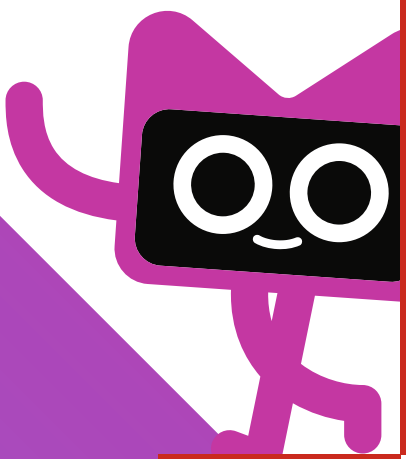
PROGRAMMIEREN LERNEN MIT

Kotlin

Grundlagen, Objektorientierung und fortgeschrittene Konzepte

2. Auflage

Christian **KOHLS**
Alexander **DOBRYNIN**



Zusatzmaterial unter
plus.hanser-fachbuch.de

HANSER

Kohls/Dobrynin

Programmieren lernen mit Kotlin

Ihr Plus – digitale Zusatzinhalte!

Auf unserem Download-Portal finden Sie zu diesem Titel kostenloses Zusatzmaterial.

Geben Sie auf **plus.hanser-fachbuch.de** einfach diesen Code ein:

plus-rn34m-tL9pr

Bleiben Sie auf dem Laufenden!

Unser **Computerbuch-Newsletter** informiert Sie monatlich über neue Bücher und Termine. Profitieren Sie auch von Gewinnspielen und exklusiven Leseproben. Gleich anmelden unter:

www.hanser-fachbuch.de/newsletter

Christian Kohls
Alexander Dobrynin

Programmieren lernen mit Kotlin

Grundlagen, Objektorientierung und fortgeschrittene Konzepte

2., aktualisierte Ausgabe

HANSER

Die Autoren:
Prof. Dr. Christian Kohls, Köln
Alexander Dobrynin, Gummersbach

Kotlin ist ein eingetragenes Warenzeichen der Kotlin Foundation

Alle in diesem Werk enthaltenen Informationen, Verfahren und Darstellungen wurden nach bestem Wissen erstellt und mit Sorgfalt getestet. Dennoch sind Fehler nicht ganz auszuschließen. Aus diesem Grund sind die im vorliegenden Werk enthaltenen Informationen mit keiner Verpflichtung oder Garantie irgendeiner Art verbunden. Autoren und Verlag übernehmen infolgedessen keine Verantwortung und werden keine daraus folgende oder sonstige Haftung übernehmen, die auf irgendeine Art aus der Benutzung dieses Programm-Materials – oder Teilen davon – entsteht. Ebensowenig übernehmen Autoren und Verlag die Gewähr dafür, dass beschriebene Verfahren usw. frei von Schutzrechten Dritter sind.Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Werk berechtigt also auch ohne besondere Kennzeichnung nicht zu der Annahme, dass solche Namen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürften.

Die endgültige Entscheidung über die Eignung der Informationen für die vorgesehene Verwendung in einer bestimmten Anwendung liegt in der alleinigen Verantwortung des Nutzers.

Aus Gründen der besseren Lesbarkeit wird auf die gleichzeitige Verwendung der Sprachformen männlich, weiblich und divers (m/w/d) verzichtet. Sämtliche Personenbezeichnungen gelten gleichermaßen für alle Geschlechter.

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.

Dieses Werk ist urheberrechtlich geschützt.
Alle Rechte, auch die der Übersetzung, des Nachdruckes und der Vervielfältigung des Buches, oder Teilen daraus, vorbehalten. Kein Teil des Werkes darf ohne schriftliche Genehmigung des Verlages in irgendeiner Form (Fotokopie, Mikrofilm oder ein anderes Verfahren), auch nicht für Zwecke der Unterrichtsgestaltung – mit Ausnahme der in den §§ 53, 54 URG genannten Sonderfälle –, reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden.

