

# Inhalt

<b>Vorwort</b>	<b>11</b>
Zum Buch	11
Danksagungen	12
<b>I    Java ist zu langsam</b>	<b>15</b>
1.1 Was ist Performance?	16
1.2 Empfundene Performance	18
1.3 Gesunder Menschenverstand	18
1.4 Wissen ist Performance	20
<b>2    Entwicklungsprozess</b>	<b>23</b>
2.1 Analyse und Design	25
2.2 Kodieren und Testen	26
2.3 Integrieren und Testen	27
2.3.1 Mikro- und Makro-Benchmarks	27
2.3.2 Testlänge	29
2.3.3 Auswertung	29
2.3.4 Wie häufig testen?	30
<b>3    Virtuelle Maschinen</b>	<b>31</b>
3.1 Bytecode-Ausführung	32
3.1.1 Interpreter	33
3.1.2 Just-in-Time-Compiler	33
3.1.3 Dynamisch angepasste Übersetzung	33
3.1.4 Ahead-of-Time-Übersetzung	36
3.1.5 Java in Silizium	38
3.2 Garbage Collection	38
3.2.1 Objekt-Lebenszyklus	39
3.2.2 Garbage Collection-Algorithmen	41
3.2.3 Performance-Maße	44
3.2.4 HotSpots Garbage Collection	44
3.3 Industrie-Benchmarks	46
3.3.1 VolanoMark	46
3.3.2 SPEC JVM98	47

3.3.3	SPEC JBB2000	48
3.3.4	jBYTEMark	48
3.3.5	ECperf	48
3.4	Die richtige VM auswählen	49
<b>4</b>	<b>Messwerkzeuge</b>	<b>51</b>
4.1	Profiler	52
4.2	Hprof	52
4.2.1	Speicherabbild erstellen	53
4.2.2	CPU-Profilng	58
4.2.3	Monitor-Information	64
4.3	HotSpot-Profilng	68
4.4	Jinsight	71
4.5	Mikro-Benchmarks	71
4.6	Makro-Benchmarks	72
4.7	Performance Metriken	72
4.8	Speicher-Schnittstellen	72
4.8.1	Speicherverbrauch	73
4.8.2	Geschwätzige Garbage Collection	81
4.8.3	Manuelle Speicherbereinigung	83
<b>5</b>	<b>Zeichenketten</b>	<b>85</b>
5.1	Strings einfügen	85
5.2	Strings anfügen	87
5.3	Bedingtes Erstellen von Strings	90
5.4	Stringvergleiche	92
5.5	Groß- und Kleinschreibung	95
5.5.1	Vergleich mittels equalsIgnoreCase()	96
5.5.2	toLowerCase() oder toUpperCase(), das ist hier die Frage	96
5.5.3	Wenn $\hat{A}$ gleich $a$ sein soll	100
5.6	Strings sortieren	103
5.7	Formatieren	104
5.7.1	Nachrichten erstellen	105
5.7.2	Datum und Zeit	106
5.8	String-Analyse	109
5.8.1	Datum und Zeit	112
5.8.2	Strings teilen	116
5.8.3	Reguläre Ausdrücke und lexikalische Analyse mit Grammatiken	120
<b>6</b>	<b>Bedingte Ausführung, Schleifen und Switches</b>	<b>121</b>
6.1	Bedingte Ausführung	121
6.1.1	Logische Operatoren	121
6.1.2	String-Switches	122
6.1.3	Befehlsobjekte	124
6.2	Schleifen	127
6.2.1	Loop Invariant Code Motion	128

