

Bernhard Steppan

Einstieg in Java 6

Auf einen Blick

Vorwort	19
Teil 1 Basiswissen	
1 Digitale Informationsverarbeitung	27
2 Programmiersprachen	43
3 Objektorientierte Programmierung	63
Teil 2 Java im Detail	
4 Sprache	89
5 Entwicklungsprozesse	147
6 Plattform	175
7 Gesetzmäßigkeiten	195
8 Klassenbibliotheken	227
9 Algorithmen	281
Teil 3 Größere Java-Projekte	
10 Konsolenprogramme	297
11 Erste Schritte mit grafischen Oberflächen	313
12 Einfache Oberflächen mit AWT und Swing	347
13 Komplexe Oberflächen mit Swing.....	377
14 Weboberflächen mit Servlets	403
15 Datenbankprogrammierung	429
16 Datenbankanwendungen	449
17 Dynamische Websites	469
Teil 4 Lösungen	
18 Lösungen zu Teil I	485
19 Lösungen zu Teil II	493
20 Lösungen zu Teil III	509
Teil 5 Anhang	
21 Werkzeuge	525
22 Computerhardware	561
23 Glossar	569
24 Literatur	579

Inhalt

Vorwort

19

Teil 1 Basiswissen

1 Digitale Informationsverarbeitung

27

1.1	Einleitung	27
1.2	Zahlensysteme	27
1.2.1	Dezimalsystem	27
1.2.2	Binärsystem	28
1.2.3	Hexadezimalsystem	30
1.3	Informationseinheiten	32
1.3.1	Bit	32
1.3.2	Byte	33
1.3.3	Wort	33
1.4	Kodierung von Zeichen	33
1.4.1	ASCII-Code	33
1.4.2	ANSI-Code	35
1.4.3	Unicode	35
1.5	Kodierung logischer Informationen	36
1.5.1	Und-Funktion	37
1.5.2	Oder-Funktion	38
1.5.3	Nicht-Funktion	39
1.6	Zusammenfassung	39
1.7	Aufgaben	40
1.7.1	Zahlensysteme	40
1.7.2	Informationseinheiten	40
1.7.3	Zeichenkodierung	40
1.7.4	Kodierung logischer Informationen	40

2 Programmiersprachen

43

2.1	Einleitung	43
2.1.1	Verständigungsschwierigkeiten	43
2.1.2	Definition	43
2.1.3	Klassifizierung	44

2.1.4	Geschichte	45
2.2	Programmiersprachen der ersten Generation	46
2.2.1	Programmaufbau	46
2.2.2	Portabilität	47
2.2.3	Ausführungsgeschwindigkeit	48
2.2.4	Einsatzbereich	48
2.3	Programmiersprachen der zweiten Generation	48
2.3.1	Programmaufbau	48
2.3.2	Portabilität	50
2.3.3	Ausführungsgeschwindigkeit	50
2.3.4	Einsatzbereich	50
2.4	Programmiersprachen der dritten Generation	51
2.4.1	Programmaufbau	51
2.4.2	Portabilität	52
2.4.3	Ausführungsgeschwindigkeit	53
2.4.4	Einsatzbereich	53
2.5	Programmiersprachen der vierten Generation	53
2.5.1	Programmaufbau	53
2.5.2	Portabilität	54
2.5.3	Ausführungsgeschwindigkeit	54
2.5.4	Einsatzbereich	54
2.6	Programmiersprachen der fünften Generation	55
2.6.1	Programmaufbau	55
2.6.2	Portabilität	56
2.6.3	Ausführungsgeschwindigkeit	56
2.6.4	Einsatzbereich	56
2.7	Programmiersprachen der sechsten Generation	56
2.7.1	Programmaufbau	56
2.7.2	Portabilität	58
2.7.3	Ausführungsgeschwindigkeit	58
2.7.4	Einsatzbereich	58
2.8	Zusammenfassung	58
2.9	Aufgaben	59
2.9.1	Programmiersprachen der ersten Generation	59
2.9.2	Programmiersprachen der zweiten Generation	59
2.9.3	Programmiersprachen der dritten Generation	59

3 Objektorientierte Programmierung 63

3.1	Einleitung	63
3.1.1	Grundbegriffe	63
3.1.2	Prinzipien	64
3.2	Objekte	64
3.3	Klassen	65

3.3.1	Attribute	65
3.3.2	Methoden	67
3.4	Abstraktion	69
3.5	Vererbung	70
3.5.1	Basisklassen	72
3.5.2	Abgeleitete Klassen	72
3.5.3	Mehrfachvererbung	73
3.6	Kapselung	73
3.7	Beziehungen	74
3.7.1	Beziehungen, die nicht auf Vererbung beruhen	75
3.7.2	Vererbungsbeziehungen	76
3.8	Designfehler	78
3.9	Umstrukturierung	78
3.10	Modellierung	78
3.11	Persistenz	78
3.12	Polymorphie	79
3.12.1	Statische Polymorphie	80
3.12.2	Dynamische Polymorphie	80
3.13	Designregeln	80
3.14	Zusammenfassung	81
3.15	Aufgaben	81
3.15.1	Fragen	81
3.15.2	Übungen	82

