

Java mit Eclipse

4. Auflage

Hans-Georg
Schumann

FÜR **KIDS**



Auf der CD:

Eclipse als Entwicklungsumgebung,
der Visual Editor, das komplette
Java-Paket von Sun und
die Beispiele aus dem
Buch



mit
CD

bhv

1

Das erste Projekt

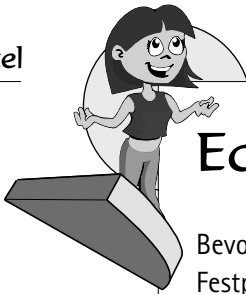


Du willst gleich loslegen? Dem Computer endlich mal etwas sagen, was er für dich tun kann? Na, dann schalte deinen PC an und lass erst mal Windows auftauchen. Von da aus geht es dann direkt zum ersten Programmprojekt in Java.

In diesem Kapitel lernst du

- ⊙ wie man Eclipse startet
- ⊙ wie man ein Programmprojekt erstellt und ausführt
- ⊙ Anweisungen für Ausgabe und Eingabe kennen
- ⊙ was Variablen und Konstanten sind
- ⊙ den Typ `String` kennen
- ⊙ wie man Eclipse beendet

1

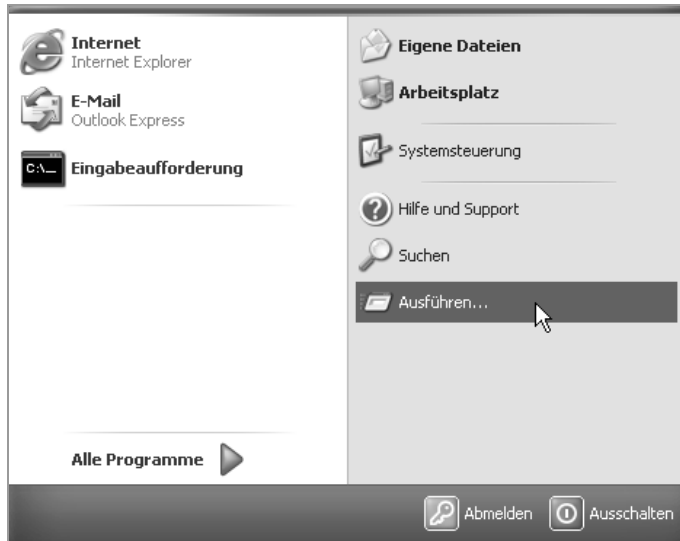


Eclipse starten

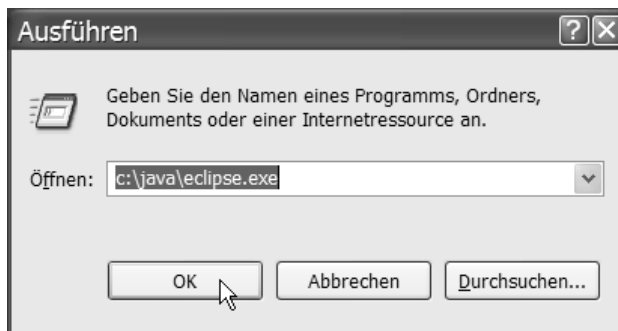
Bevor wir mit dem Programmieren anfangen können, muss *Eclipse* erst auf Festplatte kopiert und entpackt werden. Genaues erfährst du im *Anhang B*. Hier musst du dir von jemandem helfen lassen, wenn du dir das Einrichten nicht allein zutraust.

Eine Möglichkeit, Eclipse zu starten, ist diese:

➤ Klicke mit der Maus auf den Startknopf und dann auf AUSFÜHREN.



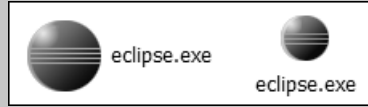
➤ Tippe im Dialogfeld ein: `c:\java\eclipse.exe` und klicke dann auf OK.



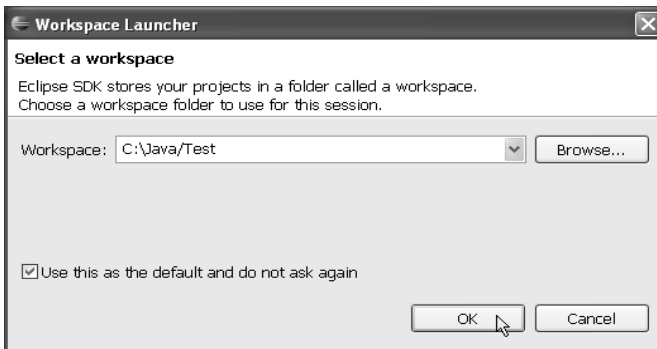
Wenn das nicht klappt, musst du über den Knopf DURCHSUCHEN nach der Datei ECLIPSE.EXE forschen.



Du kannst natürlich auch in den Ordner JAVA wechseln, in dem du das Symbol für ECLIPSE.EXE findest, und eine Verknüpfung auf dem Desktop anlegen. Von nun an kannst du Eclipse einfach per Doppelklick starten.

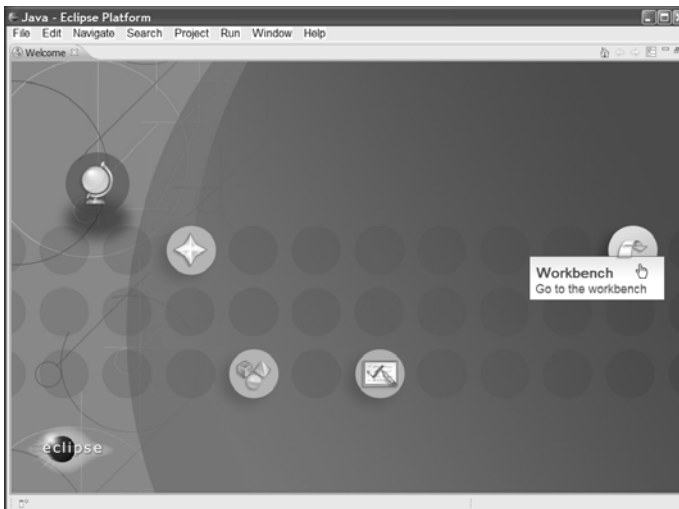


Je nach Computer kann es eine Weile dauern, bis Eclipse geladen ist. Zwischendurch fordert ein Dialogfeld einen Ordner an, in dem deine Projekte untergebracht werden – Workspace genannt.

