

# Inhaltsverzeichnis

<b>Vorwort zur 6. Auflage</b> .....	<b>V</b>
<b>1 Einleitung</b> .....	<b>1</b>
1.1 Parallelität, Nebenläufigkeit und Verteilung .....	1
1.2 Programme, Prozesse und Threads .....	2
<b>2 Grundlegende Synchronisationskonzepte in Java</b> .....	<b>6</b>
2.1 Erzeugung und Start von Java-Threads .....	6
2.1.1 Ableiten der Klasse Thread .....	6
2.1.2 Implementieren der Schnittstelle Runnable .....	8
2.1.3 Einige Beispiele .....	11
2.1.4 Parallele Abläufe .....	18
2.2 Probleme beim Zugriff auf gemeinsam genutzte Objekte .....	19
2.2.1 Erster Lösungsversuch .....	22
2.2.2 Zweiter Lösungsversuch .....	23
2.3 synchronized und volatile .....	25
2.3.1 Synchronized-Methoden .....	25
2.3.2 Synchronized-Blöcke .....	27
2.3.3 Wirkung von synchronized .....	28
2.3.4 Notwendigkeit von synchronized .....	30
2.3.5 volatile .....	31
2.3.6 Regel für die Nutzung von synchronized .....	31
2.4 Ende von Java-Threads .....	33
2.4.1 Asynchrone Beauftragung mit Abfragen der Ergebnisse .....	34
2.4.2 Zwangsweises Beenden von Threads .....	40
2.4.3 Asynchrone Beauftragung mit befristetem Warten .....	45

2.4.4	Asynchrone Beauftragung mit Rückruf (Callback) .....	47
2.4.5	Asynchrone Beauftragung mit Rekursion .....	50
2.5	wait und notify .....	54
2.5.1	Erster Lösungsversuch .....	55
2.5.2	Zweiter Lösungsversuch .....	55
2.5.3	Dritter Lösungsversuch .....	57
2.5.4	Korrekte und effiziente Lösung mit wait und notify .....	58
2.6	NotifyAll .....	67
2.6.1	Erzeuger-Verbraucher-Problem mit wait und notify .....	67
2.6.2	Erzeuger-Verbraucher-Problem mit wait und notifyAll .....	71
2.6.3	Faires Parkhaus mit wait und notifyAll .....	74
2.7	Prioritäten von Threads .....	76
2.8	Thread-Gruppen .....	84
2.9	Vordergrund- und Hintergrund-Threads .....	88
2.10	Weitere „gute“ und „schlechte“ Thread-Methoden .....	90
2.11	Thread-lokale Daten .....	91
2.12	Zusammenfassung .....	94
<b>3</b>	<b>Fortgeschrittene Synchronisationskonzepte in Java .....</b>	<b>99</b>
3.1	Semaphore .....	100
3.1.1	Einfache Semaphore .....	100
3.1.2	Einfache Semaphore für den gegenseitigen Ausschluss .....	101
3.1.3	Einfache Semaphore zur Herstellung vorgegebener Ausführungsreihenfolgen .....	103
3.1.4	Additive Semaphore .....	106
3.1.5	Semaphorgruppen .....	109
3.2	Message Queues .....	112
3.2.1	Verallgemeinerung des Erzeuger-Verbraucher-Problems .....	112
3.2.2	Übertragung des erweiterten Erzeuger-Verbraucher-Problems auf Message Queues .....	114
3.3	Pipes .....	117
3.4	Philosophen-Problem .....	120
3.4.1	Lösung mit synchronized - wait - notifyAll .....	121
3.4.2	Naive Lösung mit einfachen Semaphoren .....	124

