

Programmierkurse für Anfänger und Fortgeschrittene

Dietrich Boles

Universität Oldenburg, Department für Informatik
Escherweg 2, D-26121 Oldenburg
boles@informatik.uni-oldenburg.de

Zusammenfassung

Sicher kennen viele Veranstalter von Programmierkursen in Erstsemesterveranstaltungen an Hochschulen das Problem, dass einige Studierende völlige Programmieranfänger sind, andere Studierende jedoch aufgrund von schulischen Veranstaltungen oder aus Eigeninitiative bereits eine Menge Programmiererfahrung besitzen. Wie können solche Programmierkurse durchgeführt werden, ohne dass auf der einen Seite Programmieranfänger überfordert werden und auf der anderen Seite sich die fortgeschrittenen Programmierer gelangweilt fühlen? In diesem Artikel wird ein Lösungsansatz skizziert, der seit nunmehr sieben Jahren erfolgreich an der Universität Oldenburg praktiziert wird.

1 Einleitung

Der „Programmierkurs Java“ ist eine regelmäßig im Wintersemester stattfindende Pflichtveranstaltung für Erstsemester der Diplom- und Bachelor-Studiengänge Informatik an der Universität Oldenburg. Weiterhin nehmen auch einige Nebenfächler aus anderen Studiengängen daran teil. Die Teilnehmerzahl liegt zwischen 200 und 300 Studierenden. Bezogen auf den Kenntnisstand reicht die Spanne der Teilnehmenden dabei von absoluten Programmieranfängern (im letzten Jahr ca. 30 Prozent) bis hin zu Siegern im Bundeswettbewerb Informatik.

Der Programmierkurs Java ist inhaltlich in drei Teile aufgeteilt. Im ersten Teil werden die Grundlagen der Programmierung (Algorithmen, Sprachen, Compiler, Aussagenlogik, ...) eingeführt. Im zweiten Teil werden imperative Sprachkonzepte (Variablen, Ausdrücke, Anweisungen, Funktionen, Rekursion, ...) behandelt. Der dritte Teil geht auf die Grundkonzepte der objektorientierten Programmierung (Klassen, Objekte, Vererbung, Polymorphie, Pakete, ...) ein. Die Lehrinhalte werden in insgesamt 15 zweistündigen Vorlesungen vermittelt. In den Übungen wird durch praktische Programmieraufgaben der Stoff vertieft.

Ein wesentliches Ziel des Programmierkurses ist es, Programmieranfängern die Grundkonzepte der Programmierung zu vermitteln und die Studierenden, die bereits Programmierkenntnisse besitzen, von einem sauberen, strukturierten Programmwurf und Pro-

grammierstil zu überzeugen, denn leider sind die Kenntnisse diesbezüglich häufig mangelhaft. Der Kenntnisstand der beiden Gruppen soll durch den Programmierkurs möglichst angeglichen werden. Dazu muss der Programmierkurs jedoch auf eine Art und Weise gestaltet werden, dass die Anfänger nicht überfordert und die Könnner nicht unterfordert werden und irgendwann frustriert fortbleiben.

Der Lösungsansatz, der diesbezüglich in Oldenburg gewählt wurde, sieht folgendermaßen aus: Begleitend wird zum eigentlichen Programmierkurs das Hamster-Modell eingesetzt, das mit spielerischen Elementen in die Grundlagen der Programmierung einführt und insbesondere Anfängern aber auch Fortgeschrittenen Spaß am Erlernen der Programmierung macht. In die Übungen wird die inkrementelle Entwicklung von Spielprogrammen integriert, die insbesondere die fortgeschrittenen Programmierer herausfordert, aber so gestaltet wird, dass sie auch von Anfängern geleistet werden kann.

2 Das Hamster-Modell

Programmieranfänger leiden oft darunter, dass sie beim Programmieren ihre normale Gedankenwelt verlassen und in eher technisch-orientierten Kategorien denken müssen, die ihnen von den Programmiersprachen vorgegeben werden. Gerade am Anfang strömen häufig so viele Neuigkeiten inhaltlicher und methodischer Art auf sie ein, dass sie leicht das Wesentliche der Programmierung, nämlich das Lösen von Problemen, aus den Augen verlieren und sich in syntaktischen und technischen Einzelheiten verirren.

