

Studienseminar für Lehrämter an Schulen Hamm

Parameter - Untersuchungen zur Vermeidung verbreiteter
Fehlvorstellungen im Informatikunterricht der gymnasialen
Oberstufe.

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

Erstgutachter: Dr. Ludger Humbert

vorgelegt von:

Oliver Poth

Schmerhöfeler Weg 2

59199 Bönen

poliver@semsek2.nw.schule.de

Bönen, 28. Mai 2007

Inhaltsverzeichnis

1	Einleitung	1
1.1	Situation	1
1.2	Zielsetzung	2
1.3	Aufbau der Arbeit	2
2	Grundlagen	3
2.1	Technologische Grundlagen	3
2.1.1	Objektorientierte Grundlagen	3
2.2	Parameterübergabe	4
2.3	Lerntheoretische Grundlagen	6
2.3.1	Entwicklung kognitiver Strukturen nach Piaget	7
2.3.2	Repräsentationsebenen	9
2.4	Didaktische Grundlagen	10
2.4.1	Unterrichtsformen	10
3	Problembeschreibung	12
4	Methoden und Konzepte	16
4.1	Kriterien zur Auswahl von Methoden und Konzepten	16
4.1.1	Enaktive Repräsentation	17
4.1.2	Wechsel der Repräsentationsform	17
4.1.3	Interaktion	17
4.2	Vorstellung der Konzepte	17
4.2.1	Einführung in die objektorientierte Modellierung	18
4.2.2	Schubladenspiel (Modifiziertes Objektspiel)	19
4.3	Referenzen auf Objekte	22
4.4	Vertiefung der Lerninhalte	23
5	Umsetzung	25
5.1	Umsetzung des Schubladenspiels	25
6	Reflexion und Ausblick	27
6.1	Reflexion der Umsetzung des Schubladenspiels	27
	Abbildungsverzeichnis	30
	Literatur	31
A	Verlaufsplan und Arbeitsblätter zum Schubladenspiel	32
B	Klasse Punkt	38
C	Aufgabe: Verweise auf Objekte	39

1 Einleitung

1.1 Situation

Der Anfangsunterricht im Fach Informatik in der gymnasialen Oberstufe ist für viele Schülerinnen und Schüler der erste Kontakt mit einer höheren Programmiersprache. Die Schülerinnen und Schüler lernen viele neue, zum Teil recht abstrakte Begriffe und Darstellungsformen kennen. Zunächst lernen sie als zentrale Begriffe Objekte mit ihren Attributen und Methoden kennen. Sie üben diese Begriffe, indem sie einfache, ihnen vertraute, Objekte aus der Realität modellieren. Darauf aufbauend werden Beziehungen zwischen den Objekten definiert und erläutert, wie diese untereinander Informationen austauschen und Aktionen auslösen können. Dies geschieht auf Basis des Nachrichtenkonzeptes, das auf dem Aufruf von Methoden und der Übergabe von Parametern beruht. Da der Kommunikation von Objekten eine wesentliche Bedeutung zukommt, ist der Aufbau eines tiefsitzenden Verständnisses über ihren Ablauf bei den Schülerinnen und Schülern notwendig. Um die zeitlichen Abläufe und Nachrichten, welche zwischen Objekten ausgetauscht werden, darstellen zu können, werden Objektspiele und Sequenzdiagramme eingeführt. Durch Einführung des Begriffs der Klasse lernen die Schülerinnen und Schüler die konkrete Umsetzung in eine Programmiersprache. Sie lernen nun die Syntax einer Programmiersprache kennen und müssen die vorher nur modellierten Klassen in die Programmiersprache übersetzen. Es hat sich gezeigt, dass die praktische Umsetzung des theoretischen Wissens über Methodenaufrufe und Parameterübergabe häufig Probleme bereitet und Schülerinnen und Schüler den Zugang zu einer höheren Programmiersprache erschwert. Diese Arbeit beschäftigt sich mit den häufigsten Problemen bei der Einführung einer neuen Programmiersprache in diesem Bereich. Obwohl die meisten dieser Probleme unabhängig von Paradigma und Sprache auftreten, wird in dieser Analyse im Speziellen die objektorientierte Programmierung betrachtet.

1.2 Zielsetzung

Ziel dieser Hausarbeit ist eine Analyse möglicher auftretender Fehlvorstellungen von Schülerinnen und Schülern während des Anfangsunterrichts in der gymnasialen Oberstufe. Es werden mögliche Ursachen für typische Fehler bei der Parameterübergabe untersucht. Der Fokus liegt dabei auf der Vorstellung von Methoden und Konzepten zur Vermeidung dieser Probleme, eine Betrachtung der neurobiologischen Zusammenhänge ist nicht Bestandteil dieser Arbeit.

1.3 Aufbau der Arbeit

Zunächst werden die Problemstellung und Zielsetzung der Arbeit beschrieben. Das zweite Kapitel beschreibt notwendige technologische und didaktische Grundlagen. Eine genauere Analyse der auftretenden Probleme und ihrer möglichen Ursachen erfolgt im dritten Kapitel. In diesem Punkt bezieht sich die Arbeit auf die Lehrerfunktion "Diagnostizieren". Im vierten Kapitel werden Methoden und Konzepte zur Vermeidung von Fehlvorstellungen der Schülerinnen und Schüler vorgestellt und untersucht. Die Umsetzung einiger dieser Methoden wird im fünften Kapitel beschrieben. In diesen Kapiteln bezieht sich die Arbeit auf die Lehrerfunktion "Unterrichten". Die didaktische und methodische Umsetzung wird geplant, begründet und anschließend wird sie selbst beschrieben. Im sechsten Kapitel erfolgt eine Reflexion der im BDU des Autors durchgeführten Methoden.

