

# Inhaltsverzeichnis

	<b>Vorbemerkung</b>	<b>xi</b>
	<b>Vorwort</b>	<b>xiii</b>
	<b>Danksagung</b>	<b>xiv</b>
<b>1</b>	<b>Einleitung</b>	<b>1</b>
<b>2</b>	<b>Objekte erzeugen und auflösen</b>	<b>5</b>
2.1	Thema 1: Statische Factory-Methoden als Alternative zu Konstruktoren .....	5
2.2	Thema 2: Erwägen Sie bei zu vielen Konstruktorparametern den Einsatz eines Builders .....	10
2.3	Thema 3: Erzwingen Sie die Singleton-Eigenschaft mit einem private-Konstruktor oder einem Aufzählungstyp ..	18
2.4	Thema 4: Erzwingen Sie die Nicht-Instanzierbarkeit mit einem private-Konstruktor .....	20
2.5	Thema 5: Arbeiten Sie mit Dependency Injection statt Ressourcen direkt einzubinden .....	21
2.6	Thema 6: Vermeiden Sie die Erzeugung unnötiger Objekte ..	23
2.7	Thema 7: Löschen Sie veraltete Objektreferenzen .....	27
2.8	Thema 8: Vermeiden Sie Finalizer und Cleaner .....	30
2.9	Thema 9: Verwenden Sie try-with-resources anstelle von try-finally .....	35
<b>3</b>	<b>Methoden, die allen Objekten gemeinsam sind</b>	<b>39</b>
3.1	Thema 10: Halten Sie beim Überschreiben von equals den allgemeinen Vertrag ein .....	39
3.2	Thema 11: Überschreiben Sie, wenn Sie equals überschreiben, immer auch hashCode .....	52

3.3	Thema 12: Überschreiben Sie immer toString . . . . .	57
3.4	Thema 13: Vorsicht beim Überschreiben von clone . . . . .	60
3.5	Thema 14: Denken Sie darüber nach, Comparable zu implementieren . . . . .	69
<b>4</b>	<b>Klassen und Schnittstellen</b>	<b>77</b>
4.1	Thema 15: Minimieren Sie den Zugriff auf Klassen und Member . . . . .	77
4.2	Thema 16: Verwenden Sie in öffentlichen Klassen Accessor-Methoden und keine öffentlichen Felder . . . . .	82
4.3	Thema 17: Bevorzugen Sie unveränderliche Klassen . . . . .	84
4.4	Thema 18: Ziehen Sie Komposition der Vererbung vor . . . . .	92
4.5	Thema 19: Entwerfen und dokumentieren Sie für Vererbung oder verbieten Sie sie . . . . .	98
4.6	Thema 20: Geben Sie Schnittstellen den Vorzug vor abstrakten Klassen . . . . .	104
4.7	Thema 21: Entwerfen Sie Ihre Schnittstellen für die Nachwelt . . . . .	109
4.8	Thema 22: Verwenden Sie Schnittstellen nur zum Definieren von Typen . . . . .	112
4.9	Thema 23: Arbeiten Sie mit Klassenhierarchien statt mit Tag-Klassen . . . . .	114
4.10	Thema 24: Ziehen Sie statische Member-Klassen den nicht-statischen vor . . . . .	117
4.11	Thema 25: Beschränken Sie Quelltextdateien auf eine einzige Toplevel-Klasse . . . . .	121
<b>5</b>	<b>Java Generics</b>	<b>123</b>
5.1	Thema 26: Hände weg von Rohtypen . . . . .	123
5.2	Thema 27: Eliminieren Sie unchecked-Warnungen . . . . .	129
5.3	Thema 28: Verwenden Sie Listen statt Arrays . . . . .	131
5.4	Thema 29: Bevorzugen Sie generische Typen . . . . .	136
5.5	Thema 30: Bevorzugen Sie generische Methoden . . . . .	141
5.6	Thema 31: Eingeschränkte Wildcard-Typen machen Ihre APIs flexibler . . . . .	145
5.7	Thema 32: Vorsicht beim Kombinieren von Java Generics mit varargs-Methoden . . . . .	152
5.8	Thema 33: Nutzen Sie typsichere heterogene Container . . . . .	157

---

<b>6</b>	<b>Aufzählungen und Annotationen</b>	<b>163</b>
6.1	Thema 34: Verwenden Sie Aufzählungen statt int-Konstanten . . . . .	163
6.2	Thema 35: Verwenden Sie Instanzfelder statt Ordinalzahlen	174
6.3	Thema 36: Verwenden Sie EnumSet statt Bitfelder . . . . .	175
6.4	Thema 37: Verwenden Sie EnumMap statt Ordinalzahlindizierung . . . . .	177
6.5	Thema 38: Emulieren Sie erweiterbare Enums mit Schnittstellen . . . . .	182
6.6	Thema 39: Ziehen Sie die Annotationen den Namensmustern vor . . . . .	186
6.7	Thema 40: Verwenden Sie konsequent die Annotation Override . . . . .	194
6.8	Thema 41: Definieren Sie Typen mit Markierungsschnittstellen . . . . .	197
<b>7</b>	<b>Lambdas und Streams</b>	<b>201</b>
7.1	Thema 42: Lambdas sind oft besser als anonyme Klassen . .	201
7.2	Thema 43: Denken Sie an Methodenreferenzen als Alternative zu Lambdas . . . . .	205
7.3	Thema 44: Verwenden Sie nach Möglichkeit die funktionalen Schnittstellen aus dem Standard . . . . .	207
7.4	Thema 45: Setzen Sie Streams mit Bedacht ein . . . . .	212
7.5	Thema 46: Bevorzugen Sie in Streams Funktionen ohne Nebeneffekte . . . . .	220
7.6	Thema 47: Verwenden Sie als Rückgabewert eher Collection als Stream . . . . .	226
7.7	Thema 48: Seien Sie vorsichtig, wenn Sie Streams parallelisieren . . . . .	232
<b>8</b>	<b>Methoden</b>	<b>237</b>
8.1	Thema 49: Prüfen Sie Parameter auf Gültigkeit . . . . .	237
8.2	Thema 50: Erstellen Sie bei Bedarf defensive Kopien . . . . .	240
8.3	Thema 51: Entwerfen Sie Methodensignaturen sorgfältig . .	245
8.4	Thema 52: Verwenden Sie Überladung mit Bedacht . . . . .	247
8.5	Thema 53: Verwenden Sie varargs mit Bedacht . . . . .	254
8.6	Thema 54: Geben Sie nicht null, sondern leere Sammlungen oder Arrays zurück . . . . .	256

