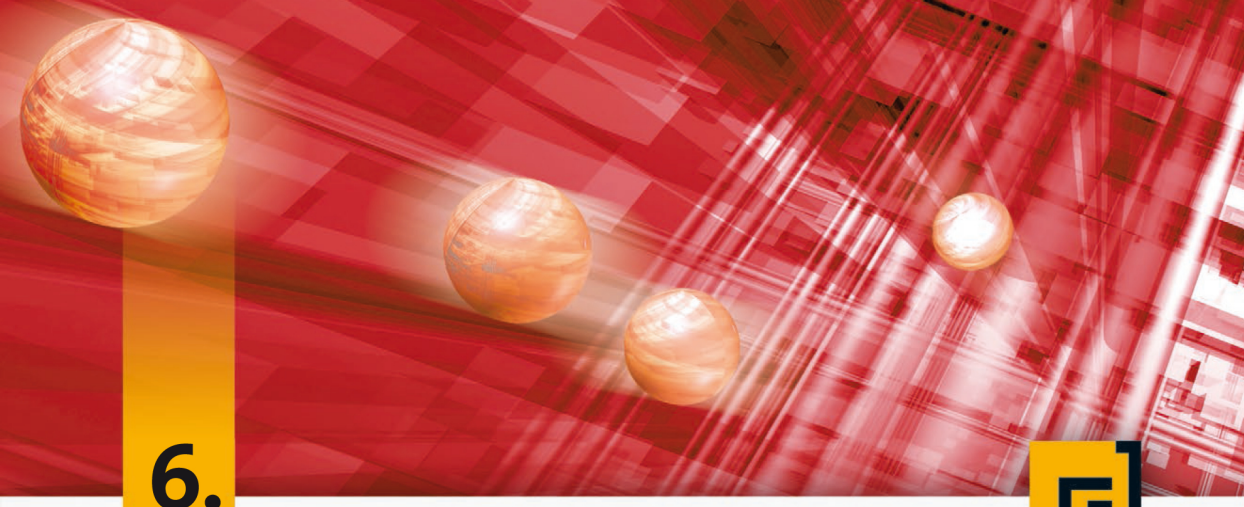

6. Auflage

Gunter Saake · Kai-Uwe Sattler

Algorithmen und Datenstrukturen

Eine Einführung mit Java

dpunkt.verlag

Gunter Saake ist Professor für Datenbanken und Informationssysteme an der Otto-von-Guericke-Universität Magdeburg und forscht unter anderem auf den Gebieten Datenbankintegration, digitale Bibliotheken, objektorientierte Informationssysteme und Informationsfusion. Er ist Koautor mehrerer Lehrbücher, u.a. zu Datenbankkonzepten und -implementierungstechniken, Datenbanken & Java.

Kai-Uwe Sattler ist Professor für Datenbanken und Informationssysteme an der TU Ilmenau. Zu seinen Arbeitsgebieten zählen Anfrageverarbeitung sowie Architekturen, Datenstrukturen und Algorithmen für das Datenmanagement auf Basis moderner Hardwaretechnologien. Er ist Koautor mehrerer Lehrbücher zu Datenbankkonzepten, -architekturen und -implementierungstechniken.

Gunter Saake · Kai-Uwe Sattler

Algorithmen und Datenstrukturen

Eine Einführung mit Java

6., überarbeitete und erweiterte Auflage

dpunkt.verlag

Prof. Dr. Gunter Saake
Institut für Technische und Betriebliche Informationssysteme
Otto-von-Guericke-Universität Magdeburg
Universitätsplatz 2, 39106 Magdeburg
E-Mail: saake@iti.cs.uni-magdeburg.de

Prof. Dr. Kai-Uwe Sattler
Fakultät für Informatik und Automatisierung
FG Datenbanken und Informationssysteme
Technische Universität Ilmenau
PF 100565, 98684 Ilmenau
E-Mail: kus@tu-ilmenau.de

Lektorat: Christa Preisendanz
Copy-Editing: Ursula Zimpfer, Herrenberg
Satz: Kai-Uwe Sattler, Ilmenau
Herstellung: Stefanie Weidner
Umschlaggestaltung: Helmut Kraus, *www.exclam.de*
Druck und Bindung: mediaprint solutions GmbH, 33100 Paderborn

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.

ISBN:
Print 978-3-86490-769-2
PDF 978-3-96910-066-0
ePub 978-3-96910-067-7
mobi 978-3-96910-068-4

6., überarbeitete und erweiterte Auflage 2021

Wieblinger Weg 17
69123 Heidelberg

Hinweis:
Dieses Buch wurde auf PEFC-zertifiziertem Papier aus nachhaltiger Waldwirtschaft gedruckt. Der Umwelt zuliebe verzichten wir zusätzlich auf die Einschweißfolie.

PEFC zertifiziert
Das Papier für dieses Buch stammt aus nachhaltig bewirtschafteten Wäldern und kontrollierten Quellen.
PEFC
PEFC/04-31-0810
www.pefc.de

Schreiben Sie uns:
Falls Sie Anregungen, Wünsche und Kommentare haben, lassen Sie es uns wissen: hallo@dpunkt.de.

5 4 3 2 1 0

Vorwort

Seit der letzten Auflage des Buches sind fast 7 Jahre vergangen – eine lange Zeit für ein schnelllebiges Gebiet wie die Informatik. So ist die Java-Umgebung mittlerweile bei Version 14 angelangt, das Java-Ökosystem ist deutlich diverser geworden und Themen wie neuronale Netze, die wir schon in der ersten Auflage des Buches als ein Paradigma behandelt haben, die aber über viele Jahre eher ein Randthema waren, sind aktuell im Kontext von künstlicher Intelligenz und »Deep Learning« in aller Munde.

