1 Einführung

1.1 Java im Überblick

Java wurde Mitte der 1990er-Jahre von der Firma Sun entwickelt und später von Oracle übernommen. Mittlerweile hat Java zwar schon mehr als 25 Jahre auf dem Buckel, wird aber nicht altersschwach, sondern kontinuierlich gepflegt und besitzt ein extrem breitgefächertes und professionelles Angebot an externen Bibliotheken und Entwicklungstools. Insbesondere gibt es mit Eclipse, IntelliJ IDEA und NetBeans sowie Visual Studio Code mehrere hervorragende sogenannte IDEs (Integrated Development Environments) zum komfortablen Programmieren.

Java eignet sich zur Entwicklung unterschiedlichster Applikationen, etwa für Businessapplikationen, Webapplikationen und sogar einige (einfachere) Datenbanken. Es wird aber auch im Bereich Mobile in Form von Android-Apps oder gar im Bereich von Spielen, z. B. Minecraft, verwendet. Java ist also eine vielseitige Programmiersprache mit breitem Einsatzspektrum. Ein guter Grund, sich damit ein wenig zu beschäftigen.

Außerdem ist Programmieren ein wunderbares Hobby sowie ein faszinierender Beruf und es macht zudem noch jede Menge Spaß, fördert die Kreativität und den Gestaltungswillen.

Darüber hinaus ist Java laut TIOBE-Index[1] seit Jahren eine der populärsten Programmiersprachen. Eine wichtige Rolle spielte vermutlich lange die freie Verfügbarkeit. Zumindest für die Originalvariante von Oracle gilt das mittlerweile nur noch für private, nicht kommerzielle Zwecke. Praktischerweise existieren inzwischen einige frei verwendbare Alternativen wie etwa das AdoptOpenJDK[2]. Zudem ist Java recht einfach zu erlernen (schwieriger als Python, aber deutlich leichter als C++) und bietet ein großes Ökosystem an Tools, Bibliotheken und Literatur sowie Hilfestellungen wie StackOverflow im Internet. Weiterhin zeichnet sich Java durch seine gute Performance aus. Das Ganze ist die Folge von jahrelangen Optimierungen und Verbesserungen. Dadurch ist die Ausführung von Java beispielsweise nur geringfügig langsamer als die von C++, aber um Längen schneller als die Ausführung von Python. Schließlich ermöglicht Java sowohl die objektorientierte als auch die funktionale Programmierung, sodass man je nach Einsatzzweck geeignet wählen kann.

[1] `https://www.tiobe.com/tiobe-index/`

[2] `https://adoptopenjdk.net/`. Ab März 2021 wird das Ganze als Eclipse Adoptium weitergeführt: `https://projects.eclipse.org/projects/adoptium`).

Wie Sie sehen, sprechen viele gute Gründe für einen Einstieg in die Programmierung mit Java. Das Wichtigste ist jedoch der Spaß am Programmieren, Tüfteln und Ausprobieren. Lassen Sie uns starten!

Bestandteile von Java-Programmen

Java als Programmiersprache besitzt wie eine natürliche Sprache auch eine Grammatik und feststehende Begriffe / Wörter. Man spricht dabei von Syntax und Schlüsselwörtern (vgl. Anhang A).

Java-Programme werden textuell verfasst. Das wird ***Sourcecode*** genannt. Schauen wir uns zum Einstieg ein einfaches Java-Programm an:

```
public class MyFirstJavaProgram
{
    public static void main(String[] args)
    {
        System.out.println("Hello World");
    }
}
```

Keine Sorge, Sie müssen das Ganze noch nicht vollständig verstehen, wir werden das alles Stück für Stück erlernen. Hier ist zunächst nur wichtig, dass Sie elementare Bestandteile von Java-Programmen grob einordnen können. Dazu gehören die Schlüsselwörter, also Java-Befehle oder -Anweisungen, hier etwa `public`, `class`, `static` und `void`. Wie die Begriffe in einer Sprache tragen diese reservierten Wörter eine besondere Bedeutung, ganz analog etwa zu Auto, Haus, Tür usw. im Deutschen.

Ebenso wie im Deutschen können (oder besser sollten) die Begriffe nicht einfach wahllos miteinander verknüpft werden, um einen gültigen Satz zu formulieren. Das wird durch die Grammatik geregelt. Auch in Java existiert eine solche. Damit wird etwa festgelegt, dass es `static void`, aber nicht `void static` heißen muss.

Zudem sehen wir geschweifte Klammern. Diese kann man sich wie Absätze in einem Text vorstellen. In Java bündeln diese Klammern Anweisungen. Man spricht dann auch von Blöcken und Sichtbarkeitsbereichen.

Genug der Vielzahl an Informationen. Nachfolgend werden wir die Dinge schön gründlich und detailliert besprechen und didaktisch immer ein neues Themengebiet ergründen, bis wir schließlich einen guten Einstieg in die Java-Programmierung vollzogen haben werden.

Vorab wollen wir aber erst einmal Java und Eclipse installieren und erste Schritte machen, um für unsere weitere Entdeckungsreise bereit zu sein.

1.2 Los geht's – Installation

Im ersten Teil dieses Buchs wird ein Hands-on-Ansatz verfolgt, bei dem wir Dinge oftmals in Form kleinerer Java-Codeschnipsel direkt ausprobieren. Sie benötigen vorab keine tiefgreifenden Programmiererfahrungen, allerdings schaden diese natürlich nicht, ganz im Gegenteil. Hilfreich wäre allerdings, wenn Sie sich einigermaßen mit dem Installieren von Programmen und grundlegend mit der Kommandozeile auskennen.