© 2023 Carl Hanser Verlag München, http://www.hanser-fachbuch.de
Lektorat: Sylvia Hasselbach
Copy editing: Jürgen Dubau, Freiburg/Elbe
Layout: le-tex publishing services GmbH
Coverkonzept: Marc Müller-Bremer, www.rebranding.de, München
Titelmotiv und Umschlagrealisation: Max Kostopoulos
Druck und Bindung: Hubert & Co. GmbH & Co. KG BuchPartner, Göttingen
Printed in Germany

Print-ISBN: 978-3-446-47712-4
E-Book-ISBN: 978-3-446-47849-7
E-Pub-ISBN: 978-3-446-48005-6

Inhalt

Vorwort

Kotlin ist inzwischen als Programmiersprache etabliert. Der Großteil aller professionellen Apps im Google-Play-Store ist in Kotlin entwickelt, und Studien von Google zeigen, dass Kotlin-Code robuster läuft. Zudem ist die Entwicklung im Vergleich zu Java sehr viel produktiver, sodass Kotlin schon aus ökonomischer Sicht viele Vorteile bietet. Vor allem aber: Kotlin macht Spaß und führt zu eleganterem Code.

Mit diesem Buch können Sie ohne Vorkenntnisse in die Programmierung einsteigen. Dabei werden Sie verschiedene Ansätze kennenlernen und praktisch anwenden. Nach der Lektüre des Buches können Sie kleinere Softwareprojekte entwickeln, also zum Beispiel eigene Ideen umsetzen, Aufgaben und Problemstellungen verstehen und lösen sowie Softwarespezifikationen in lauffähige Programme überführen. Sie können einfache Algorithmen selbst entwickeln und Standardalgorithmen und Datenstrukturen umsetzen. Sie können Apps für Android-Systeme entwickeln oder Programme für Server und Desktop-Rechner schreiben.

Die Welt des Programmcodes ist unsichtbar. Wir haben festgestellt, dass einige Konzepte besonders schwer zu begreifen sind und dass oft falsche Vorstellungen existieren. Es wurde daher großer Wert darauf gelegt, möglichst viele Konzepte mit Metaphern, praktischen Anwendungsbeispielen und Bildern zu veranschaulichen. Dabei bauen wir auf unseren langjährigen Erfahrungen in der Programmierausbildung auf. Am Ende des Buches können Sie Apps mit einer grafischen Benutzerschnittstelle entwickeln und aus unsichtbarem Code eine visuell ansprechende App entwickeln.

Dieses Buch richtet sich vor allem an Einsteiger und Anfänger. Es werden keine Vorkenntnisse vorausgesetzt. Gleichzeitig denken wir, dass auch fortgeschrittene Entwickler und Umsteiger von anderen Programmiersprachen von diesem Buch profitieren werden.

Hilfestellung bei der Umsetzung von Kotlin-Programmen bietet inzwischen auch der Online-Dienst *ChatGPT*. Sie können ChatGPT bitten, Algorithmen zu schreiben, Code zu überprüfen, Fehler zu finden und Codeabschnitte zu erklären. Das funktioniert oft sehr gut, aber nicht selten erfindet ChatGPT Lösungen, die zwar richtig aussehen, aber leider falsch sind. Daher raten wir zu einem vorsichtigen Umgang mit diesem Werkzeug. Zur Unterstützung beim Lernen ist ChatGPT sicherlich geeignet. Einzelne Codeabschnitte oder Konzepte können Sie sich von diesem Chatbot ausführlich erklären lassen. Bei einfachen Algorithmen funktioniert dies gut. Bei komplexeren Programmen kommt ChatGPT aber noch durcheinander, liefert unvollständige und eben auch falsche Lösungen.

In dieser überarbeiteten Auflage haben wir einige neue Sprachkonzepte aufgenommen und vor allem das Kapitel zur Android-App-Entwicklung vollständig überarbeitet. In der ersten

Auflage wurden die Layouts noch mit XML-Dateien beschrieben. In der nun vorliegenden Auflage geschieht die Entwicklung vollständig mit *Jetpack Compose*. Dieses Framework hat sich inzwischen für die Entwicklung von Android-Apps durchgesetzt.