Teil 2 Java im Detail

4 Sprache 89

4.1	Einleitung	89
4.1.1	Geschichte	89
4.1.2	Beschreibung mittels Text	89
4.1.3	Überblick über die Sprachelemente	90
4.2	Schlüsselwörter	92
4.3	Einfache Datentypen	93
4.3.1	Grundlagen	93
4.3.2	Festkommazahlen	97
4.3.3	Gleitkommazahlen	100
4.3.4	Wahrheitswerte	101
4.3.5	Zeichen	102
4.4	Erweiterte Datentypen	102
4.4.1	Arrays	102

4.4.2	Aufzählungstyp	105
4.5	Benutzerdefinierte Datentypen	106
4.5.1	Konkrete Klassen	106
4.5.2	Abstrakte Klassen	109
4.5.3	Interfaces	110
4.5.4	Generische Klassen	111
4.6	Variablen	112
4.7	Konstanten	112
4.8	Methoden	112
4.8.1	Methodenarten	113
4.8.2	Konstruktoren	115
4.8.3	Destruktoren	116
4.8.4	Akzessoren	116
4.8.5	Mutatoren	117
4.8.6	Funktionen	117
4.9	Operatoren	118
4.9.1	Arithmetische Operatoren	118
4.9.2	Vergleichende Operatoren	123
4.9.3	Logische Operatoren	126
4.9.4	Bitweise Operatoren	128
4.9.5	Zuweisungsoperatoren	128
4.9.6	Fragezeichenoperator	129
4.9.7	New-Operator	130
4.9.8	Cast-Operator	130
4.10	Ausdrücke	131
4.10.1	Zuweisungen	131
4.10.2	Elementare Anweisungen	133
4.10.3	Verzweigungen	134
4.10.4	Schleifen	135
4.11	Module	138
4.11.1	Klassenimport	138
4.11.2	Namensräume	140
4.12	Dokumentation	141
4.12.1	Zeilenbezogene Kommentare	141
4.12.2	Abschnittsbezogene Kommentare	141
4.12.3	Dokumentationskommentare	141
4.13	Zusammenfassung	142
4.14	Aufgaben	142
4.14.1	Fragen	142
4.14.2	Übungen	143

5 Entwicklungsprozesse 147

5.1	Einleitung	147
5.1.1	Phasen	147
5.1.2	Aktivitäten	148
5.1.3	Werkzeuge	149
5.2	Planungsphase	150
5.2.1	Missverständnisse	150
5.2.2	Anforderungen aufnehmen	150
5.3	Konstruktionsphase	151
5.3.1	Objektorientierte Analyse	151
5.3.2	Objektorientiertes Design	151
5.3.3	Implementierung in Java	152
5.3.4	Test	160
5.4	Betriebsphase	170
5.4.1	Verteilung	170
5.4.2	Pflege	170
5.5	Zusammenfassung	171
5.6	Aufgaben	171
5.6.1	Fragen	171
5.6.2	Übungen	171

6 Plattform 175

6.1	Einleitung	175
6.2	Bytecode	175
6.3	Java Runtime Environment	177
6.3.1	Virtuelle Maschine	178
6.3.2	Garbage Collector	182
6.3.3	Bibliotheken	183
6.3.4	Ressourcen und Property-Dateien	183
6.4	Native Java-Programme	183
6.5	Portabilität eines Java-Programms	185
6.5.1	Binärkompatibler Bytecode	185
6.5.2	Voraussetzungen	187
6.6	Starten eines Java-Programms	188
6.6.1	Application	188
6.6.2	Applet	190
6.6.3	Servlets und JavaServer Pages	191
6.7	Zusammenfassung	191
6.8	Aufgaben	192
6.8.1	Fragen	192
6.8.2	Übungen	192

7 Gesetzmäßigkeiten 195

7.1	Einleitung	195
7.2	Sichtbarkeit	195
7.2.1	Klassenkapselung	195
7.2.2	Gültigkeitsbereich von Variablen	203
7.3	Auswertungsreihenfolge	205
7.3.1	Punkt vor Strich	205
7.3.2	Punkt vor Punkt	206
7.4	Typkonvertierung	208
7.4.1	Implizite Konvertierung	209
7.4.2	Explizite Konvertierung	211
7.5	Polymorphie	213
7.5.1	Überladen von Methoden	213
7.5.2	Überschreiben von Methoden	215
7.6	Programmierkonventionen	218
7.6.1	Vorschriften zur Schreibweise	218
7.6.2	Empfehlungen zur Schreibweise	219
7.7	Zusammenfassung	221
7.7.1	Sichtbarkeit	221
7.7.2	Auswertungsreihenfolge	221
7.7.3	Typkonvertierung	221
7.7.4	Polymorphie	222
7.7.5	Programmierkonventionen	222
7.8	Aufgaben	222
7.8.1	Fragen	222
7.8.2	Übungen	223

8 Klassenbibliotheken 227

8.1	Einleitung	227
8.1.1	Von der Klasse zur Bibliothek	227
8.1.2	Von der Bibliothek zum Universum	228
8.1.3	Vom Universum zum eigenen Programm	228
8.1.4	Bibliotheken und Bücher	228
8.1.5	Bibliotheken erweitern die Sprache	229
8.1.6	Bibliotheken steigern die Produktivität	229
8.1.7	Kommerzielle Klassenbibliotheken	230
8.1.8	Open-Source-Bibliotheken	230
8.1.9	Bibliotheken von Sun Microsystems	230
8.2	Java 2 Standard Edition	230
8.2.1	Java-Language-Bibliothek	231
8.2.2	Klasse System	237
8.2.3	Stream-Bibliotheken	247