Dennoch ist es unserer Meinung nach weiterhin notwendig, sich in der Informatik mit grundlegenden Themen wie Algorithmen und Datenstrukturen zu beschäftigen. Auch Java ist trotz aller Konkurrenz durch Sprachen wie Python, Scala, Kotlin, Swift oder Rust nach wie vor ein geeignetes Mittel zum Erlernen einer ersten Programmiersprache.

Daher haben wir auch mit dieser Auflage des Buches versucht, zum einen gezielt einige wichtige und interessante Datenstrukturen und Algorithmen (z.B. Skip-Listen, weitere Hashverfahren und Graphalgorithmen) aufzunehmen und zum anderen relevante Neuerungen von Java aus den letzten Jahren zu berücksichtigen. Unser Fokus liegt aber weiterhin auf Algorithmen und Datenstrukturen – die Programmiersprache Java ist nur das Werkzeug.

Mit der Überarbeitung des Buches haben wir auch die Beispielprogramme aktualisiert. Der Quellcode der Beispiele ist jetzt zeitgemäß auf GitHub unter

```
https://github.com/ksattler/algoj
```

zu finden.

Unser Dank gilt allen Leserinnen und Lesern, die durch Hinweise, Kommentare und Kritiken geholfen haben, Fehler zu korrigieren und Verbesserungen vorzunehmen. Weiterhin bedanken wir uns bei

allen, die uns unterstützt haben: unseren Familien und natürlich auch dem dpunkt.verlag.

Magdeburg und Ilmenau, September 2020
Gunter Saake und Kai-Uwe Sattler

Vorwort zur 5. Auflage

Auch mit der nunmehr 5. Auflage des Buches haben wir versucht, unserem Ziel treu zu bleiben, den Rahmen einer zweisemestrigen Einführungsvorlesung in das Thema Algorithmen, Datenstrukturen und Java nicht zu sprengen. Natürlich hat jeder Dozent seine Vorlieben für bestimmte Themen und so wird man auch weiterhin vielleicht den einen oder anderen Algorithmus oder eine ganz bestimmte Datenstruktur vermissen.

In gleicher Weise haben wir bei der Überarbeitung die mit jeder neuen Java-Version eingeführten oder angekündigten Erweiterungen eher zurückhaltend berücksichtigt. Gerade beim Erlernen des Programmierens und einer Programmiersprache ist es oft einfacher, zunächst mit einem kleinen Kern von Sprachelementen zu beginnen – die »bells and whistles« erschließen sich dann später recht schnell.

So enthält die 5. Auflage als Neuerungen »nur« einen Überblick zu den mit Java 8 eingeführten Lambda-Ausdrücken, die eine schöne Anwendung des applikativen (funktionalen) Paradigmas darstellen, sowie neue Beispiele, die aus dem Einsatz des Materials in einigen Einführungsvorlesungen entstanden sind. Natürlich haben wir ebenfalls versucht, Feedback und Fehlerkorrekturen zu berücksichtigen, und möchten uns dafür bei unseren Lesern – ganz speziell bei Niklas Peter und Jan Sellner – bedanken.

Magdeburg und Ilmenau, Oktober 2013
Gunter Saake und Kai-Uwe Sattler

Vorwort zur 4. Auflage

Nach fast 5 Jahren Bestand der 3. Auflage war es an der Zeit, wieder einmal eine Überarbeitung vorzunehmen. Auch wenn die relevanten Änderungen an der Programmiersprache Java eher marginal sind – die Java-Plattform hat sich dagegen sehr viel weiter entwickelt, aber das ist für ein Buch dieser Art weniger von Bedeutung –, haben wir einige Hinweise und eigene Lehrerfahrungen in-

tegriert. Unser Anliegen bleibt jedoch weiterhin ein Begleitbuch für Erst- und Zweitsemester in Informatik-lastigen Studiengängen: Weder wollten wir den Umfang durch Aufnahme einer Vielzahl weiterer Algorithmen und Datenstrukturen sprengen noch im Interesse von Überblicksvorlesungen oder Programmierkursen abspecken. Wir haben in dieser Auflage als neue Algorithmen den für die Routenplanung wichtigen A*-Algorithmus und die Levenshtein-Distanz zum Ähnlichkeitsvergleich von Texten aufgenommen. Weiterhin ist für den B-Baum nun auch eine einfache Beispielimplementierung angegeben. Auf die Neuerungen der seit der 3. Auflage eingeführten Sprachversion 6.0 sowie der für Ende 2010 geplanten Version Java SE 7.0 wird an geeigneter Stelle eingegangen.

Schließlich möchten wir allen Lesern (Studierenden wie Kollegen) danken, die uns wertvolles Feedback geliefert haben.

Magdeburg und Ilmenau, April 2010
Gunter Saake und Kai-Uwe Sattler

Vorwort zur 3. Auflage

Die Nachfrage nach diesem Buch, einige in der 2. Auflage übersehene Fehler und nicht zuletzt die Weiterentwicklung der Sprache Java haben die 3. Auflage früher als erwartet notwendig gemacht. Somit beziehen sich die Neuerungen dieser Auflage im Wesentlichen auf die Vorstellung der neuen Sprachkonzepte von Java in der Version 5.0, die gerade im Zusammenhang mit Datenstrukturen wie Feldern oder Listen von Bedeutung sind. Weiterhin haben wir uns bemüht, alle gemeldeten Fehler zu korrigieren. Für entsprechende Hinweise von aufmerksamen Lesern möchten wir uns an dieser Stelle ausdrücklich bedanken.

