

Studienseminar für Lehrämter an Schulen Hamm

Schriftliche Hausarbeit im Rahmen der Zweiten Staatsprüfung für das Lehramt an
Gymnasien und Gesamtschulen im Fach Informatik

Entwicklung einer handlungsorientierten Einführung in die
objektorientierte Analyse und Modellierung im
Anfangsunterricht der Jahrgangsstufe 11 durch
Objekt-Rollenspiele.

vorgelegt von:

Torsten Schultz

Holbeinstr. 3

59423 Unna

Jahrgang 07/09

storstenei@semsek2.ham.nw.schule.de

Erstgutachter: Herr Dr. Ludger Humbert

Unna, den 20. Mai 2008

Inhaltsverzeichnis

1	Problem der pädagogischen Praxis und Anliegen dieser Arbeit	4
2	Verschiedene Herangehensweisen im Anfangsunterricht der Jahrgangsstufe 11	6
2.1	Klassenbegriff vor Objektbegriff	6
2.2	Programmierungsumgebungen ohne Klassenbegriff	7
2.3	strictley objects and modells first	8
3	Begründung der Auswahl dieses Ansatzes	10
4	Konzept für den Anfangsunterricht mit Objektspielen	11
4.1	Literaturlage - bisherige Ansätze zu Objektspielen	11
4.2	Geeignete Problemstellung für eine erste Modellierungsaufgabe	12
4.3	Objektkarten und Objektdiagramme	14
4.4	Das Objekt- Rollenspiel	17
4.4.1	Objektnamen und Beziehungen	17
4.4.2	Schnittstelle und Objektinterna	19
4.4.3	Attribute	19
4.4.4	Nachrichten: Dienste mit Argumenten und Parametern	19
4.4.5	Aktives Objekt	21
4.4.6	Klassen	22
4.4.7	Praktische Umsetzung	22
5	Möglicher Ablauf einer Reihe im Anfangsunterricht	23
5.1	Erste Sequenz - Einführendes Modellierungsbeispiel	24
5.2	Zweite Sequenz - Objekte im Rollenspiel	25
5.3	Dritte Sequenz - Argumente und Parameter	25
5.4	Vierte Sequenz - Objekte als Argumente und Parameter	26
5.5	Fünfte Sequenz - Kontrollstrukturen	26
5.6	Sechste Sequenz - Klassenbegriff	27
6	Grenzen des Konzeptes	28
7	Praktische Erfahrungen aus dem Unterricht	30

Inhaltsverzeichnis

8	Evaluationsmöglichkeiten	31
9	Lehrerfunktionen	32
10	Ausblick, Fazit	33
	Anhang	34
A	Hinweise zur Modellierungsaufgabe	34
B	Aufgabenblatt 1	41
C	Aufgabenblatt 2	42
D	Aufgabenblatt 3	43
E	Aufgabenblatt 4	44
F	Aufgabenblatt 5	45
G	Objektdiagramm »MauMauSpiel«	46
H	Objektkarte »spieler«	47
I	Dienstekarte »spieler«	48
J	Objektkarte »karte«	49
K	Dienstekarte »karte«	50
L	Dienstekarte »aufStapel«	51
M	Dienstekarte »abStapel«	52
N	Dienstekarte »spiel«	53
	Abschließende Erklärung	57

1 Problem der pädagogischen Praxis und Anliegen dieser Arbeit

Die Problemstellung für diese Arbeit hat sich in zwei Schritten ergeben: Im ersten Ausbildungshalbjahr habe ich in einem Informatikkurs in der Stufe 11 hospitiert sowie unter Anleitung unterrichtet. Der Kurslehrer hat in diesem Kurs nach dem Lehrbuch »Informatik mit Java« [Sch05] unterrichtet. Der Objekt- und der Klassenbegriff sind am Anfang des Schuljahres zeitgleich eingeführt worden. (Zu dieser Vorgehensweise s. Kapitel 2.1.) Mir fiel auf, dass vielen Schülern¹ die Abgrenzung zwischen den Begriffen »Objekt« und »Klasse« nicht klar war². Das kann auf das Phänomen der Ähnlichkeitshemmung zurückgeführt werden, hervorgerufen dadurch, dass zwei verwandte und ähnliche Begriffe (»Objekt« und »Klasse«) zeitnah zueinander eingeführt wurden.

Um diesem Problem zu begegnen, habe ich in meinem eigenen Informatikkurs in der Jahrgangsstufe 11, den ich im Rahmen des selbstständigen Unterrichts unterrichtet habe, nach der Maßgabe »strictly objects first« (vgl. [Die07b, Die07a]) unterrichtet. Durch diese Vorgehensweise wurde das beschriebene Problem zwar gelöst, allerdings entstand ein neues Problem: Es ist nach diesem Ansatz und mit der Programmiersprache Java nicht möglich, von Beginn an mit ausführbaren Programmen zu arbeiten. Einzige Aufgabenklasse für die erste Begegnung mit informatischer Modellierung ist deshalb das Erstellen von Objektdiagrammen gewesen. Dies kann natürlich in verschiedenen Sozialformen stattfinden, allerdings ist der Unterricht damit nicht handlungsorientiert³ und kommt einer zentralen Erwartungshaltung meiner Schüler nicht entgegen. Dennoch scheint es für das Verständnis der Schüler sinn-

¹Ich verwende in dieser Arbeit, um eine bessere Lesbarkeit zu gewährleisten, die in der deutschen Sprache einfacher zu bildende männliche Form. Weibliche Personen sind ebenso gemeint.

²Deutlich wurde dies insbesondere an folgender Stelle: Im Kurs ist ein Klassendiagramm betrachtet worden, das eine Beziehung enthält, die zwischen Objekten ein und derselben Klasse (der Klasse Waggon, vgl. [Sch05, S. 100]) besteht. Im Klassendiagramm wird dies durch eine Linie ausgedrückt, die an derselben Klasse beginnt und endet. Im Objektdiagramm (in diesem Kurs wurden sie nicht eingeführt) wären Beziehungen zwischen verschiedenen Objekten sichtbar geworden. Die Schüler formulierten hier etwa: »Ein Waggon kennt sich selbst.«, »Die Klasse Waggon kennt sich selbst.«. Der Unterschied zwischen Objekt- und Klassenbeziehungen war also unklar.

³Unter handlungsorientiertem Unterricht verstehe ich in dieser Arbeit gemäß Meyer (vgl. [Mey88], S. 214) »[...] ein[en] ganzheitliche[...][n] und schüleraktive[...][n] Unterricht, in dem die zwischen dem Lehrer und den Schülern vereinbarten Handlungsprodukte die Organisation des Unterrichtsprozesses leiten, so dass Kopf- und Handarbeit der Schüler in ein ausgewogenes Verhältnis zueinander gebracht werden können.« Die Handlungsprodukte sind in diesem Konzept die

1 Problem der pädagogischen Praxis und Anliegen dieser Arbeit

voll zu sein, den Klassenbegriff zeitlich erst mit einigem Abstand zum Objektbegriff einzuführen.

Ich möchte in dieser Arbeit ein Konzept entwickeln, um dem Problem der mangelnden Handlungsorientierung im Zusammenhang mit der Einführung in die objektorientierte Analyse und Modellierung und dem Ansatz »strictly Objects first« (s. Kapitel 2.3) zu begegnen. Der Anfangsunterricht Informatik in der Jahrgangsstufe 11 soll mit Hilfe von Objektspielen gestaltet werden. Dabei stellen die Schüler in einem Rollenspiel einzelne Objekte dar. Sie können, ähnlich wie die Objekte »im Rechner«, miteinander interagieren. Damit steht eine handlungsorientierte Methode zur Verfügung. Der Erwartungshaltung, von Beginn an am Rechner zu arbeiten, kann aber auch mit diesem Konzept nicht entgegen gekommen werden. Ob das als Vor- oder als Nachteil zu sehen ist, wird in Kapitel 3 noch herausgearbeitet.

»fertigen« Rollenspiele, die in der Vorbereitung ein Nachdenken, in der Durchführung dann ein aktives Handeln der Schüler erfordern.

2 **Verschiedene Herangehensweisen im Anfangsunterricht der Jahrgangsstufe 11**

Für den Anfangsunterricht der Jahrgangsstufe 11 gibt es verschiedene Vorgehensweisen, die hier kurz dargelegt werden sollen. In Kapitel 3 werde ich dann begründen, warum ich das Konzept »objects and models strictly first«, zusammen mit den intensiven Einsatz von Objektspielen favorisiere und hier als Konzept ausarbeiten möchte.

2.1 **Klassenbegriff vor Objektbegriff**

Aus der Logik der objektorientierten Programmiersprachen selbst folgt die Vorgehensweise, zunächst den Klassenbegriff und unmittelbar danach den Objektbegriff einzuführen. So wird in vielen einführenden Informatik- Vorlesungen an den Universitäten vorgegangen. (Als Beispiel sei hier genannt: [Kuc07] Hier heißt es in Kapitel 2 unter »Grundbegriffe der Objektorientierung«: »Ein Java-Programm besteht aus mehreren Klassen; zu jeder Klasse gibt es Objekte (Instanzen)«.)

Auch in einer Lehrbuchreihe für die gymnasiale Oberstufe [Sch05, Sch06, Sch07] wird in Band 1 für die Jahrgangsstufe 11 [Sch05] in Kapitel 2 »Klassen und Objekte« (Kapitel 1 beschreibt die Installation der Programmierumgebung) der Klassenbegriff unmittelbar vor dem Objektbegriff eingeführt: »Man kann eine Klasse als Bauplan für Objekte auffassen.« (ebd., S. 16)

Die Vorteile dieser Vorgehensweise liegen auf der Hand: Von Anfang an sind alle Begriffe eingeführt, die nötig sind, um objektorientiert programmieren zu können. Die Schüler können Programme schreiben, indem sie entweder vorhandene Klassen nutzen (z.B. die Klasse »Stift« aus der Bibliothek Stifte und Mäuse) oder sofort eigene Klassen schreiben und aus diesen eigene Objekte erzeugen. Die Programme können sofort ausgeführt werden und man kann das Ergebnis als grafische (SuM) oder textuelle Ausgabe betrachten.⁴ Einer zentralen Erwartungshaltung zumindest Vorteile

⁴Ich würde es im Rahmen dieser Vorgehensweise und entgegen der genannten Lehrbuchreihe favorisieren, von Anfang an mit eigenen Klassen und einer rein textuellen Repräsentation der Objekte zu arbeiten. Eine grafische Ausgabe, etwa mit »Stifte und Mäuse« kann dann später hinzu kommen.

2 Verschiedene Herangehensweisen im Anfangsunterricht der Jahrgangsstufe 11

meiner Schüler, nämlich von Anfang an am Rechner zu arbeiten, wird damit entgegengekommen.⁵

Allerdings bringt dieser Ansatz auch einige Nachteile mit sich. Für Schüler der Jahrgangsstufe 11 stellt objektorientiertes Modellieren eine anspruchsvolle Abstraktionsleistung dar. Diese Abstraktion läuft beim Schüler in folgenden Schritten ab: Dinge aus der Realität werden vom Schüler als Objekt erkannt; gleichartige Objekte kann man dann zu Klassen zusammenfassen; damit sind Klassen dann Baupläne für Objekte. Klassen sind also noch abstrakter, als Objekte. Wenn nun zuerst der Klassenbegriff (als Bauplan) eingeführt wird, so geht an dieser Stelle der Bezug zur Realität verloren. Außerdem werden zwei, in Schüleraugen, ähnliche Begriffe zeitnah zueinander eingeführt, was den Effekt der Ähnlichkeitshemmung⁶ hervorrufen kann. Die Schüler unterscheiden dann nicht trennscharf zwischen den Begriffen, sie begreifen nicht, dass es zwar die Klassen sind, die programmiert werden, die Objekte aber die Akteure im Programm sind. Dieses Problem liegt der vorliegenden Arbeit zugrunde und ist in der Einleitung (Kapitel 1) entsprechend beschrieben worden.

Nachteile

2.2 Programmierumgebungen ohne Klassenbegriff

Es gibt Programmierumgebungen, die ohne den Klassenbegriff auskommen, trotzdem aber als objektorientiert oder zumindest objektbasiert bezeichnet werden können.⁷ Ein Beispiel ist die Programmierumgebung »Revolution dreamcard«⁸. Hier können in einer grafischen Umgebung Objekte auf einem Bildschirm »zusammengeklickt« werden, sie haben einen global bekannten Bezeichner, mit dessen Hilfe sie sich untereinander Nachrichten senden können.

Auch mit dieser Lösung können die Schüler sehr schnell ansprechende Ergebnis-

Vorteile

⁵Diese Erwartungshaltung wurde von den Schülern meines Kurses klar geäußert. Einige Schüler hielten es in der ersten Stunde für selbstverständlich, sogleich die Plätze am Rechner einzunehmen und die Rechner einzuschalten. In einer durchgeführten Eingangsbefragung waren erwartete Inhalte v.a. »Programmieren«, »Programmiersprachen« oder auch »Java«. (Java ist die eingesetzte Sprache an meiner Ausbildungsschule.)

⁶Ähnliche oder verwandte Dinge werden vom Lernenden verwechselt oder nicht trennscharf unterschieden, wenn sie zeitnah zueinander erlernt werden sollen.

⁷Das objektorientierte Konzept ist das Umfassendere. Der Begriff »objektbasiert« ist eingeschränkter, doch nicht konsistent definiert. Gegenüber der Objektorientierung wird an einigen Stellen nur auf den Verzicht auf Vererbung hingewiesen, an anderen Stellen sogar auf den Verzicht von Klassen überhaupt.

⁸Revolution Dreamcard wird an meiner Ausbildungsschule in der Differenzierung der Klassen 9 und 10 in einigen Kursen eingesetzt.

2 Verschiedene Herangehensweisen im Anfangsunterricht der Jahrgangsstufe 11

se präsentieren. Das ist sehr motivierend und kommt der Erwartungshaltung der Schüler entgegen.

Es ergeben sich aber auch hier einige entscheidende Nachteile. Objekte aus der Realität (Lebenswelt der Schüler) können zunächst nicht abgebildet werden. Damit entfällt ein wichtiger Modellierungsschritt. Es stehen nur einige »computertypische« Objekte (Klassen) zur Verfügung: Textboxen, Buttons, etc. Das verkürzt den Objektbegriff und ruft ein falsches Verständnis hervor. Mit der hier erwähnten Programmierumgebung ist auch das Erstellen eigener Klassen möglich. Dennoch wird im späteren Verlauf ein Umstieg auf eine andere Programmierumgebung nötig, nicht zuletzt, weil das vom Lehrplan auch im Hinblick auf das Zentralabitur gefordert wird.

Nachteile

2.3 **strictly objects and models first**

»strictly objects and models first« (vgl. [Die07a, Die07b]) meint nicht nur, dass die erste Programmiersprache, die Schüler kennen lernen, eine objektorientierte ist. Nach diesem Ansatz wird zuerst nur der Objektbegriff eingeführt, der Klassenbegriff aber erst später. Wie in der Einleitung bereits beschrieben, können damit nicht von Anfang an ausführbare Programme von den Schülern erstellt werden. Im Mittelpunkt steht vielmehr von Beginn an das informatische Modellieren⁹ (zunächst von Objektdiagrammen).