Das Hamster-Modell ist mit dem Ziel entwickelt worden, dieses Problem zu lösen. Mit dem Hamster-Modell wird Programmieranfängern ein einfaches aber mächtiges Modell zur Verfügung gestellt, mit dessen Hilfe Grundkonzepte der imperativen Programmierung auf spielerische Art und Weise erlernt werden können. Programmierer entwickeln so genannte *Hamster-Programme*, in denen sie einen virtuellen Hamster durch eine virtuelle Landschaft steuern und bestimmte Aufgaben lösen lassen.

2.1 Beschreibung

Das Hamster-Territorium – die Welt, in der ein Hamster-Programmierer operiert – wird durch eine gekachelte Ebene repräsentiert. Auf den Kacheln können Weizenkörner liegen. Kacheln können auch durch Mauern blockiert sein. Auf einer Kachel steht der Hamster, der eine der Blickrichtungen Nord, Ost, Süd oder West einnimmt und prinzipiell beliebig viele Körner in seinen Backen aufbewahren kann. Der Hamster besitzt die Fähigkeit, Befehle auszuführen. Er kennt die vier Befehle `vor()`; (gehe eine Kachel in Blickrichtung nach vorne), `linksUm()`; (drehe dich um 90 Grad nach links), `nimm()`; (nimm da, wo du stehst, ein Korn auf) und `gib()`; (lege da, wo du stehst, ein Korn ab) sowie die drei Testbefehle `vornFrei()` (befindet sich vor dem Hamster eine Kachel mit einer Mauer), `kornDa()` (liegt auf der Kachel, auf der sich der Hamster gerade befindet, mindestens ein Korn) und `maulLeer()` (hat der Hamster Körner im Maul). Den Lernenden werden Hamster-Aufgaben gestellt, die sie mit Hilfe dieser Grundbefehle und inkrementell eingeführten Programmierkonstrukten lösen sollen. Beispiel: In einem rechteckigen ge-

schlossenen Raum unbekannter Größe ohne innere Mauern sind wahllos eine unbekannte Anzahl an Körnern verstreut. Der Hamster, der sich zu Anfang mit Blickrichtung Ost in der linken unteren Ecke befindet, soll alle Körner aufsammeln und dann anhalten.

Das Hamster-Modell wurde in einer einfachen Version zu Beginn der 80er Jahre in der GMD (Gesellschaft für Mathematik und Datenverarbeitung) entwickelt. Zielsprache war damals die imperative Programmiersprache ELAN. In dem Buch „Programmieren spielend gelernt mit dem Java-Hamster-Modell“ wird das Hamster-Modell an die Programmiersprache Java angepasst [Bol02]. Zum Entwerfen, Editieren, Compilieren und Testen von Hamster-Programmen steht eine Mini-Programmierungsumgebung, der so genannte *Hamster-Simulator*, zur Verfügung. Weitere Informationen finden sich unter <http://www-is.informatik.uni-oldenburg.de/~dibo/hamster>.

Andere, dem Hamster-Modell verwandte Modelle sind zum Beispiel „Karel der Roboter“ [BSRP97] oder „Kara der Marienkäfer“ [RNH00]. Sie sind sicher ähnlich gut geeignet. Ein Vorteil des Java-Hamsters ist jedoch, dass ein Präprozessor eingesetzt wird, durch den für die imperative Programmierung zunächst überflüssige (und Programmieranfänger leicht verwirrende) Sprachelemente von Java, wie import-Anweisungen und die Klassendefinition, verborgen werden.

2.2 Erfahrungen

Die Erfahrungen mit dem Hamster-Modell sind durchweg außerordentlich positiv. Die wesentlichen Prinzipien und Konzepte der imperativen Programmierung lassen sich mit dem Ansatz hervorragend vermitteln. Ein noch höherer Wert kommt dem Hamster-Modell in Verbindung mit dem Hamster-Simulator jedoch hinsichtlich des praktischen Übens der Programmentwicklung zu. Die spielerischen Elemente wirken insbesondere für Programmieranfänger stark motivierend. Es ist für sie sehr leicht, sich neben vom Veranstalter gestellten Hamster-Aufgaben immer wieder selbst neue Hamster-Übungsaufgaben auszudenken und sie zu bearbeiten.

Dass vom Hamster-Modell eine starke Motivationswirkung ausgeht, zeigt sich auch darin, dass sich die Studierenden über die Programmierung hinaus mit ihm beschäftigen. So sind durch Umbenennung der Grundbefehle und Austausch der Icons des Hamster-Simulators eine Osterhasen- und Weihnachtsmannvariante entstanden, es wurden lustige Hamster-Gedichte geschrieben und der von einem Studierenden entwickelte „Hamster-Killer“, ein Spielprogramm in Anlehnung an „Moorhuhn“, erlaubt es den Studierenden, „sich am Hamster zu rächen, wenn er mal wieder nicht das tun will, was er eigentlich tun soll“.

3 Entwicklung von Spielprogrammen

Die Implementierung von Programmen, die Schach, Reversi, Kalah oder ähnliche Zwei-Personen-Strategiespiele spielen können, birgt ein enormes Motivationspotential bei den Studierenden in sich (vergleiche auch [HMBH99]). Insbesondere liegt das auch daran, dass in Oldenburg am Ende des Semesters das schon traditionelle Spielturnier durchge-

führt wird, bei dem die entwickelten Programme gegeneinander antreten und den Sieger ermitteln. Neben einem Pokal bekommen die Entwickler des Siegerprogramms Bonuspunkte für die Klausur. Auch Anfängern wird im Programmierkurs die Chance gegeben, eigene Spielprogramme zu entwickeln, und zwar dadurch, dass die Übungsblätter ausführliche Anleitungen enthalten und Beispiele zur Verfügung gestellt werden.

3.1 Vorgehensweise

Die Spielprogrammentwicklung geschieht dabei inkrementell im Rahmen der Übungen. In einem ersten Schritt müssen die Studierenden ein Programm entwickeln, das es zwei menschlichen Spielern ermöglicht, am Rechner gegeneinander das ausgewählte Spiel zu spielen. Die Entwicklung des Programms basiert dabei auf objektorientierten Methoden, wobei ein generelles Gerüst (Klassen, Methoden, ...) vorgegeben wird.

In einem zweiten Schritt wird den Studierenden ein „Schiedsrichterprogramm“ zur Verfügung gestellt, mit dem Menschen und/oder Programme gegeneinander antreten können. Die Studierenden müssen ihr Programm so umgestalten, dass die vom Schiedsrichterprogramm vorgegebenen Schnittstellen eingehalten werden. Das Schiedsrichterprogramm basiert auf folgendem Prinzip: Beide Spielerprogramme sowie das Schiedsrichterprogramm verwalten jeweils ein eigenes Spielbrett. Die Kommunikation erfolgt durch den Austausch von Spielzügen, und zwar über Objekte einer bereitgestellten Klasse `Spielzug`. Die Spielerprogramme müssen von der Klasse `Spieler` abgeleitet sein und die Methode `Spielzug.liefereNaechstenSpielzug(Spielzug gegnerSpielzug)` überschreiben. In dieser Methode muss ein Spielerprogramm zunächst den übergebenen Gegnerspielzug auf dem eigenen Spielbrett ausführen und dann einen eigenen Spielzug berechnen, ihn ausführen und zurück liefern. Die Methode wird vom Schiedsrichterprogramm abwechselnd für beide Spielerprogramme aufgerufen. Zwischendurch kontrolliert das Schiedsrichterprogramm jeweils den neuen Spielzug und überprüft, ob das Ende des Spiels erreicht ist.

Die nächsten Schritte bei der Entwicklung der Spielerprogramme sind die Realisierung eines Stellungsbewerters und die Implementierung des Aufbaus eines Spielbaums. Im letzten Schritt erfolgt die Integration einer Zeitkomponente, denn beim Spieleturnier müssen die Programme vorgegebene Zeitbeschränkungen beachten.

