JOP
ein Java Online Praktikum der vhb

G. Fischer, J. Wolff v. Gudenberg
{fischer,wolff}@informatik.uni-wuerzburg.de

Zusammenfassung

In diesem Artikel stellen wir kurz das von der vhb (Virtuelle Hochschule Bayern) geförderte und an den Universitäten Passau (Prof. Dr. G. Snelting) und Würzburg (Prof. Dr. J. Wolff v. Gudenberg) durchgeführte Projekt *Java Online Praktikum* (JOP) vor. Details zu Tutorial und Praktomat finden sich auf der Homepage[1].

Schlüsselwörter: Interaktive Lernumgebung für Java, Webbasierte Programmierausbildung, automatische Beurteilung von Java-Programmen.

1 Einleitung

Das Java Online Praktikum (JOP) bietet Programmieranfängern eine interaktive, automatisierte Plattform zum Erlernen der objektorientierten Programmierung mit Java. JOP besteht aus zwei Teilen:

Im Tutorial wird das objektorientierte Programmieren mit Java vermittelt, wobei die Studenten Kontrollaufgaben des Tutorials bearbeiten. JOP geht dabei jedoch weit über herkömmliche Multiple-Choice-Tests hinaus: Die eingereichten Java-Programme werden durch moderne Verfahren der Programmanalyse („elektronische Tutoren“) automatisch nach Funktionalität und Form bewertet. Studenten erhalten so direkte Rückmeldung über ihren Lern- und Programmiererfolg. Darüber besteht die Möglichkeit die im Tutorial verwendeten Beispiele direkt auszuprobieren, und dabei sogar zu verändern.

Im anschließenden Praktikum werden größere Java-Programme erstellt; erneut sichern die elektronischen Tutoren die Eingangsqualität der Programme. Die abschließende Bewertung erfolgt durch Mitarbeiter, unterstützt durch die elektronischen Tutoren.

2 Zielgruppe und didaktisches Konzept

JOP will den Lernenden durch viele Beispiele und Aufgaben dazu anregen, selbst Programme zu verändern und neu zu erstellen. Die Beispiele sind als Ausschnitte vollständiger Programme in den begleitenden Text integriert und können auf Knopfdruck übersetzt und direkt und interaktiv ausgeführt werden. Sie können aber auch mit Hilfe eines Editors verändert werden. Über diesen Zugang und durch einfache Programmieraufgaben, die ebenfalls innerhalb des Tutorials gestellt werden, wird der Lernende schnell in die Lage versetzt eigene, vollständige Programme zu schreiben und damit das Lernziel eines Praktikums zu erreichen.

Über einführenden Text und Betrachtung von Beispielen wird so die hauptsächliche Lernmethode *Learning by doing* verwirklicht.

Es wird bewusst auf den Einsatz von grafischen Programmierumgebungen verzichtet. Im Vordergrund steht das „handwerkliche“ Programmieren mit Editor und Compileraufruf. Der Lernende soll jeden Schritt verstehen und nachvollziehen können.

Wichtig und ein Hauptanliegen von JOP ist auch die Qualität der Java Quelltexte. Neben der Funktionalität werden auch die Einhaltung einiger weit verbreiteter Programmierkonventionen von elektronischen Tutoren überprüft. Beispielsweise das Vorhandensein von Kommentaren, die Wahl von Bezeichnern und die Formatierung. Erste Erfahrungen haben bestätigt, dass der erstellte Quellcode durch diese Maßnahmen einerseits sehr gut lesbar wird und andererseits auch an Qualität gewinnt. Das anschließende gegenseitige Kommentieren wird so erleichtert und die Kommentare der Kommilitonen wie auch der Korrektoren können sich auf inhaltliche Belange konzentrieren.

Ein weiterer Aspekt besteht in dem schnellen und unkomplizierten Zugriff zur Dokumentation der Java Standardpakete. Vom ersten Programm an wird der Lernende auf die umfassende Bibliothek hingewiesen.

Zielgruppe sind üblicherweise Studenten zu Beginn des Grundstudiums mit keiner oder wenig Programmiererfahrung. Um innerhalb dieser Zielgruppe auch noch auf Unterschiede eingehen zu können, existieren unterschiedliche Lernpfade und Aufgaben für z.B. Studenten der Informatik oder solche anderer Fachrichtungen. Ein weiterer Lernpfad, der imperatives Programmieren voraussetzt, öffnet den Kurs auch für die Weiterbildung. Viele Vorlesungen, z.B. Anfängervorlesungen über Algorithmen und Datenstrukturen, benötigen einen Ausschnitt aus Java, ein entsprechender Lernpfad kann durch den modularen Aufbau leicht hergestellt werden.

