

Der Inhalt (im Überblick)

Inhaltsverzeichnis

| | | |
|----|--|-----|
| | Einführung | xix |
| 1 | Die Oberfläche durchbrechen: <i>Tauchen Sie ein: eine Kostprobe</i> | 1 |
| 2 | Die Reise nach Objectville: <i>Klassen und Objekte</i> | 27 |
| 3 | Verstehen Sie Ihre Variablen: <i>Elementare Datentypen und Referenzen</i> | 49 |
| 4 | Wie sich Objekte verhalten: <i>Methoden benutzen Instanzvariablen</i> | 71 |
| 5 | Methoden mit Superkräften: <i>Ein Programm schreiben</i> | 95 |
| 6 | Die Java-Bibliothek verwenden: <i>Die Java-API kennenlernen</i> | 125 |
| 7 | Besser leben in Objectville: <i>Vererbung und Polymorphie</i> | 167 |
| 8 | Ernsthafte Polymorphie: <i>Interfaces und abstrakte Klassen</i> | 199 |
| 9 | Leben und Sterben eines Objekts: <i>Konstruktoren und Garbage Collection</i> | 237 |
| 10 | Zahlen, bitte!: <i>Zahlen und Statisches</i> | 275 |
| 11 | Datenstrukturen: <i>Collections mit Generics</i> | 309 |
| 12 | Was – nicht wie!: <i>Lambdas und Streams</i> | 369 |
| 13 | Riskantes Verhalten: <i>Exception-Handling</i> | 421 |
| 14 | Innen hui, außen GUI: <i>Grafische Benutzeroberflächen</i> | 461 |
| 15 | Mehr Schwung mit Swing: <i>Swing im Einsatz</i> | 509 |
| 16 | Objekte (und Text) speichern: <i>Serialisierung und Datei-I/O</i> | 539 |
| 17 | Eine Verbindung herstellen: <i>Netzwerkprogrammierung und Threads</i> | 587 |
| 18 | Nebenläufigkeitsprobleme behandeln: <i>Konkurrenzprobleme und immutable Daten</i> | 639 |
| A | Anhang A: <i>Die finale Codeküche</i> | 673 |
| B | Anhang B: <i>Die (mehr als) zehn wichtigsten Themen, die es nicht ins Buch geschafft haben ...</i> | 683 |
| | Index | 701 |

Der Inhalt (jetzt ausführlich)

i Einführung

| | |
|---|--------|
| Für wen ist dieses Buch? | xx |
| Wir wissen, was Sie denken. | xxi |
| Und wir wissen, was Ihr <i>Gehirn</i> gerade denkt. | xxi |
| Das haben WIR getan | xxiv |
| Was Sie für dieses Buch brauchen | xxvi |
| Die Fachgutachter der dritten englischen Auflage | xxviii |
| Die Fachgutachter der zweiten englischen Auflage | xxx |

vii

1 **Die Oberfläche durchbrechen**

Java bringt Sie an neue Orte.

| | |
|--|----|
| Wie Java funktioniert | 2 |
| Was Sie in Java tun werden | 3 |
| Eine sehr kurze Geschichte von Java | 4 |
| Codestruktur in Java | 7 |
| Eine Klasse mit einer main()-Methode schreiben | 9 |
| Einfache boolesche Tests | 13 |
| Bedingte Verzweigungen | 15 |
| Eine ernsthafte Geschäftsanwendung schreiben | 16 |
| Der Phras-O-Mat | 19 |
| Übungen | 20 |

2 **Die Reise nach Objectville**

Ich dachte, hier gibt's Objekte.

| | |
|---------------------------------------|----|
| Stuhlkriege | 28 |
| Ihr erstes Objekt erstellen | 36 |
| Die Film-Objekte erstellen und testen | 37 |
| Schnell! Nichts wie raus aus main()! | 38 |
| Das Ratespiel ausführen | 40 |
| Übungen | 42 |

3 Verstehen Sie Ihre Variablen

Variablen gibt es in zwei Geschmacksrichtungen: elementare Typen und Referenztypen.

| | |
|--|----|
| Eine Variable deklarieren | 50 |
| »Einen doppelten Espresso bitte, ach nein, doch lieber einen int.« | 51 |
| Finger weg von Schlüsselwörtern! | 53 |
| Ein Dog-Objekt kontrollieren | 54 |
| Eine Objektreferenz ist einfach ein anderer Variablenwert. | 55 |
| Das Leben auf dem Garbage Collectible Heap | 57 |
| Ein Array ist wie ein Schrank voller Tassen ... ähm Becher | 59 |
| Ein Dog-Beispiel | 62 |
| Übungen | 63 |