Alle Codebeispiele und zusätzliche Übungsaufgaben finden Sie im Download-Portal von Hanser-Plus: Geben Sie auf

plus.hanser-fachbuch.de

diesen Zugangscode ein:

`plus-rn34m-tL9pr`

Unserem Ko-Autor der ersten Auflage, Florian Leonhard, möchten wir besonders danken für die gemeinsame Entwicklung und Umsetzung des Buchkonzepts. Für den fachlichen Austausch möchten wir uns bei unseren Teamkollegen an der TH Köln bedanken. Insbesondere bei David Petersen, der wesentliche Inspirationen zu diesem Buch beigetragen hat. Für intensives Feedback und fachlichen Austausch danken wir Anja Bertels, Dominik Deimel und Dennis Dubbert. Kotlin macht Spaß und mit euch zusammen besonders viel.

Und nun wünschen wir auch Ihnen viel Spaß beim Coden und Entwickeln!

Christian Kohls, Alexander Dobrynin Im Juli 2023

1 Einführung

Kotlin ist eine Programmiersprache, die verschiedene Ansätze verfolgt. Sie erlaubt objektorientierte und funktionale Programmierung. Das Entwickeln mit objektorientierten Programmiersprachen gehört heute zu den Grundanforderungen aller Informatikerinnen und Informatiker. Es gibt Ihnen die Möglichkeit, eigene Projektideen umzusetzen, betriebliche Prozesse zu modellieren und nützliche Werkzeuge für Anwender zu entwickeln. Die objektorientierte Herangehensweise schärft zudem Ihre analytischen und ganzheitlichen Denkfähigkeiten. Denn Sie müssen Sachverhalte differenziert betrachten, Theorien und Modelle entwickeln, Prozesse analysieren und optimieren, Probleme ganzheitlich betrachten und abwägen, Probleme in Teilprobleme zerlegen, die (Anwendungs-) Story verstehen und erzählen können. Zudem erlernen Sie ein systematisches Vorgehen für Problemlöseaufgaben und damit kreative Denkweisen. Objektorientierte Denker sind Weltversteher und Weltveränderer!

Weltversteher, weil sie die wahrgenommene Welt der Situation entsprechend und der Aufgabe angemessen in Klassen und Objekte, Zustandsbeschreibungen und Verhaltensweisen, Rollen und Beziehungen abbilden können. Weltveränderer, weil gute (und leider auch schlechte) Software unmittelbaren Einfluss auf den Lebensalltag der Anwender hat. Denken Sie nur daran, was mit Software alles möglich ist. Und bald werden Sie die nächste großartige App entwickeln! Während Objekte unsere Welt besonders gut abbilden, ist der Vorteil von Funktionen, dass sie Rechenregeln und Transformationen mit mathematischer Präzision festlegen können. Funktionale Programmieransätze bauen darauf auf und unterstützen uns dabei, Funktionalitäten miteinander zu verknüpfen, zu testen und wiederzuverwenden. Kotlin vereint diese und andere Konzepte zu einer modernen Sprache.

Kotlin ist eine noch recht junge Programmiersprache, die sich jedoch rasant durchsetzt. Fast alle kommerziell erfolgreichen Android-Apps werden derzeit in Kotlin entwickelt. Und das, obwohl Kotlin der Öffentlichkeit erst im Jahr 2011 erstmals vorgestellt wurde. Eine stabile Version gibt es seit 2016. Inzwischen hat Kotlin die Version 1.9 erreicht. Diese Version wurde am 6. Juli 2023 veröffentlicht, also kurz vor Redaktionsschluss dieses Buches. Wir haben die Neuerungen bereits in diesem Buch berücksichtigt und weisen an einzelnen Stellen darauf hin, wenn eine bestimmte Kotlin-Version benötigt wird. Das nächste größere Release wird Kotlin 2.0 sein.