8.2.4	Hilfsklassen	249
8.2.5	Abstract Windowing Toolkit	250
8.2.6	Swing	260
8.2.7	JavaBeans	264
8.2.8	Applets	265
8.2.9	Applications	267
8.2.10	Java Database Connectivity (JDBC)	267
8.2.11	Java Native Interface	269
8.2.12	Remote Method Invocation	269
8.3	Java 2 Enterprise Edition	270
8.3.1	Servlets	271
8.3.2	JavaServer Pages	272
8.3.3	CORBA	273
8.3.4	Enterprise JavaBeans	274
8.4	Java 2 Micro Edition	275
8.5	Zusammenfassung	277
8.6	Aufgaben	277
8.6.1	Fragen	277
8.6.2	Übungen	278

9 Algorithmen 281

9.1	Einleitung	281
9.2	Algorithmen entwickeln	281
9.3	Algorithmenarten	282
9.3.1	Sortieren	283
9.3.2	Diagramme	283
9.4	Algorithmen anwenden	288
9.4.1	Sortieren	289
9.4.2	Suchen	290
9.5	Aufgaben	291
9.5.1	Fragen	291
9.5.2	Übungen	291

Teil 3 Größere Java-Projekte

10 Konsolenprogramme 297

10.1	Einleitung	297
10.2	Projekt »Transfer«	297
10.2.1	Anforderungen	297
10.2.2	Analyse und Design	298

10.2.3	Implementierung der Klasse »TransferApp«	300
10.2.4	Implementierung der Klasse »CopyThread«	303
10.2.5	Implementierung der Properties-Datei	307
10.2.6	Test	308
10.2.7	Verteilung	308
10.3	Aufgaben	309
10.3.1	Fragen	309
10.3.2	Übungen	309

11 Erste Schritte mit grafischen Oberflächen 313

11.1	Einleitung	313
11.2	Projekt »Memory«	313
11.2.1	Anforderungen	313
11.2.2	Analyse und Design	315
11.2.3	Implementierung der Klasse »Card«	318
11.2.4	Implementierung der Klasse »CardEvent«	325
11.2.5	Implementierung des Interfaces »CardListener«	326
11.2.6	Implementierung der Klasse »CardBeanInfo«	326
11.2.7	Implementierung des Testtreibers	328
11.2.8	Implementierung der Klasse »GameBoard«	331
11.2.9	Implementierung des Hauptfensters	334
11.2.10	Implementierung der Klasse »AboutDlg«	337
11.2.11	Test	341
11.2.12	Verteilung	342
11.3	Zusammenfassung	342
11.4	Aufgaben	343
11.4.1	Fragen	343
11.4.2	Übungen	343

12 Einfache Oberflächen mit AWT und Swing 347

12.1	Einleitung	347
12.2	Projekt »Abakus«	347
12.2.1	Anforderungen	347
12.2.2	Analyse und Design	349
12.2.3	Implementierung der Applikationsklasse	353
12.2.4	Implementierung des Hauptfensters	354
12.2.5	Implementierung der Klasse »AboutDlg«	370
12.2.6	Zeichen als Unicode codieren	370
12.2.7	Dialog zentriert sich selbst	370
12.3	Zusammenfassung	371
12.4	Aufgaben	372
12.4.1	Fragen	372
12.4.2	Übungen	373

13 Komplexe Oberflächen mit Swing 377

13.1	Einleitung	377
13.2	Projekt »Nestor« – die Oberfläche	377
13.2.1	Anforderungen	377
13.2.2	Analyse und Design	379
13.2.3	Implementierung der Datenbankfassade	383
13.2.4	Implementierung der Applikationsklasse	384
13.2.5	Aufbau des Hauptfensters	385
13.2.6	Implementierung der Adresskomponente	386
13.2.7	Implementierung des Hauptfensters	389
13.2.8	Implementierung des Dialogs »Einstellungen«	395
13.2.9	Test	395
13.2.10	Verteilung	397
13.3	Zusammenfassung	397
13.4	Aufgaben	398
13.4.1	Fragen	398
13.4.2	Übungen	398

14 Weboberflächen mit Servlets 403

14.1	Einleitung	403
14.1.1	Hypertext Markup Language	403
14.1.2	Hypertext-Transfer-Protokoll	406
14.1.3	Common Gateway Interface	408
14.1.4	Servlets	408
14.2	Projekt »Xenia« – die Oberfläche	409
14.2.1	Anforderungen	409
14.2.2	Analyse und Design	411
14.2.3	Implementierung der HTML-Vorlagen	412
14.2.4	Implementierung der Klasse »GuestList«	414
14.2.5	Implementierung der Klasse »NewGuest«	419
14.2.6	Verteilung	425
14.3	Zusammenfassung	425
14.4	Aufgaben	426
14.4.1	Fragen	426
14.4.2	Übungen	426

15 Datenbankprogrammierung 429

15.1	Einleitung	429
15.1.1	Vom Modell zum Datenmodell	429
15.1.2	Vom Datenmodell zur Datenbank	429
15.1.3	Von der Datenbank zu den Daten	430

