thomas KÜNNETH

# ECLIPSE

Kennenlernen.
Verstehen.
Effizient nutzen.

HANSER


Im Internet: Beispielcodes zum Download

Künneth

Eclipse

Thomas Künneth

# Eclipse

## Kennenlernen. Verstehen. Effizient nutzen.

HANSER

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.


Lektorat: Brigitte Bauer-Schiewek
Copy editing: Petra Kienle, Fürstenfeldbruck
Umschlagdesign: Marc Müller-Bremer, München, www.rebranding.de
Umschlagrealisation: Stephan Rönigk
Gesamtherstellung: Kösel, Krugzell
Ausstattung patentrechtlich geschützt: Kösel FD 351, Patent-Nr. 0748702
Printed in Germany

Print-ISBN: 978-3-446-45466-8
E-Book-ISBN: 978-3-446-45729-4

*Für Moni*

# Inhalt

# Vorwort

Nach Callisto, Europa, Ganymede, Galileo, Helios, Indigo, Juno, Kepler, Luna, Mars, Neon und Oxygen ist am 27. Juni 2018 mit Photon das 13. Simultaneous Release von Eclipse erschienen. Mit diesem Buch möchte ich Sie praxisgerecht und ohne störendes Beiwerk in die Bedienung dieser mächtigen Entwicklungsumgebung einführen. Sie lernen Eclipse so kennen, wie es sich aktuell präsentiert. Nicht nur Eclipse entwickelt sich stetig weiter, auch Java wird kontinuierlich aktualisiert. Die Entwicklungsumgebung ist bestens für Java 10 gerüstet. Deshalb basiert mein Buch auf dieser Version.

In den letzten Jahren hat sich JavaScript zu einer der am häufigsten für den Bau von Frontends genutzten Programmiersprache entwickelt. Auch wenn Sie die Geschäftslogik Ihrer Anwendungen in Java realisieren, ist es für Sie sicher interessant zu lernen, wie Sie Eclipse auch in diesem Bereich möglichst optimal nutzen können. Deshalb zeige ich Ihnen, wie Sie JavaScript und Angular einsetzen, um die Benutzeroberfläche für den Zugriff auf einen Webservice zu implementieren.

Ich wünsche Ihnen viel Freude bei der Arbeit mit diesem Buch!

Nürnberg, Frühjahr 2019

*Thomas Künneth*

# Einleitung

Im Laufe der Jahre hat sich Eclipse zu einer der bedeutendsten Entwicklungsumgebungen für die Java-Programmierung entwickelt. Sehr wahrscheinlich haben Sie dieses Buch zur Hand genommen, weil Sie sich für diesen Aspekt des Produkts interessieren.

Eclipse ist aber auch der Name eines Projekts, das sich ganz allgemein der Schaffung von Open-Source-Tools und -Frameworks verschrieben hat, unabhängig von bestimmten Sprachen oder Technologien. Es fasst zahlreiche Teilprojekte, die letztlich die eigentlichen Werkzeuge (beispielsweise die Java-IDE) implementieren, unter einem Dach zusammen. Die aus dem Eclipse Consortium hervorgegangene Eclipse Foundation, eine gemeinnützige Organisation, kümmert sich seit 2003 um die Weiterentwicklung von Eclipse.

Eclipse ist der Nachfolger von Visual Age for Java. Visual Age[1] war eine integrierte Entwicklungsumgebung, die für zahlreiche Plattformen und Programmiersprachen zur Verfügung stand. Die meisten Mitglieder der Visual-Age-Produktlinie wurden in Smalltalk realisiert. Die IBM-Version dieser Sprache wurde von der ehemals eigenständigen Firma Object Technology International (OTI) implementiert. OTI (die mittlerweile in IBM aufgegangen war) realisierte auch die erste Version von Eclipse. Im November 2001 wurde das bereits angesprochene Konsortium gegründet, das die Weiterentwicklung als quelloffene Software vorantreiben sollte.

In den ersten Versionen war Eclipse hauptsächlich eine durch Plug-ins erweiterbare Java-IDE. Mit der Version 3.0 wurde ein Paradigmenwechsel vollzogen. Seitdem bildet eine Implementierung der OSGi Service Platform das Fundament der Eclipse-Architektur und stellt Plug-ins als gleichberechtigte Einheiten in das Zentrum der Plattform.

Verschiedenste Projekte nutzen diese Infrastruktur und den durch die Eclipse-Plattform zur Verfügung gestellten Werkzeugkasten. Es ist einleuchtend, dass solche Abhängigkeiten zu Problemen in Form von Inkompatibilitäten führen können, wenn sich die einzelnen Teile unterschiedlich schnell entwickeln. Aus diesem Grund gibt es seit 2006 sogenannte Simultaneous Releases, die Schlüsselprojekte in miteinander kompatiblen Versionen bündeln. Die folgende Tabelle zeigt das Datum der Veröffentlichung und die Zahl der teilnehmenden Projekte.

[1] *http://en.wikipedia.org/wiki/VisualAge*

Bislang erschienene Eclipse-Versionen:

| Name | Version | Veröffentlichungsdatum | Projekte |
|---|---|---|---|
| Callisto | 3.2 | 30.06.2006 | 10 |
| Europa | 3.3 | 29.06.2007 | 21 |
| Ganymede | 3.4 | 25.06.2008 | 23 |
| Galileo | 3.5 | 24.06.2009 | 33 |
| Helios | 3.6 | 23.06.2010 | 39 |
| Indigo | 3.7 | 22.06.2011 | 62 |
| Juno | 4.2/3.8 | 27.06.2012 | 71 |
| Kepler | 4.3 | 26.06.2013 | 72 |
| Luna | 4.4 | 25.06.2014 | 76 |
| Mars | 4.5 | 24.06.2015 | 79 |
| Neon | 4.6 | 22.06.2016 | 84 |
| Oxygen | 4.7 | 28.06.2017 | 82 |
| Photon | 4.8 | 27.06.2018 | 85 |

Vielleicht fragen Sie sich, wie es zu den Namen der einzelnen Releases kam. Das englische Wort eclipse bedeutet Finsternis oder Verdunkelung. Die Astronomie kennt Sonnen- und Mondfinsternisse. Insofern „zwingt“ die IDE selbst fast schon dazu, Begriffe im Zusammenhang mit Planeten und Sternen zu verwenden. Kallisto, Europa und Ganymed sind Jupitermonde. Ihr Entdecker (Galileo Galilei) wird mit Eclipse 3.5 geehrt. Es folgen Helios, Indigo, Juno usw. Ab Galileo entsprechen die Anfangsbuchstaben eines Releases der Reihenfolge im Alphabet. Eine Besonderheit an Juno war übrigens, dass zum ersten Mal die Eclipse-Plattform-Version 4 ausgerollt wurde. Sie enthielt eine ganze Reihe tiefgreifender Veränderungen. Um die Kompatibilität mit bestehenden Plug-ins zu gewährleisten, stand eine Version auf der Basis von Eclipse 3.8 zum Download bereit. Mittlerweile ist die Plattform-Version 4 stabil und schnell.

Die Zahl der Projekte eines Sammel-Releases macht deutlich, wie viele Bereiche der Softwareentwicklung Eclipse abdecken kann. Vielleicht flößt diese Tatsache aber auch ein wenig Furcht ein. Ist dieser Werkzeugkasten überhaupt noch beherrschbar? Findet man sich im Dschungel der Plug-ins und Projekte noch zurecht?