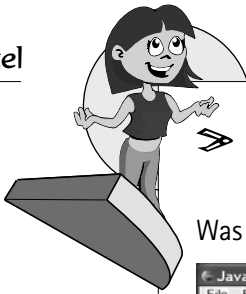


- Tippe z.B. `c:\java\test` (oder ein Verzeichnis deiner Wahl) ein und Sorge dafür, dass vor dem Eintrag `USE THIS AS DEFAULT AND DO NOT ASK AGAIN` ein Häkchen steht (sonst erscheint dieses Dialogfeld bei jedem deiner nächsten Starts von Eclipse immer wieder). Dann klicke auf OK.

Einige Zeit später landest du in einem Willkommenfenster.

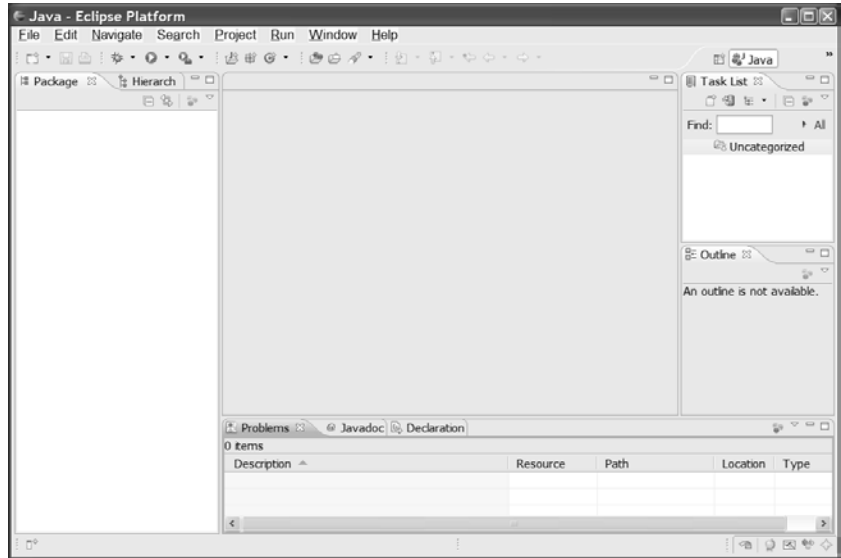


1



➤ Hier klickst du oben rechts in der Ecke auf das Symbol für GO TO THE WORKBENCH (frei übersetzt: »Geh endlich arbeiten«).

Was dich dann erwartet, könnte etwa so aussehen:



Die Startaufstellung von Eclipse

Für den ersten Augenblick ist das alles sicher ein bisschen sehr verwirrend. Nicht nur ein, sondern gleich ein paar Fenster 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 vier 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.
- ❖ Die Menüs EDIT und SOURCE helfen dir bei der Bearbeitung deines Programmtextes, aber auch bei anderen Programmelementen. Außerdem kannst du dort bestimmte Arbeitsschritte rückgängig machen oder wiederherstellen.
- ❖ Über das RUN-Menü sorgst du dafür, dass dein Projekt ausgeführt wird.
- ❖ Und das HELP-Menü bietet dir vielfältige Hilfsinformationen (auf Englisch) an.



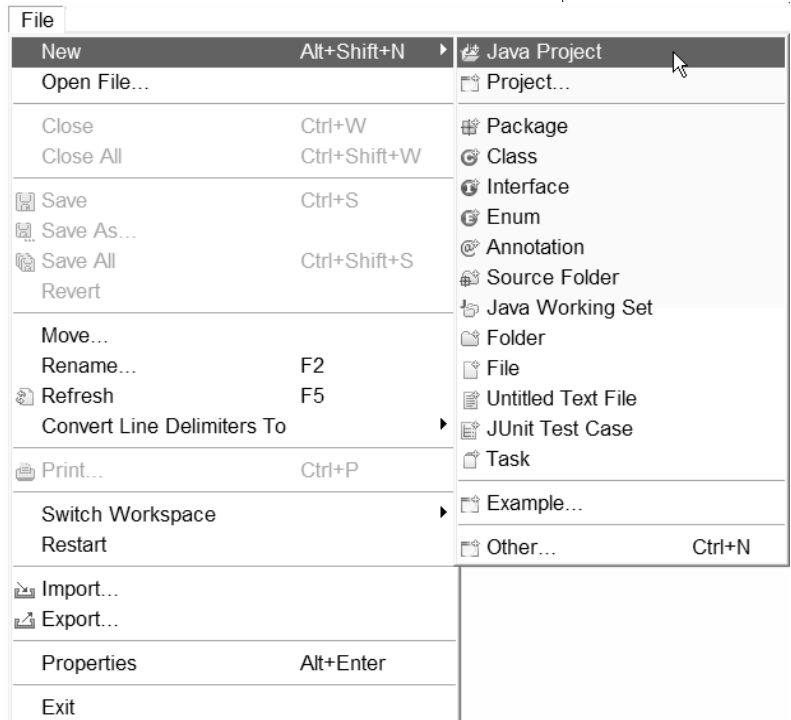
Einige wichtige Menüeinträge sind in einem so genannten *Popup*-Menü zusammengefasst. Das heißt so, weil es dort aufklappt, wo du gerade mit der *rechten* Maustaste hinklickst.

