

HANS-GEORG SCHUMANN

# JAVA

FÜR **KIDS**

8. AUFLAGE

DER EINFACHE EINSTIEG  
OHNE VORKENNTNISSE



## **Hinweis des Verlages zum Urheberrecht und Digitalen Rechtemanagement (DRM)**

Liebe Leserinnen und Leser,

dieses E-Book, einschließlich aller seiner Teile, ist urheberrechtlich geschützt. Mit dem Kauf räumen wir Ihnen das Recht ein, die Inhalte im Rahmen des geltenden Urheberrechts zu nutzen. Jede Verwertung außerhalb dieser Grenzen ist ohne unsere Zustimmung unzulässig und strafbar. Das gilt besonders für Vervielfältigungen, Übersetzungen sowie Einspeicherung und Verarbeitung in elektronischen Systemen.

Je nachdem wo Sie Ihr E-Book gekauft haben, kann dieser Shop das E-Book vor Missbrauch durch ein digitales Rechtemanagement schützen. Häufig erfolgt dies in Form eines nicht sichtbaren digitalen Wasserzeichens, das dann individuell pro Nutzer signiert ist. Angaben zu diesem DRM finden Sie auf den Seiten der jeweiligen Anbieter.

Beim Kauf des E-Books in unserem Verlagsshop ist Ihr E-Book DRM-frei.

Viele Grüße und viel Spaß beim Lesen,

*Ihr mitp-Verlagsteam*



## ***JAVA FÜR KIDS***

Neuerscheinungen, Praxistipps, Gratiskapitel,  
Einblicke in den Verlagsalltag –  
gibt es alles bei uns auf Instagram und Facebook



[instagram.com/mitp\\_verlag](https://www.instagram.com/mitp_verlag)



[facebook.com/mitp.verlag](https://www.facebook.com/mitp.verlag)



Hans-Georg Schumann

# ***JAVA FÜR KIDS***

***PROGRAMMIEREN LERNEN OHNE VORKENNTNISSE***



**mitp**

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 978-3-7475-0521-2

8. Auflage 2022

[www.mitp.de](http://www.mitp.de)

E-Mail: [mitp-verlag@sigloch.de](mailto:mitp-verlag@sigloch.de)

Telefon: +49 7953 / 7189 - 079

Telefax: +49 7953 / 7189 - 082

© 2022 mitp Verlags GmbH & Co. KG, Frechen

Dieses Werk, einschließlich aller seiner Teile, ist urheberrechtlich geschützt. Jede Verwertung außerhalb der engen Grenzen des Urheberrechtsgesetzes ist ohne Zustimmung des Verlages unzulässig und strafbar. Dies gilt insbesondere für Vervielfältigungen, Übersetzungen, Mikroverfilmungen und die Einspeicherung und Verarbeitung in elektronischen Systemen.

Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Werk berechtigt auch ohne besondere Kennzeichnung nicht zu der Annahme, dass solche Namen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürften.

Lektorat: Katja Vöpel

Sprachkorrektur: Petra Heubach-Erdmann

Covergestaltung: Christian Kalkert

Satz: Satz: III-satz, Flensburg, [www.drei-satz.de](http://www.drei-satz.de)

# INHALT

<b>EINLEITUNG</b> .....	11
Was heisst eigentlich Programmieren? .....	12
Was ist eine Entwicklungsumgebung? .....	12
Warum gerade Java? .....	13
Eclipse, die Entwicklungsumgebung zum Buch .....	13
Wie arbeitest du mit diesem Buch? .....	14
Was brauchst du für dieses Buch? .....	15
Hinweise für Lehrer .....	16
<b>ALLER ANFANG IST SCHWER: DAS ERSTE PROJEKT</b> .....	19
Eclipse starten .....	20
Willkommen in Java .....	23
Ein erstes Hallo .....	29
Objekte, Klassen und Pakete .....	33
Eclipse beenden .....	34
Zusammenfassung .....	39
Ein paar Fragen ... ..	40
... aber noch keine Aufgabe .....	40
<b>WENN, DANN, SONST: KONTROLLE UND AUSWAHL</b> .....	41
Eingabe und Ausgabe .....	42
Dialog mit »Schwung« .....	47
Gut oder schlecht? .....	50
Die if-Struktur .....	54
Strings oder Zahlen? .....	55
Plus oder minus, mal oder durch .....	58
Die if-else-Struktur .....	60
Zusammenfassung .....	61
Ein paar Fragen ... ..	62
... und ein paar Aufgaben .....	63

1

2

<b>3</b>	<b>ZENSUREN UND ZAHLENRATEN: BEDINGUNGEN</b> .....	65
	Von int zu float .....	66
	Die Sache mit try und catch .....	68
	Von 1 bis 6 .....	72
	Von Fall zu Fall .....	73
	Punkt für Punkt .....	76
	Und und Oder, oder? .....	78
	Ein kleines Spielchen .....	79
	Die while-Struktur .....	81
	Zusammenfassung .....	83
	Ein paar Fragen ... .....	84
	... und ein paar Aufgaben .....	85
<b>4</b>	<b>GELD-SPIELEREIEN: SCHLEIFEN UND FELDER</b> .....	87
	Dein PC zählt mit .....	88
	Abbruch bei Unlust? .....	89
	Auf dem Weg zum Millionär .....	91
	Schleifenvariationen .....	94
	Zählen mit for .....	95
	Variablenfelder .....	98
	Lottoziehung .....	101
	Feldsortierung .....	103
	Zusammenfassung .....	105
	Ein paar Fragen ... .....	106
	... und ein paar Aufgaben .....	106
<b>5</b>	<b>DO IT YOURSELF: FUNKTIONEN UND KLASSEN</b> .....	107
	Java ist lernfähig .....	108
	Funktionen fürs Ratespiel .....	111
	Lokal oder global? .....	113
	Parameter und Rückgabe .....	115
	Ein neues Baby? .....	118
	Laufen lernen ... .....	121
	Kapselung und Vererbung .....	124
	Zusammenfassung .....	128
	Ein paar Fragen ... .....	129
	... und ein paar Aufgaben .....	129
<b>6</b>	<b>NICHT NUR WAS FÜRS AUGE: EINSTIEG IN SWING</b> .....	131
	Erst mal ein Fenster .....	132
	Hallo, wie geht es? .....	134

Es passiert etwas .....	137
Gut oder schlecht? .....	140
Es gibt was zu erben .....	142
super, this und andere Erbstücke .....	145
Zusammenfassung .....	149
Ein paar Fragen ... ..	151
... und ein paar Aufgaben .....	151

<b>ALLES UNTER KONTROLLE: KOMPONENTENSAMMLUNG</b> .....	153
Kleine Knopfparade .....	153
Feldoptimierung .....	156
Listenwahl .....	158
Von Pünktchen ... ..	160
... und Häkchen .....	162
Körper, Geist und Seele .....	165
Der letzte Schliff .....	167
Zusammenfassung .....	169
Ein paar Fragen ... ..	171
... und ein paar Aufgaben .....	171

7

<b>WER WEISS WAS? QUIZ-PROJEKT TEIL 1</b> .....	173
Erst der Plan, dann der Bau .....	173
Frage und Antworten .....	176
Datensammlung .....	179
Datentransfer .....	182
Aufsammeln und einordnen .....	186
Zusammenfassung .....	189
Ein paar Fragen ... ..	190
... aber keine Aufgabe .....	190

8

<b>SPIELEN UND LERNEN: QUIZ-PROJEKT TEIL 2</b> .....	191
Du hast die Wahl .....	191
Aufgabenkontrolle .....	195
Antwort als Optionen .....	200
Vokabeln lernen? .....	202
Mehrfachauswahl .....	206
Zusammenfassung .....	208
Ein paar Fragen ... ..	209
... und ein paar Aufgaben .....	209

