Daniel Braun

JAVA Schnelleinstieg

Programmieren lernen in 14 Tagen

Einfach und ohne Vorkenntnisse

Zahlreiche Praxisbeispiele und Übungen

mitp

Daniel Braun

Java

Schnelleinstieg

Programmieren lernen in 14 Tagen
Einfach und ohne Vorkenntnisse

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 978-3-7475-0393-5
1. Auflage 2022

www.mitp.de
E-Mail: mitp-verlag@sigloch.de
Telefon: +49 7953 / 7189 - 079
Telefax: +49 7953 / 7189 - 082

Lektorat: Janina Bahlmann
Sprachkorrektorat: Petra Heubach-Erdmann
Covergestaltung: Janina Bahlmann, Christian Kalkert
Covergrafik & Icons: Tanja Wehr, sketchnotelovers
Satz: Petra Kleinwegen

Inhalt

Variablen

Verzweigungen und Schleifen

Funktionen

Objektorientierung

Arrays und Listen

Fehlerbehandlung

Dateisystem

Externe Bibliotheken

Grafische Benutzerschnittstellen

Multitasking

Datenbanken

REST-Schnittstellen

Stichwortverzeichnis

Einleitung

E.1 Programmieren lernen in 14 Tagen

Mit diesem Buch haben Sie sich für einen einfachen, praktischen und fundierten Einstieg in die Welt der Programmierung entschieden. Sie lernen ohne unnötigen Ballast alles, was Sie wissen müssen, um Java effektiv für Projekte in Ihrem Berufs- und Interessensgebiet einzusetzen.

Wenn Sie Zeit genug haben, können Sie jeden Tag ein neues Kapitel durcharbeiten und so innerhalb von zwei Wochen Programmieren lernen. Alle Erklärungen sind leicht verständlich formuliert und setzen keine Vorkenntnisse voraus. Am besten lesen Sie das Buch neben der Computer-Tastatur und probieren die Programmbeispiele und Übungen gleich aus. Sie werden schnell erste Erfolge erzielen und Freude an der Programmierung finden.

E.2 Der Aufbau des Buches

Das Buch beginnt mit den Grundlagen: Installation von Java, Nutzung verschiedener Entwicklungsumgebungen und Formulierung einfacher Anweisungen. Die Kapitel bauen aufeinander auf. Sie lernen Schritt für Schritt, wie man Daten lädt, verarbeitet und speichert, sowohl in Dateien als auch in Datenbanken, und erhalten eine Einführung in die Verwendung von Funktionen, objektorientierte Programmierung, die Gestaltung von grafischen Benutzeroberflächen und Programmieren von nebenläufigen Anwendungen. Das letzte Kapitel schließlich beschäftigt sich mit der Implementierung einer Web-Schnittstelle in Java und zeigt Ihnen einige Möglichkeiten, wie Sie nach dem Schnelleinstieg Ihre Programmierkenntnisse weiterentwickeln können.

Ihr Tagespensum schließt mit praktischen Programmier-Übungen, in denen Sie Ihr neu gewonnenes Wissen vertiefen können. Die Lösungen zu diesen Übungen, die relativ viel Programmtext enthalten, stehen in einem Online-Kapitel zum Download zur Verfügung. Mehr dazu im nächsten Abschnitt.

Am Ende des Buches finden Sie ein Stichwortverzeichnis, das Ihnen hilft, bestimmte Themen im Buch schneller zu finden.

E.3 Programmtexte und Lösungen zum Download

In diesem Buch wird eine vollständige Personenverwaltung mit Anbindung an eine Datenbank, grafischer Benutzungsoberfläche und Web-Schnittstelle entwickelt, um die gezeigten Techniken zu demonstrieren und anzuwenden.

Der Code der Personenverwaltung in den unterschiedlichen Entwicklungsstadien sowie die Lösungen zu den Übungen stehen Ihnen auf der Webseite des Verlags unter *www.mitp.de/0392* zum Download zur Verfügung.

Dort finden Sie außerdem ein praktisches Glossar mit den wichtigsten Fachbegriffen.

E.4 Fragen und Feedback

Unsere Verlagsprodukte werden mit großer Sorgfalt erstellt. Sollten Sie trotzdem einen Fehler bemerken oder eine andere Anmerkung zum Buch haben, freuen wir uns über eine direkte Rückmeldung an *lektorat@mitp.de*.

Falls es zu diesem Buch bereits eine Errata-Liste gibt, finden Sie diese unter *www.mitp.de/0392* im Reiter Downloads.

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

Daniel Braun und das mitp-Lektorat

1 Erste Schritte mit Java

Eine wichtige Entscheidung auf dem Weg zum Programmierer haben Sie bereits mit der Wahl dieses Buches getroffen: die Entscheidung für die Programmiersprache Java. Obwohl viele wichtige Grundkonzepte des Programmierens unabhängig von der gewählten Programmiersprache sind, gibt es doch Unterschiede. Dieses Kapitel wird einige der Besonderheiten von Java erklären und warum die Sprache eine gute Wahl für Programmiereinsteiger ist. Zuvor erhalten Sie aber zunächst eine Einführung in einige der allgemeinen Grundbegriffe und Konzepte des Programmierens.