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

Willkommen in Java

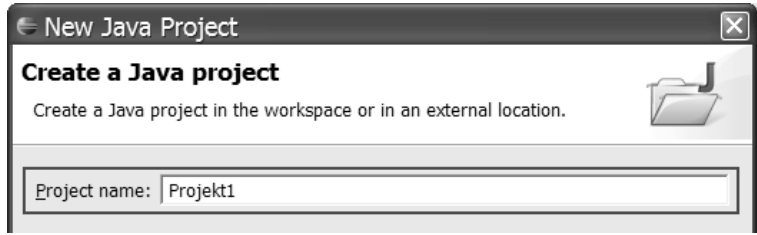
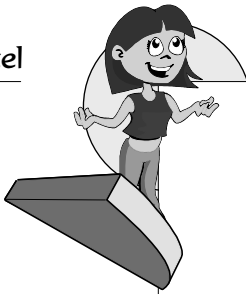
Eigentlich kann's jetzt schon losgehen. Den Umgang mit Menüs und Dialogfenstern kennst du bereits von Windows. Deshalb müssen wir uns damit nicht mehr aufhalten. Bauen wir uns jetzt ein kleines Projekt, für das erst einmal einige »Vorübungen« nötig sind.

➤ Klicke auf FILE und dann auf NEW und JAVA PROJECT.

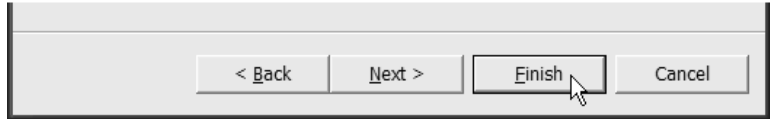


➤ Im nächsten Dialogfeld tippst du hinter PROJECT NAME **Projekt1** (oder einen Namen deiner Wahl) ein.

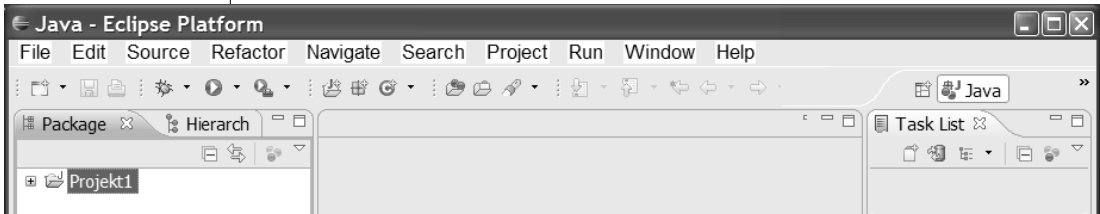
1



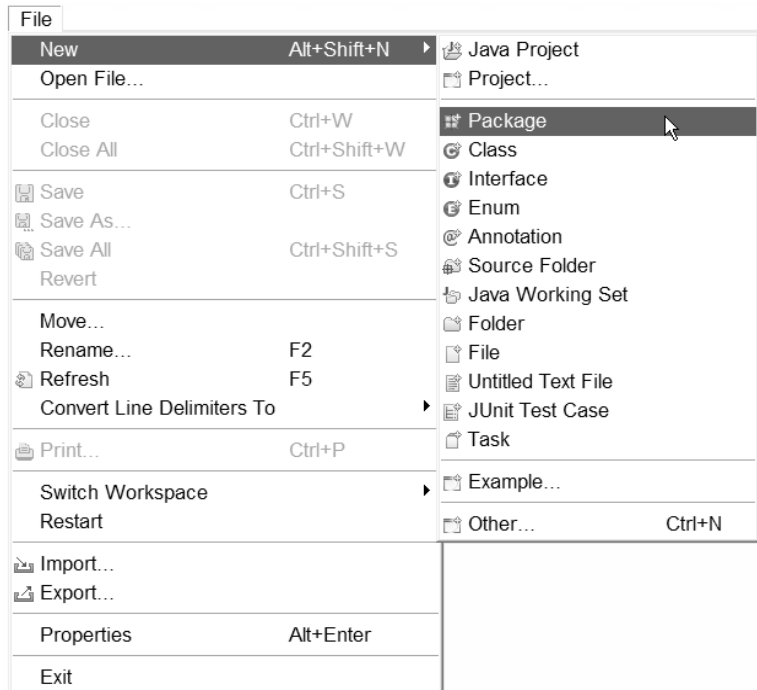
➤ Anschließend klickst du auf FINISH.



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

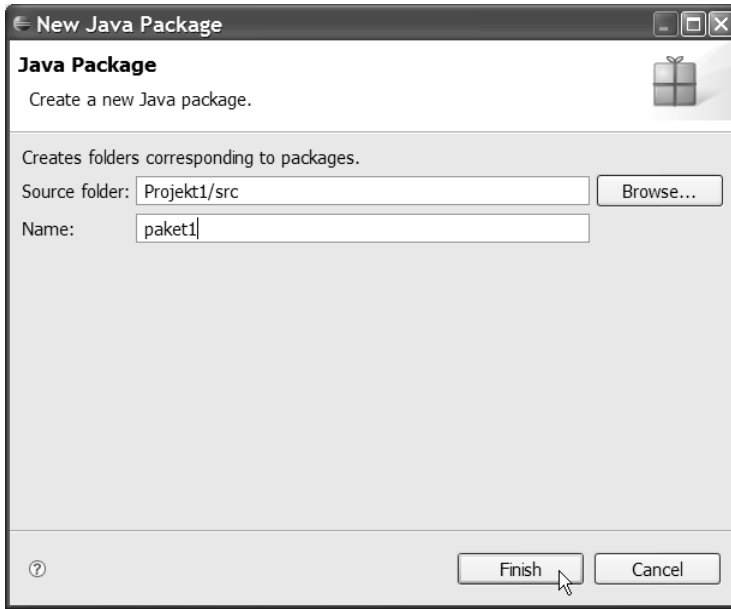


➤ Im nächsten Schritt klickst du auf FILE und dann auf NEW und PACKAGE.