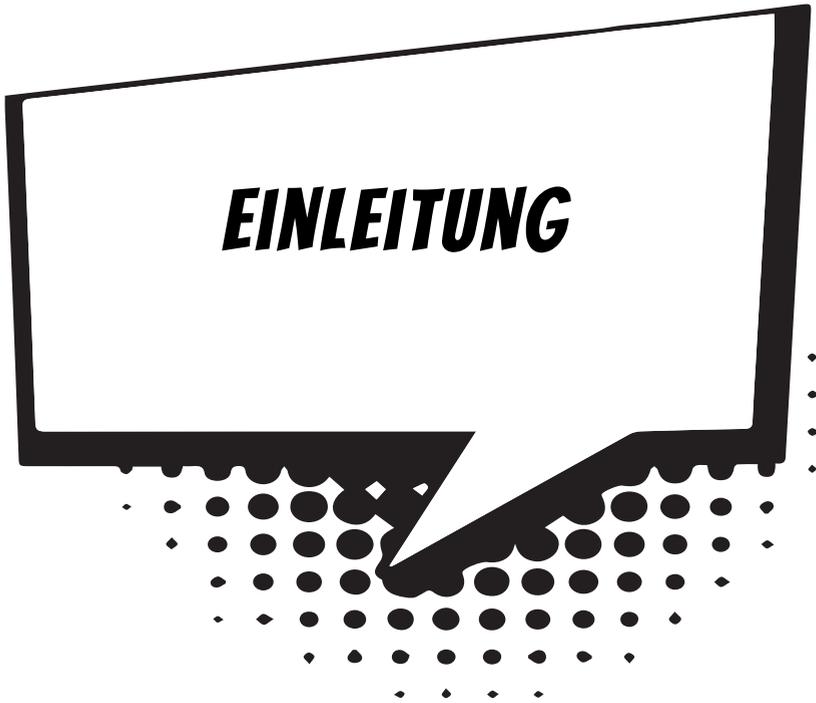
9

<b>10</b>	<b>JETZT WIRD'S BUNT: GRAFIK IN JAVA</b> .....	211
	Von Punkten und Koordinaten .....	212
	Das erste Bild .....	214
	Jetzt wird's bunt .....	218
	Eckig und rund .....	220
	Mit Text geht auch .....	222
	Farbtupfer .....	223
	Von Graphics zu Graphics2D .....	225
	Zusammenfassung .....	226
	Ein paar Fragen ... ..	227
	... und ein paar Aufgaben .....	227
<b>11</b>	<b>BILDER LERNEN LAUFEN: ANIMATIONEN</b> .....	229
	Erst mal ein Kreis .....	230
	Und es bewegt sich was .....	232
	Endlich ein (richtiges) Bild .....	235
	Bildersammlung .....	238
	Da läuft was .....	241
	Drehen, verschwinden, auftauchen .....	243
	Movie komplett .....	245
	Zusammenfassung .....	248
	Keine Fragen ... ..	248
	... doch ein paar Aufgaben .....	248
<b>12</b>	<b>BUNTES TRIO: KLEINE SPIELESAMMLUNG</b> .....	249
	Wie viele Augen? .....	249
	Anzeigen und auswerten .....	251
	Schere – Stein – Papier .....	253
	Qual der Wahl .....	256
	Das Nim-Spiel .....	257
	Streichholz-Parade .....	260
	Nur einer kann gewinnen .....	263
	Zusammenfassung .....	265
	Keine Fragen ... ..	265
	... aber zwei Aufgaben .....	265
<b>13</b>	<b>HANGMAN: RATEN ODER »BRATEN«</b> .....	267
	Anzeige und Eingabe .....	268
	Auf ein Wort .....	270
	Angst vor dem Galgen? .....	274
	Wortschatz .....	278

Komplettierung .....	279
Zusammenfassung .....	283
Keine Fragen ... ..	284
... und keine Aufgaben .....	284
<b>KLEINER KRABELKURS: TASTEN- UND MAUSSTEUERUNG</b> .....	285
Ein Käfer bewegt sich .....	286
Der KeyListener .....	287
Richtungswechsel .....	290
Kein Spiel ohne Grenzen .....	291
Maussteuerung .....	293
Ohne Mathe geht es nicht .....	296
Winkelberechnung .....	299
Zusammenfassung .....	303
Keine Fragen ... ..	304
... und keine Aufgaben .....	304
<b>INSEKTENJAGD: OPFER UND JÄGER</b> .....	305
Freilauf mit Kehrtwende .....	305
Runnable und run() .....	308
Klick und Platt .....	312
Neuerscheinung .....	314
Insektarium? .....	316
MyGame-Klasse komplett .....	319
Zusammenfassung (und Schluss) .....	322
Keine Fragen ... ..	322
... und keine Aufgaben .....	322
<b>ANHANG A</b> .....	323
Java installieren .....	323
Eclipse installieren .....	326
Einsatz der Buch-Dateien .....	331
<b>ANHANG B</b> .....	333
Kleine Checkliste .....	333
Dem Fehler auf der Spur .....	334
<b>STICHWORTVERZEICHNIS</b> .....	339

**14****15****A****B**

*Für  
Janne, Julia, Daniel und Katrin*



Java – ist das nun eine Insel mit vorwiegend warmem Klima? Oder eine Programmiersprache, die auch etwas mit dem Internet zu tun haben soll? Beides ist richtig, aber mit einem Reisebericht auf die besagte Insel kann ich hier nicht dienen, obwohl die gleichnamige Sprache durchaus etwas damit zu tun hat.

Entstanden ist diese Sprache, indem sich ihre Erfinder zuerst mal kräftig bei einer anderen Programmiersprache bedient haben, nämlich bei C++. Über die gäbe es eine Menge zu schreiben, u.a. muss man zugeben, dass sie die schwierigere Sprache von beiden ist. (Über C++ habe ich ein anderes Kids-Buch geschrieben.)

Ursprünglich wurde die Programmiersprache, um die es in diesem Buch geht, Oak genannt (zu Deutsch so viel wie »Eiche«). Kein besonders attraktiver Name, fand man schließlich, da passte dann der einer Südseeinsel schon besser.

Java zu lernen ist nicht zu schwierig, und dennoch hat diese Sprache die Fähigkeiten berühmter großer Konkurrenten wie z.B. C++, C# und Delphi. Java wird von der Firma Oracle verwaltet und kann kostenlos aus dem Internet heruntergeladen werden.

## **WAS HEISST EIGENTLICH PROGRAMMIEREN?**

Wenn du aufschreibst, was ein Computer tun soll, nennt man das Programmieren. Das Tolle daran ist, dass du selbst bestimmen kannst, was getan werden soll. Lässt du dein Programm laufen, macht der Computer die Sachen, die du ausgeheckt hast. Natürlich wird er dann dein Zimmer nicht aufräumen und dir auch keine Tasse Kakao ans Bett bringen. Aber kannst du erst mal programmieren, kannst du den Computer sozusagen nach deiner Pfeife tanzen lassen.

Allerdings passiert es gerade beim Programmieren, dass der Computer nicht so will, wie du es gerne hättest. Meistens ist das ein Fehler im Programm. Das Problem kann aber auch irgendwo anders im Computer oder im Betriebssystem liegen. Das Dumme bei Fehlern ist, dass sie sich gern so gut verstecken, dass die Suche danach schon manchen Programmierer zur Verzweiflung gebracht hat.

Vielleicht hast du nun trotzdem Lust bekommen, das Programmieren zu erlernen. Dann brauchst du ja nur noch eine passende Entwicklungsumgebung, und schon kann's losgehen.

## **WAS IST EINE ENTWICKLUNGSUMGEBUNG?**

Um ein Programm zu erstellen, musst du erst mal etwas eintippen. Das ist wie bei einem Brief oder einer Geschichte, die man schreibt. Das Textprogramm dafür kann sehr einfach sein, weil es ja nicht auf eine besondere Schrift oder Darstellung ankommt wie bei einem Brief oder einem Referat. So etwas wird Editor genannt.