2 Grundlagen

In diesem Kapitel werden zunächst einige notwendige technologische, lerntheoretische und didaktische Grundlagen beschrieben.

2.1 Technologische Grundlagen

2.1.1 Objektorientierte Grundlagen

In der sogenannten *Objektorientierung* oder *objektorientierten Softwareentwicklung* beschreibt ein *Objekt* einen Gegenstand oder einen abstrakten Begriff. Ein “Stuhl” ist ein einfaches Beispiel für ein solches Objekt. Ein Stuhl hat bestimmte Eigenschaften, die ihm zugeschrieben werden, wie beispielsweise Höhe, Tiefe, Breite oder Farbe. Diese Eigenschaften werden in der Objektorientierung als *Attribute* bezeichnet. Um ein Attribut zu modifizieren oder Anfragen an das Objekt stellen zu können, existieren *Methoden*. Sämtliche Objekte, Attribute und Methoden werden durch *Bezeichner* eindeutig identifiziert. Um z.B. die Breite eines Stuhls zu ändern, könnte eine Methode `setzeBreite(neueBreite)` aufgerufen werden. Damit das Stuhl-Objekt weiß, wie breit es ist, muss ihm dies mitgeteilt werden. Solch eine Mitteilung nennt man in der Objektorientierung *Nachricht*. Eine Nachricht besteht aus drei Teilen: Dem *Adressaten*, der Methode und den Parametern. Folgende Notation ist gebräuchlich: `adressat.methode(Parameter)`

`adressat` ist hierbei der Bezeichner des Objektes an das die Nachricht geschickt wird, `methode` ist der Bezeichner der Methode, die der Adressat ausführen soll. Die Parameter sind Daten, die zur Durchführung der Methode benötigt werden. Es können mehrere oder keine Parameter benötigt werden. Wenn nun dem Stuhl `erikasStuhl` die Nachricht geschickt werden soll, seine Breite auf 150 (cm) zu ändern, wäre der korrekte Aufruf `erikasStuhl.setzeBreite(150)`. Hierbei wird an die Methode `setzeBreite(neueBreite)` der Wert 150 übergeben. Dieser Vorgang wird auch als *Parameterübergabe*¹ bezeichnet.

¹Der Vorgang der Parameterübergabe wird im folgenden Kapitel genauer beleuchtet.

Um die Breite eines Stuhls zu erfahren, muss eine Anfrage an das Objekt gestellt werden. Eine Anfrage ist ebenfalls eine Nachricht, allerdings wird im Fall einer Anfrage eine Antwort vom Objekt erwartet. Die Anfrage `erikasStuhl.breite()` würde also den Wert 150 als Antwort erhalten.

Betrachtet man mehrere Objekte des gleichen Typs, so fällt auf, dass sie alle gemeinsame Eigenschaften besitzen. Jeder Stuhl besitzt eine Höhe, Tiefe, Breite oder Farbe. Obwohl die jeweiligen Ausprägungen der Attribute unterschiedlich sind (nicht alle Stühle weisen die gleiche Breite auf) können Objekte hierdurch in Gruppen eingeteilt werden. Diese Gruppen werden in der Objektorientierung als *Klassen* bezeichnet. Eine Klasse ist eine Art Bauplan, in dem die Attribute und Methoden beschrieben werden (vgl. [Wei06], S 34).

2.2 Parameterübergabe

Aktueller und formaler Parameter

```
1 public int quadrat(int zahl)
2 {
3     \\ hier steht die Implementierung
4 }
```

Abbildung 1: Parameterübergabe (Java)

Bei der Parameterübergabe werden die Bezeichner, die in der Methodendeklaration verwendet werden, als *formale Parameter* bezeichnet. In Abbildung 1 ist der formale Parameter die Ganzzahl `zahl`. Beim Aufruf einer Methode wird für jeden Parameter ein Wert angegeben. Würde die obige Methode mittels `adressat.quadrat(22)` aufgerufen, so wäre die Zahl 22 der *aktuelle Parameter* oder das *Argument* der Methode (vgl. [Ull06], Kap. 2.7.6). Ein aktueller Parameter kann auch selber aus einer Variable bestehen. Wurde eine ganze Zahl durch `int meineZahl=6` definiert, kann die Methode mittels `adressat.quadrat(meineZahl)` aufgerufen werden. `meineZahl` ist in diesem Fall der aktuelle Parameter der Funktion `quadrat()`.

Call by value

Bei der *Werteübergabe* übernimmt der Parameter einer Methode den Wert des beim Aufruf übergebenen Argumentes. Die Rückgabe von Werten über die Parameter ist nicht möglich.

```
1 public int quadrat(int zahl)
2 {
3     zahl = zahl*zahl;
4     return zahl;
5 }
```

Abbildung 2: Parameterübergabe (mittels Call by value (Java))

Beim Aufruf `zahl2 = quadrat(zahl1)` (s. Abbildung 2) wird `zahl2` das Quadrat von `zahl1` zugewiesen, der Wert von `zahl1` ändert sich nicht.

Call by reference

Bei der *Referenzübergabe* wird nicht der Wert, sondern der Speicherplatz der Argumente übergeben. Der Parameter wird also lediglich zu einem anderen Bezeichner für das Argument. Sämtliche Änderungen des Parameters werden daher auch am Argument selbst vorgenommen.

```
1 PROCEDURE Quadrat(VAR Zahl : Integer);
2 BEGIN
3     Zahl := Zahl*Zahl;
4 END;
```

Abbildung 3: Call by reference (PASCAL)

Beim Aufruf `Quadrat(zahl1)`; (s. Abbildung 3) ändert sich nun tatsächlich der Wert der Variable `zahl1`. Hätte `zahl1` vor dem Aufruf der Prozedur `Quadrat()` den Wert 5, so wür-

de die Ausgabe von `zahl1` nach dem Aufruf von `Quadratzahl1` aus der Zahl 25 bestehen.

