

Inhalt

WWW... 17

1 Grundlagen 21

- 1.1 Java-Überblick 21
 - 1.1.1 Code-Design 23
 - 1.1.2 Java-Applikation 25
- 1.2 Programmstruktur 26
 - 1.2.1 Packages und Namespace 27
 - 1.2.2 Java-Code-Struktur 28
 - 1.2.3 Zugriff auf Klassen und Import 29
 - 1.2.4 Dateiorganisation und Kompilierung 30
 - 1.2.5 Ausführen einer Applikation 31
- 1.3 Primitive Datentypen 32
 - 1.3.1 Regeln zu Operationen mit Zahlen 33
- 1.4 Lexikalische Grundlagen 34
 - 1.4.1 Unicode 34
 - 1.4.2 Whitespace 34
 - 1.4.3 Kommentare 35
 - 1.4.4 Identifier 35
 - 1.4.5 Separator 35
 - 1.4.6 Literale 36
 - 1.4.7 Schlüsselwort 39
 - 1.4.8 Operatoren 39
- 1.5 Konvertierung primitiver Typen 39
 - 1.5.1 Widening Conversion 40
 - 1.5.2 Narrowing Conversion und Casting 42
- 1.6 Initialisierung von Variablen 43
- 1.7 Namenskonventionen 44
 - 1.7.1 Methodename 46
- 1.8 Zusammenfassung 46
- 1.9 Testfragen 47

2 Operatoren 51

- 2.1 Überblick 51
 - 2.1.1 Regel für Operanden-Werteberechnung 54
- 2.2 Arithmetische Operatoren 55
 - 2.2.1 Unäre Operatoren 55
 - 2.2.2 Binäre Operatoren 56

2.3	Referenz-Variable	57
2.4	String-Operator	59
2.5	Vergleichs-Operatoren	60
2.5	Relationale Operatoren	60
2.5	Gleichheits-Operatoren	60
2.6	Typvergleich-Operator instanceof	63
2.7	Logische Operatoren	64
2.8	Bitmanipulationen	65
2.8.1	Duale bzw. hexadezimale Codierung	66
2.8.2	Regeln für Bit-Operationen	67
2.8.3	Invertierung (Bitwise Complement)	68
2.8.4	Bitwise AND, OR, XOR	68
2.8.5	Bitwise-Shift-Operationen	69
2.9	Zuweisungen	71
2.9.1	Compound-Assignment und Seiteneffekte	71
2.9.1	Wirkung der einfachen Zuweisung	72
2.10	Ternärer Operator	73
2.11	Zusammenfassung	74
2.12	Testfragen	76

3 Anweisungen 81

3.1	Überblick über Anweisungen	81
3.2	Grundlegende Anweisungen	83
3.2.1	Leere Anweisung	83
3.2.2	Block (Compound-Statement)	83
3.2.3	Ausdruck als Anweisung (Expression-Statement)	84
3.3	Kontrollanweisungen	84
3.3.1	Selektion	85
3.3.2	Iterationen	89
3.3.3	Sprunganweisungen	93
3.3.4	try-Anweisung	95
3.4	Deklaration	98
3.4.1	Klassen und Interfaces	98
3.4.2	Variable	99
3.4.3	Methode	103
3.5	Aufruf einer Methode	105
3.5.1	Argumente und Wertübergabe	105
3.5.2	Einfacher Methoden-Aufruf	108
3.5.3	Qualifizierender Methoden-Aufruf	108
3.6	Zusammenfassung	111
3.7	Testfragen	112