15.1.4	Von den Daten zum Programm	430
15.2	Projekt »Hades«	431
15.2.1	Anforderungen	431
15.2.2	Analyse & Design	431
15.2.3	Implementierung	432
15.2.4	Test	433
15.3	Projekt »Charon«	433
15.3.1	Anforderungen	434
15.3.2	Implementierung der Klasse »HadesDb«	435
15.3.3	Implementierung der Klasse »Charon«	438
15.3.4	Implementierung der Klasse »HadesTest«	441
15.3.5	Implementierung der Klasse »CharonTest«	443
15.3.6	Implementierung der Datei »Db.properties«	444
15.3.7	Test	445
15.3.8	Verteilung	446
15.4	Zusammenfassung	446
15.5	Aufgaben	446
15.5.1	Fragen	446
15.5.2	Übungen	446

16 Datenbankanwendungen 449

16.1	Einleitung	449
16.2	Projekt »Perseus«	449
16.2.1	Anforderungen	449
16.2.2	Analyse und Design	450
16.2.3	Implementierung der Klasse »BasisWnd«	453
16.2.4	Implementierung der Klasse »Alignment«	454
16.2.5	Implementierung der Klasse »SplashWnd«	455
16.2.6	Implementierung der Klasse »BasicDlg«	457
16.3	Projekt »Charon«	460
16.3.1	Anforderungen	460
16.3.2	Analyse und Design	460
16.3.3	Implementierung von »HadesDb«	461
16.3.4	Implementierung von »Charon«	461
16.3.5	Test	461
16.3.6	Verteilung	462
16.4	Projekt »Nestor«	462
16.4.1	Integration der Klasse »SplashWnd«	462
16.4.2	Integration der Klasse »SplashWnd«	463
16.4.3	Implementierung der Methode »showSplashScreen«	463
16.4.4	Integration der Klasse »BasicDlg«	464
16.4.5	Integration der Klasse »Charon«	465
16.4.6	Verteilung	465
16.5	Zusammenfassung	466

16.6	Aufgaben	466
16.6.1	Fragen	466
16.6.2	Übungen	466

17 Dynamische Websites 469

17.1	Einleitung	469
17.2	Projekt »Charon«	469
17.2.1	Anforderungen	469
17.2.2	Analyse und Design	469
17.2.3	Implementierung der Klasse »HadesDb«	470
17.2.4	Implementierung der Klasse »Charon«	472
17.3	Projekt »Xenia«	473
17.3.1	Anforderungen	473
17.3.2	Analyse und Design	473
17.3.3	Implementierung der Klasse »NewGuest«	474
17.3.4	Implementierung der Klasse »GuestList«	475
17.3.5	Änderungen am Projektverzeichnis	476
17.3.6	Test	477
17.3.7	Verteilung	479
17.4	Zusammenfassung	479
17.5	Aufgaben	480
17.5.1	Fragen	480
17.5.2	Übungen	480

Teil 4 Lösungen

18 Lösungen zu Teil I 485

18.1	Digitale Informationsverarbeitung	485
18.1.1	Zahlensysteme	485
18.1.2	Informationseinheiten	485
18.1.3	Zeichenkodierung	486
18.1.4	Kodierung logischer Informationen	486
18.2	Programmiersprachen	486
18.2.1	Programmiersprachen der ersten Generation	486
18.2.2	Programmiersprachen der zweiten Generation	487
18.2.3	Programmiersprachen der dritten Generation	487
18.3	Objektorientierte Programmierung	487
18.3.1	Fragen	487
18.3.2	Übungen	488

19 Lösungen zu Teil II 493

19.1	Sprache	493
19.1.1	Fragen	493
19.1.2	Übungen	495
19.2	Entwicklungsprozesse	496
19.2.1	Fragen	496
19.2.2	Übungen	497
19.3	Plattform	498
19.3.1	Fragen	498
19.3.2	Übungen	499
19.4	Gesetzmäßigkeiten	500
19.4.1	Fragen	500
19.4.2	Übungen	501
19.5	Klassenbibliotheken	501
19.5.1	Fragen	501
19.5.2	Übungen	502
19.6	Algorithmen	503
19.6.1	Fragen	503
19.6.2	Übungen	504

20 Lösungen zu Teil III 509

20.1	Konsolenprogramme	509
20.1.1	Fragen	509
20.1.2	Übungen	509
20.2	Erste Schritte mit grafischen Oberflächen	510
20.2.1	Fragen	510
20.2.2	Übungen	511
20.3	Einfache Oberflächen mit AWT und Swing	511
20.3.1	Fragen	511
20.3.2	Übungen	512
20.4	Komplexe Oberflächen mit Swing	513
20.4.1	Fragen	513
20.4.2	Übungen	514
20.5	Weboberflächen mit Servlets	515
20.5.1	Fragen	515
20.5.2	Übungen	515
20.6	Datenbankprogrammierung	516
20.6.1	Fragen	516
20.6.2	Übungen	516
20.7	Datenbankanwendungen	517

