Entdeckendes Lernen mit einem interaktiven Online-Tutorium zur Programmierung in Java

Claudia Bieg
FR 6.2 Informatik
Universität des Saarlandes
66041 Saarbrücken
cbieg@cs.uni-sb.de

Stephan Diehl
FR 6.2 Informatik
Universität des Saarlandes
66041 Saarbrücken
diehl@acm.org

Abstract: Bei den im Web angebotenen Java-Tutorials handelt es sich meist um mehr oder weniger gut aufbereitete Vorlesungsskripte oder Textbücher, die sich in Bezug auf Interaktivität und Adaptivität nicht wesentlich von ihrer Druckfassung unterscheiden. Das in diesem Papier vorgestellte Tutorial *JOSH-online* ermöglicht das schrittweise Erlernen des Programmierens in Java durch interaktives Ausprobieren. Dabei werden dem Lerner nicht nur Übungen vorgegeben, sondern er wird dazu ermutigt, sein neu erworbenes Wissen anhand eigener Fragestellungen und deren experimenteller Überprüfung zu festigen und zu vertiefen.

1 Einleitung

JOSH ist ein Java Interpreter [DB03b], der entwickelt wurde, um Anfängern einen einfachen Einstieg in die Programmiersprache Java zu bieten. Dieser Interpreter wurde zu einem server-basierten Interpreter-Applet umstrukturiert und in ein Online-Tutorial über die Programmierung in Java integriert und dadurch zu dem System *JOSH-online* [Bi03] erweitert. Im Folgenden werden wir beide Systeme kurz vorstellen und speziell *JOSH-online* im Bezug auf sein Nutzen als Lernsystem analysieren. Auf Implementierungsaspekte beider Systeme werden wir nicht eingehen, diese sind an anderer Stelle nachzulesen [Bi03] [DB03a].

1.1 JOSH

JOSH ist ein stand-alone Java Interpreter, der es dem Lerner ermöglicht, Ausdrücke auszuwerten, einfache Anweisungen auszuführen, Variablen zu deklarieren und Methoden zu definieren. Auf die Definition von Klassen kann dabei anfangs verzichtet werden.
Im Folgenden verwenden wir den Ausdruck *einfaches Codefragment* als Bezeichnung für einfache Ausdrücke und Anweisungen oder für Variablen-, Methoden-, und Klassendeklarationen. Der Ausdruck *Codefragment* bezeichnet die Folge von einem oder mehreren einfachen Codefragmenten. Sobald der Benutzer ein Codefragment eingibt, startet *JOSH* die Auswertung oder Ausführung. Genauer wird durch das Drücken des RETURN-Buttons

der folgende Prozess gestartet: *JOSH* überprüft, ob die bisherige Eingabe ein Codefragment, ein Präfix eines Codefragments oder etwas anderes darstellt. Im ersten Fall wird das Codefragment ausgeführt, im zweiten Fall wartet *JOSH* auf weitere Benutzereingabe, die das Präfix zu einem Codefragment ergänzt. In allen anderen Fällen kann die Eingabe nicht zu einem Codefragment ergänzt werden. Darum wird ein Syntaxfehler angezeigt und die Eingabe gelöscht. Der Benutzer kann den Prozess erneut starten.

Üblicherweise sehen die ersten Programme, die ein Anfänger in Java schreibt, wie folgt aus:

```
public class Hello {
  public static void main(String[] args)
    { System.out.println("Hello World");
    }
}
```

Um die Bedeutung all dieser verwendeten Konstrukte zu verstehen, muss der Lerner bereits alles über Klassen, statische Methoden, Packages, Strings, Felder usw. wissen. In *JOSH* genügt einfach folgende Eingabe:

```
> println("Hello World");
Hello world
```

Die Eingabe `println("Hallo World");` wird vom Interpreter eingelesen und ausgewertet. Als Feedback erhält der Lerner die Ausgabe `Hello World`. Die folgende Beispielsitzung zeigt weitere Möglichkeiten von *JOSH* auf:

```
> 3 + 'c'
102 : int
> class A { int x; void setX(int n) { x=n; } };
class generated
> A a = new A();
Field added
> a.setX(9);
> a.x
9 : int
```