Parameterübergabe von Referenzen mittels `call by value`

In Sprachen, in denen bei der Parameterübergabe Referenzen auf Objekte übergeben werden (z.B. Java), kann es zu ähnlichen Effekten wie beim `Call by reference` kommen. Beim Aufruf `meinPunkt2 = obj.rechne(meinPunkt1)` (s. Abbildung 4) wird nicht nur die Referenz `meinPunkt2` verändert. Das Objekt, auf das die Referenz `meinPunkt1` verweist, wird verändert, da der Parameter `punkt` auf dasselbe Objekt verweist.

```
1 public Punkt rechne(Punkt punkt)
2 {
3     punkt.verschiebePunkt(5,5);
4     return punkt;
5 }
```

Abbildung 4: `Call by value` mit Referenzen auf Objekte (Java)

Weitere Mechanismen

Weitere Mechanismen der Parameterübergabe sind “`Call by name`”, “`Call by result`” und “`Call by value and result`”. (vgl. [Cla06] und [Kas05])

2.3 Lerntheoretische Grundlagen

Um pädagogisches Handeln im Unterricht zu planen und zu analysieren, wird eine Theorie des Lernens benötigt. Lernen ist ein umfassender Begriff, er beinhaltet unter anderem das Lernen von Fakten, Fertigkeiten, Sozialverhalten. Oftmals geschieht Lernen ohne Intention und auch ohne, dass es den Lernenden bewusst ist. Da der Umgang mit Parametern in der objektorientierten Modellierung für den Großteil der Schülerinnen und Schüler neu ist, müssen sie erst eine

Vorstellung der Abläufe bei der Parameterübergabe und weiteren Verwendung von Parametern erhalten. Das Lernen im Umgang mit Parametern ist daher ein bewusstes Lernen (*explizites Lernen*). Aus diesem Grund beschränkt sich die Arbeit im Wesentlichen auf diesen Aspekt. Aus Sicht der Verhaltenspsychologie sind Vorstellungen allerdings nicht wissenschaftlich untersuchbar. Obwohl Vorstellungen und Erkenntnisprozesse beim Lernen nicht beobachtbar sind, hat die kognitive Psychologie Methoden entwickelt, um innere Prozesse, also Prozesse welche im Kopf des Lernenden ablaufen, während des expliziten Lernens zu beschreiben (vgl. [Bov04], S 195).

2.3.1 Entwicklung kognitiver Strukturen nach Piaget

Explizites Lernen basiert auf *Kognitiven Strukturen*, das sind Erkenntnisse, die im Langzeitgedächtnis gespeichert sind und das Handeln und das Sammeln weiterer Erkenntnisse steuern. Der Schweizer Psychologe Jean Piaget (1896-1980) hat beschrieben, wie die Entwicklung kognitiver Strukturen vonstatten geht. Bei der Entwicklung von Kognitiven Strukturen werden zwei gegenläufige Prozesse unterschieden: *Assimilation* und *Akkommodation*.

Assimilation bedeutet, dass vorhandene Strukturen gefestigt, eingeübt und auf neue Situationen und Sachverhalte angewandt werden. Bei der Akkommodation werden vorhandene Strukturen korrigiert, differenziert und erweitert, um sich so neuen Situationen oder Sachverhalten anzupassen (vgl. [Bov04], S 206ff).

In der Grundschule lernen Schülerinnen und Schüler im Fach Mathematik Variablen in Form von Platzhaltern kennen. In der Mittelstufe werden diese durch Buchstaben ersetzt. Die Schülerinnen und Schüler müssen ihr Wissen über Platzhalter nun auf die neue Schreibweise übertragen, dies wäre Assimilation. Später lernen die Schülerinnen und Schüler, dass mit Variablen ebenfalls wie mit Zahlen gerechnet werden kann. Im Fach Informatik lernen sie nun Variablen im Kontext einer Programmiersprache kennen. Diese Variablen sind aber nun nicht mehr auf Zahlen beschränkt. Die Vorstellung über die Funktion und Bedeutung von Variablen muss differenziert und erweitert werden, dieser Prozess ist eine Akkommodation.

Grundsätzlich finden die beiden Prozesse Assimilation und Akkommodation immer gleichzeitig

statt, allerdings in unterschiedlich starker Ausprägung. Das Rechnen mit Variablen wie mit Zahlen festigt die vorhandenen Erkenntnisse über Variablen (Assimilation), erweitert sie aber auch (Akkommodation).

Voraussetzung für Assimilation und Akkommodation sind die notwendige Reifung und Erfahrung. Reifung ist die durch Veranlagung festgelegte Entwicklung. Es müssen bestimmte Reifestände erreicht werden, um Lernfortschritte erzielen zu können. Nach Piaget ist es unsinnig, Kindern etwas anzutrainieren, wenn sie die notwendige Reife dafür noch nicht besitzen. Dabei sind sowohl körperliche wie kognitive Reife gemeint. Mit Erfahrung ist die aktive Erfahrung im Umgang mit einer Sache gemeint. Diese kann nicht durch Anschauung oder verbale Erklärungen ersetzt werden.