Ist das Programm eingetippt, kann es der Computer nicht einfach so lesen und ausführen. Jetzt muss es so übersetzt werden, dass der PC versteht, was du von ihm willst. Weil er aber eine ganz andere Sprache spricht als du, muss ein Dolmetscher her.

Du programmierst in einer Sprache, die du verstehst, und der Dolmetscher übersetzt es so, dass es dem Computer verständlich wird. So etwas heißt dann Compiler. In Java klingt dieser Dolmetscher noch ein bisschen technischer: Die Java Virtual Machine (kurz JVM) ist eine Art »Zwischencomputer«.

Das heißt: Eigentlich wird ein Java-Programm an die JVM weitergereicht, die es dann für den jeweiligen Computer passend zubereitet: Das kann dann ein Windows-PC oder ein Linux-PC sein, ein Macintosh oder irgendein anderes Computersystem. Ein und dasselbe Java-Programm funktioniert so auf jedem beliebigen Computer, der über eine JVM verfügt. Viele Apps für das Android-System, das auf den meisten Smartphones läuft, wurden in Java programmiert.

Schließlich müssen Programme getestet, überarbeitet, verbessert, wieder getestet und weiterentwickelt werden. Dazu gibt es noch einige zusätzliche Hilfen. Daraus wird dann ein ganzes System, die Entwicklungsumgebung.

## **WARUM GERADE JAVA?**

Leider kannst du nicht so programmieren, wie dir der Mund gewachsen ist. Eine Programmiersprache muss so aufgebaut sein, dass möglichst viele Menschen in möglichst vielen Ländern einheitlich damit umgehen können.

Weil in der ganzen Welt Leute zu finden sind, die wenigstens ein paar Brocken Englisch können, besteht auch fast jede Programmiersprache aus englischen Wörtern. Es gab auch immer mal Versuche, z.B. in Deutsch zu programmieren, aber meistens klingen die Wörter dort so künstlich, dass man lieber wieder aufs Englische zurückgreift.

Eigentlich ist es egal, welche Programmiersprache du benutzt. Am besten eine, die möglichst leicht zu erlernen ist.

In diesem Buch hast du es mit der Programmiersprache Java zu tun. Sie ist inzwischen eine der Sprachen, die am meisten verbreitet sind. Sie ist nicht einfach, aber auch für Anfänger geeignet, die mit Java ihre erste Programmiersprache lernen wollen. (Willst du mal in andere Sprachen hineinschnuppern, dann empfehle ich dir z.B. eines der Kids-Bücher über C++, C#, JavaScript oder Python.)

Der Weg zum guten Programmierer kann ganz schön steinig sein. Nicht selten kommt es vor, dass man die Lust verliert, weil einfach gar nichts klappen will. Das Programm tut etwas ganz anderes, man kann den Fehler nicht finden und man fragt sich: Wozu soll ich eigentlich programmieren lernen, wo es doch schon genug Programme gibt? Und dann noch ausgerechnet in Java.

Zurzeit werden gute Programmierer dringend gesucht, und dieser Bedarf wird weiter steigen. In vielen Stellenanzeigen steht unter anderem oft »Programmierkenntnisse in Java erwünscht«. Wirklich gute Programmierer werden auch wirklich gut bezahlt. Es ist also nicht nur einen Versuch wert, es kann sich durchaus lohnen, das Programmieren in Java zu erlernen.

## **ECLIPSE, DIE ENTWICKLUNGSUMGEBUNG ZUM BUCH**

Um den Kauf einer Entwicklungsumgebung für Java musst du dich nicht weiter kümmern, denn die bekommst du kostenlos aus dem Internet. Mit der freien Software Eclipse hast du eine weitverbreitete Entwicklungsumgebung und kannst damit unter allen Versionen von Windows programmieren.

Das komplette Paket besteht aus zwei Teilen, die du von diesen Seiten herunterladen kannst:

[www.java.com/de/](http://www.java.com/de/)

[www.eclipse.org/](http://www.eclipse.org/)

## **UND WAS BIETET DIESES BUCH?**

Über eine ganze Reihe von Kapiteln verteilt lernst du

- ◇ die Grundlagen von Java kennen
- ◇ mit Eclipse unter Windows umzugehen
- ◇ einiges über die objektorientierte Programmierung (OOP)
- ◇ mit Komponenten zu arbeiten (das sind Bausteine, mit denen du dir viel Programmierarbeit sparen kannst)
- ◇ die grafischen Möglichkeiten von Java kennen
- ◇ eine Reihe von Spielen selber zu programmieren

Im **Anhang** gibt es dann noch einiges an Informationen und Hilfen, u.a. über Installationen und den Umgang mit Fehlern.

## **WIE ARBEITEST DU MIT DIESEM BUCH?**

Grundsätzlich besteht dieses Buch aus einer Menge Text mit vielen Abbildungen dazwischen. Natürlich habe ich mich bemüht, alles so zuzubereiten, dass daraus lauter gut verdauliche Happen werden. Damit das Ganze noch genießbarer wird, gibt es zusätzlich noch einige Symbole, die ich dir hier gern erklären möchte:

### **ARBEITSSCHRITTE**

- Wenn du dieses Zeichen siehst, heißt das: Es gibt etwas zu tun. Damit kommen wir beim Programmieren Schritt für Schritt einem neuen Ziel immer näher.

Grundsätzlich lernt man besser, wenn man einen Programmtext selbst eintippt oder ändert. Aber nicht immer hat man große Lust dazu. Deshalb gibt es alle Projekte im Buch auch als Download:

<https://www.mitp.de/0520>

Und hinter einem Programmierschritt findest du zusammen mit diesem Pfeil → auch den jeweiligen Namen des Projekts oder einer Datei (z.B. → PROJEKT1, → GRAFIK1, → GAME1). Wenn du also das Projekt nicht selbst erstellen willst, kannst du stattdessen diese Datei laden (zu finden im Ordner PROJEKTE).

### **AUFGABEN**

Am Ende eines Kapitels findest du jeweils eine Reihe von Fragen und Aufgaben. Diese Übungen sind nicht immer ganz einfach, aber sie helfen dir, noch besser zu programmieren. Lösungen zu den Aufgaben findest du in verschiedenen Formaten

## **WAS BRAUCHST DU FÜR DIESES BUCH?**

---

ebenfalls bei den Download-Dateien. Du kannst sie dir alle im Editor von Windows oder auch in deinem Textverarbeitungsprogramm anschauen. Oder du lässt sie dir ausdrucken und hast sie dann schwarz auf weiß, um sie neben deinen PC zu legen. (Auch die Programme zu den Aufgaben liegen im Ordner PROJEKTE.)

### **NOTFÄLLE**

Vielleicht hast du irgendetwas falsch gemacht oder etwas vergessen. Oder es wird gerade knifflig. Dann fragst du dich, was du nun tun sollst. Bei diesem Symbol findest du eine Lösungsmöglichkeit. Es kann nicht schaden, auch mal ganz hinten im Anhang B nachzuschauen, wo ein paar Hinweise zur Pannenhilfe aufgeführt sind.



### **ACHTUNG**

Hin und wieder findest du ein solch dickes Ausrufezeichen im Buch. Dann ist das eine Stelle, an der etwas besonders Wichtiges steht.



### **SPEZIALWISSEN**

Wenn du ein solches »Wow« siehst, geht es um ausführlichere Informationen zu einem Thema.



### **SPICKZETTEL**

Unter den Downloaddateien findest du zudem Spickzettel zum Ausdrucken. Sie helfen dir beim Java-Programmieren nicht den Überblick zu verlieren.

## **WAS BRAUCHST DU FÜR DIESES BUCH?**

Installiert wird Java mit dem Programm SETUP in ein Verzeichnis deiner Wahl, z.B. c:\PROGRAMME\JAVA. Auch für Eclipse gibt es ein Installationsprogramm, dafür solltest du dann einen Extra-Ordner benutzen, in dem du später auch deine Java-Projekte unterbringst.