3.2 Erfahrungen

Neben dem beim überwiegenden Teil der Studierenden festzustellenden motivierenden Charakter kommt der Entwicklung eigener Spielprogramme auch unter didaktischen und methodischen Aspekten eine besondere Bedeutung zu. Der oben beschriebene Ansatz ist hervorragend dazu geeignet, einen ersten Einblick in die objektorientierte Softwareentwicklung zu geben, die in Oldenburg dann in der Vorlesung „Software Engineering“ im zweiten Semester vertieft wird. Der gewählte Ansatz fördert ganz wesentlich das Verständnis und den praktischen Umgang mit objektorientierten Programmierkonzepten, vor allem dem beim Aufruf der Methode `liefereNaechsterSpielzug` verwendeten Konzept der Polymorphie, mit dem viele Studierende anfangs Probleme haben.

Hervorzuheben ist auch der Wille einiger meist bereits erfahrener Programmierer, sich in alternative Lösungsansätze für die Entwicklung von Spielprogrammen einzuarbeiten. So sind neben Programmen mit hochgradig optimierten Spielbäumen bspw. Programme entstanden, die auf neuronalen Netzen oder im Vorfeld des Spieleturniers aufgebauten großen Datenbanken mit komplett abgespeicherten und analysierten Spielen basieren.

Nicht verschwiegen werden sollen diverse Versuche, das Schiedsrichterprogramm zu überlisten, vor allem durch illegale Manipulation der Zeitmessung.¹ Diesbezügliche Ansätze reichten vom heimlichen Start von Threads am Ende der Methode `liefernAechstenSpielzug`, die Zeit des Gegnerprogramms verbrauchen sollten, über das Überschreiben von Methoden der Klasse `Spielzug`, die versehentlich nicht als `final` deklariert worden war, bis hin zum Entdecken und Ausnutzen eines Fehlers der Java Virtual Machine, der den Zugriff auf `private`-Attribute des Schiedsrichterprogramms ermöglichte.

4 Fazit

Wir haben an der Universität Oldenburg mit dem in diesem Artikel skizzierten Aufbau des Programmierkurses Java sehr gute Erfahrungen gemacht. Das Hamster-Modell vermittelt mit spielerischen Elementen die Grundlagen der Programmierung und motiviert Programmieranfänger als auch fortgeschrittene Programmierer zu einem sauberen, strukturierten Programmmentwurf. Die Entwicklung von Spielprogrammen fördert in besonderem Maße das Verständnis grundlegender Konzepte der Softwareentwicklung. Der gewählte Ansatz unterstützt damit die Integration von Programmieranfängern und fortgeschrittenen Programmierern in einer einzigen Veranstaltung, ohne dass die eine Gruppe über- und die andere Gruppe unterfordert wird.

Weitere Informationen können der Homepage des Programmierkurs Java entnommen werden: <http://www-is.informatik.uni-oldenburg.de/~dibo/teaching/java>.

Literatur

- [Bol02] D. Boles. *Programmieren spielend gelernt mit dem Java-Hamster-Modell*. Teubner-Verlag, 2. Auflage, 2002.
- [BSRP97] J. Bergin, M. Stehlik, J. Roberts, and R. Pattis. *Karel++: A Gentle Introduction to the Art of Object-Oriented Programming*. Wiley, 1997.
- [HMBH99] M. Hirt, D. Matter, R. Bänziger, and W. Hartmann. Gruppenunterricht zum Thema Spieltheorie. Technischer Bericht, ETH Zürich, März 1999. <http://www.educeth.ch/informatik/puzzles/spiel/docs/spieltheorie.pdf>.
- [RNH00] R. Reichert, J. Nievergelt, and W. Hartmann. Ein spielerischer Einstieg in die Programmiersprache Java. *Informatik-Spektrum, Springer Verlag*, 23(5):309–315, 2000.

¹Aber auch hierin liegt ja ein Lerneffekt!