Magdeburg und Ilmenau, November 2005
Gunter Saake und Kai-Uwe Sattler

Vorwort zur 2. Auflage

Zur 1. Auflage dieses Buches haben wir eine Vielzahl von Rückmeldungen von Dozenten und Studierenden an Universitäten und Fachhochschulen, aber auch von Informatik-Lehrern an Gymnasien erhalten. Neben Lob, Kritik und Hinweisen auf einige Fehler befanden sich darunter auch einige Wünsche nach der Behandlung von Algorithmen und Datenstrukturen, die im Buch bisher fehlten. Daher haben

wir uns entschlossen, für die vorliegende 2. Auflage nicht nur Fehlerkorrekturen vorzunehmen, sondern auch einige Ergänzungen aufzunehmen. So werden nun mit den genetischen Algorithmen und den neuronalen Netzen zwei weitere »Algorithmenparadigmen« vorgestellt. Weitere Neuerungen betreffen die Aufnahme von Rot-Schwarz-Bäumen, die gern als Alternative zu AVL-Bäumen behandelt werden, sowie praktische Realisierungen von Tries und erweiterbaren Hashverfahren. Dabei haben wir jedoch versucht, dem ursprünglichen Anliegen des Buches als ein Begleitwerk zu einer einführenden »Algorithmen & Datenstrukturen«-Vorlesung für Informatik-Studiengänge an Universitäten und Fachhochschulen treu zu bleiben. Dies bedeutet für uns eine gesunde Mischung aus Theorie und Praxis, wobei jedoch Themen, die normalerweise im weiteren Verlauf des Studiums noch vertiefend behandelt werden (z.B. Theoretische Informatik, Komplexitätstheorie, Objektorientierte Programmierung oder alternative Algorithmenkonzepte), nur soweit angesprochen werden, wie es für das grundlegende Verständnis von Zusammenhängen des Stoffes notwendig ist.

Abschließend gilt unser Dank speziell Ilona Blümel, Christian Borgelt, Martin Dietzfelbinger, Horst-Michael Groß und Dominik Gruntz sowie allen Lesern der 1. Auflage, die mit ihren Hinweisen und Kommentaren zur Verbesserung und somit zu der vorliegenden 2. Auflage beigetragen haben.

Magdeburg und Ilmenau, März 2004
Gunter Saake und Kai-Uwe Sattler

Vorwort zur 1. Auflage

Genese des Buches

Das vorliegende Buch entstand aus den Begleitmaterialien einer Vorlesung »Einführung in Algorithmen und Datenstrukturen«, die die Autoren an der Universität Magdeburg im Vorlesungszyklus 1999/2000 für die Studienanfänger in den Diplomstudiengängen Informatik, Wirtschaftsinformatik und Computervisualistik neu konzipierten, da erstmals diese Grundvorlesung mit praktischen Übungen in der Programmiersprache Java angeboten wurde. Neben dem in dem Buch aufbereiteten Stoff wurden Einschübe z.B. zur Realisierung relationaler Datenbanken in der Vorlesung integriert, die zur Verdeutlichung der vermittelten Techniken anhand realer Problemstellungen dienten. Diese Einschübe dürften bei anderer Gelegenheit jeweils durch Einschübe aus dem konkreten Arbeitsgebiet der Vor-

lesenden gewählt werden, so dass sie in diesem Buch weggelassen wurden.

Zielgruppe des Buches

Die Zielgruppe dieses Buches sind somit insbesondere Studierende in universitären Grundstudiumsvorlesungen, die einen Umfang von bis zu acht Semesterwochenstunden haben und eine Einführung in die Grundkonzepte der praktischen Informatik, begleitet durch praktische Übungen in Java, geben sollen, um das Fundament für die vertiefende Behandlung der verschiedenen Teilgebiete der praktischen Informatik zu bilden. Dabei wird davon ausgegangen, dass die mathematischen Grundlagen sowie die Konzepte der theoretischen Informatik und insbesondere der technischen Informatik in parallelen oder anschließenden separaten Vorlesungen behandelt werden.

Um den Studierenden den Zugang zu erleichtern, wurde, wenn immer es möglich und sinnvoll erschien, auf etablierte Notationen und Beispiele (etwa dem Schülerduden entnommen) zurückgegriffen.

Inhalt des Buches

Der Inhalt des Buches orientiert sich an den Inhalten vergleichbarer Studienangebote an deutschen Universitäten und den bekannten Empfehlungen zu Grundstudiumsangeboten der genannten Studiengänge. Als Besonderheiten sind zu nennen:

- Einige der behandelten theoretischen Grundlagen (abstrakte Maschinenmodelle, Berechenbarkeit, Halteproblem, Algorithmenparadigmen) kommen unseren Recherchen nach oft in Programmiersprachen-gestützten Kursen zu kurz. Diese wurden bewusst aufgenommen, um durch Verknüpfung dieser Themen mit konkreter Programmierung in Java (etwa die Simulation einer Registermaschine) den Studierenden die Vernetzung dieser abstrakten Konzepte zu ermöglichen.
- Entgegen anderer Vorlesungszyklen wurde die Behandlung von parallelen und verteilten Abläufen bewusst in den dem ersten Semester zugeordneten Vorlesungsteil aufgenommen.
- Die Behandlung des üblichen Kanons von Basisdatenstrukturen wurde um einige, in der Praxis wichtige Verfahren und Algorithmen (spezielle Suchbäume, Graphenalgorithmen) erweitert.