Die Beispielprogramme in diesem Buch gibt es alle als Download von der Homepage des Verlages, falls du mal keine Lust zum Abtippen hast:

<https://www.mitp.de/0520>

Und auch die Lösungen zu den Fragen und Aufgaben sind dort untergebracht (alles im Ordner PROJEKTE).

## **BETRIEBSSYSTEM**

Die meisten Computer arbeiten heute mit dem Betriebssystem Windows. Davon brauchst du eine der Versionen 7 bis 10. (Java gibt es bei Oracle u.a. auch für Linux.)

## **SPEICHERMEDIEN**

Auch wenn du deine Programme auf der Festplatte unterbringst, kann es nicht schaden, sie zusätzlich z.B. auf einem USB-Stick als Backup zu speichern.

Gegebenenfalls bitte deine Eltern oder Lehrer um Hilfe.

# **HINWEISE FÜR LEHRER**

Dieses Buch versteht sich auch als Lernwerk für den Informatik-Unterricht in der Schule. Dort setzt natürlich jeder Lehrer seine eigenen Schwerpunkte. Benutzen Sie an Ihrer Schule bereits ein Werk aus einem Schulbuchverlag, so lässt sich dieses Buch auch als Materialienband einsetzen – in Ergänzung zu dem vorhandenen Schulbuch. Weil dieses Buch sozusagen »von null« anfängt, ist ein direkter Einstieg in Java möglich – ohne irgendwelche anderen Programmierkenntnisse.

Ein wichtiger Schwerpunkt in diesem Buch ist die objektorientierte Programmierung (OOP). Auf die wichtigsten Eigenheiten (Kapselung, Vererbung und Polymorphie) wird ausführlich eingegangen.

In den Projekten werden alle wesentlichen Elemente des Java-Wortschatzes wie auch die wichtigsten Grafik-Komponenten eingesetzt. Außerdem erfährt man hier, wie man eine eigene Komponente erstellen kann. Ein besonderer Schwerpunkt liegt auf der Spieleprogrammierung.

In den Lösungen zu den Aufgaben finden Sie weitere Vorschläge zur Programmierung in Java.

## **ÜBUNGSMEDIEN**

Für den Informatik-Unterricht sollte jeder Schüler ein anderes externes Speichermedium haben, um darauf seine Programmierversuche zu sichern. So wird verhindert, dass sich auf der Festplatte des Schulcomputers mit der Zeit allerlei »Datenmüll« ansammelt. Außerdem dient der eigene Datenträger dem Datenschutz: Nur der betreffende Schüler kann seine Daten manipulieren.

## **REGELMÄßIG SICHERN**

Es kann nicht schaden, die Programmdateien, an denen gerade gearbeitet wird, etwa alle zehn Minuten zu speichern. Denn Computer pflegen gern gerade dann »abzustürzen«, wenn man seine Arbeit längere Zeit nicht gespeichert hat.

Das ist aber nur dann nötig, wenn man ein Programm längere Zeit nicht startet. In der Regel fragt nämlich Eclipse bei jedem Programmstart nach, ob die Datei gespeichert werden soll.





# **1 ALLER ANFANG IST SCHWER: DAS ERSTE PROJEKT**

Du willst gleich mit dem ersten Programm loslegen? Deinen Computer kannst du schon mal anschalten und Windows starten lassen. Dann machen wir gemeinsam die ersten Schritte in Java. Es wird eine Weile dauern, bis du dann dein erstes Programm erschaffen hast, aber wie der Titel des Kapitels schon sagt: Aller Anfang ist schwer.

In diesem Kapitel lernst du

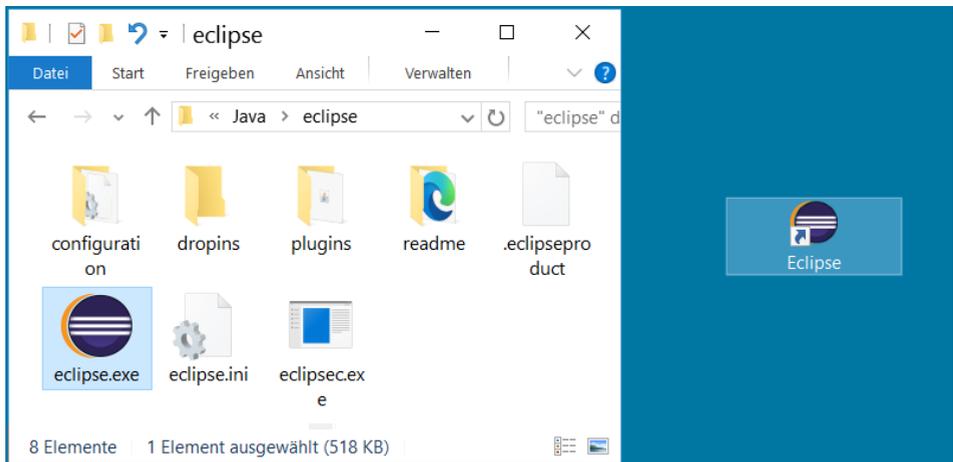
- ⊙ wie man Eclipse startet
- ⊙ wie man ein Programmprojekt erstellt und ausführt
- ⊙ aus welchen »Schalen« ein Java-Projekt besteht
- ⊙ wie man einen Text anzeigen kann
- ⊙ wie man Eclipse beendet

## ECLIPSE STARTEN

Bevor wir mit dem Programmieren anfangen können, muss Eclipse erst installiert werden. Genaueres erfährst du im Anhang A. Hier musst du dir von jemandem helfen lassen, wenn du dir das Einrichten nicht allein zutraust.

Wenn Eclipse verfügbar ist, hast du diese zwei Möglichkeiten, dieses Programm zu starten:

- Du öffnest den Ordner, in dem du Eclipse untergebracht hast, und doppelklickst mit der Maus auf das Symbol mit dem Namen ECLIPSE.EXE.
- Oder du doppelklickst auf das Desktop-Symbol mit dem Namen Eclipse – wenn es vorhanden ist.



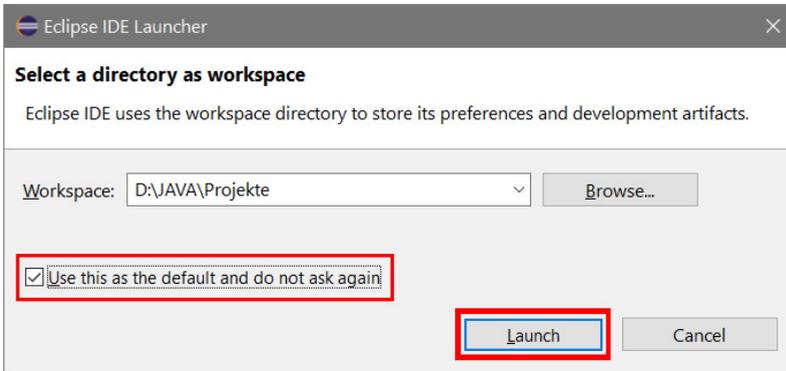
Wie kommt das Eclipse-Symbol auf den Desktop? Du brauchst dazu eine **Verknüpfung**; das bedeutet, du legst ein Symbol auf dem Desktop an, das mit dem Programm ECLIPSE.EXE verbunden ist. So etwas nennt man Verknüpfung.

- ❖ Dazu klickst du im ECLIPSE-Ordner mit der rechten Maustaste auf das Symbol für ECLIPSE.EXE. Im Kontextmenü wählst du KOPIEREN.
- ❖ Dann klicke auf eine freie Stelle auf dem Desktop, ebenfalls mit der rechten Maustaste. Im Kontextmenü wählst du VERKNÜPFUNG EINFÜGEN.
- ❖ Es ist sinnvoll, für das neue Symbol auf dem Desktop den Text ECLIPSE.EXE – VERKNÜPFUNG durch ECLIPSE zu ersetzen.