## Über dieses Buch

In diesem Buch stelle ich Ihnen Eclipse 4.8 als eine Entwicklungsumgebung für Java-Programmierer vor. Zwar lassen sich grundlegende Konzepte wie Perspektiven oder Sichten auch auf andere Einsatzgebiete übertragen, auf das Schreiben von Programmen in anderen Sprachen wie C oder C++ gehe ich allerdings nicht ein. Eine Ausnahme bildet das abschließende Kapitel 8, „Webanwendungen mit Angular“. Hier schlage ich den Bogen zu moderner Webentwicklung mit den dort dominierenden Sprachen JavaScript und TypeScript.

Das Ziel dieses Buchs ist, Sie in den folgenden acht Kapiteln mit den Möglichkeiten vertraut zu machen, die Ihnen die IDE bietet. Allerdings ist dieses Buch kein Java-Lehrgang. Ich werde also Konzepte wie Fehlersuche oder Refactoring nur so weit vorstellen, wie es für das weitere Verständnis erforderlich ist. Mein Hauptaugenmerk liegt darauf, Ihnen zu zeigen,

wie Sie mit Eclipse effizient arbeiten können. Selbstverständlich müssen Sie kein Java-Experte sein, um meinen Erklärungen folgen zu können. Grundlegende Kenntnisse der Programmiersprache sollten Sie aber mitbringen.

Die einzelnen Kapitel bilden in sich geschlossene Einheiten. Sie müssen das Buch also nicht von Buchdeckel zu Buchdeckel durchlesen, sondern können gezielt Bereiche auswählen, über die Sie sich ausführlicher informieren möchten. Eine Ausnahme bildet *Kapitel 1, „Hands-on Eclipse“*. Hier wird beschrieben, wie Sie die Java-Laufzeitumgebung, das Java Development Kit und Eclipse auf Ihrem Rechner einrichten (jeweils für die Betriebssysteme Windows, Linux und macOS). Im zweiten Teil dieses Kapitels legen Sie Ihr erstes Projekt an, schreiben ein kleines Programm und führen es aus. Schließlich machen Sie einen kurzen Rundgang durch die IDE und lernen das Eclipse-Hilfesystem kennen.

In *Kapitel 2, „Arbeiten mit Eclipse“*, mache ich Sie mit wichtigen Grundlagen vertraut. Sie lernen die Workbench, Perspektiven und Sichten kennen. Außerdem erfahren Sie, wie Sie komfortabel Java-Programme eingeben und auch in umfangreichen Quelltexten den Überblick behalten. Mehr und mehr an Bedeutung gewinnt das sogenannte Refactoring. Was es damit auf sich hat und wie Sie Eclipse hierbei unterstützt, wird in Abschnitt 2.3, „Suchen, ersetzen und umgestalten“, beschrieben.

Auch in *Kapitel 3, „Arbeitsbereiche und Projekte“*, lernen Sie wichtige Konzepte der IDE kennen. Ich stelle Ihnen den Arbeitsbereich als eine Art Container für Projekte vor und mache Sie mit der Projektverwaltung von Eclipse vertraut. In diesem Zusammenhang lernen Sie beispielsweise auch, wie Sie externe Tools einbinden.

Praktisch genauso wichtig wie die eigentliche Implementierung sind Fehlersuche und Testen. Im gleichnamigen *Kapitel 4, „Fehlersuche und Test“*, lernen Sie daher die Architektur des Eclipse Debuggers kennen. Außerdem zeige ich Ihnen hier, wie Sie die unzähligen Möglichkeiten, die Ihnen diese Komponente bietet, effizient einsetzen, um Fehler einzukreisen und zu eliminieren. Die am leichtesten zu behebenden Fehler sind diejenigen, die gar nicht erst gemacht werden. Unit Tests helfen, schon während der Implementierung die Korrektheit einzelner Programmteile sicherzustellen. Auch diesen Themenkomplex beschreibe ich ausführlich.

In *Kapitel 5, „Im Team arbeiten“*, stelle ich Ihnen das derzeit am häufigsten eingesetzte System zur Versionsverwaltung vor: Git. Sie erfahren unter anderem, wie Sie auf bestehende Repositories zugreifen. Selbstverständlich lernen Sie auch den Umgang mit den wichtigen Konzepten Branching und Merging innerhalb von Eclipse.

Die drei letzten Kapitel dieses Buchs greifen die bisher vermittelten Grundlagen auf und beschreiben die Entwicklung eines Webservices, sowie dessen Nutzung. *Kapitel 6, „Webservices mit Spring Boot“*, zeigt Ihnen, wie Sie die mächtige Plugin-Sammlung Spring Tool Suite installieren, und mit ihrer Hilfe einen Spring-Boot-basierten Zahlenrateservice programmieren. Quasi nebenbei werfen Sie auch einen Blick auf den Test von RESTful Webservices.

*Kapitel 7, „Eine JavaFX-Anwendung entwickeln“*, stellt Ihnen e(fx)clipse vor, mit dessen Hilfe Sie in Eclipse JavaFX-Anwendungen erstellen und ausführen können. Sie entwickeln ein kleines Programm, das den Webservice aus Kapitel 6 konsumiert.

Im abschließenden *Kapitel 8, „Webanwendungen mit Angular“*, wenden wir uns der modernen Webentwicklung zu. Sie entwickeln auch hier eine Anwendung, die den Zahlenrateservice aus Kapitel 6 aufruft. Sie ist aber mit Angular erstellt.

## Danksagung

Dieses Buch hätte ohne die freundliche Unterstützung vieler Menschen nicht entstehen können. Mein herzliches Dankeschön gebührt den Mitarbeiterinnen und Mitarbeitern des Hanser Verlags, die mir stets mit Rat und Tat zur Seite standen.

Herzlichen Dank sage ich auch meinen Kolleginnen und Kollegen Isabella Scholz, Marco Schillinger und Alexander Zeitz, die während des Schreibens nicht nur ein Auge auf meine Manuskripte geworfen haben, sondern mit viel Akribie jedes Kapitel geprüft haben. Ihre Hinweise und Tipps konnten mehr als nur die eine oder andere Kante glätten.

Außerdem danke ich meiner Familie für ihre Liebe und Unterstützung, ihr Verständnis und ihre Geduld: Meiner Ehefrau Moni danke ich für das unermessliche Glück, das sie mir jeden Tag aufs Neue schenkt; meinen Eltern, Rudolf und Gertraud, für alles, was sie mir mit auf den Weg gegeben haben; und meinem Bruder Andreas für die vielen Dinge, die er mir ermöglicht hat.

# 1 Hands-on Eclipse

Eclipse basiert auf Java-Technologie. Damit Sie die Entwicklungsumgebung einsetzen können, muss auf Ihrem Rechner eine möglichst aktuelle Version der Java-Laufzeitumgebung vorhanden sein. Da Eclipse im Gegensatz zu anderen IDEs seinen eigenen Compiler mitbringt, ist das Java Development Kit (JDK) nicht unbedingt erforderlich. Allerdings beinhaltet es eine Reihe wichtiger zusätzlicher Werkzeuge. Aus diesem Grund rate ich Ihnen, es in jedem Fall zu installieren.

