Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Ausarbeitung

Ein kurzer Einblick in die Integrierte Softwareentwicklungsumgebung: Eclipse

vorgelegt von

Angelina Heinrichs

Fakultät für Mathematik, Informatik und Naturwissenschaften
Fachbereich Informatik
Arbeitsbereich Wissenschaftliches Rechnen

Proseminar Softwareentwicklung in der Wissenschaft

Studiengang:	Informatik
Matrikelnummer:	7315000

Betreuer: Dr. Hermann Lenhart

Hamburg, 2020-06-29

Inhaltsverzeichnis

1 Einleitung

In diesem Kapitel beschäftigen wir uns zunächst einmal mit dem Grundbegriff einer Integrierten Softwareentwicklungsumgebung und schauen dann was alles mit einer solchen Umgebung möglich ist.

1.1 Definition einer IDE

IDE steht für Integrated Development Environment - zu deutsch Integrierte Softwareentwicklungsumbegung. Das große Ziel einer solchen IDE ist die Erleichterung für Softwareentwickler. Um dies zu ermöglichen, bringt eine IDE die wichtigsten Werkzeuge, wie Editor, Compiler, Debugger und vieles mehr, für Entwickler auf einer Oberfläche zusammen. Dabei spielt der Austausch zwischen den einzelnen Komponenten eine wichtige Rolle, denn dadurch werden Programme automatisch aufeinander abgestimmt, was ansonsten sehr viel Zeit in Anspruch nehmen würde. [Aug17b] [fil14]
Am Anfang waren die IDEs hauptsächlich textbasiert. Heutzutage finden die meisten IDEs auf visueller Basis mit grafischen Benutzeroberflächen statt. Auch die Vielfalt der Nutzung bezüglich Programmiersprachen, Betriebssyteme und Plattformen ist mittlerweile Standard. [Aug17b]

1.2 Möglichkeiten einer IDE

Viele Anwendungsbereiche wie Web, Game, Apps und Datenbanken werden durch IDEs unterstützt. Gerade bei größeren Projekten ist dann das gemeinsame und parallele Arbeiten in einer IDE sogar notwendig. Dies kann in entsprechenden Umgebungen ohne viel Mehraufwand geschehen. Hilfreich ist auch die Akzeptanz verschiedener Programmiersprachen, teilweise sogar mehrere in einem Projekt. Auch für Anfänger bieten einige IDEs viel Unterstützung an, indem zum Beispiel die grafische Benutzeroberfläche soweit ausgebaut ist, dass sich dadurch ganze Programme entwickeln lassen ohne Code zu schreiben. [Aug17b]

2 Eclipse

In diesem Kapitel wenden wir uns der Software Eclipse zu. Dabei erfahren wir zunächst ein paar grundlegende Informationen über die Software, bevor wir uns mit der Installation und Einrichtung des Programms auseinander setzen. Weiter geht es dann mit einem Einblick in den Aufbau von Eclipse. Anschließend programmieren wir ein kleines "Hello World"-Programm.

2.1 Allgemeine Informationen

Eclipse ist eine opensource IDE, die ursprünglich nur für die objektorientierte Sprache Java gedacht war, mittlerweile aber ein großes Paket an anderen Programmiersprachen unterstützt. Eclipse ist Nachfolger der Entwicklungsumbgebung IBM Visual Age for Java 4.0 und wurde im November 2001 von IBM freigegeben. Die eigenständige Eclipse Foundation wurde dann im Februar 2004 gegründet, die seitdem für die Weiterentwicklung von Eclipse zuständig ist. [Aug17a]
Aktuell gibt es 23 Versionen von Eclipse (Stand Juni 2020), wovon diese bis Juni 2018 in alphabezischer Reihenfolge mit Namen wie Callisto, Gallileo , Mars und Indigo banannt wurden. Seit September 2018 hat man dann den Standard "Jahr- Monat" für die Versionsnamen eingeführt. Dabei liefern neue Versionen meistens weitere Packages und Funktionen für die Entwicklungsumgebung. [wik20]