4 Wie sich Objekte verhalten

Zustand beeinflusst Verhalten, Verhalten beeinflusst Zustände.

| | |
|--|----|
| Erinnern Sie sich: Eine Klasse beschreibt, was ein Objekt weiß und was es tut. | 72 |
| Die Größe beeinflusst das Bellen | 73 |
| Sie können einer Methode Informationen schicken | 74 |
| Sie können von Methoden etwas zurückbekommen | 75 |
| Sie können einer Methode mehr als eine Sache übergeben | 76 |
| Parameter und Rückgabetypen | 79 |
| Kapselung | 80 |
| Wie verhalten sich Objekte in einem Array? | 83 |
| Instanzvariablen deklarieren und initialisieren | 84 |
| Variablen vergleichen (elementare und Referenztypen) | 86 |
| Übungen | 88 |

5 Methoden mit Superkräften

Jetzt wollen wir unsere Methoden mal etwas aufmöbeln.

| | |
|--|-----|
| Ein Spiel im Schiffe-versenken-Style: »Startups versenken« | 96 |
| Eine Klasse entwickeln | 99 |
| Die Methodenimplementierungen schreiben | 101 |
| Den Testcode für die SimpleStartup-Klasse schreiben | 102 |
| Die checkYourself()-Methode | 104 |
| Vorcode für die SimpleStartupGame-Klasse | 108 |
| Die main()-Methode des Spiels | 110 |
| Dann spielen wir mal ... | 113 |
| Mehr über for-Schleifen | 114 |
| Die verbesserte for-Schleife | 116 |
| Elementare Typen casten | 117 |
| Übungen | 118 |

6 Die Java-Bibliothek verwenden

Java wird mit Hunderten vorgefertigter Klassen ausgeliefert.

| | |
|---|-----|
| Das vorige Kapitel endete mit einem Cliffhanger – nämlich einem Bug | 126 |
| Wachen Sie endlich auf und sehen Sie der Java-API ins Auge | 132 |
| Einige Dinge, die Sie mit ArrayList tun können | 133 |
| ArrayList mit einem regulären Array vergleichen | 137 |
| Bauen wir das RICHTIGE Spiel: »Startups versenken« | 140 |
| Vorcode für die echte StartupBust-Klasse | 144 |
| Die finale Version der Startup-Klasse | 150 |
| Boolesche Ausdrücke mit Superkräften | 151 |
| Die Bibliothek (Java-API) verwenden | 154 |
| Übungen | 163 |

7

Besser leben in Objectville

Planen Sie Ihre Programme mit der Zukunft im Blick.

| | |
|--|-----|
| Stuhlkriege neu aufgelegt ... | 168 |
| Vererbung verstehen | 170 |
| Entwerfen wir einen Vererbungsbaum für ein Tiersimulationsprogramm | 172 |
| Weitere Vererbungsmöglichkeiten finden | 175 |
| IST EIN und HAT EIN verwenden | 179 |
| Woher wissen Sie, dass Ihre Vererbungshierarchie korrekt ist? | 181 |
| Vererbung kann man gut gebrauchen, man kann sie aber auch missbrauchen! | 183 |
| Den Vertrag einhalten: Regeln für das Überschreiben | 192 |
| Eine Methode überladen | 193 |
| Übungen | 194 |

8

Ernsthafte Polymorphie

Vererbung ist nur der Anfang.

| | |
|--|-----|
| Haben wir etwas vergessen, als wir das hier entworfen haben? | 200 |
| Der Compiler verhindert die Instanziierung von abstrakten Klassen | 203 |
| Abstrakt vs. konkret | 204 |
| Abstrakte Methoden MÜSSEN implementiert werden | 206 |
| Polymorphie in Aktion | 208 |
| Was ist mit Nicht-Animals? Warum machen wir die Klasse nicht so allgemein, dass sie mit allem umgehen kann? | 210 |
| Wenn ein Dog sich nicht wie ein Dog verhält | 214 |
| Erforschen wir ein paar Entwurfsoptionen | 221 |
| Das Pet-Interface erstellen und implementieren | 227 |
| Die Superklassenversion einer Methode aufrufen | 230 |
| Übungen | 232 |