1.1 Programmiersprachen

Computer können uns beim Lösen zahlreicher Aufgaben helfen, oft sind sie dabei sogar besser als der Mensch, zum Beispiel beim Rechnen oder dem Sortieren von Daten. Um eine Aufgabe erledigen zu können, benötigt der Computer eine genaue Anleitung zur Lösung eben jener Aufgabe. Eine solche Anleitung mit exakten Handlungsvorschriften für einen Computer nennt man auch *Algorithmus*.

Handlungsvorschriften gibt es nicht nur für Computer, sondern auch für Menschen, zum Beispiel in Form von Rezepten in Kochbüchern oder Aufbauanleitungen für Möbelstücke. Allerdings sprechen Mensch und Maschine unterschiedliche Sprachen, und das aus gutem Grund. Unsere menschlichen Sprachen, egal ob Deutsch, Englisch oder Plansprachen wie Esperanto, sind vage. Eine »Prise Salz«, zum Beispiel, ist eine vollkommen normale Angabe in einem Rezept und bringt keinen Koch ins Schwitzen. Für einen Backroboter aber wäre eine solche Angabe nutzlos, sie ist nicht eindeutig. Unsere menschlichen Sprachen sind voll von solch mehrdeutigen Aussagen. Denken Sie zum Beispiel an den Satz: »Da vorne ist eine Bank.« Ein vollständiger und korrekter deutscher Satz, aber was ist gemeint? Befindet sich »da vorne« eine Parkbank zum Sitzen oder ein Kreditinstitut zum Einzahlen und Abheben von Geld?

Ein Algorithmus muss aus *eindeutigen* Handlungsvorschriften bestehen. Daher sind natürliche Sprachen, wie Deutsch und Englisch, ungeeignet zur Beschreibung von Computerprogrammen. Die »Muttersprache« von Computern sind Einsen und Nullen, Strom an oder aus. Eine Sprache unverständlich für Menschen. Daher wurden Programmiersprachen entwickelt, als Zwischenrepräsentation. Im Gegensatz zu unserer natürlichen Sprache sind Programmiersprachen nicht doppeldeutig, gleichzeitig sind sie, obwohl sie auf den ersten Blick kompliziert scheinen können, relativ einfach für den Menschen zu verstehen.

Damit der Computer die Programmiersprache versteht, muss sie trotzdem übersetzt werden, in sogenannte *Maschinensprache*. Diese Übersetzung geschieht automatisch mit einem Programm, das den Namen *Compiler* trägt, abgeleitet vom englischen Verb »to compile«, was so viel bedeutet wie »zusammensetzen« oder »erstellen«. Das Ergebnis des *Kompilierens* sind sogenannte *Binärdateien*, die die Maschinensprache, codiert in Nullen und Einsen, enthalten, wie in Abbildung 1.1 gezeigt.

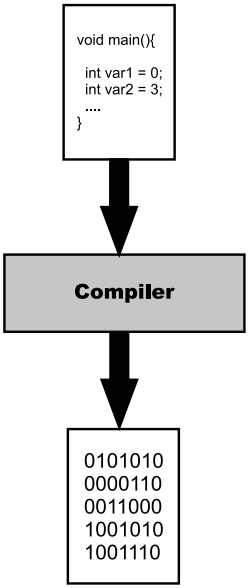

Abb. 1.1: Funktionsweise eines Compilers

Für die Übersetzung von menschlicher Sprache in Programmiersprache gibt es leider noch keine automatische Übersetzung, dafür sind Sie als Programmierer zuständig, und wie das funktioniert, lernen Sie mit diesem Buch.

1.2 Besonderheiten von Java

Als Programmierer haben Sie die Wahl zwischen Hunderten Programmiersprachen. Einige davon, wie zum Beispiel C, Python oder JavaScript, sind weit verbreitet und werden häufig eingesetzt, andere eher obskur und nur für spezielle Einsatzzwecke oder Hardware geeignet. Java gehört zur Gruppe der populären Sprachen mit zahlreichen Einsatzgebieten. Dabei bietet jede Programmiersprache ihre eigenen Vor-, aber auch Nachteile. Eine »richtige« oder »beste« Programmiersprache gibt es nicht – je nach Anwendungsfall kann der Einsatz einer anderen Programmiersprache sinnvoll sein.

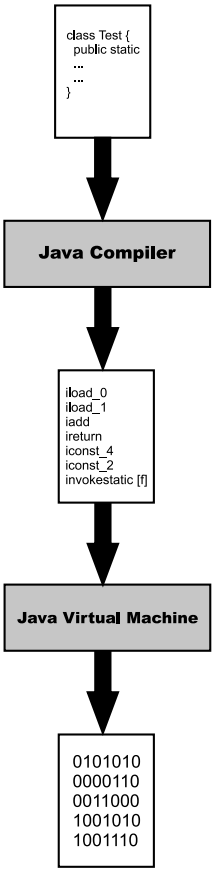

Abb. 1.2: Funktionsweise des Java-Compilers

Eine der wichtigsten Eigenschaft von Java ist die Tatsache, dass Programme, die in Java geschrieben wurden, auf allen gängigen Plattformen und Betriebssystemen, also insbesondere Windows, Linux und macOS, ausgeführt werden können. Im Vergleich dazu kann zum Beispiel ein C-Programm, das unter Windows kompiliert wurde, auch nur unter Windows ausgeführt werden. Um diese Plattformunabhängigkeit zu ermöglichen, verwendet Java eine weitere Abstraktionsebene zwischen der Programmiersprache und der Maschinensprache, den sogenannten *Bytecode*.