Dieses Verfahren folgt der von den Schülern schrittweise zu leistenden Abstraktion: Zuerst werden in der Realität Objekte identifiziert. Sie werden dann auf Eigenschaften untersucht und als Modell dargestellt. Dabei erfolgt eine Verkürzung der Eigenschaften im Hinblick auf die zu lösende Aufgabenstellung. Erst dann, mit zeitlichem Abstand, folgt ein weiterer Abstraktionsschritt: Objekte mit ähnlichen Eigenschaften werden in Klassen zusammengefasst. Zuletzt muss vom Schüler die Perspektive umgekehrt werden - er muss die Klasse als Bauplan für Objekte begreifen, also den Denkschritt gehen, aus einer bestehenden Klasse Objekte zu erstellen. Durch dieses Vorgehen wird außerdem das Problem der Ähnlichkeitshemmung umgangen, da die Begriffe »Objekt« und »Klasse« mit zeitlichem Abstand zueinander eingeführt werden.

Vorteile

⁹Unter Modellieren verstehe ich in dieser Arbeit den Prozess der informatischen Modellbildung aus der Realität heraus. Analyse geht der Modellierung voraus. Die für das Modell notwendigen Einzelheiten werden der Realität entnommen. Unter Programmieren verstehe ich das Umsetzen des Modells in eine konkrete Programmiersprache. Programmieren umfasst hier also nicht das Modellieren, sondern die beiden Tätigkeiten folgen aufeinander.

2 Verschiedene Herangehensweisen im Anfangsunterricht der Jahrgangsstufe 11

Die Vorgehensweise ermöglicht es mangels Klassenbegriff nicht, von Beginn an mit einer Programmierumgebung zu arbeiten. (Es sei denn, man arbeitet mit den in Kapitel 2.2 beschriebenen Umgebungen.) Damit kämme man einer zentralen Erwartungshaltung meiner Schüler nicht entgegen. Nachteile

Die einzigen Produkte, die Schüler in dieser Phase des Informatikunterrichts entwickeln können, sind passive, nicht ausführbare Objektdiagramme. Dies wirkt für die Schüler wenig motivierend und bietet kaum Gelegenheit, sich aktiv handelnd mit dem Gegenstand auseinanderzusetzen. Die Vorstellung von interagierenden Objekten entsteht bei den Schülern nicht zuverlässig. Hieraus ergibt sich ein zentrales Problem, das dieser Arbeit zugrunde liegt.

Der Ansatz kann durch den Aufgabentyp »Objektspiele« ergänzt werden.

3 Begründung der Auswahl dieses Ansatzes

Aus den oben als Vorteile beschriebenen Gründen bevorzuge ich den Ansatz »strictly objects first«: Linearität entlang der Abstraktionskette und Vermeidung der Ähnlichkeitshemmung.

Die Ergänzung um den Aufgaben- und Erarbeitungstyp der Objektspiele ermöglicht auch in der Anfangsphase eine handlungsorientierte Auseinandersetzung mit dem Gegenstand »objektorientierte Modellierung«. Ohne ein Informatiksystem¹⁰ benutzen zu müssen, können die Schüler Objekte als gekapselte, eigenständige Akteure erfahren. Der Klassenbegriff muss nicht eingeführt werden, da man von der Existenz der für die Problemlösung erforderlichen Objekte ausgehen kann, denn die »Objekte Schüler« sind ja einfach da. Einzig die Kursgröße begrenzt die Anzahl der zur Verfügung stehenden Objekte, was aber bei Problemen für den Anfangsunterricht keine Einschränkung darstellen sollte. Das Entstehen von Objekten aus Klassen kann später auch dargestellt werden. Dazu komme ich in den Kapiteln 4.4.6 und 5.6.

Zu diskutieren ist an dieser Stelle noch die Tatsache, dass durch die beschriebene Vorgehensweise der Erwartungshaltung der Schüler nach einem sofortigen Einsatz des Rechners nicht entgegen gekommen wird. Dies kann als Nachteil, aber auch als Vorteil angesehen werden. Ein Vorteil entsteht, wenn man es durch den Nicht-Einsatz der Rechner schafft, den Schülern zu verdeutlichen, dass der Rechner nur eines unter vielen möglichen Werkzeugen des Informatikers zur Umsetzung seiner Ideen ist. Wenn man als Lehrer auch ohne den Einsatz der Rechner einen schüleraktivierenden Unterricht gestaltet, so wird dieses Lernziel schon in den ersten Stunden erreicht und die Schüler sind gleichsam »geheilt« von dem Drang, zu Beginn jeder Informatikstunde sofort den Rechner einzuschalten.

Gerade der Aspekt der Objektspiele ist aber meines Erachtens noch nicht in befriedigendem Maße beschrieben worden. Ich möchte diese Arbeit deshalb nutzen, aufzuzeigen, wie Objektspiele von Beginn an im Informatikunterricht zur Vermittlung der objektorientierten Programmierung und Modellierung eine zentrale Rolle spielen können, und aufzeigen, wie dies praktisch umgesetzt werden kann.

¹⁰Natürlich könnte man auch einen Kurs »objektspielender Schüler« als Informatiksystem bezeichnen, dies ist hier aber nicht gemeint.

4 Konzept für den Anfangsunterricht mit Objektspielen

Die Beschreibung des Konzeptes für den Anfangsunterricht setzt mit der Einführung der Objektorientierung ein. Zuvor können und sollten im Unterricht natürlich grundsätzlichere Fragen, wie »Was ist Informatik?« geklärt werden.

Die erste Begegnung mit der objektorientierten Analyse kann dann der Zugang über die Methode von Abbott [Abb83] sein. (entsprechend modifiziert, so dass nur die Identifikation von Objekten, nicht aber die von Klassen gefordert wird.) Die Problemstellung sollte so gewählt und gestaltet sein, dass eine Umsetzung dieser ersten Problemstellung im Objektspiel möglich ist. Eine mögliche Problemstellung wird in Kapitel 4.2 beschrieben.

4.1 Literaturlage - bisherige Ansätze zu Objektspielen

Die Methode des Objektspiels ist in der Literatur mehrfach beschrieben worden.

Erstmals vorgeschlagen wurde sie von Bergin (vgl. [Ber06]). Dissmann wendet die Methode im Bereich der Erwachsenenbildung an. In [Diß03b] geht es um den Einsatz von Rollenspielen in der Informatik, auch außerhalb der Objektorientierung. In [Diß03a] beschreibt er, wie Objektspiele in der Ausbildung Erwachsener in der Objektorientierung genutzt werden können, während es in [Diß01] schwerpunktmäßig um die Weiterbildung bzw. Umschulung von Programmierern geht, die bisher nicht objektorientiert gearbeitet haben. Bei Dissmann werden für die einzelnen Objekte nicht, wie in dieser Arbeit, strenge formale Vorgaben für die einzelnen Dienste¹¹ spezifiziert. Vielmehr sind die einzelnen Objekte »Experten« für bestimmte Aufgaben, die sie innerhalb der Modellierung wahrnehmen.

Auch bei Diethelm ([Die07b, Die07a]) findet sich die Objektspielmethode, die sie im Zusammenhang mit dem Konzept »strictly models and objects first« und der didaktischen Entwicklungsumgebung »Fujaba« anwendet.

Ausgehend von diesen Ansätzen habe ich die Methode in der vorliegenden Arbeit weiterentwickelt.

¹¹Ich verwende in dieser Arbeit statt des synonymen Begriffes »Methode« den Begriff »Dienst«, um keine Irritationen in Bezug auf den Begriff der Unterrichtsmethode aufkommen zu lassen. Ich empfehle, diese Unterscheidung auch im Unterricht einzuhalten.

4.2 Geeignete Problemstellung für eine erste Modellierungsaufgabe

Eine Problembeschreibung für eine erste Modellierungsaufgabe sollte folgende zehn Bedingungen erfüllen:

- Die Objekte stammen aus der Lebenswelt der Schüler. Das wirkt motivierend und die Schüler können sich mit der Aufgabenstellung identifizieren. Ein Beispiel aus dem direkten Schulumfeld hat den Vorteil, dass alle Schüler das gleiche Vorwissen haben, allerdings den Nachteil, dass dies weniger motivierend wirken könnte. Schön wäre deshalb ein Beispiel aus dem Umfeld »Freizeit« das allgemein bekannt ist, so dass alle Schüler gleichermaßen an der Aufgabe arbeiten können.
- Das Beispiel sollte genügend komplex sein. Das heißt, die zu identifizierenden Objekte sollten mindestens zwei Attribute besitzen, die sich auch in ihren Attributwerten unterscheiden.¹²
- Es soll zu jeder im Beispiel vorkommenden Klasse mindestens zwei Objekte geben. Ist dies nicht möglich, sollten zumindest Klassen vorkommen, zu denen es mehr als ein Objekt gibt. Auch wenn der Klassenbegriff nicht zu Beginn eingeführt werden soll, so kann er doch im späteren Verlauf des Unterrichts auf das Einstiegsbeispiel angewendet werden. Für eine eindeutige Unterscheidung der Begriffe ist dieses Kriterium deshalb wichtig. Mit Blick auf bestimmte Programmierumgebungen ist es evtl. notwendig, ein zentrales Steuerobjekt einzuführen. Auch hier gibt es zu einer Klasse genau ein Objekt.
- Es sollten nicht zu viele Objekte (einer Klasse) im Objektspiel vorkommen. So können die Situationen auch in Kleingruppen im Kurs dargestellt werden, was die Aktivität der Schüler erhöht. Gerade, wenn man keine Nebenläufigkeit zulässt¹³, wird der einzelne Schüler als Objekt umso seltener aktiv, je

¹²Das von Diethelm in [Die07b] vorgeschlagene Einstiegsbeispiel »Mensch-ärgere-dich-nicht« halte ich deshalb für ungünstig, denn ein Objekt »Spielfeld« (nicht »Spielbrett«, sondern das einzelne Feld, auf dem sich eine Figur befinden kann) hat auf den ersten Blick keine Attribute. Die verschiedenen Objekte lassen sich damit, aufgrund dieses Kriteriums, nicht unterscheiden. Der Objekt- und der Klassenbegriff verwischen so.

¹³Im Objektspiel wäre das kein Problem, man sollte den Einsatz durch die Schüler aber m.E. dann von Anfang an unterbinden, wenn auch die später zu benutzende Programmierumgebung

4 Konzept für den Anfangsunterricht mit Objektspielen

mehr Objekte sich am Objektspiel beteiligen. Überträgt man Nachrichten so, dass tatsächlich nur die beiden beteiligten Objekte etwas davon mitbekommen (mehr dazu in Kapitel 4.4.4), kann schnell Langeweile entstehen, weil man lange auf seinen nächsten »Einsatz« warten muss. Wenn man möchte, dass einzelne Schüler (nicht als Objekte!) einen Gesamtüberblick über das Programm bekommen, sind kleine Modellierungen ebenfalls sinnvoll.^{14 15}

- Im Beispiel sollten Objekte mehrerer Klassen vorkommen, um den Klassenbegriff an diesem Beispiel einführen oder auf dieses Beispiel anwenden zu können.
- Die dargestellten Objekte sollten nicht (alle) »Mensch-Objekte« sein, um deutlich zu machen, dass sich mit Hilfe der Objektspiel-Methode beliebige Objekte aus der Realität darstellen und simulieren lassen.
- Jedes Objekt hat eine endliche und relativ kleine Anzahl von Bezugsobjekten. »zu-n-Beziehungen« sollte also nicht modelliert werden, da sie erstens im Objektspiel nicht darstellbar sind und ausserdem auch zu Anfang von den Schülern nicht programmiert werden können, da jedes Bezugsobjekt einzeln im Code deklariert wird.
- Objekte werden von mehreren Objekten referenziert, so dass der Unterschied zwischen Objektbezeichner und Beziehungsbezeichner deutlich wird. Das Problem hat also keine baumartige, sondern eine allgemein graphenartige Struktur.
- Um den Objektbegriff möglichst konkret zu machen, sollte man möglichst greifbare Objekte aus der Wirklichkeit (»Dinge, die man anfassen kann«) modellieren.
- Mit Blick auf eine konkrete Ausführbarkeit wäre es schön, ein Problem zu finden, das auch für die Schüler ein Problem darstellt. Die Modellierung, deren

keine Nebenläufigkeit zulässt, oder man die Schüler damit nicht konfrontieren möchte. Ob es, bei einfacher Umsetzung in einer Sprache, sinnvoll wäre, Nebenläufigkeit zu betrachten (oder als intuitiv verständlich einfach als Selbstverständlichkeit betrachtet) soll hier nicht weiter diskutiert werden.

¹⁴Es könnte sinnvoll sein, gerade auf große Modellierungen zurückzugreifen, um den Schülern bewusst zu machen, dass kein Objekt (kein Schüler) einen Überblick über das gesamte Problem haben muss, um zu einer Lösung zu gelangen. Evtl. lassen sich solche Probleme auch einmal mit einem ganzen Kurs darstellen. Dieser Gedanke soll hier aber nicht weiter verfolgt werden.

¹⁵Das Beispiel »Mensch-ärgere-dich-nicht« benötigt eine Vielzahl von Feldern. Allerdings lässt sich dieses Problem, wie auch bei anderen, ähnlich gearteten Gesellschaftsspielen, dadurch umgehen, dass man eine geringere Anzahl Felder betrachtet. Das Spiel funktioniert dann ebenfalls.

Implementierung und Ausführung würde dann unmittelbar der Problemlösung dienen.

Ein Beispiel für eine Problemstellung, die diesen Kriterien weitgehend genügt, findet sich in Anhang.¹⁶

4.3 Objektkarten und Objektdiagramme

In der UML- Notation gibt es Objektkarten, die wie die zugehörigen Klassenkarten je einen Bereich für den Objektnamen, die Attribute und die Dienste enthalten. Um eine bessere Unterscheidung zwischen Klassendiagrammen und Objektdiagrammen zu ermöglichen, hat die Didaktik der Informatik Objektkarten mit »runden Ecken« für das Objektdiagramm eingeführt. Die Beziehungen zwischen den Objekten werden durch Verbindungslinien zwischen den Objekten dargestellt.¹⁷

Die UML- Notation ermöglicht, eine Objektkarte mit »:BeispielKlasse« zu überschreiben. Das dargestellte Objekt ist ein Objekt vom Typ »BeispielKlasse«. So lange der Klassenbegriff noch nicht eingeführt ist, kann diese Notation aber nicht verwendet werden. Jedem Objekt muss damit ein Name gegeben werden, mit dem die Objektkarte überschrieben wird. Nach der Einführung des Klassenbegriffes aber kann die beschriebene Notation verwendet werden.