Um die Reifestufen von Kindern einzuordnen, unterscheidet Piaget vier Stufen der kognitiven Entwicklung:

- Sensorische Stufe (Geburt bis 2. Lebensjahr)
- Präoperationale Stufe (2.-7. Lebensjahr)
- Konkret-operationale Stufe (7.-11. Lebensjahr)
- Formal-logische Stufe (ab 11./12. Lebensjahr)

In der sensorischen Stufe verfügt das Kind nur über einfache, angeborene Reflexe wie Saugen oder Greifen.

Auf der präoperationalen Stufe kann sich das Denken des Kindes vom eigenen Tun lösen, ist aber noch nicht abstrakt oder logisch. Es ist an die aktuelle Wahrnehmung des Kindes gebunden.

In der konkret-operationalen² Stufe hingegen kann sich das Kind in seinem Denken von der aktuellen Wahrnehmung lösen. Es kann Sachverhalte logisch begreifen, ist allerdings an konkrete, anschaulich vorstellbare Dinge gebunden.

Auf der formal-logischen Stufe ist das Kind in der Lage zu abstrahieren und einem formal-logischen Gedankengang zu folgen.

²Eine Operation ist nach Aebli eine abstrakte Handlung.

Piagets Erklärungen legen nahe, dass Lernen auf eigenen praktischen Erfahrungen beruhen muss. Unterricht muss dies berücksichtigen. Es kann zwischen Lernangeboten, die primär dem Üben und Anwenden (Assimilation) dienen, und denen, die neue Denkstrukturen erfordern (Akkommodation), unterschieden werden.

2.3.2 Repräsentationsebenen

Aus Piagets Ergebnissen ergeben sich für den Unterricht verschiedene Stufen der Darstellung eines Unterrichtsgegenstandes. Nach Bruner wird zwischen drei Repräsentationsebenen unterschieden (vgl. [Har06] S 116):

Enaktive Repräsentation beschreibt das Erfassen von Sachverhalten durch eigenes Tun. Kinder lernen durch eigenes Handeln, z.B. zu laufen. Sie beobachten zwar andere, benötigen aber keine Gebrauchsanweisung oder abstrakte Theorien, um laufen zu lernen.

Die *ikonische Repräsentation* meint das Beschreiben von Sachverhalten durch bildhafte Veranschaulichungen. Konkrete Gegenstände oder Vorgänge kann sich ein Mensch auch bildhaft vorstellen. Ein Stadtplan reicht aus, um sich in einer fremden Stadt zurechtzufinden, sofern man mit dieser Darstellungsform vertraut ist.

Symbolische Repräsentation letztlich ist die Darstellung eines Sachverhaltes durch Symbole (Text, Zeichen). Unter dem Begriff Haus kann man sich bereits etwas vorstellen ohne das Haus zu sehen. Allerdings können hierbei auch Missverständnisse auftreten, da Menschen aus verschiedenen Kulturen möglicherweise eine sehr unterschiedliche Vorstellung von einem Haus haben. Symbolische Darstellungen bringen außerdem nur dann einen Vorteil, wenn bereits genaue Vorstellungen zu einem Symbol vorliegen. Jemand, der noch nie ein Auto gesehen hat, kann sich unter dem Begriff nichts vorstellen, genausowenig wie jemand, der zwar schon einmal ein Auto gesehen hat aber nie erlebt hat, darin zu fahren.

Diese Hierarchisierung wird auch als EIS-Prinzip (**E**naktiv-**I**konisch-**S**ymbolisch) bezeichnet (vgl. [Hum05] S 34).

Im Unterricht ist der Wechsel zwischen diesen Repräsentationsformen wichtig. Denkooperationen sollten, wenn möglich, immer auf mehreren Ebenen stattfinden.

2.4 Didaktische Grundlagen

Didaktik bezeichnet die Wissenschaft vom Unterricht. Die Allgemeine Didaktik erforscht Unterricht unabhängig vom konkreten Fach und entwickelt Analyse- und Planungs-Konzepte für den Unterricht(vgl. [Bov04] S 31). Im folgenden werden einige allgemeine Didaktische Grundlagen erläutert.

2.4.1 Unterrichtsformen

Unterricht besteht im Wesentlichen aus Interaktionen zwischen den Lernenden und dem Lehrenden sowie zwischen den Lernenden untereinander. Hierbei wird zwischen den *Sozialformen* und den *Aktionsformen* unterschieden.

Unter Sozialform versteht man die Form der Interaktion zwischen Lehrendem und Lernendem. Man unterscheidet vier Sozialformen:

- Frontalunterricht (FA)
- Gruppenarbeit (GA)
- Partnerarbeit (PA)
- Einzelarbeit (EA)

Beim Frontalunterricht erläutert der Lehrende der Lerngruppe einen Sachverhalt oder führt ein Unterrichtsgespräch mit den Schülerinnen und Schülern. Diese Sozialform ist eher lehrerzentriert und für die selbständige Erarbeitung eines neuen Gegenstandes nicht geeignet. In der Gruppenarbeit arbeiten mehrere Schülerinnen und Schüler zusammen an einer Aufgabe. Die Gruppengröße beträgt in der Regel 4-5 Personen. Partnerarbeit ähnelt der Gruppenarbeit, allerdings bestehen die Gruppen nur aus zwei Personen. In Einzelarbeitsphasen arbeiten die Schülerinnen und Schüler alleine und selbständig an einer Aufgabe.

Partner- und Gruppenarbeit sind hierbei als Sozialform zu bevorzugen. Wie bereits in Kapitel 2.3.2 beschrieben, ist Interaktion für das Lernen besonders wichtig. Darüberhinaus fördert

Gruppenarbeit die Kommunikationsfähigkeit sowie die soziale Kompetenz der Schülerinnen und Schüler . Hierbei kann zwischen Arbeitsgleicher und Arbeitsteiliger Gruppenarbeit unterschieden werden. Bei arbeitsgleicher Gruppenarbeit bekommen alle Gruppen die gleiche Aufgabe gestellt, dies hat den Vorteil, dass die Gruppen ihre Ergebnisse vergleichen können und sich während der Arbeitsphase Hilfestellungen bei anderen Gruppen holen können. In der arbeitsteiligen Gruppenarbeit erhalten die Gruppen verschiedene Aufgaben. Die Aufgaben sollten aber alle bedeutsam für das Unterrichtsthema sein, so dass jede Gruppe einen einzigartigen Beitrag zum Unterricht beiträgt. So wird keine Gruppe bei der Präsentation ihrer Ergebnisse übergangen. Arbeitsteilige Gruppenarbeit bietet sich dementsprechend besonders an, wenn sich eine Aufgabe oder Problem in Teilaspekte aufteilen lässt. Auf diese Weise arbeiten alle Gruppen an einer gemeinsamen Aufgabe und ihre Ergebnisse werden zur Lösung der Aufgabe zusammengefasst. Darüber hinaus lässt sich mit arbeitsteiliger Gruppenarbeit eine *innere Differenzierung* der Lerngruppe erreichen. Dies bedeutet, dass für Schülerinnen und Schüler mit verschiedenen Fähigkeiten und Kompetenzen individuelle Aufgaben bereitgestellt werden können.

Die *Aktionsform* ist die Form, wie der Unterrichtsgegenstand vermittelt wird. Die Aktionsform wird auch als Arbeitsform oder Methode bezeichnet. Es wird zwischen den drei Arten

- der darbietenden,
- der erarbeitenden und
- der entdecken-lassenden

Arbeitsform unterschieden. Zur darbietenden Aktionsform gehört das Präsentieren und Vormachen, zur erarbeitenden die unterschiedlichen Gesprächsformen und zur entdecken-lassenden die Problem lösenden Verfahren (vgl. [Bov04] S 70ff).

3 Problembeschreibung

Im Einführungsunterricht des Faches Informatik tauchen bei der Verwendung von Parametern verschiedene Probleme und Fehler bei den Schülern auf. Dabei wird zwischen zwei Arten von Problemen unterschieden:

1. Probleme, die sich aus dem Verständnis der (objektorientierten) Modellierung ergeben und somit unabhängig von der konkreten Programmiersprache auftreten.
2. Fehler in der Syntax, welche daher nicht in allen Programmiersprachen auftreten.

In diesem Kapitel werden einige häufige Probleme vorgestellt und mögliche Ursachen erörtert, somit bezieht sich dieses Kapitel auf die Lehrerfunktion “Diagnostizieren”.

Die hier dargestellten Beispiele stammen aus dem BdU des Autors im Fach Informatik. In dem Grundkurs in der Jahrgangsstufe 11 wird die Klassenbibliothek “Stifte und Mäuse” (SuM) mit Java eingesetzt.

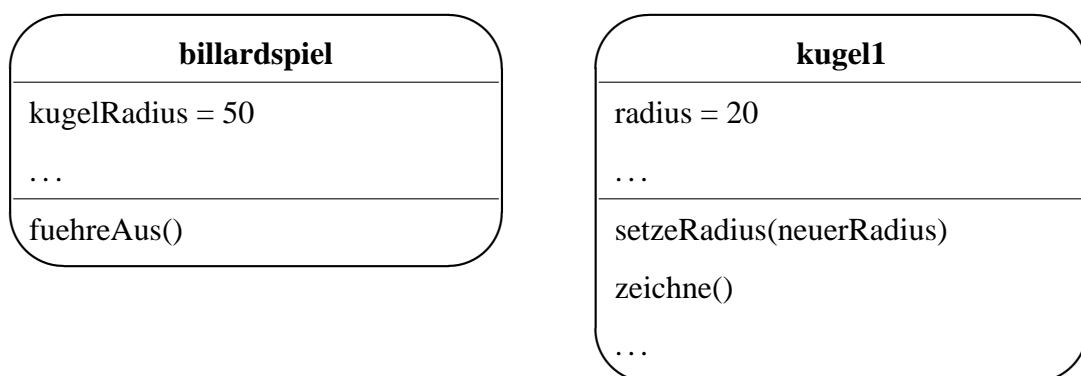


Abbildung 5: Objektkarten

Beispiel 1:

Das Programm “Billard” stellt eine einfache Simulation von Kugeln, dargestellt durch Kreise, auf dem Bildschirm dar. Die Klasse `Billardspiel` enthält die Methode `fuehreAus()`. In dieser läuft in einer Schleife das Programm ab. Die Klasse “Kugel” stellt einen Kreis im Programm-

fenster dar, welcher sich bewegen kann und dabei vom Fensterrand abprallt. Eine Kollision von zwei “Kugeln” ist nicht implementiert.

Das Objekt “billardspiel” ist ein Exemplar der Klasse “Billard” und hat mit “kugel1”(s. Abbildung 5) ein Exemplar der Klasse “Kugel”. “billardspiel” hat mit dem Attribut “kugelRadius” ein Datum, welches “kugel1” benötigt. “billardspiel” schickt über `kugel1.setzeRadius(kugelRadius)` den Auftrag an “kugel1”, dem Attribut “radius” den Wert des Attributes “kugelRadius” also 50 zuzuweisen.

Ein Schüler fragt daraufhin, wieso sich das Objekt “kugel1” nicht einfach direkt den Wert von “kugelRadius” holt.