Wir haben alle Kotlin-Entwicklerkonferenzen (KotlinConf) der letzten Jahre besucht und sind von der großen und aktiven Entwicklercommunity sehr begeistert. Neue Sprachfeatures werden mit der Community diskutiert, und Google bevorzugt Kotlin nicht nur als

Entwicklungssprache für die Android-Plattform, sondern stellt selbst große Teile des eigenen Codes auf Kotlin um. Viele bekannte Firmen setzen auf Kotlin und teilen auf der KotlinConf ihre Erfahrungen. Damit ist Kotlin im Gegensatz zu vielen anderen neuen Sprachen keine „akademische Sprache“, die nur für die Hochschullehre interessant ist. Ganz im Gegenteil: Immer mehr Firmen, Entwickler und Open-Source-Projekte wechseln zu Kotlin, da sich mit dieser Sprache Lösungen schneller und sicherer entwickeln lassen. Mit Jetpack Compose steht zudem ein sehr guter und moderner Ansatz für die Entwicklung grafischer Benutzeroberflächen bereit, mit dem sich nicht nur mobile Apps sehr schnell umsetzen lassen, sondern auch Desktop-Anwendungen für Ihren Mac oder PC.

Kotlin wurde von der Firma JetBrains entwickelt und steht unter einer Open Source-Lizenz. JetBrains ist ein führender Anbieter für integrierte Entwicklungsumgebungen wie z. B. IntelliJ. Bei JetBrains hat man irgendwann festgestellt, dass die Entwicklung mit der Programmiersprache Java an vielen Stellen zu aufwendig ist. Kotlin wurde basierend auf dem Erfahrungswissen, was effiziente Softwareentwickler wirklich benötigen, konzipiert. Kotlin vereint dabei viele erprobte Konzepte aus den letzten Jahrzehnten. Kotlin setzt konsequent auf Mechanismen, die sich bewährt haben und die in der Programmierpraxis zu einer erhöhten Produktivität führen – und dadurch letztlich mehr Freude bereitet. Ja, Kotlin macht Spaß!

1.1 Eine Sprache für viele Plattformen

Mit Kotlin können Sie für viele verschiedene Plattformen entwickeln.

Android: Kotlin ist die primäre Entwicklungssprache für die Android-Plattform von Google. Damit hat sie eine hohe Relevanz, da sich Apps für Android mit Kotlin schneller, besser und leichter entwickeln lassen.

JVM: Programme, die in Kotlin geschrieben werden, können für die Java-Plattform kompiliert werden. Das heißt: Die Kotlin-Programme werden in Java-Bytecode übersetzt und laufen dann auf jeder Java Virtuellen Maschine (JVM). Diese JVMs gibt es für verschiedene Betriebssysteme. Ihre Programme können also auf verschiedenen Rechnersystemen ausgeführt werden.

JavaScript: Kotlin kann aber auch nach JavaScript übersetzt werden. Das heißt: Sie können die Programmierung von Webanwendungen auch mit Kotlin durchführen. Dies gilt sowohl für die Programmierung des Clients (also JavaScript, das im Browser ausgeführt wird) als auch des Servers (also JavaScript, das auf dem Server ausgeführt wird).

Kotlin Native: Darüber hinaus gibt es mit Kotlin Native den Ansatz, ein Kotlin-Programm direkt für eine bestimmte Rechnerarchitektur zu kompilieren. Es wird also nicht Bytecode für eine virtuelle Maschine wie die JVM erzeugt, sondern echter Maschinencode für die jeweilige Plattform (z. B. iOS, Mac oder Linux).

Kotlin Multiplattform: Die Multiplattform-Entwicklung zielt darauf ab, dass Sie Ihre Programme einmal schreiben und dann auf verschiedenen Plattformen wie z. B. Android, iOS, Desktop-Systeme und im Web laufen lassen können. Dabei wird die Anwendungslogik geteilt und für alle Systeme verwendet. Sie legen z. B. nur einmal fest, wie Daten verarbeitet

werden sollen. Die Gestaltung der Benutzeroberfläche kann dann die nativen Bedienelemente der verschiedenen Plattformen berücksichtigen. Mit Jetpack Compose, das wir ebenfalls in diesem Buch behandeln, ist es sogar möglich, die gesamte Benutzeroberfläche plattformübergreifend zu gestalten. Diese Technologie befindet sich allerdings aktuell noch zum Teil in der Alpha-Version (Stand Juli 2023).