Der Java-Compiler wandelt den Programmcode, wie in Abbildung 1.2 gezeigt, zunächst in diese Zwischenrepräsentation um. Dieser Bytecode kann dann von einem weiteren Programm, einem sogenannten *Interpreter*, ausgeführt werden. Der Java-Interpreter wird auch *Java Virtual Machine* (*JVM*) genannt. So muss nur ein Programm, nämlich die JVM, auf das jeweilige Betriebssystem angepasst werden und sofort können sämtliche Java-Programme ausgeführt werden.

Allerdings hat dieser Vorteil auch seinen Preis. Die JVM muss den Bytecode in Echtzeit – das heißt, während das Programm ausgeführt wird – in Maschinensprache übersetzen. Bei einem klassischen Compiler ist das nicht notwendig.

Das führt dazu, dass Java, aber auch beispielsweise Python, das nach demselben Prinzip funktioniert, in der Ausführung weniger schnell ist als zum Beispiel die Programmiersprache C. Für sehr ressourcenintensive Anwendungen, zum Beispiel 3D-Grafik, ist Java deshalb häufig nicht die beste Wahl.

Eine weitere Besonderheit von Java ist die strikte *Typisierung* und *Objektorientierung*. Was genau es damit auf sich hat, das werden Sie in den folgenden Kapiteln erfahren. Das ist ein Nachteil für Programmiereinsteiger, den Java mit sich bringt. Gerade am Anfang gibt es ein gewisses Grundgerüst für jedes Programm, das Sie einfach als gegeben hinnehmen müssen, bis Sie Schritt für Schritt besser verstehen werden, was dahintersteckt. Dafür lernen Sie aber von Beginn an wichtige Grundlagen explizit kennen, die andere Programmiersprachen, wie zum Beispiel Python oder JavaScript, vor Ihnen verstecken. Deshalb ist Java vielleicht nicht die einfachste Sprache zum Programmierenlernen, aber trotzdem eine sehr gute Wahl für den Einstieg. Nicht umsonst ist Java elementarer Bestandteil des Informatikstudiums an den meisten Universitäten.

Außer dem Namen haben die beiden Programmiersprachen Java und JavaScript nicht viel gemein. JavaScript, eine Programmiersprache, die ursprünglich hauptsächlich zum Einsatz auf Websites entwickelt wurde, hieß zunächst auch LiveScript. Durch eine Partnerschaft zwischen dem ursprünglichen Entwickler von LiveScript und dem Entwickler von Java, Sun Microsystems, kam es zur namentlichen Annäherung.

Java ist universell einsetzbar. Es wird zum Beispiel häufig zur Programmierung von betrieblichen Anwendungen verwendet, unter anderem, da es einfach ist, ein Programm über mehrere Rechner zu verteilen, was zum Beispiel dabei hilft, steigende Hardwareanforderungen bei wachsenden Nutzerzahlen zu erfüllen. Ebenso lassen sich Android-Apps mit Java programmieren – ja selbst die im Juni 2003 von der NASA gestartete Raumsonde »Spirit« bewegte sich mithilfe von Java über die Oberfläche des Mars.

1.3 Installation und Einrichtung

Bevor Sie damit beginnen können, Java-Programme zu entwickeln – egal ob für Raumsonden oder Smartphones –, müssen Sie zunächst Ihren Rechner so einrichten, dass er Java-Programme ausführen und kompilieren kann. Die JVM, die benötigt wird, um Java-Programme auszuführen, ist als Teil des sogenannten *Java Runtime Environments (JRE)* bereits auf den meisten Compu-

tern installiert. Um selbst Java-Programme erstellen zu können, reicht das JRE allerdings nicht aus, dafür wird das *Java Development Kit (JDK)* benötigt, das unter anderem den Java-Compiler beinhaltet.

1.3.1 Linux

Unter Linux lässt sich das JDK am einfachsten über den Paketmanager apt installieren. Dazu müssen Sie lediglich den Befehl `sudo apt-get install openjdk-17-jdk` im Terminal eingeben, wobei die Zahl 17 die Versionsnummer beschreibt und je nach gewünschter Version angepasst werden muss. Welche Version gerade aktuell ist, können Sie zum Beispiel unter *https://jdk.java.net* einsehen.

1.3.2 macOS

Um das JDK auf Ihrem Apple-Rechner zu installieren, müssen Sie zunächst die gewünschte Version unter *https://jdk.java.net* herunterladen. Achten Sie dabei darauf, die macOS-Version auszuwählen. Danach öffnen Sie das Terminal und wechseln mit dem Befehl `cd ~/Downloads` in den Download-Ordner, wo sich die noch gepackten Dateien des JDK befinden. Mit dem Befehl `tar xf openjdk-17_osx-x64_bin.tar.gz` können Sie die Dateien entpacken, wobei die Zahl 17 der Versionsnummer entspricht und je nach Version, die Sie heruntergeladen haben, angepasst werden muss.

Die entpackten Dateien müssen dann noch in den richtigen Ordner verschoben werden, damit sie vom System auch gefunden werden können. Dies geschieht über den Befehl `sudo cp -Rv jdk-17.jdk /Library/Java/JavaVirtualMachines/`, wobei auch hier wieder die Versionsnummer entsprechend angepasst werden muss.