Das Problem ist scheinbar, dass die grundsätzliche Trennung der einzelnen Objekte noch nicht klar ist. Die Schülerinnen und Schüler sehen das gesamte, im Anfangsunterricht meist auch überschaubare, Problem und wissen, welchen Wert die Variable annehmen soll. Variablen sind bisher hauptsächlich aus dem Mathematikunterricht bekannt, dort werden die Daten aber üblicherweise direkt in Funktionen eingesetzt. Ein “Umweg” über weitere Variablen wird hier nicht gemacht. Der genaue Ablauf einer Nachricht zwischen zwei Objekten muss hier deutlich machen, wie und wozu Parameter benötigt werden.

Beispiel 2:

Wenn die Schülerinnen und Schüler nun die Methode “setzeRadius()” implementieren sollen, wissen sie nicht, wie sie den Parameter “neuerRadius” richtig zu verwenden haben. Einige vorher besprochene Beispiele für Methodendeklarationen reichen nicht aus, um die Wertübergabe vom Attribut “kugelRadius” über den Parameter “neuerRadius” zum Attribut “radius” zu verstehen. Dieses Problem ist unabhängig von der verwendeten Programmiersprache.

Hierbei kann möglicherweise an die Erfahrungen der Schülerinnen und Schüler mit Funktionen angeknüpft werden. Man berechnet den Funktionswert, indem ein Wert in den Term an die Stelle gesetzt wird, an der die Variable steht. Genau so können auch Parameter aufgefasst werden. Allerdings können Parameter in der objektorientierten Programmierung wiederum andere Objekte darstellen und sind somit nicht nur Zahlen. Darüber hinaus kennen Schüler in der Regel nur Funktionen mit einer Variablen. Beim Programmieren kommt man aber schnell zu

Methoden, die mehrere Parameter besitzen.

Beispiel 3:

“kugel1” hat, um sich zu zeichnen, einen Stift “meinStift”. Mithilfe des Stiftes soll nun die Methode `zeichne()` (s. Abbildung 6) der Klasse `Kugel` geschrieben werden. Der Stift soll einen Kreis mit dem Wert des Attributes “radius” als Radius zeichnen. Viele Schülerinnen und Schüler schreiben nun `meinStift.zeichne(50)`. Es ist also nicht klar, dass der Wert 50 im Attribut “kugelRadius” gespeichert ist. Dieses Problem ist unabhängig von der Programmiersprache.

```
1 public void zeichne ()
2 {
3     meinStift.zeichneKreis (radius );
4 }
```

Abbildung 6: Methode `zeichne()`

Selbst wenn der Unterschied zwischen Klasse und Objekt klar hervorgehoben wurde, sehen die Schülerinnen und Schüler nicht sofort, an welchen Stellen die Attribute der Klasse benutzt werden müssen. Hier muss klar herausgestellt werden, dass aus der Klasse `Kreis` nur dann verschiedene Objekte mit unterschiedlichen Eigenschaften (Attributen) erzeugt werden können, wenn die Attributwerte verwendet werden.

Beispiel 4:

Wenn anstatt einfacher Datentypen Objekte als Parameter übergeben werden, ändert sich zwar die prinzipielle Vorgehensweise nicht, bei der Parameterübergabe von Referenzen kann es aber bei der Wertübergabe zu Effekten kommen, die “Call by reference” ähneln (vgl. Kapitel 2.2). Dieses Verhalten wird von den Schülerinnen und Schüler selten vorausgesehen, intuitiv stellen sie sich die Parameterübergabe als “call by value” vor. Die Schülerinnen und Schüler müssen daher die Funktionsweise von Verweisen (Referenzen) auf Objekte genau verstehen.

Beispiel 5:

Der nun vorgestellte Fehler bezieht sich nur auf die Syntax, im Speziellen auf die Syntax von Java. Der unterschiedliche Aufbau zwischen Methodendeklaration und dem Aufruf einer Methode innerhalb einer Nachricht führt oft zu Verwechslungen. Bei der Deklaration wird der Typ des Parameters weggelassen, beim Aufruf einer Methode taucht er allerdings auf. Weiterhin wird beim Aufruf der Methode der Parameterbezeichner aus der Methodendeklaration verwendet anstatt den konkreten Wert oder eine entsprechende Variable zu übergeben. Also:

```
public void setzeRadius(neuerRadius) bzw.  
meinKreis.setzeRadius(int neuerRadius)
```

4 Methoden und Konzepte

In diesem Kapitel werden Methoden und Konzepte vorgestellt, um die in Kapitel 3 beschriebenen Fehler und Fehlvorstellungen zu vermeiden. Da hier die Vermeidung von Fehlvorstellungen im Anfangsunterricht behandelt wird, befasst sich dieser Teil der Arbeit mit der Lehrerfunktionen "Unterrichten". Zunächst werden Kriterien angegeben, an denen sich die vorgestellten Methoden und Konzepte messen lassen. Anschließend folgen die Methoden und Konzepte selbst. Dabei wird unterstellt, dass zumindest einige Schülerinnen und Schüler zu Anfang der Oberstufe keine Vorkenntnisse im Fach Informatik besitzen. Daher müssen die Schülerinnen und Schüler zu Beginn des Einführungsunterrichts im Fach Informatik in der Oberstufe die Methoden der Modellierung eines Objektes aus der Realität oder einem beschreibenden Text kennenlernen. Da bereits hierbei Fehlvorstellungen vermieden werden können, werden auch kurz Konzepte zur Einführung in die objektorientierte Modellierung beschrieben.