1.2 Deshalb ist Kotlin so besonders

Hier noch einmal die wichtigsten Gründe, die für Kotlin sprechen:

Einfacher Einstieg: Kotlin lässt sich schneller lernen. Es ist verständlicher, und die Sprachelemente sind prägnanter, also ausdrucksstärker. Sie können sich besser auf die wesentlichen Konzepte fokussieren, da unnötiger „Boilerplate Code“ entfällt. Das ist Code, der eigentlich nicht nötig ist, z. B. weil er keine neuen Sachverhalte ausdrückt. Ein Beispiel sind die vielen setter- und getter-Methoden, die man in anderen Programmiersprachen wie zum Beispiel Java schreiben muss.

Eleganter: Viele Programme lassen sich mit Kotlin eleganter schreiben. Mehr noch: Man muss die Programme eleganter schreiben. Sie lernen also gleich den richtigen Stil und können diesen auch in anderen Sprachen einsetzen.

Effiziente Entwicklung: Vieles lässt sich in Kotlin kürzer und effizienter ausdrücken. Das spart nicht nur Zeit beim Schreiben des Quellcodes. Durch kurze, übersichtliche Programme lassen sich Fehler besser vermeiden, und Sie behalten besser den Überblick.

Effiziente Ausführung: Kotlin stellt viele optimierte Algorithmen für immer wieder anfallende Aufgaben zur Verfügung. Das Verarbeiten oder Sortieren von Listen ist in Kotlin sehr gut gelöst und leicht nutzbar.

Erprobte Konzepte: Kotlin baut auf erprobten Konzepten auf. Die Sprache enthält konzeptionell nichts Neues, dafür (fast) alles Gute aus den letzten 50 Jahren Softwareentwicklung. Die Sprache ist praktisch und effizient, bietet mehr Sicherheit und vereint objektorientierte und funktionale Programmierkonzepte.

Erklärt sich selbst: Der Quellcode ist lesbarer für Menschen. Sie können den Code von anderen Entwicklern (und womöglich Ihren eigenen Code selbst nach ein paar Wochen) viel besser verstehen.

Error-less: Viele Fehler, die man in anderen Sprachen leicht machen kann, werden in Kotlin gar nicht erst zugelassen oder zumindest leichter vermieden. So gibt es in Kotlin beispielsweise keine sogenannten *Null-Fehler* mehr.

Erfrischend: Viele Entwickler berichten, dass das Entwickeln mit Kotlin wieder mehr Spaß macht! Und tatsächlich bekommt man beim Entwickeln mit Kotlin leuchtende Augen. Und manchmal auch tränende Augen – vor Freude über Kotlin und Ärger darüber, dass früher so vieles anstrengender war.

1.3 Darauf dürfen Sie sich freuen

Aufbau des Computers: Programme laufen auf dem Prozessor eines Computers. Dabei verändern sie Daten im Speicher. Um zu verstehen, wie aus dem Quellcode, den Sie schreiben, lauffähige Programme auf dem Rechner werden, schauen wir uns zunächst den grundlegenden Aufbau von Computern an. Dabei werden wir sehen, wie aus lauter Nullen und Einsen bunte Bilder werden können. Da wir Daten im Speicher liegen haben, werden wir uns auch mit der Organisation des Speichers beschäftigen.

Basics: Zur Steuerung von Programmen benötigen wir ein paar grundlegende Zutaten. Dazu gehören Ausdrücke und Anweisungen, um dem Rechner zu sagen, was er ausführen soll. Mit Variablen können wir Daten speichern und später wieder darauf zugreifen. Mit Kontrollstrukturen können wir steuern, wie ein Programm abläuft. Somit können wir auf Benutzereingaben und verschiedene Zustände reagieren. Mit Funktionen werden wir wiederverwendbare Codebausteine definieren, aus denen sich komplexe Programme zusammensetzen lassen.