4	Modellierung und UML 117
4.1	Generalisierung und Spezialisierung 118
4.1.1	Klassifikation: Denken in Hierarchien 118
4.1.2	Dynamische Polymorphie 123
4.2	Klassen-Beziehungen in der UML 124
4.2.1	Klasse 124
4.2.2	Generalisierung und Spezialisierung 126
4.2.3	Klassifikation von Beziehungen 126
4.2.4	Assoziation 127
4.2.5	Aggregation und Komposition 130
4.2.6	Nützliche UML-Erweiterungen 134
4.3	Objekt-Diagramm 137
4.3.1	Objekt-Notation 137
4.3.2	Objekt-Diagramme 137
4.4	Interaktions-Modellierung 138
4.4.1	Sequenz-Diagramme 139
4.5	Zusammenfassung 141
5	Vererbung 143
5.1	Deklaration von Subklassen 143
5.2	Overriding vs. Overloading 146
5.3	Superklasse Object 148
5.4	Vererbung und Modifikatoren 149
5.4.1	abstract 149
5.4.2	final 149
5.4.3	Zugriffs-Modifikatoren (Access-Modifier) 150
5.4.4	Unverträglichkeiten von Modifikatoren 152
5.5	Konstruktoren 153
5.5.1	Default-Konstruktor 153
5.5.2	Deklaration und Initialisierungs-Regeln 153
5.5.3	Signatur-Regel für Konstruktoren 156
5.5.4	Zusammenarbeit von Konstruktoren 156
5.6	Pros und Kons der Initialisierung 157
5.6.1	Design-Pros 157
5.6.2	Design-Kons 158
5.7	Initialisierer 160
5.7.1	Instanz-Initialisierer 160
5.7.2	Statischer Initialisierer 161
5.8	Overriding vs. Shadowing 163
5.8.1	Statischer vs. virtueller Aufruf von Methoden 163
5.8.2	Shadowing von Feldern und super 165
5.8.3	Aufruf überschriebener Methoden mittels super 166

5.9	Speicherverwaltung	167
5.9.1	Garbage Collection	167
5.9.2	GC und Finalization	168
5.9.3	Alternative zu finalize	170
5.9.4	Einsatz von finalize	172
5.10	Kapselung (Encapsulation)	174
5.11	Zusammenfassung	176
5.12	Testfragen	177
<hr/>		
6	Interfaces und Pattern	183
6.1	Interface: Definition und Regeln	184
6.1.1	Beispiele zu den Interface-Regeln	185
6.2	Referenz-Konvertierung und -Casting	188
6.2.1	Regeln zur Referenz-Konvertierung	188
6.2.2	Regeln zum Referenz-Casting	189
6.2.3	Exemplarische Konvertierungs- und Cast-Beispiele	189
6.3	API: Interface vs. Klasse	191
6.3.1	Beispiel JDK: Löschen bzw. Ändern im API	191
6.3.2	Erweiterungen im API	191
6.4	Interface vs. Vererbung	192
6.4.1	Service-Beziehung auf Basis von Vererbung	192
6.4.2	Service-Beziehung auf Basis von Interfaces	193
6.4.3	Nachteile der Vererbung	193
6.5	Design-Pattern: generelle Entwurfsmuster	194
6.6	Interface-Pattern	195
6.6.1	Filter-Template, Kollektion	196
6.6.2	Interface – eine Firewall?	198
6.7	Delegation-Pattern	199
6.7.1	Probleme bei der Vererbung: zwei Beispiele	199
6.7.2	Delegation vs. Vererbung	201
6.8	Vererbung, Interfaces und Delegation	203
6.9	Immutable	204
6.10	Marker-Interface	205
6.10.1	Sticker-Interface: Erweiterung des Marker-Interfaces	207
6.10.2	Marker-Interface Cloneable	209
6.11	Factory-Pattern	211
6.11.1	Ein »Teile-Fabrik«-Beispiel	212

- 6.12 Konzeptionelle Schwächen von Interfaces 217**
 - 6.12.1 Constraints (Zusicherungen) 217
 - 6.12.2 Interfaces bieten keinen Zugriffsschutz 217
 - 6.12.3 Ein Template-Dilemma 218
- 6.13 Zusammenfassung 218**
- 6.14 Testfragen 219**

7 Ausnahmen 223

- 7.1 Konzeption 224**
- 7.2 Ausnahme-Mechanismus 225**
 - 7.2.1 Ausnahme-Auslösung (exception throwing) 225
 - 7.2.2 Ausnahme-Behandlung (exception handling) 226
- 7.3 Details zur Ausnahme-Behandlung 229**
 - 7.3.1 catch-Reihenfolge 229
 - 7.3.2 Geschachtelte Ausnahmen 230
 - 7.3.3 finally und unbehandelte Ausnahmen 231
 - 7.3.4 Rethrowing in catch 232
- 7.4 Ausnahme-Kategorien in Java 233**
 - 7.4.1 Hierarchie-Konzept 233
 - 7.4.2 Checked vs. unchecked Exceptions 234
 - 7.4.3 Basisklasse Throwable 235
 - 7.4.4 Ausnahmen vom Typ Error 235
 - 7.4.5 Ausnahmen vom Typ Exception 236
 - 7.4.6 Ausnahmen vom Typ RuntimeException 236
 - 7.4.7 Checked Exceptions 237
- 7.5 Overriding von Ausnahmen 237**
- 7.6 Deklaration neuer Ausnahmen 239**
- 7.7 Einsatz von Ausnahmen 241**
 - 7.7.1 Missbrauch von Ausnahmen 241
 - 7.7.2 Ausnahmen zur Einhaltung von Kontrakten 241
 - 7.7.3 Kommunikation per Ausnahme 243
- 7.8 Ausnahmen-Behandlung 245**
 - 7.8.1 Beispiel: Messwert-Import 245
 - 7.8.2 Design neuer Ausnahmen 248
 - 7.8.3 Ein Idiom zur Behandlungs-Strategie 252
 - 7.8.4 Behandlungs-Strategie beim Messwert-Import 253
- 7.9 Zusammenfassung 257**
- 7.10 Testfragen 258**