6.2.2	Teure Array-Zugriffe	129
6.2.3	Loop Unrolling	130
6.2.4	Schleifen vorzeitig verlassen	131
6.2.5	Ausnahmeterminierte Schleifen	132
6.2.6	Iteratoren oder nicht?	134
6.3	Optimale Switches	137
<b>7</b>	<b>Ausnahmen</b>	<b>141</b>
7.1	Ausnahmen durch sinnvolle Schnittstellen vermeiden	141
7.2	Kosten von Try-Catch-Blöcken in Schleifen	143
7.3	Keine eigenen Ausnahme-Hierarchien	145
7.4	Automatisch loggende Ausnahmen	147
7.5	Ausnahmen wieder verwenden	147
<b>8</b>	<b>Datenstrukturen und Algorithmen</b>	<b>151</b>
8.1	Groß-O-Notation	151
8.2	Collections-Framework	153
8.2.1	Collections, Sets und Listen	153
8.2.2	Maps	156
8.2.3	Hashbasierte Strukturen optimieren	158
8.2.4	Collections	160
8.3	Jenseits des Collections-Frameworks	162
8.3.1	Zahlen sortieren	162
8.3.2	Große Tabellen	166
8.4	Caches	174
8.4.1	Austauschstrategien	175
8.4.2	Elementspezifische Invalidierung	176
8.4.3	Schreibverfahren	176
8.4.4	Gecachte Map	177
8.4.5	Caches mit LinkedHashMap	182
8.4.6	Schwache Referenzen	184
<b>9</b>	<b>Threads</b>	<b>187</b>
9.1	Gefährlich lebt sich's schneller	187
9.1.1	Sicherheit durch Synchronisation	189
9.1.2	Synchronisationskosten	189
9.1.3	Threadsichere Datenstrukturen	193
9.1.4	Double-Check-Idiom	194
9.1.5	Sprunghafte Variablen	196
9.2	Allgemeine Threadprogrammierung	196
9.2.1	Threads starten	196
9.2.2	Threadpool	197
9.2.3	Kommunikation zwischen Threads	204
9.2.4	Warten oder schlafen?	206
9.2.5	Prioritäten setzen und Vorrang lassen	206
9.3	Skalieren mit Threads	207

9.4	Threads in Benutzeroberflächen	211
9.4.1	Lebendige AWT-Oberflächen	211
9.4.2	Threads in Swing	215
<b>10</b>	<b>Effiziente Ein- und Ausgabe</b>	<b>217</b>
10.1	Fallstudie Dateikopieren	217
10.2	Texte ausgeben	221
10.3	Texte einlesen	226
10.4	Dateicache	227
10.5	Skalierbare Server	237
10.5.1	Httpd der alten Schule	238
10.5.2	Nicht-blockierender Httpd	242
10.5.3	Vergleichende Rechenspiele	255
<b>11</b>	<b>RMI und Serialisierung</b>	<b>259</b>
11.1	Effiziente Serialisierung	259
11.1.1	Datenmenge verkleinern	260
11.1.2	Optimierte logische Darstellung	266
11.2	Latenzzeiten und Overhead	270
11.3	Verteilte Speicherbereinigung	271
<b>12</b>	<b>XML</b>	<b>273</b>
12.1	SAX, DOM & Co	273
12.1.1	SAX	273
12.1.2	DOM	275
12.1.3	Pull-Parser	276
12.2	Kleiner Modellvergleich	279
12.3	Den richtigen Parser wählen	281
12.4	XML ausgeben	282
12.5	DOM-Bäume traversieren	285
12.6	XML komprimieren	286
12.6.1	HTTP	287
12.6.2	Binärformate	291
<b>13</b>	<b>Applikationen starten</b>	<b>301</b>
13.1	Klassen laden und initialisieren	301
13.2	Verzögertes Klassenladen	303
13.3	Frühes Klassenladen	304
13.4	Geschwätziges Klassenladen	306
13.5	Klassenarchive	307
13.6	Start-Fenster für große Applikationen	308
13.7	Mehrere Applikationen in einer VM starten	312
	<b>Letzte Worte</b>	<b>317</b>
	<b>Literatur</b>	<b>319</b>
	<b>Index</b>	<b>321</b>