Von nun an kannst du auf das neue Symbol doppelklicken und damit Eclipse starten.



Je nach Computer kann es eine Weile dauern, bis Eclipse geladen ist. Zwischen-  
durch fordert ein Dialogfeld einen Ordner an, in dem deine Projekte untergebracht  
werden – Workspace genannt.



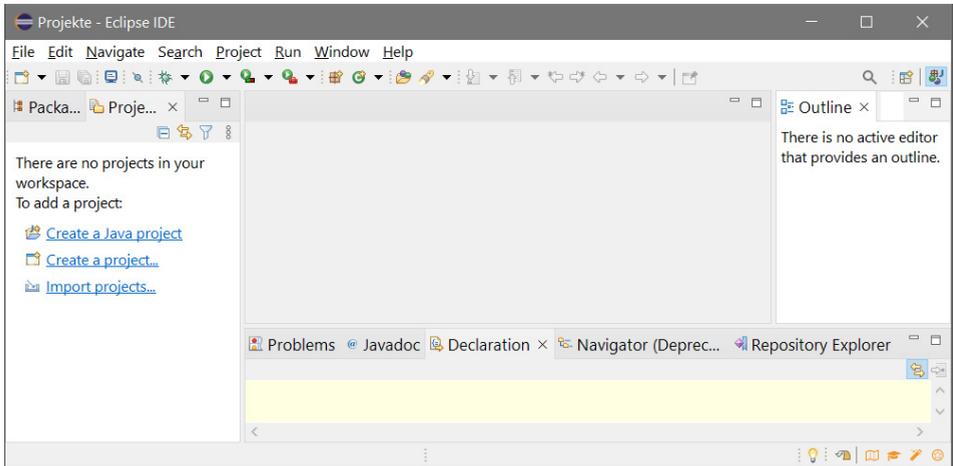
- Du kannst das Verzeichnis so lassen oder ein eigenes angeben. Sorge auf jeden  
Fall dafür, dass vor dem Eintrag USE THIS AS DEFAULT AND DO NOT ASK AGAIN ein  
Häkchen steht (sonst erscheint dieses Dialogfeld bei jedem deiner nächsten  
Starts von Eclipse immer wieder). Dann klicke auf OK.

Einige Zeit später landest du in einem Willkommens-Fenster.



- Hier klickst du oben rechts auf das Symbol, unter dem HIDE steht.

Damit schließt sich das Willkommens-Fenster, und was dich erwartet, könnte etwa so aussehen:



Für den ersten Augenblick ist das alles sicher ein bisschen sehr verwirrend. Nicht nur ein Fenster, sondern gleich ein paar Fensterbereiche tummeln sich da auf dem Bildschirm.

Ganz oben kann man die Menüleiste erkennen. Darunter befinden sich mehrere Symbole, die man mit der Maus anklicken kann.



Diese Menüs von Eclipse wirst du wahrscheinlich am meisten benutzen:

Über das FILE-Menü kannst du Dateien speichern, laden (öffnen), ausdrucken, neu erstellen oder Eclipse beenden.

Das Menü EDIT hilft dir bei der Bearbeitung deines Programmtextes, aber auch bei anderen Programmelementen. Außerdem kannst du dort bestimmte Arbeitsschritte rückgängig machen oder wiederherstellen.

Über das PROJECT-Menü verwaltest du dein aktuelles Projekt.

Das RUN-Menü bietet dir Möglichkeiten für die Ausführung des aktuellen Programms.

Und das HELP-Menü bietet dir vielfältige Hilfsinformationen (auf Englisch) an.

Dein Hauptarbeitsplatz ist eigentlich eine Fenstergruppe. Dazu gehört ein Editorfenster, wie du es vielleicht von einem Texteditor oder Textverarbeitungsprogramm her kennst. Welcher Bereich das ist, wirst du schon bald sehen.

Ein Editor ist ein Programm oder ein Fensterbereich, in das oder dem man eine größere Menge Text eingeben und bearbeiten kann.



## WILLKOMMEN IN JAVA

Nun kann es mit dem Programmieren losgehen. Den Umgang mit Menüs und Dialogfenstern kennst du bereits von Windows. Deshalb müssen wir uns damit nicht mehr aufhalten. Bauen wir uns jetzt ein kleines Projekt, für das erst einmal einige Vorbereitungen nötig sind.

Zuerst ein kurzer Überblick. Für jedes neue Projekt sind drei Schritte nötig:

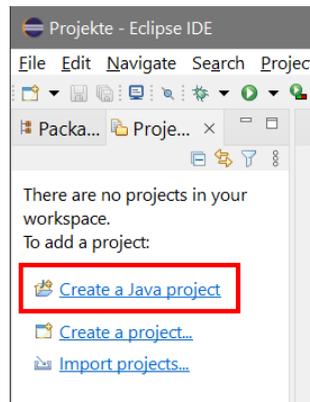
- ❖ das Projekt erzeugen (JAVA PROJECT), das umfasst einen Ordner, in dem dann alles drin ist, was zum Projekt gehört
- ❖ ein Paket erzeugen (PACKAGE), darin sind alle Bibliotheken und Werkzeuge, die das Programm benötigt
- ❖ eine Klasse erzeugen (CLASS), darin befindet sich dann der Programmtext, den du selber schreibst (auch Quelltext genannt)

Leider musst du dir diese Mühe jedes Mal machen, wenn du ein neues Projekt anlegen willst. Du erfährst aber auch noch, wie man aus alten schon vorhandenen Projekten ein neues machen kann.

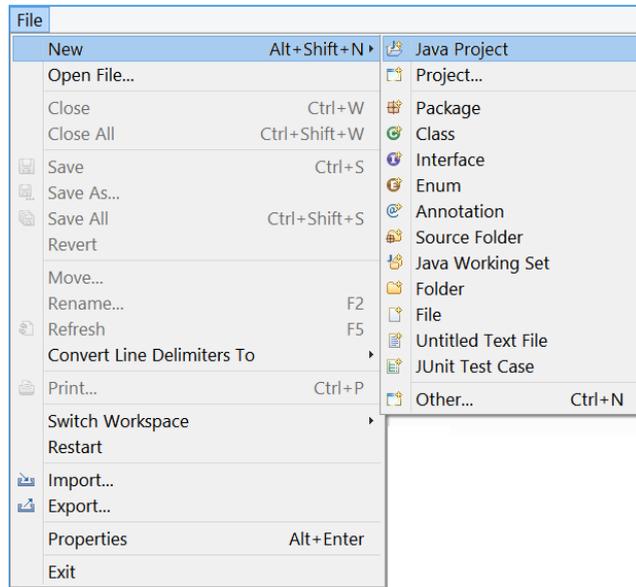
Übrigens bestehen in Java Paketnamen nur aus Kleinbuchstaben, während Klassennamen mit einem Großbuchstaben beginnen (sollten).



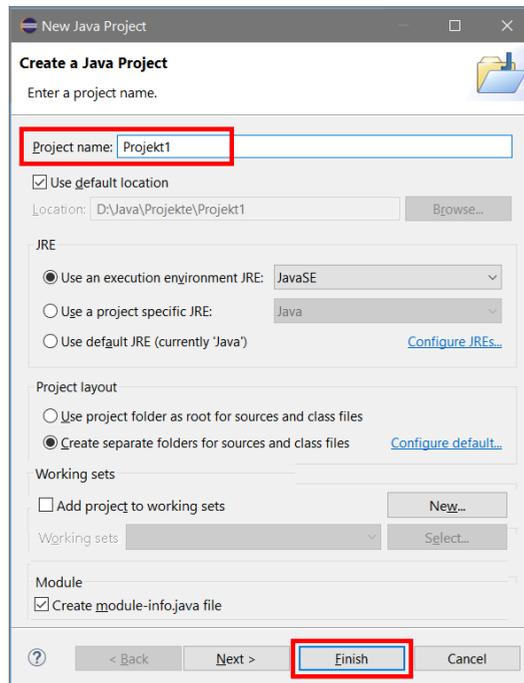
- Für ein neues Projekt kannst du nun im linken Fensterbereich direkt auf den Eintrag CREATE A JAVA PROJECT klicken.