3 Tutorial Inhalt

Das Tutorial stellt den Sprachumfang von Java mit Ausnahme von Threads in knapper Form dar.

Das Tutorial ist ein in XML geschriebener Hypertext, der als eine vernetzte Struktur von Konzepten dargeboten wird. UML Klassendiagramme illustrieren die Klassen- bzw. Objektstruktur, sind aber selbst nicht Lerninhalt. Außerdem gehören dazu: ein Glossar, eine Syntax-Zusammenfassung, Verweise auf die API Dokumentation und eine Funktion für den direkten Zugang dorthin. Die Navigation ist grundsätzlich frei, wird jedoch durch voreingestellte Lernpfade unterstützt.

Der Standard Lernpfad enthält folgende Kapitel:

- **Einführung** in den objektorientierten Ansatz, die Sprache Java und das Entwicklungssystem
- **Anfänge** – elementare Datentypen und Standardklassen, Ausdrücke und Zuweisungen
- **Algorithmen und Anweisungen** – imperative Programmierung
- **OOP** – Vereinbarung von Klassen
- **Programmieren durch Vertrag** – Implementierung abstrakter Datentypen, Interfaces, Pakete, Listen und Iteratoren
- **Vererbung** – Generalisierung, Spezialisierung, Hierchiebildung, Polymorphie
- **Ausnahmen**
- **Ein- und Ausgabe** – Datenströme, Serialisierung
- **Grafische Benutzeroberflächen** – AWT Komponenten, Ereignisse, Beobachter
- **Applets**
- **Anhang** – Erste Hilfe, Dokumentation zur bereitgestellten Bibliothek Simple, Information zur interaktiven Ausführung mit JSEx, die Java Syntax als Syntaxdiagramme, und die verwendeten Quelltextkonventionen.

Das Tutorial umfasst dabei mehr als 100 (vollständige) Beispiele sowie über 50 Aufgaben, davon mehr als 20 Programmieraufgaben.

4 Praktomat

Der an der Universität Passau vorhandene Praktomat [5, 4] übernimmt die Annahme, Verwaltung, Speicherung, elektronische Überprüfung und unterstützt die abschließende Beurteilung der Teilnehmerprogramme.

Insbesondere gehören zu den Aufgaben:

- Teilnehmer- und Lösungsverwaltung,
- Aufgabenverwaltung mit Varianten
- Vorbeurteilung der Lösungen
- Unterstützung der Korrektoren bei der abschließenden Beurteilung
- Plagiatsprüfung
- Gegenseitiges Kommentieren von Benutzerlösungen

Dabei wird die Beurteilung der Lösungen von sogenannten *elektronischen Tutoren* nach bestimmten Kriterien durchgeführt.

Für das JOP-Projekt wurden für den Praktomaten spezielle Java-Tutoren entwickelt, die eine tiefergehende Beurteilung der Programme zulassen:

- **Stilprüfung**: Prüfung des Programmstils mit dem Programm Checkstyle[1]
- **Formatierungsprüfung**: Prüfung der Quellcode Formatierung
- **Kommentarprüfung**: Prüfung der Kommentarkonventionen
- **Strukturprüfung**: Prüfung auf Verwendung bzw. nicht Verwendung bestimmter Klassen
- **Gray-Box-Test**: Prüfung der Funktionalität anhand einer Musterlösung

Die Tutoren sind pro Aufgabe konfigurierbar und auch das Bestehen kann als optional markiert werden, so dass mögliche Fehlermeldungen rein informativ sind.

5 Tutorial Aufbau

Innerhalb des Tutorials können zwei grundlegende Teile unterschieden werden, die Inhalte und die Präsentation.

Beide Teile werden vom speziell zu diesem Zweck entwickelten *CMPS*[2] verwaltet. CMPS ist ein *Cocoon*[2] basiertes Framework zur Erstellung, Verwaltung und Darstellung von XML-basierten Lerninhalten. Es bietet über die Präsentation der Inhalte hinaus u.a. auch Funktionen für:

- Webbasiertes Benutzerinteface zur Bearbeitung und Verwaltung der Inhalte
- Automatische Versionsverwaltung
- Automatische Syntaxprüfung einzelner Elemente (Wohlgeformtheit / Gültigkeit für XML, Kompilierbarkeit für Java-Quellcode, ...)

[1] http://checkstyle.sourceforge.net/

[2] Das XML Publishing Framework *Cocoon*: http://xml.apache.org/cocoon/

- Automatische Konsistenzprüfung des gesamten Tutorials (Prüfung von internen Links, Existenz verwendeter Resourcen, ...)

- Automatische Überdeckungsanalyse für Lernpfade.

Wir wollen zunächst auf die Struktur der Inhalte, und anschließend auf deren Präsentation eingehen. Danach wird noch auf die Realisierung der dynamischen Elemente, die ja einen entscheidenden Punkt bei der Umsetzung des *Learning by doing* darstellen, eingegangen.

5.1 Inhaltsstruktur

5.1.1 Konzepte

Die Inhalte des Tutorials gliedern sich auf oberster Ebene in sogenannte *Konzepte*, die eine Lerneinheit bilden.

Ein Konzept besteht intern aus *Fragmenten*, die zu verschiedenen *Sichten* angeordnet werden können. Mögliche Sichten auf Konzepte sind z.B. ein Einstieg, ein Überblick, eine komplette Abhandlung, oder eine Referenz.

Die Fragmente stellen den eigentlichen „textuellen“ Inhalt des Tutorials dar. Sie liegen in XML, entsprechend einer eigens dafür erstellten DTD, die jedoch sehr allgemein gehalten und daher ohne Schwierigkeiten an andere Themengebiete anpassbar ist, vor.

Zu erwähnen ist noch, dass sowohl Fragmente als auch Sichten auf ein Konzept automatisch erzeugt werden können. Beispiele für automatische Fragmente sind z.B. eine automatisch erzeugte Verweisliste, oder eine Zusammenfassung von Beispielen, Aufgaben oder Definitionen. Automatische Sichten können sich dann auch insbesondere auf solche automatischen Fragmente beziehen, so dass z.B. eine Ansicht mit allen Aufgaben eines Konzepts vollautomatisch erstellt werden kann.

Die Konzepte mit ihren zugehörigen Fragmenten und Definitionen von Views sind zu einer XML-Einheit zusammengefasst, da sie gemeinsam den Inhalt ausmachen.

Eine Alternative wäre gewesen, eine Sicht auf ein Konzept aus externen Fragmenten aufzubauen, oder ganz auf die Konzepte zu verzichten und den Lernpfad aus Fragmenten zu bilden. Durch die räumliche Trennung von Daten und Metadaten hätte diese Vorgehensweise die Komplexität und Fehleranfälligkeit der Datenhaltung erhöht.

Kommt es doch einmal zu einer Überschneidung von Inhalten zwischen Konzepten, so können, um das *Single Source Prinzip* zu wahren, *externe Fragmente* verwendet werden. Diese können dann von mehreren Konzepten eingebunden werden. Damit ließe sich auch die vollständige Trennung von Daten und Metadaten realisieren.

5.1.2 Lernpfade

Prinzipiell sind die Konzepte in sich geschlossene Lerneinheiten, die (mit dem entsprechenden Vorwissen) für sich gelernt werden wollen. Daher kann man sich grundsätzlich frei im Tutorial bewegen. Allerdings bevorzugen viele eine mehr geführte Lernweise, die durch *Lernpfade* ermöglicht wird.

Ein Lernpfad definiert einen Weg durch das Tutorial, indem bestimmte Sichten bestimmter Konzepte zusammengesetzt werden. Auf diese Weise lassen sich sehr flexibel auf bestimmte Zielgruppen angepasste Lernpfade erstellen. In JOP gibt es z.B. spezielle Pfade für Einsteiger (die noch keine Programmierkenntnisse haben), Umsteiger (die nur die objektorientierte Programmierung erlernen wollen) und Nebenfächler (die zuweilen eine etwas vereinfachte Sicht bevorzugen).

5.2 Präsentation

Die Präsentation des Tutorials erfolgt auch mit Hilfe von CMPS. Die Inhalte werden dabei innerhalb von Pipelines erstellt und verarbeitet.

Zunächst werden die Inhalte generiert. Das kann statisch, z.B. aus XML-Dateien oder mit speziellen Generatoren auch aus z.B. Java-Quelltext[3] geschehen, oder dynamisch, z.B. mit Hilfe von XSPs[3]. Die Inhalte lassen sich dabei natürlich aus anderen Teilen zusammensetzen. So werden die Sichten der Konzepte aus den Fragmenten zusammengesetzt, und die Lernpfade aus den Sichten der Konzepte.

Die generierten Inhalte werden dann mit Hilfe von Transformatoren, z.B. XSLT in das entsprechende gewünschte Zielformat gewandelt und anschließend ausgegeben. Dabei können mit dem Tutorial unterschiedliche Zielformate wie HTML aber auch PDF generiert werden.

Die Erzeugung geschieht dabei in zwei Schritten. Zunächst wird ein Zwischenformat aus den Konzepten erzeugt.

- Generierung der Inhalte aus den Quellen
- Einbau von externen und automatischen Fragmenten.
- Auswahl der zur Ansicht gehörigen Fragmente
- Automatisches Markieren von Glossareinträgen
- Einbettung von Java Quelltext

Für die Darstellung kompletter Lernpfade wird dann aus dem Zwischenergebnis der Transformation für die einzelnen Konzepte ein gesamtes Dokument erzeugt. Das Zwischenformat wird auch von den dynamischen Komponenten und aus dem Glossar erzeugt.

Basierend auf diesem Zwischenformat kann dann, nach Realisierung der Navigation, Layouting, Theming, Hinzufügen von Benutzerinformation und Realisierung der Links HTML erzeugt werden.

[3] XML Server Pages

Weiterhin kann über XSL-FO[4] als Formatierungsstufe PDF erzeugt werden.

5.3 Dynamische Elemente

Über die Präsentation statischer Inhalte hinaus bietet das Tutorial auch eine Reihe dynamischer Elemente an, die das *Learning by doing* erst ermöglichen. Das sind zum einen die interaktiven Beispiele, zum anderen die direkt ausgewerteten Aufgaben.

5.3.1 Beispiele

Alle Beispiele des Tutorials sind nicht nur Programmfragmente sondern immer vollständige Programme. Diese Programme lassen sich vom Benutzer verändern und dann mit Hilfe des speziell zu diesem Zweck entwickelten Programms JSEx[5] ausführen. Damit können die Beispiele direkt – ohne mühsames herunterladen, compilieren und mit den richtigen Hilfsdateien in der richtigen Umgebung ausführen – ausprobiert werden. Das gilt insbesondere nicht nur für die vorgegebene Version der Beispiele, sondern auch für vom Benutzer abgeänderte Varianten. Der Benutzer kann so Beispiele mit minimalen Aufwand ausprobieren und sogar Auswirkungen eigener Modifikationen betrachten.

JSEx ist ein auf *Java Web Start*[6] basierendes Programm, dass die Ausführung von anderen Java Programmen in einer sicheren Umgebung gestattet. Es simuliert eine Eingabeaufforderung und stellt Ausgaben der Programme dar. Auch Kommandozeilenparameter lassen sich angeben. Darüber hinaus können natürlich nicht nur kommandozeilenbasierte, sondern auch Programme mit graphischer Benutzeroberfläche ausgeführt werden. Damit ist es geeignet alle Beispielprogramme des Tutorials auszuführen.

Natürlich steht es dem Benutzer auch frei den Quellcode herunter zu laden und auf „herkömmliche“ Weise zu verwenden.

5.3.2 Aufgaben

Ein weiteres interaktives Element des Tutorials sind die eingebetteten Aufgaben. Es gibt verschiedene Aufgaben:

- Single bzw. Multiple Choice Aufgaben,
- Freitext Aufgaben und
- Programmieraufgaben.

Die Prüfung der Lösungen erfolgt sofort durch die entsprechendenden Evaluatoren. Die Programmieraufgaben werden von dem Evaluator an den Praktomaten durchgereicht, der die Überprüfung entsprechend der vorgegebenen Einstellungen für jede Aufgabe individuell durchführt.

[4] XSL Formatting Objects: http://www.w3.org/TR/xsl/

[5] Java Secure EXecution: http://www2.informatik.uni-wuerzburg.de/staff/fischer/research/jsex/

[6] http://java.sun.com/products/javawebstart/

6 Praktomat Aufbau

Der Praktomat ist ein in Python realisiertes CGI-Programm, dass die Durchführung eines Programmierpraktikums (teilweise) automatisiert und überwacht. Die zugehörigen Daten werden dabei in einer SQL Datenbank gehalten [4].

Zur Anbindung des Tutorials an den Praktomaten wurde eine Schnittstelle geschaffen, die es erlaubt die Lösungen unter Umgehung eines Web-Servers an den Praktomaten zu geben und dessen Bewertung zu erhalten. Diese Schnittstelle wird verwendet, um Lösungen zu den Programmieraufgaben des Tutorials zu bewerten.

Literatur

[1] JOP; `http://jop.informatik.uni-wuerzburg.de/`

[2] G. Fischer: *Ein XML basiertes Content Management und Publishing System (CMPS) zur dynamischen Informationspräsentation (am Beispiel eines Java Online Tutorials)*, Diplomarbeit, Lehrstuhl für Informatik II, Universität Würzburg, 2001; `http://www-info2.informatik.uni-wuerzburg.de/staff/fischer/research/cmps/`

[3] G. Fischer: *JaML, eine XML-basierte Auszeichnungssprache für Java Source Code*; `http://www2-info2.informatik.uni-wuerzburg.de/staff/fischer/research/jaml/`

[4] J. Krinke, M. Störzer, A. Zeller: Webbasierte Programmierpraktika mit Praktomat, Softwaretechnik-Trends 22:3, S.51-53, 2002

[5] A. Zeller: *Funktionell und verständlich programmieren - so lernen es die Passauer*, Softwaretechnik-Trends 19:3, Nov. 1999