Damit Sie die nachfolgend beschriebenen Java-Programme ausführen können, benötigen Sie ein sogenanntes JDK (Java Development Kit). Dort finden sich alle für den Moment benötigten Tools. Beginnen wir also mit der Installation von Java.

1.2.1 Java-Download

Die aktuelle Java-Version ist frei auf der folgenden Oracle-Webseite verfügbar: `http://www.oracle.com/technetwork/java/javase/downloads/index.html`

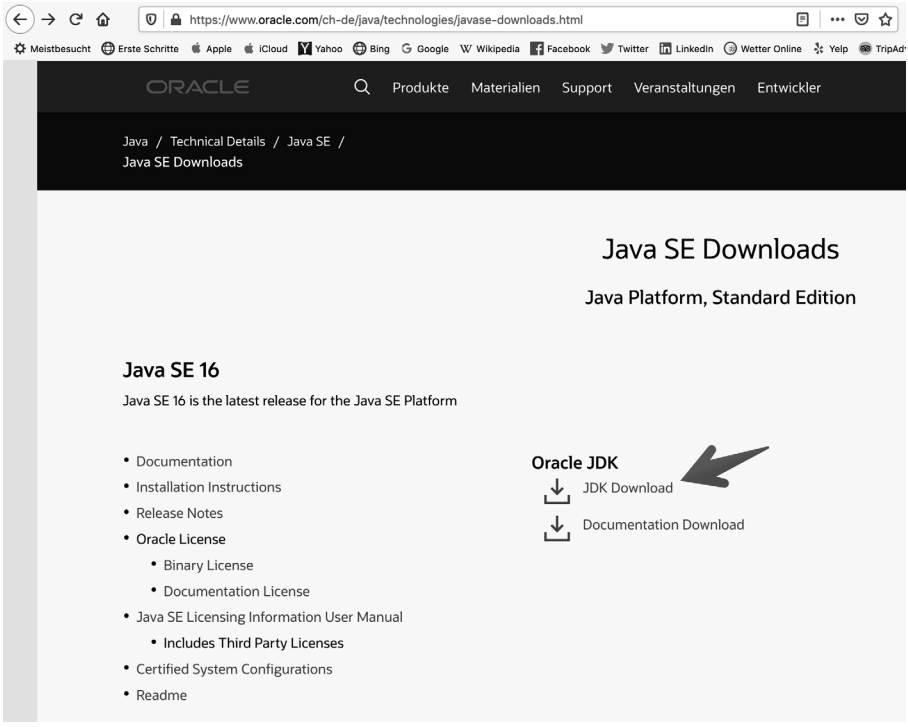

Abbildung 1-1 *Oracle-Seite für Java*

Nachdem Sie auf den Link zum JDK-Download geklickt haben, erscheint in etwa eine Darstellung wie die folgende:

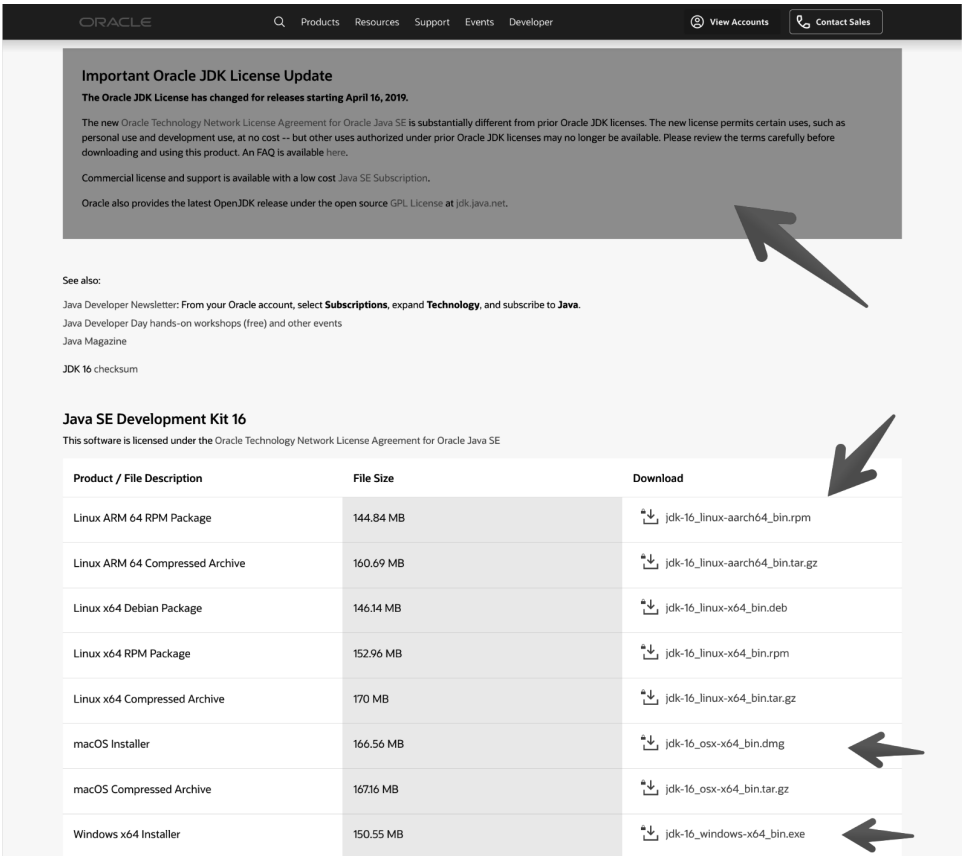

Abbildung 1-2 *Java-Download-Seite*

Im oberen Bereich sehen wir einige Hinweise zur neuen Lizenzpolitik, die ich nachfolgend im Hinweiskasten thematisiere. Im unteren Bereich finden Sie verschiedene Links für unterschiedliche Betriebssysteme. Wählen Sie den für Sie passenden Link.

