

Inhaltsverzeichnis

1	Aufbau & Gliederung *	1
2	Nebenläufigkeit: Schnelleinstieg **	5
2.1	Anwendungen vs. Prozesse **	6
2.2	Programme & ihre Ausführung **	9
2.3	threads & scheduling **	16
2.4	Vorteile & Probleme nebenläufiger Programme **	19
2.4.1	Verbesserung der Performance **	21
2.4.2	Synchronisation **	26
2.4.3	Realisierung kritischer Abschnitte **	32
2.4.4	Monitore **	39
2.4.5	Lebendigkeit **	45
2.4.6	Verklemmungen **	50
3	Threads in Java: Schnelleinstieg **	57
3.1	Hello World mit threads **	57
3.2	Die Schnittstelle Runnable **	60
3.3	Zum Entwurf nebenläufiger Anwendungen **	65
3.3.1	Freizeitpark-Simulation **	68
3.4	Synchronisation von threads **	71
3.5	Monitore in Java *	78
3.6	Verklemmungen in Java **	86
3.7	Lagerverwaltung *	93
3.7.1	LVS 1: nebenläufige Roboter **	95
3.7.2	LVS 2: Synchronisation **	99
4	UML-Modellierung von Nebenläufigkeit **	101
4.1	Klassen- und Objekt-Diagramme **	102
4.2	Aktivitäts-Diagramme **	105
4.3	Interaktions- und Sequenz-Diagramme **	111
4.4	Zustandsautomaten **	117
5	Neues zur Nebenläufigkeit in Java 5 **	123
5.1	Zeiteinheiten angeben mit TimeUnit **	126
5.2	Nebenläufige Container-Klassen *	130
5.2.1	Synchronisation vs. Nebenläufigkeit **	131
5.2.2	Warteschlangen **	135
5.2.3	Warteschlangen mit Prioritäten *	143
5.2.4	Container und Copy-On-Write-Semantik **	147
5.2.5	Nebenläufige HashMap **	154

- 5.3 Auftragsorientierte Architektur *** 155
 - 5.3.1 Ausführungsdienst für Aufträge ** 157
 - 5.3.2 Implementierungen von ExecutorService *** 163
 - 5.3.3 Aufträge mit Ergebnis ** 166
- 5.4 Flexible Monitore ** 171
- 5.5 Leser-Schreiber-Synchronisation ** 181
- 5.6 Synchronisations-Objekte in Java 5 * 186
- 5.7 Atomare Operationen *** 191
- 6 Fortgeschrittene Java-Konzepte für Nebenläufigkeit *** 195**
 - 6.1 Swing-GUIs und Nebenläufigkeit * 196
 - 6.1.1 Arbeits-Thread: Hauptfenster ** 202
 - 6.1.2 Arbeits-Thread: Fortschritts-Dialog ** 205
 - 6.1.3 Arbeits-Thread: Sortier-Thread * 207
 - 6.2 LVS 5: GUI ** 209
 - 6.3 Threads kontrolliert beenden ** 212
 - 6.3.1 Produktionsstrasse ** 214
 - 6.3.2 Beenden mit stop() ** 215
 - 6.3.3 shutdown flag ** 217
 - 6.3.4 Threads unterbrechen ** 221
 - 6.3.5 Herunterfahren oder Unterbrechen ** 224
 - 6.4 Java-Threads im Details *** 228
 - 6.4.1 Die Klasse Thread ** 229
 - 6.4.2 Caches & Synchronisation *** 236
 - 6.4.3 Sicherung von run() *** 243
- 7 Realisierung von Nebenläufigkeit *** 245**
 - 7.1 Stapel & Halde *** 246
 - 7.2 Speicherverwaltung mit mehreren threads ** 251
 - 7.3 Realisierung von threads 1.0 ** 253
 - 7.4 Stapel & Halde in Java ** 259
 - 7.5 Prozessor & BS * 264
- 8 Verteilte Anwendungen ** 267**
 - 8.1 Hello World mit Java-RMI * 269
 - 8.1.1 RMI-Begriffe * 279
 - 8.2 Stummel-Objekte ** 281
 - 8.2.1 Identität und Gleichheit von Stummel-Objekten * 288
 - 8.3 Objekte als RMI-Parameter ** 291

- 8.3.1 Arrays und Sammlungen als
RMI-Parameter ** 299
- 8.4 Verteilung & Installation *** 301
 - 8.4.1 Bereitstellung von class-Dateien *** 304
 - 8.4.2 Abläufe beim dynamischen Laden *** 309
- 8.5 RMI & UML * 311
 - 8.5.1 Artefakte ** 312
 - 8.5.2 Verteilungs-Diagramme ** 315
- 8.6 Entwurfsprinzipien für verteilte
Anwendungen ** 320
 - 8.6.1 Fabrik-Dienst ** 322
 - 8.6.2 Wert-Objekte ** 324
 - 8.6.3 Fassaden ** 328
- 8.7 RMI und Nebenläufigkeit *** 332
- 8.8 LVS 6: RMI-Unterstützung ** 339
- 8.9 LVS 7: Optimierungen ** 344

Glossar 347**Literatur 355**

- A Innere Klassen in Java ** 357**
- B Generische Klassen in Java ** 363**
- C Petri-Netze ** 371**

Sachindex 375