2.2 Installation

Da Eclipse eine Java orientierte IDE ist, sollte man vor der Installation von Eclipse sicher stellen, dass man auch eine passende Java Runtime Environment (JRE) installiert hat.[1] Grundsätzlich gilt aber, wer auch mit Java in Eclipse arbeiten möchte installiert sich besser gleich das JDK. Auch wichtig ist, dass entsprechende bit-Versionen heruntergeladen werden. Für 32-bit Eclipse wird eine 32-bit Java Vitual Machine (JVM)[2] und entsprechend dazu bei 64-bit Eclipse auch eine 64-bit JVM benötigt. [Dah20]
Wenn es um die eigentliche Installation von Eclipse geht, stehen einem einige Packages zur Auswahl. Diese liefern bestimmte Funktionen für entprechende Anwendungsbereiche direkt mit. Wenn man zum Beispiel weiß, dass man in C++ statt in Java programmieren möchte sollte man sich direkt das Package "Eclipse IDE for C/C++ Developers" herunterladen. Nachdem man sich für ein Package entschieden hat, muss man nur noch den

[1] für macOS Benutzer muss sogar das Java Development Kit (JDK) instlliert werden
[2] Die JVM ist sowohl im JRE als auch im JDK enthalten

.zip-Ordner entpacken und den Installer mithilfe der .exe Datei starten. [tut] [fil14] Beim ersten Start von Eclipse muss man seinen Workspace einrichten, das ist der Ort an dem in Zukunft die persönlichen Projekte abgespeichert werden. Dies kann man jederzeit in den Einstellungen ändern. Nun befindet man sich im sogenannten Workbench Menü. Von hier aus kann man dann sein erstes Projekt erstellen. Bei jedem weiteren Start öffnet Eclipse dann direkt das Projekt, an dem man zuletzt gearbeitet hat.

2.3 Aufbau und Einrichtung der Software

Sobald man in Eclipse ein Projekt geöffnet hat, befindet man sich in einem Fester mit einigen Menüs und Ansichten, wie in Abbildung 2.1:"Eclipse Aufbau". Im Folgenden werden die wichtigsten und nützlichsten davon besprochen.

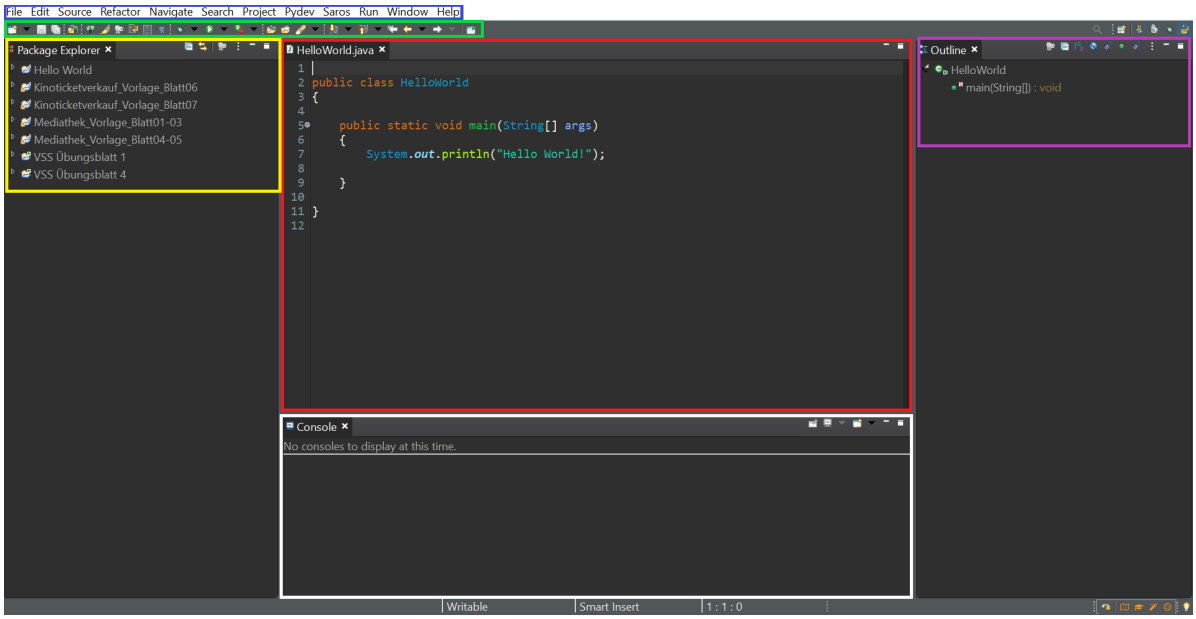

Figure 2.1: Eclipse Aufbau

1) Der Editor (rot)
   Mittig befindet sich der Editor. Hier findet die Hauptbeschäftigung eines Softwareentwicklers statt: das Programmieren. Wie in den meisten Editoren, werden hier Schlüsselwörter farblich markiert und die Zeilennummerierung angezeigt. Wenn weiterer Code geöffnet wird (auch aus verschiedenen Projekten), erscheint ein neuer Tab in der oberen Leiste des Editors.