Durch die Einführung eines solchen Objektnamen entsteht beim Schüler leicht der Eindruck, es handele sich um einen global bekannten Objektbezeichner, unter dem alle in der Modellierung vorhandenen Objekte mit dem betreffenden Objekt kommunizieren können. Dies ist aber ein falscher Eindruck, ein Objekt hat keinen individuellen Namen. Ein, als Objektkarte mit beispielsweise »karteA« überschriebenes, Objekt weiß in einem realen System nicht einmal von sich selbst, dass es den Namen »karteA« trägt.¹⁸ Vielmehr hat jedes Objekt für seine Bezugsobjekte eigene Bezeichner. Objekte werden immer über Objektreferenzen angesprochen.

Um dies im Objektdiagramm darstellen zu können, schlage ich vor, den Standard für das Objektdiagramm zu verändern. Neben den Attributen und Diensten wird ein dritter Bereich in der Objektkarte eingeführt, in dem die Bezugsobjekte angegeben

¹⁶An dieser Stelle werden meines Erachtens noch gute Ideen benötigt. Geeignete Modellierungsbeispiele sind wichtig, aber nicht leicht zu finden.

¹⁷Dabei gibt es, auch grafisch dargestellt, die Unterscheidung zwischen hat- und kennt- Beziehung bzw. Aggregation und Assoziation.

¹⁸Davon zu unterscheiden ist ein evtl. eingeführtes Attribut »Name« des Objektes, das eine Zeichenkette mit einer Namensinformation enthalten kann.

4 Konzept für den Anfangsunterricht mit Objektspielen

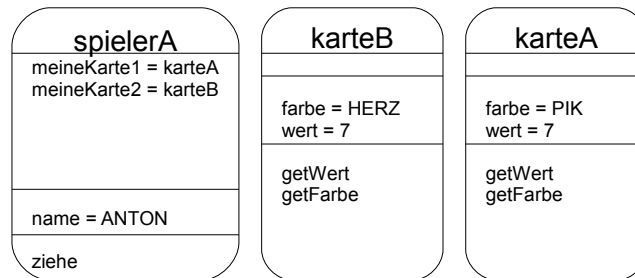


Abbildung 1: Objektdiagramm mit eingetragenen Referenzen

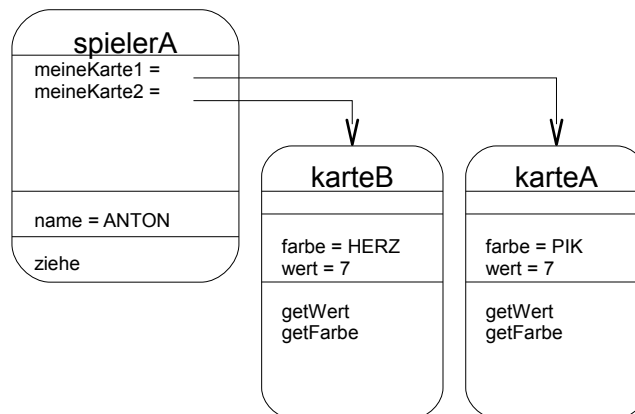


Abbildung 2: Objektdiagramm mit Beziehungspfeilen

werden. Objektkarten sehen dann aus, wie in den Abbildungen 1 und 2 in einem kleinen Objektdiagramm beispielhaft dargestellt.

Es ergeben sich für ein Objektdiagramm zwei verschiedene Darstellungsarten. Zum einen kann hinter den Objektbezeichner als Wert der Objektname eingetragen werden, (Abbildung 1) zum anderen ist es möglich, auf die bekannte Darstellung mit Pfeilen zurückzugreifen. Die Pfeile gehen nun allerdings nicht mehr vom Rand der Objektkarte aus, sondern vom entsprechenden Bezeichner des Bezugsobjektes. (Abbildung 2) Den Schülern sollte deutlich gemacht werden, dass die Semantik dieser beiden Darstellungsarten vollkommen identisch ist.

Die Bezeichner der Bezugsobjekte sollten so gewählt werden, dass sie über die Art der Beziehung informieren. Solange der Unterschied zwischen hat- und kennt-

4 Konzept für den Anfangsunterricht mit Objektspielen

Beziehungen noch nicht eingeführt ist, kann diese Unterscheidung natürlich noch nicht getroffen werden. Dennoch lassen sich oft Informationen über die Art der Beziehung ausdrücken, wie im Beispiel die Information, ob es sich um das Vorderrad oder das Hinterrad handelt.

Der Objektname kann fast beliebig gewählt werden. Dies kommt in der Beschreibung des Objektspiels noch einmal zum Ausdruck. (s. Kapitel 4.4) Für »reine« Objektdiagramme, die nicht im Objektspiel umgesetzt werden, schlage ich vor, Namen zu wählen, die zwar etwas über die Eigenschaft des Objektes aussagen, aber keine Informationen über Beziehungen oder Besitzverhältnisse enthalten.¹⁹ Eine noch »radikalere« Alternative wäre die Bezeichnung aller Objekte mit »objektA«, »objektB« und so weiter. Dies ist die Notation, die der Organisation im Speicher des Rechners am nächsten kommt, da die Objektnamen als Speicheradressen der Objekte interpretiert werden können. Ob diese Variante beim Schüler zu Verständnisproblemen führt, müsste sich in der Praxis zeigen.

Aufgrund dieses Vorschlags ergeben sich Konsequenzen für die Darstellung von Klassendiagrammen. Die Bezugsobjekte können auch im Klassendiagramm analog zum Objektdiagramm eingetragen werden. Es ergibt sich eine Darstellung, die größere Analogie zum Quelltext der Umsetzung in eine Programmiersprache hat, als die reine UML- Darstellung, da im Quelltext alle Bezugsobjekte deklariert werden. Allerdings müssen zwischen zwei Klassen nun gegebenenfalls mehrere parallele Beziehungen eingezeichnet werden. Kardinalitäten entfallen auf der rechten Seite²⁰ der Beziehung zunächst.²¹ Nachteilig an dieser Darstellungsweise ist, dass zunächst keine Beispiele modelliert werden können, die »zu-n-Beziehungen« beinhalten. Sie konnten auch bisher nicht implementiert werden, wurden aber als realitätsnah von den Schülern problemlos erkannt und modelliert.

Aus diesem Grund halte ich es für wichtig, in einem weiteren Abstraktionsschritt das Klassendiagramm gemäß UML einzuführen und die Beziehungen nur noch durch Kanten zwischen den Klassen darzustellen. Dann können auch die Kardinalitäten eingeführt werden. Dieser Schritt ist auch aus wissenschaftspropädeutischer Sicht notwendig, da es sich bei UML um einen industriellen und akademischen Standard handelt.

¹⁹Im Grunde ist darin dann ein Klassenname enthalten. Gleichartige Objekte müssen vermutlich häufig durchnummeriert werden, wie auch im Beispiel.

²⁰Wenn man davon ausgeht, dass eine Beziehung von links nach rechts verläuft.

²¹Ich schlage vor, Kardinalitäten damit zunächst gar nicht zu betrachten.

4.4 Das Objekt- Rollenspiel

Bevor die Schüler das Objekt-Rollenspiel kennen lernen, sollten sie ein Objektdiagramm erstellt haben. Dies kann anhand des Beispiels geschehen, mit dem auch im Objektspiel gearbeitet wird.

Dem Objekt- Rollenspielen liegt folgende Idee zugrunde: Objekte sind innerhalb eines Informatiksystems autonome Akteure. Sie sind gegenüber den anderen beteiligten Objekten gekapselt, es gilt das Geheimnisprinzip. Sie kommunizieren mit anderen Objekten über eine klar definierte Schnittstelle durch den Aufruf von Diensten.

Dieses Vorgehen kann im Rollenspiel nachgestellt werden. Je ein Schüler stellt ein Objekt dar. Die Schüler kommunizieren untereinander über eine ebenfalls klar definierte Schnittstelle, sie simulieren so den Ablauf in einem objektorientierten Programm. Die Schüler erfahren damit aktiv handelnd, wie eine objektorientierte Software arbeitet. Sie erhalten dazu je eine Objektkarte, die Informationen über das Objekt enthält, das sie jeweils darstellen. Dabei handelt es sich um eine Objektkarte, so wie oben beschrieben. Sie wird so weit vergrößert, dass der Schüler bequem Informationen darauf notieren kann. (Zur praktischen Umsetzung siehe Kapitel 4.4.7.)

Wie die Umsetzung der verschiedenen Details aussehen kann, soll in diesem Kapitel dargestellt werden.

4.4.1 Objektnamen und Beziehungen

Oben wurde bereits erläutert, dass der Objektname selbst, mit dem eine Objektkarte überschrieben ist, keinerlei Semantik über das modellierte Problem enthält. Deshalb kann als Objektname auch der Name des das Objekt darstellenden Schülers verwendet werden.²² Dieser wird oben auf der Objektkarte, die jeder Schüler als Objekt erhält, eingetragen.

Jedes Objekt hat je einen Bezeichner für seine Objektbeziehungen. Als »Wert« dieser Objektbeziehungen (analog zum Wert der Attribute) wird der Name des Schülers eingetragen, der gerade das entsprechende Bezugsobjekt darstellt. Abbildung 3 zeigt einen Ausschnitt aus einer Modellierung mit Schülernamen als Objektreferenzen und Objektnamen.

²²Meist der Vorname, bei nicht- Eindeutigkeit ergänzt um einen für Eindeutigkeit hinreichenden Zusatz, z.B. dem Anfangsbuchstaben des Nachnamens.

4 Konzept für den Anfangsunterricht mit Objektspielen

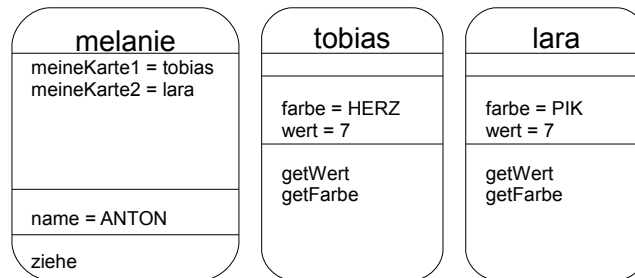


Abbildung 3: Objektdiagramm mit Schülernamen als Referenzen

Auf diese Weise entsteht nicht das Problem, dass Objekte im Objektspiel global bekannte Bezeichner haben, was nicht der Realität einer objektorientierten Software entspricht. Ein Objekt (ein Schüler) kann also von verschiedenen Objekten, deren Bezugsobjekt er ist, unter verschiedenem Namen angesprochen werden. Jedes Objekt bzw. jeder Schüler muss durch das Nachsehen auf seiner Objektkarte seinen lokalen Bezeichner für das Bezugsobjekt auflösen. Dies entspricht dem Auflösen eines Objektbezeichners in eine Speicheradresse. Dieser Aspekt kann im Verlauf des Kurses (zu einem späteren Zeitpunkt) thematisiert werden.

Eine alternative Darstellung der Beziehungen wäre der Einsatz von Schnüren, die die Beziehungen zwischen den Objekten darstellen. Eine Schnur würde dann von den beiden an der Beziehung beteiligten Objekten festgehalten. Hier ergibt sich aber das Problem, dass nicht unmittelbar deutlich wird, dass es sich um eine gerichtete Beziehung handelt. Eine weitere Alternative wäre, dass ein Schüler als Objekt seine Bezugsobjekte anfasst. Gerichtetheit kann hier dargestellt werden, allerdings kann ein Objekt maximal zwei Bezugsobjekte haben. Außerdem könnte es den Schülern unangenehm sein, sich anzufassen.

Ich schlage vor, die didaktische Unterscheidung zwischen hat- und kennt-Beziehung erst mit der Einführung des Klassenbegriffes zu betrachten. Anfangs gibt es nur Beziehungen zwischen existierenden Objekten. Der Bezeichner für ein Bezugsobjekt könnte etwa »meineKarte« heißen. Erst mit der Einführung des Klassenbegriffes können die Schüler Programme am Rechner umsetzen. Dann muss jedes Objekt erzeugt werden, was von einem anderen Objekt erledigt wird. Zwischen diesen Objekten besteht dann eine hat-Beziehung. Alle anderen Beziehungen sind kennt-Beziehungen.

4.4.2 Schnittstelle und Objektinterna

Es sollte von Anfang an klar gestellt werden, dass nur die Dienste²³ die Schnittstelle eines Objektes definieren. Objekte können dann mit ihren Bezugsobjekten (und nur mit diesen) kommunizieren, indem sie deren Dienste aufrufen und eventuell Rückgabewerte entgegennehmen.²⁴ Alle anderen Informationen hat ein Objekt nur über sich selbst. Ein Objekt kennt sich selbst und kann entsprechend all seine Dienste aufrufen.

4.4.3 Attribute

Attribute sind private Informationen, mit deren Hilfe ein Objekt seinen aktuellen Zustand repräsentiert. Die Schüler notieren »ihre« jeweiligen Attributwerte auf ihrer Objektkarte. Es ist wichtig, sie darauf aufmerksam zu machen, dass sie kein Wissen über sich selbst haben, außer genau dem, das auf der Objektkarte notiert ist.

4.4.4 Nachrichten: Dienste mit Argumenten und Parametern

Auf der Objektkarte sind die Dienste als Schnittstelle des Objektes definiert. Ein Schüler kann damit auf den ersten Blick erkennen, ob der Dienst, die bei ihm aufgerufen wird, gültig ist.

Auf einem weiteren Blatt sind die Dienste als Struktogramm²⁵ spezifiziert. Es ist also angegeben, was das Objekt tun soll, wenn der entsprechende Dienst aufgerufen wird. Diese Anweisungen sind in der ich-Form formuliert. Zu einem Dienst »setzeWert« würde die Anweisung also lauten: »Ich **setze** den Wert meines Attributes **wert** auf den im Parameter **neuWert** übergebenen Wert«. Dadurch wird den Schülern noch einmal deutlich, dass sie selbst und die Objektkarte nicht getrennt zu betrachten sind, sondern dass beide gemeinsam stets ein konkretes Objekt repräsentieren. Die wichtigen Teile der Anweisungen sind auf den Dienstekarten **fett** dargestellt. Es sind genau die Teile, die bei der objektorientierten Punktnotation noch notiert werden, ergänzt um bestimmte Zeichen, die die Syntax der Programmiersprache definiert. Die oben angegebene Anweisung würde als Java- Anweisung

²³Genauer: Der öffentliche Teil der Dienste. Dieses Detail muss hier jedoch noch nicht thematisiert werden, denn private Dienste werden für die bisher betrachteten einfachen Probleme vermutlich nicht zwingend benötigt.

²⁴Begrifflich kann hier nach [Sch05, Sch06, Sch07] zwischen »Aufträgen« und »Anfragen« unterschieden werden. »Dienst« (oder synonym »Methode«) ist der Oberbegriff.

²⁵oder Nassi-Shneiderman-Diagramme; nach DIN 66216

4 Konzept für den Anfangsunterricht mit Objektspielen

etwa lauten: »wert = neuWert«. Die Anweisungen sind möglichst so formuliert, dass sich die Reihenfolge der einzelnen Elemente beim Übergang zu dieser Kurzschreibweise nicht verändert. Entsprechende Beispiele finden sich im Anhang. In Anhang I ist ein Dienst schon in der Kurzschreibweise notiert, die anderen Dienste sind in der ausführlicheren »ich-Schreibweise« dargestellt.