8	Innere Klassen 263
8.1	Arten von inneren Klassen 263
8.2	Statische innere Klassen und Interfaces 265
8.2.1	Design-Beispiel: 2D-Klassen 268
8.3	Nicht statische innere Klassen 268
8.4	Member-Klassen 269
8.4.1	Anlage und Zugriff auf Instanzen einer Member-Klasse 270
8.4.2	Member-Klassen und Vererbung 273
8.4.3	Member-Klasse vs. Instanz-Variable 275
8.5	Lokale Klassen 277
8.5.1	Gültigkeit vs. Lebensdauer 279
8.6	Anonyme Klassen 280
8.6.1	Zugriff auf interne Methoden anonymer Klassen 284
8.6.2	Probleme mit anonymen Klassen 285
8.7	Firewall-Idiom: Quasi-Objekte von Interfaces 286
8.8	Generisches Verhalten 289
8.8.1	Pluggable-Behavior-Pattern 289
8.9	Zusammenfassung 292
8.10	Testfragen 293

9	Threads 297
9.1	Grundlegende Begriffe 297
9.1.1	Multi-Tasking, Multi-Threading 297
9.1.2	Scheduling, Priorität und Preemption 299
9.1.3	Synchronisation 300
9.2	Thread-Start 301
9.2.1	Methoden run() und start() 302
9.3	Thread-Zustände 304
9.3.1	Zustand: aktiv bzw. tot 304
9.3.2	Zustand: schlafend 305
9.3.3	Zustand: blockiert vs. nicht blockiert bei I/O 305
9.3.4	Zustand: Warten auf Lock 305
9.3.5	Monitor 305
9.3.6	Methoden wait(), notify() bzw. notifyAll() 306
9.3.7	Zustand: wartend 306
9.3.8	Unterbrechen der Zustände 307
9.4	Thread-Methoden 308
9.4.1	Konstruktoren 308
9.4.2	Statische Methoden 308
9.4.3	Prioritäten 309
9.4.4	Instanz-Methoden 309
9.4.5	Diverse Methoden im Beispiel 310

9.5	Race-Condition 312
9.5.1	Atomare vs. nicht atomare Operationen 312
9.5.2	Reentrant, Race-Condition und thread-sicher 313
9.6	Synchronisation und Deadlock 314
9.6.1	Methoden- und Block-Synchronisation 315
9.6.2	Voll- bzw. teilsynchronisierte Objekte 317
9.6.3	Deadlock durch Synchronisation 317
9.7	Vermeidungsstrategien zu Deadlocks 320
9.7.1	Ressourcen-Anordnung bei Locks 320
9.7.2	Deadlocks in Objekt-Kompositionen 321
9.8	Guarded-Method: wait() und notify() 322
9.8.1	Auswirkung der wait()-Regel 322
9.8.2	Wait-Regel bzw. -Idiom 323
9.8.3	Guarded-Method und Guarded-Action-Idiom 324
9.9	Weitere thread-sichere Maßnahmen 328
9.9.1	Threads und immutable Objekte 328
9.9.2	Zustandslose Methoden 329
9.9.3	Thread-sichere Dekoration/Wrapper 329
9.10	Thread-Mechanismen 331
9.10.1	Passive Objekte 331
9.10.2	Aktive Objekte 331
9.10.3	Aktive Client- und Server-Objekte 332
9.10.4	Asynchroner Service 333
9.11	Client-aktivierte asynchrone Methoden 334
9.11.1	One-Shot-Objekt 334
9.11.2	Asynchrone Methode ohne Ergebnis 335
9.12	Server-Aktivierung 336
9.12.1	Autonomes Objekt 336
9.12.2	Asynchrone Methode mit Ergebnis 337
9.12.3	join(): Warten ohne Polling, sofern notwendig 337
9.12.4	Callback-Technik 340
9.13	Thread-Unterbrechung 342
9.13.1	interrupt() und InterruptedException 342
9.13.2	interrupt()-Probleme und Gegenmaßnahmen 345
9.13.3	interrupt()-Konzept 348
9.14	Unbehandelte Ausnahmen in Threads 348
9.14.1	ThreadGroup: uncaughtException() 350
9.15	Zusammenfassung 351
9.16	Testfragen 352