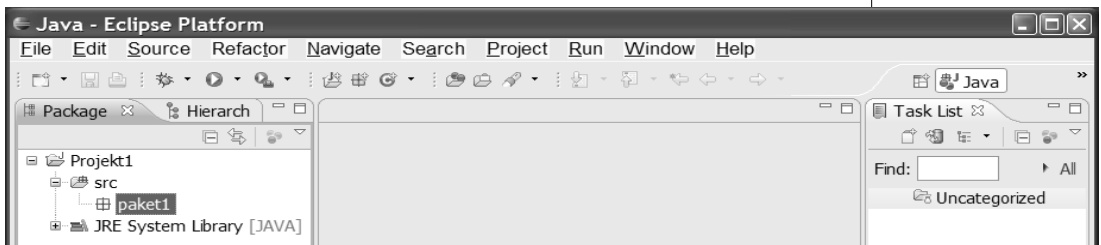




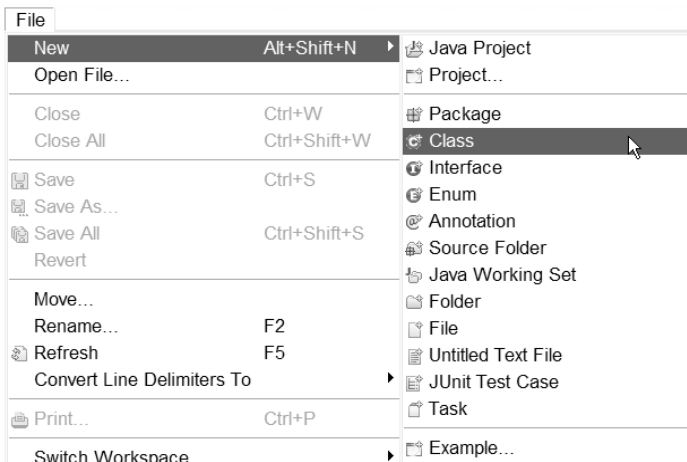
- Im Dialogfeld NEW JAVA PACKAGE tippst du hinter NAME **paket1** (oder wieder einen Namen deiner Wahl) ein. Dann klicke auf FINISH.



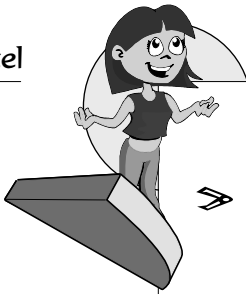
Und schon findet sich ein weiterer Eintrag im linken Fenster.



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



1



- Im Dialogfeld NEW JAVA CLASS sollten oben bereits PROJEKT1 und PAKET1 eingetragen sein. Hinter NAME tippst du Klasse1 ein.

New Java Class

Java Class

⚠ This package name is discouraged. By convention, package names usually start with a lowercase letter

Source folder:

Package:

Enclosing type:

Name:

- 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 as configured in the properties of the current project?

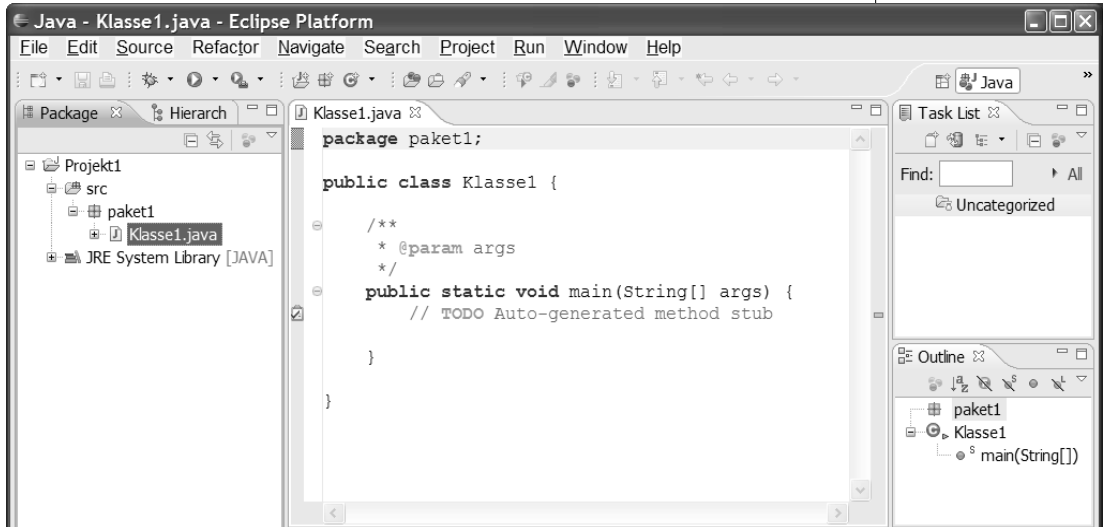
Generate comments



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



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



Nun weißt du auch, wo der Editor ist. In der Mitte steht unter dem Titel KLASSE1.JAVA dieser Text – auch *Quelltext* oder Programmtext genannt:

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

Was das im Einzelnen bedeutet, lässt sich erst nach und nach klären. (Einiges davon sind Kommentare, die wir im Folgenden nicht benötigen.)



1

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 eigentlich noch nichts – jedenfalls nichts für uns Sichtbares.

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

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

➤ Für mehr Übersicht solltest du aber zuerst mal einiges löschen bzw. umordnen und dann den neuen Text an der passenden Stelle ergänzen, damit das Ganze anschließend so aussieht:

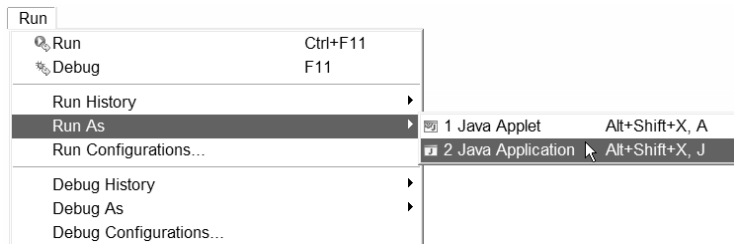
```
package paket1;
public class Klasse1
{
    public static void main (String[] args)
    {
        System.out.println ("Hallo, wer bist du?");
    }
}
```



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.