Der Inhalt der Dienste ist meist intuitiv und einfach. Er folgt häufig kanonisch aus dem Dienstbezeichner. Das Nachschlagen wird deshalb nur selten notwendig sein.

Der Austausch von Nachrichten kann im Objektspiel auf verschiedene Art erfolgen:

- Die Nachrichten können schriftlich ausgetauscht werden. Dies hat den Vorteil, dass nur die beiden beteiligten Objekte den Inhalt der Kommunikation zur Kenntnis nehmen können. Den Schülern wird deutlich, dass die Objekte autonom und gekapselt handeln, alle anderen Objekte im Modell unbeteiligt und ohne Kenntnis der Vorgänge sind. Gerade bei längeren Parameterlisten kann durch schriftliche Nachrichten Eindeutigkeit erzielt werden. Von Nachteil ist, dass es jedes Mal Zeit in Anspruch nimmt, Nachrichten zu verfassen und schriftlich zu beantworten. So kann, besonders in großen Gruppen mit vielen beteiligten Objekten, leicht Langeweile entstehen, da nicht aktive Objekte über einen größeren Zeitraum nichts zu tun haben.
- Die Nachrichten können mündlich ausgetauscht werden. Eine Anweisung wie »Ich gebe meinem Bezugsobjekt **meinMitspieler** den Auftrag **ziehe**.«, oder in der Kurzschreibweise »meinMitspieler.ziehe()« würde dann im Objektspiel so aussehen: Der Schüler, der das sendende Objekt darstellt, löst den Bezeichner »meinMitspieler« mit Hilfe der notierten Bezugsobjekte auf. Er findet dort den Namen des Schülers, der dieses Bezugsobjekt darstellt und kann dann etwa formulieren: »Lisa, ich gebe dir den Auftrag ziehe!«. Lisa, die ein Karte-Objekt darstellt, würde dann den Dienst »ziehe« ausführen. Vorteilhaft an dieser Vorgehensweise ist die einfache und schnelle Umsetzung. Ein Nachteil kann sein, dass alle beteiligten Objekte die Kommunikation mitbekommen, was keine Entsprechung im Informatiksystem hat. Dies kann man aber auch als Vorteil ansehen, denn alle Schüler nehmen so das Geschehen zur Kenntnis, erfahren, wie der Ablauf des Programmes vonstatten geht und können eventuell auftretende Fehler korrigieren. Dies gilt auch für gegebenenfalls vorhandenes »Publikum«, wenn nicht jeder Schüler des Kurses ein Objekt spielen

4 Konzept für den Anfangsunterricht mit Objektspielen

kann. Die Schüler müssen dann aber von ihrer Rolle als Objekt abstrahieren können, also zum gleichen Zeitpunkt »Objekt« und »Schüler« spielen. Das gelingt meiner Erfahrung nach aber gut.

- Ein Kompromiss zwischen diesen beiden Varianten wäre das Zuflüstern von Nachrichten. Der Zeitaufwand, Nachrichten schriftlich zu formulieren entfällt. Gleichzeitig wird deutlich, dass an der Kommunikation stets nur zwei Objekte beteiligt sind. Schülern könnte das Zuflüstern aber unangenehm sein, da es ein Mindestmaß an körperlicher Nähe erfordert. Der Lehrer sollte immer in Bezug auf seine jeweilige Lerngruppe entscheiden, ob diese Möglichkeit praktikabel ist.

Beim Ausführen von Diensten kann das aktive Objekt, die auf den Dienstekarten angegebenen Anweisungen, entweder laut vorlesen und dann ausführen, oder, ohne sie vorzulesen, einfach ausführen. Lautes Vorlesen würde wieder allen Beteiligten einen Eindruck vom Geschehen vermitteln, was nicht objektorientierungs-typisch, aber aus didaktischen Gründen eventuell dennoch erwünscht ist. Stilles Abarbeiten kann wiederum zu Langeweile führen. Zumindest teilweise sollte man aber ohne lautes Vorlesen arbeiten, um den Schülern deutlich zu machen, dass kein Objekt wissen muss, *wie* ein anderes Objekt seine Dienste ausführt.

4.4.5 Aktives Objekt

Um deutlich zu machen, dass zu einem Zeitpunkt stets nur ein Objekt aktiv sein kann, bietet es sich an, eine Art Staffelstab einzuführen, der mit jedem Nachrichtenaufruf an das Bezugsobjekt weitergegeben wird. Hier kommt eine Komponente ins Spiel, die unvollständig abgebildet wird: Die Objekte müssen sich merken, von welchem Objekt sie den Staffelstab erhalten haben und an welches sie ihn zurückgeben müssen. (Sie haben zu diesem Objekt nicht notwendigerweise eine eigene Beziehung!) Das gilt auch bei geschachtelten Aufrufen. Diese Information wird nicht notiert, denn es gibt keine Entsprechung mit einem Objekt. In einer objektorientierten Programmierungsumgebung wird dieser Aspekt über den Stack organisiert, der im Objektspiel nicht abgebildet werden kann.

Mit der Übergabe des Staffelstabes werden Dienste aufgerufen, mit der Rückgabe bei Anfragen Rückgabewerte zurückübermittelt. Wenn man mit schriftlichen Nachrichten arbeitet, kann man diese in geeigneter Weise am Staffelstab befestigen.

4.4.6 Klassen

Das Erstellen von Objekten aus Klassen kann im Rollenspiel nach der Einführung des Klassenbegriffes durch Aufstehen der Schüler geschehen. Sitzende Schüler existieren als Objekt nicht. Wenn ein Objekt aus einer Klasse erstellt wird, steht der betreffende Schüler auf und ist ein Objekt. Er führt als erstes seinen Konstruktor aus. Der Vorgang des Aufstehens kann auch interpretiert werden als Belegen von Speicher im freien Speicherraum (sitzende Schüler).

4.4.7 Praktische Umsetzung

An dieser Stelle gebe ich einige Hinweise zur praktischen Umsetzung der Rollenspiele im Unterricht, insbesondere zu den einzusetzenden Materialien.

- Jeder Schüler, der ein Objekt darstellt, erhält seine Objektkarte auf einem Klemmbrett. So kann er auch im Stehen leicht Veränderungen seines Zustandes notieren.
- Objekte, die Berechnungen anstellen müssen, erhalten u.U. dazu einen Taschenrechner, der auf dem Klemmbrett befestigt werden kann. Der Taschenrechner ist nicht als Objekt zu begreifen!
- Die Objektkarten werden in Prospekthüllen gesteckt. Attributwerte und Bezugsobjekte können so mit einem Folienstift oder einem Whiteboard-Marker (trocken korrigierbar) vorgenommen werden. So lassen sich Veränderungen beliebig oft vornehmen und die Objektkarten wiederverwenden. Wenn man als Lehrer eine Version der Objektkarten gefunden hat, die sich als nicht mehr verbesserungswürdig erwiesen hat, so kann man die Objektkarten auch laminieren.

5 Möglicher Ablauf einer Reihe im Anfangsunterricht

In diesem Kapitel wird aufgezeigt, wie das oben dargestellte Konzept im Unterricht umgesetzt werden kann. Insbesondere soll hier zum Ausdruck kommen, in welcher Reihenfolge die einzelnen, dargestellten Elemente eingeführt werden können. Dieses Kapitel gliedert sich demzufolge in Unterrichtssequenzen, die einer möglichen chronologischen Folge im Unterricht entsprechen. Wie viele Unterrichtsstunden für die jeweiligen Sequenzen verwendet werden, hängt vom Kurs und von individuellen Schwerpunktsetzungen ab.

Je nachdem, welchen Wert man darauf legt, dass die Schüler ihre Modellierungen auch implementieren, wird man die Reihenfolge der einzelnen Inhalte variieren müssen. Insbesondere die Einführung des Klassenbegriffes kann mehr oder weniger früh geschehen. Ausführlich dargestellt werden im Folgenden nur die Sequenzen, die sich unmittelbar auf das Rollenspiel beziehen, oder die vor dem Rollenspiel notwendig sind, um Begriffe einzuführen. Phasen der Implementierung und des Erlernens der Syntax einer konkreten Programmiersprache können an geeigneten Stellen eingeschoben werden. Ausserdem sollten die Inhalte immer wieder anhand von anderen Beispielen geübt werden. Hier werden, in Stichworten, verschiedene denkbare Vorgehensweisen dargestellt. Die ausführlich erläuterten Teile, die ich auf das Rollenspiel beziehen, sind **fett** gedruckt:

Einführung aller grundlegenden Begriffe ohne Implementierung: **Objekt – Attribut – Dienst – Rollenspiel – Argument und Parameter²⁶ (in Objektdiagramm und Rollenspiel) – Kontrollstrukturen in den Diensten – Klasse – Implementierung der bereits erarbeiteten Elemente.**

Frühe Implementierungsphasen: **Objekt – Attribut – Dienst – Rollenspiel – Klasse – Implementierung – Argument und Parameter im Objektdiagramm**

²⁶Die Begriffe Argument und Parameter verwende ich hier wie folgt: Ein Argument ist der Wert, der beim Dienstaufruf mit übergeben wird. (z.B. `hatKreis.bewegeUm(7)`; 7 ist der Parameter.) Dieses Konstrukt wird auch als »aktueller Parameter« bezeichnet (vgl. [CS06, S. 488]). Als Parameter bezeichne ich den Platzhalter (die Variable) im empfangenden Objekt bzw. in der empfangenden Klasse. Hier findet man auch die Bezeichnung »formaler Parameter« (vgl. ebd.) Um die Begriffe eindeutig zu unterscheiden, schlage ich vor, auch im Unterricht von »Argument« und »Parameter« zu sprechen. Mehr zu diesem Thema findet sich bei [Pot07].

und Rollenspiel – (Parameter im Klassendiagramm²⁷ –) Implementierung Parameter – **Kontrollstrukturen im Objektdiagramm und im Rollenspiel** – Implementierung von Kontrollstrukturen.

Eine Mischung dieser beiden Vorgehensweisen ist ebenso denkbar. Entscheidend ist, an welcher Stelle man den Klassenbegriff einführt und damit die Möglichkeit schafft, mit einer objektorientierten Programmiersprache zu arbeiten.

Im Anhang wird diese Vorgehensweise noch einmal auf die konkrete Modellierungsaufgabe »MauMauSpiel« angewendet. Zu diesem Einführungsbeispiel finden sich einige Unterrichtsmaterialien. Außerdem kann die konkrete Vorgehensweise an diesem Beispiel noch deutlicher werden.

5.1 Erste Sequenz - Einführendes Modellierungsbeispiel

Ziel dieser Sequenz: Die Schüler lernen die Begriffe »Objekt«, »Dienst«, »Attribut« und »Objektbeziehung« kennen. Sie können die Begriffe auf einen Ausschnitt der Realität anwenden und in geeigneter Weise grafisch darstellen.

Möglicher Ablauf: Den Schülern wird ein geeignetes Modellierungsbeispiel vorgelegt. Im Anhang A findet sich ein Vorschlag für eine Situationsbeschreibung. Die Schüler identifizieren in diesem Beispiel zunächst beteiligte Objekte. Dazu kann man den Schülern als Werkzeug die Methode von Abbott an die Hand geben. Allerdings habe ich die Erfahrung gemacht, dass der Objektbegriff so intuitiv ist, dass er sich aufgrund eines geeigneten Beispiels gut ohne zusätzliche Informationen erarbeiten lässt.

Die identifizierten Objekte werden als Objektkarten dargestellt. Als Objektbezeichner dient ein Bezeichner, der hinreichend allgemein ist, also keine Informationen über Beziehungen enthält.

Attribute, Dienste und Bezugsobjekte werden identifiziert und ebenfalls in den Objektkarten dargestellt. Bezüglich der Objektbeziehungen kann auf die beiden möglichen Darstellungsweisen hingewiesen werden, die in Kapitel 4.4.1 beschrieben wurden. So entsteht ein Objektdiagramm.

²⁷Entgegen der UML- Notation ist es möglich, Parameter auch im Objekt- und Klassendiagramm darzustellen. Ob diese Möglichkeit genutzt wird, bleibt hier offen.

5.2 Zweite Sequenz - Objekte im Rollenspiel

Ziel dieser Sequenz: Die Schüler erfahren, dass Objekte im Rollenspiel darstellbar sind. Sie erarbeiten sich, dass Objekte dynamisch miteinander agieren können und dabei laufend ihren Zustand ändern. Sie erkennen, dass mit Hilfe eines Objektspiels Problemlösungen simulierbar sind. Sie erhalten so eine Vorstellung vom Ablauf eines objektorientierten Programmes.

Möglicher Ablauf: Zunächst arbeiten die Schüler in Paaren: Schüler A stellt ein Objekt dar, das keine Bezugsobjekte hat. Im Beispiel sind das die Karten. Schüler B stellt das Objekt dar, das eine Beziehung zu dem von Schüler A dargestellten Objekt hat. Er kann mit diesem Objekt kommunizieren. Schüler B gibt »seinem« Objekt nun verschiedene Aufträge und Anfragen, die das Objekt bearbeitet. Dabei sollten die verschiedenen Diensttypen vorkommen. Bisher können nur parameterlose Dienste betrachtet werden.

Nun wird die Komplexität des Problems erweitert: Das ganze, in der ersten Sequenz dargestellte, Objektdiagramm wird im Rollenspiel dargestellt. Dabei erfahren die Schüler, dass es wichtig ist, die Bezugsobjekte und Attributwerte auf den Objektkarten zu notieren, um einen Überblick zu behalten. Die Schüler spielen einfache Ketten von Dienstaufrufen durch.

5.3 Dritte Sequenz - Argumente und Parameter

Ziel dieser Sequenz: Die Schüler erkennen durch die eigene Erarbeitung im Rollenspiel die Notwendigkeit von Parametern und Argumenten. Sie können Parameter und Argumente problemadäquat anwenden.

Möglicher Ablauf: Die Schüler erhalten die Aufgabe, im Rollenspiel eine Problemlösung darzustellen, die die Übergabe von Argumenten und den Einsatz von Parametern einfacher Typen (Zahlen, Zeichenketten) erfordert. Nachdem die Schüler beschreiben, dass es notwendig ist, den Bezugsobjekten zum Dienstauf-ruf bestimmte Daten mitzuteilen, wird der Begriff »Argument« für diese Daten eingeführt, der Begriff »Parameter« für das empfangende Objekt.

Die Schüler üben den Einsatz von Argumenten und Parametern an geeigneten Beispielen im Rollenspiel. Dabei werden auch Attribute als Argument über-

5 Möglicher Ablauf einer Reihe im Anfangsunterricht

geben. Ein Schüler muss also zuerst den Attributwert auflösen, dann kann er das Argument übergeben.

5.4 Vierte Sequenz - Objekte als Argumente und Parameter

Ziel dieser Sequenz: Die Schüler verstehen auch Objektreferenzen als Werte, die als Argument zu übergeben sind und als Parameter weiterverarbeitet werden können. Sie können Objektreferenzen übergeben, um einfache Probleme zu lösen

Möglicher Ablauf: Die Schüler erarbeiten im Rollenspiel, dass die auf ihren Objektkarten eingetragenen Objektreferenzen auch als Parameter zu übergeben sind. Sie verändern durch geeignete Aktionen die Struktur des Objektdiagramms.