Besonderheiten

Wenn man dieses Buch mit anderen Büchern für Grundlagenvorlesungen »Algorithmen und Datenstrukturen« vergleicht, erscheint es auf den ersten Blick widersprüchlich: Einerseits beinhaltet es eine ganze Reihe von Grundlagenthemen, die sich nicht stark von entsprechenden Materialien von vor 20 Jahren unterscheiden, andererseits wird mit den Abschnitten über Java-Programmierung eine der modernsten Programmiersprachen zur Illustration der Konzepte ge-

nutzt. Dieser Widerspruch ist Methode: Die Autoren wollen hiermit verdeutlichen, dass die Informatik die Reife einer Wissenschaftsdisziplin mit etablierten methodischen und theoretischen Grundlagen erlangt hat und auf einem reichen Schatz an gefestigtem Basiswissen beruht, und dieses mit dem (zum Teil spielerischen, zum Teil ernsthaften) Umgang mit Methoden und Sprachen moderner Softwareerstellung verbinden.

Das Lehrziel des Buches fußt dabei auf beiden Aspekten: Studierende sollen eine Grundlage für die theoretischen und praktischen Vertiefungen eines intensiven Hauptstudiums bekommen und diese Grundkenntnisse direkt umsetzen können in den »praktischen Alltag« des Arbeitens mit Programmen, Spezifikationen und Modellierungen. Das vorliegende Buch hat weder den Anspruch eines Basiswerkes über die Theorie der Algorithmen und Datenstrukturen, noch ist es eine reine Einführung in die Programmierung mit Java.

Einsatz des Buches

Das Buch ist in drei Teile aufgeteilt, wobei die ersten beiden Teile den Stoff des ersten Semesters abdecken. Der dritte Teil, ergänzt um spezifische Inhalte wie oben erläutert, bildet den Stoff eines dem Thema »Datenstrukturen« gewidmeten zweiten Semesters. Beide Vorlesungen sollten durch Veranstaltungen zur Einführung in die Programmiersprache Java begleitet werden, wobei der Stoff eine schrittweise Einführung über die Stufen »Java als imperative Programmiersprache«, »Funktionen und Rekursion in Java«, »Objektorientierung: Klassen und Methoden« und abschließend »Methoden des Software Engineering in Java« nahe legt. Im Laufe des zweiten Semesters sollte eine über eine längere Zeit zu bearbeitende größere Programmieraufgabe, eventuell bereits in Kleingruppen, gelöst werden oder (wie in unserer Veranstaltung) in Form eines Programmierwettbewerbs die Studierenden zur kreativen Nutzung des erarbeiteten Wissens animiert werden.

Einführung in Java

Die Trennung von Algorithmen und Datenstrukturen erscheint im Zeitalter von Objektorientierung auf den ersten Blick vielleicht anachronistisch. Erfahrungen der Autoren haben aber gezeigt, dass ein Zugang zu dieser Thematik gerade Studienanfängern leichter fällt, wenn der Fokus zunächst auf funktionale und imperative Konzepte zur Formulierung und Implementierung von Algorithmen gelegt wird und die (objektorientierten) Eigenschaften der Programmiersprache nur so weit wie notwendig vorgestellt werden. Probleme wie Suchen oder Sortieren lassen sich am einfachsten ohne den »Ballast« von Klassen oder Objekten erfassen. Das Verständnis für Objektorientierung ergibt sich später mit der Einführung von abstrakten Datentypen und in der praktischen Arbeit mit der Java-Klassenbibliothek. Nicht vergessen sollte man dabei auch, dass Ob-

Funktionale und imperative Konzepte

Objektorientierung

jektorientierung nur *ein* Paradigma neben anderen (z.B. funktional) ist.

Der Buchstoff kann (und sollte) durch animierte Algorithmen und Datenstrukturen ergänzt und insbesondere in den Übungen durch »best practice«-Programmfragmente (und deren abschreckende Gegenstücke) vertieft werden. Auf der Webseite diese Buches findet sich ein Vorrat derartiger Ergänzungen, der laufend erweitert werden soll:

`www.dpunkt.de/buch/alg_dat.html`

Dort wird auch Folienmaterial zur Verfügung gestellt.

Danksagungen

Ein Buch über derartig grundlegende Themen basiert natürlich auf den Vorarbeiten und der Unterstützung einer ganzen Reihe von Personen, von denen wir hier einigen besonders danken wollen.

Die Notationen und Beispiele im Kapitel Registermaschinen sind an die Ausführungen in einem Skript von Jürgen Dassow angelehnt.

Viele Beispiele und Notationen betreffend Grundlagen und Paradigmen von Algorithmen sind durch Vorlesungsunterlagen von Hans-Dieter Ehrich beeinflusst. Dies erfolgte direkt und indirekt über Materialien von Rudolph Kruse und Gunter Saake, die ihrerseits auf den ursprünglichen Materialien von Ehrich aufbauten.

Weiterhin wollen wir allen danken, die durch Hinweise oder Korrekturlesen des Manuskriptes zum Gelingen des Buches beigetragen haben. Hier sind zuerst unsere Kollegen aus der Arbeitsgruppe Datenbanken Sören Balko, Oliver Dunemann, Martin Endig, Ingolf Geist, Hagen Höpfner, Eike Schallehn, Ingo Schmitt und Nadine Schulz zu nennen sowie alle Übungsleiter zur Vorlesung »Algorithmen & Datenstrukturen« 1999/2000, wobei wir stellvertretend Ilona Blümel hervorheben möchten, die uns mit zahlreichen Beispielen und Hinweisen unterstützt hat. Last but not least danken wir den Studierenden, die die erste Version des Skriptes im Rahmen der Vorlesung kritisch begleitet haben.

Unser Dank geht auch an unsere Lektorin Christa Preisendanz vom dpunkt.verlag, die uns anfangs zu diesem Projekt ermutigt und später mit Geduld begleitet hat, sowie an Ursula Zimpfer für die vielen Korrekturhinweise.