Der erste Abschnitt zeigt Ihnen zunächst, wie Sie die Java-Laufzeitumgebung sowie das Java Development Kit installieren und optimal einrichten. Auch wenn Sie Java schon auf Ihrem Rechner haben, empfehle ich Ihnen einen kurzen Blick in den zu Ihrem Rechner passenden Unterabschnitt. Vielleicht entdecken Sie den einen oder anderen Tipp, mit dem Sie Ihre Installation optimieren können. Danach geht es um die Installation von Eclipse. Sie lernen, die Entwicklungsumgebung auf Ihrem Rechner einzurichten. Anschließend zeige ich Ihnen, wie Sie Ihre Eclipse-Installation mithilfe automatischer Aktualisierungen auf dem neuesten Stand halten. Ein erstes, eigenes Projekt bildet den Schwerpunkt des zweiten Abschnitts. Sie werden das Projekt anlegen, eine Java-Klasse erzeugen sowie das Programm bearbeiten und ausführen. Der dritte Abschnitt enthält einen Rundgang durch die Menüs und Dialoge der IDE.

## 1.1 Java und Eclipse installieren

Dieser Abschnitt beschäftigt sich mit der Installation von Java und Eclipse. Da dieser Vorgang vom Betriebssystem Ihres Entwicklungsrechners abhängig ist, gibt es für die drei populärsten Betriebssysteme, Windows, Linux und macOS, entsprechende Unterabschnitte.

**HINWEIS:** Dieses Buch bezieht sich auf die 64-Bit-Versionen von Java und Eclipse

### 1.1.1 Installation von Java

Eclipse benötigt für die Ausführung eine möglichst aktuelle Version der Java-Laufzeitumgebung oder des Java Development Kits. Letzteres hält eine Reihe interessanter Erweiterungen (beispielsweise Java Mission Control) und zahlreiche Werkzeuge (etwa zum Signieren von Anwendungen) bereit. Diese dienstbaren Geister fehlen in der als eigenständiger Download erhältlichen Laufzeitumgebung. Aus diesem Grund verwende ich hier das JDK als Grundlage für die Eclipse-Installation.

#### Java unter Windows installieren

Laden Sie die aktuelle Version (zum Zeitpunkt der Drucklegung Java SE 10.0.2; jdk-10.0.2_windows-x64_bin.exe) von der Website des Java-Herstellers herunter *(https://www.oracle.com/technetwork/java/javase/downloads/index.html)*. Nach dem Start des Setup-Programms müssen Sie zunächst den Installationsumfang bestimmen und das Zielverzeichnis festlegen. Die Java-Quelltexte sollten Sie auf jeden Fall installieren. Sie können bei der Fehlersuche eine große Hilfe sein.

Nun sollten Sie durch Anpassen von Umgebungsvariablen sicherstellen, dass auch bei einer parallelen Installation mehrerer Java-Releases immer auf die gewünschte Version zugegriffen wird. Sie erreichen dies, indem Sie die Umgebungsvariable JDK_HOME definieren, die auf das JDK-Installationsverzeichnis verweist (beispielsweise C:\Program Files\Java\jdk-10.0.2). Anschließend löschen Sie alle Verweise auf Java-Installationen aus der Umgebungsvariablen PATH und setzen stattdessen eine Referenz auf %JDK_HOME%\bin als allerersten Pfad. Möchten Sie später mit einer anderen Java-Version arbeiten, reicht es, diese Umgebungsvariable zu modifizieren.

Sie können Ihre Einstellungen testen, indem Sie in der Eingabeaufforderung oder PowerShell die Befehle java -version sowie javac -version eingeben. Die beiden Programme sollten trotz der fehlenden Pfadangaben gefunden werden. Falls nicht, überprüfen Sie bitte Ihre Umgebungsvariablen.

#### Java unter Linux installieren

Laden Sie die aktuelle Version (zum Zeitpunkt der Drucklegung Java SE 10.0.2; jdk-10.0.2_linux-x64_bin.tar.gz) von der Website des Java-Herstellers herunter *(https://www.oracle.com/technetwork/java/javase/downloads/index.html)*. Verschieben Sie die Installationsdatei mit Administratorrechten in das Verzeichnis /opt und starten Sie den Entpackvorgang. Falls /opt nicht existiert, legen Sie das Verzeichnis vorher durch Ausführen des Befehls sudo mkdir /opt an:

```
cd <Verzeichnis, in dem jdk-10.0.2_linux-x64_bin.tar.gz liegt>
sudo mv jdk-10.0.2_linux-x64_bin.tar.gz /opt
cd /opt
sudo tar xzf jdk-10.0.2_linux-x64_bin.tar.gz
```

Die Anweisung sudo rm jdk-10.0.2_linux-x64_bin.tar.gz löscht die nun nicht mehr benötigte Setup-Datei. Das JDK befindet sich nun in einem Verzeichnis, dessen Name den aktuellen Patchlevel enthält. Um von diesem unabhängig zu sein, rate ich Ihnen, einen symbolischen Link anzulegen, der auf das eigentliche Installationsverzeichnis zeigt: sudo ln -s jdk-10.0.2 java.

Damit Sie Java ohne Pfadangaben starten können, empfehle ich Ihnen ferner, der Umgebungsvariablen JDK_HOME den absoluten Pfad des Installationsverzeichnisses zuzuweisen. Wenn Sie den zuvor beschriebenen symbolischen Link erzeugt haben, können Sie ihn in der Pfadangabe gleich verwenden. Der Wert für JDK_HOME lautet dann /opt /java. Löschen Sie alle Verweise auf Java-Installationen aus der Umgebungsvariablen PATH und setzen Sie stattdessen eine Referenz auf $JDK_HOME/bin an die allererste Stelle.

In welcher Datei Sie die Umgebungsvariablen am besten setzen, hängt von Ihrer Linux-Distribution ab. Es hat sich bewährt, die Einstellungen in .bashrc im Heimatverzeichnis des aktuellen Benutzers vorzunehmen (auch wenn die Modifikationen dann nicht systemweit gültig sind). Wenn Sie anstelle der bash eine andere Shell verwenden, tragen Sie die Anweisungen in deren Konfigurationsdatei ein.

Möchten Sie später mit einer anderen Java-Version arbeiten, müssen Sie nur die Umgebungsvariable JDK_HOME modifizieren oder den weiter oben angesprochenen symbolischen Link aktualisieren. Sie können Ihre Einstellungen testen, indem Sie in der bash (oder natürlich jeder anderen Shell) die Befehle java version und javac version eingeben. Die beiden Programme sollten trotz der fehlenden Pfadangaben gefunden werden. Ist dies nicht der Fall, überprüfen Sie ihre Umgebungsvariablen.

### Java unter macOS installieren

Laden Sie die aktuelle Version (zum Zeitpunkt der Drucklegung Java SE 10.0.2; jdk-10.0.2_osx-x64_bin.dmg) direkt von der Website des Java-Herstellers herunter *(https://www.oracle.com/technetwork/java/javase/downloads/index.html)*. Öffnen Sie das Diskimage. Ein weiterer Doppelklick auf die nun sichtbare Datei (JDK 10.0.2.pkg) startet den Installationsassistenten. Folgen Sie dessen Anweisungen. Die heruntergeladene .dmg-Datei ist nun nicht mehr erforderlich. Sie können sie deshalb löschen. Beachten Sie aber, vorher das gemountete Diskimage auszuwerfen.

**HINWEIS:** Alle Java Development Kits werden unterhalb des Verzeichnisses /Library/Java/JavaVirtualMachines installiert. Öffnen Sie ein Terminal und wechseln Sie durch Eingabe von cd /Library/Java/JavaVirtualMachines dorthin. Lassen Sie sich mit ls den Inhalt anzeigen. Sie sollten einen Ordner mit Namen jdk-10.0.2.jdk vorfinden.

