 ... 102
7.4 Zusammenfassung ... 104

8 Lock-Objekte und Semaphore ... 105
8.1 Lock-Objekte ... 106
8.1.1 Das Lock-Interface ... 106
8.1.2 ReentrantLock ... 109
8.1.3 Das Condition-Interface ... 111
8.1.4 ReadWriteLock ... 115
8.1.5 StampedLock ... 117
8.2 Semaphore ... 120
8.3 Zusammenfassung ... 123

9 Thread-sichere Container ... 125
9.1 Collection-Typen ... 125
9.2 Thread-sichere Collections ... 127
9.2.1 Synchronisierte Collections ... 127
9.2.2 Unmodifiable Collections ... 129
9.2.3 Concurrent Collections ... 130
9.3 Zusammenfassung ... 135

III Ergänzende Synchronisationsmechanismen 137

10 Exchanger und BlockingQueue ... 139
10.1 Exchanger ... 139
10.2 Queues ... 143
10.3 Das Erzeuger-Verbraucher-Muster ... 146
10.4 Varianten ... 149
10.4.1 Pipeline von Erzeugern und Verbrauchern ... 149
10.4.2 Erzeuger-Verbraucher-Muster mit Empfangsbestätigung ... 150
10.4.3 Erzeuger-Verbraucher-Muster mit Work-Stealing ... 151
10.5 Zusammenfassung ... 157

11 CountDownLatch und CyclicBarrier 159
11.1 CountDownLatch 159
11.2 CyclicBarrier 162
11.3 Zusammenfassung 167

12 Phaser 169
12.1 Das Konzept des Phasers 169
12.1.1 Phaser als CountDownLatch 170
12.1.2 Phaser als CyclicBarrier 173
12.2 Phaser als variable Barriere 174
12.3 Zusammenspiel mit dem ForkJoin-Threadpool 178
12.4 Zusammenfassung 179

IV Parallelisierungsframeworks 181

13 Das ForkJoin-Framework 183
13.1 Grundprinzip des ForkJoin-Patterns 183
13.2 Programmiermodell 184
13.2.1 Einsatz von RecursiveAction 186
13.2.2 Einsatz von RecursiveTask 189
13.2.3 Einsatz von CountedCompleter 191
13.3 Work-Stealing-Verfahren 194
13.4 Zusammenfassung 197

14 Parallele Array- und Stream-Verarbeitung 199
14.1 Parallele Array-Verarbeitung 199
14.1.1 Parallele Transformation 199
14.1.2 Paralleles Sortieren 200
14.1.3 Parallele Präfixbildung 201
14.2 Funktionsprinzip der Stream-Verarbeitung 203
14.2.1 Funktionale Interfaces 204
14.2.2 Erzeugung von Streams 205
14.2.3 Transformations- und Manipulationsoperationen 208
14.2.4 Auswertungen von Streams 211
14.2.5 Eigenschaften und Operationsoptimierung 216
14.3 Parallele Stream-Verarbeitung: Datenparallelität 217
14.3.1 Arbeitsweise und korrekte Benutzung 218
14.3.2 Parallele Reduzierer 220
14.3.3 Parallele Collectoren 223
14.3.4 Funktionsweise von Spliteratoren 228
14.3.5 Benutzerdefinierte Spliteratoren 230
14.4 Zusammenfassung 235

15 CompletableFuture ... 237
15.1 CompletableFuture als Erweiterung des Future-Patterns ... 237
15.2 Design von asynchronen APIs ... 241
15.2.1 Asynchrone APIs mit Future ... 242
15.2.2 Asynchrone APIs mit CompletableFuture ... 242
15.3 Asynchrone Verarbeitung: Task-Parallelität ... 244
15.3.1 Das Starten einer asynchronen Verarbeitung ... 244
15.3.2 Definition einer asynchronen Verarbeitungskette ... 245
15.4 Das Arbeiten mit CompletableFutures ... 247
15.4.1 Das Konzept des CompletionStage ... 248
15.4.2 Lineare Kompositionsmöglichkeiten ... 249
15.4.3 Verzweigen und Vereinen ... 252
15.4.4 Synchronisationsbarrieren ... 256
15.5 Fehlerbehandlung und Abbruch einer Verarbeitung ... 257
15.6 Zusammenfassung ... 259

V Fallbeispiele 261

16 Asynchrones Logging ... 263
16.1 Lösung mit Thread-lokalen Daten ... 264
16.2 Verbesserte Version (Exchanger) ... 266

17 Datenstrukturen in Multithreaded-Umgebungen ... 271
17.1 Liste als sortierte Menge ... 271
17.2 Blockierende Lösungen (Locks) ... 276
17.2.1 Grobgranulare Synchronisierung ... 276
17.2.2 Feingranulare Synchronisierung ... 276
17.2.3 Optimistische Synchronisierung ... 279
17.3 Lockfreie Lösung (AtomicMarkableReference) ... 280

18 The Dining Philosophers Problem ... 287
18.1 Basisalgorithmus ... 288
18.2 Lösungsvarianten (Semaphore und Lock) ... 288
18.2.1 Lösung mit einem Semaphor ... 289
18.2.2 Lösung mit asymmetrischer Lock-Anforderung ... 290
18.2.3 Lösung mithilfe eines Koordinators ... 291
18.2.4 Lösung mit asymmetrischer Wait-Release-Strategie ... 293

19 Minimal aufspannende Bäume ... 295
19.1 Graphen und Spannbäume ... 295
19.2 Der Prim-Algorithmus ... 297
19.2.1 Funktionsweise des Algorithmus ... 297

19.2.2 Implementierung des Algorithmus ... 299
19.3 Parallelisierung (Phaser) ... 301

20 Mergesort ... 305
20.1 Funktionsprinzip des Algorithmus ... 305
20.2 Parallelisierung (ForkJoin-Framework) ... 307

21 Der k-Mean-Clusteralgorithmus ... 309
21.1 Der k-Mean-Algorithmus ... 309
21.2 Parallelisierung (Parallel Streams) ... 311
21.2.1 Datenmodell ... 311
21.2.2 Hilfsmethoden ... 311
21.2.3 Implementierung ... 312
21.2.4 Variante mit benutzerdefiniertem Collector ... 316

22 RSA-Schlüsselerzeugung ... 321
22.1 Verfahren für die Schlüsselerzeugung ... 321
22.2 Parallelisierung (CompletableFuture) ... 323

23 Threads bei JavaFX ... 327
23.1 Ein einfaches Beispiel ... 327
23.2 JavaFX-Concurrent-API ... 329

24 Handler-Konzept bei Android ... 335
24.1 UI-Thread und nebenläufige Aktivitäten ... 335
24.2 Messages, Message-Queue, Looper ... 336
24.3 Handler ... 338

25 Aktoren ... 341
25.1 Aktorenmodell ... 341
25.2 Beispielimplementierung mit Akka ... 342
25.2.1 Nachrichten ... 343
25.2.2 Beteiligte Aktoren ... 345
25.2.3 Starten der Anwendung ... 347

VI Anhang 349

A Ausblick auf Java 9 ... 351
A.1 Die Flow-Interfaces ... 351

Literaturverzeichnis ... 357

Index ... 361