1.3.3 Windows

Um das JDK unter Windows zu installieren, müssen Sie zunächst die gewünschte Version unter *https://jdk.java.net* herunterladen. Achten Sie dabei darauf, die Windows-Version auszuwählen, die Sie an der Endung `.zip` erkennen.

Nach dem Download muss das ZIP-Archiv zunächst entpackt werden. Hierzu klicken Sie mit der rechten Maustaste auf das Archiv und wählen den Eintrag Alle Extrahieren aus. Im extrahierten Archiv befindet sich dann ein Ordner, der den Namen `jdk-17` trägt, wobei die Zahl 17 der Versionsnummer entspricht und entsprechend abweichen kann, je nachdem, welche Version Sie

heruntergeladen haben. Diesen Ordner können Sie nun an einen beliebigen Platz verschieben. Es empfiehlt sich standardmäßig, einen Ordner `C:\Programme\Java` anzulegen, falls dieser noch nicht existiert, und den `jdk`-Ordner dorthin zu verschieben.

Nun muss dem Betriebssystem noch mitgeteilt werden, wo das JDK zu finden ist. Dafür muss die sogenannte *Path-Variable* angepasst werden, die in den erweiterten Systemeinstellungen zu finden ist. Diese lassen sich auf verschiedene Arten öffnen: Durch Drücken der Windows-Taste und des Buchstabens R öffnet sich das Fenster Ausführen. Dort können Sie `sysdm.cpl` eingeben und durch einen Klick auf OK die Systemeigenschaften öffnen. Hier können Sie dann die Kategorie Erweitert auswählen.

Unter Windows 8 und 10 können Sie alternativ auch den Begriff Erweiterte Systemeinstellungen in die Windows-Suche eingeben und so direkt in das entsprechende Fenster springen.

Egal, welchen Weg Sie wählen, danach sehen Sie das in Abbildung 1.3 gezeigte Fenster auf Ihrem Bildschirm.

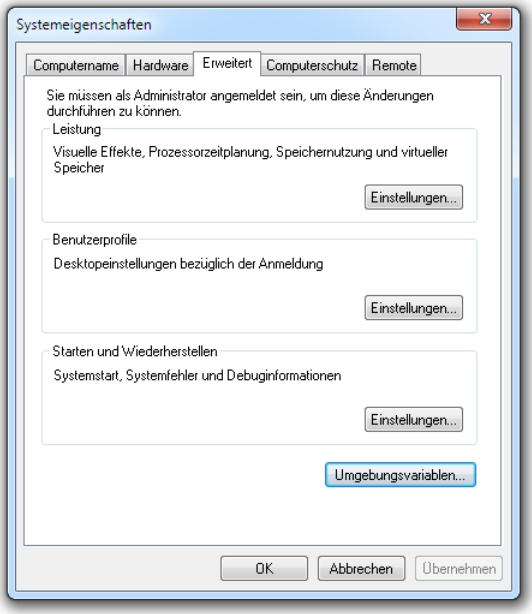

Abb. 1.3: Erweiterte Systemeinstellungen

Dort klicken Sie dann auf den Button Umgebungsvariablen..., woraufhin sich das in Abbildung 1.4 gezeigte Fenster öffnet. Hier wählen Sie nun im oberen Bereich der Benutzervariablen den Eintrag PATH aus und klicken auf den Bearbeiten...-Button. Im sich daraufhin öffnenden Fenster legen Sie dann einen neuen Eintrag an, der den Pfad zum `bin`-Ordner innerhalb des JDK enthält. Also zum Beispiel `C:\Programme\Java\jdk-17\bin`, wobei auch hier die Zahl 17 wieder für die Versionsnummer steht und entsprechend der von Ihnen verwendeten Version angepasst werden muss. Danach müssen die Änderungen nur noch mit einem Klick auf OK bestätigt werden.

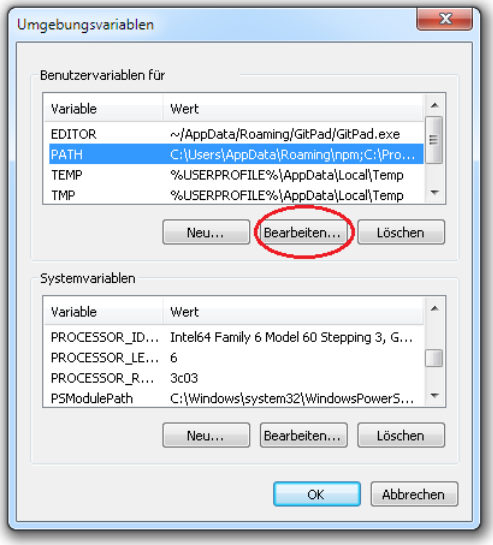

Abb. 1.4: Umgebungsvariablen

1.3.4 Installation testen

Damit ist die Installation des JDK bereits abgeschlossen und Sie können testen, ob alles korrekt eingerichtet ist. Dazu öffnen Sie ein Terminal – beziehungsweise unter Windows die Eingabeaufforderung – und geben den Befehl `javac` ein. Hierdurch wird der Java-Compiler aufgerufen. Ist dieser korrekt installiert und eingerichtet, so erscheint als Ausgabe auf dem Bildschirm die Hilfe für den Befehl, die dessen Nutzung erklärt, wie in Abbildung 1.5 gezeigt. Erscheint dagegen eine Fehlermeldung, so ist bei der Installation etwas schiefgelaufen, und Sie sollten noch einmal überprüfen, ob alle Eingaben und ins-