10	Package java.lang 357
10.1	Object 357
10.1.1	Overriding equals() 357
10.1.2	getClass() vs. instanceof 359
10.1.3	Semantik von Gleichheit 361
10.1.4	Beziehung zwischen equals() und hashCode() 361
10.1.5	Berechnung des Hashcodes 363
10.2	Class 364
10.3	Wrapper-Klassen 366
10.4	String und StringBuffer 367
10.4.1	Compiler-Optimierung 367
10.4.2	Notwendigkeit der manuellen Optimierung 367
10.4.3	Größe eines StringBuffer-Objekts 368
10.5	Math und StrictMath 369
10.5.1	Singleton-Pattern und Lazy Initialization 370
10.5.2	Singleton-Komposition 371
10.5.3	Singleton und Double-Check Locking 371
10.5.4	Probleme mit komplexen Zahlen 372
10.6	System 373
10.6.1	Byte-orientierte Konsol-Ausgabe 373
10.6.2	Zeichen-Codierung und Unicode 374
10.6.3	Zeichen- vs. Byte-Streams 375
10.6.4	Default-Codierung und System.out 376
10.6.5	Byte-orientierte Eingabe mit System.in 378
10.6.6	System-Properties 380
10.6.7	Methode: load() bzw. loadLibrary() 381
10.6.8	Methode: identityHashCode() 381
10.6.9	Methode: currentTimeMillis() 381
10.6.10	Methode: arraycopy() 382
10.6.11	Methode: exit() 382
10.7	Process 382
10.8	Runtime 383
10.8.1	Methode: exec() 383
10.9	Zusammenfassung 386
10.10	Testfragen 387

11	Package java.io 391
11.1	Überblick 391
11.2	File 393
11.3	FileDescriptor 397
11.4	Interfaces DataInput und DataOutput 398
11.5	RandomAccessFile 400
11.6	Stream-Konzept 403
11.7	Decorator-Pattern 409
11.8	Streams im Einsatz 411
11.9	Digitale Signatur für Dokumente (Byte-Streams) 417
11.9.1	Package java.security 418
11.10	Pipe-Streams 421
11.11	Zusammenfassung 429
11.12	Testfragen 430
12	Serialisierung 435
12.1	Serialisierung: Kommunikation auf Basis von Objekten 435
12.2	Grundlagen der Serialisierung 436
12.2.1	Standard-Serialisierung 437
12.3	Übertragungs-Protokolle 438
12.3.1	Protokoll zu primitiven Typen 438
12.3.2	Protokolle zur Objekt-Serialisierung 441
12.3.3	Protokoll zu Externalizable 442
12.3.4	Protokoll zu Serializable 445
12.4	Einfache Anpassungen von Serializable 449
12.4.1	serialPersistentFields: Ersatz für transient 449
12.4.2	Externalizable: Kapselung unmöglich 450
12.4.3	Klasseninterne Anpassung der Default-Serialisierung 450
12.4.4	Broker-Pattern: Stream-Ersatzobjekte 453
12.5	Klassen-Evolution 456
12.5.1	Stream-Kompatibilität 457
12.6	Anpassungen der Object-Streams 459
12.6.1	annotateClass() und resolveClass() für Klassendaten 459
12.6.2	replaceObject() und resolveObject() 461
12.6.3	writeObjectOverride() und readObjectOverride() 465
12.7	Zusammenfassung 468
12.8	Testfragen 469