2) Menu Bar (blau)
   Die Menu Bar ist ganz oben im Fenster angesiedelt. Unter *File* sind Funktionen, um das Projekt zu speichern und neue Projekte zu erstellen oder importieren. Unter

Anderem kann man hier auch in einen anderen Workspace wechseln. Bei *File* sind typische verändernde Funktionen wie Kopieren, Ausschneiden, Einfügen und Rückgängig machen zu finden. Des weiteren kann man unter *Run* sein Programm sowohl starten und als auch testen. Zudem kann man im Debug-Modus auf Fehlersuche gehen. Wichtig ist auch der Reiter *Window*. Hier lassen sich neue Ansichten öffnen und unter *Preferences* sämtliche Einstellungen ändern(worauf später in diesem Kapitel noch eingegangen wird). Es befinden sich noch deutlich mehr Reiter in der Menu Bar, auf die wir aber nicht im Detail eingehen wollen.

3) Tool Bar (grün)
   In der Tool Bar gibt es viele nützliche Shortcuts. Mit einem Klick kann hier das Projekt gespeichert und gestartet werden sowie neue Klassen und Packages schnell hinzugefügt werden können.

4) Package Explorer View (gelb)
   Hier werden alle Projekte angezeigt, die sich in dem entsprechenden Workspace befinden. Die Ordner kann man einfach aufklappen. Mit einem Doppelklick lassen sich die verschiedenen Ressourcen eines Projekts als neuer Tab im Editor öffnen. Etwas Verändern, Hinzufügen oder zum Laufen bringen, geht mit einem Rechtsklick auf den Projektordner.

5) Outline View (lila)
   Ein sehr hilfreiches Element ist die Outline View. Sie zeigt dem Entwickler Exemplarvariablen, Methoden und deren Typ an. Bei sehr viel Zeilen lässt sich so eine bestimmte Variable oder Methode besser finden und durch Klick zu der entsprechenden Stelle im Code springen.

6) Console (weiß)
   Auf der Console werden Fehlermeldungen ausgegeben. Durch Ort und Beschreibung der Fehlerquelle, lassen sich zumindest Flüchtigkeitsfehler wie ein vergessenes Semikolon schnell beheben.

7) Der Debugger
   Auch ein wichtiges Werkzeug fürs Programmieren ist der Debugger. Den Debug-Modus kann man unter *Run* → *Debug As...* aktivieren. Zuvor sollte man sich aber an den Stellen, die man untersuchen möchte, sogenannte Haltepunkte in seinem Programm setzen. Am einfachsten macht man dafür einen Rechtsklick in der entsprechenden Zeile und wählt *Toggle Breakpoint* aus. An diesen Haltepunkten stoppt das Programm bei der Ausführung und man kann sich nun Schritt für Schritt angucken, wie sich Variablen verändern und welche Zeile was bewirkt. In der Abbildung 2.2: "Eclipse Debugger" auf der linken Seite sieht man, wo wir uns gerade beim Debuggen befinden. Rechts gibt es die Variables View, wo alle Variablen und ihre aktuellen Belegungen angezeigt werden. Weiterhin zählt die Breakpoint View alle Haltepunkte auf, die gesetzt wurden, wobei diese auch als blaue Punkte neben der Zeilennummerierung zu erkennen sind. Ein kleiner blauer

Pfeil ebenfalls neben der Zeilennummerierung zeigt an, welche Zeile aktuell gelesen wird. In der Tool Bar haben wir nun die Möglichkeit mithilfe des grünen Dreiecks das Programm zum nächsten Haltepunkt weiterlaufen zu lassen, das Debuggen mit dem roten Quadrat zu stoppen oder mit den gelben Pfeilen im das Debuggen zu navigieren.