1.2 JOSH-online

Bei der Verwendung von *JOSH* in der Lehre sind einige Probleme aufgetreten. Zum einen musste die Java-Entwicklungsumgebung JDK installiert werden, zum anderen mussten die Umgebungsvariable CLASSPATH und andere Pfade in der Konfigurationsdatei korrekt gesetzt werden. Um den Zugriff auf *JOSH* zu vereinfachen und um *JOSH* einem größeren Lernkreis zur Verfügung zu stellen, wurde *JOSH-online* entwickelt. Die grundlegende Idee

ist, *JOSH* in einem Browser zu verwenden, der Java unterstützt. *JOSH-online* integriert den Interpreter in ein interaktives, netzbasiertes Tutorium über die Programmierung in Java. Abbildung 1 zeigt die Definition und Ausführung der Methode `fak()` im in das Tutorial integrierten Interpreter.

2 Das online Java Tutorium

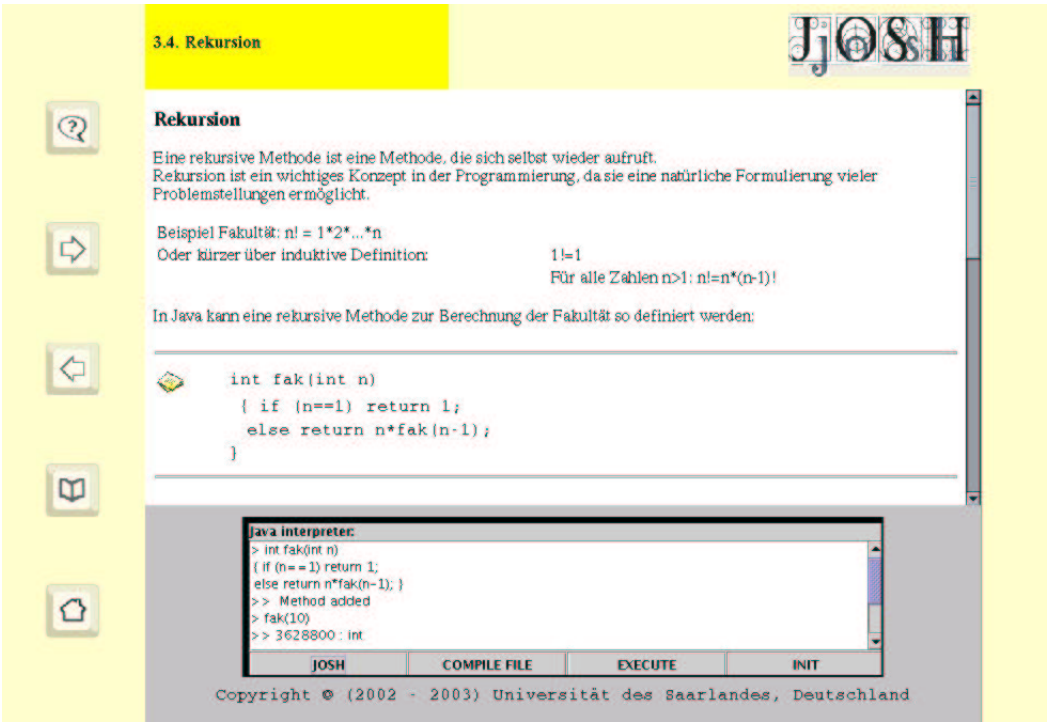

Abbildung 1: Screenshot: *JOSH-online* Tutorium

Der Aufbau des Tutorium richtet sich nach dem intuitiven Weg: Angefangen bei den Grundlagen wie einfache Datentypen und Variablen, führt das Tutorium über Methodendeklarationen zu der Abstraktion durch Klassen. Zu den Besonderheiten von *JOSH* gehört, dass bei der Auswertung von Ausdrücken nicht nur der Wert, sondern auch der Ergebnistyp ausgegeben wird (vgl. Abb. 1). Dies erleichtert den Lernenden, das Typensystem und die Typenkonvertierung in Java zu verstehen. Jede Lerneinheit des Tutoriums endet mit Beispielen und Übungen, die interaktiv mittels des Interpreters im Textbuch bearbeitet werden können. Der interessanteste Aspekt des Tutoriums ist die Kommunikation zwischen Textbuch und Interpreter-Applet. Jedes Quellcode-Beispiel aus dem Textbuch kann direkt getestet werden, indem der Button neben dem Beispiel gedrückt wird. Der Quellcode wird