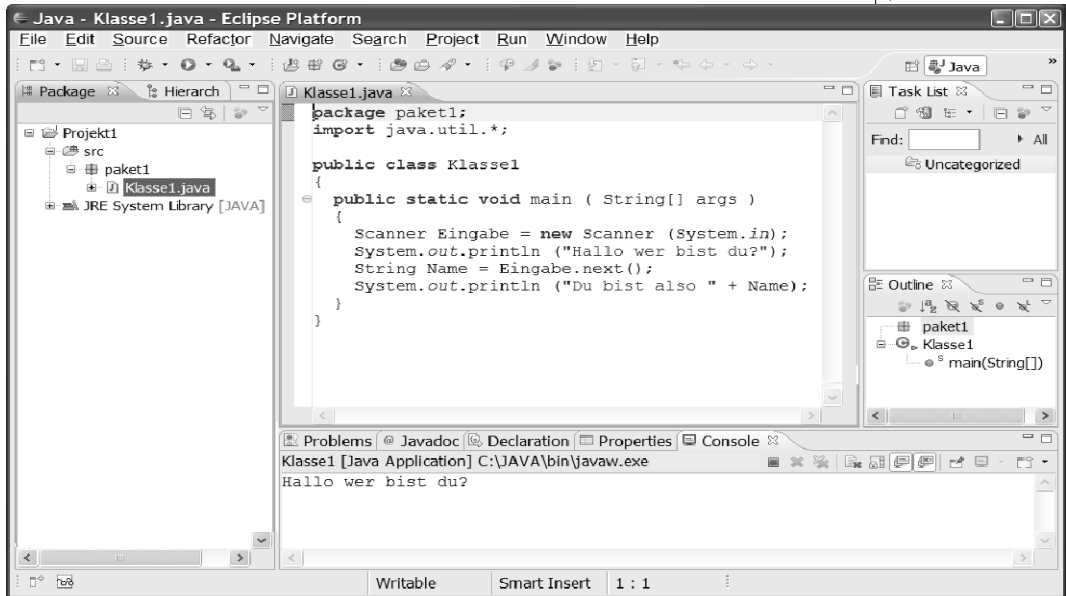
Und nun machen wir unseren ersten Probelauf:

➤ Klicke auf RUN und dann auf RUN AS und JAVA APPLICATION.





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



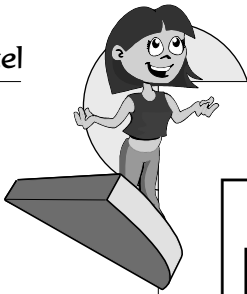
Ein bisschen dürftig, aber immerhin haben wir jetzt schon unser erstes Java-Programm erstellt. Betrachten wir unser Werk einmal genauer. Ein Projekt ist in seiner einfachsten Form so aufgebaut:

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

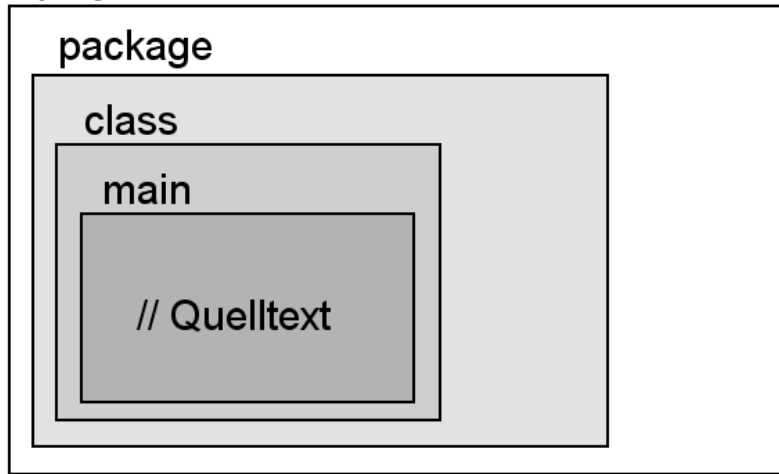
Das erinnert irgendwie an ein »Zwiebelsystem«: Die Außenhaut ist das Projekt mit eigenem Ordner. Darin liegt ein Paket (englisch: package). Offenbar muss ein Projekt nicht nur aus einem Paket bestehen. Im Paket-Ordner finden wir die Daten einer Klasse (englisch: class). Auch hier liegt die Vermutung nahe, dass es mehr als eine Klasse 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.

1



project



Sehr wichtig sind die geschweiften Klammern ({ }). Dazwischen stehen die Zeilen, die dem Programm erst richtig zum Leben verhelfen. Und die stammen größtenteils von uns. (Wenn wir es bisher auch nur zu einer Zeile gebracht haben, aber wir sind ja erst am Anfang.)

Sehr wichtig ist, dass es zu *jeder* öffnenden Klammer { auch eine schließende Klammer } geben muss! 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
    }
}
```

Oder gar so:

```
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.