20.7.1	Fragen	517
20.7.2	Übungen	517
20.8	Dynamische Websites	518
20.8.1	Fragen	518
20.8.2	Übungen	518

Teil 5 Anhang

21 Werkzeuge 525

21.1	Einleitung	525
21.1.1	Einzelwerkzeuge versus Werkzeugsuiten	525
21.1.2	Zielgruppen	526
21.2	Kriterien zur Werkzeugauswahl	527
21.2.1	Allgemeine Kriterien	528
21.2.2	Projektverwaltung	531
21.2.3	Modellierungswerkzeuge	531
21.2.4	Texteditor	533
21.2.5	Java-Compiler	534
21.2.6	Java-Decompiler	535
21.2.7	GUI-Builder	535
21.2.8	Laufzeitumgebung	536
21.2.9	Java-Debugger	537
21.2.10	Werkzeuge zur Verteilung	538
21.2.11	Wizards	539
21.3	Einzelwerkzeuge	539
21.3.1	Modellierungswerkzeuge	539
21.3.2	Texteditor	540
21.3.3	Java-Compiler	541
21.3.4	Java-Decompiler	542
21.3.5	GUI-Builder	542
21.3.6	Laufzeitumgebungen	543
21.3.7	Java-Debugger	544
21.3.8	Versionskontrollwerkzeuge	544
21.3.9	Werkzeuge zur Verteilung	545
21.4	Werkzeugsuiten	545
21.4.1	Eclipse	546
21.4.2	JBuilder	548
21.4.3	Java Development Kit	549
21.4.4	NetBeans	555
21.4.5	Rational XDE	556
21.4.6	Sun One Studio	556
21.4.7	Together	556
21.4.8	VisualAge Java	557

21.4.9	WebSphere Studio	558
--------	------------------------	-----

22 Computerhardware 561

22.1	Einleitung	561
22.2	Aufbau eines Computers	561
22.3	Bussystem	561
22.4	Prozessoren	562
22.4.1	Central Processing Unit	562
22.4.2	Grafikprozessor	563
22.5	Speichermedien	563
22.5.1	Hauptspeicher	563
22.5.2	Festplattenspeicher	564
22.6	Ein- und Ausgabesteuerung	564
22.7	Taktgeber	565
22.8	Zusammenfassung	565

23 Glossar 569

24 Literatur 579

24.1	Basiswissen	579
24.2	Java im Detail	579
24.3	Größere Java-Projekte	580
24.4	Anhang	581

Index 583

3 Objektorientierte Programmierung

»Ich sehe ein Pferd, dann sehe ich noch ein Pferd – dann noch eins. Die Pferde sind nicht ganz gleich, aber es gibt etwas, das allen Pferden gemeinsam ist, und das, was allen Pferden gemeinsam ist, ist die Form des Pferds. Was unterschiedlich oder individuell ist, gehört zum Stoff des Pferds.«

(Jostein Gaarder)

3.1 Einleitung

Mitte der 60er-Jahre des letzten Jahrhunderts kam es zu einer Softwarekrise. Die Anforderungen an Programme stiegen, und die Software wurde dadurch komplexer sowie fehlerhafter. Auf Kongressen diskutierten Experten die Ursachen der Krise und die Gründe für die gestiegene Fehlerrate. Ein Teil der Softwareexperten kam zu dem Schluss, dass die Softwarekrise nicht mit den herkömmlichen Programmiersprachen zu bewältigen sei. Sie begannen deshalb, eine Generation von neuen Programmiersprachen zu entwickeln.

Die Entwickler dieser Sprachen kritisierten an den herkömmlichen Programmiersprachen vor allem, dass sich die natürliche Welt bisher nur unzureichend abbilden lasse. Um dem zu entgehen, gingen sie von natürlichen Begriffen aus, wie sie die Formenlehre der klassischen griechischen Philosophie geprägt hat, und wandelten sie für die Programmierung ab. Da sich alles um den Begriff des Objekts dreht, nannten sie die neue Generation von Sprachen »objektorientiert«.

3.1.1 Grundbegriffe

Alan Kay, einer der Erfinder der Programmiersprache *Smalltalk*, hat die Grundbegriffe der objektorientierten Programmierung folgendermaßen zusammengefasst:

- ▶ Alles ist ein Objekt.
- ▶ Objekte kommunizieren durch Nachrichtenaustausch.
- ▶ Objekte haben ihren eigenen Speicher.
- ▶ Jedes Objekt ist ein Exemplar einer Klasse.
- ▶ Die Klasse modelliert das gemeinsame Verhalten ihrer Objekte.
- ▶ Ein Programm wird ausgeführt, indem dem ersten Objekt die Kontrolle übergeben und der Rest als dessen Nachricht behandelt wird.

3.1.2 Prinzipien

Neben diesen Grundbegriffen sind folgende Prinzipien wichtig:

- ▶ Abstraktion
- ▶ Vererbung
- ▶ Kapselung
- ▶ Beziehungen
- ▶ Persistenz
- ▶ Polymorphie

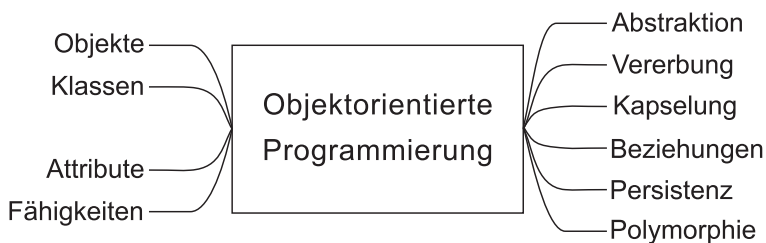


Abbildung 3.1 Hauptbegriffe der objektorientierten Programmierung

3.2 Objekte

Objekte sind für ein Java-Programm das, was Zellen für einen Organismus bedeuten: Aus diesen kleinsten Einheiten setzt sich eine Anwendung zusammen. Objekte haben eine bestimmte Gestalt (Aussehen, Attribute, Kennzeichen) und bestimmte Fähigkeiten (Methoden, Funktionen). Gestalt und Fähigkeiten eines Objekts werden durch seine Erbinformationen bestimmt. Diese Erbinformationen sind der Bauplan, nach dem das Objekt erzeugt wird. In der objektorientierten Programmierung ist der Bauplan eines Objekts seine *Klasse*.