## 1.1.2 Installation von Eclipse

In diesem Abschnitt geht es um die Installation von Eclipse. Um optimal auf die jeweilige Zielplattform eingehen zu können, gibt es auch hier Unterabschnitte für die Betriebssysteme Windows, Linux und macOS. Diese behandeln jeweils das Einrichten auf dem Computer sowie den allerersten Start. Wie Sie Eclipse an Ihre Bedürfnisse anpassen, erfahren Sie in Abschnitt 1.1.3 , „Der erste Start“, der dann wieder plattformunabhängig gehalten ist.

Für dieses Buch verwende ich Eclipse IDE for Java Developers als Grundlage. Diese Variante beinhaltet neben der Eclipse Platform die Java Development Tools, einen Git Client, Maven- und Gradle-Integration, sowie einen XML-Editor. Neben diesem für den Einstieg in die Pro-

grammierung mit Java idealen Paket bietet die Eclipse Foundation zahlreiche weitere Versionen oder Packages an, beispielsweise Eclipse IDE for PHP Developers, Eclipse IDE for C/C++ Developers und Eclipse for JavaScript and Web Developers. Eine Übersicht über die verfügbaren Pakete finden Sie unter *https://www.eclipse.org/downloads/packages/release/Photon/R/*.

### Eclipse unter Windows installieren

Um Eclipse IDE for Java Developers zu installieren, genügt es, den Inhalt des Archivs eclipse-java-photon-R-win32-x86_64.zip in dem Verzeichnis zu entpacken, das Eclipse später enthalten soll. Möchten Sie die IDE beispielsweise unter C:\Programme ablegen, müssen Sie diesen Pfad als Ziel des Entpackvorgangs angeben. Ihr Archivprogramm sollte automatisch das Verzeichnis Eclipse erzeugen, sodass die Komponenten des SDKs letztlich unter C:\Programme\Eclipse zu finden sind. Damit ist die Installation eigentlich schon abgeschlossen. Um nicht für jeden Startvorgang ein Verzeichnisfenster öffnen zu müssen, ist es allerdings ratsam, im Startmenü oder auf dem Desktop eine Verknüpfung mit eclipse.exe anzulegen.

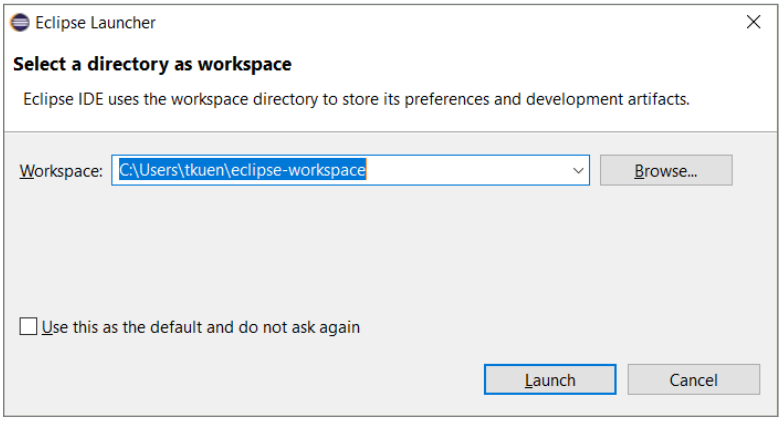


**Bild 1.1** Der Dialog „Eclipse Launcher“ unter Windows

Starten Sie nun Eclipse. Sie sehen den in Bild 1.1 gezeigten Dialog *Eclipse Launcher*, in dem Sie einen sogenannten Arbeitsbereichsordner (engl. workspace folder) auswählen müssen. Vereinfacht ausgedrückt, ist der Arbeitsbereich eine Art Container für Projekte. Auf Dateisystemebene ist jedes Projekt ein Verzeichnis unterhalb des Arbeitsbereichsverzeichnisses, das Sie hier angeben müssen. Auch wenn Sie während der Arbeit mit Eclipse eher selten darauf zugreifen werden, sollten Sie den Arbeitsbereichsordner an einer bekannten Stelle ablegen oder sich die von Eclipse vorgeschlagene Position zumindest gut merken. In Abschnitt 1.1.3 zeige ich Ihnen, wie Sie Eclipse Ihren Wünschen entsprechend einrichten. Übernehmen Sie die voreingestellten Werte und schließen Sie den Dialog mit **OK**.

### Eclipse unter Linux installieren

Sie finden Eclipse IDE for Java Developers für Linux unter *https://www.eclipse.org/downloads/packages/release/Photon/R/*. Um sie zu installieren, müssen Sie nur den Inhalt der Datei eclipse-java-photon-R-linux-gtk-x86_64.tar.gz in dem Verzeichnis entpacken, das

Eclipse später enthalten soll. Es hat sich bewährt, die Entwicklungsumgebung wie das Java Development Kit unter /opt abzulegen.

```
cd <Verzeichnis mit eclipse-java-photon-R-linux-gtk-x86_64.tar.gz >
sudo mv eclipse-java-photon-R-linux-gtk-x86_64.tar.gz  /opt
cd /opt
sudo tar xzf eclipse-java-photon-R-linux-gtk-x86_64.tar.gz
```

Um Speicherplatz zu sparen, können Sie anschließend die Setup-Datei mit sudo rm eclipse-java-photon-R-linux-gtk-x86_64.tar.gz löschen. Außerdem bietet es sich an, eine Verknüpfung auf dem Desktop oder im Programmmenü der von Ihnen verwendeten Desktop-Umgebung anzulegen. Damit ist die Installation von Eclipse abgeschlossen.

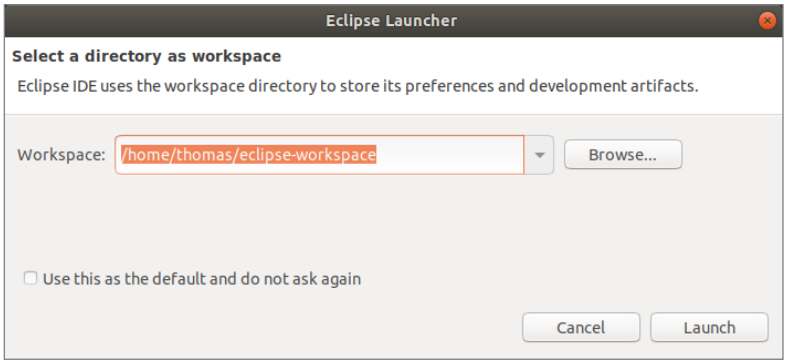


**Bild 1.2** Der Dialog „Eclipse Launcher“ unter Linux

Nach dem Start der IDE sehen Sie den in Bild 1.2 gezeigten Dialog *Eclipse Launcher*, in dem Sie einen sogenannten Arbeitsbereichsordner (engl. workspace folder) auswählen müssen. Vereinfacht ausgedrückt, ist der Arbeitsbereich eine Art Container für Projekte. Auf Dateisystemebene ist jedes Projekt ein Verzeichnis unterhalb des Arbeitsbereichsverzeichnisses, das Sie hier angeben müssen. Auch wenn Sie während der Arbeit mit Eclipse eher selten darauf zugreifen werden, sollten Sie den Arbeitsbereichsordner an einer bekannten Stelle ablegen oder sich die von Eclipse vorgeschlagene Position zumindest gut merken. In Abschnitt 1.1.3 zeige ich Ihnen, wie Sie Eclipse Ihren Wünschen entsprechend einrichten können. Übernehmen Sie die voreingestellten Werte und schließen Sie den Dialog mit **OK**.