13	Reflexion 473
13.1	Klasse, Interface und Reflexion 473
13.2	Übersicht über Reflexion 474
13.2.1	Package java.lang.reflect 474
13.2.2	Object und Class 475
13.2.3	Constructor, Field und Method 476
13.2.4	Modifier 477
13.2.5	Array 477
13.2.6	Proxy 478
13.3	Inspektion von Klassen und Interfaces 479
13.3.1	Class-Objekt 479
13.3.2	Methoden get<X> vs. getDeclared<X> 480
13.3.3	Package, Superklassen und Interfaces 481
13.3.4	Konstruktoren und Modifikatoren 482
13.3.5	Felder 484
13.3.6	Methoden 484
13.3.7	Array 485
13.4	Manipulation von Klassen und Arrays 486
13.4.1	Anlage von Objekten 486
13.4.2	Lesen und Schreiben von Feldern 486
13.4.3	Ausführung von Methoden 488
13.4.4	InvocationTargetException 489
13.4.5	Anlage von Arrays 490
13.4.6	Lesen und Schreiben von Array-Komponenten 491
13.4.7	Unterstützende Array-Methoden 492
13.5	Introspection und Kommandos 493
13.5.1	Inspektion mit Hilfe der Klasse Introspector 494
13.5.2	Kommandos 495
13.6	Command-Pattern 496
13.6.1	Realisierung mittels Interfaces oder Reflexion 496
13.7	Dynamisches Proxy-Pattern 500
13.7.1	Allgemeine Eigenschaften 500
13.7.2	Dynamische Proxy-Variante 500
13.7.3	Dynamische Proxy-Implementation 501
13.7.4	Beispiel Broker- bzw. Façade-Proxy 502
13.8	Zusammenfassung 511

14	Collection-Framework 513
14.1	Aufbau 513
14.2	Interface-Hierarchie 514
14.2.1	Unterstützende Interfaces 516
14.2.2	Bedeutung der Interfaces 516
14.2.3	Konventionen, Kontrakte und Constraints 517
14.2.4	Kontrakte 518
14.3	Kollektions-Klassen 520
14.3.1	Auswahl der Kollektions-Klasse 521
14.3.2	Hash-Kollektionen und Tuning 522
14.3.3	HashMap 525
14.3.4	Listen 526
14.3.5	Comparable und Comparator 530
14.3.6	Bäume (Trees) 531
14.3.7	Bereichs-Operationen: Bäume und Listen 533
14.4	Iterator-Pattern 535
14.4.1	Iteratoren zu Maps 537
14.5	Fundamentale vs. Template-Methoden 538
14.5.1	Das Template-Problem 538
14.5.2	Sortieren vs. Shuffling 539
14.5.3	Generische Listen-Operationen 541
14.6	Dekorieren von Kollektionen 543
14.6.1	Synchronisierte Wrapper 543
14.6.2	Immutable Wrapper 545
14.7	Zusammenfassung 545
14.8	Testfragen 547

15	Java Foundation Classes 551
15.1	Vorbemerkungen 551
15.2	Begriffe 553
15.3	Top-Level-Container 554
15.3.1	JRootPane 555
15.3.2	JLayeredPane 559
15.3.3	JWindow 560
15.3.4	JFrame vs. Frame 562
15.3.5	JDialog und JOptionPane 563
15.4	Layout-Manager 567
15.4.1	Überblick 569
15.4.2	BorderLayout und FlowLayout 570
15.4.3	CardLayout 572
15.4.4	GridLayout 573
15.4.5	GridBagLayout 574

- 15.4.6 **BoxLayout** 578
- 15.5 Ereignisbehandlung 579**
 - 15.5.1 Observer-Pattern 579
 - 15.5.2 Ereignis-Delegations-Modell 581
 - 15.5.3 Ereignisverarbeitung 585
 - 15.5.4 Fokus-Management 588
 - 15.5.5 Ereignisverteilung (Event-Dispatching) 591
 - 15.5.6 Ereignisbearbeitung in abgeleiteten Komponenten 592
- 15.6 MFC-Architektur 594**
 - 15.6.1 Model-Delegate-Architektur in Swing 594
 - 15.6.2 BoundedRangeModel für Bars und Sliders 596
- 15.7 Multi-Threading in Swing 600**
 - 15.7.1 Erlaubte Zugriffe von Threads auf Swing 600
 - 15.7.2 Notwendigkeit swing-externer Threads 601
 - 15.7.3 Kommunikation mit Swing von externen Threads 601
- 15.8 Applet 605**
 - 15.8.1 Eigenschaften 605
 - 15.8.2 Erstellen eines (J)Applets 606
 - 15.8.3 Wichtige Getter-Methoden 607
- 15.9 Zusammenfassung 608**
- 15.10 Chamäleon-Architektur 609**

Index 611