1 Einführung

Klassischerweise beginnt nahezu jedes Schriftstück, das die Einführung in eine Programmiersprache behandelt, mit einem Hello-World-Programm. Natürlich würde ich nicht im Traum daran denken, diese Tradition zu brechen. Es handelt sich hier gewissermaßen um ein Dogma der Vereinigung der Programmierbuchautoren, dem sich zu widersetzen mit dem Fegefeuer oder Ähnlichem bestraft wird. Aus diesem Grund folgt hier das einführende Programm aller einführenden Programme:

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

   // Gibt "Hello World" aus
   System.out.println("Hello World");
}
```

Das Programm ist sehr kurz. Dementsprechend bewirkt es auch herzlich wenig; um genau zu sein, beschränkt es sich darauf, den Text *Hello World* auf der Bildschirmkonsole auszugeben. `public static void main` usw. mit der anschließenden geschweiften öffnenden Klammer und der geschweiften schließenden Klammer am Ende stellen gewissermaßen das Grundgerüst des Programms dar. `main()` ist die Funktion, die jedes Javaprogramm beinhalten muss, da es mit dieser Funktion startet – immer. Die kryptischen Zeichenfolgen links und rechts von `main`, also `public static void` und die `String[] args`, beschreiben die Funktion näher. Weiteres dazu folgt später. `System.out.println` schließlich ist die Funktion, die die Bildschirmausgabe bewirkt. Als Parameter erhält sie den String, die Zeichenfolge, *Hello World*. Solche Strings werden in Java in Anführungszeichen geschrieben.

Hello World ist natürlich nur irgendein x-beliebiger String, der ausgegeben werden könnte. Genauso könnte man `System.out.println("Hallo Welt")` schreiben, um *Hallo Welt* auf dem Bildschirm auszugeben oder `System.out.println("Hallo Hans")` oder `System.out.`

println("Hallo Christian") oder System.out.println ("Das ist ein ganz blöder Text, der wahrscheinlich noch nie auf einer Bildschirmkonsole ausgeben wurde.") oder man gibt alles auf einmal aus, indem man all diese System.out.println() nacheinander in dem Programm angibt.

Hierbei anzumerken seien noch folgende beiden Sachverhalte: Zum einen eine Kleinigkeit: Mit System.out.println() wird am Ende des Strings immer ein Zeilenumbruch durchgeführt. Es gibt auch eine System.out.print()-Anweisung, mit der kein Zeilenumbruch am Ende des Strings erfolgt.
Zum anderen: Am Ende der Anweisung System.out.println ("Hello World") steht ein Semikolon („;“). Das ist bei Weitem die wichtigere Anmerkung, da in Java grundsätzlich jede einfache Anweisung mit einem Semikolon endet. Daran sollte man immer denken, da dies erfahrungsgemäß gerade von Programmierneulingen gerne vergessen wird.

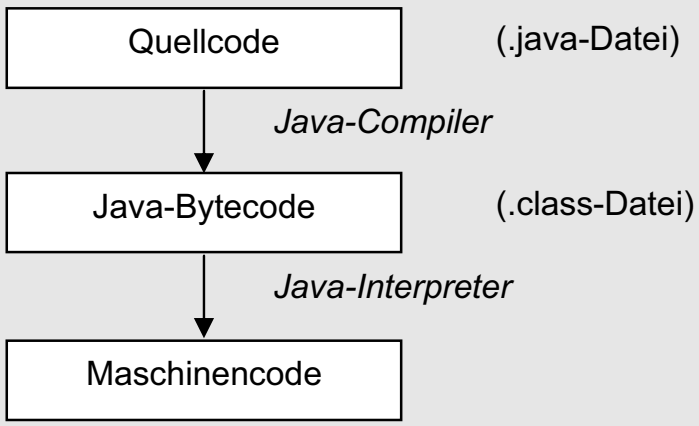
Wie in nahezu jeder Programmiersprache gibt es in Java auch *Kommentare*. Kommentare gehören nicht wirklich zum Programmcode. Wird das Programm übersetzt, werden Kommentare ignoriert. Vielmehr helfen sie dem Programmierer, seinen eigenen Code zu verstehen. In Java gibt es zwei Möglichkeiten, um zu kommentieren. Sie verwenden zwei Schrägstriche („//“) zur Einleitung eines Kommentars, wodurch alles, was hinter diesen beiden Schrägstrichen bis zum Zeilenende steht, ein Kommentar ist. Sie können Ihren Kommentar aber auch zwischen einem „/*“ und einem „*/“ einklammern und damit einen beliebigen Bereich als Kommentar verwenden, ganz gleich ob über mehrere Zeilen hinweg oder nur über einen Teil einer Zeile.