5.5 Fünfte Sequenz - Kontrollstrukturen

Ziel dieser Sequenz: Die Schüler können Verzweigungen und Schleifen in Diensten anwenden.

Möglicher Ablauf: Die Schüler werden mit geeigneten Problemstellungen konfrontiert, die es erforderlich machen, Verzweigungen und Schleifen in Diensten anzuwenden. Sie formulieren diese Dienste als Struktogramm und wenden sie im Rollenspiel an. Für Verzweigungen eignet sich die Betrachtung von Randbedingungen, etwa: »Erhöhe einen Wert nur, wenn er noch unter einem Grenzwert liegt.« Auch für Schleifen sollte man ein Beispiel finden, das in einem normalen Dienst Anwendung findet.²⁸ Im Anhang A finden sich weitere Vorschläge für die Betrachtung von Kontrollstrukturen.

Auch Verzweigungen und Schleifen sind für Schüler ähnliche Konzepte. Deshalb kann auch hier das Phänomen der Ähnlichkeitshemmung auftreten, wenn man die beiden Konzepte zeitnah zueinander einführt. Sie werden hier nur aufgrund des logischen Zusammenhangs in einer Sequenz dargestellt. Man sollte zwischen den beiden Konzepten andere Inhalte platzieren. Es bietet sich ein

²⁸Im Lehrbuch [Sch05] wird die Schleife ausschließlich verwendet, um die Benutzereingaben so lange zu verarbeiten, bis das Programm beendet wird. Dies verkürzt meines Erachtens den Schleifenbegriff.

Exkurs in ein anderes Themengebiet der Informatik oder eine erste Implementierungsphase an.

5.6 Sechste Sequenz - Klassenbegriff

Ziel dieser Sequenz: Die Schüler lernen den Klassenbegriff kennen. Sie können gleichartige Objekte zu Klassen zusammenfassen und Objekte aus Klassen als Baupläne erzeugen.

Möglicher Ablauf: Der Klassenbegriff wird unabhängig vom Rollenspiel erarbeitet. Die Schüler sollen nun im Rollenspiel den Teil eines Programmablaufes durchspielen, der bisher schlicht als gegeben angenommen wurde: Bisher gab es als Vorlage für ein Rollenspiel ein Objektdiagramm, das durch die Schüler nach den gegebenen Regeln umgesetzt wurde. Die Situation wurde unter Umständen modifiziert, die Objekte aber existierten zu jedem Zeitpunkt. Es gab auch keine Möglichkeit, neue Objekte zu »erschaffen«.

Nun kann die »Entstehung« der Objekte ebenfalls dargestellt werden. Lediglich ein Objekt (ein »zentrales Steuerobjekt«) muss von Anfang an existieren oder durch einen Benutzer erstellt werden. Die Aufgabe des zentralen Objektes ist es nun, seine Bezugsobjekte zu erschaffen. In dem Moment, in dem ein Schüler Objekt wird, steht er auf und nimmt seine Objektkarte zur Hand. Die Objektkarten liegen, klassenweise geordnet, bereit. Den Schülern sollte verdeutlicht werden, dass mehrere Exemplare nur schon existieren, um das Rollenspiel zu vereinfachen. Streng genommen müsste aus einer Klassenkarte (eckige Ecken) mit jeder Objekterstellung eine Objektkarte (runde Ecken) durch Kopieren erstellt werden. Das neu erschaffene Objekt führt zuerst seinen Konstruktor (dieser Begriff wird hier eingeführt) aus. Darin werden unter Umständen weitere Bezugsobjekte erschaffen. Auch beim Aufruf des Konstruktors können Parameter übergeben werden. Dieser Aspekt muss eigens thematisiert werden.

6 Grenzen des Konzeptes

Wenn man Objektdiagramme mit Objektbezeichnern ohne in den Objektkarten notierte Bezugsobjekte direkt in ein Objektspiel umsetzt, so entsteht bei den Lernenden leicht der Eindruck, Objekte hätten global bekannte Bezeichner, mit deren Hilfe alle im Modell vorkommenden Objekte miteinander kommunizieren können. Das ist nicht der Fall. Objektbezeichner sind immer Bezeichner für eine Referenz, Objekte selbst haben, entgegen der häufig gewählten Darstellung in Objektkarten, keinen eigenen Bezeichner. Dieses Problem wird in diesem Konzept dadurch vermieden, dass jedes Objekt seine Bezugsobjekte unter einem anderen Bezeichner anspricht. Als Objektreferenzen werden die Namen der Schüler benutzt, die offenkundig keine Semantik in Bezug auf das aktuell modellierte Problem haben. Auch hier muss aber ausdrücklich betont werden, dass Kommunikation mit einem Schüler (einem Objekt) nur stattfinden kann, wenn eine entsprechende Beziehung auf der eigenen Objektkarte notiert ist und ein Bezeichner für eine Objektreferenz in einen Namen aufgelöst werden kann. Als Schüler (nicht als Objekt) kennt natürlich jeder Schüler die Namen seiner Mitschüler.

Je offener man die Kommunikation der Objekte untereinander gestaltet (mündlich statt schriftlich und damit für alle mitzuverfolgen), desto mehr erhalten alle Schüler eine Draufsicht auf das modellierte Problem. Sie nehmen, auch unabhängig von der Kommunikation, immer alle Beteiligten wahr, da sie die im Raum existierenden »Objekte« sehen. Hier muss thematisiert werden, dass sie dies als Schüler können, jedoch nicht als Objekte.²⁹ Eine Draufsicht ist natürlich teilweise auch wünschenswert, wenn man sich als Entwickler einen Gesamtüberblick über die Modellierung verschaffen möchte. Unter Umständen wird aber nicht ausreichend deutlich, dass die Objekte diesen Überblick nicht haben, sondern gekapselt nur auf Aufträge und Anfragen reagieren und nach deren Abarbeitung wieder vollkommen inaktiv sind.

Bei größeren Modellierungen, an denen viele Objekte beteiligt sind, müssen auch viele Schüler beteiligt sein, um die Modellierung im Rollenspiel darzustellen. Es entsteht dadurch das Problem, dass Schüler eventuell längere Zeit nur passiv sind. Gerade wenn man darauf achtet, dass alle Zustandsänderungen sorgfältig auf den Objektkarten notiert werden, und dafür Zeit in Anspruch genommen wird, kann bei nicht beteiligten Schülern, wie bereits erwähnt, leicht Langeweile entstehen.

²⁹Diethelm (vgl. [Die07b]) arbeitet mit verbundenen Augen. Dies schließt sich hier aber aus, da die Schüler Information auf ihren Objektkarten notieren müssen.

6 Grenzen des Konzeptes

Aus den erforderlichen großen Gruppengrößen resultiert auch das Problem, dass gegebenenfalls pädagogisch sinnvolle Gruppengrößen von vier bis fünf Schülern überschritten werden. Die Gruppen können dann Modellierungen zwar gut darstellen, sind aber zu groß, um Inhalte in Gruppenarbeit zu erarbeiten. Diesem Problem kann zum Beispiel begegnet werden, indem man die Gruppen zur Erarbeitung neuer Inhalte teilt und dann die Lösungen der verschiedenen Teilgruppen von der Großgruppe durchspielen und eventuell diskutieren lässt.

7 **Praktische Erfahrungen aus dem Unterricht**

Das hier beschriebene Konzept habe ich in dieser Form bisher nicht im Unterricht umgesetzt. Es bezieht sich auf den Anfangsunterricht in der Jahrgangsstufe 11. Einen solchen Kurs habe ich zwar im Rahmen des selbstständigen Unterrichts unterrichtet, die Idee für dieses Konzept hat sich aber erst im Laufe des Unterrichts ergeben, wie auch in der Einleitung dieser Arbeit beschrieben.

Dennoch habe ich Objektspiele immer wieder im Unterricht durchgeführt. Den Schülern wird durch diese Methode bewusst, wie autonome Objekte interagieren. Dass ihnen die Methode beim Verständnis der objektorientierten Denkweise hilft, melden sie immer wieder zurück.

Als großen Vorteil im Unterricht habe ich es empfunden, dass die Methode im Kleinen immer wieder aufgreifbar ist. In Implementierungsphasen treten typische Schwierigkeiten auf. So werden etwa Argumente vergessen oder es ist nicht klar, wie verschiedene Objekte zueinander in Beziehung stehen. Der Lehrer kann dann den oder die Schüler »zu einem Objekt machen«, selbst »zum Objekt werden« und kurze Ausschnitte aus einem Objektspiel durchspielen. Den Schülern werden bestimmte Inhalte schnell wieder klar.³⁰

Die Erweiterung der Objektkarte um die Bezeichner für die Bezugsobjekte wurde von den Schülern offenbar gut verstanden und als hilfreich empfunden. Mein Kurs hat beide Darstellungsarten kennen gelernt. Noch einfacher wäre es für die Schüler, wenn sie nur die Darstellungsart mit Bezugsobjekten kennen gelernt hätten.

³⁰Ein Beispiel für eine solche Kommunikation: Der Schüler hat beim Aufruf des Dienstes `hat-Kreis.bewegeBis()` die Argumente vergessen. Auch nach einem einfachen Hinweis kommt er nicht auf diese Tatsache. Lehrer: »Ich bin das Programmobjekt, du bist der Kreis, es besteht eine `hat-` Beziehung von mir zu dir. Ich erteile dir den Auftrag: `>bewegeBis<`. Was musst du jetzt wissen?« Schüler: »Wohin soll ich mich bewegen? - Argumente!«

8 Evaluationsmöglichkeiten

Die Evaluation des dargestellten Konzeptes ist nicht zentraler Bestandteil dieser Arbeit. Um aber die Wirksamkeit der Objektspielmethode zu überprüfen, wäre eine geeignete Evaluation notwendig.

Eine wissenschaftlich korrekte Vorgehensweise würde es erfordern, zahlreiche Kurse mit und ohne Objektspiele in die objektorientierte Modellierung einzuführen und im Anschluss mit einem geeigneten Instrument zu erfassen, welche Gruppe die Inhalte besser verstanden hat.

Ein solches Vorgehen ist im Rahmen einer zweiten Staatsarbeit nicht möglich.

Innerhalb eines Kurses könnte man aber Elemente einer solchen Evaluation durchführen, indem man den Kurs teilt und einzelne Aspekte durch eine Gruppe mit Hilfe von Objektspielen, durch eine andere Gruppe etwa nur durch das Zeichnen von Objektdiagrammen erarbeiten lässt. Man kann dann erfassen, welche Teilgruppe danach gestellte Aufgaben besser erledigt. Da man als Lehrer aber für den Lernerfolg der ganzen Gruppe verantwortlich ist, müsste man die erfolgreichere Methode im Anschluss mit allen Schülern durchführen. Ausserdem sind Ergebnisse nicht gegen den Zufall abzusichern.

9 Lehrerfunktionen

Diese Arbeit berührt verschiedene Lehrerfunktionen: Zunächst wird ein Problem der pädagogischen Praxis beschrieben. Dies beruht auf einer entsprechenden Diagnose. Insofern spielt die Lehrerfunktion »Diagnostizieren« für diese Arbeit eine Rolle.

Zentraler ist die Lehrerfunktion des Innovierens. In dieser Arbeit wird ein Konzept aufgezeigt, dass in der Literatur so bisher nicht beschrieben wurde. Neu sind vor allem die Aspekte der in die Objektkarten aufgenommenen Objektreferenzen und der Objektkarten als »Notizzettel« für die objektspielenden Schüler. Dies betrifft insbesondere das Kapitel 4.

Die Lehrerfunktion des Unterrichtens ist ebenfalls zentral, da auf konkrete Umsetzungsmöglichkeiten des erarbeiteten Konzeptes hingewiesen wird. In Kapitel 5 wird dazu eine mögliche Unterrichtsreihe in Sequenzen beschrieben.

Da Evaluationsmöglichkeiten des Konzeptes zumindest angedeutet wurden, ist auch die Lehrerfunktion der Evaluation berührt (Kapitel 8).

10 Ausblick, Fazit

Die vorliegende Arbeit hat gezeigt, dass sich eine Einführung in die objektorientierte Analyse und Modellierung erstens handlungsorientiert und zweitens ohne das gleichzeitige Erlernen einer konkreten Programmiersprache gestalten lässt. Meine Erfahrungen aus dem Unterricht lassen zudem den Schluss zu, dass das zusätzliche Erlernen der gewünschten Programmiersprache einfacher gelingt, wenn das »objektorientierte Geschehen«, also der Ablauf einer objektorientiert modellierten und implementierten Software, zuvor handelnd erlebt wurde. Ohne Objektspiele bleibt das, was »im Rechner« passiert, abstrakt. Die Vorstellung von Objekten als autonome Einheiten muss vom Schüler im mentalen Modell selbst gebildet werden. Mit Objektspielen haben die Schüler ein konkretes Geschehen vor Augen, auf dem sie ihr mentales Modell aufbauen können.

Eine geeignete Evaluation (siehe Kapitel 8) müsste zeigen, ob das Konzept tatsächlich zu besseren Lernerfolgen bei den Schülern führt, und ob dies mit einem vertretbaren Zeitaufwand geschieht.

Der Schritt hin zur Programmiersprache erfordert dann, neben der Einführung des Klassenbegriffes, das Erlernen der Syntax einer Programmiersprache. Da auch die Bezugsobjekte auf den Objektkarten (und Klassenkarten) dargestellt wurden, fällt der Schritt zu einer Notation in der Programmiersprache vermutlich leicht. Alle Elemente sind bekannt, sie müssen nur noch in der richtigen Syntax eingegeben werden.

Wichtig ist, den Schülern im Anfangsunterricht geeignete Modellierungsbeispiele vorzulegen. Im Anhang findet sich ein Beispiel für eine Problemstellung, an dem die in Kapitel 5 dargestellten Sequenzen abgearbeitet werden können. Für ein intensives Üben sind aber weitere Beispiele, die immer wieder in den Unterrichtsverlauf eingebaut werden, notwendig. Hier gilt es, geeignete Beispiele zu entwickeln.

Das Konzept erfordert einen relativ hohen Materialaufwand (Objekt- und Dienstekarten, Klemmbretter, Stifte). Besonders die Objekt- und Dienstekarten bedeuten einen intensiven Vorbereitungsaufwand. Sie müssen für jedes Modellierungsbeispiel neu erstellt werden. Es wäre deshalb wünschenswert, wenn es eine Plattform gäbe, auf der erprobte und nicht mehr der Verbesserung bedürftige Materialien unter Kollegen ausgetauscht werden könnten. So könnte der Aufwand, der mit dem Einsatz der Methode für den Einzelnen verbunden ist, minimiert werden.

Anhang

A Hinweise zur Modellierungsaufgabe

Hier wird ein Beispiel vorgestellt, an dem sich die im Hauptteil der Arbeit, in Kapitel 5 aufgezeigten Sequenzen verwirklichen lassen.

