



Rolf Dornberger | Rainer Telesko

Java-Training zur Objektorientierten Programmierung

Leitfaden für Lehre, Unterricht und Selbststudium

```
public static void main (String[] args) {  
    main (String[] args) {  
        public static void main (String[] args) {  
            public static void main (String[] args) {  
                public static void main (String[] args) {  
                    void main (String[] args) {
```



Java-Training zur Objektorientierten Programmierung

Leitfaden für Lehre, Unterricht und
Selbststudium

von
Prof. Dr. Rolf Dornberger und
Prof. Dr. Rainer Telesko

Oldenbourg Verlag München

Prof. Dr. Rolf Dornberger ist Leiter des Instituts für Wirtschaftsinformatik an der Hochschule für Wirtschaft der Fachhochschule Nordwestschweiz FHNW, Basel und Olten.

Prof. Dr. Rainer Telesko lehrt und forscht im Bereich Information & Knowledge Management des Instituts für Wirtschaftsinformatik an der Hochschule für Wirtschaft der Fachhochschule Nordwestschweiz FHNW, Basel und Olten.

Bibliografische Information der Deutschen Nationalbibliothek

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de> abrufbar.

© 2010 Oldenbourg Wissenschaftsverlag GmbH
Rosenheimer Straße 145, D-81671 München
Telefon: (089) 45051-0
oldenbourg.de

Das Werk einschließlich aller Abbildungen ist urheberrechtlich geschützt. Jede Verwertung außerhalb der Grenzen des Urheberrechtsgesetzes ist ohne Zustimmung des Verlages unzulässig und strafbar. Das gilt insbesondere für Vervielfältigungen, Übersetzungen, Mikroverfilmungen und die Einspeicherung und Bearbeitung in elektronischen Systemen.

Lektorat: Kathrin Mönch
Herstellung: Anna Grosser
Coverentwurf: Kochan & Partner, München
Gedruckt auf säure- und chlorfreiem Papier
Gesamtherstellung: Grafik + Druck GmbH, München

ISBN 978-3-486-58739-5

Vorwort

Der fortschreitende Einzug von Computern aller Art ist mittlerweile aus unserem Alltag nicht mehr wegzudenken – sowohl bei der Arbeit als auch im privaten Umfeld: E-Mails schreiben und im Internet bzw. World Wide Web surfen sind nichts Besonderes mehr. Die Unterhaltungselektronik setzt Multimedia Devices ein; Telefone sind mobil, multimedial brillant und noch dazu online-, E-Mail- und GPS-fähig. Das reale Treffen mit Freunden wird durch virtuelle Welten und Social-Networks abgelöst. Behördengänge werden mit elektronischen Portalen und Prozessen im Internet unterstützt und werden dadurch teilweise in der realen Welt sogar schon obsolet. Das Einkaufserlebnis findet in verschiedenen Webshops mithilfe von Suchmaschinen für die günstigsten Angebote statt. Das Lesen von Romanen und Tageszeitungen geschieht zunehmend auf E-Books oder anderen mobilen Geräten. Autos regeln ihre Motorleistung entsprechend ihrer Fahrleistungen sowie Verkehrsverhältnissen und parken außerdem noch autonom in der kleinsten Lücke am Straßenrand. In der Schule lehrt die E-Learning-Software neben den Lehrkräften. In Zügen fährt und in Flugzeugen fliegt mehr der Computer als der Mensch ...

Wir Menschen nehmen dies zunehmend als Selbstverständlichkeit wahr. Doch eine Frage, neben vielen anderen, bleibt: Wo kommt eigentlich die Software her, welche die Hardware erst dazu bringt, diese Funktionalitäten zu ermöglichen?

Während gegen Ende des letzten Jahrhunderts viel von Software-Entwicklung im großen Stil und speziell von der Programmierung gesprochen wurde, geht heute dieses Thema fast unter, obwohl mehr Software benötigt wird als je zuvor. Sicher, die Erstellung von Software ist einen großen Schritt über das alte Klischee hinausgekommen, dass bärtige, notorisch übernächtigte Programmierer im dunklen Kämmerlein Zeichen in den Computer hacken. Heute wird Software in industriellen Prozessen mit sinnvollen, unterstützenden Software-Werkzeugen entwickelt (welche übrigens auch einmal programmiert werden mussten).

Warum aber noch ein weiteres Buch über Java? Denn es gibt ja schon viele gute Bücher zur Programmierung mit Java, ob objektorientiert oder nicht.

Dieses Buch ist aus dem Bedürfnis entstanden, dass wir – zwei Professoren der Wirtschaftsinformatik – über Jahre hinweg versuchen, einen bestmöglichen Zugang für unsere Studierende zum Programmieren zu finden. Programmieren bedeutet hierbei nicht nur, syntaktisch korrekte Programme zu schreiben, sondern insbesondere auch die Philosophie der Programmierung und den Einstieg in die Objektorientiertheit zu verstehen. Da naturgemäß unsere Wirtschaftsinformatik-Studierenden nicht die Personen sind, die bereits mit den besten Programmierkenntnissen oder der größten Motivation starten, mussten wir das Buch ganz einfach beginnen und dennoch bemüht sein, innerhalb von zwei Semestern ein größeres Interes-

se für die Programmierung angefacht und fundiertes Wissen vermittelt zu haben. Das Buch ist dabei so konzipiert, dass jedes Kapitel mit den Unterkapiteln Lernziele und Aufgaben schließt, die zur Wiederholung bzw. Vertiefung des Stoffinhaltes dienen. Außerdem fassen wir in verschiedenen Kapiteln immer wieder einen Teil des Stoffes aus den vorherigen Kapiteln zusammen, so dass diese neben der Wiederholung auch den Einstieg von Quereinsteigern, die bereits gewisse Vorkenntnisse mitbringen, ermöglicht.

Nach einer kleinen Einleitung in Kapitel 1, was Java ist, und in Kapitel 2, wie Java auf dem eigenen Computer zum Laufen gebracht und mit Java programmiert werden kann, kommen wir in Kapitel 3 zur syntaktischen Struktur von Java, also welche Computersprachelemente, Datentypen und Operatoren in dieser Programmiersprache existieren. Kapitel 4 vertieft die Datenfelder, Datentypen und deren Umwandlung. Während sich Kapitel 5 mit elementaren Anweisungen und Bedingungen beschäftigt, stellt Kapitel 6 die verschiedenen Arten von Schleifen vor. Die erste Zusammenfassung der Möglichkeiten mit Java wird in Kapitel 7 geboten und sogleich auch für die Entwicklung von Algorithmen in Java angewendet. Kapitel 8 startet mit einem Einblick in die Objektorientiertheit mit Klassen und Objekten und deren Beziehung untereinander. Kapitel 9 erlaubt sich dann einen Ausblick auf die Möglichkeiten von Java und stellt Java-Applets – kleine Java-Programme fürs Internet bzw. World Wide Web – vor. Kapitel 10 fasst danach nochmals das Wichtigste zusammen und erweitert die objektorientierte Sichtweise um zusätzliche Elemente in Java.

Nach all diesen grundlegenden Themen bieten wir in Kapitel 11 einen Einstieg in die Grafik-Programmierung mithilfe der Java-Grafikpakete AWT, Swing und des Java 2D API. In Kapitel 12 gehen wir dann über zur Programmierung von grafischen Benutzeroberflächen (GUI) und dem Event-Handling. In Kapitel 13 fassen wir nochmals unser Wissen zur objektorientierten Programmierung zusammen und reichern es um fortgeschrittene Konzepte wie Interfaces und Polymorphismus an. Kapitel 14 stellt Klassen und Methoden für die Programmierung von Input und Output von Daten vor. In Kapitel 15 geht es um Threads, eine interessante Möglichkeit in Java-Programmen parallele Aufgaben gleichzeitig ausführen zu lassen. Kapitel 16 schlägt den Bogen zur Modellierung von Software-Spezifikationen mit der Modellierungssprache UML und deren Programmierung in Java. Kapitel 17 stellt danach die Java Enterprise Edition Technologie vor, mit welcher internetfähige Anwendungen und Datenbankverbindungen mit Java Server Pages (JSP) programmiert werden können. Die nachfolgenden Kapitel beschließen dann das Buch mit Ausblicken, Einblicken und Lösungen zu den Aufgaben.

Dieses Buch erscheint uns nun für die Lehre an der Hochschule (beispielsweise in der Wirtschaftsinformatik, Informatik, dem Ingenieurwesen oder sonstigen Disziplinen), aber auch für den Oberstufen-Informatikunterricht an Schulen sehr gut geeignet zu sein. Als Leitfaden führt es sanft in Java ein, zeigt das Wichtigste zu Objektorientierter Programmierung und bietet in übersichtlicher Form eine Vertiefung in verwandte Themen. Somit ist das Buch natürlich auch bestens fürs Selbststudium geeignet.

Wir wünschen Ihnen nun viel Spaß und Erfolg mit Java und der Objektorientierten Programmierung!

Prof. Dr. Rolf Dornberger und Prof. Dr. Rainer Telesko

Inhalt

Vorwort	V
1 Einleitung	1
1.1 Programmierung und Programmiersprachen	1
1.1.1 Programmierung	2
1.1.2 Grundlegende Programmstrukturen	3
1.1.3 Programmiersprachen	5
1.2 Was ist Java?	9
1.2.1 Ursprung von Java	9
1.2.2 Vorzüge von Java	10
1.2.3 Missverständnisse zu Java	11
1.3 Beispiele für Java	12
1.4 Lernziele und Aufgaben	13
1.4.1 Lernziele	13
1.4.2 Aufgaben	13
2 Java Development Kit und Java Entwicklungswerkzeuge	15
2.1 JDK und seine Versionen	15
2.1.1 Übersicht JDK	15
2.1.2 Versionen JDK	17
2.2 Installation und Dokumentation	18
2.2.1 Erste Installation von Java	19
2.2.2 Nach der Installation von Java	21
2.2.3 Erstes Java-Programm – der Klassiker <i>Hello World</i>	21
2.3 Java Entwicklungs-Tools	23
2.3.1 Standard-Tools	23
2.3.2 Integrated Development Environments (IDE)	24
2.4 Lernziele und Aufgaben	25
2.4.1 Lernziele	25
2.4.2 Aufgaben	25

3	Token, Kommentare, Datentypen, Operatoren	27
3.1	Java-Token	27
3.1.1	Schlüsselwörter	28
3.1.2	Bezeichner	28
3.1.3	Literale	29
3.1.4	Trennzeichen	29
3.1.5	Operationen	29
3.2	Kommentare	30
3.3	Datentypen	32
3.3.1	Primitive Datentypen	32
3.3.2	Deklaration von Variablen	33
3.3.3	Literale	34
3.4	Operatoren	37
3.4.1	Arithmetische Operatoren	37
3.4.2	Arithmetische Zuweisungsoperatoren	38
3.4.3	Vergleichsoperatoren	39
3.4.4	Logische Operatoren	39
3.4.5	Bitweise Operatoren	40
3.4.6	Logische und bitweise Zuweisungsoperatoren	40
3.4.7	Fragezeichen-Operator	41
3.4.8	Operatoren-Priorität	42
3.5	Lernziele und Aufgaben	43
3.5.1	Lernziele	43
3.5.2	Aufgaben	43
4	Datenfelder, Parameterübergabe, Casting	45
4.1	Datenfelder	45
4.1.1	Erstellen von Arrays	46
4.1.2	Zugriff auf Array-Elemente	47
4.1.3	Dreiecksmatrizen	49
4.1.4	Referenztyp <code>String</code>	49
4.1.5	Kopieren von Datenfeldern	51
4.2	Parameterübergabe	53
4.3	Casting	55
4.3.1	Automatische Typenkonvertierung	55
4.3.2	Type-Cast-Operator	56
4.4	Lernziele und Aufgaben	58
4.4.1	Lernziele	58
4.4.2	Aufgaben	59

5	Elementare Anweisungen und Bedingungen	61
5.1	Elementare Anweisungen.....	61
5.1.1	Ausdrucksanweisung.....	61
5.1.2	Leere Anweisung.....	61
5.1.3	Blockanweisung.....	62
5.1.4	Variablendeklaration.....	62
5.2	Bedingungen.....	63
5.2.1	if-Bedingung.....	63
5.2.2	if-else-Bedingung.....	63
5.2.3	switch-Bedingung.....	65
5.3	Lernziele und Aufgabe.....	67
5.3.1	Lernziele.....	67
5.3.2	Aufgaben.....	68
6	Schleifen	69
6.1	Klassische Schleifen.....	69
6.1.1	do-Schleife.....	69
6.1.2	while-Schleife.....	70
6.1.3	for-Schleife.....	71
6.1.4	Varianten von for-Schleifen.....	72
6.1.5	Geschachtelte Schleifen.....	74
6.2	Sprunganweisungen.....	76
6.2.1	break-Anweisung.....	76
6.2.2	continue-Anweisung.....	78
6.3	Lernziele und Aufgaben.....	80
6.3.1	Lernziele.....	80
6.3.2	Aufgaben.....	80
7	Methoden, Algorithmen und Rekursion in Java	83
7.1	Auffrischung der grundlegenden Programmstrukturen.....	83
7.2	Grundelemente und Syntax in Java.....	84
7.2.1	Grundkonzept von Java.....	84
7.2.2	Sprachelemente, Datentypen und Variablen.....	84
7.2.3	Operatoren.....	85
7.2.4	Datenfelder.....	85
7.2.5	Zeichenketten.....	86
7.2.6	main-Methode und Parameterübergabe.....	86
7.2.7	Bedingungen und Schleifen.....	87
7.3	Methoden.....	88

7.4	Algorithmen	90
7.4.1	Sechsstellige Zufallszahl	90
7.4.2	Quadratwurzel nach Heron	94
7.5	Rekursion	96
7.6	Lernziele und Aufgaben	97
7.6.1	Lernziele	97
7.6.2	Aufgaben	98
8	Klassen, Objekte, Methoden, Vererbung, Konstruktoren	101
8.1	Konzepte objektorientierter Programmiersprachen	101
8.1.1	Klassen, Objekte, Instanz und Abstraktion	101
8.1.2	Attribute, Methoden, Kapselung und Botschaften	102
8.1.3	Beziehungen, Vererbung, Komposition, Aggregation, Assoziation	103
8.1.4	Polymorphismus, Überladen und Wiederverwendung	104
8.2	Implementierung objektorientierter Konzepte in Java	105
8.2.1	Klassen	105
8.2.2	Objekte	107
8.2.3	Mehrere Objekte von einer Klasse	111
8.2.4	Sukzessives Vorgehen in der Implementierung von Klassen und Objekten	113
8.3	Arrays von Objekten	113
8.3.1	Implementierung von Arrays von Objekten	113
8.3.2	Default-Werte von Objekten	116
8.3.3	Sukzessives Vorgehen in der Implementierung von Arrays von Objekten	117
8.4	Methoden	118
8.5	Vererbung	119
8.6	Konstruktoren	123
8.6.1	Definition von Konstruktoren	123
8.6.2	Anwendung von Konstruktoren	128
8.7	Lernziele und Aufgaben	133
8.7.1	Lernziele	133
8.7.2	Aufgaben	134
9	Applets	135
9.1	Internet	135
9.2	Funktionsweise des Internet	135
9.3	Java-Applets	138
9.3.1	Einführung Java-Applet	138
9.3.2	Standard-Methoden für Applets	143
9.3.3	Threads – Das Wichtigste für Applets in Kürze	146
9.3.4	Interfaces – Das Wichtigste für Applets in Kürze	146

9.3.5	Beispiel Applet mit Thread.....	148
9.3.6	Java-Archiv	150
9.4	Lernziele und Aufgaben	152
9.4.1	Lernziele.....	152
9.4.2	Aufgaben.....	152
10	Wiederholung der Grundelemente in Java	157
10.1	Java allgemein.....	157
10.1.1	Grundkonzept von Java	157
10.1.2	Sprachelemente, Datentypen und Variablen	158
10.1.3	Operatoren.....	159
10.1.4	Datenfelder	159
10.1.5	Zeichenketten	160
10.1.6	main-Methode und Parameterübergabe	160
10.1.7	Bedingungen und Schleifen.....	160
10.2	Wiederholung der grundlegenden Programmstrukturen.....	161
10.3	Konzepte objektorientierter Programmiersprachen	161
10.3.1	Klassen und Objekte.....	162
10.3.2	Attribute, Methoden und Kapselung	162
10.3.3	Vererbung und Assoziation.....	162
10.3.4	Polymorphismus und Überladen	163
10.4	Implementierung objektorientierter Konzepte in Java	164
10.4.1	Klassen	164
10.4.2	Objekte	164
10.4.3	Mehrere Objekte von einer Klasse	165
10.4.4	Schrittweises Vorgehen zur Implementierung von Klassen und Objekten	165
10.5	Arrays von Objekten	166
10.5.1	Anlegen von Arrays von Objekten	166
10.5.2	Default-Werte von Objekten.....	166
10.6	Methoden.....	166
10.7	Vererbung	167
10.8	Konstruktoren.....	167
10.8.1	Definition von Konstruktoren.....	167
10.8.2	Anwendung von Konstruktoren	168
10.9	Pakete	168
10.9.1	Pakete und Klassen importieren	168
10.9.2	Wichtige Pakete.....	169
10.9.3	Pakete erstellen.....	170
10.10	Java-Dokumentation.....	171

10.11	Modifiers – Modifikatoren.....	173
10.11.1	Modifiers für Zugriffsspezifizierung.....	173
10.11.2	Modifiers für statische und konstante Ausdrücke	174
10.11.3	Anwendbarkeit der Modifiers	175
10.12	Lernziele und Aufgaben	175
10.12.1	Lernziele	175
10.12.2	Aufgaben.....	176
11	Grafik	177
11.1	Einführung	177
11.2	Erste Schritte mit AWT	178
11.2.1	Fenster öffnen	178
11.2.2	Fenster schließen.....	179
11.3	Grundlegende Grafikoperationen in AWT	182
11.3.1	Klasse <code>Graphics</code>	182
11.3.2	Klasse <code>Font</code>	184
11.3.3	Klasse <code>Color</code>	185
11.4	Beispiele mit AWT.....	186
11.5	Grundlegende Grafikoperationen in Swing.....	189
11.6	Grafik-API Java 2D	190
11.6.1	Einführung	190
11.6.2	Grundbegriffe und Definitionen.....	191
11.6.3	Farben	192
11.6.4	Formen	195
11.6.5	Painting und Stroking.....	196
11.6.6	Weitere Möglichkeiten mit Java 2D.....	198
11.7	Lernziele und Aufgaben	199
11.7.1	Lernziele	199
11.7.2	Aufgaben.....	200
12	Grafische Benutzeroberflächen	201
12.1	Einleitung.....	201
12.2	Behälter – Container	202
12.3	GUI-Bedienelemente.....	204
12.4	Menüs.....	207
12.5	Layout	209
12.5.1	Allgemeines zu Layouts.....	209
12.5.2	Flow-Layout.....	210
12.5.3	Border-Layout.....	212

12.5.4	Grid-Layout.....	214
12.5.5	Weitere Layouts.....	216
12.6	Ereignisse und Verarbeitung.....	216
12.6.1	Event-Handling.....	216
12.6.2	<i>Handler</i> -Methode oder <i>inline</i>	219
12.6.3	<i>Listener</i> -Interfaces und <i>Adapter</i> -Klassen.....	220
12.7	Lernziele und Aufgaben.....	221
12.7.1	Lernziele.....	221
12.7.2	Aufgaben.....	221
13	Fortgeschrittene Konzepte der Objektorientierung	223
13.1	Exceptions.....	223
13.1.1	Behandlung von Ausnahmen mit <code>try</code> , <code>catch</code> und <code>finally</code>	223
13.1.2	Auswertung von Ausnahmen.....	225
13.1.3	Einlesen von Tastatureingaben mit <code>try</code> und <code>catch</code>	228
13.1.4	Weitergabe von Ausnahmen mit <code>throws</code>	230
13.1.5	Auslösen von Ausnahmen mit <code>throw</code>	231
13.2	Pakete.....	233
13.3	Zugriffsspezifizierung.....	235
13.3.1	Modifier für Zugriffsspezifizierung.....	235
13.3.2	Botschaften.....	238
13.4	Abstrakte Klassen und Methoden.....	239
13.4.1	Abstrakte Klassen.....	240
13.4.2	Abstrakte Methoden.....	241
13.5	Methodenaufrufe und Polymorphismus.....	242
13.5.1	Methoden überlagern.....	242
13.5.2	Polymorphismus.....	243
13.5.3	Dynamische Methodenaufrufe.....	244
13.6	Die Klasse <code>Object</code>	244
13.7	Klassenmethoden, Klassenattribute und Konstanten.....	246
13.7.1	Klassenattribute und Konstanten.....	247
13.7.2	Klassenmethoden.....	250
13.8	Interfaces.....	250
13.8.1	Interfaces anlegen.....	251
13.8.2	Interfaces verwenden.....	252
13.9	Lernziele und Aufgaben.....	256
13.9.1	Lernziele.....	256
13.9.2	Aufgaben.....	256

14	Input/Output	257
14.1	Einführung	257
14.2	Datei- und Verzeichnisverwaltung: File	259
14.3	Byte-Streams: InputStream und OutputStream	262
14.4	Zeichen-Streams: Reader und Writer	264
14.5	Filter-Streams	266
14.5.1	Filtern von bestimmten Datentypen beim Lesen und Schreiben	266
14.5.2	Ausgaben auf der Konsole	268
14.5.3	Eingaben von der Konsole	269
14.6	Lernziele und Aufgaben	270
14.6.1	Lernziele	270
14.6.2	Aufgaben	270
15	Threads	271
15.1	Was sind Threads?	271
15.2	Threads in Java	272
15.2.1	Klasse Thread und Interface Runnable	273
15.2.2	Der Einsatz von Threads	275
15.3	Schwierigkeiten mit Threads	277
15.3.1	Race Conditions	278
15.3.2	Deadlocks	279
15.3.3	Synchronisation	280
15.4	Thread-Zustände	280
15.4.1	Thread-Zustände NEW, WAITING und RUNNABLE	281
15.4.2	Thread-Zustände TIMED_WAITING und BLOCKED	282
15.4.3	Thread-Zustand TERMINATED	282
15.5	Organisation von Threads	283
15.5.1	Prioritäten	283
15.5.2	Thread-Gruppen	284
15.6	Lernziele und Aufgaben	285
15.6.1	Lernziele	285
15.6.2	Aufgaben	285
16	UML und Java	287
16.1	Grundlagen der Objektorientierten Modellierung	287
16.2	Zusammenhang zwischen UML und Java	287
16.3	Programmierungsumgebung Eclipse	290
16.3.1	Installation	290
16.4	UML-Plug-in für Eclipse	291

16.5	Lernziele und Aufgaben	296
16.5.1	Lernziele.....	296
16.5.2	Aufgaben.....	296
17	Einführung in die Java Enterprise Edition Technologie	297
17.1	Kurzüberblick Java EE.....	297
17.2	Installation der Infrastruktur für JSP	298
17.3	Sprachelemente von JSP	301
17.3.1	JSP-Ausdrücke (Expressions)	302
17.3.2	JSP-Scriptlets	302
17.3.3	JSP-Deklarationen.....	303
17.3.4	Implizite Objekte.....	304
17.3.5	JSP-Direktiven	305
17.3.6	Kommentare	305
17.4	Datenbankanbindung.....	306
17.5	Lernziele und Aufgaben	310
17.5.1	Lernziele.....	310
17.5.2	Aufgaben.....	310
18	Lösungen zu den Aufgaben	311
18.1	Kapitel 6: Schleifen.....	311
18.2	Kapitel 7: Methoden, Algorithmen und Rekursion in Java	314
18.3	Kapitel 8: Klassen, Objekte, Methoden, Vererbung, Konstruktoren	315
18.4	Kapitel 9: Java-Applets	316
18.5	Kapitel 11: Grafik.....	318
19	Literatur- und Bildnachweis	321

1 Einleitung

In diesem Kapitel versuchen wir, eine Übersicht über verschiedene Ausdrücke und Themenfelder, die im Umfeld der Programmierung anfallen, zusammenzustellen. Da in der Literatur manche Fachbegriffe nicht einheitlich verwendet werden, stellt dieses Kapitel einen Leitfaden dar, wie die Fachausdrücke in den nachfolgenden Kapiteln unseres Buches zu verstehen sind. Einerseits stellen wir damit einen roten Faden in diesem Buch her, andererseits erlauben wir der Leserschaft, Querbezüge zur Literatur herzustellen, wo oftmals das Gleiche mit anderen Fachausdrücken beschrieben wird.

1.1 Programmierung und Programmiersprachen

Software-Engineering ist die Wissenschaft, die Methodiken zur systematischen Erstellung von Software zur Verfügung stellt. Ein wichtiger Teil im Software-Engineering sind Konzepte bzw. Vorgehensweisen für eine gesteuerte bzw. planmäßige Entwicklung von Software. Die Software-Entwicklung lässt sich – unabhängig von den verschiedenen Vorgehensweisen – in Phasen einteilen. Der Entstehungs- und Lebenszyklus einer Software lässt sich im Allgemeinen mit folgenden Phasen charakterisieren:

- Planungsphase
- Definitionsphase
- Entwurfsphase
- Implementierungsphase
- Abnahme- und Einführungsphase
- Wartungs- und Pflegephase

Dass es nun genau sechs bis acht Phasen sein müssen, ist in der Literatur in den unterschiedlichen Konzepten nicht einheitlich.

Im Prinzip geht es darum, dass am Anfang die Kundenanforderungen aufgenommen werden (Requirements Engineering in der Planungs- und Definitionsphase). Dann werden diese Anforderungen in verschiedener Granularität auf Spezifikationen und Software-Architekturen heruntergebrochen (Definitionsphase und Entwurfsphase). Danach wird programmiert, also die Software-Komponenten erstellt (Programmcode geschrieben), die mittels der Spezifikationen alle gestellten Anforderungen erfüllen (Implementierungsphase). Die Software wird vom Auftraggeber (welches auch ein firmeninternes Team sein kann) abgenommen und bei ihm eingeführt. Nachträglich erstellte Software-Komponenten dienen zur War-

tung der Software, um deren Lauffähigkeit über längere Zeit sicherzustellen, sowie zur Pflege der Software, um neue, erwünschte Funktionalitäten hinzuzufügen. Dieser gesamte Prozess ist begleitet von ständigem Testen, Dokumentieren und Projekt- sowie Qualitätsmanagement; je nach Prozessmodell werden verschieden häufig und umfangreich Iterationen mit oder ohne Einbezug des Kunden durchgeführt.

Im vorliegenden Buch widmen wir uns speziell der Implementierungsphase. In dieser Phase werden die Spezifikationen, die beschreiben, was die Software erfüllen und wie sie arbeiten soll, in Programmcode umgesetzt: Hier findet also die eigentliche Programmierung statt. Nur wo notwendig, werden wir Ihnen kurze Ausblicke in die anderen Phasen angeben. Zunächst starten wir aber noch mit ein paar allgemeinen Definitionen.

1.1.1 Programmierung

Einfach gesprochen bedeutet Programmierung, einen Text zu schreiben, den ein Computer verstehen und ausführen kann. Mittels der Programmierung wird ein *Algorithmus* zu einem lauffähigen Computerprogramm, auch *Programm* oder *Software* genannt. Ein Programm ist eine maschinenlesbare und maschinenverständliche Beschreibung eines Algorithmus und der benötigten Daten. (In unserem Fall denken Sie sich für Maschine noch Rechenmaschine oder Computer; dann passt dies.)

Man unterscheidet verschiedene Arten von Programmierung, wobei wir uns in diesem Buch auf zwei große Gruppen beschränken:

- Strukturierte Programmierung (oder auch modulare, imperative oder prozedurale Programmierung genannt)
- Objektorientierte Programmierung

Algorithmus

Ein *Algorithmus* definiert ein Verfahren, bei dem aufgrund eines Systems von Regeln gegebene Größen (auch Eingabeinformationen oder Aufgaben) in andere Größen (auch Ausgabeinformationen oder Lösungen) transformiert werden. Durch Algorithmen werden komplizierte Prozesse nachgebildet, welche dann von Computern (Rechnern, Maschinen, Automaten) abgearbeitet werden können.

Software

Als *Software* bezeichnet man alle Programme und Programmteile eines Computers; diese „soften“ Teile stehen damit im Gegensatz zur „harten“ Hardware. Zur Software werden auch diejenigen Daten (Initialisierungsdaten) hinzugerechnet, die beim Start eines Programms bereits bekannt sind.

Die Software wird unterschieden in Systemsoftware, die zusammen mit der Hardware das Rechnersystem bildet, und Anwendungssoftware, die einzelne spezielle Aufgaben erledigt: Systemsoftware dient der Funktionsfähigkeit des Computers. Anwendungssoftware bezeichnet entweder Standardsoftware, welche von vielen Anwendern eingesetzt werden kann, oder

Individuallösungen, welche jeweils nur ganz spezifische Anwendungen für einen kleinen Benutzerkreis ausführen.

Strukturierte Programmierung

Unter *strukturierter Programmierung* versteht man einen Programmieransatz für die Erstellung von Software, welcher die einzelnen Teile bzw. Module eines Programms als hierarchisch geordnete Bausteine repräsentiert. In jedem Modul werden spezifische Funktionen (Prozeduren, Methoden, Operationen) ausgeführt. Im Idealfall besteht jedes Modul aus einem speziellen Algorithmus.

Da die Unterscheidung zur modularen, imperativen oder prozeduralen Programmierung fließend und nicht immer offensichtlich ist, sprechen wir im Folgenden nur von strukturierter Programmierung.

Objektorientierte Programmierung

Unter *objektorientierter Programmierung*, abgekürzt als OOP, versteht man einen Programmieransatz, welcher die einzelnen Module eines Programms in separate Einheiten – so genannte Klassen – zusammenfasst. Diese Klassen beinhalten neben den benötigten und anfallenden Daten auch die auf diese Daten anzuwendenden Operationen (Methoden, Algorithmen). Da diese Klassen im Allgemeinen nur ein generisches Muster darstellen, werden während des Programmablaufs konkrete Exemplare der Klassen – als Objekte bezeichnet – erstellt und mit spezifischen Daten gefüllt.

Auf die Vorteile der objektorientierten Programmierung, wie Datenkapselung, Vererbung und Polymorphismus, kommen wir im zweiten Drittel des Buches im Detail zu sprechen.

1.1.2 Grundlegende Programmstrukturen

Setzt man Algorithmen in eine computerverständliche Form um, so tauchen nachfolgende drei (bzw. vier) grundlegende Programmstrukturen unabhängig von einer Programmiersprache immer wieder auf:

- Anweisungen
- Verzweigungen aufgrund von Bedingungen
- Wiederholungen mit Schleifen
- (Blöcke mit Aufruf anderer Algorithmen bzw. Module)

Diese Programmstrukturen werden entsprechend den verfügbaren Elementen (Schlüsselwörtern) der Programmiersprachen im Programmcode implementiert.

Die ersten drei Programmstrukturen sind ausreichend, um (nahezu) alle Algorithmen zu beschreiben. Fasst man mehrere einzelne solche Programmstrukturen zusammen, so ergibt sich eine weitere grundlegende Programmstruktur, hier die vierte, die je nach Programmiersprache respektive Programmierartyp irgendwie umschrieben werden kann als Modul, Block, Funktion, Methode, Unterroutine oder auch Objekt. Aber eigentlich ist dieser Aufruf nichts