besondere Pfade korrekt waren, insbesondere auch, ob Sie überall die korrekte Versionsnummer angegeben haben.

```
Usage: javac <options> <source files>
where possible options include:
  -g                         Generate all debugging info
  -g:none                    Generate no debugging info
  -g:<lines,vars,source>     Generate only some debugging info
  -nowarn                    Generate no warnings
  -verbose                   Output messages about what the compiler is doing
  -deprecation               Output source locations where deprecated APIs are u
sed
  -classpath <path>          Specify where to find user class files and annotati
on processors
  -cp <path>                 Specify where to find user class files and annotati
on processors
  -sourcepath <path>         Specify where to find input source files
  -bootclasspath <path>      Override location of bootstrap class files
  -extdirs <dirs>            Override location of installed extensions
  -endorseddirs <dirs>       Override location of endorsed standards path
  -proc:<none,only>          Control whether annotation processing and/or compil
ation is done.
  -processor <class1>[,<class2>,<class3>...] Names of the annotation processors
to run; bypasses default discovery process
  -processorpath <path>      Specify where to find annotation processors
  -parameters                Generate metadata for reflection on method paramete
rs
  -d <directory>             Specify where to place generated class files
  -s <directory>             Specify where to place generated source files
  -h <directory>             Specify where to place generated native header file
s
  -implicit:<none,class>     Specify whether or not to generate class files for
implicitly referenced files
  -encoding <encoding>       Specify character encoding used by source files
  -source <release>          Provide source compatibility with specified release

  -target <release>          Generate class files for specific VM version
  -profile <profile>         Check that API used is available in the specified p
rofile
  -version                   Version information
  -help                      Print a synopsis of standard options
  -Akey[=value]              Options to pass to annotation processors
  -X                         Print a synopsis of nonstandard options
  -J<flag>                   Pass <flag> directly to the runtime system
  -Werror                    Terminate compilation if warnings occur
  @<filename>                Read options and filenames from file
```

Abb. 1.5: Ausgabe bei korrekt installiertem Java-Compiler

1.4 Entwicklungsumgebungen

Damit haben Sie bereits alles zusammen, was Sie zum Programmieren mit Java unbedingt benötigen. Theoretisch können Sie Java-Programme nämlich in jedem Texteditor oder sogar direkt in der Eingabeaufforderung (beziehungsweise dem Terminal) schreiben. Das ist auf Dauer allerdings sehr mühsam.

Wie es mit Microsoft Word oder OpenOffice Writer spezielle Programme zum Schreiben von Briefen und anderen Dokumenten gibt, so gibt es auch spezielle Programme zum Schreiben von Software. Häufig bieten diese, neben einem reinen Texteditor, der speziell auf die Bedürfnisse von Programmierern ausgerichtet ist, auch weitere nützliche Tools zum Entwickeln von Software, wie zum Beispiel einen integrierten Compiler. Ein solches Programm wird auch

integrierte Entwicklungsumgebung genannt, oder kurz *IDE*, vom englischen *integrated development environment*.

Zu den beliebtesten IDEs für Java zählen die Programme *Eclipse* und *IntelliJ IDEA*, die jeweils einen Marktanteil von knapp über 40% unter den Java-IDEs haben. Während es sich bei Eclipse um eine frei verfügbare Open-Source-Software handelt, gibt es von IntelliJ IDEA zwei Versionen: eine sogenannte »Community«-Edition, die ebenfalls frei verfügbar ist, und eine sogenannte »Ultimate«-Version, die kostenpflichtig ist. Beide Programme sind für alle großen Plattformen, inklusive Windows, Linux und macOS, verfügbar.

In den folgenden beiden Abschnitten wird zunächst Eclipse (*Abschnitt 1.4.1*) und dann die Community-Edition von IntelliJ IDEA (*Abschnitt 1.4.2*) kurz vorgestellt. Sie können diesem Buch aber auch folgen, wenn Sie sich entscheiden, eine andere IDE oder überhaupt keine IDE zu verwenden.

Welche der beiden IDEs die »bessere« ist, ist Gegenstand zahlreicher Diskussionen unter Java-Entwicklern und lässt sich pauschal nicht beantworten. Eclipse gilt mitunter als das mächtigere Tool, das, insbesondere dank zahlreicher Erweiterungen, viele Schritte des Entwicklungsprozesses unterstützen kann, auch solche, die weit über das eigentliche Programmieren hinausgehen. IntelliJ IDEA wird im Gegenzug häufig zugeschrieben, dass es einsteigerfreundlicher sei und weniger überladen. Für Einsteiger in die Welt des Programmierens kann daher IntelliJ IDEA besser geeignet sein, wobei diese zurzeit lediglich in englischer Sprache verfügbar ist, eine gute Wahl sind aber beide IDEs.

1.4.1 Eclipse

Die Entwicklung von Eclipse wird von der gemeinnützigen Eclipse Foundation geleitet. Auf der offiziellen Webseite *www.eclipse.org* kann das Installationsprogramm für Eclipse im Download-Bereich kostenlos heruntergeladen werden.

Installation

Nach dem Starten des Installationsprogramms erscheint zunächst der in Abbildung 1.6 gezeigte Bildschirm zur Auswahl der zu installierenden Version. Eclipse bietet nicht nur eine IDE zur Entwicklung von Java-Programmen, es gibt auch Versionen für C, PHP und andere Programmiersprachen und Einsatzzwecke.