automatisch in das Textfeld des Applets eingefügt. Der Benutzer kann den Text dann nach Belieben verändern und ausführen lassen. Einige Übungen bauen kapitelübergreifend aufeinander auf. Der Interpreter garantiert, dass die Werte erhalten bleiben, die den bereits deklarierten Variablen, Methoden oder Klassen zugewiesen wurden. Im Verlauf der Arbeit mit dem Tutorium wird der Lerner oft dazu ermutigt, eigene Beispiele zu finden und diese interaktiv mit dem Interpreter zu testen. Hier liefert das Tutorium nützliche Vorschläge, die den Benutzer anleiten, Eigenheiten der Programmiersprache Java zu erkunden.
Der Text des Tutoriums wurde in zwei Kursen entwickelt, die an den Universitäten in Duisburg und Saarbrücken gehalten wurden [DG01]. In diesen Kursen wurde der Stand-alone Interpreter *JOSH* verwendet. In Zukunft werden wir die Online-Version in Programmierkursen einsetzen.

3 JOSH-online als Lernsystem

Das Tutorium vermittelt dem Benutzer ein Basiswissen über die Programmierung in Java. Sein Aufbau gliedert sich in Lerneinheiten. Jede Lerneinheit verfolgt zwei Ziele: die Wissensvermittlung und die Wissensfestigung. Daher besteht jede Lerneinheit aus folgender Instruktionssequenz [Bl98]: Zuerst werden Informationen präsentiert, danach folgen zum Thema passende Fragestellungen. Im Interpreter wird die Antwort des Benutzers analysiert und ausgewertet. Bei einer fehlerhaften Eingabe oder einer falschen Antwort kann die Problemstellung neu bearbeitet werden. Sonst folgt die nächste Fragestellung oder eine neue Informationseinheit. Die Gestaltung der Beispiele einer Lerneinheit basiert auf der Idee, dass der Benutzer die Beispiele selbständig testet. Der integrierte Interpreter liefert das Feedback. Analog gestalten sich die Übungen. Das Tutorium gibt Hilfestellungen zur Bearbeitung der Übungen. Die Antworten gibt der Benutzer in den Interpreter ein und startet die Auswertung (Abb. 2). Falls der Benutzer nach mehreren Lösungsversuchen keine Antwort findet, werden Musterlösungen angeboten.
Ein großer Vorteil des Interpreters ist seine Flexibilität. Der Benutzer ist nicht an die Beispielvorgaben oder die angebotenen Übungen gebunden. Er wird angeregt, sich eigene Beispiele zu überlegen. Vermeintliche Wissenslücken kann der Benutzer durch Experimente selbständig füllen. Getreu dem Motto *'Probieren geht über Studieren'* wird dem Benutzer die Möglichkeit des aktiven Experiementierens geboten. *JOSH* erlaubt damit den Übergang von **Drill & Practice** zu einem entdeckenden Lernen [Br66], also einem an die wissenschaftliche Methode angelehnten Lernen aufgrund von **Hypothesize & Test**. Statt fest vorgegebener Aufgabenstellung können beliebige Experimente (eventuell unter Anleitung) ausgeführt werden. Die Ergebnisse sollen vom Benutzer reflektiert und in einen Zusammenhang gebracht werden. Durch weitere Tests können die Ergebnisse gefestigt und die zugrundeliegenden abstrakten Konzepte erkannt werden. David Kolb [Ko84] beschreibt einen **Lern-Zyklus** (Abb. 3), der auf dieser Idee basiert:

1. **Experimentieren:** Der Benutzer arbeitet sich in die Aufgabenstellung ein. In *JOSH-online* bietet das Tutorium Anhaltspunkte. Notwendige Beispiele werden vorgegeben mit der Intention, den Benutzer zu einer Vermutung zu leiten. In dieser Experimentierphase kann das **Drill & Practice** Schema aus Abbildung 2 eingebunden