Wenn Sie eine Reihe von Pferden betrachten, fällt Ihnen auf, dass ihnen die prinzipielle *Gestalt* gemeinsam ist. Ihre Unterschiede sind die *Attribute*, die das einzelne Pferd kennzeichnen. Ein Objekt der Klasse *Pferd* ist beispielsweise ein Pferd mit dem Namen *Xanthos*, ein anderes Pferd heißt *Balios*. Beide *Exemplare*¹ haben einen ähnlichen Körperbau (*Gestalt*) und ähnliche Fähigkeiten. Sie können beispielsweise beide laufen und wiehern.

¹ Exemplar und Objekt sind gleichbedeutend. Im Gegensatz dazu ist der Begriff »Instanz« eine Fehlübersetzung (engl. instance: Exemplar) und taucht in diesem Buch deshalb nicht auf.

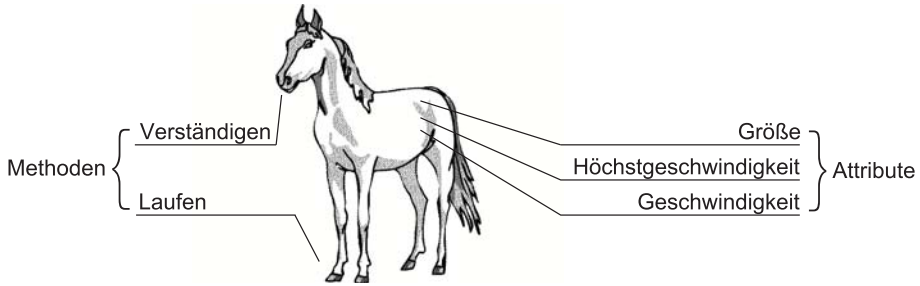


Abbildung 3.2 Jedes Objekt besitzt sein bestimmtes Verhalten und Aussehen.

Die beiden Objekte weisen aber auch einige deutliche Unterschiede auf: *Xanthos* ist weiß und *Balios* braun, und *Xanthos* kann schneller laufen als *Balios*. Obwohl *Xanthos* und *Balios* zur gleichen Klasse gehören, sind nur ihre *prinzipiellen* Fähigkeiten identisch, nicht aber ihre *individuellen* Attribute. Was das bedeutet, wird im nächsten Abschnitt deutlich.

3.3 Klassen

Xanthos und *Balios* gehören zu der Klasse *Pferd*. Die Klasse ist es, die die prinzipielle Gestalt und die Fähigkeiten der beiden Pferde-Objekte festlegt. Man bezeichnet Klassen daher als

- ▶ *Bauplan* für Objekte *oder* als
- ▶ *Oberbegriff* für verschiedene Objekte (Klassifizierung) *oder* als
- ▶ *Schablone* für verschiedene Objekte.

3.3.1 Attribute

Die eingangs erwähnte Klasse *Pferd* soll die Attribute *Größe*, *Höchstgeschwindigkeit* und *Geschwindigkeit* besitzen. Wenn aus dieser Klasse neue Objekte (neue Exemplare) entstehen, besitzen *alle* eine bestimmte *Größe*, eine bestimmte *Höchstgeschwindigkeit* und eine bestimmte *Geschwindigkeit* – aber welche Werte haben diese Attribute? Sie werden erst beim Entstehen (Erzeugen) des Objekts mit den individuellen Werten belegt.

Konstanten

Beispielsweise soll *Xanthos* über folgende Attribute verfügen: *Größe* (Stockmaß) 1,90 m, *Höchstgeschwindigkeit* 65 km/h, *Geschwindigkeit* 0 km/h. *Balios* hingegen soll 1,85 m groß sein, maximal 60 km/h schnell laufen können und momentan gerade 5 km/h laufen.

Obwohl beide Pferde nach dem gleichen Bauplan erzeugt worden sind, sind zwei deutlich unterschiedliche Objekte entstanden: Beide sind unterschiedlich groß, können laufen, aber unterschiedlich schnell, beide besitzen eine Geschwindigkeit, aber ein Pferd steht und das andere bewegt sich langsam.