### Eclipse unter macOS installieren

Sie finden Eclipse IDE for Java Developers für macOS unter *https://www.eclipse.org/downloads/packages/release/Photon/R/*. Um sie zu installieren, müssen Sie zunächst den Inhalt der Datei eclipse-java-photon-R-macosx-cocoa-x86_64.dmg durch Doppelklick anzeigen und anschließend das Eclipse-Icon auf das Symbol für das Verzeichnis Programme schieben. Beide Icons sind in dem Finder-Fenster zu sehen, das sich nach dem Doppelklick auf die .dmg-Datei geöffnet hat. Damit ist die Installation eigentlich schon abgeschlossen. Um nicht für jeden Startvorgang ein Verzeichnisfenster öffnen zu müssen, ist es allerdings ratsam, auf dem Desktop ein Alias anzulegen oder das Icon der Anwendung im Dock zu behalten. Die heruntergeladene Datei ist nun nicht mehr erforderlich und kann gelöscht werden. Bitte werfen Sie vorher das durch den Doppelklick auf die .dmg-Datei gemountete Verzeichnis aus. Starten Sie nun Eclipse.

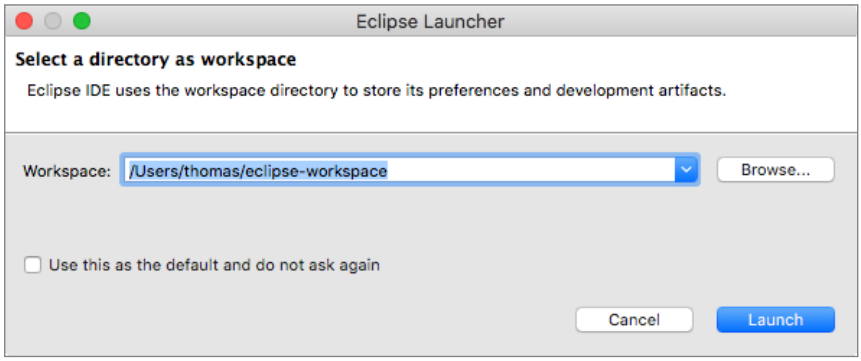


**Bild 1.3** Der Dialog „Eclipse Launcher“ unter macOS

Sie sehen den in Bild 1.3 dargestellten Dialog *Eclipse Launcher*, in dem Sie einen sogenannten Arbeitsbereichsordner (engl. workspace folder) auswählen müssen. Vereinfacht ausgedrückt, ist der Arbeitsbereich eine Art Container für Projekte. Auf Dateisystemebene ist jedes Projekt ein Verzeichnis unterhalb des Arbeitsbereichsverzeichnisses, das Sie hier angeben müssen. Auch wenn Sie während der Arbeit mit Eclipse eher selten darauf zugreifen werden, sollten Sie den Arbeitsbereichsordner an einer bekannten Stelle ablegen oder sich die von Eclipse vorgeschlagene Position zumindest gut merken. Übernehmen Sie die voreingestellten Werte und schließen Sie den Dialog mit **OK**. Im folgenden Abschnitt erfahren Sie, wie Sie Eclipse Ihren Wünschen entsprechend einrichten können.

### 1.1.3 Der erste Start

Sowohl Java als auch Eclipse sind jetzt einsatzbereit. Der Arbeit mit der IDE steht also nichts mehr entgegen. Wenn Sie es nicht bereits getan haben, sollten Sie Eclipse nun starten. Es erscheint der Dialog *Eclipse Launcher*, in dem Sie den sogenannten Arbeitsbereich oder Workspace festlegen müssen. Eclipse arbeitet auf der Grundlage von Projekten. Ein Projekt fasst Dateien und Einstellungen themenbezogen zusammen. Alle Projekte werden in einem Arbeitsbereich abgelegt. Wie Sie später noch sehen werden, sind Projekte Unterverzeichnisse des Arbeitsbereichsordners. Wo Sie diesen anlegen, ist letztlich egal. Wichtig ist nur, dass das entsprechende Verzeichnis ständig verfügbar ist. Im Hinblick auf die Arbeitsgeschwindigkeit sollten Sie den Arbeitsbereich also nach Möglichkeit nicht auf einem Netzlaufwerk anlegen.

Es ist übrigens auch im laufenden Betrieb möglich, zwischen Arbeitsbereichen umzuschalten. Von dieser Möglichkeit werden Sie wahrscheinlich Gebrauch machen, wenn sich viele Projekte angesammelt haben und Sie durch sinnvolle Gruppierungen wieder Ordnung in den Dschungel bringen müssen. Zu Beginn ist es allerdings ausreichend, für alle Projekte einen Arbeitsbereich vorzusehen. Aus diesem Grund sollten Sie vor **Use this as the default and do not ask again** ein Häkchen setzen. Dadurch wird der Dialog *Eclipse Launcher* beim nächsten Start nicht mehr angezeigt.

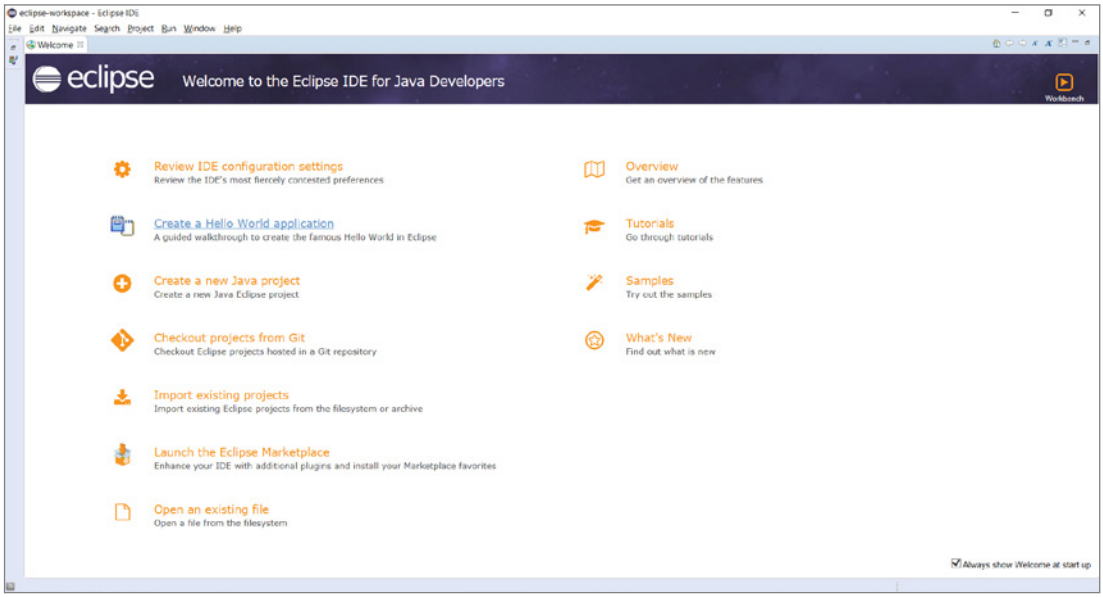


**Bild 1.4** Eclipse nach dem ersten Start

Nachdem alle benötigten Komponenten geladen wurden, präsentiert Eclipse den in Bild 1.4 gezeigten Willkommensbildschirm. Er enthält eine Reihe von Links, mit denen Sie beispielsweise ein neues Java-Projekt anlegen, eine vorhandene Datei öffnen oder Projekte aus einen Git-Repository auschecken können. Das Symbol 

**(Workbench)** aktiviert die sogenannte Workbench. Dort erstellen und bearbeiten Sie Ihre Projekte und Quelltexte. Über den Menüpunkt **Help/Welcome** kommen Sie übrigens jederzeit zum Willkommensbildschirm zurück.