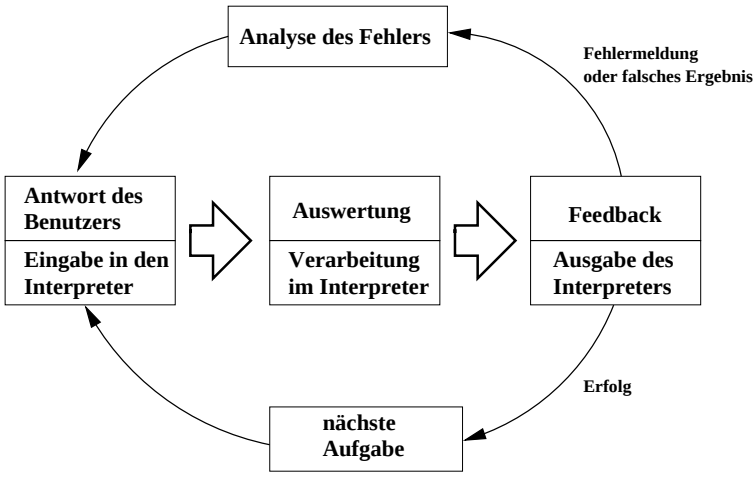

Abbildung 2: Drill & Practice Schema von *JOSH-online*

werden.

2. **Reflexion:** Der Benutzer muss sich von der ursprünglichen Aufgabenstellung entfernen. Ziel ist es, Erfahrungen zu sammeln, die man aus den erzielten Ergebnissen gewonnen hat.

3. **Konzeptgewinnung:** Die Ergebnisse müssen interpretiert und in einen Zusammenhang gebracht werden. So werden neue Konzepte gewonnen. Die theoretischen Aspekte, die das Tutorium liefert, helfen, die Ergebnisse der Experimente zu erklären.

4. **Planung:** Die gewonnenen Ergebnisse werden verwendet, um Vorhersagen zu treffen, welches Resultat bei weiteren Tests erwartet wird. Die erzielten Erkenntnisse können dadurch gefestigt oder sogar präzisiert werden.

JOSH-online realisiert diesen Kreislauf, indem es zu Experimenten anregt. Bereits im Tutorium angeeignetetes Wissen kann in der Reflexionsphase mit den experimentellen Ergebnissen in Zusammenhang gebracht werden. Aus den Ergebnissen soll ein Konzept gewonnen werden. Falls erforderlich, können beliebig viele weitere Experimente ausgeführt werden. Das Feedback des Interpreters bestätigt oder widerlegt die Ergebnisse, die in der Planungsphase erwartet wurden.

3.1 Eigenschaften

Tutorielle Systeme werden als *linear organisierte Programme mit hohem Grad an Systemsteuerung* [Bl98] beschrieben. Üblicherweise arbeitet der Lerner die vorgegebenen Sequenzen ab. Interaktivität wird dabei wenig gefördert. Folgende Eigenschaften ergänzen

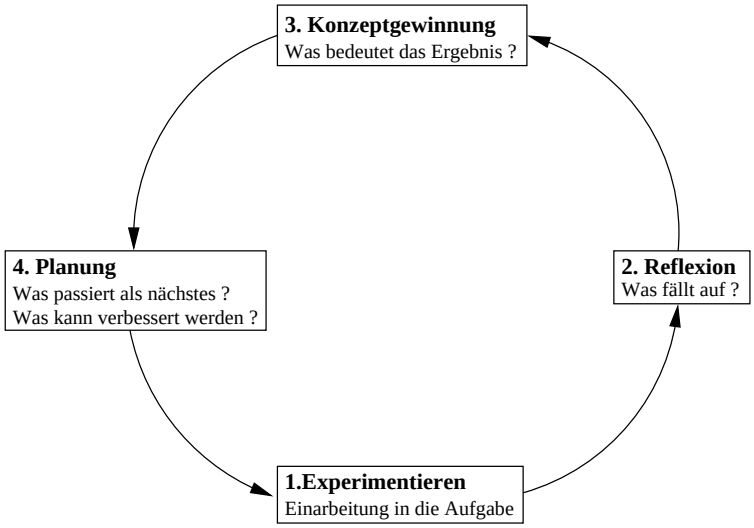

Abbildung 3: Der Lern-Zyklus von D.Kolb

JOSH-online von einem Tutorium zu einem komplexeren System:

Multimedia Die Inhalte werden in Buchform visualisiert und durch zahlreiche Bilder ergänzt. *JOSH-online* wurde einfach gestaltet, um Übersichtlichkeit und einfache Benutzung zu ermöglichen.