Hinweis: Neue Lizenzpolitik bei Oracle

Wenn Sie Ihre Software kommerziell vertreiben oder dies planen, dann sollten Sie beim Herunterladen von Java unbedingt die Lizenzpolitik von Oracle beachten: **Das bis Java 8 selbst in Produktionssystemen immer kostenfrei verwendbare Oracle JDK ist dafür nun leider kostenpflichtig**. Als Alternative können Sie auf das OpenJDK (`https://openjdk.java.net/`) ausweichen. Für dieses Buch das Wichtigste: ***Für private Projekte und während der Entwicklung kann das Oracle JDK weiterhin kostenfrei genutzt werden***.

1.2.2 Installation des JDKs

Unter MacOS doppelklicken Sie auf die `.dmg`-Datei, um die Installationsdatei zu starten, und folgen den Aufforderungen. Möglicherweise müssen Sie das Administrator-Passwort eingeben, um fortzufahren. Nachdem die Installation abgeschlossen ist, können Sie die `.dmg`-Datei löschen, um Speicherplatz zu sparen.

Für Windows doppelklicken Sie bitte auf die `.exe`-Datei. Auch diese kann nach erfolgreicher Installation gelöscht werden. Führen Sie also das heruntergeladene Installationsprogramm aus (z. B. `jdk-16.0.1_windows-x64_bin.exe`). Damit wird Java installiert – standardmäßig ins Verzeichnis `C:\Programme\Java\jdk-16.0.1`, wobei der Verzeichnisname von der gewählten Version abhängt. Akzeptieren Sie die Standardeinstellungen und befolgen Sie die Anweisungen während der Installation.

1.2.3 Nacharbeiten nach der Java-Installation

Damit Java bei Ihnen nach dem Download und der Installation auch in der Konsole korrekt funktioniert, sind noch ein paar Nacharbeiten nötig. Dazu müssen wir es zur leichteren Handhabung in den Pfad aufnehmen. Dies wird im Anschluss für die weitverbreiteten Betriebssysteme Windows und MacOS beschrieben. Falls Sie ein Unix-Derivat nutzen, dann finden Sie weitere Informationen auf dieser Seite: `https://www.java.com/de/download/help/download_options.html`. Für Windows und Mac gibt es dort auch noch einige ergänzende Informationen.

Nacharbeiten für Windows

Das Installationsverzeichnis muss in die Umgebungsvariable `PATH` aufgenommen werden. Diese können Sie unter »Umgebungsvariablen« ändern. Drücken Sie die Win-Taste und geben Sie dann »umgeb« ein, bis »Systemumgebungsvariablen bearbeiten« erscheint. Mit Enter erscheint der Dialog »Systemvariablen«. Klicken Sie auf den Button »Bearbeiten« zum Öffnen eines Bearbeitungsdialogs. Fügen Sie in der Liste das Installationsverzeichnis gefolgt von `bin`, etwa `C:\Programme\Java\jdk-16.0.1\bin`, hinzu. Um nicht den gesamten Pfad eingeben zu müssen, bietet es sich an, eine weitere Umgebungsvariable namens `JAVA_HOME` anzulegen.

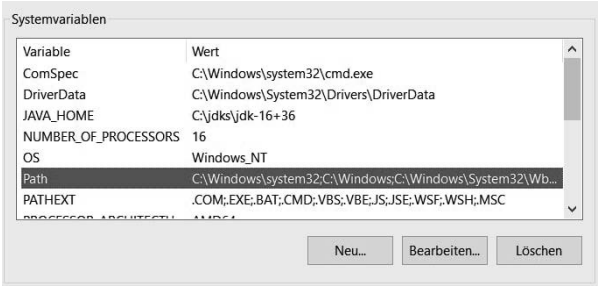

Abbildung 1-3 *Umgebungsvariablen bearbeiten*

Außerdem sollte der Eintrag möglichst ganz oben stehen:

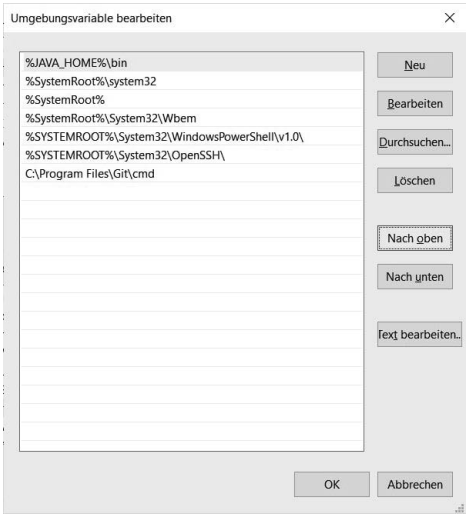

Abbildung 1-4 *Umgebungsvariablen ordnen*

Beachten Sie bitte noch Folgendes: Bestätigen Sie die gesamten Dialoge bitte immer mit OK, sodass die Variablen gesetzt sind. Eventuell geöffnete Konsolen müssen geschlossen und neu geöffnet werden, um die geänderten Variablen wirksam werden zu lassen.

Nacharbeiten für MacOS

Auch unter MacOS empfiehlt es sich, einen Verweis auf Java im Pfad in der jeweiligen Shell (dem Terminal) passend zu setzen.

```
export JAVA_HOME=/Library/Java/JavaVirtualMachines/jdk-16.jdk/Contents/Home
export PATH=$JAVA_HOME/bin:$PATH
```

1.2.4 Java-Installation prüfen

Nach dem Ausführen der obigen Schritte sollte Java auf Ihrem Rechner installiert und von der Konsole startbar sein und Sie damit bereit für die nächsten Schritte.

Öffnen Sie eine Konsole und geben Sie folgendes Kommando ein – im folgenden Text nutze ich immer $ zur Kennzeichnung von Eingaben auf der Konsole, also dem Terminal bei MacOS bzw. der Windows-Eingabeaufforderung:

```
$ java --version
java 16 2021-03-16
Java(TM) SE Runtime Environment (build 16+36-2231)
Java HotSpot(TM) 64-Bit Server VM (build 16+36-2231, mixed mode, sharing)
```

JShell prüfen

Prüfen Sie der Vollständigkeit halber bitte auch noch den Aufruf sowie das Beenden des Tools JShell, das wir für den ersten Teil des Buchs intensiv nutzen werden:

```
$ jshell
|  Welcome to JShell -- Version 16
|  For an introduction type: /help intro

jshell> /exit
|  Goodbye
```

Wenn die Programme bzw. Tools starten und Sie ähnliche Meldungen erhalten (möglicherweise mit kleinen Abweichungen bei den Versionsangaben), so können wir uns auf die Entdeckungsreise zur Java-Programmierung machen.

1.3 Entwicklungsumgebungen

Zum Schreiben von umfangreicheren Java-Programmen (also zum Bearbeiten von viel Sourcecode) empfehle ich den Einsatz einer IDE anstelle von Texteditoren oder anstatt rein auf der Konsole in der JShell zu arbeiten. Für kleine Experimente ist aber gerade die JShell ein wunderbares Hilfsmittel.

Für Änderungen an größeren Java-Programmen kann man zwar auch mal einen Texteditor nutzen, aber dieser bietet nicht die Annehmlichkeiten einer IDE: In IDEs laufen verschiedene Aktionen und Sourcecode-Analysen automatisch und im Hintergrund ab, wodurch gewisse Softwaredefekte direkt noch während des Editierens erkannt und angezeigt werden können, etwa in einer To-do-/Task-Liste. IDEs bereiten zudem vielfältige Informationen auf. Weiterhin werden diverse Annehmlichkeiten wie Quick Fixes zur Korrektur kleinerer Probleme sowie automatische Transformationen und Änderungen von Sourcecode, sogenannte ***Refactorings***, unterstützt.

Für Java existieren verschiedene IDEs. Sowohl Eclipse als auch NetBeans sind kostenlos. In den letzten Monaten wird auch das frei verfügbare Visual Studio Code von Microsoft immer populärer. Schließlich gibt es IntelliJ IDEA als kostenlose Community Edition sowie als kostenpflichtige Ultimate Edition. Alle IDEs haben ihre speziellen Vorzüge, aber auch (kleinere) Schwächen.

Für dieses Buch werden wir Eclipse nutzen. Wenn Sie bereits etwas Erfahrung haben, dann sind Sie natürlich frei, sich die anderen IDEs anzuschauen und auszuprobieren. Vieles geht über persönliche Präferenzen. Entscheiden Sie also später selbst und besuchen Sie dazu folgende Internetadressen:

- `https://www.eclipse.org/`
- `https://www.jetbrains.com/idea/`
- `https://netbeans.apache.org/`
- `https://code.visualstudio.com/`

1.3.1 Installation von Eclipse

Öffnen Sie einen Browser und gehen Sie auf die Seite `www.eclipse.org`. Diese präsentiert sich ähnlich zu Abbildung 1-5. Dort finden Sie oben rechts einen Download-Button, den Sie bitte drücken.

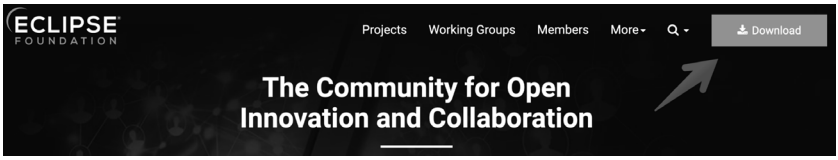

Abbildung 1-5 *Eclipse-Hauptseite zum Download*

Dadurch wird die Download-Seite geöffnet. Dort sehen Sie verschiedene Möglichkeiten, bitte klicken Sie auf »Download Packages«.

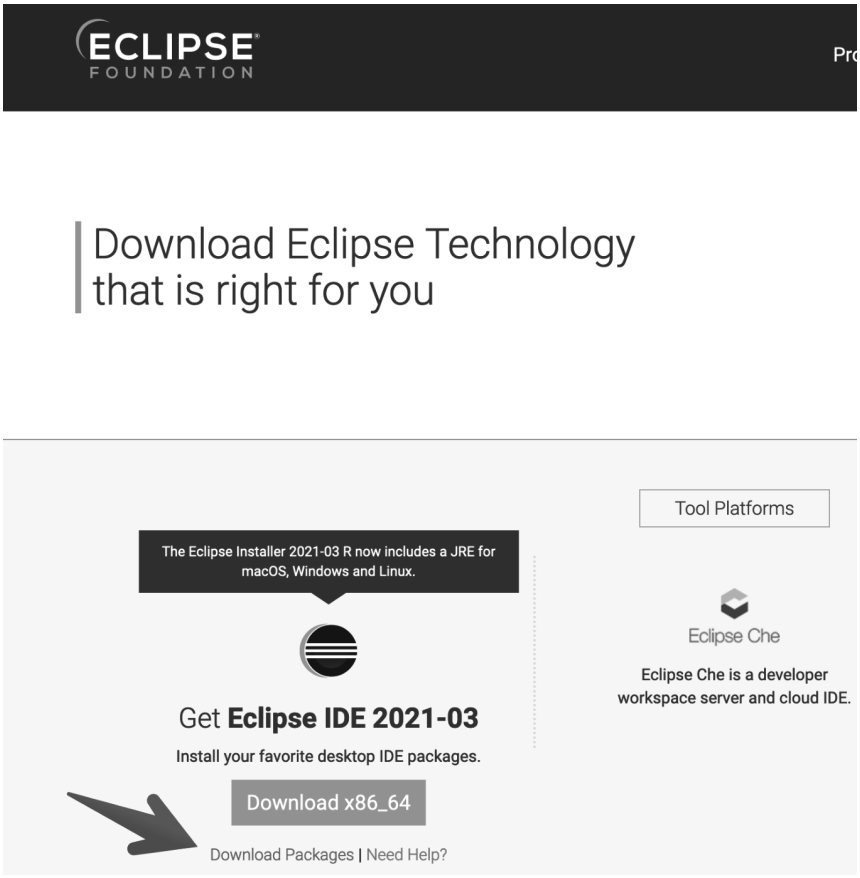

Abbildung 1-6 *Eclipse-Installation »Download Packages«*

Nun sollte sich die Seite mit den verfügbaren Packages öffnen. Diese sieht ähnlich zu Abbildung 1-7 aus. Dort wählen Sie abhängig von Ihrem Betriebssystem die passende Version und klicken diese zum Download an.

Abbildung 1-7 *Betriebssystemspezifische Eclipse-Versionen*

Nachdem der Download abgeschlossen ist, entpacken oder starten Sie bitte das heruntergeladene ZIP, DMG oder die Datei im jeweiligen Linux-Format. Im Fall von Windows müssen Sie das entpackte Archiv noch in Ihren Programme-Ordner kopieren.

Auf der gezeigten Webseite wird der Eclipse Installer in einem Kasten angepriesen. Dieser ist sicher eine gute Alternative.

1.3.2 Eclipse starten

Nach den beschriebenen Installationsschritten sollte Ihnen nun Eclipse als Programm im Start-Menü bzw. der Programmauswahl zur Verfügung stehen. Starten Sie es bitte, etwa durch einen Doppelklick auf das Programm-Icon.

Bei MacOS erhalten Sie gegebenenfalls noch einen Warnhinweis, dass es sich um ein aus dem Internet heruntergeladenes Programm handelt. Dies können Sie ignorieren und durch einen Klick auf Öffnen fortfahren.

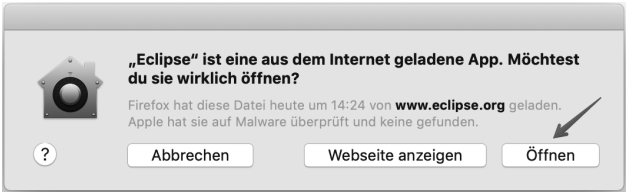

Abbildung 1-8 *Warnmeldung (MacOS)*

Als Nächstes wird ein Dialog angezeigt, um das Hauptverzeichnis für Ihre Projekte festzulegen. Sie können hier den Standard akzeptieren und ein Klick auf »Launch« startet dann Eclipse.

Abbildung 1-9 *Hauptverzeichnis festlegen*

Nun öffnet sich der Startbildschirm von Eclipse, der sich je nach Version leicht unterschiedlich präsentiert.

Möglicherweise erscheint folgendes Bild, wobei Sie mit einem Klick auf »Hide« dann auf dem Hauptbildschirm landen.

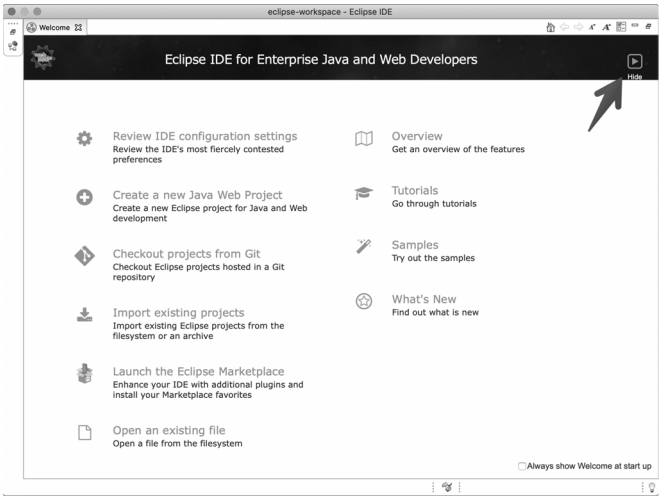

Abbildung 1-10 *Startbildschirm von Eclipse*

Der Hauptbildschirm präsentiert Ihnen verschiedene Möglichkeiten, um neue Projekte anzulegen oder bestehende zu importieren.

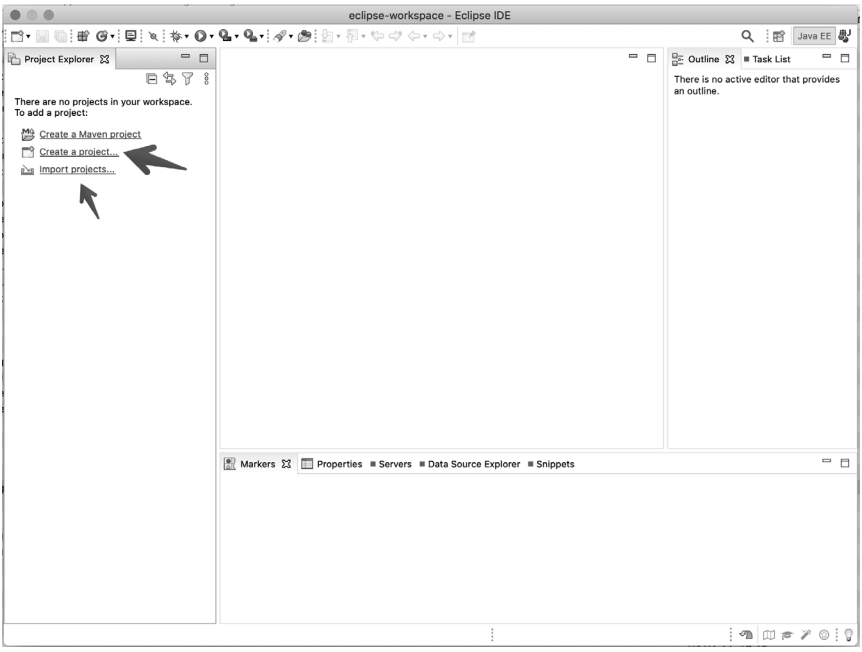

Abbildung 1-11 *Hauptbildschirm*

Das Importieren wird separat in der Beschreibung zum Download der Sourcen zu diesem Buch thematisiert. Wir schauen uns nachfolgend das Anlegen eines Projekts mitsamt einer einfachen Klasse sowie deren Start an.

1.3.3 Erstes Projekt in Eclipse

Nun wollen wir uns dem Anlegen eines Projekts und einer ersten Klasse widmen, um die Abläufe exemplarisch einmal durchgespielt zu haben, die für spätere Aktionen notwendig sind. Generell können Sie viele der hier gezeigten Aktionen auch zum Nachvollziehen der für die JShell gezeigten Programmschnipsel nutzen – dann als Teil einer Klasse und deren `main()`-Methode.

Keine Sorge, Sie müssen diese Schritte nicht im Detail verstehen. Es baut sich aber ein erstes Verständnis auf, das sich dann weiter vertieft, wenn Sie dies häufiger machen.

Beginnen wir mit dem Anlegen eines Java-Projekts. Dazu dient ein Wizard, also eine geführte Abfolge von Dialogen. Zunächst wählen wir »Java Project« und klicken dann auf »Next«.

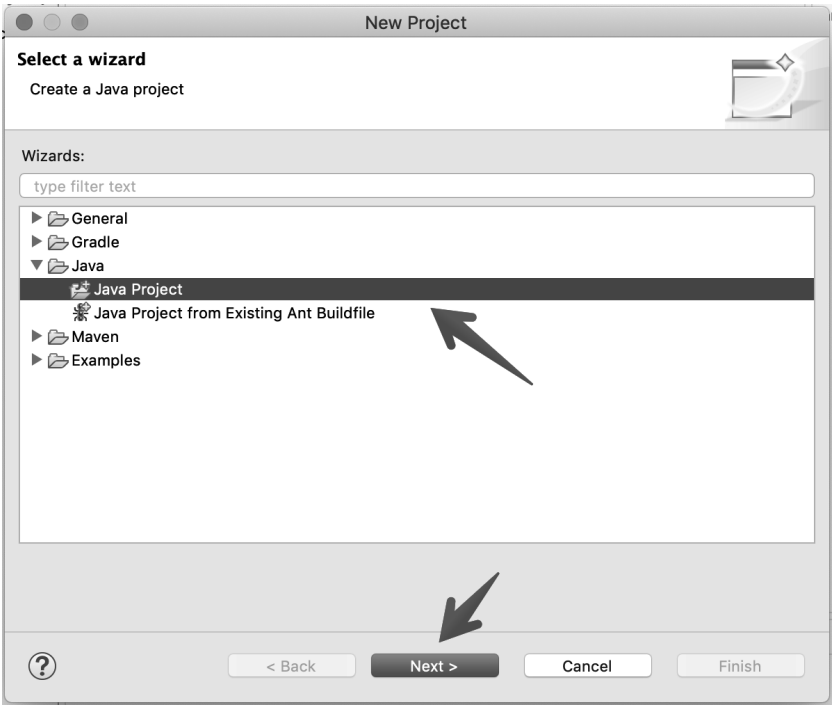

Abbildung 1-12 *Anlegen eines neuen Java-Projekts*

Damit landen wir auf der nächsten Seite zum Anlegen eines Java-Projekts mitsamt der Eingabe des Namens sowie der Auswahl der passenden Java-Version – beides durch Pfeile gekennzeichnet.

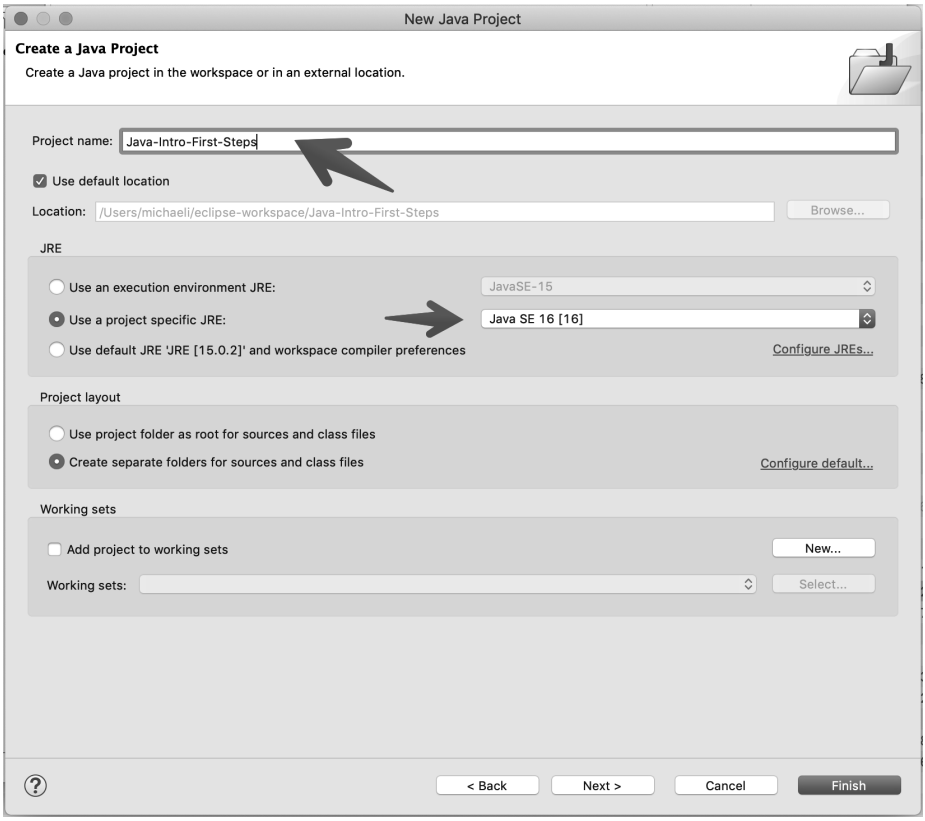

Abbildung 1-13 *Dialog »New Java Project«*

Danach wird ein Dialog angezeigt, um ein korrespondierendes Modul anzulegen. Das wollen wir nicht tun, somit geht es durch einen Klick auf »Don't Create« weiter.

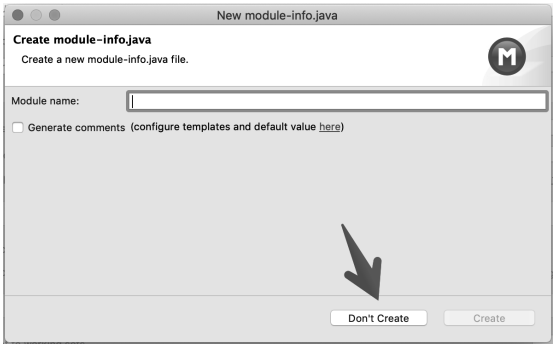

Abbildung 1-14 *Frage nach Modulerzeugung*

Es erscheint abschließend die Frage, ob die Java-Perspektive geöffnet werden soll. Das ist praktisch, um gleich die nächsten Schritte ausführen zu können.

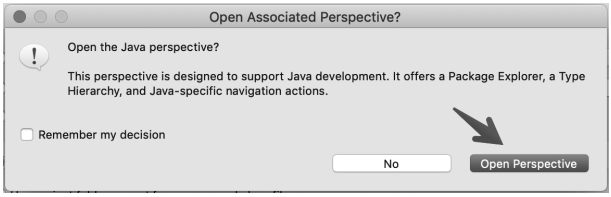

Abbildung 1-15 *Frage nach Öffnen der Java-Perspektive*

1.3.4 Erste Klasse in Eclipse

Unser erstes Java-Projekt ist jetzt angelegt und ist bereit, um mit Leben in Form von Klassen gefüllt zu werden. Ausgangspunkt zum Anlegen einer Klasse ist wieder die Baumdarstellung auf der rechten Seite im Package Explorer. Dort öffnen wir ein Kontextmenü und wählen den Eintrag `New > Class`.

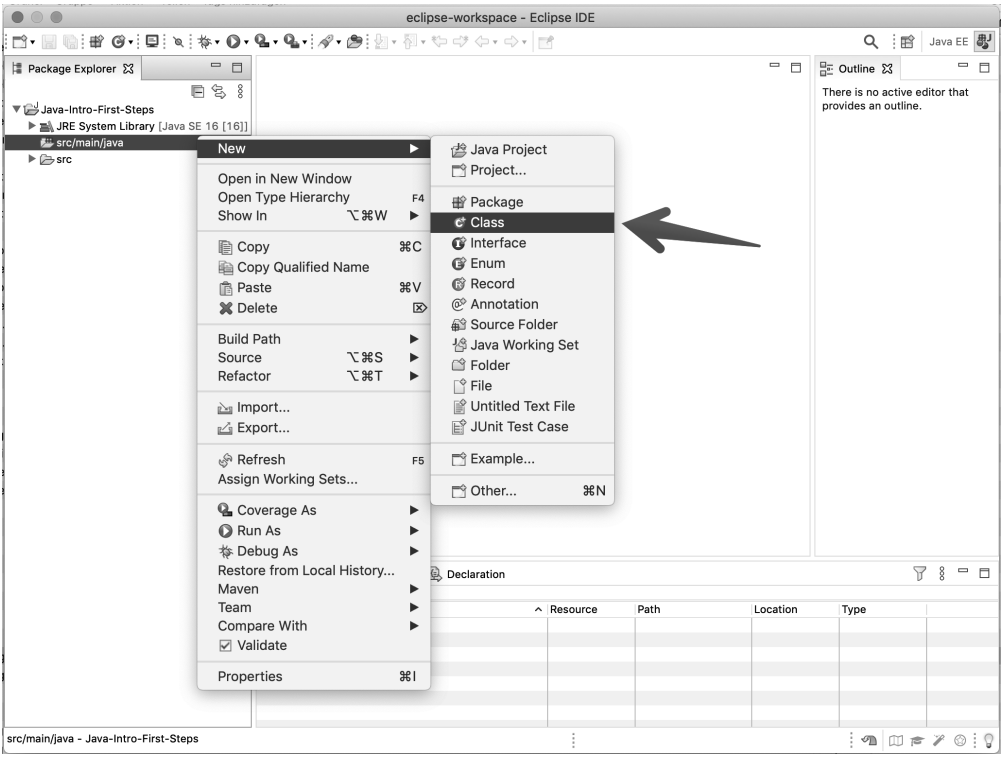

Abbildung 1-16 *Kontextmenü zum Anlegen einer Klasse*

Durch Auswahl im Kontextmenü öffnet sich folgender Dialog zum Erzeugen einer Klasse.

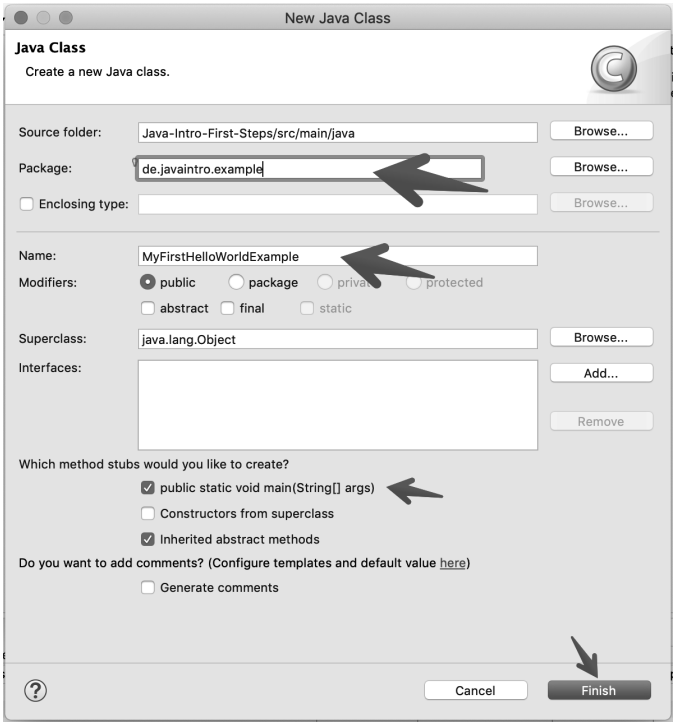

Abbildung 1-17 *Dialog »New Java Class«*

Im Dialog muss man das gewünschte Package sowie den Klassennamen eingeben. Bei Bedarf kann man das Erzeugen einer `main()`-Methode per Checkbox aktivieren. Ein Klick auf»Finish« erzeugt schließlich eine neue Klasse.

Nachdem das Grundgerüst steht, können Sie dann den Sourcecode aus den Beispielen im Editor einfügen bzw. abtippen.

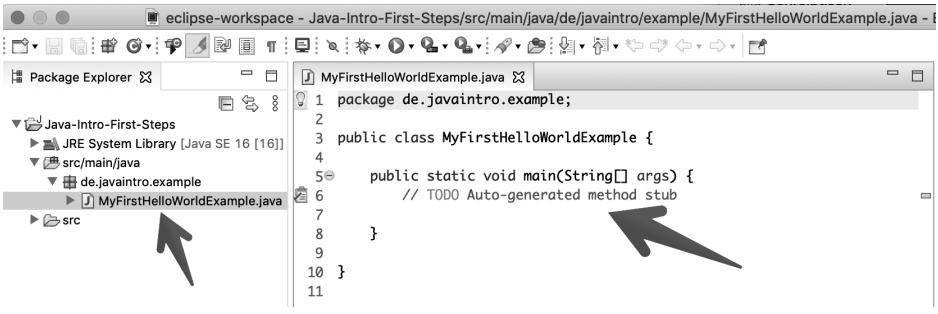

Abbildung 1-18 *Sourcecode editieren*

Schließlich wollen Sie sicherlich das so entstandene Java-Programm auch einmal in Aktion erleben. Dazu benötigt es bekanntermaßen eine `main()`-Methode. Mithilfe des Kontextmenüs oder des grünen Knopfs mit weißem Play-Pfeil kann man dann die Ausführung starten.

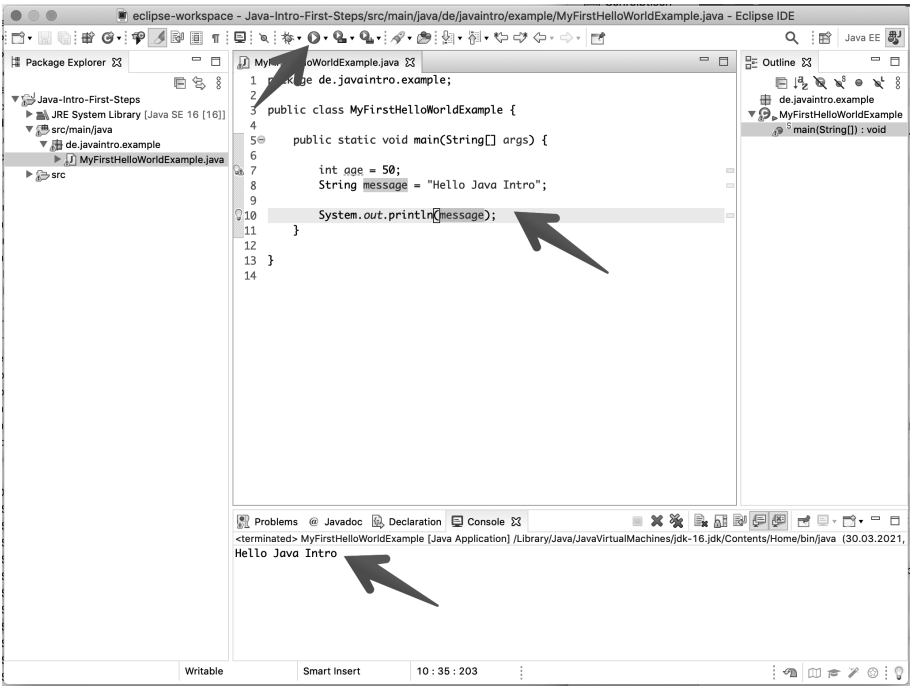

Abbildung 1-19 *Programm ausführen*