Auch weiter oben in unserem ersten Beispielprogramm kommt ein Kommentar vor, und zwar: ‚Gibt „Hello World“ aus’.

Das Übersetzen eines Java-Programms

Lassen Sie es mich einmal so formulieren: Im Inneren seines Herzens ist Ihr Computer ein ziemlich primitiver Zeitgenosse. Er versteht nur sehr einfache, präzise Anweisungen wie z.B. „Addiere die zwei Zahlen x und y“, „Springe in dem Programm an eine bestimmte Stelle und fahre dort mit der Abarbeitung fort“ oder „Speichere einen Wert z an einer bestimmten Stelle im Speicher“. Dies tut er aber äußerst schnell! Theoretisch können Sie, wenn Sie wollen, mit Ihrem Computer auf diesem Niveau kommunizieren. Sie können ihm Anweisungen geben, die er dann eins zu eins mit seinem *Befehlssatz* umsetzen kann. Aber glauben Sie mir: Das macht keinen Spaß!

Empfehlenswert ist es deshalb, eine Programmiersprache wie Java zu verwenden, welche es erlaubt, Ihrem Computer – hauptsächlich dem sog. *Prozessor* – Anweisungen auf einem abstrakteren Niveau zu erteilen. Das funktioniert so: Sie schreiben abstrakte Anweisungen und erstellen somit den sog. *Quellcode* Ihres Programms. Dieses Programm wird dann üblicherweise durch einen sog. *Compiler* in *Maschinensprache* übersetzt, also in jene Sprache, die Ihr Prozessor versteht und deshalb Befehl für Befehl abarbeiten kann. Der Compiler wandelt dabei jede einzelne abstrakte Anweisung in eine Vielzahl konkreter maschinenlesbarer Anweisungen um. Tatsächlich ist der Übersetzungsvorgang eines Java-Programms noch ein wenig komplizierter, wie die folgende Abbildung zeigt:



Sie erstellen den Quellcode, welcher in einer Datei mit der Endung „.java“ gespeichert wird. Mittels Java-Compiler wird daraus dann zunächst ein Zwischencode, ein sog. *Java-Bytecode*, erzeugt. Klassischerweise wird beim Aufruf des Programms der Bytecode sukzessive durch den Java-Interpreter in Maschinencode umgewandelt und zur Ausführung gebracht. Warum erzeugt der Java-Compiler nicht gleich Maschinencode? Nun ja, durch diesen Zwischenschritt wird für Plattformunabhängigkeit gesorgt. Der zu erzeugende Maschinencode ist abhängig vom Rechner, auf welchem er ausgeführt werden soll, und auch vom Betriebssystem. Deshalb wird zunächst der Quellcode in einen Java-Bytecode übersetzt, der unabhängig von diesen Faktoren ist. In Kombination mit einem plattformspezifischen Java-Interpreter kann dann der Bytecode auf beliebigen Systemen ausgeführt werden.

Getting started

Sie wollen das Hello-World-Programm zum Laufen bringen, wissen aber nicht so recht, wie? Mal sehen, ob ich Ihnen dabei ein wenig helfen kann.

Grundsätzlich können Sie Ihren Java-Quellcode in einem beliebigen Texteditor schreiben, die entsprechende Datei mit der Endung „.java“ speichern, diese mit Hilfe des Java-Compilers übersetzen und die daraus resultierende „.class“-Datei unter Verwendung des Java-Interpreters zur Ausführung bringen. Compiler, Interpreter und das notwendige Equipment erhalten Sie kostenfrei auf der Homepage der Firma *Sun*, der Herstellerfirma von Java, unter:

http://java.sun.com/javase/downloads

Sie müssten dazu auf dieser Seite das *Java SE Development Kit* (JDK) herunterladen.