Adaptivität Dem Benutzer werden Lerninhalte angeboten, deren Bearbeitungsreihenfolge er selbst festlegen kann. Für den richtigen Lernweg wird ihm eine Anleitung geboten. So ist es für Fortgeschrittene kein Problem, direkt in den weiterführenden Kapiteln einzusteigen. In den einzelnen Kapiteln werden die bereits getesteten Beispiele und Übungsaufgaben markiert. Die Übungen können mit Hilfe des Tutoriums berarbeitet werden. Die Anzahl der Lösungsversuche ist unbeschränkt. Sollte der Benutzer nicht selbstständig eine Lösung finden, werden ihm Musterlösungen angeboten.

Interaktivität Über den integrierten Interpreter können Benutzer und System miteinander kommunizieren. Die Testeingaben werden im Interpreter ausgeführt und der Benutzer erhält ein Feedback: bei korrekter Eingabe wird das Ergebnis der Auswertung ausgegeben, bei fehlerhafter Eingabe wird auf die Fehlerquelle hingewiesen.

JOSH-online ist einfach gestaltet, aber sehr vielseitig. Der Schwerpunkt liegt auf der Interaktivität des Systems: Interpreter und Benutzer führen einen Dialog. Zur Zeit ist *JOSH-online* passiv adaptiv. Durch den Aufbau des Systems ist es leicht zu einem aktiv adaptiven System erweiterbar, wie es zum Beispiel in Abschnitt 3.3 beschrieben wird.

3.2 Kritik

Computerbasierte Lernsysteme müssen sich vielen Kritiken ausssetzen (vgl. [BBMG02]). Komplexeren Systemen wird unterstellt, dass motivationsfördernde Äußerlichkeiten wie Cover-Stories zu sehr vom eigentlichen Lerninhalt ablenken oder dass die Benutzung zu unübersichtlich ist. Durch Verzicht auf inhaltsbezogene Ausschmückungen und durch eine einfache Gliederung des Systems, haben wir versucht, *JOSH-online* benutzerfreundlich zu gestalten. In anderen Fällen wie z.B. bei Mathematik-Programmen sind Fehler im Programm aufgetaucht, die zu unkorrekten Ergebnissen führten. Der *JOSH-online*-Interpreter wurde als Front-End des Java Compilers [Bi03] [DB03a] implementiert. Dadurch gewährleisten wir, dass die Java Syntax und Semantik erhalten bleibt. Andere Probleme konnten noch nicht behoben werden. Der Benutzer kann eventuell in eine Endlosschleife geraten, wenn das Programm bei einem Problem immer wieder die selben, nicht weiterführenden Hinweise gibt. Vieleicht helfen die Lösungsvorschläge in dieser Situation weiter, die im Tutorium angegeben sind.

3.3 Mögliche Erweiterungen

Die Grundlage der Implementierung von *JOSH-online* bildet die Verteilung der Systemelemente auf Client und Server. Die Auswertung der Codefragmente erfolgt dabei zentral auf dem Server, der die Daten der verschiedenen Benutzer verwaltet. Diesen Ansatz kann man nutzen, um das System adaptiver zu gestalten. Zur Zeit sind diese Daten nur gesichert, solange der Benutzer aktiv ist. Durch Einführung eines Benutzernamens können die Benutzer dauerhaft auf dem Server verwaltet werden. Es ist dann möglich, Informationen wie zum Beispiel den Entwicklungsgrad oder die vom Benutzer bereits bearbeiteten Lerneinheiten zu speichern. Der Schwierigkeitsgrad der Übungsaufgaben kann an die Benutzerkenntnisse angepasst werden.
JOSH-online wurde insbesondere für den Einsatz im Intranet von Schulen entwickelt. Hier treten Aspekte wie die Verwendung des Systems in Prüfungssituationen in den Vordergrund. Alle Benutzereingaben fliessen über den Server. An dieser zentralen Stelle hat der Lehrer die Möglichkeit, die Eingaben der einzelnen Schüler zu prüfen und kann ihre Lernfortschritte erkennen. Die Prüfungsaufgaben können online gestellt werden. Um die Bearbeitungszeit abzugrenzen, kann die Lebensdauer des Applets eingeschränkt werden. Bei der Weiterentwicklung von *JOSH-online* sollen einige der angeführten Ideen berücksichtigt werden.