8.7	Thema 55: Verwenden Sie den Rückgabetypp <code>Optional</code> mit Bedacht .....	258
8.8	Thema 56: Schreiben Sie Doc-Kommentare für alle offengelegten API-Elemente .....	263
<b>9</b>	<b>Allgemeine Programmierung</b>	<b>271</b>
9.1	Thema 57: Minimieren Sie den Gültigkeitsbereich lokaler Variablen .....	271
9.2	Thema 58: Ziehen Sie <code>for-each</code> -Schleifen den traditionellen <code>for</code> -Schleifen vor .....	274
9.3	Thema 59: Machen Sie sich mit den Bibliotheken vertraut und nutzen Sie sie .....	277
9.4	Thema 60: Vermeiden Sie <code>float</code> und <code>double</code> , wenn genaue Antworten benötigt werden .....	280
9.5	Thema 61: Ziehen Sie die elementaren Datentypen den Wrapper-Typen vor .....	283
9.6	Thema 62: Vermeiden Sie Strings, wenn andere Typen besser geeignet sind .....	286
9.7	Thema 63: Denken Sie an die Leistungseinbußen bei der String-Verkettung .....	289
9.8	Thema 64: Referenzieren Sie Objekte über ihre Schnittstellen .....	290
9.9	Thema 65: Ziehen Sie Schnittstellen der Java Reflection vor .....	292
9.10	Thema 66: Vorsicht bei der Arbeit mit nativen Methoden ..	295
9.11	Thema 67: Optimieren Sie mit Bedacht .....	296
9.12	Thema 68: Halten Sie sich an die allgemein anerkannten Namenskonventionen .....	300
<b>10</b>	<b>Ausnahmen</b>	<b>305</b>
10.1	Thema 69: Verwenden Sie Ausnahmen nur für Ausnahmebedingungen .....	305
10.2	Thema 70: Verwenden Sie geprüfte Ausnahmen für behebbare Situationen und Laufzeitausnahmen für Programmierfehler .....	308
10.3	Thema 71: Vermeiden Sie den unnötigen Einsatz von geprüften Ausnahmen .....	310
10.4	Thema 72: Ziehen Sie Standardausnahmen vor .....	312

10.5	Thema 73: Werfen Sie Ausnahmen passend zur Abstraktion . . . . .	314
10.6	Thema 74: Dokumentieren Sie alle Ausnahmen, die jede Methode auslöst . . . . .	317
10.7	Thema 75: Geben Sie in Detailnachrichten Fehlerinformationen an . . . . .	318
10.8	Thema 76: Streben Sie nach Fehleratomizität . . . . .	320
10.9	Thema 77: Ignorieren Sie Ausnahmen nicht . . . . .	322
<b>11</b>	<b>Nebenläufigkeit</b>	<b>325</b>
11.1	Thema 78: Synchronisieren Sie den Zugriff auf gemeinsam genutzte, veränderliche Daten . . . . .	325
11.2	Thema 79: Vermeiden Sie übermäßige Synchronisation . . .	330
11.3	Thema 80: Ziehen Sie Exekutoren, Aufgaben und Streams den Threads vor . . . . .	337
11.4	Thema 81: Ziehen Sie die Nebenläufigkeitsdienste den Methoden <code>wait</code> und <code>notify</code> vor . . . . .	339
11.5	Thema 82: Dokumentieren Sie die Thread-Sicherheit . . . . .	344
11.6	Thema 83: Verwenden Sie die späte Initialisierung mit Bedacht . . . . .	347
11.7	Thema 84: Verlassen Sie sich nicht auf den Thread-Planer .	351
<b>12</b>	<b>Serialisierung</b>	<b>353</b>
12.1	Thema 85: Verwenden Sie statt der Java-Serialisierung besser deren Alternativen . . . . .	353
12.2	Thema 86: Implementieren Sie <code>Serializable</code> mit großer Vorsicht . . . . .	357
12.3	Thema 87: Verwenden Sie möglichst eine eigene serialisierte Form . . . . .	361
12.4	Thema 88: Implementieren Sie <code>readObject</code> defensiv . . . . .	367
12.5	Thema 89: Ziehen Sie zur Instanzenkontrolle die Aufzählungstypen der Methode <code>readResolve</code> vor . . . . .	373
12.6	Thema 90: Verwenden Sie möglichst Serialisierungs-Proxy's anstelle von serialisierten Instanzen . . . . .	377
	<b>Index</b>	<b>381</b>
	<b>Literatur</b>	<b>393</b>