Selbst für die Entwicklung mit Java gibt es zwei unterschiedliche Versionen: »Eclipse IDE for Java Developers« und »Eclipse IDE for Enterprise Java and Web Developers«. Für uns genügt die erste Version. Die Enterprise-Version ist insbesondere für die Entwicklung von Anwendungen, die auf mehrere Systeme verteilt werden können und dadurch besonders skalierbar sind, geeignet.

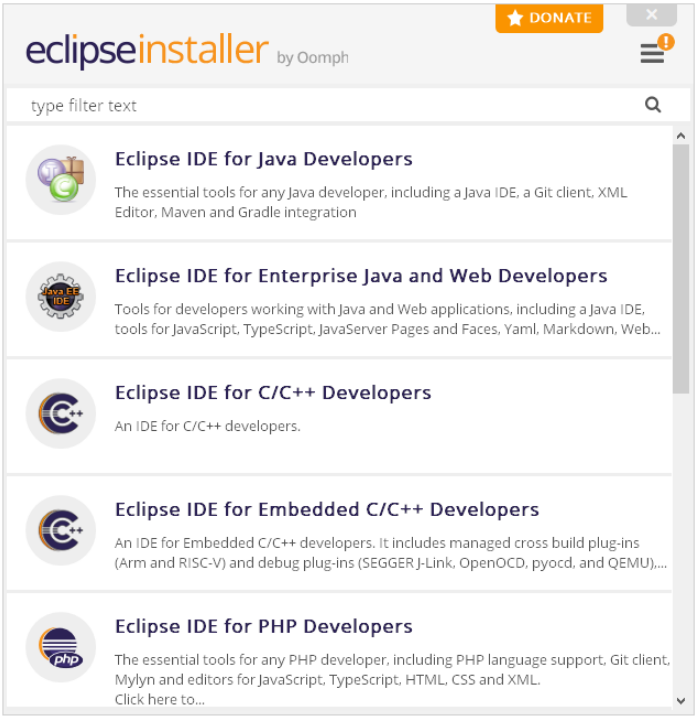

Abb. 1.6: Versionsauswahl im Eclipse-Installationsprogramm

Nach der Auswahl der korrekten Version müssen Sie nur den Anweisungen der Installationsroutine folgen, die Eclipse – und wenn nötig auch das JDK – auf Ihrem System installiert. Nach dem Abschluss der Installation kann die IDE durch einen Klick auf den LAUNCH-Button gestartet werden.

Sprache

Beim ersten Start ist Eclipse, wie das Installationsprogramm, zunächst komplett auf Englisch. Es gibt allerdings ein deutsches Sprachpaket, das nachinstalliert werden kann, und die Oberfläche übersetzt. Bevor Sie dies installieren

können, müssen Sie aber erst den Ordner für den sogenannten *Workspace* festlegen. Beim ersten Starten von Eclipse öffnet sich nach dem Ladebildschirm nämlich zunächst das in Abbildung 1.7 gezeigte Fenster, der *Eclipse Launcher*.

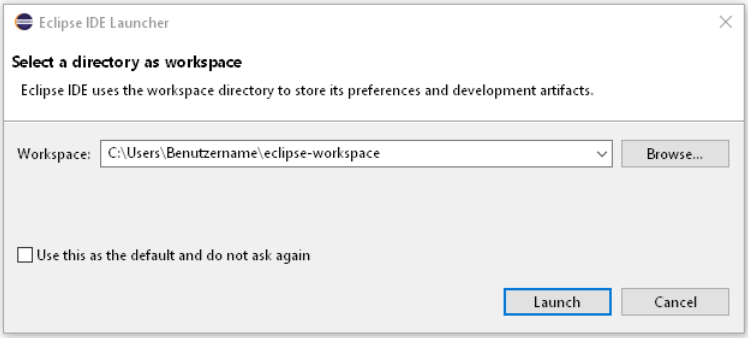

Abb. 1.7: Eclipse Launcher

Der Eclipse Launcher fragt Sie, unter welchem Pfad der Workspace angelegt werden soll. Der Workspace ist der Ordner, in dem Eclipse all Ihre Programme und Projekte speichern wird. Sollten Sie keine besondere Präferenz haben, können Sie einfach den Standardpfad verwenden, der in Ihrem Benutzerverzeichnis einen Unterordner `eclipse-workspace` anlegt und die Dateien dort abspeichert.

Bevor Sie Eclipse mit einem Klick auf den LAUNCH-Button starten, sollten Sie noch den Haken vor »Use this as the default and do not ask again« setzen, dann merkt sich Eclipse das Verzeichnis dauerhaft und fragt Sie nicht bei jedem Start des Programms erneut. Bei Bedarf können Sie den Pfad später trotzdem über die Einstellungen von Eclipse ändern.

Beim ersten Starten von Eclipse zeigt sich nun der in Abbildung 1.8 sichtbare Willkommen-Bildschirm. Um das deutsche – oder auch ein beliebiges anderes – Sprachpaket zu installieren, öffnen Sie in der oberen Navigationsleiste zunächst das Menü HELP und wählen dort den Eintrag INSTALL NEW SOFTWARE... aus.

Es öffnet sich das in Abbildung 1.9 gezeigte Installationsfenster. Dort müssen Sie nun auf den ADD-Button klicken, um eine neue Installationsquelle hinzuzufügen.

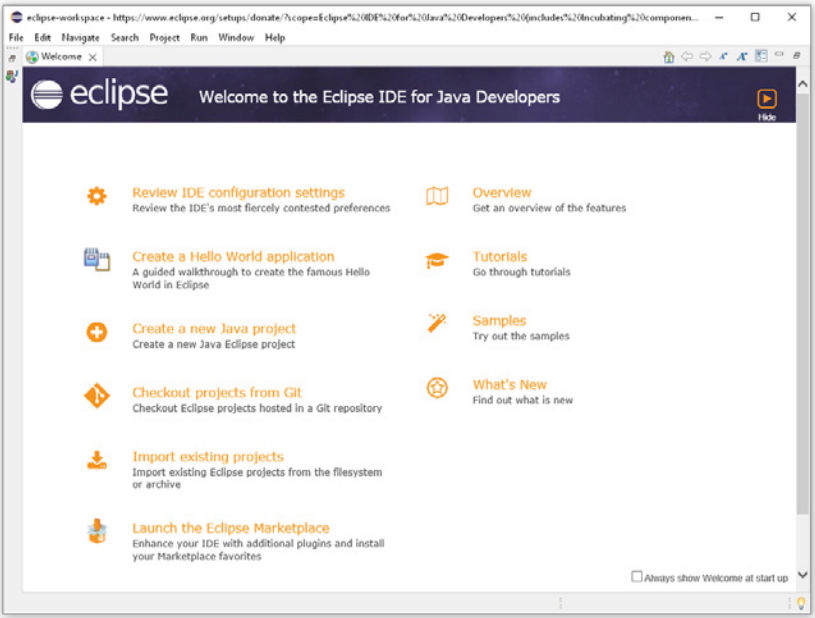

Abb. 1.8: Eclipse-Willkommen-Bildschirm

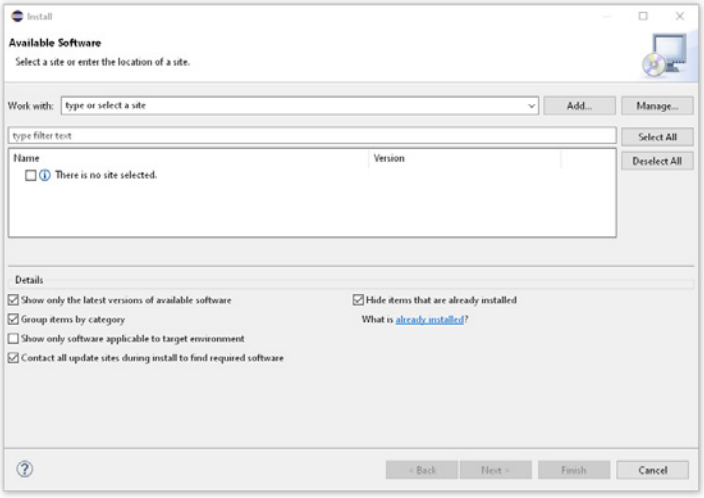

Abb. 1.9: Eclipse-Installationsfenster

Das tun Sie, indem Sie im sich öffnenden Fenster (Abbildung 1.10) unter Location eine URL angeben. Die korrekte URL für das jeweils aktuelle Sprachpaket finden Sie auf *www.eclipse.org/babel* unter »Downloads«. Zum Zeitpunkt des Verfassens dieser Zeilen lautet die URL wie folgt:

https://download.eclipse.org/technology/babel/update-site/R0.19.1/2021-09/

Die Wahrscheinlichkeit ist allerdings hoch, dass die URL zum Zeitpunkt, zu dem Sie diese Zeilen lesen, nicht mehr aktuell ist.

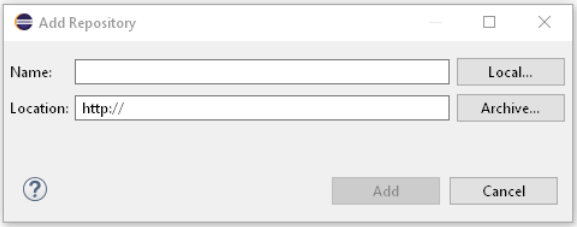

Abb. 1.10: Hinzufügen einer Installationsquelle in Eclipse

Sobald Sie die korrekte URL eingegeben haben, können Sie die Installationsquelle mit einem Klick auf den Add-Button hinzufügen. Daraufhin kehren Sie wieder zum Installationsfenster zurück, das Sie bereits aus Abbildung 1.9 kennen, allerdings finden Sie dort nun eine Liste der verfügbaren Sprachpakete.

Aus dieser Liste können Sie, wie in Abbildung 1.11 gezeigt, das passende Sprachpaket auswählen. Das Paket für die Sprache Deutsch heißt Babel Language Packs in German. Nach einem Klick auf den Button Next > wird das Sprachpaket zunächst heruntergeladen. Anschließend müssen Sie die Installation sowie die Lizenzvereinbarung noch bestätigen.

Nach einem Klick auf den Button Finish schließt sich das Fenster. Danach kann es zunächst so aussehen, als würde nichts passieren, tatsächlich wird das Sprachpaket nun aber im Hintergrund installiert, wie in der unteren Statusleiste zu sehen ist. Da die Fortschrittsanzeige leicht übersehen werden kann, ist sie in Abbildung 1.12 durch einen Pfeil hervorgehoben.