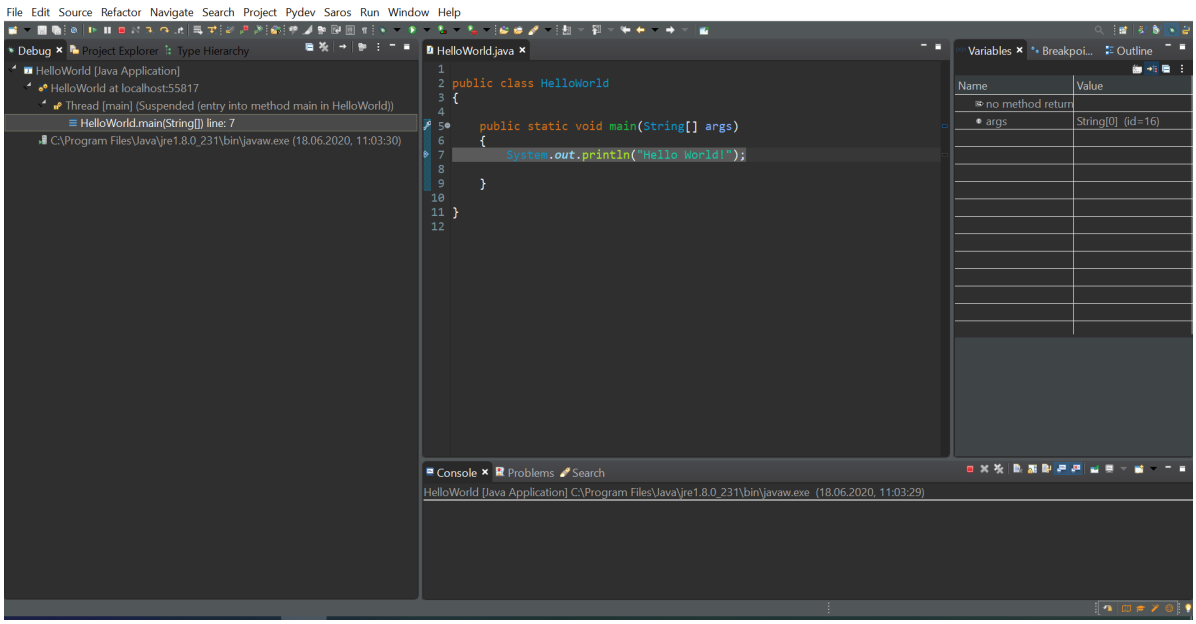

Figure 2.2: Eclipse Debugger

Natürlich gibt es noch weitere Views, die man sich mithilfe von $Windows \rightarrow Show\ View$ anzeigen lasssen kann. Auch die Anordnung und Größe der einzelnen Fenster lassen sich leicht verändern, indem man diese einfach per drag and drop verschiebt und an den Rändern entsprechend skaliert.

Kommen wir nun zu ein paar nützlichen Einstellungen, die wir unter $Window \rightarrow Preferences$ vornehmen.

1) Codierung der Text Dateien (UTF-8)
   Um einen üblichen Standard für die Text-Codierung einzustellen, muss man unter $General \rightarrow Workspace$ ganz unten bei $Text\ file$ unter $Other$ UTF-8 auswählen. Natürlich kann man auch andere Standards wie ASCII benutzen.

2) Rechtschreibprüfung
   Da man im Programmcode englische Schlüsselwörter benutzt, jedoch die Kommentare eher auf deutsch geschrieben werden, ist es ratsam, die automatische Rechtschreibprüfung komplett auszustellen, damit nicht alle Kommentare rot unterstrichen werden. Dies tut man, indem man unter $General \rightarrow Editors \rightarrow Text\ Editors \rightarrow Spelling$ das Häkchen bei $Enable\ Spell\ Checking$ herausnimmt.

3) Quelltextkonventionen
   Um die Form, in der der Code angezeigt wird, nach seinen persönlichen Quelltextkonventionen anzupassen, muss man zunächst eine Beispiel .xml-Datei erzeugen. Unter $Java \rightarrow Code\ Style \rightarrow Formatter$ kann man dann diese Datei importieren und anschließend auswählen.

4) Dark Theme
   Möchte man das Dark Theme von Eclipse nutzen, wählt man einfach bei $General \rightarrow Appearance$ unter $Theme$ Dark aus. Hier sind nach Belieben natürlich auch andere Themes zu nutzen.

Bei allen Änderungen, die man vornimmt, sollte man darauf achten, dass die ausgewählten Einstellungen immer mit Klick auf Apply auch wirklich bestätigt werden.