3.4.3	Einschränkende Lösung mit gegenseitigem Ausschluss .....	125
3.4.4	Gute Lösung mit einfachen Semaphoren .....	126
3.4.5	Lösung mit Semaphorgruppen .....	130
3.5	Leser-Schreiber-Problem .....	132
3.5.1	Lösung mit synchronized - wait - notifyAll .....	133
3.5.2	Lösung mit additiven Semaphoren .....	136
3.6	Schablonen zur Nutzung der Synchronisationsprimitive und Konsistenzbetrachtungen .....	138
3.7	Concurrent-Klassenbibliothek aus Java 5 .....	142
3.7.1	Executors .....	143
3.7.2	Locks und Conditions .....	149
3.7.3	Atomic-Klassen .....	157
3.7.4	Synchronisationsklassen .....	161
3.7.5	Queues .....	164
3.8	Das Fork-Join-Framework von Java 7 .....	165
3.8.1	Grenzen von ThreadPoolExecutor .....	165
3.8.2	ForkJoinPool und RecursiveTask .....	167
3.8.3	Beispiel zur Nutzung des Fork-Join-Frameworks .....	169
3.9	Das Data-Streaming-Framework von Java 8 .....	171
3.9.1	Einleitendes Beispiel .....	172
3.9.2	Sequenzielles Data-Streaming .....	174
3.9.3	Paralleles Data-Streaming .....	177
3.10	Die CompletableFutures von Java 8 .....	179
3.11	Ursachen für Verklemmungen .....	185
3.11.1	Beispiele für Verklemmungen mit synchronized .....	186
3.11.2	Beispiele für Verklemmungen mit Semaphoren .....	190
3.11.3	Bedingungen für das Eintreten von Verklemmungen .....	191
3.12	Vermeidung von Verklemmungen .....	192
3.12.1	Anforderung von Betriebsmitteln „auf einen Schlag“ .....	195
3.12.2	Anforderung von Betriebsmitteln gemäß einer vorgegebenen Ordnung .....	196
3.12.3	Weitere Verfahren .....	197
3.13	Zusammenfassung .....	199

<b>4</b>	<b>Parallelität und grafische Benutzeroberflächen</b>	<b>200</b>
4.1	Einführung in die Programmierung grafischer Benutzeroberflächen mit JavaFX	201
4.1.1	Allgemeines zu grafischen Benutzeroberflächen	201
4.1.2	Erstes JavaFX-Beispiel	202
4.1.3	Ereignisbehandlung	203
4.2	Properties, Bindings und JavaFX-Collections	207
4.2.1	Properties	207
4.2.2	Bindings	210
4.2.3	JavaFX-Collections	211
4.3	Elemente von JavaFX	212
4.3.1	Container	212
4.3.2	Interaktionselemente	215
4.3.3	Grafikprogrammierung	217
4.3.4	Weitere Funktionen von JavaFX	223
4.4	MVP	224
4.4.1	Prinzip von MVP	224
4.4.2	Beispiel zu MVP	226
4.5	Threads und JavaFX	232
4.5.1	Threads für JavaFX	232
4.5.2	Länger dauernde Ereignisbehandlungen	234
4.5.3	Beispiel Stoppuhr	239
4.5.4	Tasks und Services in JavaFX	244
4.6	Zusammenfassung	253
<b>5</b>	<b>Verteilte Anwendungen mit Sockets</b>	<b>254</b>
5.1	Einführung in das Themengebiet der Rechnernetze	255
5.1.1	Schichtenmodell	255
5.1.2	IP-Adressen und DNS-Namen	259
5.1.3	Das Transportprotokoll UDP	259
5.1.4	Das Transportprotokoll TCP	261
5.2	Socket-Schnittstelle	262
5.2.1	Socket-Schnittstelle zu UDP	262

5.2.2	Socket-Schnittstelle zu TCP .....	263
5.2.3	Socket-Schnittstelle für Java .....	266
5.3	Kommunikation über UDP mit Java-Sockets .....	267
5.4	Multicast-Kommunikation mit Java-Sockets .....	276
5.5	Kommunikation über TCP mit Java-Sockets .....	280
5.6	Sequenzielle und parallele Server .....	292
5.6.1	TCP-Server mit dynamischer Parallelität .....	293
5.6.2	TCP-Server mit statischer Parallelität .....	297
5.6.3	Sequenzieller, „verzahnt“ arbeitender TCP-Server .....	302
5.6.4	Horizontale Skalierung mit Lastbalancierung .....	305
5.7	Verschlüsselte Kommunikation über TLS .....	306
5.8	Zusammenfassung .....	310
<b>6</b>	<b>Verteilte Anwendungen mit RMI .....</b>	<b>311</b>
6.1	Prinzip von RMI .....	311
6.2	Einführendes RMI-Beispiel .....	314
6.2.1	Basisprogramm .....	314
6.2.2	RMI-Client mit grafischer Benutzeroberfläche .....	318
6.2.3	RMI-Registry .....	323
6.3	Parallelität bei RMI-Methodenaufrufen .....	327
6.4	Wertübergabe für Parameter und Rückgabewerte .....	331
6.4.1	Serialisierung und Deserialisierung von Objekten .....	332
6.4.2	Serialisierung und Deserialisierung bei RMI .....	336
6.5	Referenzübergabe für Parameter und Rückgabewerte .....	341
6.6	Transformation lokaler in verteilte Anwendungen .....	356
6.6.1	Rechnergrenzen überschreitende Synchronisation mit RMI .....	357
6.6.2	Asynchrone Kommunikation mit RMI .....	359
6.6.3	Verteilte MVP-Anwendungen mit RMI .....	360
6.7	Dynamisches Umschalten zwischen Wert- und Referenzübergabe - Migration von Objekten .....	361
6.7.1	Das Exportieren und „Unexportieren“ von Objekten .....	361
6.7.2	Migration von Objekten .....	364
6.7.3	Eintrag eines Nicht-Stub-Objekts in die RMI-Registry .....	371

