

Hands-On Programming mit Editing Java Easily

Erste Schritte

Roland Küstermann

September 3, 2011

Contents

1	Hands-On Programming mit Editing Java Easily	1
1.1	Allgemeine Vorgehensweise	2
1.1.1	Umgang mit Dateien	2
1.1.2	Compilieren eines Java-Programms	2
1.1.3	Ausführen eines Java-Programms	3
1.1.4	Arbeiten mit der Kommandozeile?	3
1.2	Arbeiten mit Editing Java Easily	3
1.2.1	Umgang mit Dateien	5
1.2.2	Compilieren eines Java-Programms	6
1.2.3	Umgang mit Compilerfehlern	8
1.2.4	Ausführen eines Java-Programms	9
1.3	Exkurs: Arbeiten mit Jeliot	11
1.3.1	Installation	12
1.3.2	Ausführen von Java-Programmen mit Jeliot	13
1.4	Zusammenfassung	15

1 Hands-On Programming mit Editing Java Easily

In diesem Kapitel werden wir zeigen, wie wir einen Algorithmus in ein lauffähiges Programm umsetzen können. Um einfach Editieren, Compilieren und Ausführen zu können, setzen wir eine Entwicklungsumgebung ¹ ein, die speziell für Programmieranfänger entwickelt wurde und seit 2004 in der Programmierausbildung eingesetzt wird. Wir gehen davon aus, dass das Java Entwickler Kit (Java Development Kit ²) bereits auf Ihrem Rechner installiert ist und gehen deshalb direkt in medias res.

Von der Bearbeitung bis zur Ausführung des Java-Programms werden wir uns des Beispiels

```
1 import static Prog1Tools.IOTools.*;
2 /**
3  * Programm zur Berechnung des Umfangs u
4  * und des Flaecheninhalts f eines Kreises
5  * mit gegebenem Radius r.
6  */
7 public class Kreis
8 {
9     public static void main (String [] args)
10    {
11        double r, u, f;
12        r = readDouble("Radius : ");
13        u = 2 * r * 3.1415926;
14        f = r * r * 3.1415926;
15        System.out.println("Umfang : " + u);
16        System.out.println("Flaeche: " + f);
17    }
18 }
```

bedienen. Bevor wir uns der Entwicklungsumgebung zuwenden, erklären wir zunächst die allgemeine Vorgehensweise.

¹Auf dem Markt sind viele Entwicklungsumgebungen verfügbar. Für jede eine Anleitung zu schreiben ist uns leider nicht möglich. Wir verwenden hier die Entwicklungsumgebung Editing Java Easily, die eigens für die Ausbildung konzipiert wurde. Alle Aufgaben können aber auch mit anderen Entwicklungsumgebungen bearbeitet werden.

²Das Java Development Kit (JDK) enthält die benötigte Java Ausführungsumgebung sowie unter anderem den Compiler und Interpreter. Alle Informationen zur Installation des JDKs finden Sie unter <http://www.eje-home.de>.

1.1 Allgemeine Vorgehensweise

Damit wir ein Java-Programm ausführen können, müssen zwei Bedingungen im Quelltext erfüllt sein:

1. Der Programmbezeichner wird, entsprechend der Zeile 7 in unserem Beispiel, mit den Schlüsselwörtern `public class` eingeleitet.
2. Das Programm besitzt eine Hauptmethode `main`, entsprechend der Zeile 9 in unserem Beispiel.

Bitte achten Sie insbesondere auf die Groß- und Kleinschreibung bzw. auf die genaue Schreibweise.

1.1.1 Umgang mit Dateien

Damit der Computer unseren Algorithmus, den wir in Java formulieren, verarbeiten kann, müssen wir diesen in einer Datei abspeichern. Bis auf weiteres legen wir fest, dass jedes Programm auch in einer eigenen Datei abgespeichert wird. Dabei legt der Programmbezeichner den Namen der Datei fest.

Beispiel: In unserem Beispiel wird unserer Programmbezeichner in Zeile 7 festgelegt. Hier also `Kreis`.

Ausgehend vom Programmbezeichner speichern wir ein Java-Programm also in einer gleichnamigen Datei. Die Dateiendung lautet `java`.

Beispiel: In unserem Beispiel wird unser Quelltext in einer Datei mit dem Dateinamen `Kreis.java` abgespeichert.

1.1.2 Compilieren eines Java-Programms

Damit der Computer das Java-Programm ausführen kann, muss es zunächst compiliert werden. Vereinfacht formuliert ist der Compiler eine eigenständige Anwendung, die zunächst für die Prüfung der Syntax und Semantik des Java-Programms zuständig ist. Ist unser Programm syntaktisch und semantisch korrekt, so übersetzt der Compiler dann das Programm in einen plattformunabhängigen Zwischencode, den sogenannten Bytecode³.

³Dieser Zwischenschritt ist Java spezifisch und sorgt für Plattformunabhängigkeit. Compiler anderer Programmiersprachen übersetzen den Quelltext direkt in einen plattformabhängigen maschinennahen Code.

Beispiel: Wenn das Programm `Kreis`, gespeichert in der Datei `Kreis.java`, erfolgreich kompiliert wurde, hat der Compiler eine Datei mit dem Dateinamen `Kreis.class` erzeugt.

1.1.3 Ausführen eines Java-Programms

Mit Hilfe des Interpreters – wiederum eine eigenständige Anwendung – kann das kompilierte Java-Programm – der Bytecode – ausgeführt werden. Dazu übersetzt der Interpreter den Bytecode Schritt für Schritt in eine für den Computer verständliche Sprache und führt diese aus. Er beginnt dabei in der Regel mit der Ausführung der Hauptmethode `main` und arbeitet dann Anweisung für Anweisung das Programm ab.

1.1.4 Arbeiten mit der Kommandozeile?

Für den geschilderten Prozess sind im Prinzip ein einfacher Texteditor sowie die Kommandozeilentools Compiler und Interpreter aus dem Java Development Kit ausreichend. Allerdings sind eine aufwendige Konfiguration und Einrichtung sowie ein heutzutage nicht mehr vorauszusetzendes Wissen im Umgang mit der Kommandozeile notwendig. Um nicht schon hier unüberwindbare Hürden aufzustellen, verwenden wir eine integrierte Entwicklungsumgebung, die es uns ermöglicht, in wenigen Minuten loszulegen. Daher beschreiben wir den Entwicklungsprozess nun anhand der Entwicklungsumgebung Editing Java Easily.

1.2 Arbeiten mit Editing Java Easily

In diesem Kapitel wird nur das eingeführt, was Sie unbedingt zur Bearbeitung der Aufgaben wissen müssen. Für eine ausführliche Dokumentation, auch zur Installation, verweisen wir schon jetzt auf die Webseite der Anwendung⁴.