2.4 "Hello World!" in Eclipse

Schreiben wir nun ein kleines Hello World Programm in Eclipse mit der Programmiersprache Java. Dazu erstellen wir zunächst ein neues Java Projekt, indem wir auf $File \rightarrow New \rightarrow Project$ gehen $Java\ Project$ auswählen und $Next$ anklicken. Anschließend geben wir den Projektnamen Hello World ein und klicken auf $Finish$. Mit Rechtsklick auf den Projektordner fügen wir dem Ganzen eine neue Klasse mit $New \rightarrow Class$ hinzu. Wieder schreiben wir HelloWorld, dieses Mal ohne Leerzecihen. Dann setzen wir noch ein Häkchen bei $public\ static\ void\ main(String[]\ args)$[3] und klicken auf $Finish$. Nun schreiben wir ein Hello World Programm in die Klasse hinein. Der Code dafür sieht wie folgt aus.

```
1 public class HelloWorld
2 {
3     public static void main(String[] args)
4     {
5         System.out.println("Hello World!");
6     }
7 }
```

Listing 2.1: Programmcode Hello World

Jetzt speichern wir alles einmal ab, indem wir das Speicher-Symbol aus der Tool Bar nutzen und führen dann das Programm mithilfe des grünen Buttons mit weißem Dreieck, ebenfalls in der Tool Bar, aus. Wenn alles richtig funktioniert, sollte die Console nun Hello World! ausgeben.

[3] Dies ist die Hauptmethode in Java. Sie markiert den Start des Programms und muss irgendwo im Projekt auftauchen

3 Erweiterungen

In diesem Kapitel soll der Fokus darauf liegen, wie man Eclipse mithilfe von Plug-ins anpassen und erweitern kann, wovon besonders nützliche Tools kurz vorgestellt werden.

3.1 Plug-Ins

Allgemein kann man Plug-Ins für Eclipse in zwei Arten unterscheiden. Die einen, mit denen die Nutzung unterschiedlicher Programmiersprachen abseits von Java möglich ist und die anderen, die dem Entwickler allgemein beim Programmieren unterstützen sollen. Davon gibt es wiederum kostenlose und gebührenpflichtige Erweiterungen. Wenn man das Java Package für Eclipse ausgewählt hat, werden sämtliche Java Development Tools (JDT) mitgeliefert. Ähnlich ist das bei weiteren Packages für spezielle Programmiersprachen. Insgesamt findet man auf dem Eclipse Marketplace [mar], wo die meisten Plug-Ins heruntergeladen werden, Erweiterungen für Python, C++, Ruby, HTML, C und PHP, um nur einige der bekanntesten zu nennen. [wik20]

3.2 Weitere nützliche Tools

Nun werden ein paar kostenlose Tools, die Erleichterung schaffen sollen, kurz vorgestellt. Mithilfe von Git kann man sein Projekt in verschiedenen Versionen abspeichern. Das ist besonders dann hilfreich, wenn Projekte immer weiter entwickelt werden oder man auf eine ältere Version zurückgreifen möchte. [mar]
Will man gemeinsam programmieren kann man dafür Saros nutzen. Hier ist es möglich, dass mehrere Entwickler auf einmal den Code verändern. Dabei teilt der Host sein Projekt mit den anderen Nutzern, die das Projekt verändern dürfen. [mar]
Wer sich etwas Schreibaufwand ersparen möchte, kann sich Codota herunterladen. Dieses Tool vergleicht den eigenen Code mit Open-Source-Code und schlägt darauf aufbauend verschiedene Möglichkeiten zur Vervollständigung vor. [Bru19]
Testen ist ein elementarer Baustein in der Softwareentwicklung. Mit Jacoco soll das Test-Schreiben an sich unterstützt werden. Außerdem liefert das Tool nach Durchlaufen der Tests eine detaillierte Auswertung. [Bru19]

Schnittstellen mit dem Nutzer werden sehr oft mit Graphical User Interfaces (GUI) umgesetzt. Den Code für solche GUIs zu schreiben ist wegen der großen Möglichkeit an Eigenschaften und Anordnung der einzelnen Elemente sehr zeitaufwendig. Durch den WindowBuilder müssen Entwickler keinen Code mehr dafür schreiben, sondern können einfach per drag-and-drop Elemente hinzufügen und anschließend anpassen. So hat man direkt ein visuelles Feedback und der eigentliche Code wird vom Tool selbst generiert. [mar]