Statt einen gewöhnlichen Texteditor für die Programmierung zu verwenden, würde ich Ihnen empfehlen, eine ordentliche Java-Entwicklungsumgebung einzusetzen. Derartige Programme unterstützen Sie bestmöglich bei der Java-Programmierung. Sie integrieren z.B. den Compiler und den Interpreter, stellen einen Debugger zur Verfügung oder unterstützen Sie direkt bei der Eingabe des Quellcodes, indem sie Schlüsselwörter hervorheben oder bekannte Namen automatisch vervollständigen. Eine meiner Meinung nach sehr gute Entwicklungsumgebung für Java bietet *Eclipse*. Sie können Eclipse kostenfrei unter

http://www.eclipse.org/downloads

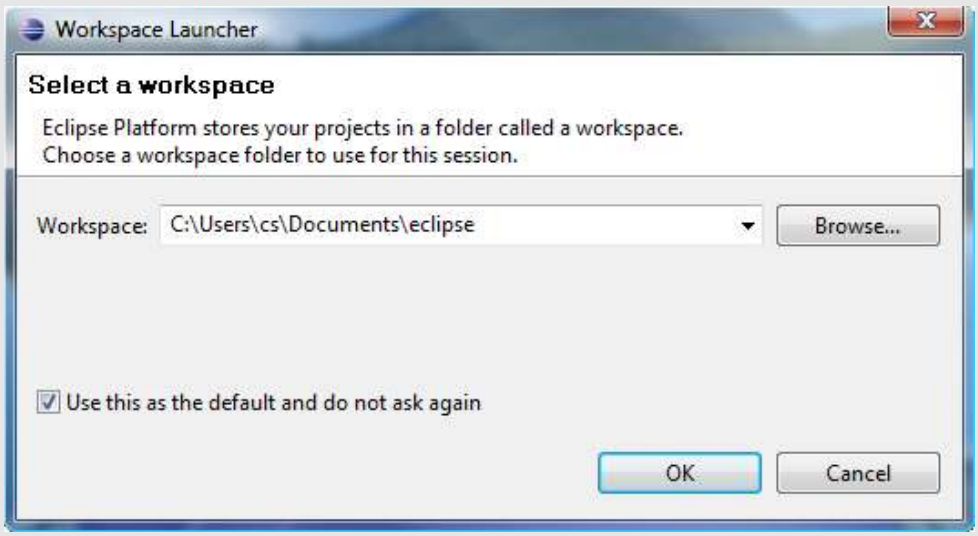
herunterladen. Wählen Sie auf der entsprechenden Seite das Produkt *Eclipse IDE for Java Developers*. Die Installation von Eclipse besteht lediglich im Entpacken der ZIP-Datei. Entpacken Sie Eclipse am besten in das Verzeichnis, in welches Sie auch Ihre übrigen Programme installieren (z.B. unter Windows in C:\Programme). Sie können Eclipse im Anschluss daran starten, indem Sie in dem Verzeichnis *eclipse* die gleichnamige Anwendung ausführen.

Bevor Eclipse gestartet werden kann, muss auf Ihrem Rechner die Java-Laufzeitumgebung (Java Runtime Environment, JRE) installiert sein. Die JRE benötigen Sie, um Java-Programme ausführen zu können. Sie können die JRE unter derselben Adresse herunterladen, unter der auch das JDK erreichbar ist. Möglicherweise ist die JRE auf Ihrem Rechner aber ohnehin bereits installiert (Haben Sie z.B. schon einmal ein Java-Applet im Internet gestartet?). Sie können dies feststellen, indem Sie in der Systemsteuerung in dem Verzeichnis Ihrer installierten Software nachsehen (falls Sie Windows nutzen). Oder Sie versuchen einfach Eclipse zu starten. Falls es einwandfrei hochfährt, ist die JRE installiert.

Das JDK ist eine echte Obermenge der JRE; es beinhaltet die JRE. Das JDK ist eigentlich für Java-Entwickler – also für Sie – gedacht (*Java Development Kit*). Wenn Sie Eclipse benutzen, ist aber dennoch die JRE ausreichend, da Eclipse selbst die notwendigen Entwicklungswerkzeuge bereithält. In der JRE befindet sich beispielsweise kein Java-Compiler, dafür ist ein solcher aber Bestandteil von Eclipse.