4 Verwandte Arbeiten

In den letzten Jahren wurde eine Reihe von Java-Interpretern entwickelt [Ro01, Hi, Ni]. Sie wurden aber bisher nicht in Online-Tutorials integriert. Die sogenannten interaktiven Tutorials, die wir bisher für LISP [EG03], Ruby [RU03] und NESL [Bl] gefunden haben, sind rein textbasiert, d.h. der Benutzer gibt ein Programmfragment in ein Formularfenster ein, dieses wird auf dem Server ausgeführt und dem Client bleibt nur die Aufgabe,

den Text anzuzeigen. Im Gegensatz dazu werden bei *JOSH-online* Programmfragmente tatsächlich auf dem Client ausgeführt, und der Programmzustand wird erhalten. Somit können Übungen aufeinander aufbauen und dabei auch APIs, wie zum Beispiel das *Abstract Windowing Toolkit*, verwenden:

```
> java.awt.Frame f
   = new java.awt.Frame("Mein erstes Fenster");
>>
> f.setSize(300,100);
>>
> f.setVisible(true);
>>
> f.getGraphics().drawString("Hello World",20,70);
>>
```

Abbildung 4: Screenshot: Vom Interpreter aus erzeugtes Fenster

5 Zusammenfassung

Es gibt viele Gründe, die für Java als erste Programmiersprache an Schulen und an Universitäten sprechen, dazu gehören die hohe Gegenwartsbedeutung und die damit verbundene Motivation, sowie die Verfügbarkeit vielfältiger Programmierbibliotheken. Allerdings wird der Einstieg durch die Tatsache, dass man vollständige Klassen definieren muss, erschwert. Java-Interpreter im Stile von *JOSH* schaffen hier Abhilfe, indem sie es erlauben, Programmfragmente direkt auszuführen.
Durch die Intergration eines solchen Interpreters in ein Programmiersprachentutorial ist es möglich, Beispiele direkt im Lehrtext auszuprobieren und Programmieraufgaben zu stellen, die direkt im Fenster des Webbrowsers bearbeitet werden können. Darüberhinaus werden die Lernenden ermutigt, eigene Fragestellungen durch eigene Programmierexperimente zu beantworten.

Literatur

[BBMG02] Bull, G., Bell, R., Mason, C., und Garofalo, J.: *Handbook of Information Technologies for Education and Training*. Springer Verlag. 2002. Elementary/Secondary Education.

[Bi03] Bieg, C.: Ein Server-basierter Interpreter für ein Online-Tirorial über Programmieren in Java. Master's thesis. Universität des Saarlandes, Fachbereich Informatik. Saarbrücken. 2003.

[Bl] Blelloch, G. An interactive tutorial for NESL. `http://www-2.cs.cmu.edu/~scandal/nesl.html`.

[Bl98] Blumstengel, A.: *Entwicklung hypermedialer Systeme*. Wissenschaftlicher Verlag. Berlin. 1998.

[Br66] Bruner, J. S. Toward a Theory of Instruction. 1966.

[DB03a] Diehl, S. und Bieg, C.: A new Approach for Implementing stand-alone and web-based Interpreters for Java. PPPJ'03. 2003.

[DB03b] Diehl, S. und Bieg, C. JOSH Homepage. `http://www.cs.uni-sb.de/~diehl/JOSH`. 2003.

[DG01] Diehl, S. und Görg, C.: Veranstaltungsskript: Software und Programming II. *im Rahmen der IT-Qualifizierung*. 2001.

[EG03] ELM-Gruppe. Episodic Learner Model - The Adaptive Remote Tutor. `http://art2.ph-freiburg.de/Lisp-Kurs`. 2003.

[Hi] Hillion, S. DynamicJava. `http://koala.ilog.fr/djava/`.

[Ko84] Kolb, D.: *Experiential Learning*. Prentice-Hall. 1984.

[Ni] Niemeyer, P. BeanShell. `http://www.beanshell.org/` .

[Ro01] Roberts, E.: An overview of MiniJava. ACM SIGCSE. 2001.

[RU03] RUBY.CHannel. Ruby Tutorial. `http://www.ruby.ch/tutorial` . 2003.