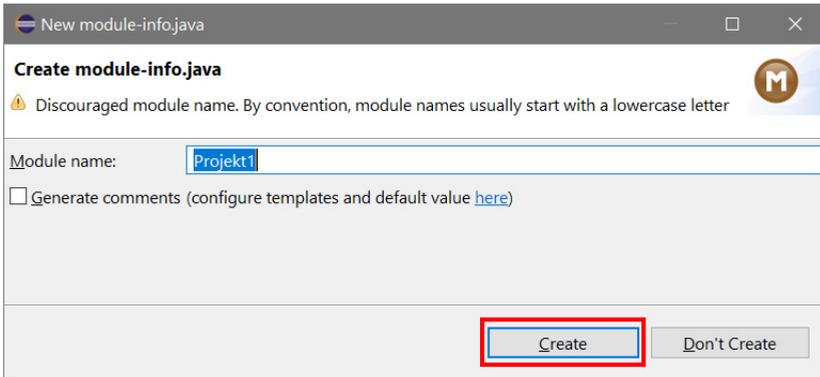
➤ Oder du klickst auf FILE und dann auf NEW und JAVA PROJECT.



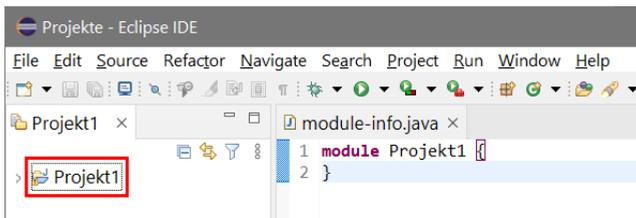
➤ Im nächsten Dialogfeld tippst du hinter PROJECT NAME PROJEKT1 (oder einen Namen deiner Wahl) ein.



➤ Sonst lass alles, wie es ist, und klicke anschließend auf FINISH.

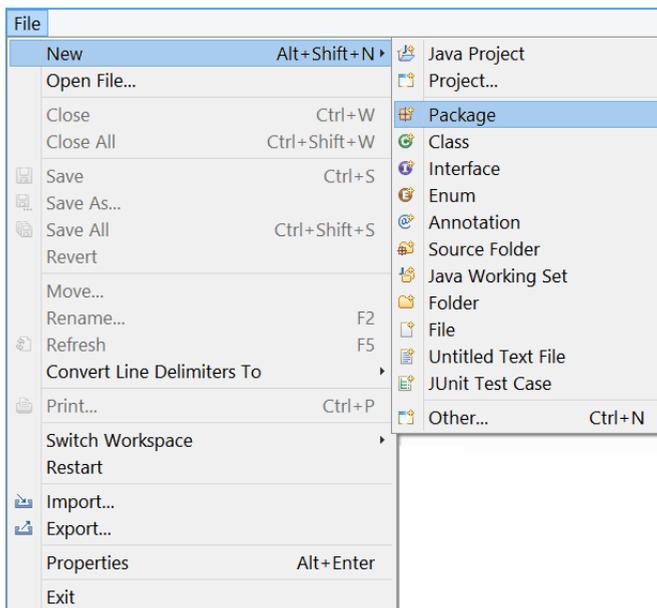


➤ Nun kommt ein weiteres Dialogfeld, in dem du nur auf CREATE klicken solltest. Einige Zeit später steht im linken Fenster der Name deines ersten Projekts.



Daneben steht schon der erste Programmtext, den du aber nicht weiter beachten musst.

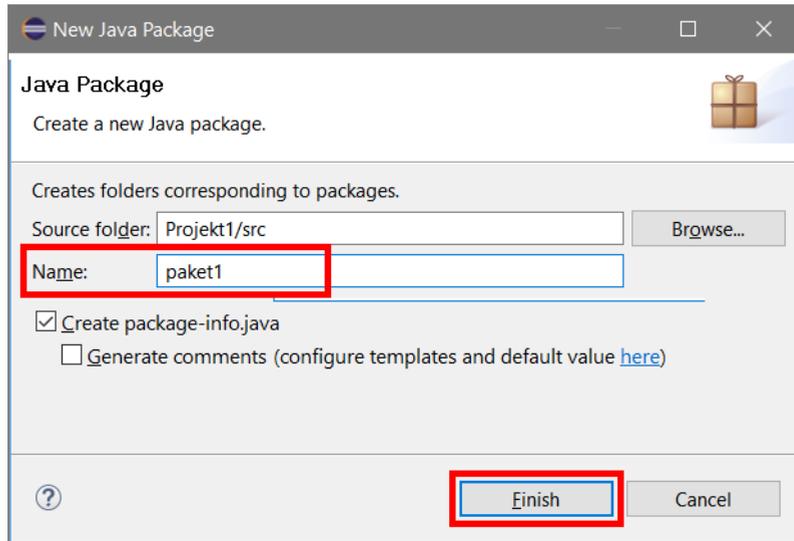
➤ Im nächsten Schritt klickst du auf FILE und dann auf NEW und PACKAGE.



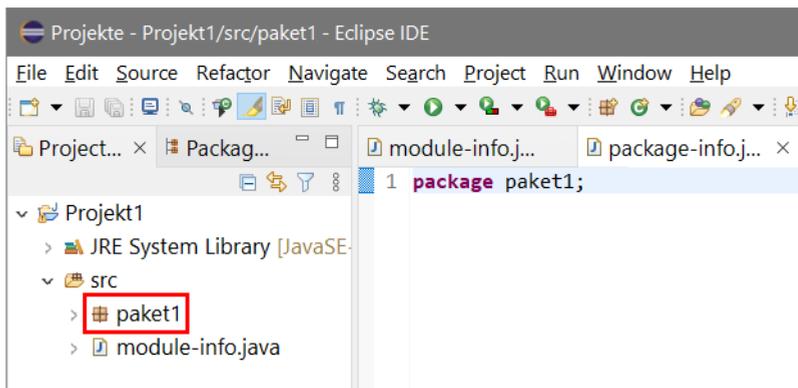
- Im Dialogfeld NEW JAVA PACKAGE tippst du hinter NAME PAKET1 (oder wieder einen Namen deiner Wahl) ein. Dann klicke auf FINISH.



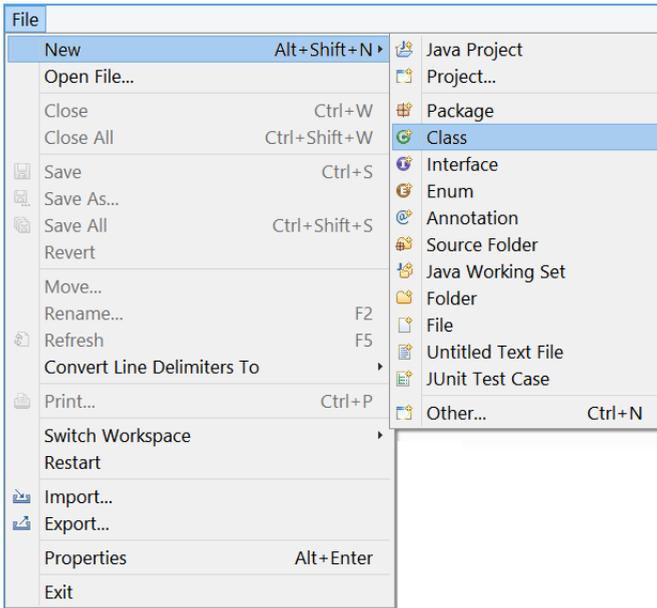
Noch mal zur Erinnerung: Die Namen für Java-Pakete werden in der Regel kleingeschrieben (was nicht zwingend nötig, aber üblich ist).



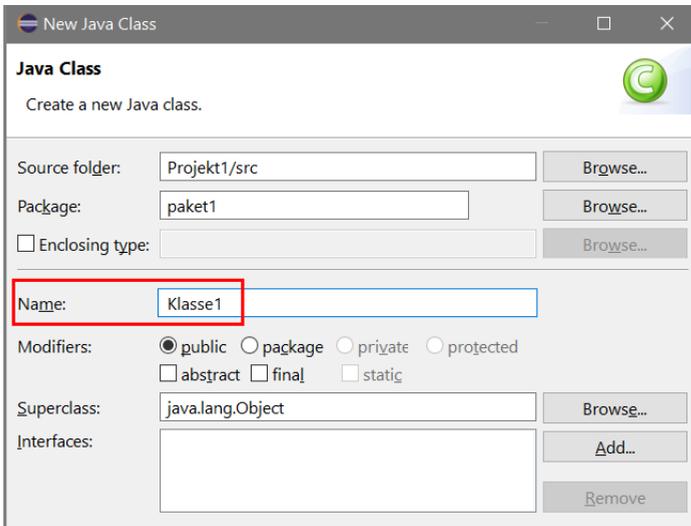
Und schon findet sich ein weiterer Eintrag im linken Fenster. Und rechts daneben steht ein weiterer (kurzer) Programmtext.



- Nun kommt der vorläufig letzte Schritt: Klicke auf FILE und auf NEW und CLASS.



➤ Im Dialogfeld NEW JAVA CLASS sollten oben bereits PROJEKT1 und PAKET1 eingetragen sein. Wenn nicht, findest du einen fehlenden Eintrag über BROWSE. Hinter NAME tipst du KLASSE1 ein.



Genau genommen steht hinter SOURCE FOLDER: PROJEKT1/SRC. Die letzten drei Buchstaben sind eine Abkürzung für »Source«, was auf Deutsch eigentlich »Quelle« heißt, hier ist damit gemeint, dass in diesem Ordner die Dateien mit dem Quelltext von Java zu finden sind, also die Dateien, in denen dein Programmtext steht.



- Sorge außerdem dafür, dass vor `PUBLIC STATIC VOID MAIN (STRING[] ARGS)` ein Häkchen steht, damit du ein lauffähiges Programm erhältst. Dann schließe das Dialogfeld mit Klick auf `FINISH`.

Which method stubs would you like to create?

`public static void main(String[] args)`

Constructors from superclass

Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))

Generate comments

? Finish Cancel



Damit der Computer ein Java-Programm ausführen kann, muss es innerhalb der Klasse eine Hauptfunktion geben, die den passenden Namen `main` trägt. Wie du weiter unten sehen wirst, trägst du auch die Anweisungen, die der Computer nach dem Programmstart ausführen soll, dort ein.

Nun tut sich einiges mehr, denn einen Moment später sieht das Fenstersystem von Eclipse so aus:

Projekte - Projekt1/src/paket1/Klasse1.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Project... x Packag... x

Projekt1

- JRE System Library [JavaSE...]
- src
  - paket1
    - Klasse1.java**
    - package-info.java
    - module-info.java

module-info.j... package-info.j... **Klasse1.java x**

```

1 package paket1;
2
3 public class Klasse1 {
4
5     public static void main(String[] args) {
6         // TODO Auto-generated method stub
7     }
8 }
9
10 }
11

```



Wenn du zu voreilig auf `FINISH` geklickt hast, musst du diese Klasse löschen und eine neue erstellen. Das Löschen funktioniert so: Name der Klasse markieren und die Taste `Entf` drücken.

Inzwischen weißt du sicher auch, wo der Editor ist, also der Bereich, wo du den Quelltext eintippen kannst. In der Mitte steht unter dem Titel `KLASSE1.JAVA` dieser Text – auch Quelltext oder Programmtext genannt:

```
package paket1;
public class Klasse1 {
    public static void main (String[] args) {
        // TODO Auto-generated method stub
    }
}
```

Was das im Einzelnen bedeutet, lässt sich erst nach und nach klären. Eine Zeile ist Kommentartext, den wir im Folgenden nicht benötigen.

Ein **Kommentar** oder eine Erläuterung wird immer mit zwei Schrägstrichen (//) eingeleitet. Dieser Text wird nicht zum ausführbaren Programm gezählt, er ist nur für den Programmierer von Bedeutung.

Der Computer überspringt die Zeilen mit Kommentaren, für ihn sind sie uninteressant. Aber du als Programmierer kannst dort wichtige Bemerkungen zum Programm unterbringen, z.B.:

```
// Spiel-Start
// Kreis-Berechnung
```

Kommentare werden dir in den nächsten Kapiteln immer wieder begegnen.



## EIN ERSTES HALLO

Eigentlich könntest du dieses Programmprojekt schon laufen lassen (über das RUN-Menü). Zu sehen bekommen würdest du aber nichts, denn das Programm tut im Moment noch nichts für uns Sichtbares.

Das lässt sich aber schnell ändern, wenn du die folgende Anweisung hinzufügst:

```
System.out.println ("Hallo, wer bist du?");
```

➤ Ergänze den neuen Text an der passenden Stelle und lösche die Kommentarzeile mit den zwei Schrägstrichen. Passe alles so an, dass es anschließend so aussieht:

```
package paket1;
public class Klasse1 {
    public static void main (String[] args) {
        System.out.println ("Hallo, wer bist du?");
    }
}
```

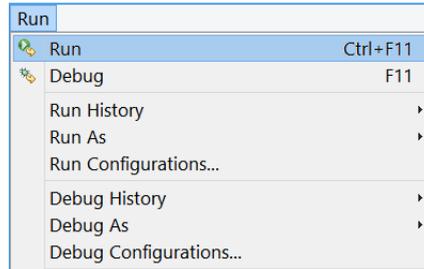


Falls du geschweifte Klammern mal selbst eintippen musst – und irgendwann musst du das bestimmt: Mit `[AltGr] [7]` erhältst du die öffnende Klammer, mit `[AltGr] [0]` die schließende Klammer.

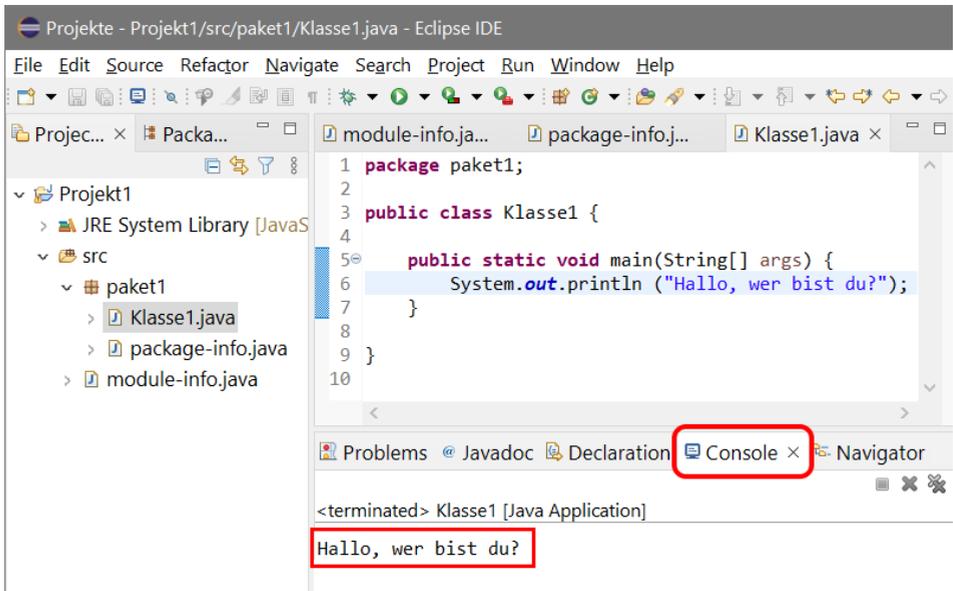
Wie du siehst, wird eine Anweisung immer mit einem Semikolon (;) abgeschlossen.