Ausgegangen wird vom Realitätsausschnitt »MauMauSpiel«. Dieses soll objektorientiert modelliert und im Rollenspiel dargestellt werden.

Hier finden sich die Materialien, die für die Umsetzung dieses Beispiel im Objekt-Rollenspiel benötigt werden, sowie entsprechende Aufgabenstellungen.

Das Mau-Mau-Spiel wird für die Modellierung stark vereinfacht:

- Es spielen genau zwei Spieler.
- Es gibt nur vier Karten: Zwei Farben mit jeweils zwei Werten. (z.B. Pik sieben und acht sowie Herz sieben und acht)
- Sonderregeln, wie »sieben = zwei Karten nehmen« oder ähnliche gibt es nicht.
- Jeder Spieler bekommt zu Spielbeginn genau eine Karte ausgeteilt.

Teilweise fassen die hier im Anhang zu findenden Materialien (Objektdiagramme, Objektkarten und Dienstekarten für die Objekte) mehrere im Unterricht benötigte Materialien zusammen. Es sind zum Beispiel auf den Dienstekarten alle Dienste dargestellt, die im Verlauf der ganzen Reihe benötigt werden. Für die Schüler sollten hier für die verschiedenen Aufgaben verschiedene passende Versionen erstellt werden.

Zur Sozialform (Einzel-, Partner-, Gruppenarbeit) mache ich nur Angaben, wenn aufgrund des zu bearbeitenden Inhalts bestimmte Formen notwendig sind. So erfordert das Objektspiel bestimmte Gruppengrößen, da die Anzahl der beteiligten Objekte vorgegeben ist. Wenn der Kurs sich nicht entsprechend aufteilen lässt, so muss ein Teil der Schüler »Publikum« sein.

Sequenz 1 - Objektdiagramm

- Die Schüler spielen in Paaren MauMau mit den vereinfachten Regeln, um zu erfahren, dass das Spiel auch so stark vereinfacht noch spielbar ist.

A Hinweise zur Modellierungsaufgabe

- Die Schüler identifizieren am Spiel beteiligte Objekte: Vier Karten, zwei Spieler, ein Stapel zum Ablegen, ein Stapel zum Aufnehmen.
- Im Unterrichtsgespräch werden Attribute, Dienste und Objektbeziehungen identifiziert. Die Schüler lernen die Darstellung im Objektdiagramm kennen. Ein vollständiges Objektdiagramm findet sich im Anhang G, es berücksichtigt einige (unrealistische und hier noch nicht relevante) Notwendigkeiten, die in der zweiten Sequenz wichtig werden. Diese Attribute und Dienste sind eingeklammert dargestellt und werden an dieser Stelle noch nicht erarbeitet.
- Das Objektdiagramm wird sowohl mit Pfeilen als auch mit textuellen Referenzen dargestellt. Die gleichwertige Semantik dieser beiden Darstellungsarten wird deutlich gemacht.

Aufgabenblatt 1, das die Schüler zum Objektdiagramm hinführt, findet sich im Anhang B.

Sequenz 2 - Objektspiele

- Die Schüler erfahren handelnd, dass alle Objekte im Objektrollenspiel durch Schüler darstellbar sind. In Gruppen zu acht Schülern erhalten sie alle notwendigen Materialien (Objektkarten auf Klemmbrettern und Stifte). Sie stellen die im Objektdiagramm skizzierte Situation dar. Kommunikation zwischen den Objekten soll noch nicht stattfinden. Statt der textuellen Referenz (etwa »karteA«) werden jetzt die Namen eingetragen. Es geht darum, zu erkennen, dass auch diese Darstellung äquivalent ist.
- In Dreiergruppen wird ein anderer möglicher Ausschnitt aus dem Objektdiagramm betrachtet: Ein Spieler hält zwei Karten auf der Hand. Um zunächst parameterlose Aufträge zu betrachten, wird folgende (völlig unrealistische) zusätzliche Eigenschaft von Karten betrachtet: »Eine Karte hat eine Beleuchtung. Man kann sie ein- und ausschalten.«
- Die Schüler spielen gemäß Aufgabenblatt verschiedene Situationen durch. Sie müssen teilweise Dienste selbst formulieren.

Die Aufgabenblätter 2 und 3 können den Schülern als Arbeitsauftrag dienen. Die Objektkarten und Dienstekarten für Aufgabenblatt 3 finden sich im Anhang H, I,

J und K. Die kursiv dargestellten Teile sind dort von den Schülern handschriftlich einzutragen. Sie dienen hier nur der Verdeutlichung. Die Objektkarten für Aufgabenblatt 2 müssten entsprechend gestaltet werden.

Sequenz 3 - Parameter und Argumente

- Wiederum in Gruppen zu drei Schülern sollen sie nun erfahren, dass es Dienste gibt, mit deren Aufruf ein Argument übergeben werden muss. Sie betrachten dazu die Dienste »setzeWert« und »setzeFarbe« der Karte- Objekte. Auch diese Dienste sind eigentlich nicht realistisch, da Wert und Farbe einer Karte nicht verändert werden können. Sie werden hier dennoch betrachtet, um das Lernziel zu erreichen.
- Die Schüler erkennen, dass die genannten Dienste es erfordern, zusätzliche Informationen zu übergeben. Einige Gruppen werden vermutlich intuitiv so etwas wie ein Argument übergeben. Ein anderer Lösungsvorschlag der Schüler wird vermutlich sein, viele Dienste, wie »setzeFarbeAufPik«, »setzeWertAuf3« etc. einzuführen. Die Nachteile dieses Vorschlags (Aufwändig, nicht allgemein möglich) können von den Schülern verstanden und formuliert werden.
- Die Schüler lernen die Begriffe Argument und Parameter kennen.
- Die Dienste werden nun noch einmal korrekt formuliert. Danach spielen die Gruppen sie noch einmal durch.

Aufgabenblatt 4 dient dazu, den Schülern die Notwendigkeit von Argumenten und Parametern bewusst zu machen. Auf den Dienstekarten sind die Dienste formal korrekt notiert.

- Nun wird für das Spieler- Objekt das zusätzliche Attribut »lieblingsfarbe« eingeführt. Dieses kann die Werte »PIK« und »HERZ« annehmen. Die Schüler erhalten in den Gruppen die Aufgabe, einen Dienst »setzeKartenAufLieblingsfarbe« für das Spielerobjekt zu erstellen. Hier muss ein Attribut als Argument übergeben werden. Ein Aufruf auf der Dienstekarte müsste dann lauten: »Ich gebe meinem Bezugsobjekt **meineKarte1** den Auftrag **setzeFarbe** mit dem Wert meines Attributes **lieblingsfarbe** als Argument.«

Sequenz 4 - Objektreferenzen als Argument und Parameter

- Die Schüler sollen nun handelnd erfahren, dass auch die Information, die eine Objektreferenz enthält, als Argument übergeben werden kann.

Aufgabenblatt 5 führt die Schüler zu dieser Erkenntnis. Der Dienst »nimmKarte« des Objektes »abStapel« ist vorgegeben. Er enthält eine Idee, wie auch der ziehe-Dienst realisiert werden kann.

- Die erste Version des ziehe-Dienstes enthält nun vermutlich zunächst zwei Anweisungen: 1) Ich gebe meinem Bezugsobjekt **meinAbStapel** den Auftrag **nimmKarte** mit dem Wert meiner Objektreferenz **meineKarte1** als Argument. 2) Ich gebe meinem Bezugsobjekt **meinMitspieler** den Auftrag **ziehe**.
- Nun entstehen mehrere Probleme, die von den Schülern erkannt und mit ihnen diskutiert werden müssen:
 - Nachdem die Objektreferenz übergeben wurde, haben zwei Objekte eine Beziehung zu der Karte: `spielerA/B` und `abStapel`. Das ist unrealistisch.
 - Lösung: Die Objektreferenz der gerade abgelegten Karte muss auf »null«³¹ gesetzt werden.
 - Wenn die Objektreferenz auf »null« gesetzt wurde, so kann der Spieler nicht mit derselben Anweisung die zweite Karte ablegen, denn die Beziehung zu dieser Karte hat er mit einer anderen Objektreferenz.
 - Lösung: Nach dem Ablegen von `meineKarte1` wird `meineKarte2` zu `meineKarte1` und die Objektreferenz `meineKarte2` wird auf »null« gesetzt.
- Der Dienst wird formal korrekt notiert. Die Schüler benutzen den Dienst im Objektspiel. Sie bilden dazu Gruppen zu je acht Schülern.

Sequenz 5 - Kontrollstrukturen

- Die normale Regel, dass Karten nur »passend« abgelegt werden dürfen, wird wieder eingeführt. Mit den Schülern wird diskutiert, wie sich der ziehe-Dienst

³¹Der Begriff »null« für eine Referenz ins »nichts« muss hier eingeführt werden. Man kann auch den entsprechenden Begriff aus der Syntax einer anderen Programmiersprache oder einen geeigneten programmiersprachenunabhängigen Begriff verwenden. Ich verwende hier den Begriff »null« aus der Sprache Java.

A Hinweise zur Modellierungsaufgabe

nun verändern müsste. Die Schüler formulieren hier vermutlich intuitiv wenn-dann-sonst-Beziehungen.

- Die formale Darstellung der Verzweigung im Struktogramm wird eingeführt.
- Die Verzweigung muss gegebenenfalls anhand einfacher Beispiele geübt werden.
- Die Schüler erstellen einen vollständigen ziehe-Dienst. Die anderen notwendigen Dienste der Objekte »stapelAuf« (gibKarte) und »stapelAb« (»getWertOberste«, »getFarbeOberste«) werden vorgegeben.

Ein vollständiger ziehe-Dienst ist auf der Dienstekarte angegeben. Sie ist durch den Einsatz von Pseudocode vereinfacht dargestellt, da die vollständige Formulierung in der gewohnten »ich-Syntax« sehr lang ist. Eine weitere Vereinfachung gibt es beim Dienst »gibKarte« des Objektes »stapelAuf«. Hier sind zwei Versionen angegeben, die einfachere funktioniert dann nicht mehr, wenn der Stapel leer ist, und nach den Spielregeln eigentlich Karten vom anderen Stapel nachgenommen werden müssten. Die schwierigere Version des Dienstes realisiert das, hier wird allerdings schon eine Schleife benötigt. Das Problem ließe sich umgehen, wenn man hier eine nicht-streng-algorithmische Handlungsanweisung formuliert, die dafür sorgt, dass der Stapel sich mit neuen Karten versorgt.

Als nächste Kontrollstruktur würde nun die Schleife eingeführt. Um das Phänomen der Ähnlichkeitshemmung bezüglich der Konzepte »Verzweigung« und »Schleife« zu vermeiden, empfiehlt es sich, die beiden Konzepte zeitlich voneinander getrennt einzuführen. Man sollte also nicht direkt nach der Einführung der Verzweigung die Schleife einführen. An dieser Stelle bietet sich ein Exkurs an, z.B. zu einem Thema aus dem Bereich »Informatik und Gesellschaft«. Eine andere Möglichkeit wäre, an dieser Stelle in die Implementierung mit einer konkreten Sprache einzusteigen. Dann müsste schon jetzt der Klassenbegriff eingeführt werden, das Spiel könnte man dann (evtl. mit durch den Lehrer vorgegebenen Teilen) implementieren.

- Zur Einführung der Schleife könnte man eine Regeländerung betrachten: Kann ein Spieler nicht ablegen, so muss er so lange Karten vom Stapel aufnehmen, bis er ablegen kann. Der ziehe-Dienst kann man dann entsprechend verändern. Damit dies sinnvoll wird, müsste man das Beispiel aber zuvor auf mehr als vier Karten erweitern.

- Eine andere Möglichkeit ist die Betrachtung des oben beschriebenen schwierigen gibKarte-Dienstes. Hier werden allerdings Verzweigung und Schleife gleichzeitig benötigt.

Sequenz 6 - Klassenbegriff

Ausgehend vom Objektdiagramm wird der Klassenbegriff eingeführt. Die Schüler identifizieren die Klassen »Spieler«, »Karte«, »StapelAuf« und »StapelAb«. Die Notationsform »Klassendiagramm« wird eingeführt. Dabei werden die Objektreferenzen zunächst auch im Klassendiagramm eingetragen. Es entstehen parallele Verbindungen zwischen den Klassen. Von da aus kann dann weiter abstrahiert werden, indem man die Notation der Objektreferenzen weglässt und die parallelen Kanten zusammenfasst.

Außerdem müssen nun einige andere Änderungen und Ergänzungen eingeführt werden:

- Es muss ein weiteres Objekt (und damit eine weitere Klasse) eingeführt werden - ein zentrales Steuerobjekt. Es kann z.B. mit »Spiel« bezeichnet werden. Dieses Objekt soll zunächst wieder von Anfang an existieren. Es hat einen Dienst »baueSpielAuf«, die es von allein aufruft. Dieser Dienst findet sich auf der Dienstekarte im Anhang N. Hier werden alle erforderlichen Objekte erzeugt, die Objektbeziehungen werden hergestellt. An dieser Stelle wird noch darauf verzichtet, die Karten zu mischen, d.h. zufällig zu verteilen. Dieser Aspekt kann später hinzugefügt werden. Er erfordert den Einsatz weiterer Verzweigungen und eines Konzeptes zur Zufallssteuerung. Desweiteren brauchen die Spieler-Objekte einen neuen Dienst zum »kennenlernen« des Mitspielers. Dies kann wegen der Symmetrie nicht schon im Konstruktor geschehen.
- Der Begriff des Konstruktors muss eingeführt werden. Es werden Konstrukto- ren für die einzelnen Klassen erarbeitet.
- Man kann an dieser Stelle den Unterschied zwischen hat- und kennt-Beziehungen thematisieren. Dann müssen die Objektbeziehungen hier untersucht und ggf. umbenannt werden. Hat-Beziehungen sind diejenigen, die nach dem Erzeugen eines neuen Bezugsobjektes entstehen. Alle Beziehungen, die durch Weitergabe einer Objektreferenz geknüpft werden, sind kennt-Beziehungen.

A Hinweise zur Modellierungsaufgabe

- Auch auf den Objektkarten sollte nun der Klassenname der Klasse, aus der das Objekt erstellt wurde, notiert werden. Eine Objektkarte ist nun also nicht mehr mit z.B. »melanie« überschrieben, sondern mit »melanie: Spieler«. In Objektdiagrammen (nicht im Objektspiel) kann man sogar ganz auf einen Objektbezeichner verzichten und eine Objektkarte mit z.B. »:Spieler« für »ein Objekt der Klasse Spieler« überschreiben.

Im Objektspiel wird das Entstehen eines neuen Objektes durch Aufstehen des Schülers dargestellt. Die Schüler spielen die neuen Dienste, also die Konstruktoren und die Dienste »baueSpielAuf« des Objektes »Spiel«, im Objektspiel durch.