Der Dank von Gunter Saake gilt denjenigen aus seiner Familie und dem Bekanntenkreis, die auch bei diesem Buchprojekt in unterschiedlichem Grade unter der Bucherstellung zu leiden hatten.

Kai-Uwe Sattler dankt seinem Sohn Bennett, der aufmerksam darüber gewacht hat, dass Papas »Klackern« nicht zulasten so wichtiger Dinge wie Legobauen und Rollerfahren ging, und natürlich seiner Frau Britta, die die »Nur-noch«-Ausreden (Nur noch dieses Kapitel!, Nur noch diese Woche! etc.) dieses Mal noch länger ertragen musste und dennoch für den notwendigen Rückhalt gesorgt hat, ohne den ein Buchprojekt wohl nicht möglich wäre. Er dankt weiterhin seinen Eltern für das Verständnis, wenn gerade in der Endphase der Buchherstellung die wenigen Besuche auch noch durch das mitgebrachte Notebook gestört wurden. Ein abschließender Dank gilt Fred Kreutzmann und Steffen Thorhauer, die im Kaffeekochen inzwischen nicht nur uneinholbar vorn liegen, sondern auch ihren oftmals gestressten Bürokollegen mit Geduld ertragen haben.

Magdeburg, August 2001
Gunter Saake und Kai-Uwe Sattler

Inhaltsverzeichnis

Teil I

Grundlegende Konzepte

1 Vorbemerkungen und Überblick

Im beginnenden neuen Jahrtausend ist es eigentlich nicht mehr notwendig, Begriffe wie Computer, Programm oder Software einzuführen. Wir werden in diesem Kapitel trotzdem einige Vorbemerkungen machen, um den Kontext dieses Buches und der behandelten Begriffe zu verdeutlichen.

1.1 Informatik, Algorithmen und Datenstrukturen

Informatik

Informatik ist ein Kunstwort aus den 60er Jahren, das die Assoziationen Informatik gleich Information oder Technik *oder* Informatik gleich Information und Mathematik erwecken sollte. Bei der Begriffsbildung sollte durchaus bewusst ein Gegensatz zum amerikanischen Begriff *Computer Science* aufgebaut werden, um zu verdeutlichen, dass die Wissenschaft Informatik nicht nur auf Computer beschränkt ist. Informatik als Begriff ist insbesondere im nicht englischsprachigen europäischen Raum gebräuchlich. Die Informatik hat zentral zu tun mit

- systematischer Verarbeitung von Informationen und
- Maschinen, die diese Verarbeitung automatisch leisten.

Wichtige Grundkonzepte der Informatik können in einer maschinenunabhängigen Darstellung vermittelt werden. Der Bezug zu den Themen diese Buches kann durch die folgende Aussage hergestellt werden:

Algorithmen und Daten

> Die »systematische Verarbeitung« wird durch den Begriff *Algorithmus* präzisiert, Information durch den Begriff *Daten*.

In einer ersten Näherung kann man das Konzept des Algorithmus wie folgt charakterisieren:

> Ein *Algorithmus* ist eine eindeutige Beschreibung eines in mehreren Schritten durchgeführten (Bearbeitungs-)Vorganges.

In der Informatik werden nun speziell *Berechnungsvorgänge* statt allgemeiner Bearbeitungsvorgänge betrachtet, wobei der Schwerpunkt auf der *Ausführbarkeit* durch (abstrakte) Maschinen liegt, die auch als Prozessoren bezeichnet werden:

Prozessor

> Ein *Prozessor* führt einen Prozess (Arbeitsvorgang) auf Basis einer eindeutig interpretierbaren Beschreibung (dem Algorithmus) aus.

In diesem Buch werden eine Reihe von Fragestellungen behandelt, die Aspekte des Umgangs mit Algorithmen betreffen. Die verschiedenen *Notationen* für die Beschreibung von Algorithmen führen direkt zu Sprachkonzepten moderner Programmiersprachen.

Ausdrucksfähigkeit verschiedener Notationen

Wenn verschiedene Notationen verwendet werden können, stellt sich die Frage der *Ausdrucksfähigkeit* dieser Algorithmensprachen. Man kann sich diese Fragestellungen daran verdeutlichen, dass man sich überlegt, wie ausdrucksfähig die Bienensprache, die ja konkrete Wegbeschreibungen ermöglicht, im Vergleich zu einer Programmiersprache zur Wegfindungsprogrammierung von Robotern ist, oder ob dressierte Hunde tatsächlich dieselbe Sprache verstehen wie Menschen.

Korrektheit und Zeitbedarf

Spätestens beim Übergang zu konkreten Programmen in einer Programmiersprache müssen natürlich die Fragestellungen der *Korrektheit* und des *Zeitbedarfs* bzw. der Geschwindigkeit von Programmen betrachtet werden.

In diesem Buch geht es primär um Algorithmen für Rechner, also schnelle (aber dumme) Prozessoren (ein Rechner ist im Vergleich zu einem Menschen ein »Hochgeschwindigkeitstrottel«, der falsche Anweisungen prinzipiell nicht erkennen kann, aber sehr schnell als Programm vorgegebene Anweisungen ausführen kann). Hierzu werden mathematisch formale Grundlagen von Algorithmen eingeführt – ein Rechner »versteht« nur Bits, so dass jede Anweisungssprache auf diese einfache Ebene heruntergebrochen werden muss. Die Umsetzung von menschenverständlichen Algorithmen in eine maschinennahe Darstellung geht natürlich nur, wenn die Bedeutung beider Darstellungen exakt formal festgelegt ist.