4.1 Kriterien zur Auswahl von Methoden und Konzepten

Da beim Lernen sowohl Assimilation und Akkomodation³ von Bedeutung sind, muss jedes Konzept beide Prozesse im Blick haben. Besonders auf die Akkomodation muss geachtet werden, da Menschen dazu neigen, an vertrauten Schemata festzuhalten. "Es wird assimiliert, wo akkomodiert werden sollte!" ([Bov04] S 211) Akkomodation kann nur stattfinden, wenn die Lernenden ihr Wissen mithilfe von Fragen, Sammeln von Informationen, Überprüfungen, argumentativen Auseinandersetzungen und sinnfälligen Erfahrungen selbst konstruieren (vgl. [Bov04] S 112). Es wird deutlich, dass Interaktion mit anderen eine wichtige Voraussetzung für Akkomodation darstellt (vgl. [Hum05] S 35).

Zunächst werden Kriterien zur Bewertung spezifischer Methoden und Konzepte zur Einführung und Vertiefung des Umgangs mit Parametern aufgestellt.

³Zu den Begriffen Assimilation und Akkomodation s. Seite 7

4.1.1 Enaktive Repräsentation

Bezüglich des EIS-Prinzips⁴ finden in der Informatik Veranschaulichungen durch Diagramme und Abbildungen (also die ikonische Stufe) und die symbolische Stufe eine häufige Anwendung. Da die enaktive Stufe aber als erste Stufe des EIS-Prinzips ebenso bedeutsam ist, ist es um so wichtiger, die abstrakten Begriffe der Informatik auch enaktiv zugänglich zu machen. Die Schülerinnen und Schüler sollten den abstrakten Gegenstand “Parameter” also sehen, anfassen und manipulieren können.

4.1.2 Wechsel der Repräsentationsform

In Kapitel 2.3.2 wird auf die Bedeutung des Wechsels der Repräsentationsformen hingewiesen. Zusätzlich zur enaktiven Repräsentation müssen die Schülerinnen und Schüler auch in Lage sein ikonische Repräsentationen, also in diesem Fall z.B. Objektdiagramme und Sequenzdiagramme, verstehen und anfertigen zu können. Es sollte also eine Wechselwirkung zwischen der enaktiven und ikonischen Repräsentation bestehen. Optimal wäre das Einbeziehen der symbolischen Ebene z.B., indem die Schülerinnen und Schüler ihr Handeln mit Fachbegriffen beschreiben.

4.1.3 Interaktion

Da zur argumentativen Auseinandersetzung mit einem Gegenstand Interaktion zwischen den Lernenden notwendig ist und auch sowohl die enaktive als auch die symbolische Stufe durch das miteinander Handeln und miteinander Kommunizieren angesprochen wird, sollte eine Methode die Interaktion zwischen den Schülerinnen und Schülern fördern und fordern.

4.2 Vorstellung der Konzepte

⁴Eis-Prinzip s. Kapitel 2.3.2

4.2.1 Einführung in die objektorientierte Modellierung

Üblicherweise geschieht die Einführung in die objektorientierte Modellierung mit dem Beschreiben einiger alltäglicher Objekte (z.B. einem Auto oder Fahrrad). Die Modellierung beschränkt sich naturgemäß auf wichtige Aspekte der zu modellierenden Objekte. Die Schülerinnen und Schüler lernen in diesem Zusammenhang die Darstellung von Objekten mittels Objektkarten kennen.

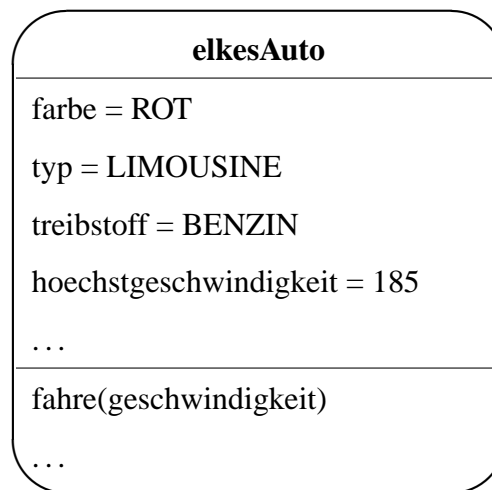


Abbildung 7: Objektkarte elkesAuto

Um dem Problem aus Beispiel 3 (s. S 14) entgegen zu wirken, werden Beispiele benötigt, in denen ein Objekt Attributwerte mittels einer Nachricht an ein anderes Objekt sendet. Um das Nachrichtenkonzept und somit auch Parameter einzuführen, benötigt man demnach komplexere Beispiele, in denen mehrere Objekte in einer Beziehung zueinander stehen.

Es wird die Situation in einem Wartezimmer einer Arztpraxis modelliert. Die Patienten melden sich in der Praxis an und werden in eine Warteliste eingetragen (s. Abbildung 8). Es wird angenommen, dass sich die Patienten über ihre Nummer identifizieren. Die Anmeldung des Patienten Meier erfolgt durch die Nachricht `anmeldung.meldePatientAn(patientennummer)`. Hierbei übergibt also das Objekt `patientMeier` ein Attribut als aktuellen Parameter an das Objekt `anmeldung`.