9 **Leben und Sterben eines Objekts**

Objekte werden geboren und Objekte sterben.

| | |
|---|-----|
| Der Stack und der Heap: wo das Leben spielt | 238 |
| Methoden werden gestapelt | 239 |
| Was ist mit lokalen Variablen, die Objekte sind? | 240 |
| Das Wunder der Objekterstellung | 242 |
| Ein Duck-Objekt konstruieren | 244 |
| Erstellt der Compiler nicht immer einen argumentlosen Konstruktor für Sie? | 248 |
| Kurzer Rückblick. Vier Dinge, die Sie sich zu Konstruktoren merken sollten! | 251 |
| Die Rolle der Superklassenkonstruktoren im Leben eines Objekts | 253 |
| Kann ein Kind vor seinen Eltern existieren? | 256 |
| Was ist mit Referenzvariablen? | 262 |
| Mir gefällt nicht, worauf das hinausläuft. | 263 |
| Übungen | 268 |

10 **Zahlen, bitte!**

| | |
|---|-----|
| MATH-Methoden: Näher werden Sie einer globalen Methode nie wieder kommen | 276 |
| Der Unterschied zwischen regulären (nicht statischen) und statischen Methoden | 277 |
| Eine statische Variable initialisieren | 283 |
| Math-Methoden | 288 |
| Elementartypen verpacken | 290 |
| Autoboxing funktioniert fast überall | 292 |
| Und umgekehrt ... eine elementare Zahl in einen String verwandeln | 295 |
| Zahlenformatierung | 296 |
| Der Format-Spezifizierer | 300 |
| Übungen | 306 |

11

Datenstrukturen

Sortieren ist in Java ein Klacks.

| | |
|---|-----|
| Die java.util-API, List und Collections entdecken | 314 |
| Generics sorgen für mehr Typsicherheit | 320 |
| Erneut ein Blick auf die sort()-Methode | 327 |
| Die neue, verbesserte Song-Klasse | 330 |
| Mit Comparators sortieren | 336 |
| Die Jukebox mit Lambdas aktualisieren | 342 |
| Ein HashSet anstelle einer ArrayList verwenden | 347 |
| Was Sie über TreeSet wissen MÜSSEN ... | 353 |
| Nachdem wir Lists und Sets gesehen haben, verwenden wir jetzt eine Map | 355 |
| Endlich wieder zurück zu Generics | 358 |
| Lösung zur Übung | 364 |

12

Lambdas und Streams: Was – nicht wie!

Was wäre, wenn ... Sie dem Computer nicht mitteilen müssten,
WIE er etwas tun soll?

| | |
|---|-----|
| Sagen Sie dem Computer, WAS Sie wollen | 370 |
| Wenn for-Schleifen schief laufen | 372 |
| Einführung in die Streams-API | 375 |
| Ergebnisse von einem Stream erhalten | 378 |
| Richtlinien für die Arbeit mit Streams | 384 |
| Hallo Lambda, mein (nicht ganz so) alter Freund | 388 |
| Erkennen Sie funktionale Interfaces | 396 |
| Lous Herausforderung Nr. 1: Finde alle »Rock«-Songs | 400 |
| Lous Herausforderung Nr. 2: Alle Genres auflisten | 404 |
| Übungen | 415 |

13 Riskantes Verhalten

Dinge passieren.

| | |
|---|-----|
| Bauen wir eine Musikmaschine | 422 |
| Zuerst brauchen wir einen Sequencer | 424 |
| Eine Exception ist ein Objekt ... des Typs Exception | 428 |
| Flusskontrolle in try/catch-Blöcken | 432 |
| Eine Methode kann mehr als eine Exception werfen | 435 |
| Mehrfache catch-Blöcke müssen von klein nach groß geordnet werden | 438 |
| Ausweichen (durch Deklaration) verzögert nur das Unausweichliche | 442 |
| Codeküche | 445 |
| Version 1: Ihre allererste Soundplayer-App | 448 |
| Version 2: Kommandozeilenargumente für das Experimentieren mit Sounds | 452 |
| Übungen | 454 |

14 Innen hui, außen GUI

Sehen Sie den Tatsachen ins Auge: Früher oder später werden Sie GUIs erstellen müssen.