Objekte und Klassen: Wir werden Grundlagen über Objekte und Abstraktion als Klassen kennenlernen. Wie können wir ermitteln, welche Eigenschaften von Objekten relevant sind? Wie abstrahiere ich von konkreten Objekten in der Welt auf eine allgemeine Klasse? Klassen beschreiben die Struktur, die Eigenschaften und die Verhaltensweisen von allen Objekten, die zu einer Klasse gehören. Man spricht von Objektinstanzen (oder einfache Instanzen), wenn man sich auf die Exemplare einer Klasse bezieht. Das Erzeugen eines neuen Objekts erfolgt über Konstruktoren. Wir werden uns ansehen, wie Konstruktoren die erste Version eines Objekts bauen. Zudem werden wir Objekte miteinander in Beziehung setzen. Zum Beispiel werden wir einen Film aus Budget, Schauspieler und Regisseur zusammensetzen, um dies in einem Spiel zu verwenden.

Vererbung und Polymorphie Dies ist ein weiteres zentrales Thema der objektorientierten Programmierung. Klassen stehen in einer Vererbungshierarchie zueinander. Klassen können andere Klassen erweitern, wobei die Eigenschaften und Methoden einer Oberklasse an die Unterklasse vererbt werden. Wenn die übergeordnete Klasse Tasse die Eigenschaft „Volumen“ und die Methode „trinken“ besitzt, dann wird auch die untergeordnete Klasse Kaffeetasse diese Eigenschaften haben, sie kann aber um spezifische Eigenschaften (z. B. „Kaffeesorte“) und Methoden (z. B. „istNochHeiss“) ergänzt werden. Dabei kann das Verhalten von Methoden auch durch Überschreiben geändert werden (z. B. weil man aus einer Thermotasse anders trinkt). Eng verbunden mit der Definition von Klassen und Schnittstellen sind Typkonzepte und Polymorphie. Polymorphie bedeutet Vielgestaltigkeit. Wenn Sie an „Fahrzeug“ denken, dann kann dieses Fahrzeug sehr viele unterschiedliche Gestalten haben: Auto, Fahrrad, Kutsche. Dennoch können Sie allgemeine Eigenschaften und Methoden nutzen, wenn diese Gestalten auf einer abstrakteren Ebene vom gleichen Typ sind. So lässt sich für alle Fahrzeuge sagen: „fahre los“. Egal, ob es sich um eine Kutsche oder ein Fahrrad handelt. Die Umsetzung wird aber ganz unterschiedlich aussehen.

Robustheit: Damit wir auch ordentliche Programme schreiben können, werden wir uns mit verschiedenen Konzepten zur Robustheit von Code beschäftigen. Wie können Sie auf Fehler und Ausnahmesituationen reagieren? Was passiert, wenn eine Datei gelesen werden soll, die gar nicht existiert? Was passiert, wenn ein Server nicht erreichbar ist? Ausnahmezustand! Wir werden lernen, wie wir eine Operation versuchen können und das Scheitern schon

einplanen und auffangen. Das funktioniert bestens für Situationen, wo Sie selbst keinen Einfluss darauf haben, ob etwas wie gewünscht funktioniert. Dann können Sie nämlich darauf reagieren. Denn das Auffangen ist viel komplizierter als die Fehlerkorrektur. Und vor allem wollen Sie bei der Entwicklung ja auch, dass Fehler erkannt und somit beseitigt werden können. Wir werden uns die Vor- und Nachteile für verschiedene Software-Designoptionen anschauen. Aus diesen Überlegungen heraus haben sich über die Jahre hinweg Entwurfsmuster entwickelt, die gute Lösungen für bestimmte Aufgaben darstellen.

Datensammlungen: Wir schauen uns verschiedene Möglichkeiten an, um Datensammlungen (z. B. Listen, Verzeichnisse, Paare, Mengen) abzubilden und Operationen darauf auszuführen. Zudem werden wir selbst Datenstrukturen entwickeln, um Objekte zu speichern. Dies wird unter anderem am Beispiel einer verketteten Liste illustriert.