Datenstrukturen

Auch wenn sich beim Programmieren scheinbar alles um die Algorithmen dreht, ist das Gegenstück, die *Datenstrukturen*, ebenfalls ein zentraler Aspekt der Grundlagen der Informatik. Erst rechnerverarbeitbare Darstellungen von Informationen erlauben das Program-

mieren realistischer Probleme auf einer angemessenen Abstraktionsebene – die Zeit, in der ein Programmierer sich an den Rechner anpassen und direkt in Bits und Bytes denken musste, ist zumindest in der nicht stark hardwarenahen Programmierung vorbei.

1.2 Historischer Überblick: Algorithmen

Die in diesem Buch behandelten Konzepte der Informatik sind älter als die Geschichte der Computer, die Programme ausführen können. Die Informatik behandelt Konzepte, die auch ohne existierende Computer gültig sind – aber zugegebenermaßen erst durch den Siegeszug der Computer die heutige praktische Relevanz erlangten.

In diesem Abschnitt wollen wir kurz einige Ergebnisse aus der Zeit *vor* dem Bau des ersten Computers nennen, um dies zu verdeutlichen:

300 v. Chr.: Euklids Algorithmus zur Bestimmung des `ggT` (beschrieben im 7. Buch der Elemente), also des größten gemeinsamen Teilers, mit dem etwa

$$\texttt{ggT}(300, 200) = 100$$

sehr effizient berechnet werden kann, ist die erste Beschreibung eines Verfahrens, das modernen Kriterien für Algorithmen gerecht wird.

800 n. Chr.: Der persisch-arabische Mathematiker Muhammed ibn Musa abu Djafar alChoresmi (oft auch als »al Chworesmi« oder »al Charismi« geschrieben) veröffentlicht eine Aufgabensammlung für Kaufleute und Testamentsvollstrecker, die später ins Lateinische als *Liber Algorithmi* übersetzt wird.
Das Wort Algorithmus ist ein Kunstwort aus dem Namen dieses Mathematikers und dem griechischen »arithmos« für Zahl.

1574: Adam Rieses Rechenbuch verbreitet mathematische Algorithmen in Deutschland.

1614: Die ersten Logarithmentafeln werden algorithmisch berechnet – ohne Computer dauerte dies 30 Jahre!

1703: Binäre Zahlensysteme werden von Leibnitz eingeführt. Diese Zahlensysteme bilden die Grundlage der internen Verarbeitung in modernen Computern.

1815: Augusta Ada Lovelace, die erste »Computer-Pionierin«, wird geboren. Sie entwickelt schon früh Konstruktionspläne für verschiedenartige Maschinen, wird Assistentin von Babbage und entwirft Programme für dessen erste Rechenmaschinen.

1822: Charles Babbage entwickelt die sogenannte *Difference Engine*, die in einer verbesserten Version 1833 fertiggestellt wird. Später entwickelt er auch die *Analytical Engine*, die bereits die wichtigsten Komponenten eines Computers umfasst, aber niemals vollendet wird.

1931: Gödels Unvollständigkeitssatz beendet den Traum vieler damaliger Mathematiker, die gesamte Beweisführung aller Sätze in der Mathematik mit algorithmisch konstruierten Beweisen durchzuführen.

1936: Die Church'sche These vereinheitlicht die Welt der Sprachen zur Notation von Algorithmen, indem für viele der damaligen Notationen die gleiche Ausdrucksfähigkeit postuliert wird.

danach: Mit der Realisierung der ersten Computer erfolgte natürlich der Ausbau der Algorithmentheorie zu einem eigenen Wissenschaftsgebiet.

Auch für den Bereich der Datenstrukturen können lang zurückreichende Wurzeln gefunden werden, etwa das Indexierungssystem historischer Bibliotheken.

Ein historischer Überblick, der auch die technische Informatik, die Programmierung und Aspekte des Software Engineering behandelt, würde den Rahmen des vorliegenden Buches sprengen. So werden wir nur noch im folgenden Abschnitt, der die Wahl der Sprache Java als Ausbildungssprache motivieren soll, die Historie der Programmiersprachen kurz anreißen.

1.3 Historie von Programmiersprachen und Java

Bevor wir näher auf die in diesem Buch verwendete Programmiersprache Java eingehen, lohnt es sich, einen Blick in die Geschichte zu werfen. Die Ursprünge höherer Programmiersprachen reichen zurück an den Anfang der fünfziger Jahre, als erste Sprachen wie *Autocode* vorgeschlagen wurden, die bereits arithmetische Ausdrücke, Schleifen, bedingte Sprünge und Prozeduren umfassten. Daraus wurde dann *Fortran* (FORmula TRANslator) entwickelt, deren erste Fassung 1954 vorgestellt wurde und die auch heute noch speziell im wissenschaftlich-technischen Bereich zum Einsatz kommt. Anfang der sechziger Jahre wurden dann *Algol* (ALGOrithmic Language), eine Sprache, die u.a. Pascal nachhaltig beeinflusst hat, *Lisp*, als die Grundlage aller funktionalen Programmiersprachen und im Bereich der künstlichen Intelligenz immer noch häufig eingesetzt,

Autocode

Fortran

Algol

Lisp

und die vielleicht noch am weitesten verbreitete Programmiersprache *COBOL* (COmmon Business Oriented Language) entwickelt. Auch die Entwicklung von *BASIC* geht auf die Mitte der sechziger Jahre zurück.

COBOL

Die Methode der strukturierten Programmierung wurde wesentlich durch *Pascal* begründet, eine Entwicklung von Niklaus Wirth zu Beginn der siebziger Jahre. Mit Pascal wurden nicht nur Sprachmittel eingeführt, die heute in allen modernen Programmiersprachen zu finden sind, sondern es wurde auch erstmals die Programmausführung durch die Interpretation von Zwischencode (sogenannten *P-Code*) verwirklicht. Dieses Verfahren wurde dann beispielsweise für Java »wiederentdeckt« und hat wesentlich zur Verbreitung der Sprache beigetragen.