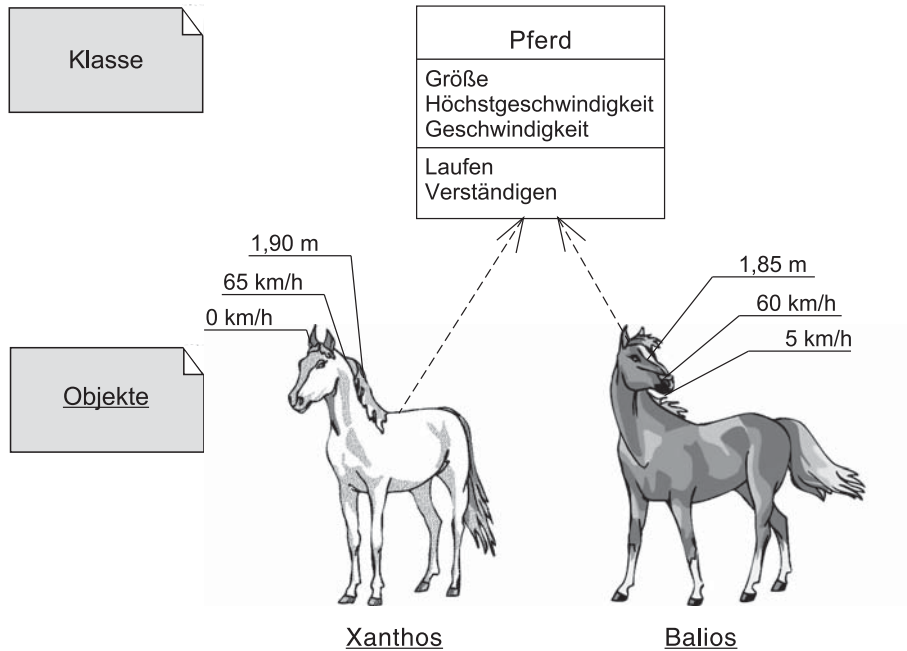


Abbildung 3.3 Die Klasse »Pferd« liefert den Bauplan für Pferde-Objekte.

Zustände

Es ist Ihnen vielleicht aufgefallen, dass bei den bisherigen Attributen der beiden Pferde einige mit festen Werten belegt waren, andere hingegen mit veränderlichen Werten. Die flexiblen Attribute beschreiben den *Zustand* des Objekts. Zum Beispiel beschreibt die *Geschwindigkeit*, wie schnell sich Xanthos gerade bewegt. Der Zustand eines Objekts kann sich im Laufe der Zeit ändern.

Kennungen

Was würde passieren, wenn man Xanthos und Balios so erzeugen würde, dass sie die gleiche *Größe*, die gleiche *Höchstgeschwindigkeit* und die gleiche momentane *Geschwindigkeit* besitzen? Wie könnte man sie dann unterscheiden? In diesem Fall haben beide Objekte zwar individuelle Werte für ihre Attribute bekommen, aber diese sind zufällig gleich. Damit gleichen sich auch die Objekte in einem Programm wie eineiige Zwillinge.

Um die Pferde zu unterscheiden, benötigt man so etwas wie einen genetischen Fingerabdruck. In der Programmierung vergibt der Entwickler eine so genannte Kennung. Diese Kennung ist ein zusätzliches Attribut, bei dem darauf geachtet wird, dass es *eindeutig* ist. Objekte der *gleichen Klasse* besitzen also die gleichen Attribute, aber mit individuellen Werten. Erst die Kennung eines Objekts sorgt dafür, dass das Programm unterschiedliche Exemplare auch dann unterscheiden kann, wenn ihre Attribute zufällig die gleichen Werte besitzen.

3.3.2 Methoden

Angenommen, Sie wollen dem Objekt *Balios* mitteilen, dass es nun springen soll. Im wirklichen Leben geben Sie ihm dazu ein Zeichen. In der objektorientierten Programmierung müssen Sie stattdessen eine Methode des Objekts *Balios* aufrufen. Statt »Methode« werden Sie auch öfter auf die Begriffe »Botschaft« (Smalltalk), »Nachricht« oder »Operation« stoßen, die das Gleiche bedeuten sollen.

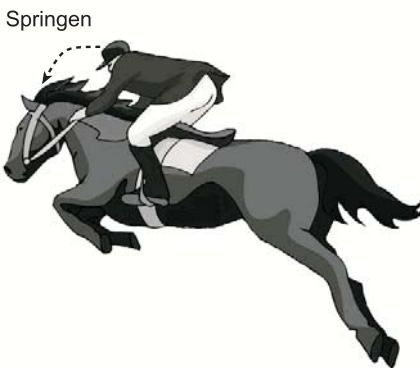


Abbildung 3.4 Objekte verständigen sich durch den Austausch von Nachrichten.

Egal, wie der Begriff nun bei den verschiedenen Programmiersprachen und in der Literatur genannt wird, eines ist gleich: Verhaltensweisen wie *Laufen* und *Verständigen* bestimmen die Fähigkeit eines Objekts zu kommunizieren und Aufgaben zu erledigen. Objekte verständigen sich also über Methoden.

Es existiert nicht nur eine Art von Methoden, sondern es gibt folgende fünf Grundtypen: *Konstruktoren* (»Erbauer«), *Destruktoren* (»Zerstörer«), *Mutatoren* (»Veränderer«), *Akzessoren* (»Zugriffsmethoden«) und *Funktionen* (»Tätigkeiten«).