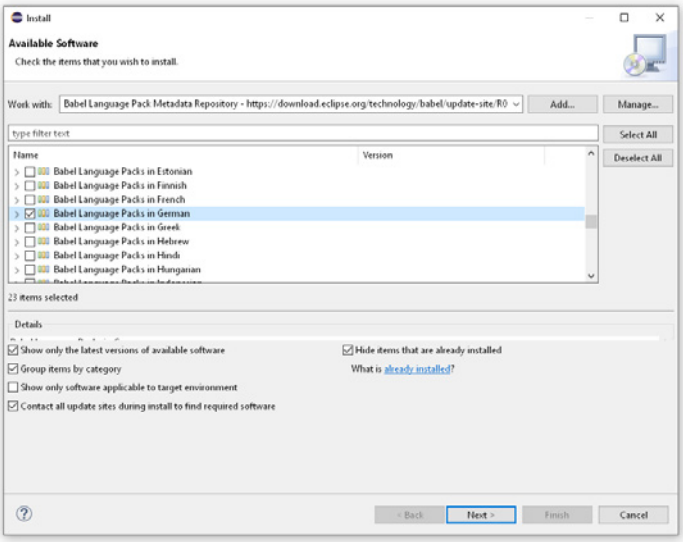

Abb. 1.11: Installationsfenster mit Sprachpaketen

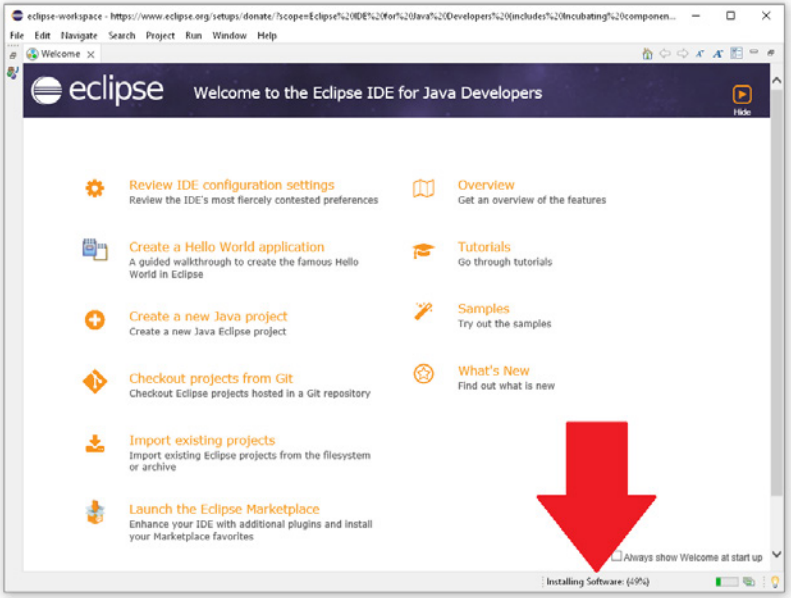

Abb. 1.12: Installation des Sprachpakets im Hintergrund, Fortschritt markiert durch Pfeil

Während der Installation kann es passieren, dass eine Sicherheitswarnung wie in Abbildung 1.13 angezeigt wird. Wenn Sie, wie zuvor beschrieben, die offizielle URL als Installationsquelle genutzt haben, können Sie hier ohne Bedenken die Installation mit einem Klick auf den Button INSTALL ANYWAY fortsetzen.

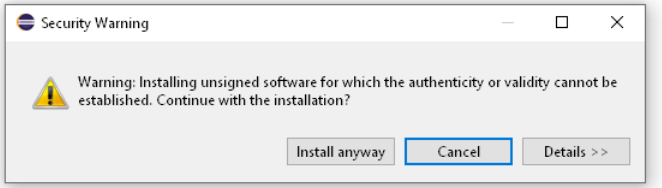

Abb. 1.13: Sicherheitswarnung während der Installation

Sobald die Installation abgeschlossen ist, muss Eclipse nur noch neu gestartet werden und ist dann ab sofort auf Deutsch verfügbar.

Neues Projekt anlegen

Nun fehlt nur noch ein wichtiger Schritt in der Vorbereitung, bevor Sie im nächsten Kapitel mit dem Programmieren beginnen können: Eclipse organisiert die Entwicklung in sogenannten *Projekten*. Ein Projekt fungiert dabei wie ein Ordner, in dem Dateien organisiert werden können, in der Projektdatei werden darüber hinaus aber auch noch weitere Einstellungen, wie zum Beispiel die verwendete Java-Version, gespeichert. Ein Projekt kann dabei nicht nur ein, sondern gleich mehrere Programme enthalten.

Um Ihr erstes Projekt zu erstellen, klicken Sie in der oberen Navigationsleiste zunächst auf das DATEI-Menü, wählen dann den Eintrag NEU aus und dort schließlich den Eintrag JAVA-PROJEKT. Daraufhin öffnet sich das in Abbildung 1.14 gezeigte Fenster. Dort können Sie zunächst im Feld PROJEKTNAME einen entsprechenden Namen vergeben – wir nehmen an dieser Stelle als Beispiel den Namen »Java-Schnelleinstieg«. Den Haken ganz am Ende des Fensters beim Eintrag *Modul* können Sie zunächst entfernen. Die restlichen Einstellungen können für den Anfang unverändert bleiben. Mit einem Klick auf FERTIGSTELLEN wird das neue Projekt erstellt.