UI Design und App-Entwicklung: Die Android-Plattform stellt spezielle Bibliotheken zur Verfügung, um grafische Oberflächen zu gestalten und die Funktionen von Android-Geräten zu nutzen. Wir werden eine grafische Oberfläche bauen und Code definieren, um auf Eingabeevents (Klick auf einen Button) zu reagieren.

Nebenläufigkeit: Häufig müssen mehrere Dinge gleichzeitig ausgeführt werden. Bei Spielen sollen zum Beispiel gleichzeitig mehrere Figuren bewegt werden. Auf Ihrem Smartphone sollen gleichzeitig mehrere Bilder aus dem Internet heruntergeladen werden, während Sie weiterhin mit dem Programm interagieren. Für diese Nebenläufigkeit führt Kotlin das Konzept der *Coroutines* ein. Wir werden uns anschauen, wie sich nebenläufige Programme gestalten lassen und auf welche Stolpersteine geachtet werden muss.

TEIL I

Konzeptioneller Aufbau von Computern und Software

Um ein besseres Verständnis zu erlangen, was es eigentlich heißt, Software zu entwickeln und lauffähige Programme zu schreiben, müssen wir uns ein wenig mit der Hardware beschäftigen – dem Computer. Computer sind heute allgegenwärtig. Damit sind nicht nur die großen Kisten unterm Schreibtisch oder die mobilen Laptops gemeint. Auch Ihr Smartphone ist ein Computer! Und zwar ein Computer, der sehr viel leistungsfähiger ist als die großen Geräte, die noch vor ein paar Jahren unter dem Schreibtisch standen. Darüber hinaus erhalten kleine Computer jedoch auch Einzug in immer mehr Alltagsgegenstände und Maschinen: von der Zahnbürste über die Waschmaschine, über den Fahrkartenautomat hin zum Autopiloten in Bahnen oder Flugzeugen und natürlich der Bordcomputer in unseren Autos. Viele Waschmaschinen nutzen inzwischen dieselben Chips, die einst in unseren ersten Heimcomputern eingebaut wurden.

Doch was ist eigentlich ein Computer und woraus setzt er sich zusammen? Der Begriff *Computer* leitet sich aus dem englischen Verb *compute* ab, welches mit *rechnen* übersetzt werden kann. Daher wird im deutschsprachigen Raum der Computer auch häufig als *Rechner* bezeichnet. Tatsächlich macht ein Computer nichts anderes: Er rechnet und rechnet und rechnet und rechnet.

Das klingt vielleicht etwas merkwürdig, wenn wir an die uns vertrauten Anwendungen denken: Was hat die Kurznachricht bei WhatsApp oder Twitter mit Berechnungen zu tun? Wo bitte schön sind die Berechnungen bei grafikintensiven Spielen wie SimCity, Angry Birds oder Fortnite? Die Antwort lautet: ziemlich versteckt, aber dafür überall. Und daher wollen wir uns in diesem Kapitel ein kleines bisschen auf die Suche nach diesen Berechnungen und Zahlen machen. Denn wenn wir verstehen, was unsere Software und Hardware im Innersten zusammenhält, dann bekommen wir auch ein besseres Verständnis für die Funktionsweise der Programme, die wir selbst entwickeln. Und wir werden verstehen, was eigentlich ein „Programm“ ist!

2 Komponenten eines Computers

Zunächst besteht ein Computer nur aus diesen wesentlichen Komponenten:

- Speicher, in dem Daten liegen
- Prozessoren, die diese Daten verarbeiten
- Eingabe- und Ausgabegeräte