6.8	Realisierung von Stubs und Skeletons .....	372
6.8.1	Realisierung von Skeletons .....	373
6.8.2	Realisierung von Stubs .....	374
6.9	Verschiedenes .....	376
6.10	Zusammenfassung .....	377
<b>7</b>	<b>Verteilte Anwendungen mit indirekter Kommunikation .....</b>	<b>378</b>
7.1	Prinzip der indirekten Kommunikation .....	379
7.2	Kommunikationsmodelle .....	381
7.2.1	Kommunikationsmodell Queue .....	381
7.2.2	Kommunikationsmodell Topic .....	382
7.3	Nutzung der indirekten Kommunikation in Java .....	383
7.4	Unidirektionale Kommunikation .....	385
7.5	Bidirektionale Kommunikation mithilfe eines Rückkanals .....	391
7.6	Empfangsbestätigungen .....	396
7.7	Transaktionen .....	397
7.8	Verschiedenes .....	398
<b>8</b>	<b>Webbasierte Anwendungen mit Servlets und JSF .....</b>	<b>401</b>
8.1	HTTP und HTML .....	402
8.1.1	GET .....	403
8.1.2	Formulare in HTML .....	406
8.1.3	POST .....	408
8.1.4	Format von HTTP-Anfragen und -Antworten .....	409
8.2	Einführende Servlet-Beispiele .....	409
8.2.1	Allgemeine Vorgehensweise .....	409
8.2.2	Erstes Servlet-Beispiel .....	411
8.2.3	Zugriff auf Formulardaten .....	413
8.2.4	Zugriff auf die Daten der HTTP-Anfrage und -Antwort .....	414
8.3	Parallelität bei Servlets .....	416
8.3.1	Demonstration der Parallelität von Servlets .....	416
8.3.2	Paralleler Zugriff auf Daten .....	418
8.3.3	Anwendungsglobale Daten .....	421

8.4	Sessions und Cookies	425
8.4.1	Sessions	425
8.4.2	Realisierung von Sessions mit Cookies	430
8.4.3	Direkter Zugriff auf Cookies	433
8.4.4	Servlets mit länger dauernden Aufträgen	434
8.5	Asynchrone Servlets	439
8.6	Filter	444
8.7	Übertragung von Dateien mit Servlets	445
8.7.1	Herunterladen von Dateien	445
8.7.2	Hochladen von Dateien	447
8.8	JSF (Java Server Faces)	450
8.8.1	Einführendes Beispiel	451
8.8.2	Managed Beans und deren Scopes	457
8.8.3	MVP-Prinzip mit JSF	461
8.8.4	AJAX mit JSF	463
8.9	RESTful WebServices	467
8.9.1	Definition von RESTful WebServices	468
8.9.2	JSON	469
8.9.3	Beispiel	471
8.10	WebSockets	476
8.11	Zusammenfassung	480
<b>9</b>	<b>Verteilte Anwendungen in der Cloud</b>	<b>483</b>
9.1	Cloud Computing	483
9.2	AWS (Amazon Web Services)	487
9.2.1	AWS-Infrastruktur	487
9.2.2	AWS-Dienste	488
9.2.3	Nutzung der AWS-Dienste	492
9.3	Nutzung der AWS-Dienste von außerhalb der Cloud	494
9.3.1	Nutzung des AWS-Dienstes S3	496
9.3.2	Nutzung des AWS-Dienstes DynamoDB	501
9.3.3	Nutzung des AWS-Dienstes Translate	507
9.4	Nutzung von EC2 als Server	512

9.5	Nutzung von ECS als Server .....	518
9.5.1	Isolationsstufen .....	518
9.5.2	Linux-Grundlagen für die Realisierung von Containern .....	520
9.5.3	Docker .....	523
9.5.4	ECS .....	528
9.6	Nutzung von Lambda als Server .....	529
9.6.1	Idee der zu entwickelnden Anwendung .....	531
9.6.2	Lambda-Funktion .....	532
9.6.3	API Gateway .....	537
9.6.4	Kommandozeilenbasierter Client .....	540
9.6.5	Java-basierter Client mit grafischer Benutzeroberfläche .....	542
	<b>Literatur .....</b>	<b>553</b>
	<b>Index .....</b>	<b>555</b>