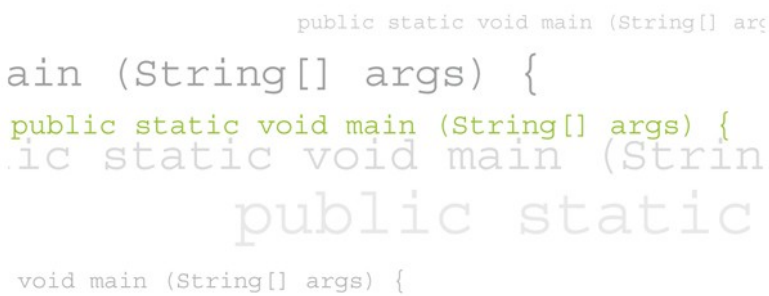
Rolf Dornberger | Rainer Telesko

Java-Training zur Objektorientierten Programmierung

Leitfaden für Lehre, Unterricht und Selbststudium



Oldenbourg

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.

Rosenheimer Straße 145, D-81671 München
Telefon: (089) 45051-0
oldenbourg.de

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 Datenbankanbindungen 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

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 objektorientierte 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 Programmiertyp irgendwie umschrieben werden kann als Modul, Block, Funktion, Methode, Unterroutine oder auch Objekt. Aber eigentlich ist dieser Aufruf nichts