Die Realität sieht natürlich sehr viel komplexer aus, denn es gibt verschiedene Speicherarten, auf die unterschiedlich schnell zugegriffen werden kann. Zudem kann Speicher unterschiedlich organisiert werden – ganz so, wie Sie auch Ihre Wäsche unterschiedlich im Schrank anordnen können. Dabei gibt es für verschiedene Zwecke optimierte Vorgehensweisen. Und es gibt unterschiedliche Prozessoren, die für die Verarbeitung bestimmter Daten optimiert sind. So gibt es etwa Prozessoren, die allgemeine Rechenoperationen ausführen können, und spezialisierte Prozessoren, die besonders schnell Berechnungen für die Grafikausgabe oder die Sounderzeugung anstellen können. Zudem müssen die Daten aus den Speichern zu den Prozessoren gelangen, und auch die Ein- und Ausgabegeräte wollen natürlich etwas mit den Daten anfangen. Um diese Daten zu befördern, gibt es sogenannte Datenbusse. Auch diese können unterschiedlich schnell sein. Auch bei den Ein- und Ausgabegeräten gibt es viele verschiedene, Ihnen bekannte Geräte. Typische Eingabegeräte sind die Tastatur, eine Maus, eine touchfähige Oberfläche über dem Bildschirm, Sensoren für Temperatur, Bewegung oder Helligkeit, Scanner, Kameras und viele mehr. Zu den typischen Ausgabegeräten gehören Bildschirme, Lautsprecher, Drucker, aber z. B. auch Roboter, die durch Computer gesteuert werden.

2.1 Beliebige Daten als binäre Zahlen

Und wenn wir von Daten reden, dann sind diese intern stets und ohne Ausnahme als Zahlen repräsentiert. Ja, jede Kurznachricht, jedes Meme, jedes Video, jeder Text und jede Fahrbewegung eines Roboters, Autos oder Flugzeugs wird intern durch Zahlen abgebildet. Genauer gesagt durch Binärzahlen, also Zahlen, die nur aus 0 und 1 bestehen. Dies hat einen guten Grund: Zwei Zustände lassen sich sehr gut durch physische Hardware abbilden, z. B. durch unterschiedliche magnetische Ausrichtungen oder Spannungen. Mit diesen zwei voneinander unterscheidbaren Zuständen lassen sich prinzipiell alle Arten von Daten und Informationen abbilden, indem sie unterschiedlich interpretiert werden:

Physischer Zustand	Interpretation als Schaltung	Interpretation als Antwort	Interpretation als Wahrheits-aussage	Interpretation als Zahl
Oberhalb Grenzwert	EIN	Ja	WAHR	1
Unterhalb Grenzwert	AUS	Nein	FALSCH	0

Bei der Interpretation als Zahl können wir mehrstellige Zahlen abbilden, wenn mehrere Bits gemeinsam betrachtet werden. Dies ist vergleichbar mit dem uns vertrauten Dezimalsystem, also dem Zahlensystem mit den Ziffern 0–9. Während dieses Dezimalsystem aus nur 10 verschiedene Ziffern besteht, können wir dennoch beliebig große Zahlen abbilden, z. B.:

```
> 342359
```

Die Position der Ziffer gibt dabei die Wertigkeit an. So bedeutet die 9 an letzter Position, dass es sich um den Wert 9 handelt. Die Ziffer an der vorletzten Position bedeutet dagegen, dass sie mit dem Faktor 10 multipliziert wird. Die fünf bedeutet also 50. Genauer gesagt wird die Ziffer an der Position n (von hinten ausgehend) mit dem Faktor 10^n multipliziert. Die letzte Position ist dabei die 0. Stelle, sodass für die Beispielzahl gilt:

Ziffer	Konvertierung	Wertigkeit als Dezimalzahl	Summe
3	x 10^5	100000	300000
4	x 10^4	10000	40000
2	x 10^3	1000	2000
3	x 10^2	100	300
5	x 10^1	10	50
9	x 10^0	1	9
Summe:			342359

Während sich beim Dezimalsystem die Wertigkeit bei jeder Position verzehnfacht (also Faktor 10), verdoppelt sich beim Binärsystem die Wertigkeit bei jeder Position (also Faktor 2). Auf diese Weise lässt sich eine Binärzahl mit mehreren Ziffern zusammensetzen. In Computern werden dabei jeweils mindestens 8 Stellen zu einer Einheit, die *Byte* genannt wird, zusammengefasst. Eine Binärzahl ist z. B.:

```
> 00010101
```

Wenn man diese Binärzahl in eine Dezimalzahl umrechnen möchte, dann multipliziert man jede Stelle mit 2^n statt mit 10^n. So ergibt sich für das gerade angeschaute Beispiel: