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

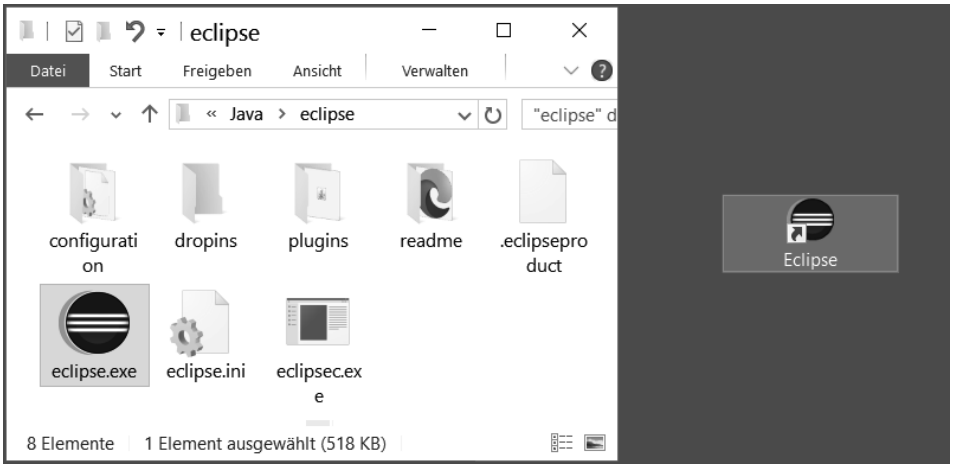
- wie man Eclipse startet
- wie man ein Programmprojekt erstellt und ausführt
- aus welchen »Schalen« ein Java-Projekt besteht
- wie man einen Text anzeigen kann
- wie man Eclipse beendet

ECLIPSE STARTEN

Bevor wir mit dem Programmieren anfangen können, muss Eclipse erst installiert werden. Genaues 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.

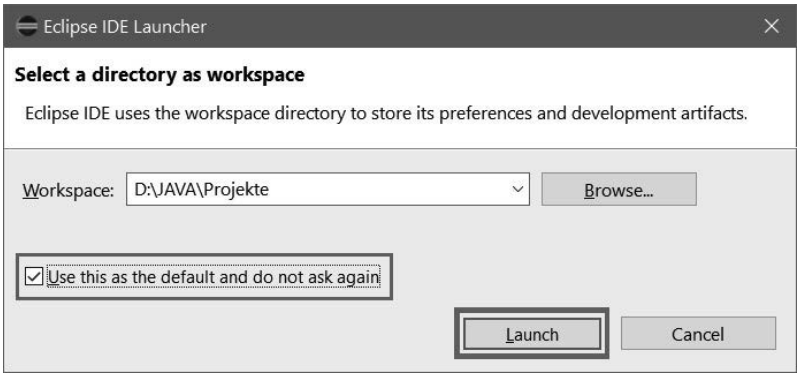


Wie kommt das Eclipse-Symbol auf den Desktop? Du brauchst dazu eine **Verknüpfung**; das bedeutet, du legst ein Symbol auf dem Desktop an, das mit dem Programm ECLIPSE.EXE verbunden ist. So etwas nennt man Verknüpfung.

- 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. Zwischendurch fordert ein Dialogfeld einen Ordner an, in dem deine Projekte untergebracht werden – Workspace genannt.

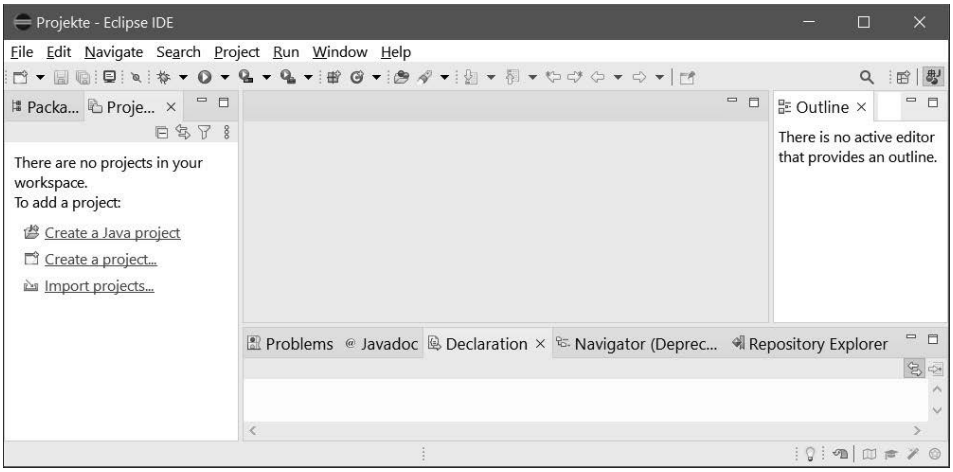


≫ Du kannst das Verzeichnis so lassen oder ein eigenes angeben. Sorge auf jeden Fall dafür, dass vor dem Eintrag USE THIS AS DEFAULT AND DO NOT ASK AGAIN ein Häkchen steht (sonst erscheint dieses Dialogfeld bei jedem deiner nächsten Starts von Eclipse immer wieder). Dann klicke auf OK.

Einige Zeit später landest du in einem Willkommens-Fenster.

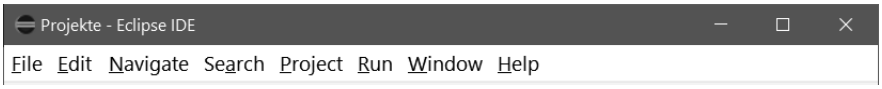
≫ Hier klickst du oben rechts auf das Symbol, unter dem HIDE steht.

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.

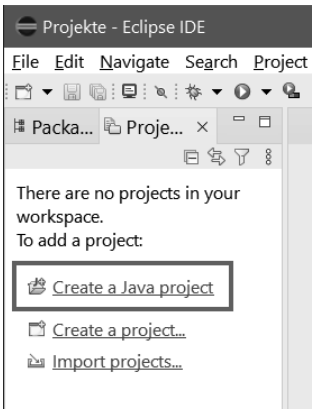
Zuerst ein kurzer Überblick. Für **jedes** neue Projekt sind drei Schritte nötig:

- 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)

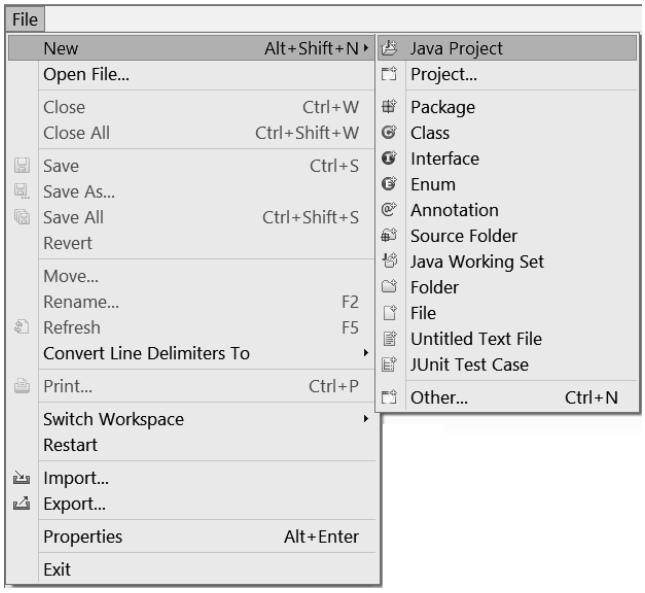
Leider musst du dir diese Mühe jedes Mal machen, wenn du ein neues Projekt anlegen willst. Du erfährst aber auch noch, wie man aus alten schon vorhandenen Projekten ein neues machen kann.

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

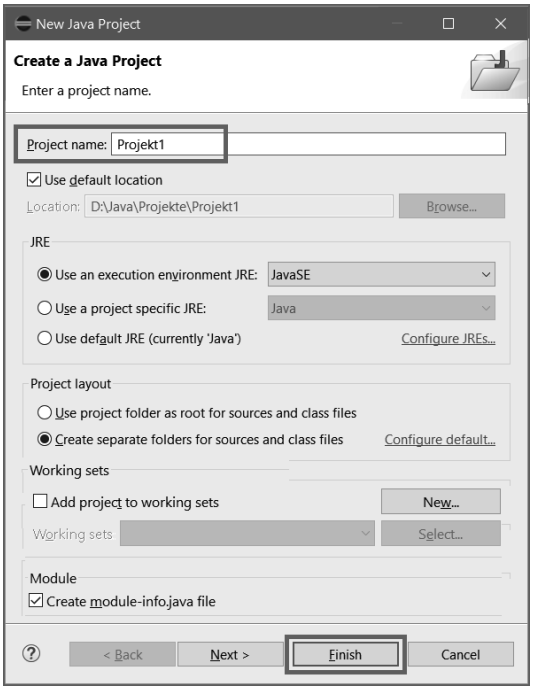
≫ Für ein neues Projekt kannst du nun im linken Fensterbereich direkt auf den Eintrag CREATE A JAVA PROJECT klicken.



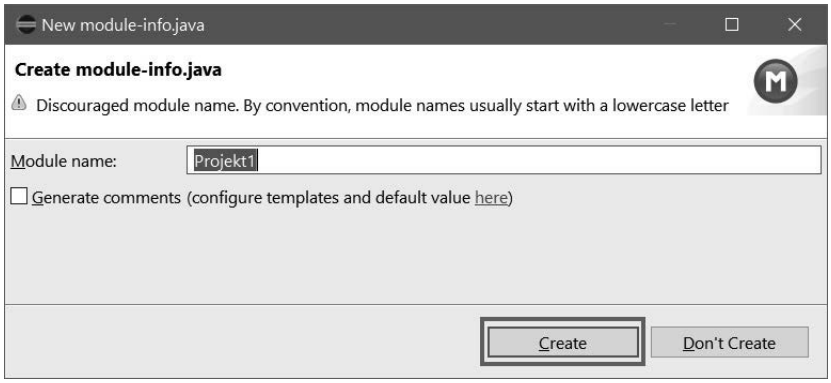
≫ Oder du klickst auf FILE und dann auf NEW und JAVA PROJECT.



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

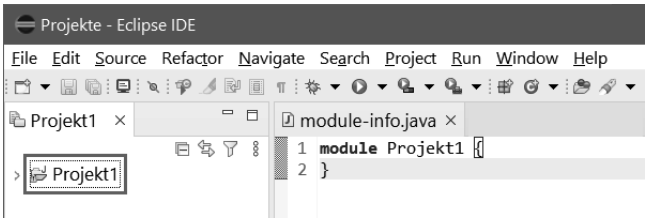


≫ Sonst lass alles, wie es ist, und klicke anschließend auf FINISH.



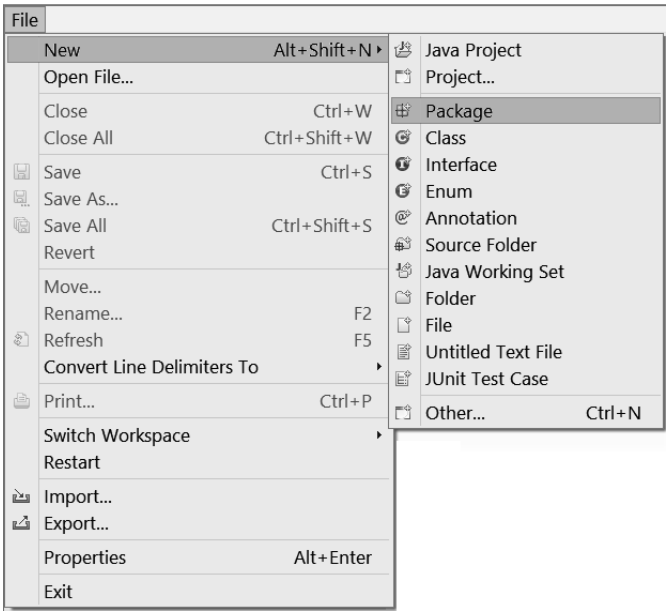
≫ Nun kommt ein weiteres Dialogfeld, in dem du nur auf CREATE klicken solltest.

Einige Zeit später steht im linken Fenster der Name deines ersten Projekts.



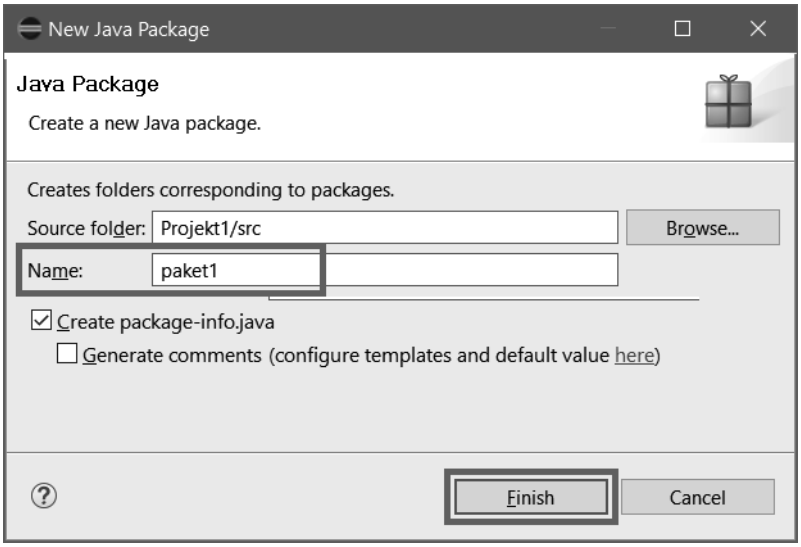
Daneben steht schon der erste Programmtext, den du aber nicht weiter beachten musst.

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

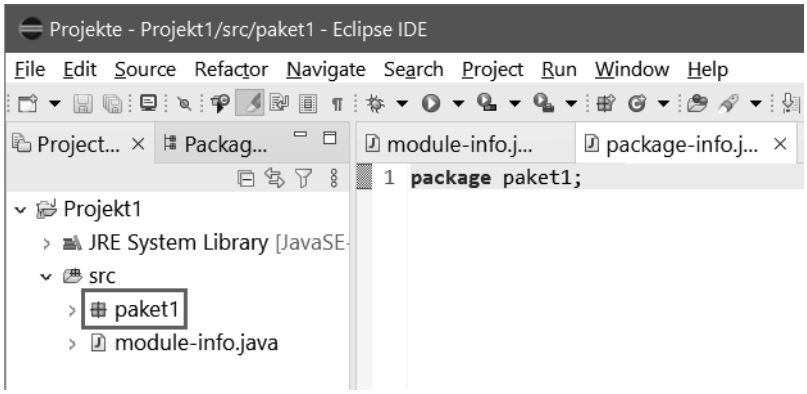


≫ Im Dialogfeld NEW JAVA PACKAGE tippst du hinter NAME PAKET1 (oder wieder einen Namen deiner Wahl) ein. Dann klicke auf FINISH.

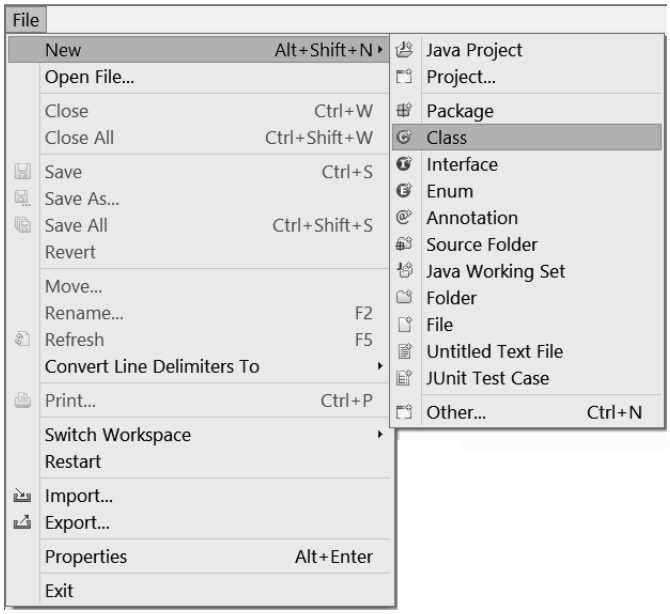
Noch mal zur Erinnerung: Die Namen für Java-Pakete werden in der Regel kleingeschrieben (was nicht zwingend nötig, aber üblich ist).



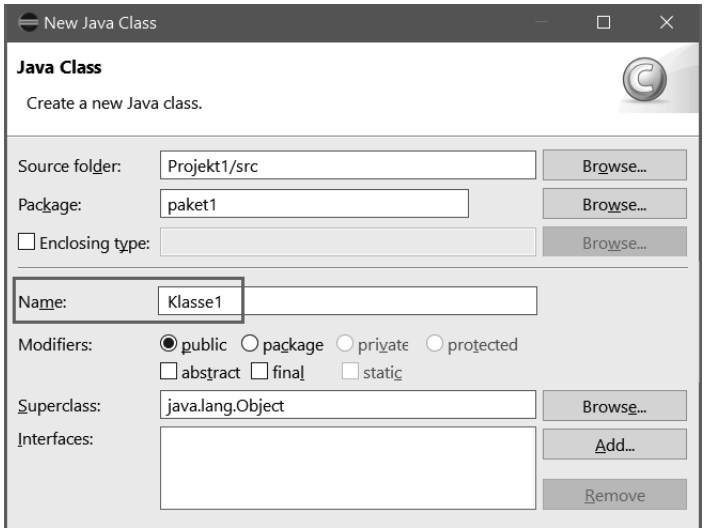
Und schon findet sich ein weiterer Eintrag im linken Fenster. Und rechts daneben steht ein weiterer (kurzer) Programmtext.



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

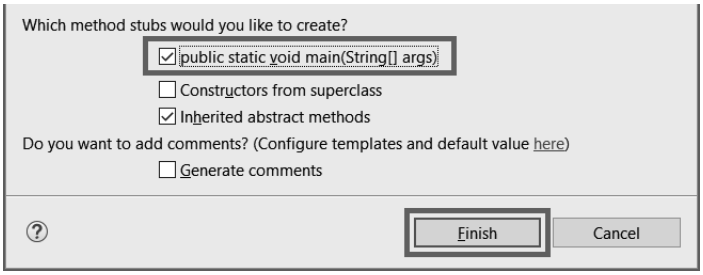


≫ Im Dialogfeld NEW JAVA CLASS sollten oben bereits PROJEKT1 und PAKET1 eingetragen sein. Wenn nicht, findest du einen fehlenden Eintrag über BROWSE. Hinter NAME tippst du KLASSE1 ein.



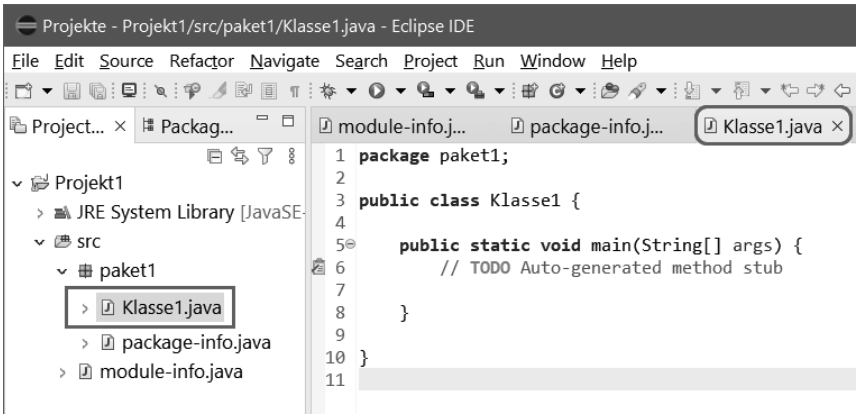
Genau genommen steht hinter SOURCE FOLDER: PROJEKT1/SRC. Die letzten drei Buchstaben sind eine Abkürzung für »Source«, was auf Deutsch eigentlich »Quelle« heißt, hier ist damit gemeint, dass in diesem Ordner die Dateien mit dem Quelltext von Java zu finden sind, also die Dateien, in denen dein Programmtext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



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:



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:

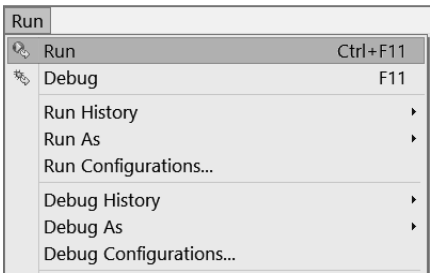
```
package paket1;
public class Klasse1 {
  public static void main (String[] args) {
    System.out.println ("Hallo, wer bist du?");
  }
}
```

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.

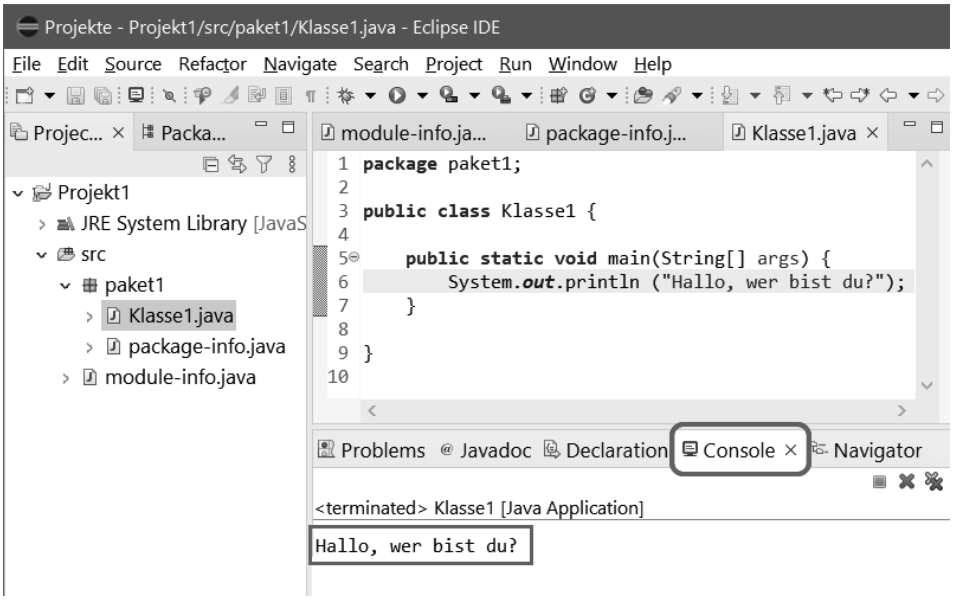
Wie du siehst, wird eine Anweisung **immer** mit einem Semikolon (;) abgeschlossen.

Und nun machen wir unseren ersten Probelauf:

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



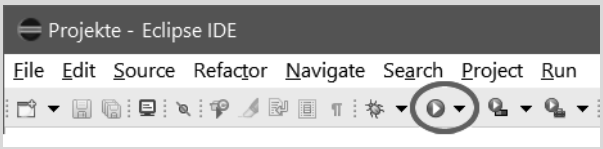
Und es dauert nicht lange, bis ganz unten im Fenster unter CONSOLE der Text »Hallo, wer bist du?« erscheint:



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.

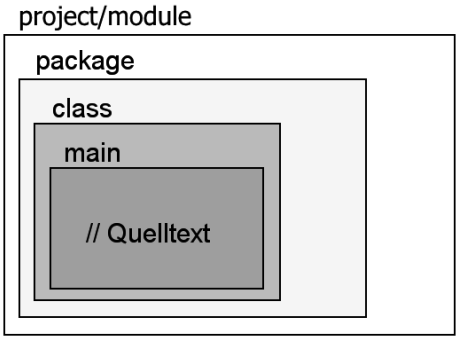


Betrachten wir unser Werk einmal genauer. Ein Projekt ist in seiner einfachsten Form so aufgebaut:

```
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 Dinger, 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.

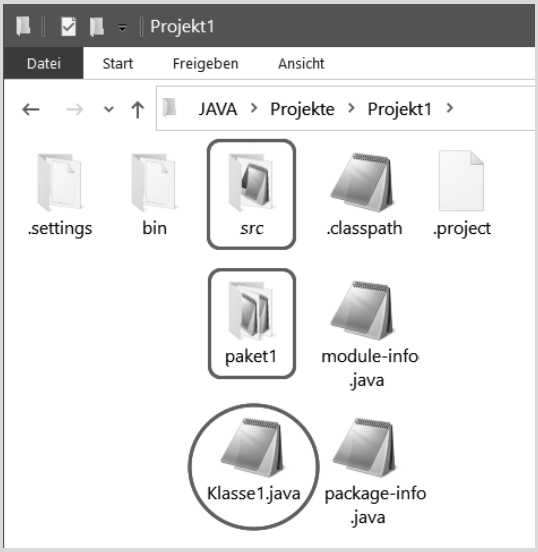
Jedes Java-Programm stellt viele Klassen und Methoden automatisch zur Verfügung. Dazu gehört die Klasse mit dem Namen `System`, die drei wichtige Bereiche anzubieten hat:

`System.out`	für die Standardausgabe	z.B. über Monitor
`System.err`	für die Fehlerausgabe	z.B. über Monitor
`System.in`	für die Standardeingabe	z.B. von Tastatur

Du wirst im Laufe dieses Buches weitere Klassen und Objekte kennenlernen und später auch selbst erstellen. Klassen werden in einer `package` zusammengefasst, wie ein Java-Paket genannt wird. (Einige gut gefüllte Päckchen stellt Java ja bereits »von Haus aus« zur Verfügung.)

Wenn du übrigens mal in den Ordner für dein Projekt hineinschaust, siehst du, dass es hier gleich eine ganze Reihe von Ordnern und Dateien gibt:

- Dateien, die den Programm- bzw. Quelltext beinhalten, werden mit der Kennung JAVA gespeichert und liegen im Ordner SRC.
- Klasseninformationen werden in Dateien mit der Kennung CLASS abgelegt. Sie sind im Ordner BIN zu Hause.



Für uns von Interesse ist nur der Ordner SRC. Dort findest du auch die Datei KLASSE1.JAVA.

Daneben gibt es noch einige zusätzliche Dateien, deren Bedeutung du hier nicht kennen musst.

ECLIPSE BEENDEN

Nun wird es Zeit für eine kleine Pause, auch wenn unser Projekt noch ziemlich mager aussieht. Denn du hast ja hier schon recht viel geleistet. Eclipse sorgt dafür, dass deine Daten sicher auf deiner Festplatte oder SSD landen. Du kannst aber alles über FILE und eine der SAVE-Optionen auch immer wieder selbst speichern.

Möglicherweise hast du keine Lust, immerzu alles abzutippen, deshalb kannst du **alle** Beispiel-Projekte zu diesem Buch auch als ZIP-Paket von der Homepage des mitp-Verlages herunterladen. Benutze dazu bitte diese Adresse:

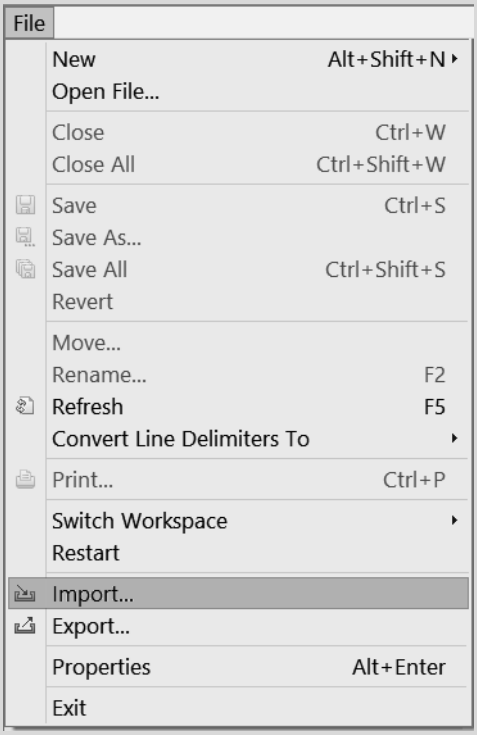
https://www.mitp.de/0520

Du musst dann nur das Paket entpacken und den betreffenden Ordner für dein aktuelles Projekt suchen.

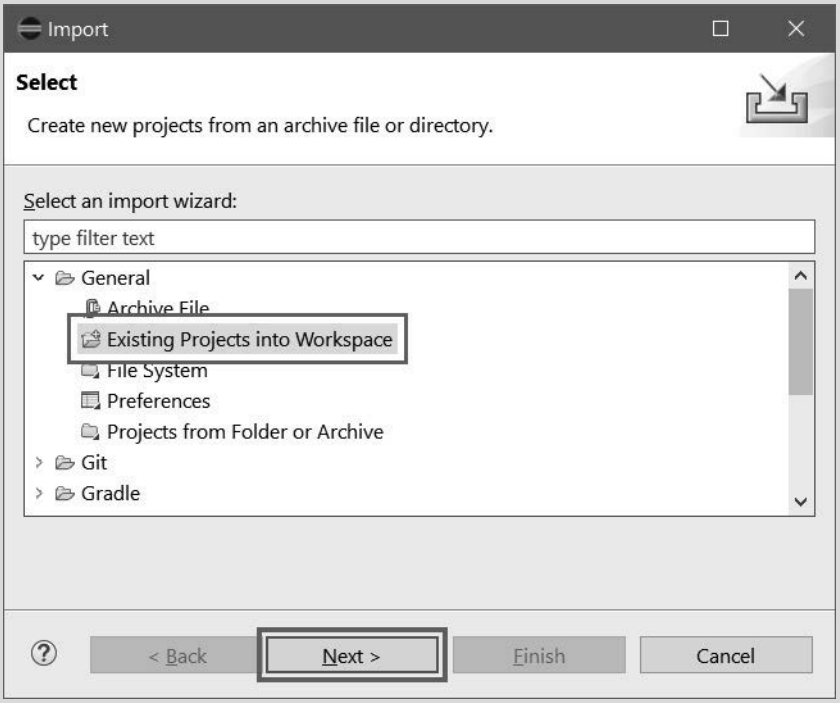
Nun weißt du möglicherweise nicht, wie du Eclipse dazu bringen kannst, ein solches Projekt zu übernehmen. Eine Möglichkeit wäre, einfach den Quelltext der entsprechenden Dateien mit der Kennung JAVA zu markieren und nach dem Erzeugen eines neuen Projekts in das Editorfenster von Eclipse zu kopieren.

Aber dieser Weg ist umständlich und es gibt ja eine elegantere Methode, direkt in der Eclipse-Umgebung, die du nutzen solltest:

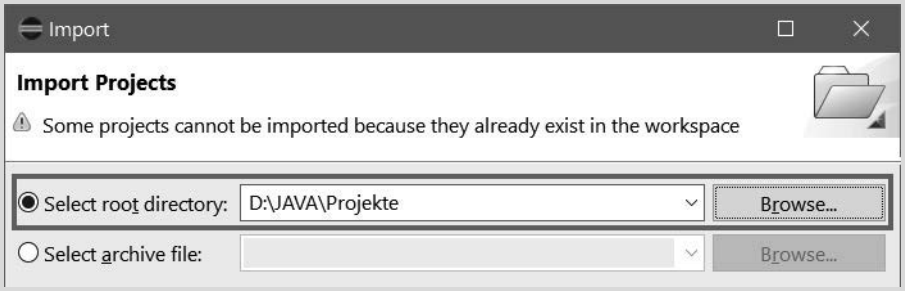
Klicke auf FILE und IMPORT.



Öffne im Dialogfeld den Eintrag GENERAL, markiere dort EXISTING PROJECTS INTO WORKSPACE und klicke dann auf NEXT.



Im nächsten Dialogfenster suchst du über BROWSE den Ordner mit den Projekten.

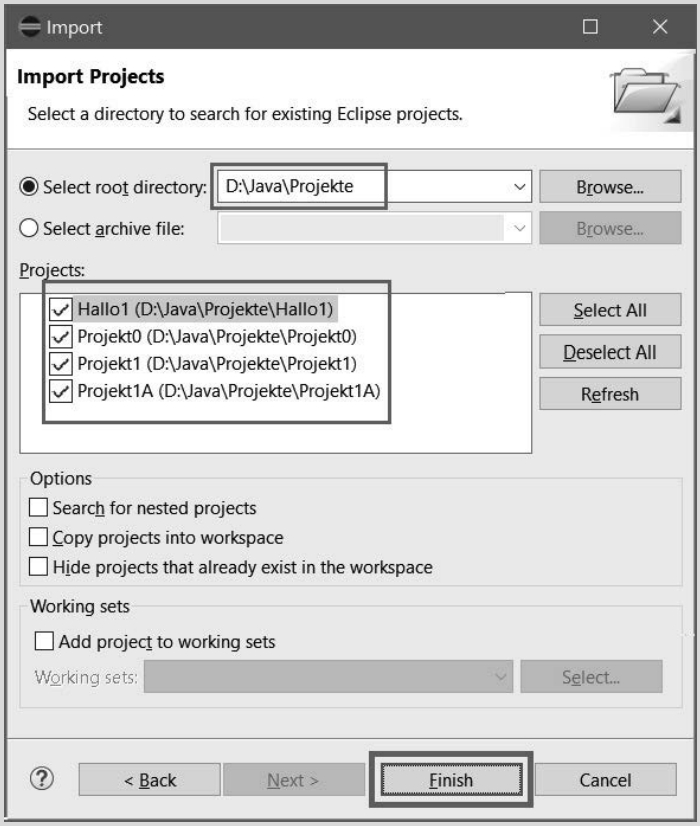


Du kannst den Namen auch direkt eintippen, z.B. D:\JAVA\PROJEKTE. Klicke dann auf OK.

Nun erscheint eine Liste, in der du alle Projekte markieren kannst, die du importieren möchtest.

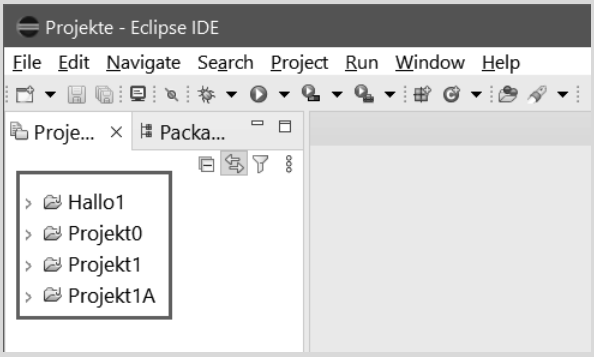
Wenn du willst, kannst du die Option COPY PROJECTS INTO WORKSPACE aktivieren. Das empfiehlt sich, wenn alle Projekte in einem anderen als dem Workspace-Ordner liegen.

Abschließend klickst du auf FINISH.



Einige Zeit später stehen dir die gewünschten Projekte zur Verfügung, wie du links im Projektfenster sehen kannst.

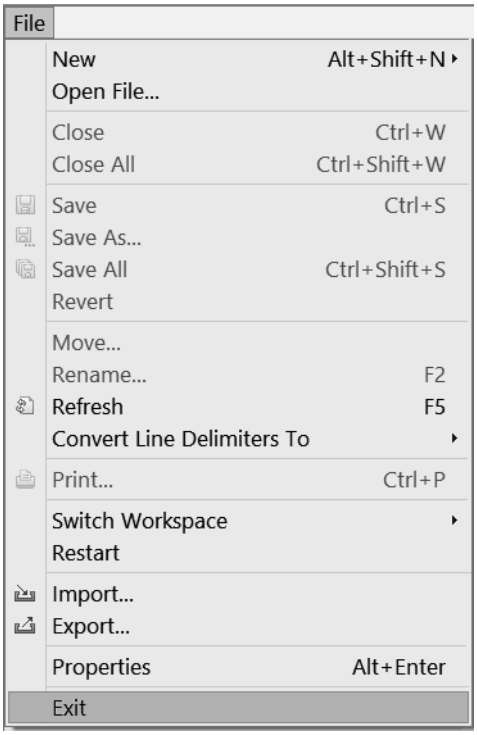
Dort kannst du dich bis zur Java-Quelltextdatei durchklicken und diese dann durch Doppelklick im Editorfenster öffnen.



Von da aus genügt die Anweisung RUN/RUN AS JAVA APPLICATION, um das Projekt zu starten.

Nun kannst du Eclipse beruhigt verlassen. Und das geht so:

≫ Klicke auf FILE und dann auf EXIT.



(Oder du klickst im Hauptfenster ganz oben rechts auf das kleine X.)

≫ Sollte dieses Dialogfeld auftauchen, klicke auf OK.

Wenn du ein Häkchen vor den Text ALWAYS EXIT WITHOUT PROMPT setzt, erscheint diese Meldung künftig nicht mehr.

ZUSAMMENFASSUNG

Mit deinem ersten Projekt gehörst du zwar noch lange nicht zur Gilde der Java-Programmierer, aber die Anfangsschritte hast du hinter dir. Mal sehen, was du von diesem Kapitel behalten hast. Da wären zuerst mal ein paar Operationen im Umgang mit Eclipse:

Eclipse starten	Doppelklicke auf das Symbol für Eclipse.
Neues Projekt erzeugen	Klicke auf FILE/NEW/JAVA PROJECT
Dateien speichern	Klicke auf FILE/SAVE (ALL)
Dateien neu speichern	Klicke auf FILE/SAVE AS
Programmprojekt starten	Klicke auf RUN/RUN
Projekt importieren	Klicke auf FILE/IMPORT
Hilfesystem aufrufen	Klicke auf HELP oder drücke F1
Eclipse beenden	Klicke auf FILE/EXIT

Und ein bisschen was vom Wortschatz von Java hast du auch schon kennengelernt:

`package`	Ein »Paket« aus Klassen
`class`	Eine Klasse umfasst unter anderem Eigenschaften und Methoden von Objekten.
`main`	Das Hauptprogramm, die »Hauptmethode«, in der dein Programmtext steht
`println()`	Einen Text anzeigen
`;` (Semikolon)	Damit schließt du eine Anweisung ab.
`{ }`	Damit wird alles umklammert, was zu einer Klasse oder zu `main()` gehört.
`( )`	Diese Klammern umfassen die Parameter einer Methode.
`//`	Dahinter steht ein Kommentar oder eine Erläuterung.

EIN PAAR FRAGEN ...

1. Warum spricht man in Eclipse nicht nur von Programm, sondern von Projekt?
2. Was ist der Unterschied zwischen `package` und `class`?
3. Wo findest du die Datei KLASSE1.JAVA?

... ABER NOCH KEINE AUFGABE