Die integrierte Entwicklungsumgebung Editing Java Easily (EJE) übernimmt unter anderem die folgenden Aufgaben

- Syntaxhervorhebung für den Quelltext
- Einfaches Compilieren von Java-Programmen

⁴Informationen zur Installation der Anwendung finden Sie auf der Webseite des Programms (<http://www.eje-home.de>).

1 Hands-On Programming mit Editing Java Easily

- Einfaches Ausführen von Java-Programmen
- Unterstützung bei der Behebung von Compilerfehlern
- Unterstützung von Quelltext-Vorlagen

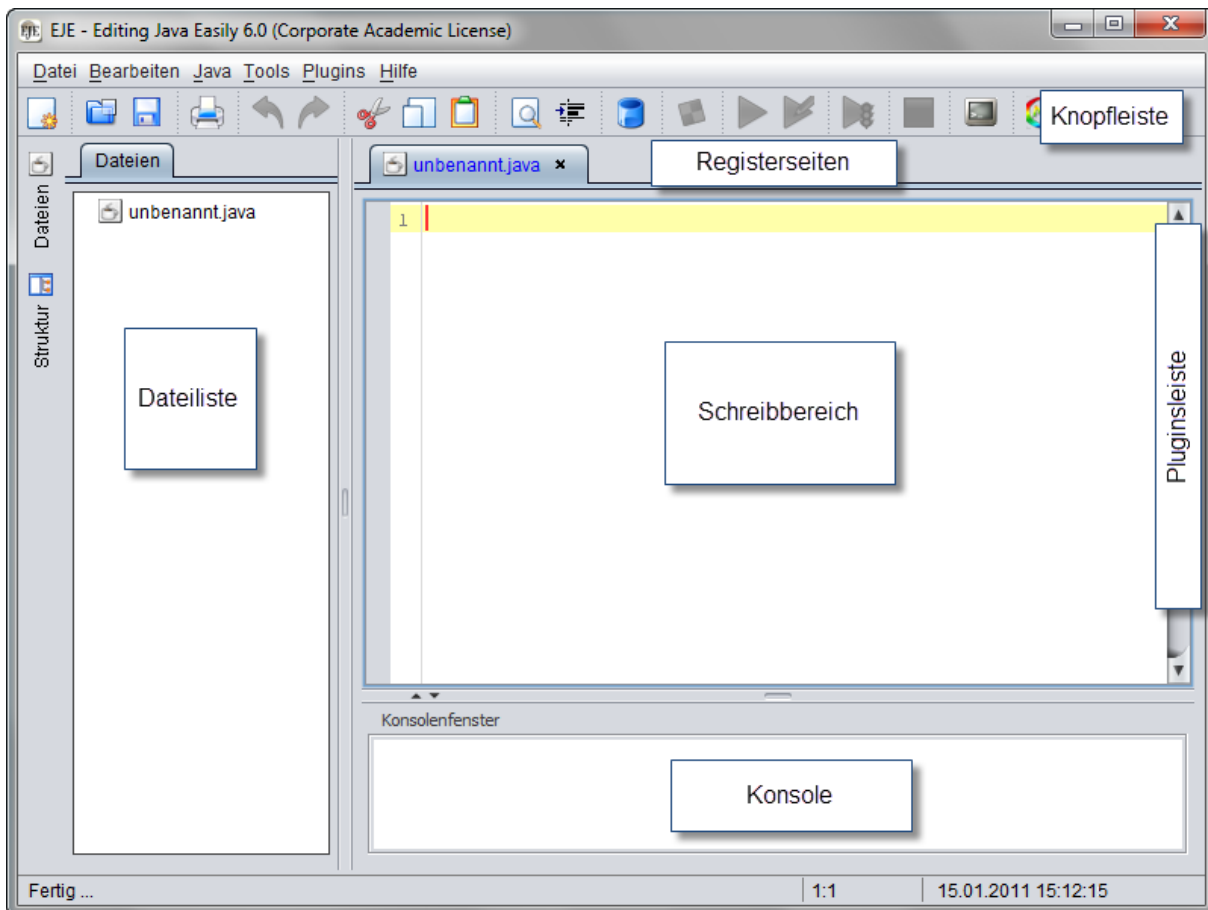


Fig. 1.1: Hauptfenster

Das Hauptfenster von EJE ist wie folgt aufgebaut: Der Schreibbereich ist durch eine Komponente realisiert, die jeweils aus einem Eingabebereich und einer Konsole besteht. Wenn mehr als eine Datei geöffnet ist, wird je Datei eine Registerseite angezeigt und ein Eintrag der Dateiliste hinzugefügt. Über die Dateiliste und über die Registerseiten können Sie dann zwischen den einzelnen Dateien umschalten.

Die Pluginleiste enthält – sofern installiert – den Zugang zu verschiedenen Plugins, die für die Programmierausbildung sinnvoll sein können. Da die Plugins aber nicht

Bestandteil der Anwendung sind, verweisen wir hier auf die Webseite, wo Sie alles Wichtige nachlesen können.

Die Buttons in der Toolbar sowie die Menüpunkte in den einzelnen Menüs beziehen sich, mit wenigen Ausnahmen wie „Neu“ und „Öffnen“, immer auf die aktuell selektierte Registerseite bzw. Datei. Buttons sind nur dann aktiv, wenn die dahinterliegende Aktion auch ausgeführt werden kann.

Beispiele:

1. Solange eine neue Datei nicht mindestens einmal gespeichert wurde, kann diese nicht compiliert werden. Der Programmbezeichner, dem der Dateiname entspricht, ist unter Umständen noch nicht bekannt. Diesen benötigt aber der Compiler, damit er den Quelltext compilieren kann.
2. Ein Programm kann nur dann ausgeführt werden, wenn es unverändert und compiliert ist.

1.2.1 Umgang mit Dateien

In EJE gibt es drei verschiedene Arten Programme zu schreiben bzw. zu verändern.

1. Wir beginnen auf der „grünen Wiese“ und erzeugen einen leeren Schreibbereich.
2. Wir verändern ein bestehendes Programm durch
 - a) Verwendung einer Programm-Vorlage
 - b) Öffnen eines selbstgeschriebenen Programms

Für die Umsetzung der Varianten sind die folgenden Menüpunkte bzw. Buttons zuständig.

Durch Betätigen des Menüpunkts Neu oder des zugehörigen Knopfes in der Toolbar wird ein leerer Arbeitsbereich erzeugt. Wenn Sie den Quelltext abgetippt haben, müssen Sie diesen abspeichern (Menüpunkt Speichern, Speichern unter oder zugehöriger Button). Wenn Sie sich an obigen Vorgaben halten (`public class ...`) schlägt EJE beim Speichern automatisch den richtigen Dateinamen (hier: `Kreis.java`) vor. Falls nicht, macht er Sie mit einer Dialogbox darauf aufmerksam.

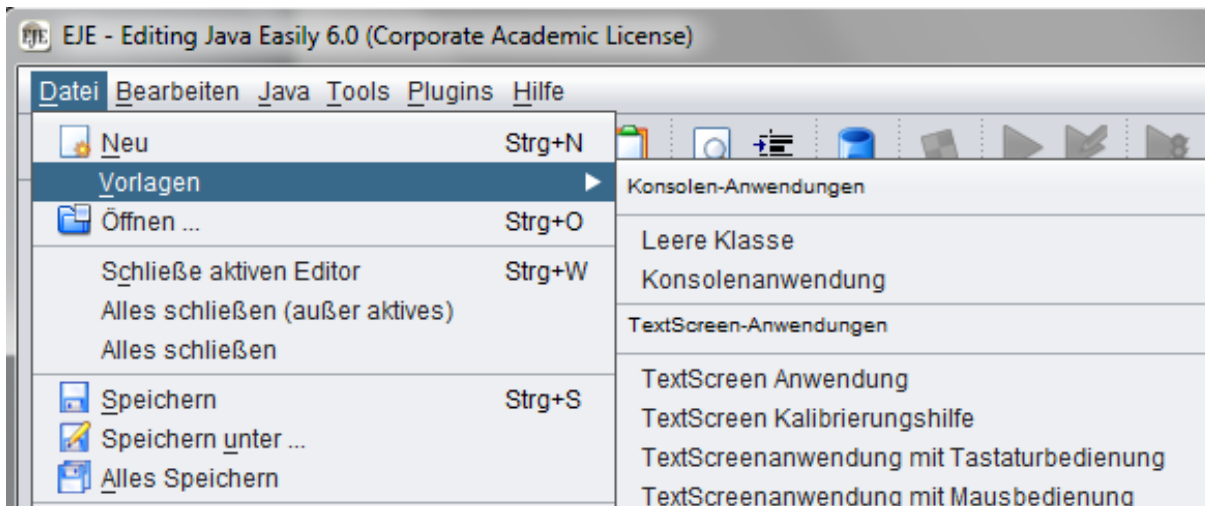


Fig. 1.2: Erzeugen neuer Schreibbereiche über das Menü



Fig. 1.3: Erzeugen neuer Schreibbereiche über die Toolbar

Wenn Sie den Menüpunkt Vorlagen auswählen, können Sie zwischen verschiedenen Vorlagen auswählen. Über den Menüpunkt Öffnen oder den zugehörigen Button in der Toolbar wird ein neuer Schreibbereich für eine existierende Datei.

In unserem Beispiel öffnen die Datei `Kreis.java` aus dem Verzeichnis `Quellen`. Das Resultat sieht dann wie folgt aus. Interessant ist, dass automatisch die Datei als verändert gekennzeichnet (blaue Schriftfarbe) im Gegensatz zum Schreibbereich unbekannt.java allerdings der Compiler-Button aktiviert ist.

1.2.2 Compilieren eines Java-Programms

Damit wir nun unser Programm compilieren können, müssen wir nur noch die Aktion Compilieren oder den zugehörigen Button in der Toolbar betätigen. Ein Speichern vorab ist nur dann notwendig, wenn das Programm neu erzeugt und noch nie gespeichert wurde. In allen anderen Fällen kennt EJE denn Dateinamen schon und speichert das Programm vor dem Compilieren automatisch.

Ein erfolgreiches Compilieren erkennt man zum einen an der Ausgabe in der Konsole (hier: Programm compiliert) und daran, dass die Buttons zum Ausführen eines Pro-

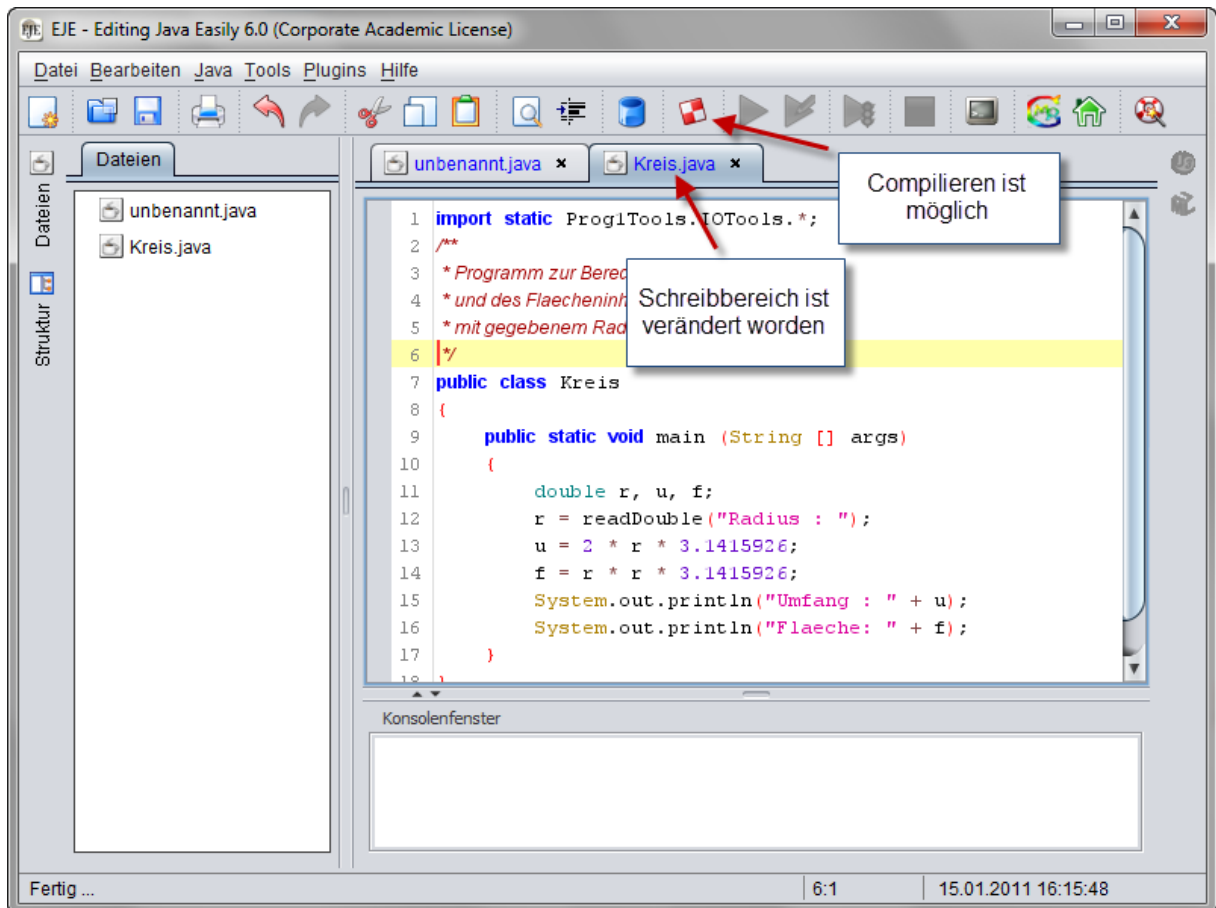


Fig. 1.4: Geöffnete Datei Kreis.java mit Möglichkeit zum Compilieren

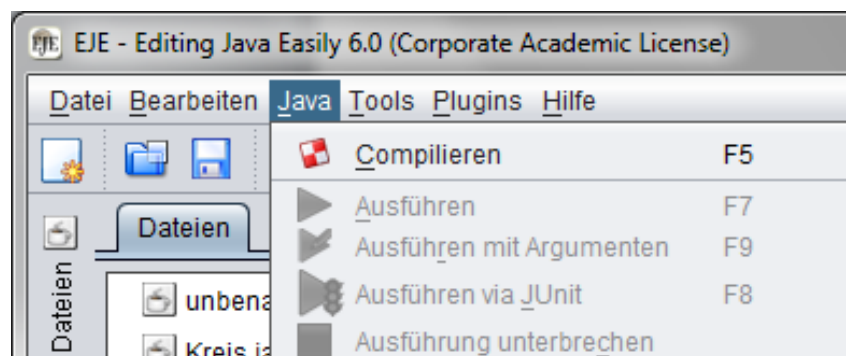


Fig. 1.5: Menü Java mit aktivierter Compilieren-Aktion

gramms aktiviert sind. Im gleichen Verzeichnis wurde nun vom Compiler eine Datei mit dem Namen Kreis.class erzeugt.

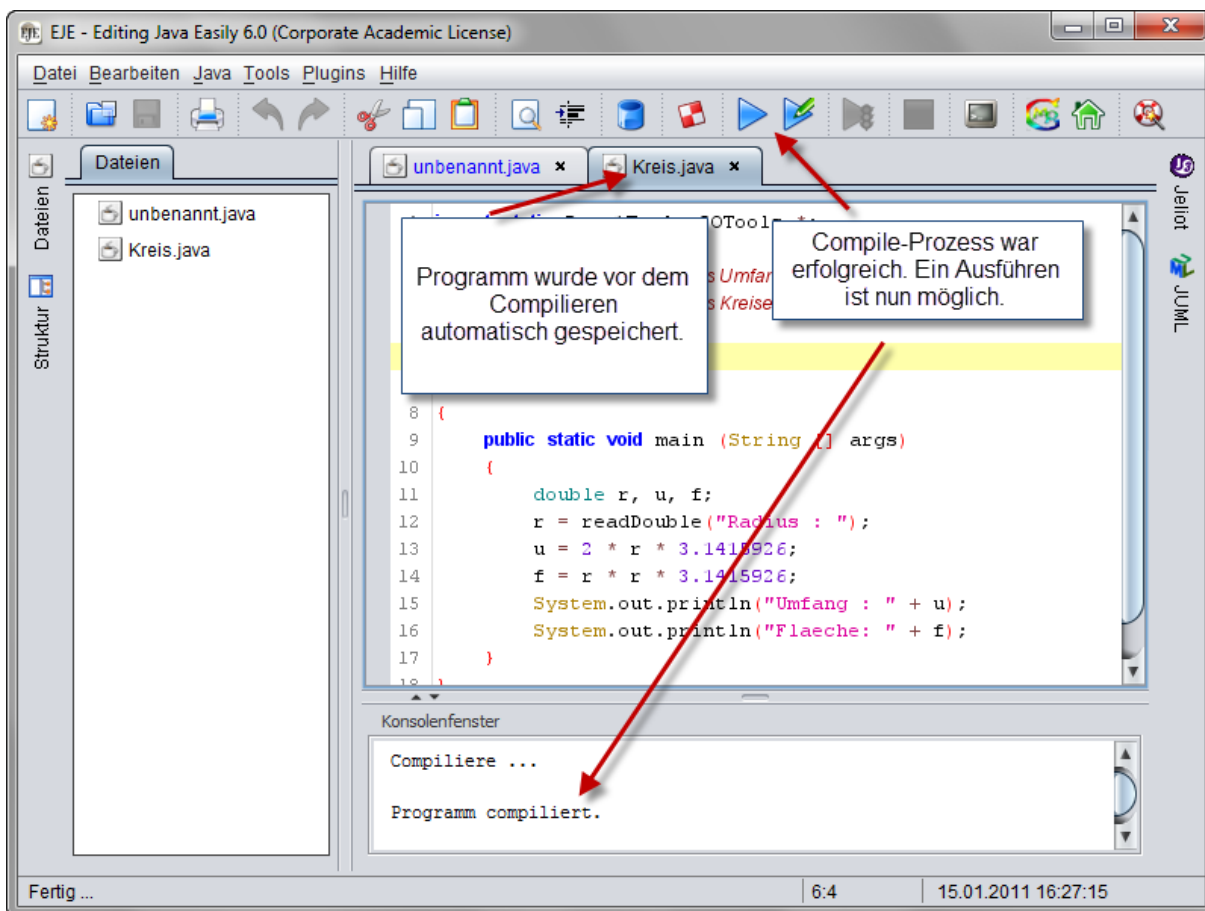


Fig. 1.6: Programm wurde erfolgreich compiliert. Eine Ausführung ist möglich.

1.2.3 Umgang mit Compilerfehlern

Der weitaus häufigere Fall ist jedoch, dass wir beim Compilieren Fehler erhalten. Um dies an einem Beispiel zu demonstrieren, modifizieren wir den Quelltext. Wir löschen in der Zeile 11 hinter dem `f` das Semikolon. Beim erneuten Compilieren wird uns folgendes Resultat präsentiert:

Die Konsole enthält eine Fehlerbeschreibung, die übersetzt bedeutet: In der Datei `Kreis.java` wurde in Zeile 11, Spalte 23 festgestellt, dass ein Semikolon fehlt.

Dies ist logisch, denn genau dieses haben wir ja auch gelöscht. Durch einfaches Markieren der Fehlermeldung springt der Cursor automatisch in die fehlerhafte Stelle.

Leider sind aber nicht alle Fehlermeldungen des Compilers so verständlich wie diese. Um bei der Behebung des Fehlers (hier: Einfügen des Semikolons an markierter Stelle) zu helfen, bietet EJE die MindProd-Fehlermeldungsbeschreibungsdatenbank an. Mit einem Doppelklick auf die Fehlermeldung öffnet sich ein neuer Dialog.

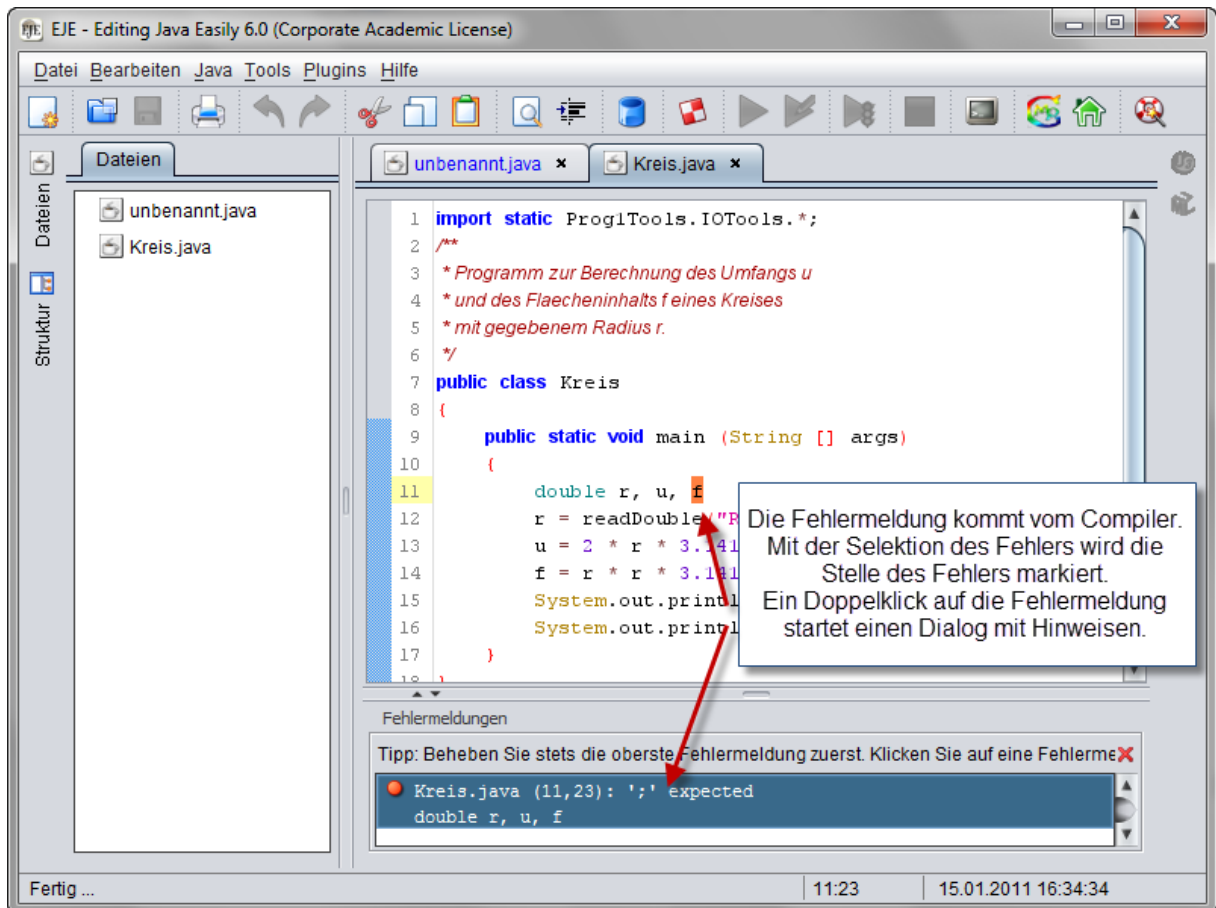


Fig. 1.7: Datei mit Syntaxfehler

Wie man unschwer erkennen kann, passt der erste Hinweis (hier: „Usually this is just a missing semicolon“) zur Behebung unseres Problems. Allerdings ist auch ersichtlich, dass auch andere Fehler im Quelltext diese Fehlermeldung provozieren können. Da im Regelfall mehrere Fehlermeldungen auftauchen, empfiehlt es sich also stets mit der Behebung der ersten Fehlermeldung zu beginnen und danach neu zu compilieren. Häufig bedingt der erste Fehler die Folgefehlermeldung.

Nach der Korrektur unseres Programms compilieren wir es erneut und erhalten die in 6 dargestellte Ansicht.

1.2.4 Ausführen eines Java-Programms

Unser Java-Programm können wir nun durch Betätigen des gleichnamigen Menüpunkts oder Knopfs ausführen.

1 Hands-On Programming mit Editing Java Easily

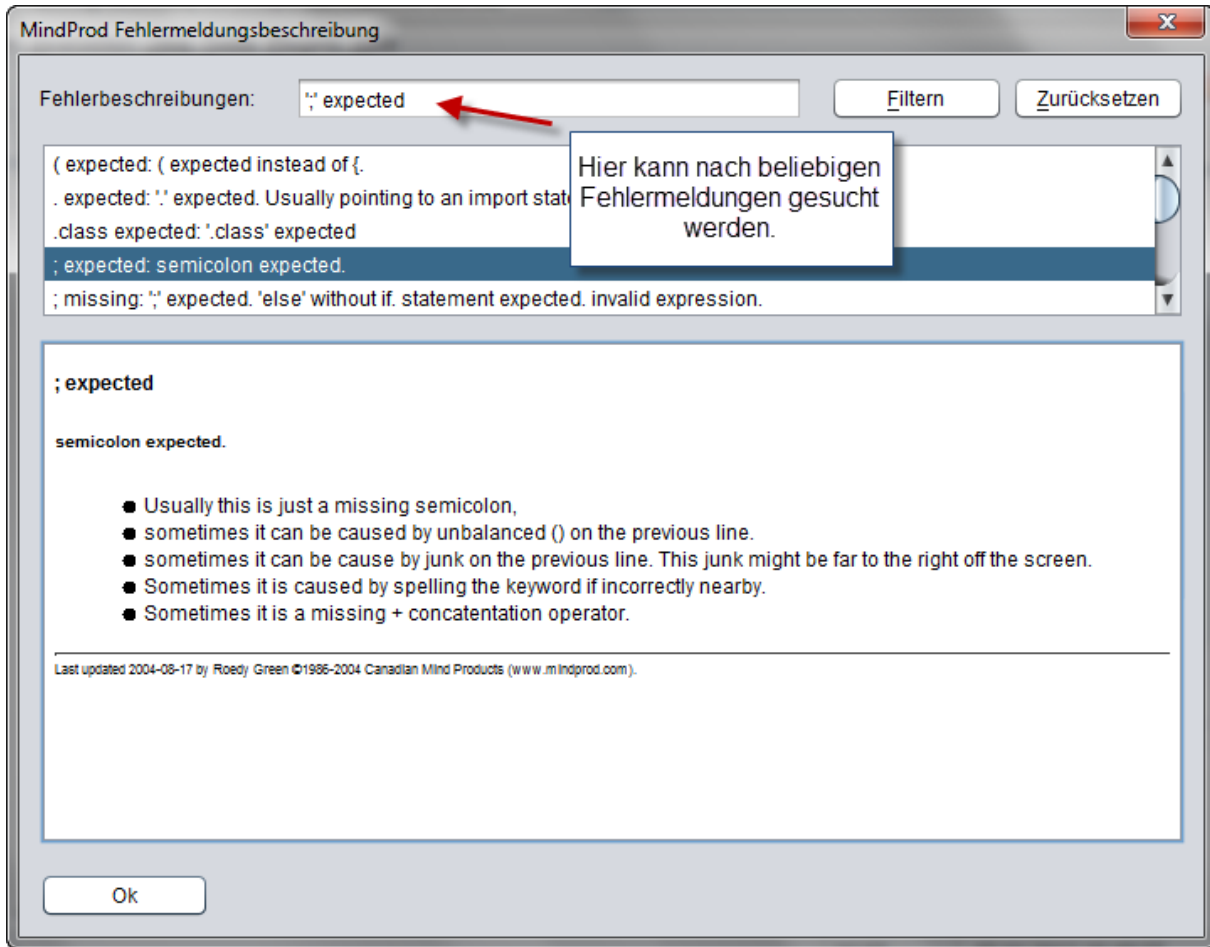


Fig. 1.8: Mindprod- Fehlermeldungsbeschreibung

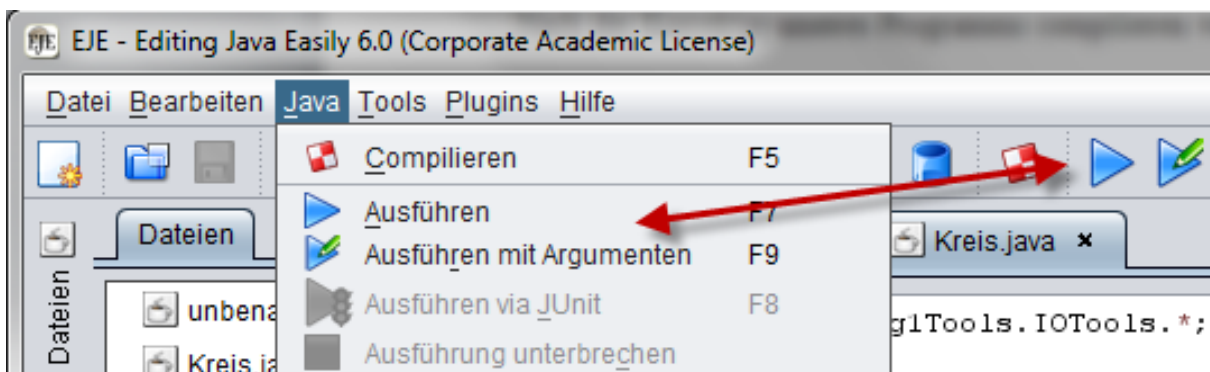


Fig. 1.9: Menü Java für ausführbares Programm

Der Interpreter beginnt in der Hauptmethode `main` und arbeitet sich dann Anweisung für Anweisung durch das Programm.

In der Konsole werden Ausgaben sichtbar (zum Beispiel in Zeile 15: `System.out.println`) gemacht und Eingaben (zum Beispiel in Zeile 12: `readDouble`) verarbeitet. Dass ein

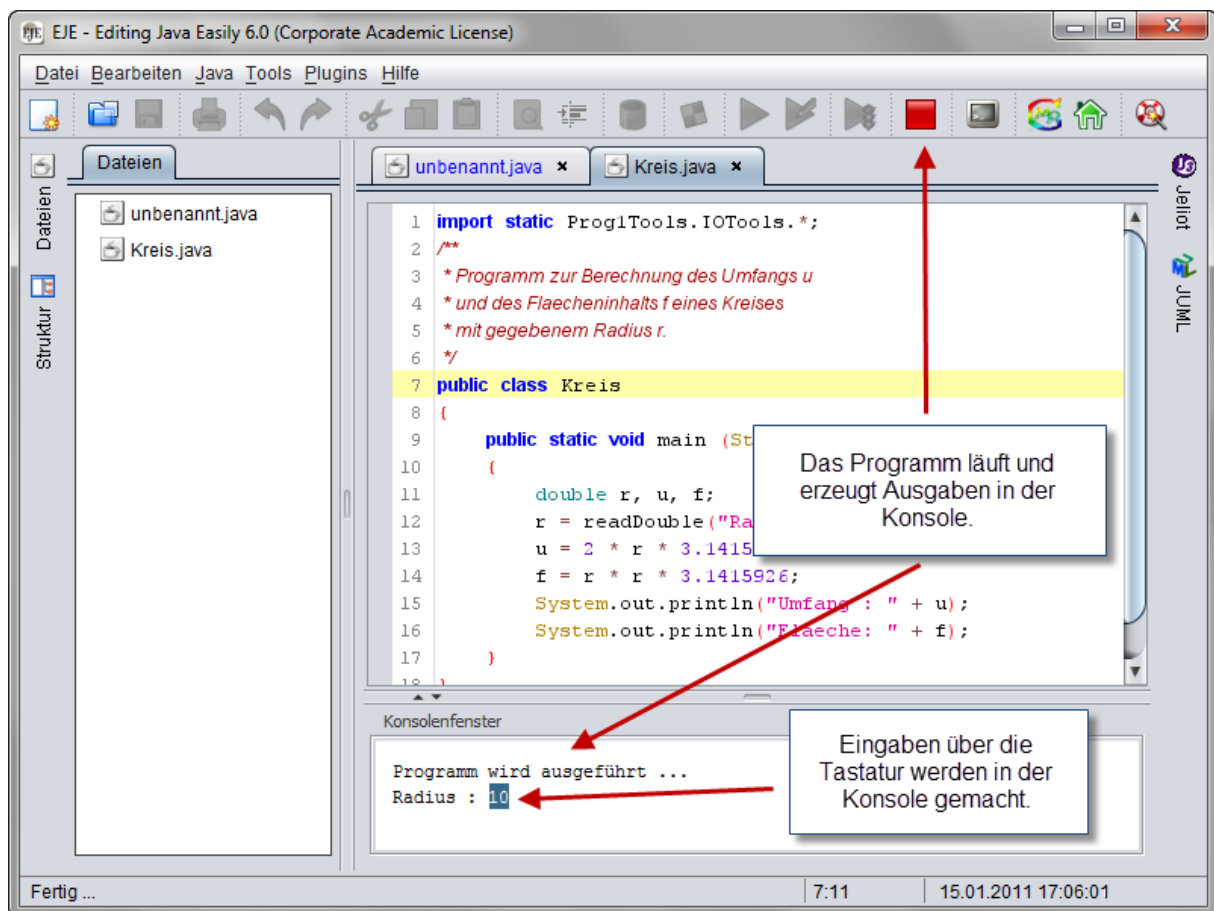


Fig. 1.10: Java-Programm wird ausgeführt.

Programm noch läuft, weil es zum Beispiel auf eine Eingabe wartet, ist unter anderem daran erkennbar, dass der Stopp-Button für den Schreibungsbereich aktiviert ist. Durch Betätigen des gleichnamigen Menüpunkts bzw. Knopfes kann die Ausführung eines Java-Programms jederzeit unterbrochen werden.

1.3 Exkurs: Arbeiten mit Jeliot

Jeliot ist ein Algorithmenvisualisierungswerkzeug für Programmieranfänger und ein Produkt der Universität Joensuu, Finnland. Es animiert beliebige⁵ Quelltexte und soll damit einen Quelltext für einen Programmieranfänger leichter nachvollziehbar darstellen.

⁵Es gibt einige wenige Einschränkungen im Vergleich zum Sprachstandard Java. Diese sind aber für die Bearbeitung der Aufgaben dieses Buches nicht von Bedeutung.

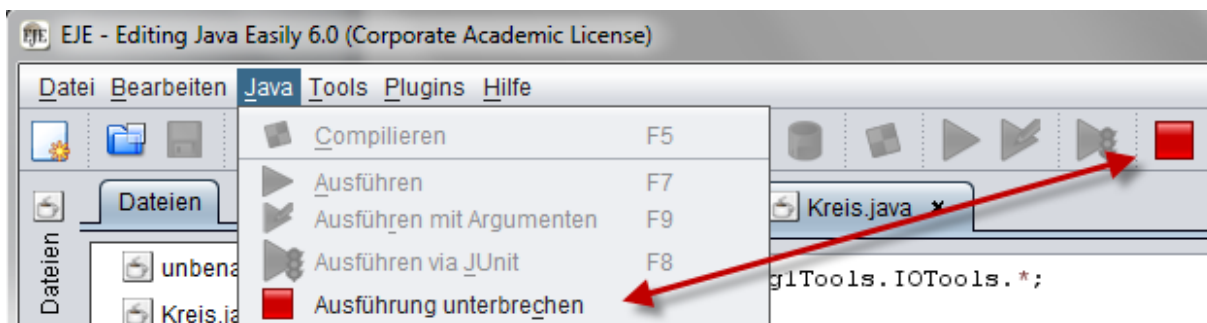


Fig. 1.11: Java-Programm in der Ausführung unterbrechen

Die Anwendung ist auch als Standalone⁶-Variante erhältlich. Für EJE wurde am KIT im Rahmen eines Studierendenprojekts ein Plugin entwickelt, das Jeliot integriert.

1.3.1 Installation

Jeliot als Plugin für EJE kann über den Menüpunkt Plugins – Konfiguration installiert werden. Dazu aktualisiert man den sich öffnenden Dialog und installiert das Plugin Jeliot.

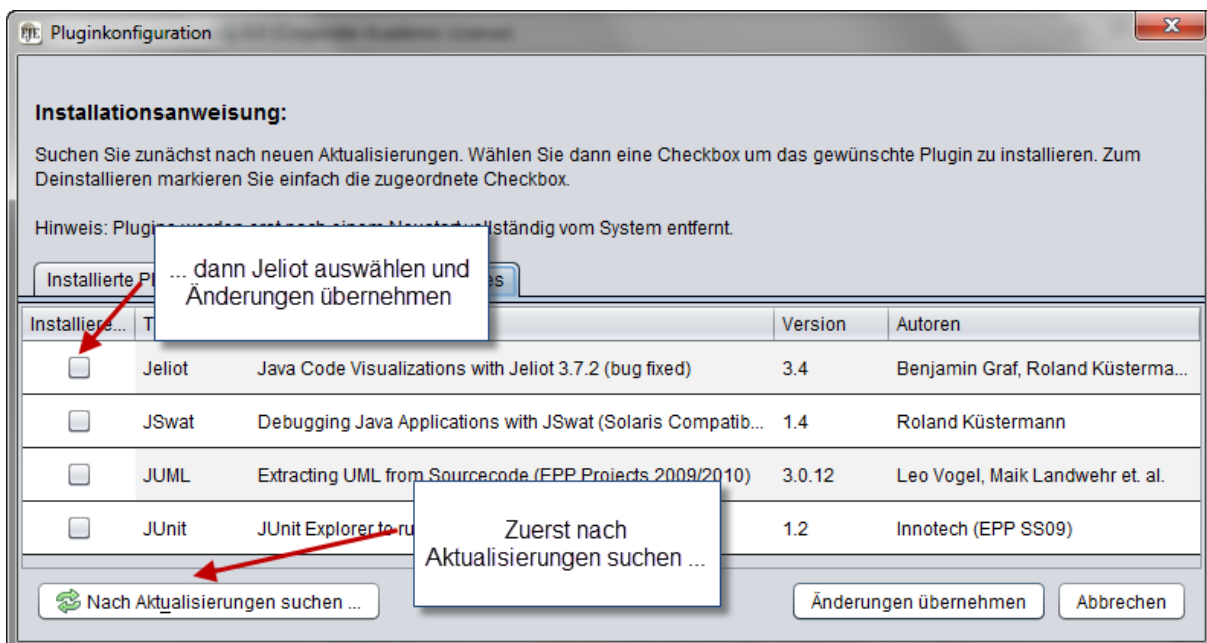


Fig. 1.12: Plugins aktualisieren und auswählen

⁶Das Werkzeug Jeliot kann man unter <http://cs.joensuu.fi/jeliot/> herunterladen.

Danach werden die notwendigen Bibliotheken automatisch aus dem Internet geladen und installiert. Bitte starten Sie die Anwendung neu. Eine erfolgreiche Installation erkennt man an dem deaktivierten Symbol in der Pluginleiste.

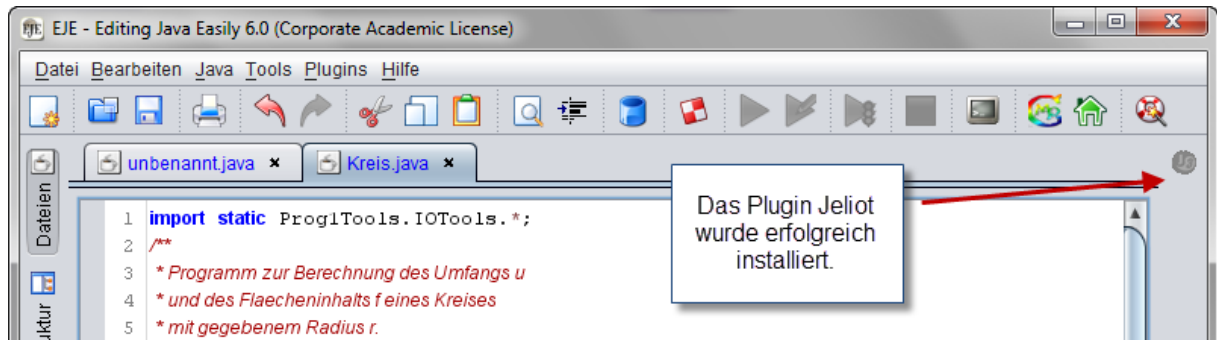


Fig. 1.13: Plugin wurde erfolgreich installiert

1.3.2 Ausführen von Java-Programmen mit Jeliot

Eine notwendige Voraussetzung um ein Programm mit Jeliot auszuführen ist, dass das Java-Programm kompiliert wurde. Sofern noch nicht geschehen, öffnen Sie wieder die Datei `Kreis.java` und kompilieren diese.