Objekte, Klassen und Pakete



Jetzt willst du endlich wissen, was diese eine Anweisung bedeutet, mit deren Hilfe der PC offenbar zu einem freundlichen Gruß bereit ist:

```
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:

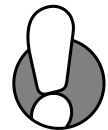
```
("Hallo, wer bist du?")
```

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

Diese Anweisung ist eine Methode. Und die gehört zu einem Objekt namens `out`, das wiederum Element einer Klasse ist, die auf den Namen `System` hört. Verbunden wird alles über den so genannten *Zugriffsoperator*, einen einfachen Punkt (`.`). Alles zusammen ergibt dann die Anweisung `System.out.println`. Abgeschlossen wird eine Anweisung immer mit einem Semikolon (`;`).

Es ist übrigens nicht egal, ob du für die Wörter große oder kleine Buchstaben benutzt. 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 `Ma in` oder `MAIN` statt `main` schreiben!

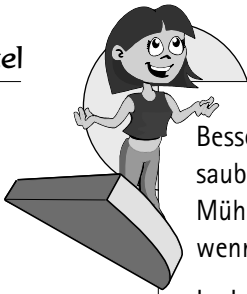


Objekte, Klassen und Pakete

Alles unklar? Bevor wir weiter programmieren, müssen wir uns wohl erst einmal mit der »Religion« von Java auseinander setzen.

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 schnell überfordert, wenn man in einer solchen Sprache programmiert. Vor allem aber wächst die Anfälligkeit eines Projektes für Fehler. Und die dann alle zu entdecken, kann zur Qual werden.

1



Besser ist da ein System, das den Programmierer von Anfang an zu einem sauberen klaren Programmierstil zwingt – allerdings kostet das zuerst mehr Mühe und auch einigen Frust. Dass es sich dennoch lohnt, sieht man dann, wenn man die ersten Kapitel »überwunden« hat.

In Java beschaffst du dir zuerst ein (leeres) Paket. Dort kommen dann alle die Sachen hinein, die du dir mit der Zeit zurechtprogrammierst. Damit die 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. So ein Objekt ist beispielsweise `out`, das mit seiner Methode `println` dafür sorgt, dass ein Text angezeigt wird.

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

Jedes Java-Programm stellt automatisch wichtige Systemanweisungen zur Verfügung. Die Klasse mit dem Namen `System` bietet drei wichtige »Kandidaten« an:

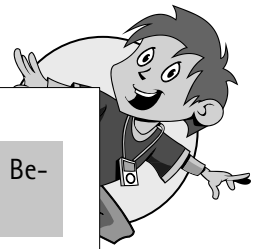
<code>System.out</code>	für die Standardausgabe
<code>System.err</code>	für die Fehlerausgabe
<code>System.in</code>	für die Standardeingabe

Du wirst im Laufe dieses Buches weitere Klassen und Objekte kennen lernen und 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 Projektordner 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.
- ❖ Klasseninformationen werden in Dateien mit der Kennung `CLASS` abgelegt.





Daneben verwaltet Eclipse noch einige zusätzliche Dateien, deren Bedeutung du hier nicht kennen musst.

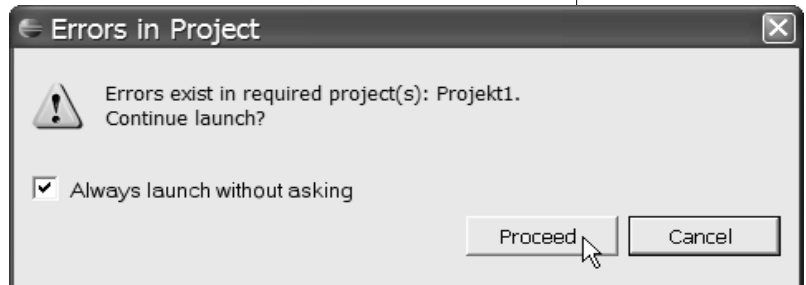
Ausgabe und Eingabe

Kehren wir zurück zu unserem Programmprojekt. Bisher sind wir so weit, dass ein Begrüßungstext angezeigt wird, der so formuliert ist, dass er eigentlich eine Antwort verlangt: Einen Namen, den man als Text eintippen kann.

Nahe liegend wäre eine Zeile wie diese:

```
System.in.readLine (Name);
```

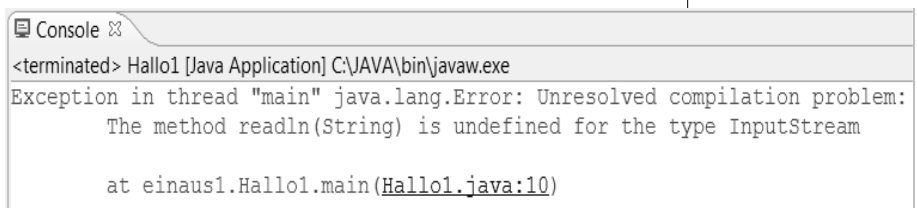
Das Objekt `System.in` ist schon richtig, bloß heißt die Methode `readln` dort `readLine`. Aber wenn du jetzt versuchst, das Programm mit dieser Zeile zum Laufen zu bringen, erntest du eine Fehlermeldung.



Wenn du dafür sorgst, dass vor ALWAYS LAUNCH WITHOUT ASKING ein Häkchen steht, erscheint diese Dialogbox nicht bei jedem Fehler.



➤ Klicke auf PROCEED.



Die Fehlerinformationen unten im Fenster weisen darauf hin, dass `readln` undefiniert ist. Wie es scheint, ist der Weg bis zur ersten Eingabe nicht so leicht wie der für die Ausgabe.

1



Wenn du ein geändertes Projekt erneut – über RUN AS JAVA APPLICATION – ausführen willst, kann dieses Dialogfeld auftauchen:



Mit Klick auf OK bestätigst du, dass der veränderte Quelltext vor dem Programmstart gespeichert wird.

Direkt speichern lässt sich deine Quelltextdatei übrigens mit FILE/SAVE bzw. FILE/SAVE AS (wenn's ein neuer Name sein soll).

Damit du nun nicht endlos probieren musst, versuchen wir es mal damit (→ PROJEKT1\PAKET1\KLASSE1.JAVA):

```
package paket1;
import java.util.*;

public class Klasse1
{
    public static void main ( String[] args)
    {
        Scanner Eingabe = new Scanner (System.in);
        System.out.println ("Hallo, wer bist du?");
        String Name = Eingabe.next();
        System.out.println ("Du bist also " + Name);
    }
}
```

Dialog mit »Schwung«



Ein bisschen umständlicher ist das mit der Eingabe schon, denn wir brauchen dazu erst ein Objekt vom Typ `Scanner` (das eben keine Bildvorlagen, sondern die Tastatur scannt). Die Methode `next` sorgt dafür, dass der eingegebene Text in einem `String` gespeichert wird, den wir sinnvollerweise auch `Name` nennen.

Unter einem `String` versteht man eine *Zeichenkette*. Und etwas Einge-
tipptes wie ein Name ist ja auch eine Kette von Zeichen.



Dialog mit »Schwung«

Es gibt zum `Scanner` noch eine interessante Alternative, die wir uns mal im folgenden Quelltext anschauen sollten (→ HALLO1\EINAUS1\HALLO1.JAVA):