| | |
|---|-----|
| Alles beginnt mit einem Fenster | 462 |
| Wie man an ein Benutzer-Event kommt | 465 |
| Listener, Quellen und Events | 469 |
| Machen Sie sich Ihr eigenes Grafik-Widget | 472 |
| Spaß mit paintComponent() | 473 |
| GUI-Layouts: mehrere Widgets in einen Frame packen | 478 |
| Die Rettung: Innere Klassen! | 484 |
| Rettung durch Lambdas! | 490 |
| Innere Klassen für Animationen einsetzen | 492 |
| Eine einfachere Möglichkeit, Messages und Events zu erstellen | 498 |
| Übungen | 502 |

15 Mehr Schwung mit Swing

Swing ist einfach.

| | |
|--|-----|
| Swing-Komponenten | 510 |
| Layoutmanager | 511 |
| Die drei wichtigsten Layoutmanager: Border, Flow und Box | 513 |
| Es gibt keine Dummen Fragen | 522 |
| Spielen mit Swing-Komponenten | 523 |
| Codeküche | 526 |
| Die BeatBox | 529 |
| Übungen | 534 |

16 Objekte (und Text) speichern

Objekte lassen sich flach drücken und wieder aufblasen.

| | |
|--|-----|
| Ein serialisiertes Objekt in eine Datei schreiben | 542 |
| Soll eine Klasse serialisierbar sein, implementieren Sie | 547 |
| Deserialisierung: ein Objekt wiederherstellen | 551 |
| Version-ID: Ein großes Serialisierungsproblem | 556 |
| Einen String in ein Textdatei schreiben | 559 |
| Aus einer Textdatei lesen | 566 |
| Quiz Card Player (Code grob skizziert) | 567 |
| Path, Paths und Files (mit Verzeichnissen spielen) | 573 |
| Endlich, ein genauer Blick auf finally | 574 |
| Ein BeatBox-Pattern speichern | 579 |
| Übungen | 580 |

17 Eine Verbindung herstellen

Verbindung zur Außenwelt aufnehmen.

| | |
|---|-----|
| Verbinden, senden, empfangen | 590 |
| Der DailyAdviceClient (»Tipp des Tages«) | 598 |
| Ein einfaches Serverprogramm schreiben | 601 |
| Java hat mehrere Threads, aber nur eine Thread-Klasse | 610 |
| Die drei Zustände eines neuen Threads | 616 |
| Einen Thread schlafen schicken | 622 |
| Zwei (oder mehr!) Threads erzeugen und starten | 626 |
| Feierabend am Thread-Pool | 629 |
| Der neue und verbesserte SimpleChatClient | 632 |
| Übungen | 634 |

18 Nebenläufigkeitsprobleme behandeln

Es ist schwer, mehrere Dinge gleichzeitig zu tun.

| | |
|---|-----|
| Das Ryan-und-Monica-Problem, in Code ausgedrückt | 642 |
| Das Schloss eines Objekts verwenden | 647 |
| Das gefürchtete »Problem der verlorenen Aktualisierung« | 650 |
| Die increment()-Methode atomar machen. Synchronisieren Sie sie! | 652 |
| Blockade, eine tödliche Seite der Synchronisierung | 654 |
| »Vergleichen und tauschen« mit atomaren Variablen | 656 |
| Immutable Objekte verwenden | 659 |
| Mehr Probleme mit geteilten Daten | 662 |
| Verwenden Sie eine threadsichere Datenstruktur | 664 |
| Übungen | 668 |

A

Anhang A

Die finale Codeküche.

Das endgültige BeatBox-Clientprogramm 674

Das endgültige BeatBox-Serverprogramm 681

B

Anhang B

Die (mehr als) zehn wichtigsten Themen, die es nicht ins Buch geschafft haben ... Noch können wir Sie nicht ganz gehen lassen. Ein paar Sachen haben wir noch, aber dann sind Sie fertig. Diesmal meinen wir es ernst.

| | |
|--|-----|
| #11 JShell (Java REPL) | 684 |
| #10 Packages | 685 |
| #9 Immutabilität in Strings und Wrappern | 688 |
| #8 Zugriffsebenen und Zugriffsmodifier (wer sieht was) | 689 |
| #7 Varargs | 691 |
| #6 Annotationen | 692 |
| #5 Lambdas und Maps | 693 |
| #4 Parallele Streams | 695 |
| #3 Enumerations (enumerierte Typen/Enums) | 696 |
| #2 Lokale Typinferenz für Variablen (var) | 698 |
| #1 Records | 699 |

i Index

701