**Bild 1.5** Symbolleiste des Willkommensbildschirms

Um dem Willkommensbildschirm weitere Symbole hinzuzufügen, öffnen Sie dessen Einstellungsdialog, den Sie über die in Bild 1.5 gezeigte Symbolleiste erreichen. Mit den ersten drei Symbolen navigieren Sie durch die Lernprogramme und Informationsseiten. Die folgenden beiden verkleinern bzw. vergrößern die Schrift. **Customize page** (der Name der Funktion erscheint als Tooltip, wenn Sie die Maus auf das Symbol bewegen) öffnet den Dialog *Customize*, den Sie in Bild 1.6 sehen. Fügen Sie dem Willkommensbildschirm ein neues Symbol hinzu, indem Sie ein Häkchen vor **Migrate** setzen. Jede aktivierte Root Page ist über eine eigene Registerkarte konfigurierbar. Die neuen Symbole werden sichtbar, sobald Sie den Dialog mit **OK** schließen.

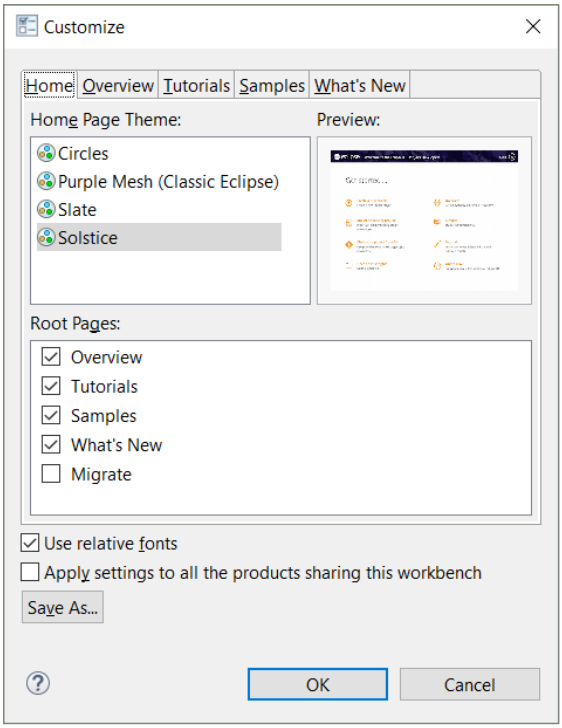


**Bild 1.6** Dialog zum Einrichten des Willkommensbildschirms

### Benutzervorgaben ändern

Unter Windows und Linux erreichen Sie die meisten Einstellungen, um Eclipse und die Plug-ins an Ihre Bedürfnisse anzupassen, über den Menüpunkt **Window/Preferences**. Dieser öffnet den in Bild 1.7 gezeigten gleichnamigen Dialog. macOS-Anwender finden ihn unter Eclipse/Einstellungen.

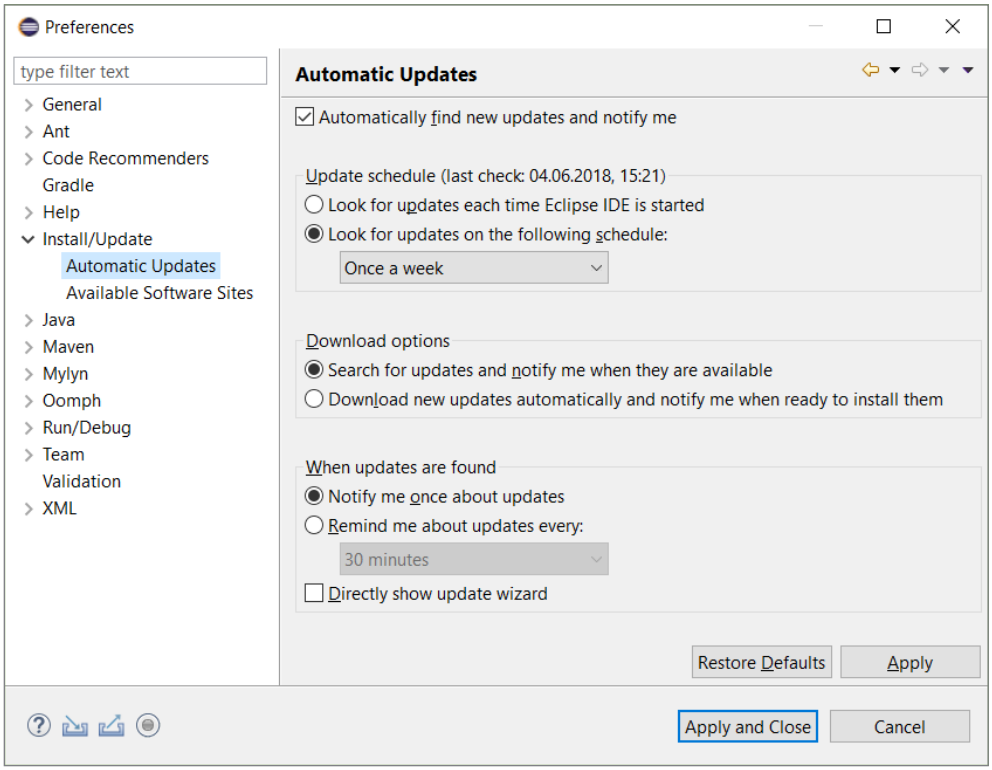


**Bild 1.7** Der Dialog „Preferences“

Eine sehr interessante Option ist, Eclipse selbstständig nach Programmaktualisierungen suchen zu lassen. Sie finden die zugehörigen Einstellungen unterhalb des Knotens **Install/Update** unter **Automatic Updates**.

### Eclipse aktualisieren

Öffnen Sie den Dialog *Preferences* und navigieren Sie zu der Seite Automatic Updates. Aktivieren Sie nun **Automatically find new updates and notify me** und wählen Sie dann einen der beiden Update-Pläne (Update schedule) aus. In der Einstellung Download options schließlich legen Sie fest, ob Sie über Updates nur informiert oder ob diese automatisch heruntergeladen werden sollen. Der unterste Bereich (When updates are found) steuert, wie oft Sie Eclipse über vorhandene Updates informiert. Falls Sie sich gegen automatische Aktualisierungen entscheiden, rate ich Ihnen, regelmäßig manuell nach neuen Programmversionen zu suchen.

Wenn Sie sich neugierig durch die zahlreichen Knoten des Dialogs *Preferences* geklickt haben, fühlen Sie sich von der Vielzahl der möglichen Einstellungen sehr wahrscheinlich überfordert. Ich darf Sie an dieser Stelle beruhigen. Die Grundeinstellungen von Eclipse wurden so gewählt, dass Sie hier nur sehr selten Änderungen vornehmen müssen. Sehen Sie die Einstellmöglichkeiten vielmehr als ein Angebot an den Profi, die IDE bis ins letzte Detail an seine Bedürfnisse anzupassen. Ich werde im Verlauf des Buchs noch häufiger auf diesen Dialog hinweisen.