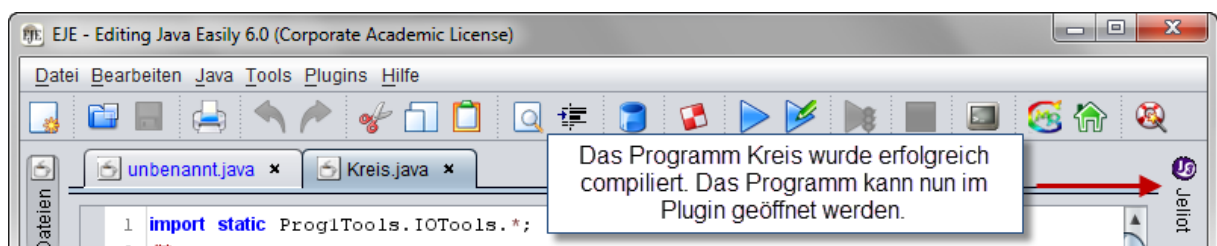


Fig. 1.14: Jeliot ist ausführbar.

Durch Betätigen des Plugin-Knopfes in der Pluginleiste wird das Plugin gestartet. Wenn Jeliot das Programm erfolgreich verarbeiten konnte, können Sie die Animation mit dem

1 Hands-On Programming mit Editing Java Easily

Play-Button abspielen. Die Geschwindigkeit wird über den Schieberegler eingestellt. Zu jedem Zeitpunkt kann die Animation gestoppt und zurückgesetzt werden. Wahlweise ist auch ein schrittweises Weiterschalten möglich.

Jeliot verarbeitet die Ausdrücke in unserem Java-Programm in der gleichen Reihenfolge wie der Interpreter. Ausdruck für Ausdruck wird abgearbeitet und visualisiert. Die Eingabe erfolgt im Bereich des Plugins. Dort werden Sie aufgefordert entsprechend des Programmablaufs den Eingabewert einzugeben. Die Ausgabe erfolgt nach wie vor in der Konsole.

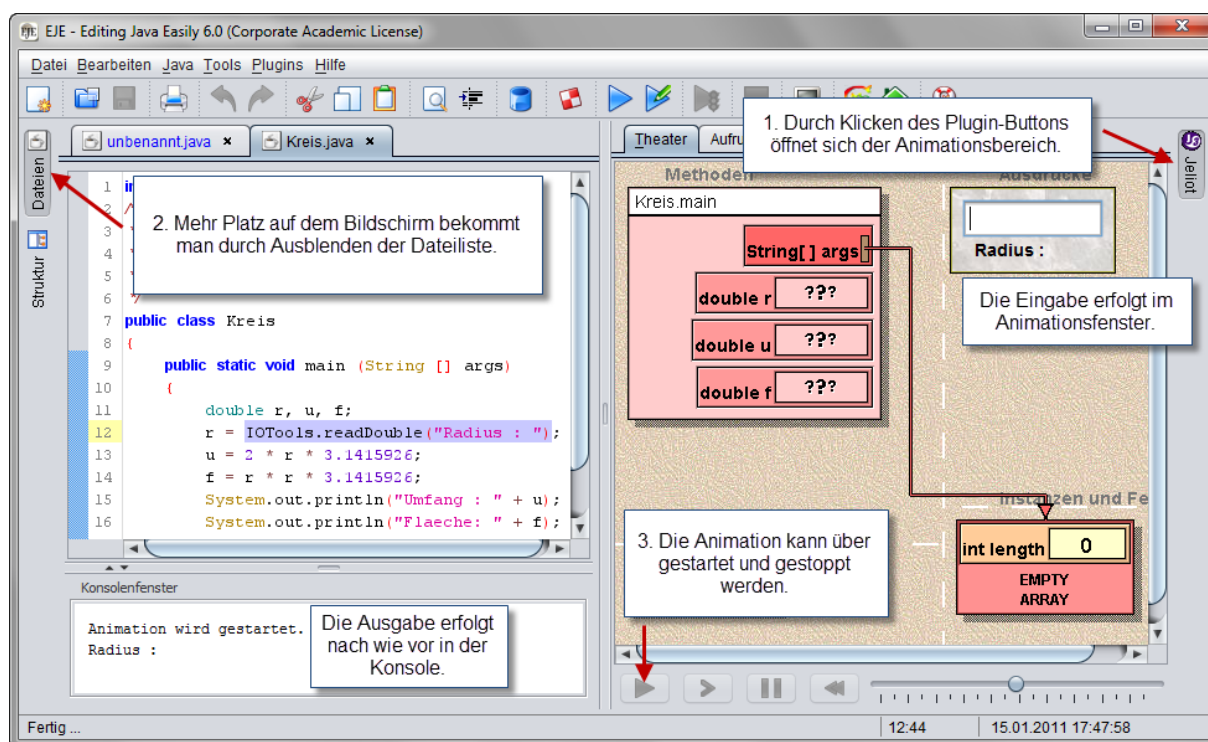


Fig. 1.15: Jeliot bei der Ausführung

Die Animation eines Ausdrucks soll zum einen dabei helfen, eine Vorstellung zu entwickeln, wie ein Interpreter arbeitet. Zum anderen hilft es beim Auffinden von logischen Programmfehlern. Logische Programmfehler sind solche, die weder beim Compilieren (Compilerfehler) noch beim Ausführen (Laufzeitfehler) eine Fehlermeldung erzeugen.

Beispiel: wenn wir in der Formel zur Berechnung des Umfangs des Kreises in Zeile 13 des Programms + statt * schreiben würden, wären schlichtweg die Ergebnisse falsch. Jeliot kann dabei helfen solche Fehler zu finden.

Für den interessierten Leser verweisen wir auf die integrierte Hilfe, die entweder über das Popupmenü im Plugin oder über das Jeliot-Untermenü im Plugins-Menü zugreifbar ist. Hier werden Hilfestellungen im Umgang mit Jeliot für Anfänger, Fortgeschrittene und Dozenten gegeben.

1.4 Zusammenfassung

In diesem Kapitel haben wir gezeigt, wie mit Hilfe der Entwicklungsumgebung Editing Java Easily auf einfache Art und Weise Programme editiert, kompiliert und ausgeführt werden können. Darüber hinaus haben wir in einem Exkurs den Umgang mit der freiverfügbaren Anwendung Jeliot gezeigt. Mit dieser Anwendung können Sie Ihre eigenen Programme visualisieren und analysieren. Aufgrund der Tatsache, dass Software laufend weiterentwickelt wird, verweisen wir auf die Webseite bzw. auf die integrierte Hilfe sowohl der Anwendung Editing Java Easily als auch des Plugins Jeliot.