B Aufgabenblatt 1

MauMau spielen

Spielt zu zweit das Kartenspiel MauMau! Dabei gelten folgende, stark vereinfachte Regeln:

- Es spielen genau zwei Spieler.
- Es gibt nur vier Karten: Zwei Farben mit jeweils zwei Werten. (z.B. Pik sieben und acht sowie Herz sieben und acht)
- Sonderregeln, wie »sieben = zwei Karten nehmen« o.ä. gibt es nicht.
- Jeder Spieler bekommt zu Spielbeginn genau eine Karte ausgeteilt.
- Der Aufnehm-Stapel und der Ablege-Stapel bestehen demnach jeweils aus einer Karte.

Informatische Modellierung

Dieses Kartenspiel soll nun mit den Werkzeugen der Informatik betrachtet und modelliert werden. Dabei wenden wir die sogenannte »objektorientierte Modellierungstechnik« an. Ein informatisches Modell hilft uns, später Programme zu schreiben, die sich auf ein Stück Realität beziehen.

Identifiziert die an dem Spiel beteiligten Objekte. Kandidaten für Objekte sind »Dinge«. In der Spielbeschreibung oder ähnlichen Texten werden oft Nomen dafür verwendet.

C Aufgabenblatt 2

Darstellung als Objektspiel

Stellt das gemeinsam erarbeitete Objektdiagramm nun in einem Objektspiel dar. Geht dabei vom Objektdiagramm mit textuellen Referenzen aus. Jedes Objekt (jeder Schüler) erhält dazu eine Objektkarte. Statt der Objektnamen müsst ihr nun eure Namen eintragen. (Achtung, doppelt vorkommende Namen müsst ihr eindeutig machen!) Ausserdem müsst ihr die Attributwerte eintragen.

Das von euch dargestellte Modell ist noch vollkommen statisch. Wir müssen jetzt daran arbeiten, wie wir im Objektspiel den Aufruf von Diensten darstellen können.

D Aufgabenblatt 3

Ein Ausschnitt aus dem Objektdiagramm

Für ein erstes Rollenspiel mit Objekten betrachtet nun einen Ausschnitt aus einem Objektdiagramm, das so im Laufe eines MauMau-Spiels entstehen könnte:

Ein Spieler hält genau zwei Karten auf der Hand: die »Pik sieben« und die »Pik acht«.

Außerdem nehmen wir folgende, unrealistische Veränderung der Situation an: Jede Karte hat eine Beleuchtung. Man kann sie ein- und ausschalten. (Wir brauchen das für den Lernfortschritt an dieser Stelle.)

- Welche zusätzlichen Attribute und Dienste braucht jede Karte nun?
- Verteilt die Rollen und tragt die Objektreferenzen und Attributwerte so in die großen Objektkarten ein, dass die beschriebene Situation repräsentiert wird.

Aufruf von Diensten

Für die beiden Karten- Objekte sind die Dienste (Aufträge) »beleuchtungEin« und »beleuchtungAus« vorgegeben. Für das Spieler- Objekt ist der Dienst (Auftrag) »schalteAlleKartenAn« vorgegeben.

- Geht von der Situation aus, die ihr gerade hergestellt habt. Der »Spieler« soll nun seinen Dienst »schalteAlleKartenAn« ausführen. Wer den Dienst beim Spieler aufruft, soll hier erst einmal nicht betrachtet werden. Er wird einfach »von allein« aktiv. Spielt das Rollenspiel entsprechend durch.

Für die beiden Karten- Objekte sind außerdem die Dienste (Anfragen) »getWert« und »getFarbe« vorgegeben. Für das Spieler- Objekt ist der Dienst (Auftrag) »berechneKartenwert« vorgegeben.

- Was ist der grundsätzliche Unterschied zwischen Aufträgen und Anfragen?
- Der »Spieler« soll nun seinen Dienst »berechneKartenwert« ausführen. Wer den Dienst beim Spieler aufruft, soll hier erst einmal nicht betrachtet werden. Er wird einfach »von allein« aktiv. Spielt das Rollenspiel entsprechend durch.

E Aufgabenblatt 4

Erste Veränderungen der Situation: Argumente und Parameter

Jetzt soll sich das Objektdiagramm zum ersten Mal verändern. Zuerst werden von diesen Veränderungen nur Attributwerte betroffen sein. An dieser Stelle müssen wir wieder etwas Unrealistisches annehmen: Karten können in Wert und Farbe verändert werden. Sie haben dazu je einen Dienst »setzeWert« und »setzeFarbe«.

- Stellt, wiederum in Gruppen zu drei Schülern, die bekannte Situation her: Ein Spieler hat zwei Karten auf der Hand. Benutzt dazu die entsprechenden Materialien (Objektkarten und Methodenkarten).
- Der Spieler soll nun einen Dienst »setzeAllePikSieben« ausführen. Darin muss beiden Karten der Auftrag »setzeFarbe« und »setzeWert« gegeben werden. Anders als bei den bisher betrachteten Diensten entsteht hier ein Problem. Formuliert dieses Problem!
- Macht euch außerdem Gedanken, wie dieses Problem zu lösen sein könnte. Was muss zusätzlich eingeführt werden?
- Macht einen Vorschlag, wie der Inhalt der Dienste »setzeAllePikSieben« (Spieler), »setzeFarbe« und »setzeWert« (Karten) aussehen könnte.

F Aufgabenblatt 5

Veränderungen der Objektbeziehungen

Nun soll sich unser Objektdiagramm stärker verändern. Ihr sollt euch Gedanken über den wichtigen Dienst »ziehe« machen. Wir nehmen dabei zunächst an, dass ein Spieler immer ablegen darf. Es gibt also nicht die Regel, dass man nur auf passende Farben beziehungsweise Werte abgelegt werden darf.

- Arbeitet bitte in Gruppen zu vier Schülern.
- Geht von folgender Ausgangssituation aus: `spielerA` hat zwei Karten auf der Hand, `spielerB` eine Karte, die vierte Karte liegt auf dem `stapelAb`. `stapelAuf` ist leer. Zeichnet ein Objektdiagramm mit allen acht Objekten, das diese Situation darstellt.
- Überlegt nun, was im `ziehe`-Dienst der Spieler-Objekte passieren muss. Notiert die Dienste auf der Dienstekarte der Spieler-Objekte. Bei euren Überlegungen könnt ihr davon ausgehen, dass `spielerA` an der Reihe ist. Das Objekt `abStapel` stellt den Dienst »`nimmKarte`« zur Verfügung. Dieser Dienst ist vorgegeben. Seht sie euch genau an.
- Versucht, den Dienst im Objektspiel zu benutzen. Tut euch dazu mit einer weiteren Gruppe aus vier Schülern zusammen, so dass jedes Objekt dargestellt werden kann.
- Zeichnet (wieder in den Vierergruppen) im Objektdiagramm auf, was in eurem Rollenspiel passiert ist. Benutzt ein neues Diagramm für jede neue Situation (veränderte Objektbeziehungen).

G Objektdiagramm »MauMauSpiel«

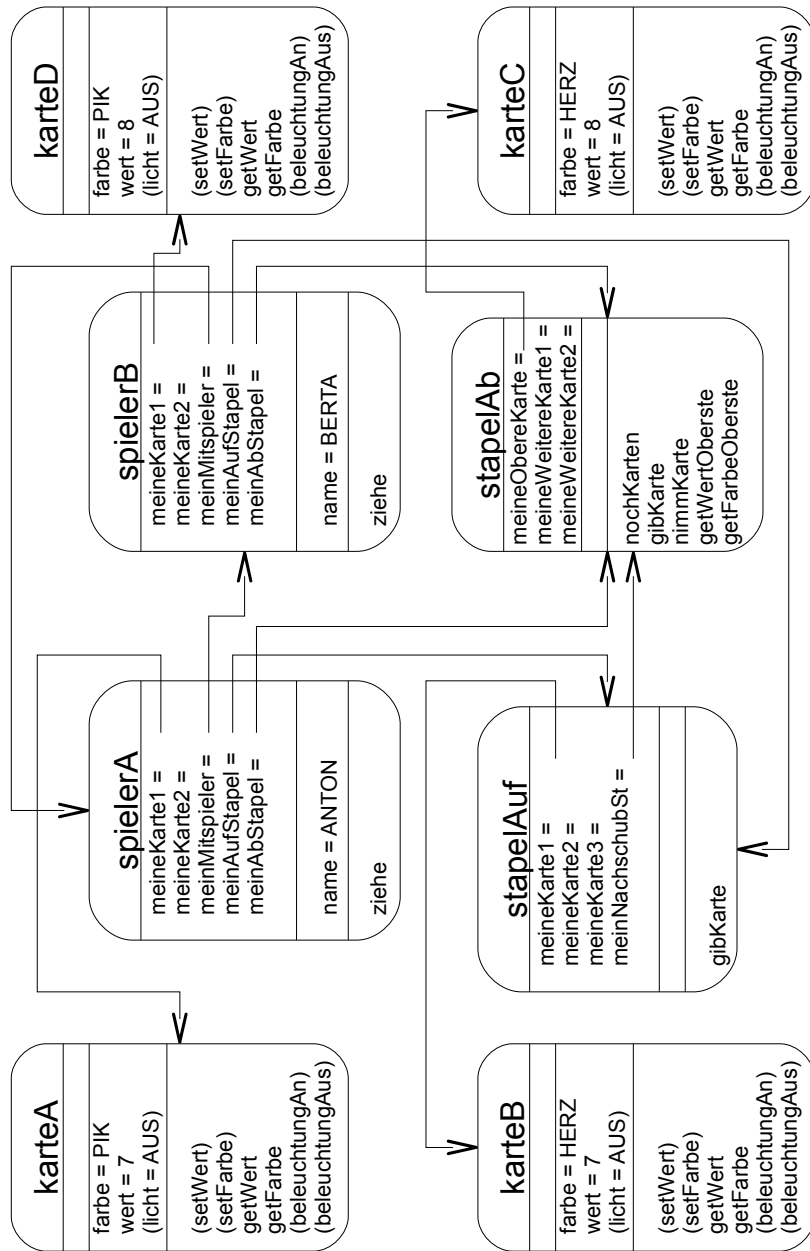


Abbildung 4: Objektdiagramm »MauMauSpiel«

H Objektkarte »spieler«

Die kursiv gesetzten Teile sind von den Schülern handschriftlich einzutragen und dienen hier nur der Verdeutlichung. Die Karte ist für das Objektspiel auf DIN A4 zu vergrößern.

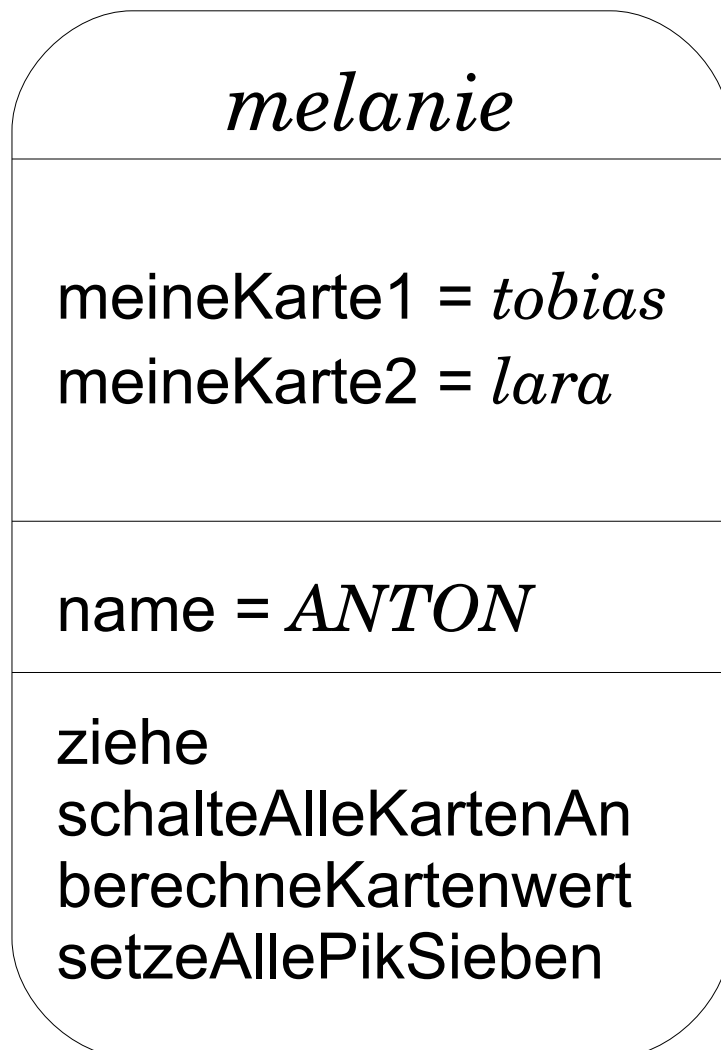


Abbildung 5: Objektkarte »Spieler«

I Dienstekarte »spieler«

I Dienstekarte »spieler«

Die Dienstekarten werden hinter die Objektkarten geheftet und den Schülern, die die entsprechenden Objekte darstellen, an die Hand gegeben.

Dienste Spieler-Objekt

Dienst **ziehe** (Version 1, zu Sequenz 4)

Ich gebe meinem Bezugsobjekt meinAbStapel den Auftrag nimmKarte mit dem Wert meiner Objektreferenz meineKarte1 als Argument.
Ich setze meine Objektreferenz meineKarte1 auf den Wert meiner Objektreferenz meineKarte2 .
Ich setze den Wert meiner Objektreferenz meineKarte2 auf null .
Ich gebe meinem Bezugsobjekt meinMitspieler den Auftrag ziehe .

Dienst **ziehe** (Version 2, zu Sequenz 5, durch Pseudocode und Punktnotation vereinfacht)

<i>Passt meineKarte1?</i>		
ja	nein	
meinAbStapel. nimmKarte(meineKarte1)	<i>Passt meineKarte2?</i>	
	ja	nein
meineKarte1 := meineKarte2	meinAbStapel. nimmKarte (meineKarte2)	meineKarte2 := meinAufStapel.gibKarte()
meineKarte2 := null	meineKarte2 := null	<i>Passt meineKarte2</i>
	ja	nein
	meinAbStapel. nimmKarte(meineKarte2)	meineKarte2 := null
meineKarte1 = null		
ja	nein	
Ich habe gewonnen. meinMitspieler.ziehe		

Dienst **schalteAlleKartenAn**

Ich gebe meinem Bezugsobjekt meineKarte1 den Auftrag beleuchtungEin .
Ich gebe meinem Bezugsobjekt meineKarte2 den Auftrag beleuchtungEin .

Dienst **berechneKartenwert**

Ich berechne die Summe aus dem Ergebnis der Anfrage getWert an mein Bezugsobjekt meineKarte1 und dem Ergebnis der Anfrage getWert an mein Bezugsobjekt meineKarte2 .

Dienst **setzeAllePikSieben**

Ich gebe meinem Bezugsobjekt meineKarte1 den Auftrag setzeWert mit dem Argument 7 .
Ich gebe meinem Bezugsobjekt meineKarte1 den Auftrag setzeFarbe mit dem Argument PIK .
Ich gebe meinem Bezugsobjekt meineKarte2 den Auftrag setzeWert mit dem Argument 7 .
Ich gebe meinem Bezugsobjekt meineKarte2 den Auftrag setzeWert mit dem Argument PIK .