Pascal

Ebenfalls Anfang der siebziger Jahre wurde ausgehend von Sprachen wie *CPL* und *BCPL* die Programmiersprache *C* entwickelt und zur Implementierung des Betriebssystems UNIX eingesetzt. Damit wurde erstmals eine höhere Programmiersprache für die Betriebssystementwicklung verwendet. Der sich daraus ergebenden weitgehenden Maschinenunabhängigkeit ist es letztendlich zu verdanken, dass UNIX und die Vielzahl der Derivate (wie u.a. Linux) heute auf nahezu allen Hardwareplattformen zu finden sind.

C

Die Idee modularer Programmiersprachen wurde ab Mitte der siebziger Jahre wiederum von Wirth mit *Modula* entwickelt und speziell auch in *Ada* umgesetzt.

Modula

Objektorientierte Programmiersprachen wie Smalltalk, C++ oder Java haben ihren Ursprung in *Simula-67*, die um 1967 in Norwegen als eine Sprache für die ereignisorientierte Simulation entstanden ist. Das Potenzial objektorientierter Programmierung wurde zu dieser Zeit jedoch noch nicht wirklich erkannt. Erst mit der Entwicklung von *Smalltalk* durch Xerox PARC Ende der siebziger Jahre fanden Konzepte wie »Klasse« oder «Vererbung« Einzug in die Programmierung. Ausgehend vom objektorientierten Paradigma einerseits und der Systemprogrammiersprache C andererseits wurde 1983 bei AT&T die Programmiersprache C++ entworfen, die nicht zuletzt wegen der »Abwärtskompatibilität« zu C schnell eine weite Verbreitung fand.

Simula-67

Smalltalk

C++

Als Entwicklungen der letzten Jahre vor Java sind insbesondere *Eiffel* von Bertrand Meyer mit dem neuen Konzept der Zusicherungen bzw. des vertraglichen Entwurfs (*design by contract*), *Oberon* von Wirth mit dem Ziel einer Rückbesinnung auf einen einfachen, kompakten und klaren Sprachentwurf sowie die Vielzahl von Skriptsprachen wie *Perl*, *Tcl* oder *Python* zu nennen.

Eiffel

Die Arbeiten zu Java lassen sich bis in das Jahr 1990 zurückverfolgen, als bei Sun Microsystems eine Sprache für den Consumer-

Java

Electronics-Bereich unter dem Namen *Oak* entwickelt werden sollte. Entwurfsziele dieser Sprache waren bereits Plattformunabhängigkeit durch Verwendung von Zwischencode und dessen interpretative Ausführung, Objektorientierung und die Anlehnung an C/C++, um so den Lernaufwand für Kenner dieser Sprachen gering zu halten. Mit der Entwicklung des World Wide Web um 1993 fand im Entwicklerteam eine Umorientierung auf Webanwendungen und damit eine Umbenennung in *Java* – die bevorzugte Kaffeesorte der Entwickler – statt.

HotJava

1995 wurde die Sprache dann zusammen mit einer Referenzimplementierung und dem *HotJava*-Browser der Öffentlichkeit vorgestellt. Dieser Browser ermöglichte erstmals aktive Webinhalte – durch sogenannte Applets. Hinter diesem Begriff verbergen sich kleine Java-Programme, die als Teil eines Webdokumentes von einem Server geladen und in einem geschützten Bereich des Browsers ausgeführt werden können. Da der HotJava-Browser selbst in Java geschrieben wurde, war dies gleichzeitig die Demonstration der Eignung von Java für größere Projekte. Als Netscape – damals unbestrittener Marktführer bei Webbrowsern – kurz darauf die Unterstützung von Java-Applets in der nächsten Version des Navigators bekannt gab, fand Java in kürzester Zeit eine enorme Verbreitung. Dies wurde auch noch dadurch verstärkt, dass die Entwicklungsumgebung für Java von Sun kostenlos über das Web zur Verfügung gestellt wurde. Im Jahre 1997 wurde dann die Version 1.1 des *Java Development Kit (JDK)* freigegeben und alle großen Softwarefirmen wie IBM, Oracle oder Microsoft erklärten ihre Unterstützung der Java-Plattform. Mit der Veröffentlichung von Java 2 im Jahr 1998 und der Folgeversionen des JDK hat sich dann nicht nur die Anzahl der Systemplattformen, auf denen Java-Programme lauffähig sind, vergrößert, sondern insbesondere auch die Zahl der Klassenbibliotheken, die für Java verfügbar sind. In der im Sommer 2005 freigegebenen Version *J2SE 5.0* (Java 2 Platform Standard Edition) wurden schließlich auch einige neue Sprachkonzepte wie Generics und variable Parameterlisten eingeführt, auf die wir an geeigneter Stelle eingehen werden. Die Nachfolgeversion *Java SE 6.0* erschien im Dezember 2006 und hat seitdem einige Updates erfahren. Wesentliche Änderungen an der Sprache selbst bietet diese Version jedoch nicht. Seit 2006 ist Java auch als Open-Source-Software – das OpenJDK – verfügbar. 2009/2010 wurde die Firma Sun Microsystems, und damit auch die Java-Technologien, vom Datenbankhersteller Oracle gekauft. Die Weiterentwicklung von Java, die im sogenannten *Java Community Process (JCP)* stattfindet, wird nun also von Oracle geleitet. Die nächste Version *Java SE 7.0* wurde im Juli 2011 veröffentlicht und enthält neben API-Erweiterungen nur ei-

Netscape

Java Development Kit

J2SE 5.0

Java SE 6.0

Java SE 7.0

nige kleinere Spracherweiterungen. 2014 wurde *Java SE 8* veröffentlicht. In dieser Version wurde mit den *Lambda*-Ausdrücken ein sehr interessantes Konzept aus dem Bereich funktionaler Programmiersprachen in die Sprache aufgenommen, das wir ebenfalls kurz vorstellen werden.