4 Zusammenfassung

Eclipse ist eine frei erhältliche IDE, die viele Einsatzmöglichkeiten hinsichtlich Anwendung, Betriebssystem und Programmiersprache bietet. Sie verbindet die wichtigsten Bausteine für Entwickler zu einer Oberfläche. Grundsätzlich war und ist diese auf Java ausgerichtet, allerdings lassen sich heutzutage mithilfe von Packages und Plug-Ins andere Sprachen in die IDE einbinden. Auch grundlegende Einstellungen und weitere Tools lassen sich je nach Projekt individuell einrichten. Zudem erfordert die Software aufgrund ihrer Komplexität etwas Geduld und Durchhaltevermögen.

5 Beurteilung

Betrachten wir zunächst einmal die Vorteile von Eclipse. Der wohl wichtigste Faktor bei Eclipse, ist die freie Zugänglichkeit. Sei es ein kleiner Hobby-Entwickler oder ein riesiges Unternehmen mit tausenden Mitarbeitern, jeder kann sich das Programm kostenlos herunterladen. Erst bei einigen der vielen Plug-Ins spielt das Geld eine Rolle. Im Gegenzug stehen dafür aber auch etliche frei erhältliche Erweiterungen zur Verfügung. Insgesamt ist Eclipse ein sehr mächtiges Werkzeug, denn es lassen sich ziemlich große Projekte für verschiedenste Bereiche damit umsetzen, wobei ein Großteil der Betriebssysteme und Programmiersprachen unterstützt wird. Im Vordergrund steht dabei immer die Erleichterung für den Entwickler, der sich dadurch komplett auf den Code konzentrieren kann.
Kommen wir nun zu den negativen Aspekten von Eclipse. Eine deutliche Einschränkung der Geschwindigkeit erkennt man, wenn mehr und mehr Projekte und Plug-Ins hinzukommen. Dies kann sehr hinderlich fürs Vorankommen sein, besonders unter Zeitdruck. Dadurch dass Eclipse eine Open-Source-Software ist, bietet sie keine Haftung für das Eclipse-Framework. Das heißt die Haftung für das geschriebene Programm, liegt immer beim Entwickler selbst. Es gibt auch keinen Support für Eclipse. Gerade bei großen Unternehmen stellt das ein Problem da, weil diese den Kundensupport für Eclipse dann selbst leisten müssen. Ein Nachteil für Anfänger ist die Komplexität der Software. Wer nicht die nötige Zeit mitbringt, tut sich sehr schwer bei der Entwicklung in solch einer IDE wie Eclipse. [Bre06]
Das allgemeine Fazit lautet: Eclipse ist eine komplexe IDE, mit der man große und kleine Projekte umsetzen kann. Theoretisch kann jeder die Software nutzen, es empfiehlt sich aber einige Grundkenntnisse mitzubringen. Eine klare Empfehlung für Java-Nutzer ist hier auszusprechen, allerdings können mittlerweile auch andere Programmiersprachen viel aus Eclipse herausholen.

Quellen

[Aug17a] Stephan Augsten. Was ist eclipse? https://www.dev-insider.de/was-ist-eclipse-a-602702/, 2017.

[Aug17b] Stephan Augsten. Was ist eine ide? https://www.dev-insider.de/was-ist-eine-ide-a-600703/, 2017.

[Bre06] Martin Brenda. Integrationsmöglichkeiten einer swing-anwendung in eclipse am beispiel des abaxxprocess modelers. https://hdms.bsz-bw.de/frontdoor/deliver/index/docId/511/file/Integrationsmoeglichkeiten_-einer_Swing_Anwendung_in_Eclipse_a.pdf:13–16, 2006.

[Bru19] Ilana Brudo. 14 best (and free) plugins for eclipse ide in 2019. https://blog.codota.com/14-free-plugins-for-eclipse-ide/, 2019.

[Dah20] Nitin Dahyabhai. Eclipse/installation. https://wiki.eclipse.org/Eclipse/Installation, 2020.

[fil14] filip. Java eclipse tutorial. https://javatutorial.net/java-eclipse-tutorial, 2014.

[mar] https://marketplace.eclipse.org/.

[tut] What is eclipse? https://www.tutorialspoint.com/eclipse/eclipse_-overview.htm.

[wik20] Eclipse (ide). https://de.wikipedia.org/wiki/Eclipse_(IDE), 2020.