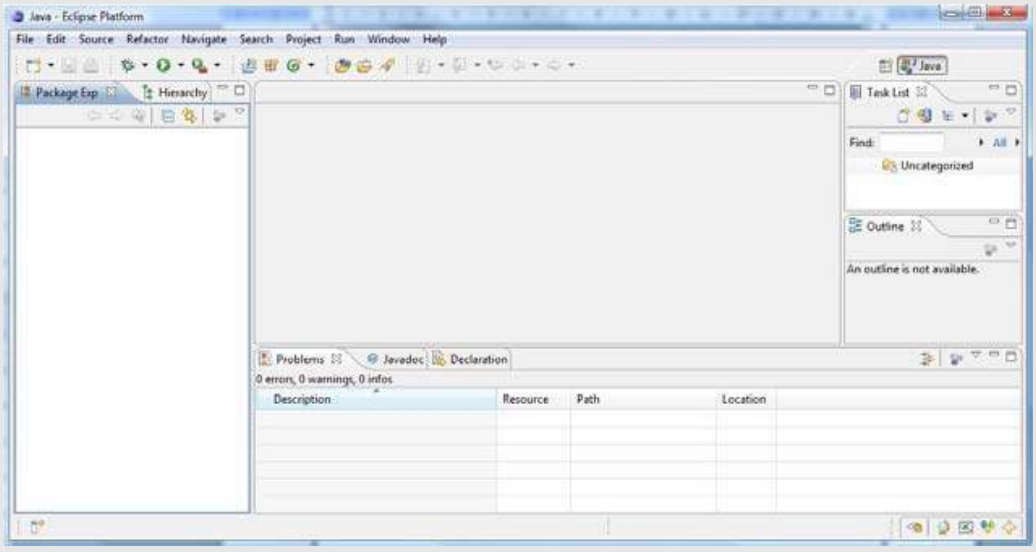
Die JRE (und demzufolge auch das JDK) beinhaltet eine große Anzahl an vorgefertigtem Java-Code, den wir in unsere Programme integrieren können. Beispielsweise wird die oben verwendete Funktion `System.out.println()` von der JRE bereitgestellt. Aus diesem Grund müssen wir nicht selbst die Ausgabe auf den Bildschirm programmieren, sondern wir verwenden einfach diese Funktion. Dieser vorgefertigte Java-Code ist – wie jeder Java-Code – in sog. *Klassen* organisiert (wir werden auf das Thema *Klassen* später noch sehr ausführlich eingehen). Wir bezeichnen diese Zusammenstellung von Standardklassen, welche durch die JRE bereitgestellt werden, als *Java-Klassenbibliothek* oder als *Standardklassenbibliothek*.

Sofern Sie eine JRE installiert haben, sollte Eclipse beim ersten Start folgendes Fenster anzeigen:

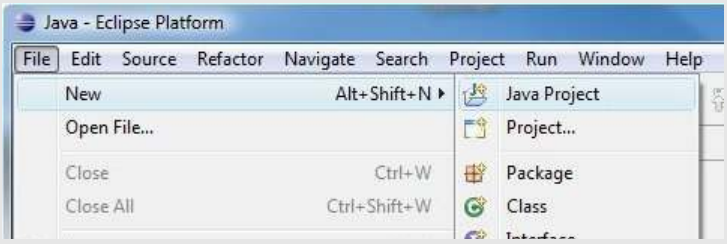


Eclipse fordert Sie auf, einen *Workspace* auszuwählen. Sie müssen also angeben, in welchem Verzeichnis Ihre zukünftigen Java-Projekte gespeichert werden sollen. Suchen Sie sich dafür am besten ein Verzeichnis, in dem Sie auch sonst Ihre eigenen Dokumente aufbewahren.