```
package einaus1;
import javax.swing.*;

public class Hallo1
{
    public static void main (String[] args)
    {
        String Name = JOptionPane.showInputDialog
            ("Hallo, wer bist du?");
        System.out.println ("Du bist also " + Name);
    }
}
```

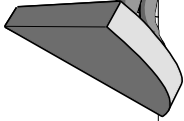
Ist angenehm kurz und führt im Prinzip zu dem gleichen Ergebnis wie das obige Beispiel – mit etwas Komfort als Zugabe.

➤ Starte das Projekt über `RUN/RUN AS/JAVA APPLICATION` oder mit `[F11]`.

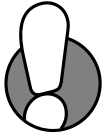
Diesmal erscheint der Hallo-Gruß nicht unten in einem Anzeigefenster von Eclipse, sondern es öffnet sich ein Dialogfeld, in das du deinen Namen eingeben kannst.



1



Nach einem Klick auf OK erscheint unten im Eclipse-Fenster der entsprechende Antworttext.



Bei dir läuft das Programm nicht? Stattdessen erscheint eine Fehlermeldung.

Zusätzlich ist die Stelle markiert, in der Eclipse den Fehler vermutet. Oft handelt es sich dabei um einen *Syntaxfehler*. Zum Beispiel könntest du `ShowInputDialog` statt `showInputDialog` geschrieben, also die Regel der Groß- und Kleinschreibung nicht beachtet haben.

Andere Fehler sind das Vergessen von Semikolon oder Komma, Klammern oder Anführungsstrichen.

Bessere die Stelle aus und starte dann das Projekt einfach noch einmal.

Picken wir uns jetzt die entscheidenden Zeilen heraus und betrachten wir sie näher:

```
String Name = JOptionPane.showInputDialog
    ("Hallo, wer bist du?");
```

Mit `String Name` wird eine *Variable* vereinbart, die wir `Name` nennen.

Den Begriff *Variable* liest du ja jetzt hier nicht zum ersten Mal. So genau erinnerst du dich aber nicht mehr, was das eigentlich ist? Aus dem Matheunterricht kennst du wahrscheinlich den Begriff *Platzhalter*. Platzhalter werden meist mit Buchstaben wie x oder y bezeichnet. Und weil diese Platzhalter in jeder Aufgabe einen anderen Wert annehmen können, also keinen von vornherein festgelegten Wert haben, nennt man so etwas Variablen (das Fremdwort »variabel« heißt auf Deutsch so viel wie »veränderlich«).

Im Gegensatz dazu gibt es natürlich in Java auch *Konstanten*. Die haben dann einen festgelegten Wert, der sich während des Programmlaufs nicht verändert. Und auch bei jedem neuen Programmstart behält eine Konstante ihren Wert.

Ein Beispiel ist der Text »Hallo, wer bist du?«. Aber auch Zahlen wie z. B. 0, 1, -1, 3.14 lassen sich als Konstanten einsetzen (wie du noch sehen wirst).

Der Name einer Variablen darf übrigens nicht mit einer Ziffer beginnen. (Probier einfach aus, was geht!)

Eclipse beenden



`JOptionPane` ist eine Klasse, die mehrere Dialogboxen zur Verfügung stellt. Eine davon haben wir eben durch den Aufruf der Methode `showInputDialog` verwendet. Diese Methode übernimmt in runden Klammern einen Text als Parameter, ähnlich wie wir es schon von `println` kennen.

So weit, so schön. Der kleine Haken bei der Sache ist, dass `JOptionPane` sich neben anderen hübschen und nützlichen Klassen in einem Extra-Paket befindet, das wir erst bei der Post abholen müssen:

```
import javax.swing.*;
```

Die Anweisung `import` ermöglicht uns den Zugriff auf ein ganzes Paket oder Teile davon. Hier ist es die `package javax.swing` mit allen zugehörigen Klassen – deshalb das Sternchen (*). (Bei der Klasse `Scanner` hieß die »Postsendung« `java.util`.)

Fast untergegangen ist die etwas seltsam anmutende Verwendung des Pluszeichens (+) bei `println`: Da werden doch anscheinend zwei Zeichenketten addiert – oder? Dass es funktioniert, hast du ja gesehen. Genannt wird das Verkettung von Strings.



Eclipse beenden

Dein allererstes Projekt ist sicher auf deiner Festplatte gelandet. Zeit also für eine kleine Pause.

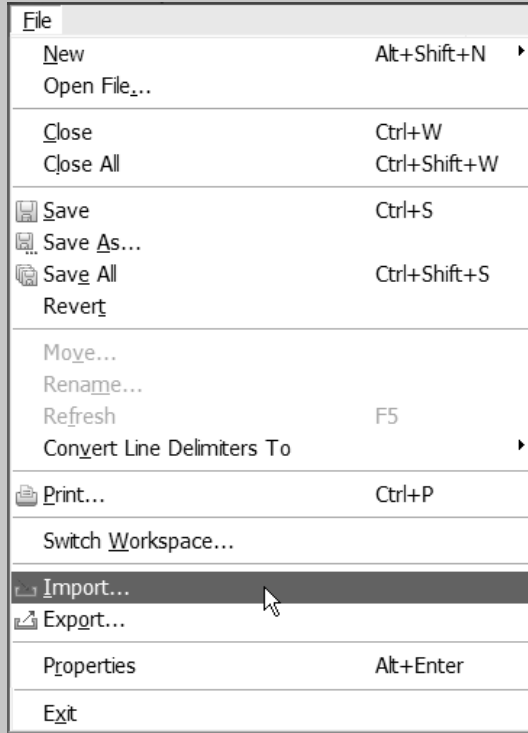
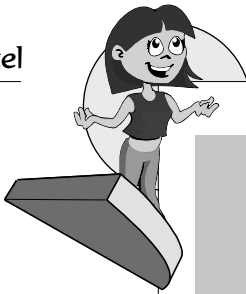
Möglicherweise hast du keine Lust, immerzu alles abzutippen, weißt aber nicht, wie du Eclipse dazu bringen kannst, die Projekte von der Buch-CD zu übernehmen. Ein Methode wäre, einfach den Quelltext der entsprechenden Dateien mit der Kennung `JAVA` zu markieren und nach dem Erzeugen eines neuen Projekts in das Editorfenster von Eclipse zu kopieren.

Aber es geht auch eleganter:

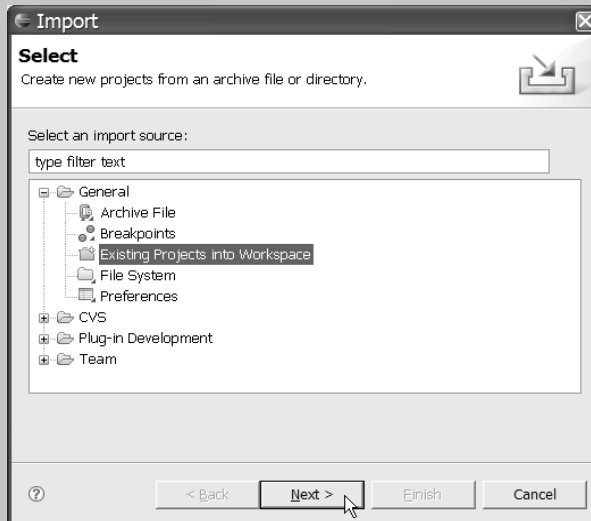
➤ Klicke auf `FILE` und `IMPORT`.



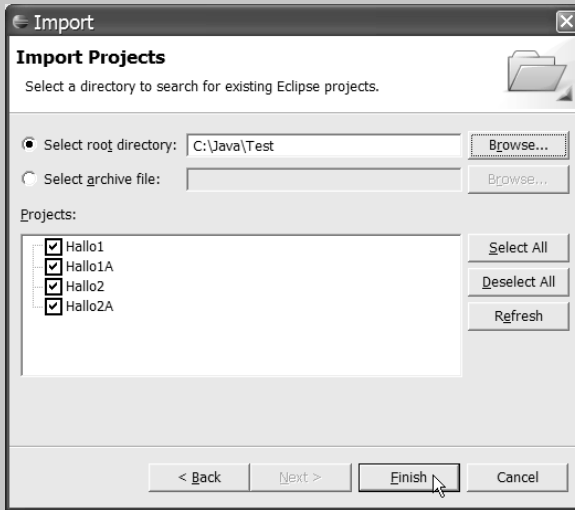
1



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



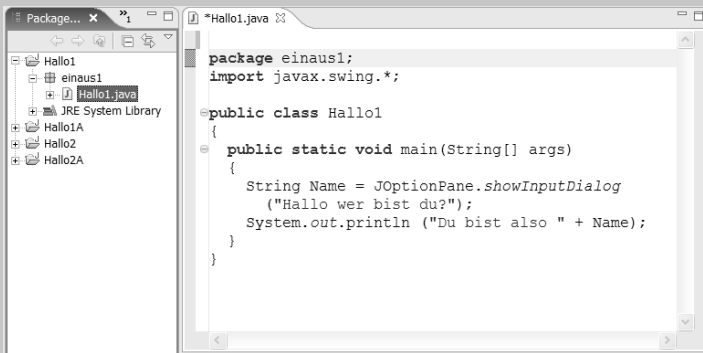
- Im nächsten Dialogfenster suchst du über BROWSE den Arbeitsordner oder tippst ihn direkt ein, z. B. C:\Java\Test.



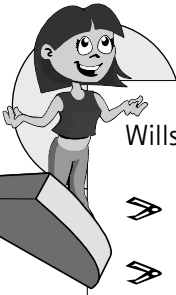
➤ Nun erscheint eine Liste, in der du alle Projekte markieren kannst, die du importieren möchtest. Abschließend klickst du auf FINISH.

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.



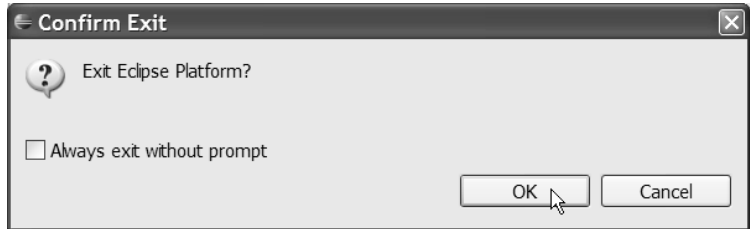
Von da aus genügt die Anweisung RUN/RUN AS JAVA APPLICATION, um das Projekt zu starten.

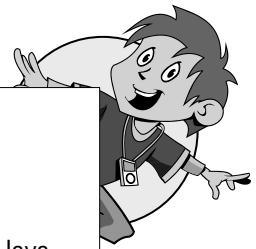


1

Willst du Eclipse verlassen, dann geht das 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.





Zusammenfassung

Mit deinem ersten Projekt gehörst du zwar noch 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	Klicke auf START/AUSFÜHREN und tippe den kompletten Pfad für ECLIPSE.EXE ein. Oder doppelklicke auf das Symbol für Eclipse.
Dateien speichern	Klicke auf FILE/SAVE
Dateien neu speichern	Klicke auf FILE/SAVE AS
Programmprojekt starten	Klicke auf RUN/RUN AS/JAVA APPLICATION
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 kennen gelernt:

<code>package</code>	Ein »Paket« aus Klassen
<code>import</code>	Ein Paket (oder Teile davon) in ein Projekt importieren
<code>class</code>	Eine Klasse umfasst unter anderem Eigenschaften und Methoden von Objekten.
<code>main</code>	Das Hauptprogramm, die »Hauptmethode«, in der dein Programmtext steht
<code>String</code>	Typ für Variablen: Zeichenkette
<code>println</code>	Einen Text anzeigen
<code>Scanner</code>	Eine Klasse zum Scannen der Tastatur
<code>next</code>	Einen Text von Tastatur aufnehmen
<code>showInputDialog</code>	Eine Dialogbox zur Texteingabe anzeigen
<code>JOptionPane</code>	Eine Klasse für Dialogboxen
<code>.</code> (Punkt)	Zugriffsoperator für die Verbindung von Objekt und Methoden
<code>;</code> (Semikolon)	Damit schließt du eine Anweisung ab.
<code>{ }</code>	Damit wird alles umklammert, was zu einer Klasse oder zu <code>main</code> gehört.
<code>()</code>	Diese Klammern umfassen die Parameter einer Methode.



Ein paar Fragen ...

1. Warum spricht man in Eclipse nicht nur von Programm, sondern von Projekt?
2. Was ist ein `String`?
3. Was ist der Unterschied zwischen `package` und `class`?

... aber noch keine Aufgabe