Index

A

Abakus 347
Abgeleitete Klasse 72
Ableiten 108
Abschnittsbezogene Kommentare 141
abstract 92
Abstract Windowing Toolkit 250, 569
Abstrakte Klasse 106, 109, 569
Abstrakte Methode 569
Abstraktion 64, 69
Acceleratoren 569
Active Server Pages 272
Aggregation 75
Aktivitäten 148
Akzessor 68, 116
Algol 51
Algorithmen 281
 anwenden 288
Algorithmen entwickeln 281
Algorithmenarten 282
American National Standards Institute 569
American Standard Code for Information Interchange 569
Analyse 148
Anforderungsaufnahme 148
Anonyme Klasse 108
ANSI 569
ANSI-Code 35
Anweisungen 131
Anwendungsarchitektur 569
Anwendungsfall 569
API 569
Applets 265
appletviewer 553
Application 267
Application Objects 274
Application Programming Interface 569
Application Server 543

Applikationsschicht 270
Architektur 569, 571, 574
ArgoUML 540
Argument 114
Argumente 114, 117
Argumente übergeben 117
Arithmetische Operatoren 118
Arrays 102
ASCII 569
ASCII-Code 33
ASP 272
Assembler-Sprache 48
assert 92
Assoziation 75
Attribut 64, 65
Aufbau eines Computers 561
Aufzählungstyp 105
Ausdruck 131
Ausnahmebehandlung 240
AWT 250, 569

B

BASIC 51
Basisklasse 72, 569
Behälterklasse 569
Betriebsphase 147
Bezeichner 95
Beziehung 74, 75
Beziehungen 64
Bildlauffeld 569
Bildlauffeiste 569
Binärcode 47, 50
Binärformat 46
Binärprogramm 28
Binärsystem 28
Binärzahlen 28, 31
Bit 32
Bitweise Operatoren 128
Block 133
BMP 274
boolean 92, 94, 101
Border-Layout 255
break 92

Bussystem 561
Button 569
BX for Java 542
Byte 33
byte 92, 94, 98, 104

C

C 51, 118, 134, 141, 269
C++ 51, 116, 118, 134, 269
C# 51
CardListener 326
case 92
Case-Verzweigung 134
Cast-Operator 130, 208
catch 92
Central Processing Unit 562
CGI 271, 403, 408, 570
CGI-Programm 570
char 92, 94, 102, 104
Charon 433, 469
class 92
Clipboard 575
CMP 274
COBOL 51
Combo Box 570
Common Facilities 274
Common Gateway Interface 271, 408, 570
Common Object Services 274
Compiler 154
Compilieren 154
Computer Aided Software Engineering 570
Computerhardware 561
const 92
Container 250
Container Managed Persistence 274
Container-Klasse 570
continue 92
Coprozessor 96
CORBA 273, 537, 570

CPU 562

CVS 545

D

Datenbank 429

Datenbank-Management-
system 570

Datenbank-
anwendungen 449

Datenbanken 537

Datenbank-
programmierung 429

Datenmodell 429

Datentyp 94

DBMS 570

Debugger 537

default 92, 195

Deklaration 94

Deployment 545

Derby 543

Design 148

Designfehler 78

Designregel 80

Destruktor 68, 116

Dezimaldarstellung 27

Dezimalsystem 27

Dezimalzahl 29

Dialog 570

Dialogfeld 570

Dialogfenster 570

Differenz 120

Digitalcomputer 28

Digitalsystem 28

Digitalzahlen 28

do 92

Do-Schleife 136

doGet 414

Dokumentation 141

Dokumentations-
kommentare 141

Doppelwort 32

double 92, 94, 101

Dreamweaver 542

Dualsystem 28

Dünnere Treiber 268

Dynamische Polymor-
phie 80

Dynamische Websites 469

E

Eclipse 546

Editieren 153

Ein- und Ausgabesteue-
rung 564

Einfache Klassentypen 235

Einfacher Datentyp 93

Einfachvererbung 570

Einzelwerkzeuge 539

EJB 274, 537

Elementare Anwei-
sungen 133

else 92

Enterprise JavaBeans 270,
274, 275

Entity Beans 275

Entwicklungsprozess 147

Entwurfsmuster 570

enum 92, 93

Enumeration 570

ER-Modell 431, 570

Ereignis 570

Ereignisbehandlung 252

Ereignissteuerung 252, 264

Erweiterter Datentyp 102

Event-Handling 252

Excelsior 541

Exception Handling 240,
247

Exemplar 570

extends 92, 108, 109

Extensible Markup

Language 570

Extranet 570

F

Fachliche Architektur 571

Fachliches Klassen-
modell 571

false 92

Fehlerbehandlung 240,
241

Festkommazahl 97

Festplattenspeicher 564

FileReader 247

FileWriter 248

final 92

finalize 116

finally 92

Firebird 544

Firewalls 266

Flash 403

float 92, 94, 100

Floating Window 571

Fokus 571

for 92

For-Schleife 136

FORTRAN 51

Fragezeichenoperator 129

FTP 571

Funktion 69, 117

G

Ganzzahl 96

Garbage Collector 182

GByte 32

Genauigkeit 95

Generalisierung 76, 571

Generics 106, 111

Generische Klasse 106,
111

Gleitkommazahl 96

goto 92

Grafikprozessor 563

Graphics 284

GridBag-Layout 257

Group Box 571

Gruppenfeld 571

GUI 571

GUI-Builder 535, 542

H

Hades 431

HadesTest 441

Handler 340, 394

Hauptspeicher 563

Heap 563

Hexadezimalsystem 30

Hilfsklasse 249

Höhere Datentypen 106

Home Interface 275

Hot Swap 538

HotJava 89

hsqldb 544