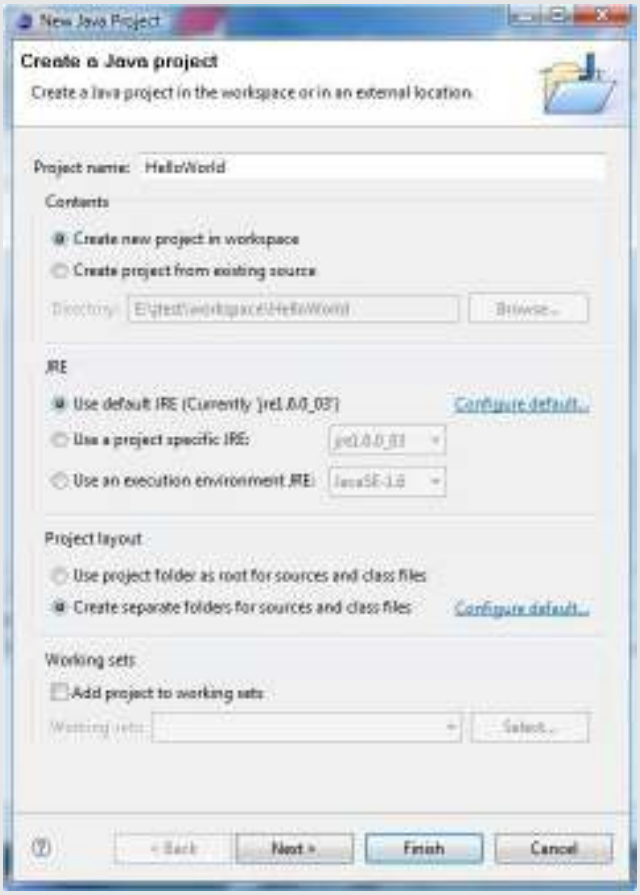
Als nächstes wird Ihnen beim ersten Start von Eclipse die obige Willkommensmaske angezeigt. Wechseln Sie zur Workbench, und Sie sehen Ihre übliche Eclipse-Arbeitsoberfläche wie sie die folgende Abbildung zeigt:



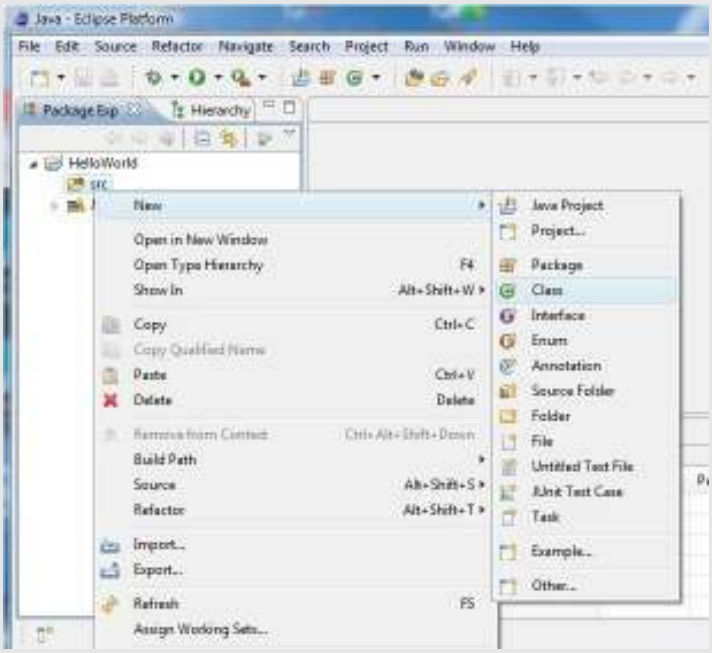
Um nun ein neues Projekt anzulegen, klicken Sie auf *File/New/Java Project*:



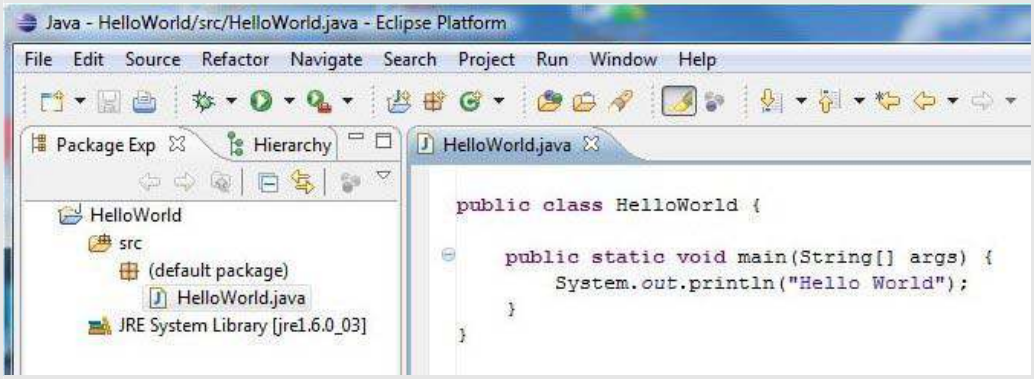
Geben Sie in der folgenden Maske den Projektnamen an, hier *HelloWorld*, und klicken Sie auf *Finish*:



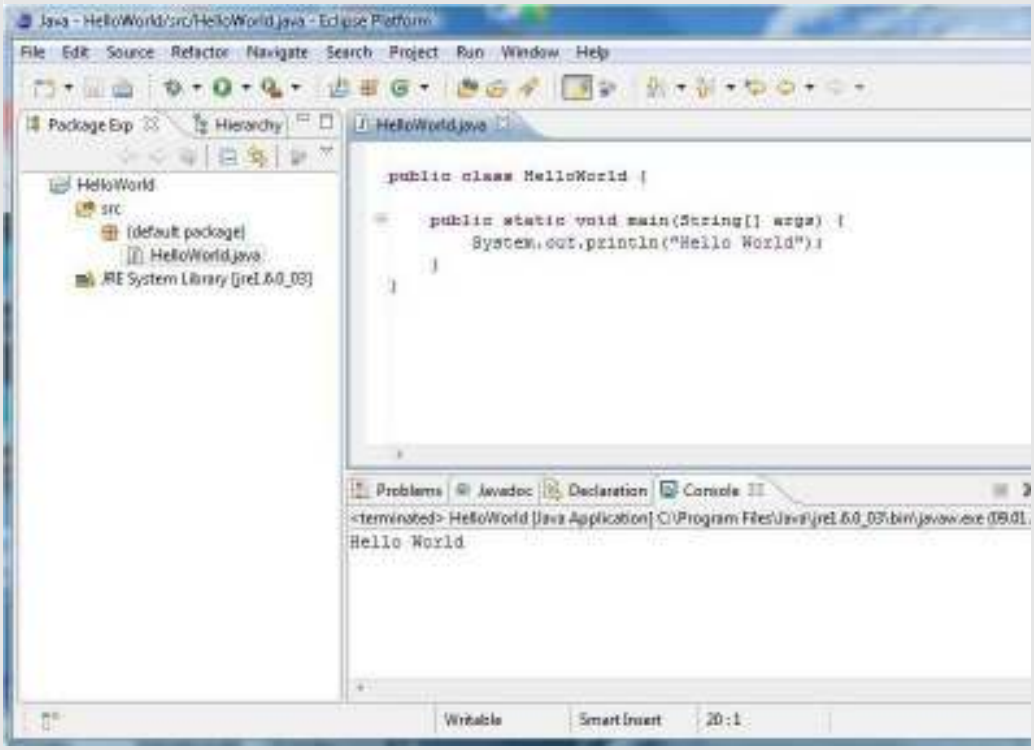
Im Anschluss daran legen Sie in dem Projekt *HelloWorld* eine Klasse mit der Bezeichnung `HelloWorld` an:



Dann kann’s ja los gehen! Geben Sie das Hello-World-Programm in Eclipse ein:



Sie können nun das fertige Programm übersetzen, indem Sie den Menüpunkt *Run/Run* wählen oder das nachfolgend markierte Icon in der Symbolleiste anklicken:



Das war's! Ihr Hello-World-Programm wurde ausgeführt. Sie sehen die Ausgabe *Hello World* in der Konsole im unteren Drittel des Fensters.

Beachten Sie bitte, dass sich das Hello-World-Programm von dem eingangs beschriebenen Beispiel unterscheidet. Damit daraus tatsächlich ein echtes Java-Programm wird, müssen Sie die `main()`-Funktion in eine Klasse packen, sie also in folgendes Gerüst stecken:

```
public class Klassenname {
    …
}
```

Erklärungen zu diesem Gerüst folgen in Kapitel *3. Objektorientierung* (S. 65). **Bitte berücksichtigen Sie dies auch für die noch folgenden Beispielprogramme: Sie müssen die angegebenen Funktionen stets innerhalb einer Klasse positionieren, damit sie als vollständige Javaprogramme übersetzt und ausgeführt werden können.**

Einen letzten Aspekt zu Eclipse möchte ich an dieser Stelle noch ansprechen: Das Thema *Dokumentation der Java-Klassenbibliothek*. Sie erhalten zu jeder Komponente der Java-Klassenbibliothek eine Dokumentation, wenn Sie mit dem Cursor einen entsprechenden Bezeichner fokussieren (setzen Sie den Cursor beispielsweise in Ihrem Programm auf `println()`) und dann die Tastenkombination *Strg+F1* eintippen. Standardmäßig wird Eclipse versuchen, die gewünschte Dokumentation aus dem Internet herunterzuladen. Sollten Sie keine permanente Internetverbindung besitzen, empfehle ich Ihnen, die komplette Dokumentation von der Sun-Homepage herunterzuladen und anschließend Ihre lokale Dokumentation in Eclipse zu integrieren. Sie erhalten die *Java SE Documentation* auf derselben Seite, auf der sich auch das JDK und die JRE befinden.