### Aufrufparameter

Sie können das Verhalten von Eclipse steuern, indem Sie der IDE beim Start Parameter übergeben. Hier gilt es, zwei Gruppen von Optionen zu unterscheiden. Die eine beeinflusst Eclipse selbst, die andere wirkt sich auf die virtuelle Maschine aus, in der Eclipse ausgeführt wird. Hierzu ein Beispiel:

```
eclipse -vmargs -Xmx1024m
```

Das Schlüsselwort -vmargs leitet die Übergabe von Optionen zur Steuerung der Java-Laufzeitumgebung ein. Das Argument -Xmx1024m setzt die Maximalgröße des sogenannten memory allocation pools (vereinfacht ausgedrückt, wie viel Speicher maximal zur Verfügung stehen soll) auf 1024 MB. Eine Aufstellung der Aufrufoptionen der Java-Laufzeitumgebung finden Sie auf der Seite *https://docs.oracle.com/javase/10/tools/java.htm#GUID-3B1CE181-CD30-4178-9602-230B800D4FAE*. Optionen, die Eclipse selbst beeinflussen, werden im Workbench User Guide unter Tasks/Running Eclipse aufgelistet. Sie erreichen ihn über **Help/Help Contents**. Weitere Informationen über das in Eclipse eingebaute Hilfesystem finden Sie übrigens in Abschnitt 1.3.1, „Die Hilfefunktionen von Eclipse“.

Sie können mit -data das Verzeichnis des zu verwendenden Arbeitsbereichs angeben. Ebenfalls sehr praktisch ist -showlocation. Diese Option zeigt den Pfad des Arbeitsbereichsordners in der Fenstertitelzeile an. Mit vm <Pfad auf das bin-Verzeichnis einer Java-Laufzeitumgebung> bestimmen Sie die virtuelle Maschine, die für das Ausführen von Eclipse verwendet wird. nosplash schließlich blendet die während des Starts angezeigte Begrüßungsgrafik mit Fortschrittsbalken aus. Was sich recht praktisch anhört, will aber dennoch gut überlegt sein. Denn der Start der IDE dauert eine gewisse Weile. Dann ist es gut zu wissen, dass sich „etwas bewegt“.

Sie haben nun einen ersten Einblick in die Arbeit mit Eclipse gewonnen, indem Sie die IDE in ein paar Bereichen Ihren Bedürfnissen angepasst und die Möglichkeit der automatisierten Produktaktualisierung kennengelernt haben. Sicher brennen Sie schon darauf, Ihr erstes Java-Programm mit Eclipse zu schreiben. Dies ist Gegenstand des folgenden Abschnitts.

## 1.2 Das erste eigene Projekt

In diesem Abschnitt möchte ich Ihnen eine kleine Java-Anwendung vorstellen und auf diese Weise erste Erfahrungen im Umgang mit Projekten vermitteln. Die Aufgabe, die das Programm lösen soll, ist unspektakulär: Es gilt, eine Liste der System-Properties auszugeben. Zu seiner Funktionsweise werde ich wenig sagen, schließlich geht es Ihnen ja nicht um eine Einweisung in Java, sondern in die Funktionsweise von Eclipse.

### 1.2.1 Ein neues Projekt anlegen

Nachdem Sie Eclipse gestartet haben, klicken Sie im Willkommensbildschirm auf **Create a new Java project**. Falls Sie ihn schon geschlossen haben, können Sie ihn über die Menü-

leiste mit **Help/Welcome** erneut anzeigen. Ein anderer Weg, den Assistenten zum Anlegen von Java-Projekten aufzurufen, ist ebenfalls über die Menüleiste: **File/New/Java Project**. In beiden Fällen sehen Sie den in Bild 1.8 dargestellten Dialog *New Java Project*.

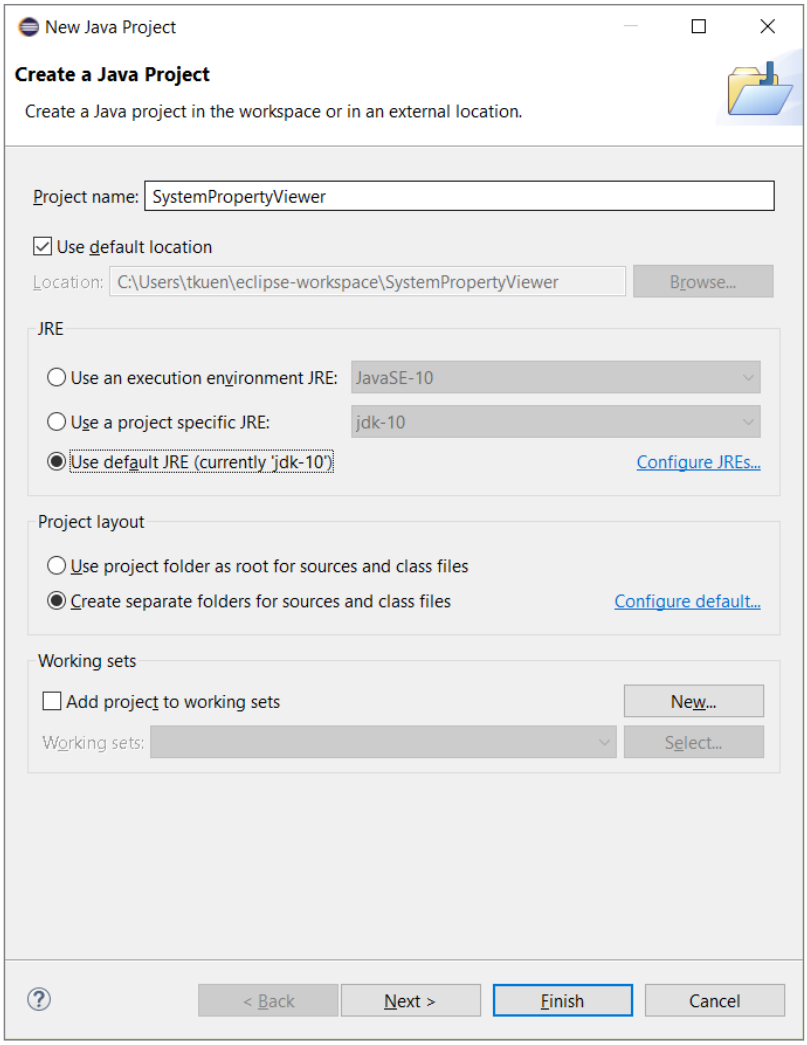


**Bild 1.8** Dialog zum Anlegen neuer Projekte

Geben Sie Ihrem Projekt einen aussagekräftigen Namen, beispielsweise „SystemProperty-Viewer“. Die übrigen Einstellungen sollten Sie so vornehmen, dass **Use default location**, **Use default JRE** und **Create separate folders for sources and class files** ausgewählt sind. Durch Klicken auf **Finish** schließen Sie den Dialog *New Java Project*. Wenn Sie Java 9 oder neuer verwenden, möchte Eclipse nun einen Modulnamen wissen (Bild 1.9). Module beschränken die Sichtbarkeit bzw. den Zugriff auf Packages und ermöglichen dadurch, Implementierungsdetails außerhalb des Moduls zu verbergen. Per Konvention beginnen Modulnamen mit einem Kleinbuchstaben. Da Eclipse den Projektnamen als Modulnamen

vorschlägt, erscheint in meinem Beispiel ein entsprechender Warnhinweis. Ersetzen Sie einfach das „S“ durch ein „s“.