Java SE 8

Nach einer dreijährigen Pause wurde 2017 die Version *Java SE 9* freigegeben, mit der u.a. die Java-Shell eingeführt wurde. Die aktuelle Version ist die im März 2020 veröffentlichte Version *Java SE 14*.

Java SE 9

Java SE 14

Geblieben sind die ursprünglichen Ansprüche: eine leicht erlernbare, klare und kompakte Sprache, die die Komplexität beispielsweise von C++ vermeidet, aber dennoch das objektorientierte Paradigma umsetzt. Gleichzeitig ist Java architekturneutral, d.h., durch Verwendung eines plattformunabhängigen Zwischencodes sind Programme auch in kompilierter Form portabel – eine Eigenschaft, die für eine moderne Internet-Programmiersprache fundamental ist. Über die Sprache selbst hinaus spielt die Java Virtual Machine (JVM) eine wichtige Rolle als Plattform für andere Sprachen. Neben Umsetzungen anderer Sprachen auf die JVM wie JRuby oder JPython zählen hierzu insbesondere neue Sprachen wie Scala, Groovy, Clojure oder Kotlin.

1.4 Grundkonzepte der Programmierung in Java

Ein fundamentaler Begriff beim Programmieren – eigentlich beim Umgang mit Computern allgemein – ist das *Programm*. Egal ob man ein Textverarbeitungssystem oder ein kleines, selbst geschriebenes Java-Programm zur Berechnung der Fakultät einer Zahl betrachtet – in beiden Fällen handelt es sich um ein Programm (oder eine Sammlung davon).

Programm

Ein Programm ist die Formulierung eines Algorithmus und der zugehörigen Datenstrukturen in einer Programmiersprache. □

Definition 1.1 *Programm*

Die Grundbausteine von Programmen in höheren Programmiersprachen werden wir erst in den nächsten Kapiteln kennen lernen. Intuitiv wissen wir aber, dass zumindest Anweisungen wie elementare Operationen, bedingte Anweisungen und Schleifen, Datentypen zur Festlegung der Struktur der Daten und darauf definierte Operationen sowie die Möglichkeit der Formulierung von Ausdrücken benötigt werden.

Die syntaktisch korrekte Kombination dieser Bausteine allein führt jedoch noch nicht zu einem ausführbaren Programm. Da Programme meist in einer für Menschen schreib- und lesbaren Form erstellt werden, sind sie für Computer, die nur einen von der jeweiligen Prozessorarchitektur (etwa Intels x86-Familie, die SPARC-Prozessoren von Sun oder die PowerPC-Prozessorfamilie der Macs bzw. IBM-Rechner) abhängigen Maschinencode »verstehen«, nicht direkt ausführbar. Es ist daher eine Übersetzung oder auch Kompilierung (engl. compile) des Programms aus der Programmiersprache in den Maschinencode notwendig. Die Übersetzung erfolgt durch ein spezielles Programm, das als Compiler bezeichnet wird. Neben der Übersetzung in Maschinencode führt ein solcher Compiler auch noch verschiedene Überprüfungen des Programms durch, z.B. ob das Programm hinsichtlich der Syntax korrekt ist, ob benutzte Variablen und Prozeduren deklariert sind u.Ä. Programmiersprachen wie C, C++ oder (Turbo-)Pascal basieren auf diesem Prinzip.

Compiler

Da jede Änderung am Programmtext eine Neukompilierung erfordert, verzichtet man bei einigen Sprachen auf die Übersetzung, indem die Programme interpretiert werden. Das bedeutet, dass ein spezielles Programm – der sogenannte Interpreter – den Programmtext schrittweise analysiert und ausführt. Der Interpreter muss also Syntax und Semantik der Sprache kennen! Der Vorteil dieses Ansatzes ist, dass die Kompilierung entfällt und dadurch Änderungen vereinfacht werden. Auch ist das Programm (man spricht häufig auch von Skripten) unabhängig von einer konkreten Prozessorarchitektur. Diese Vorteile werden durch die Notwendigkeit eines Interpreters und eine durch die interpretierte Ausführung verschlechterte Performance erkauft. Beispiele für Sprachen, die mit diesem Prinzip realisiert werden, sind insbesondere die im Webumfeld beliebten Skriptsprachen wie JavaScript.

Interpreter

Skriptsprachen

Java basiert auf einem kombinierten Ansatz (Abbildung 1–1). Programme werden zwar kompiliert, jedoch nicht in einen prozessorspezifischen Maschinencode, sondern in einen sogenannten Bytecode. Hierbei handelt sich um Code für eine abstrakte Stack-Maschine. Dieser Bytecode wird schließlich von einem Java-Interpreter, der virtuellen Java-Maschine (*Java VM* – engl. Virtual Machine), ausgeführt. Da der Bytecode relativ maschinennah ist, bleibt die Interpretation und Ausführung einfach, wodurch der Performance-Verlust gering gehalten werden kann. Andererseits lassen sich auf diese Weise auch übersetzte Java-Programme auf unterschiedlichen Prozessorarchitekturen ausführen, sofern dort eine Java-VM verfügbar ist.

Bytecode

Aufbau von Java-Programmen

Wie ist nun ein Java-Programm aufgebaut? Ein solches Programm besteht aus einer oder mehreren Klassen, die damit das Hauptstrukturierungsmittel in Java darstellen. Wir werden auf den Begriff der