Abbildung 6: Dienstekarte »spieler«

J Objektkarte »karte«

Die kursiv gesetzten Teile sind von den Schülern handschriftlich einzutragen und dienen hier nur der Verdeutlichung. Die Karte ist für das Objektspiel auf DIN A4 zu vergrößern.

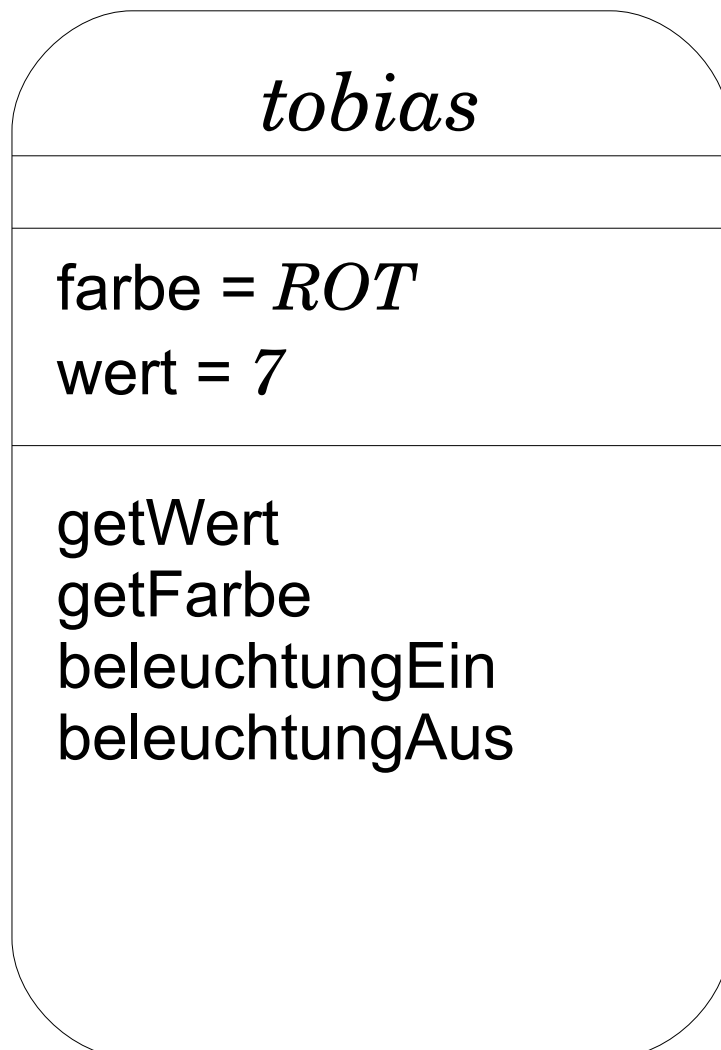


Abbildung 7: Objektkarte »Karte«

K Dienstekarte »karte«

Dienste Karte-Objekt

Dienst **getWert**

Ich **antworte** mit dem Wert meines **Attributes wert**.

Dienst **getFarbe**

Ich **antworte** mit dem Wert meines **Attributes farbe**.

Dienst **beleuchtungEin**

Ich **setze** den Wert meines **Attributes beleuchtung** auf **EIN**.

Dienst **beleuchtungAus**

Ich **setze** den Wert meines **Attributes beleuchtung** auf **AUS**.

Dienst **setFarbe (Parameter neueFarbe)**

Ich **setze** den Wert meines **Attributes farbe** auf den im **Parameter neueFarbe** übergebenen Wert.

Dienst **setWert (Parameter neuerWert)**

Ich **setze** den Wert meines **Attributes wert** auf den im **Parameter neuerWert** übergebenen Wert.

Abbildung 8: Dienstekarte »karte«

L Dienstekarte »aufStapel«

Dienste aufStapel-Objekt

Dienst **gibKarte** (einfach)

Ich merke mir dem Wert meiner Objektreferenz meineKarte1 .
Ich setze meine Objektreferenz meineKarte1 auf den Wert meiner Objektreferenz meineKarte2 .
Ich setze meine Objektreferenz meineKarte2 auf den Wert meiner Objektreferenz meineKarte3 .
Ich antworte mit dem Wert, den ich mir gemerkt habe.

Dienst **gibKarte** (schwierig)

Der Wert meiner Objektreferenz meineKarte1 ist null ?	
ja	nein
Die Anfrage an mein Bezugsobjekt meinNachschubSt, nochKarten , wird mit ja beantwortet?	
Ich setze den Wert meiner Objektreferenz meineKarte3 auf den Wert meiner Objektreferenz meineKarte2 .	
Ich setze den Wert meiner Objektreferenz meineKarte2 auf den Wert meiner Objektreferenz meineKarte1 .	
Ich setze den Wert meiner Objektreferenz meineKarte1 auf den Wert, den ich aufgrund der Anfrage an mein Bezugsobjekt meinNachschubSt, gibKarte , erhalte.	
Ich merke mir dem Wert meiner Objektreferenz meineKarte1 .	
Ich setze meine Objektreferenz meineKarte1 auf den Wert meiner Objektreferenz meineKarte2 .	
Ich setze meine Objektreferenz meineKarte2 auf den Wert meiner Objektreferenz meineKarte3 .	
Ich antworte mit dem Wert, den ich mir gemerkt habe.	

Abbildung 9: Dienstekarte »aufStapel«

M Dienstekarte »abStapel«

Dienste abStapel-Objekt

Dienst **nimmKarte** (Parameter **neueObereKarte**)

Ich setze meine Objektreferenz meineWeitereKarte2 auf den Wert meiner Objektreferenz meineWeitereKarte1 .
Ich setze meine Objektreferenz meineWeitereKarte1 auf den Wert meiner Objektreferenz meineObereKarte .
Ich setze meine Objektreferenz meineObereKarte auf den im Parameter neueObereKarte übergebenen Wert.

Dienst **getWertOberste**

Ich antworte mit dem Wert, den ich von meinem Bezugsobjekt meineObereKarte durch die Anfrage getWert erhalte.
--

Dienst **getFarbeOberste**

Ich antworte mit dem Wert, den ich von meinem Bezugsobjekt meineObereKarte durch die Anfrage getFarbe erhalte.

Dienst **gibKarte**

Ich merke mir dem Wert meiner Objektreferenz meineWeitereKarte1 .
Ich setze meine Objektreferenz meineWeitereKarte1 auf den Wert meiner Objektreferenz meineWeitereKarte2 .
Ich antworte mit dem Wert, den ich mir gemerkt habe.

Dienst **nochKarten**

Ist der Wert meiner Objektreferenz meineWeitereKarte1 gleich null ?	
ja	nein
Ich antworte mit nein	Ich antworte mit ja

Abbildung 10: Dienstekarte »abStapel«

N Dienstekarte »spiel«

Dienste spiel-Objekt

Dienst **baueSpielAuf**

Ich setze den Wert meiner Objektreferenz meineKarte1 auf die Referenz zu einem von mir neu erschaffenen Objekt der Klasse Karte . Ich rufe den Konstruktor auf mit den Argumenten 7 und PIK .
Ich setze den Wert meiner Objektreferenz meineKarte2 auf die Referenz zu einem von mir neu erschaffenen Objekt der Klasse Karte . Ich rufe den Konstruktor auf mit den Argumenten 8 und PIK .
Ich setze den Wert meiner Objektreferenz meineKarte3 auf die Referenz zu einem von mir neu erschaffenen Objekt der Klasse Karte . Ich rufe den Konstruktor auf mit den Argumenten 7 und HERZ .
Ich setze den Wert meiner Objektreferenz meineKarte4 auf die Referenz zu einem von mir neu erschaffenen Objekt der Klasse Karte . Ich rufe den Konstruktor auf mit den Argumenten 8 und HERZ .
Ich setze den Wert meiner Objektreferenz meinAufStapel auf die Referenz zu einem von mir neu erschaffenen Objekt der Klasse StapelAuf . Ich rufe den Konstruktor mit dem Wert meiner Objektreferenz meineKarte3 als Argument auf.
Ich setze den Wert meiner Objektreferenz meinAbStapel auf die Referenz zu einem von mir neu erschaffenen Objekt der Klasse StapelAb . Ich rufe den Konstruktor mit dem Wert meiner Objektreferenz meineKarte4 als Argument auf.
Ich setze den Wert meiner Objektreferenz meinSpieler1 auf die Referenz zu einem von mir neu erschaffenen Objekt der Klasse Spieler . Ich rufe den Konstruktor auf mit den Argumenten ANTON , den Werten meiner Objektreferenzen meineKarte1 , meinAbStapel , meinAufStapel .
Ich setze den Wert meiner Objektreferenz meinSpieler2 auf die Referenz zu einem von mir neu erschaffenen Objekt der Klasse Spieler . Ich rufe den Konstruktor auf mit den Argumenten BERTA , den Werten meiner Objektreferenzen meineKarte2 , meinAbStapel , meinAufStapel .
Ich gebe meinem Bezugsobjekt meinSpieler1 den Auftrag lerneMitspielerKennen mit dem Wert meiner Objektreferenz meinSpieler2 als Argument.
Ich gebe meinem Bezugsobjekt meinSpieler2 den Auftrag lerneMitspielerKennen mit dem Wert meiner Objektreferenz meinSpieler1 als Argument.

Abbildung 11: Dienstekarte »spiel«

Abbildungsverzeichnis

1	Objektdiagramm mit eingetragenen Referenzen	15
2	Objektdiagramm mit Beziehungspfeilen	15
3	Objektdiagramm mit Schülernamen als Referenzen	18
4	Objektdiagramm »MauMauSpiel«	46
5	Objektkarte »Spieler«	47
6	Dienstekarte »spieler«	48
7	Objektkarte »Karte«	49
8	Dienstekarte »karte«	50
9	Dienstekarte »aufStapel«	51
10	Dienstekarte »abStapel«	52
11	Dienstekarte »spiel«	53

Literatur

- [Abb83] Abbott, Russell J.: *Program Design by Informal English Descriptions*. In: *Comm. ACM* 26 (1983), November, Nr. 11, S. 882-894
- [Ber06] Bergin, Joseph: *The Object Game. An Exercise for Studying Objects*, November 2006. – <http://pcl.c.pace.edu/~bergin/patterns/objectgame.html> – geprüft: 18. Mai 2008
- [CS06] Claus, Volker ; Schwill, Andreas: *Duden Informatik A–Z. Fachlexikon für Studium und Praxis*. 4., überarb. u. aktualis. Aufl. Mannheim, Leipzig, Wien, Zürich : Bibliographisches Institut, Februar 2006
- [Die07a] Diethelm, Ira: *Strictly models and objects first – Unterrichtskonzept für objektorientierte Modellierung*. In: Schubert, Sigrid (Hrsg.): *Didaktik der Informatik in Theorie und Praxis. 12. GI-Fachtagung »Informatik und Schule - INFOS 2007«*. 19. - 21. September 2007 an der Universität Siegen. Bonn : Gesellschaft für Informatik, 2007, S. 45–56
- [Die07b] Diethelm, Ira: *»Strictly models and objects first« – Unterrichtskonzept und -methodik für objektorientierte Modellierung im Informatikunterricht*. Dissertation (dr. rer. nat.), Universität – Fachbereich Elektrotechnik/Informatik, Kassel, Mai 2007. – <http://kobra.bibliothek.uni-kassel.de/bitstream/urn:nbn:de:hebis:34-2007101119340/1/DissIraDruckfassungA5.1.pdf> – geprüft: 18. Mai 2008
- [Diß01] Dißmann, Stefan: *Erfahrungen mit dem Einsatz von Rollenspielen zur Einführung in die objektorientierte Denkweise*. In: *Proceedings Net.ObjectDays 2001*, Erfurt, 2001, S. 355–360 – <ftp://ls10-ftp.cs.uni-dortmund.de/home/dissmann/public/NetObjectDays-2001-dissmann.pdf> – geprüft: 18. Mai 2008
- [Diß03a] Dißmann, Stefan: *Einführung objektorientierter Gestaltungsprinzipien anhand von Rollenspielen*. In: Siedersleben, Johannes ; Weber-Wulff, Debora (Hrsg.): *Software-Engineering im Unterricht der Hochschulen*, SEUH 8. Berlin : Springer, 2003, S. 30–40. – <ftp://ls10-ftp.cs.uni-dortmund.de/home/dissmann/public/SEUH-2003-dissmann.pdf> – geprüft 18. Mai 2008

Literatur

- [Diß03b] Dißmann, Stefan: *Handlungsorientiertes Erlernen von Programmkonstruktionen anhand von Rollenspielen*. In: Hubwieser, Peter (Hrsg.): *Informatische Fachkonzepte im Unterricht. INFOS 2003. 10. GI-Fachtagung 17.-19. September in Garching bei München.*, Bonn : Gesellschaft für Informatik, 2003, S. 249–260. – <ftp://ls10-ftp.cs.uni-dortmund.de/home/dissmann/public/INFOS-2003-dissmann.pdf> – geprüft: 18. Mai 2008
- [Kuc07] Kuchen, Herbert: *Script zur Vorlesung Informatik I, Kapitel 2*, Münster, November 2007. – <http://www-wi.uni-muenster.de/pi/lehre/ws0708/info1/fohlen/info1k2a.pdf> – geprüft: 18. Mai 2008
- [Mey88] Meyer, Hilbert: *Unterrichtsmethoden*. Band I: Theorieband. 2. Auflage. Frankfurt a.M. : Scriptor Verlag 1988
- [Pot07] Poth, Oliver: *Parameter – Untersuchungen zur Vermeidung verbreiteter Fehlvorstellungen im Informatikunterricht der gymnasialen Oberstufe*. Hausarbeit gem. OVP, Studienseminar für Lehrämter an Schulen - Seminar für das Lehramt für Gymnasien und Gesamtschulen, Hamm, Juni 2007. – <http://www.ham.nw.schule.de/pub/bscw.cgi/d675586/Examensarbeit-Poth.pdf> – geprüft: 18. Mai 2008
- [Sch05] Schriek, Bernard: *Informatik mit Java. Eine Einführung mit BlueJ und der Bibliothek Stifte und Mäuse*. Band I. Werl : Nili-Verlag, September 2005.
- [Sch06] Schriek, Bernard: *Informatik mit Java. Eine Einführung mit BlueJ und der Bibliothek Stifte und Mäuse*. Band II. Werl : Nili-Verlag, September 2006.
- [Sch07] Schriek, Bernard: *Informatik mit Java. Eine Einführung mit BlueJ und der Bibliothek Stifte und Mäuse*. Band III. Werl : Nili-Verlag, Oktober 2007.

Abschließende Erklärung

Abschließende Erklärung

Ich versichere, dass ich die Arbeit eigenständig verfasst, keine anderen Quellen und Hilfsmittel als die angegebenen benutzt und die Stellen der Arbeit, die anderen Werken dem Wortlaut oder Sinn nach entnommen sind, in jedem einzelnen Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe. Das Gleiche gilt auch für beigegebene Zeichnungen, Kartenskizzen und Darstellungen.

Unna, 20. Mai 2008