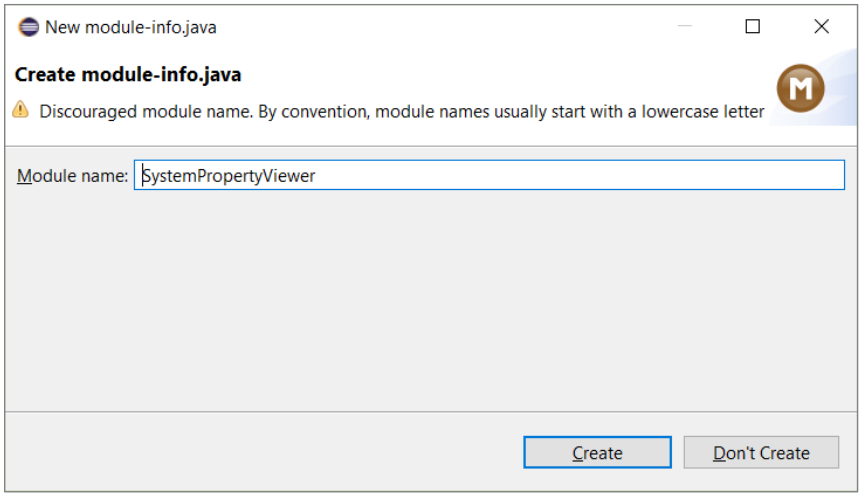


**Bild 1.9** Einen Modulnamen eingeben

Ihr Eclipse-Fenster sollte nun in etwa Bild 1.10 entsprechen.

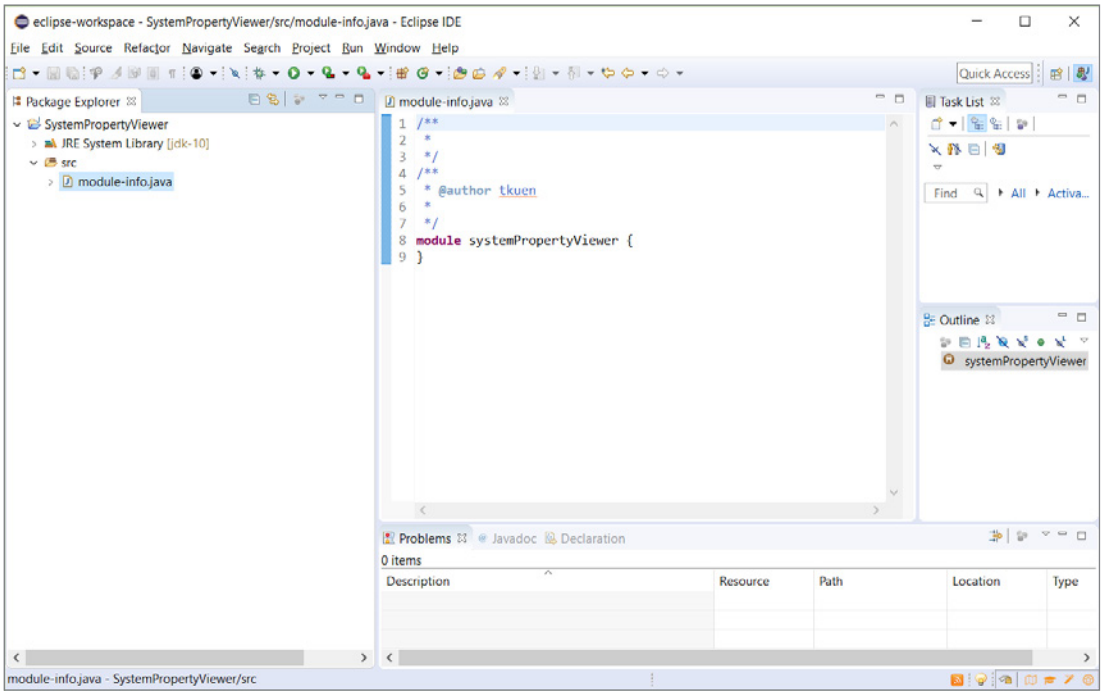


**Bild 1.10** Eclipse nach Anlegen eines Projekts

Falls nicht, wechseln Sie zunächst in die sogenannte *Java*-Perspektive. Klicken Sie hierzu in der Toolbar das Icon (Open Perspective) an oder wählen Sie den Menüpunkt **Window/Perspective/Open Perspective/Other**. In beiden Fällen sehen Sie daraufhin den Dialog *Open Perspective* (Bild 1.11). Klicken Sie zuerst auf **Java (default)**, dann auf **Open**.

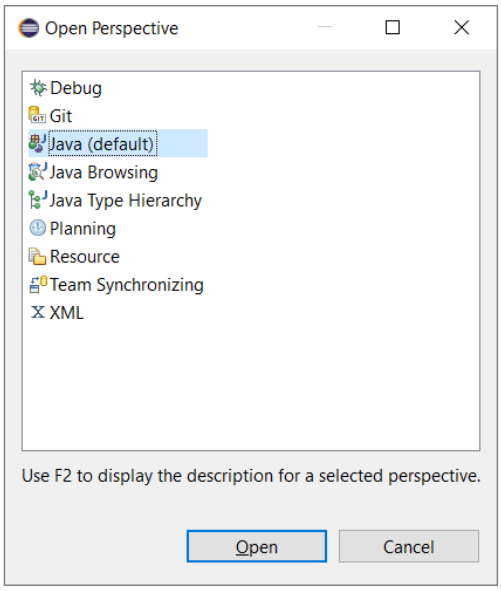


**Bild 1.11** Der Dialog „Open Perspective“

Anschließend lassen Sie den Package Explorer anzeigen, indem Sie **Window/Show View/Package Explorer** anklicken. Bitte lassen Sie sich nicht von den vielen neuen Begriffen verwirren. In Kapitel 2, „Arbeiten mit Eclipse“, werden wir uns ausführlich mit der Bedienphilosophie der IDE und den ihr zugrunde liegenden Konzepten beschäftigen. Fürs Erste wollen wir uns damit begnügen, dass der Package Explorer eine nach Paketen und Klassen gruppierte Sicht (engl. View) auf die Elemente eines Projekts ermöglicht.

### Klassen hinzufügen

Fahren Sie im Package Explorer mit der Maus auf den Eintrag **SystemPropertyViewer** und öffnen Sie mit einem Rechtsklick dessen Kontextmenü. Nach Anklicken des Eintrags **New/Class** öffnet sich der Dialog *New Java Class*, in dem Sie die Kriterien festlegen, die die neu anzulegende Klasse erfüllen soll. Bitte tragen Sie als Name `SystemPropertyViewer` und als Package `com.thomaskuenneth.systempropertyviewer` ein. Sie sollten dieses Feld nicht leer lassen, weil Klassen ohne Paketzugehörigkeit nicht gern gesehen sind. Da Ihr Java-Programm nur aus einer Klasse bestehen wird, ist es ferner nützlich, gleich den Einsprungspunkt, also die Methode `main()`, generieren zu lassen. Sie erreichen dies durch das Setzen eines entsprechenden Häkchens im Bereich Which method stubs would you like to create. In Bild 1.12 sehen Sie, wie Sie den Dialog zum Erzeugen der neuen Java-Klasse `SystemPropertyViewer` ausfüllen sollten.