Und nun machen wir unseren ersten Probelauf:

➤ Klicke in der Menüleiste auf RUN und dann noch mal auf den Eintrag RUN.



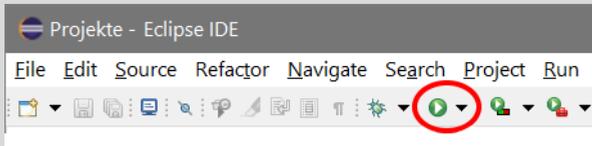
Und es dauert nicht lange, bis ganz unten im Fenster unter CONSOLE der Text »Hallo, wer bist du?« erscheint:



Vielleicht musst du noch etwas genauer hinschauen und das untere Fensterchen mit dem Titel CONSOLE etwas vergrößern, um den Anzeigetext zu erkennen.

Ein bisschen dürftig, aber immerhin haben wir jetzt schon unser erstes Java-Programm erstellt.

Eine weitere Möglichkeit, dein Programm zum Laufen zu bringen, ist ein Klick auf das linke grüne Pfeilsymbol direkt unter der Menüleiste.

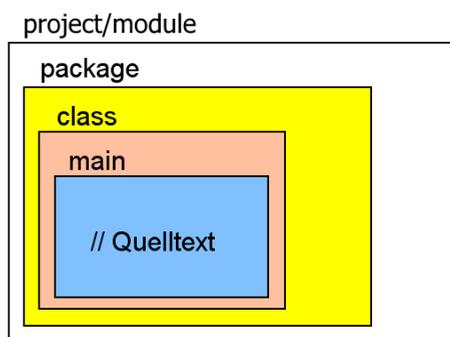


Betrachten wir unser Werk einmal genauer. Ein Projekt ist in seiner einfachsten Form so aufgebaut:

```
package paket1;
public class Klasse1 {
    public static void main (String[] args) {
    }
}
```

Was `public` und `static` bedeuten, dazu kommen wir erst in einem späteren Kapitel. Schauen wir mal auf die »Verschachtelung«, die irgendwie an ein Zwiebelsystem erinnert: Die Außenhaut ist das Projekt mit eigenem Ordner. Darin liegt ein Paket (englisch: `package`). Offenbar kann ein Projekt auch aus mehr als einem Paket bestehen. Im Paket-Ordner finden wir die Daten einer Klasse (englisch: `class`). Auch hier liegt die Vermutung nahe, dass es mehrere Klassen geben kann.

Und als ob es nicht schon genug wäre, gibt es darin noch etwas mit dem Namen `main`. Das ist der Hauptprogrammteil. Man nennt es auch die Hauptfunktion oder `main`-Methode.



Sehr wichtig sind die geschweiften Klammern (`{ }`). Dazwischen stehen die Zeilen, die dem Programm erst richtig zum Leben verhelfen. Und die stammen größtenteils von uns. Das ist jetzt erst mal nur eine Zeile, aber wir sind ja noch am Anfang.



Zu jeder öffnenden Klammer »{« muss es auch eine schließende Klammer »}« geben! Wo genau du die Klammern hinsetzt, ist Geschmackssache. Der obige Programmtext könnte also auch so aussehen:

```
public class Klasse1
{
    public static void main (String[] args)
    {
        // hier stehen deine Anweisungen
    }
}
```

Die zwei **Schrägstriche** (//) benutzen wir immer, wenn wir einen **Kommentar** bzw. eine Bemerkung einsetzen wollen.

Jetzt willst du endlich wissen, was diese eine Anweisung bedeutet, mit deren Hilfe der Computer offenbar bereit ist, dich freundlich zu grüßen:

```
System.out.println ("Hallo, wer bist du?");
```

Fangen wir von hinten an. Dort steht der Text, der angezeigt werden soll, eingepackt in Anführungsstriche und zusätzlich in Klammern:

```
("Hallo, wer bist du?")
```

Für die Anzeige sorgt die Anweisung `println()`, was eine Abkürzung für `PrintLine` ist und ausführlich heißt: »Schreib etwas auf dem Bildschirm und gehe danach in die nächste Zeile.«

Eine solche Anweisung nennt man auch **Methode**. In Java gibt es zahlreiche Methoden, sie gehören immer zu einer Klasse oder einem sogenannten Objekt – das hier `System.out` heißt. Was genau das bedeutet, erfährst du schon bald.

Man spricht bei `println()` auch von einer **Funktion**. Methode und Funktion meinen also dasselbe. Und das, was in den runden Klammern steht, wird als **Parameter** bezeichnet.



Es ist übrigens nicht egal, ob für die Wörter große oder kleine Buchstaben benutzt werden. Java unterscheidet eindeutig zwischen Groß- und Kleinschreibung.

Lass am besten stehen, was Eclipse dir bereits vorgibt. Und achte beim Eintippen genau darauf, wann mal ein großer Buchstabe zwischen den vielen Kleinbuchstaben steht. Du kannst also nicht `Main` oder `MAIN` statt `main` schreiben!

## OBJEKTE, KLASSEN UND PAKETE

Bevor wir weiter programmieren, sollten wir uns erst einmal mit der »Religion« von Java auseinandersetzen. Java ist eine Programmiersprache, die auch das Erstellen von professioneller Software ermöglicht.

Einfachste Programmiersprachen machen es dem Anfänger leicht: Ein oder zwei Zeilen genügen und schon erscheint ein netter Gruß wie »Hallo«. Ein paar Zeilen mehr zaubern Zahlen, weiteren Text oder sogar eine Grafik auf den Bildschirm.

Geht es jedoch um große Projekte, so ist man in diesen einfachen Systemen schnell überfordert, wenn man in einer solchen Sprache programmiert. Vor allem aber wächst die Anfälligkeit eines Projekts für Fehler. Und die dann alle zu entdecken, kann zur Qual werden.

Java macht es einem Anfänger nicht so leicht, es zwingt von Anfang an zu einem sauberen klaren Programmierstil – und das kostet zuerst mehr Mühe und bringt auch einigen Frust mit sich. Dass es sich dennoch lohnt, wirst du sehen, wenn du die ersten Kapitel überstanden hast.

Für ein Projekt in Java beschaffst du dir zuerst ein (leeres) Paket. Wie das geht, hast du ja weiter oben schon gesehen. Dort kommen dann alle die Sachen hinein, die du dir mit der Zeit zurechtprogrammierst. Damit die ganzen Elemente nicht einfach so in der Paketschachtel herumliegen, sind sie zu Objekten bzw. Klassen zusammengefasst.

**Objekte?** Das sind doch eigentlich diese Dinge, die ständig irgendwo herumstehen oder sich um uns herum bewegen. Also z.B. Häuser, Bäume, Autos, Leute. Auch du bist ein Objekt. Und zwar vom Typ Mensch. Objekte in Java sind natürlich nur künstlich.

Dabei kann es in Java durchaus mehrere Objekte eines Typs geben – so wie es im richtigen Leben auch (viele) verschiedene Menschen gibt. Daher spricht man hier von **Objekttyp**, womit dasselbe gemeint ist wie mit **Klasse**. Und ein Objekt wird auch als **Instanz** einer Klasse bezeichnet. Demnach bist du eine Instanz der Klasse Mensch.



Jedes Java-Programm stellt viele Klassen und Methoden automatisch zur Verfügung. Dazu gehört die Klasse mit dem Namen `System`, die drei wichtige Bereiche anzubieten hat:

<code>System.out</code>	für die Standardausgabe	z.B. über Monitor
<code>System.err</code>	für die Fehlerausgabe	z.B. über Monitor
<code>System.in</code>	für die Standardeingabe	z.B. von Tastatur