Zu diesem Zeitpunkt haben die Schülerinnen und Schüler bereits einige bildhafte Darstellun-

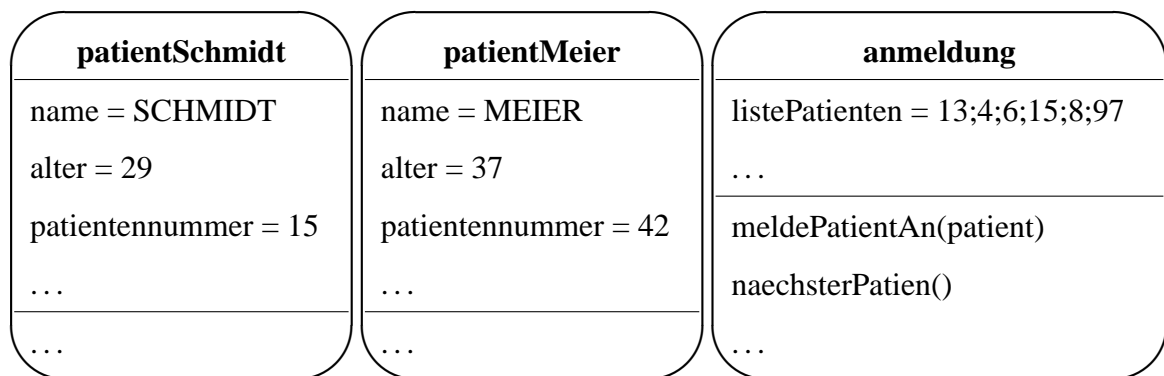


Abbildung 8: Objektkarten Wartezimmer

gen abstrakter Begriffe kennengelernt. Es ist daher notwendig, den Schülerinnen und Schülern die bisherigen Gegenstände enaktiv, also durch aktives Handeln, zugänglich zu machen. Nur so können sie Neues durch Assimilation lernen und ihre kognitiven Strukturen durch neue Erfahrungen erweitern (vgl. [Hum05], S 35).

Um zeitliche Abläufe und das Zusammenspiel zwischen Objekten (z.B. Patienten und Anmeldung) für die Lernenden erlebbar zu machen, bietet sich ein Objektspiel an (vgl. [Lif07] und [Hum05], S 78). Hierbei spielen die Schülerinnen und Schüler einzelne Objekte. Die einzelnen Operationen, die während des Spiels von den "Objekten" durchgeführt werden, können in einem Storyboard festgehalten werden. Aus diesem lässt sich anschließend ein Sequenzdiagramm erstellen. Damit lernen die Schülerinnen und Schüler eine fachliche Darstellung konkreter zeitlicher Abläufe zwischen Objekten. Sie können das Dargestellte gleichzeitig mit dem soeben gespielten Rollenspiel verbinden. Dies führt zu einer Akkomodation neuartiger Strukturen.

Im Anschluß daran kann die Implementierung erster Programme erfolgen. Die nun folgenden Methoden und Konzepte lassen sich im Prinzip in jede Reihe einbinden.

4.2.2 Schubladenspiel (Modifiziertes Objektspiel)

Das folgende Rollenspiel ist eine Modifikation des bereits erwähnten Objektspiels. Das Spiel soll den Ablauf eines Programms darstellen, dabei wird nun besonders auf die Übergabe von Werten durch Parameter geachtet. Das Spiel kann noch vor dem ersten Kontakt mit einer kon-

kreten Programmiersprache durchgeführt werden, sollte aber wiederholt werden, sobald die Schülerinnen und Schüler die Syntax der im Unterricht verwendeten Programmiersprache kennengelernt haben. Je nach Stand kann dann ein Beispielprogramm vorgegeben werden oder die Schülerinnen und Schüler entwickeln zunächst selbst ein Programm⁵. In jedem Fall sollte das Programm einfach gehalten sein, jedoch die wichtigsten Fälle im Umgang mit Parametern aufweisen. Das Schubladenspiel behandelt also die Probleme für Beispiel 1-3 (s. Kapitel 3).

1. In einer Methode werden die Parameter direkt in der Methode benötigt.
2. Im Konstruktor oder einer Methode wird einem Attribut der Wert durch einen Parameter zugewiesen.
3. In einer Methode wird ein Parameter für eine Nachricht an ein drittes Objekt benötigt.
4. Der formale Parameter einer Methode einer Klasse wird innerhalb dieser Methode an eine weitere Methode derselben Klasse übergeben. In einer Methode wird der Parameter für eine Nachricht an das Objekt, das zur Methode gehört, selbst benötigt.

Zu Beginn des Spiels sollte ein Sequenzdiagramm vorliegen. Dieses kann genau wie das Programm selbst von den Schülerinnen und Schüler zuvor angefertigt werden oder wird von der Lehrperson vorgegeben.

Grundsätzlich läuft das Spiel ebenso ab wie das Objektspiel. Die Schülerinnen und Schüler führen anhand des Sequenzdiagramms die einzelnen Aktionen der Objekte durch. Zusätzlich wird für jedes Objekt und jede seiner Methoden eine Darstellung dieser Methode aufgebaut. Dieser Aufbau besteht aus der Methodendeklaration und einem Behälter für jeden Parameter der Methode. Als Behälter können Schubladen (daher der Name), Klarsichtfolien oder andere Behälter dienen, welche sich gut sichtbar für die Lerngruppe aufbauen lassen. Die Behälter sind mit den Bezeichnern der jeweiligen Parameter beschriftet. Für die Methode `istGetroffen(hPosition, vPosition)` würden also zwei Behälter mit den Beschriftungen `hPosition` und `vPosition` benötigt. Weiterhin existiert für jedes Attribut jeweils ein Behälter mit dem Attributsbezeichner als Beschriftung.

⁵Ein Beispiel zur Durchführung des Spiels findet sich im Anhang A

Die Darstellung der Parameterübergabe durch das Legen von Blättern in die entsprechenden Schubladen sorgt für eine enaktive Repräsentation des Sachverhaltes. Die Schülerinnen und Schüler stellen die Parameterübergabe handelnd dar. Das verwendete Sequenzdiagramm präsentiert den Ablauf des Spiels ikonisch. Die symbolische Repräsentation erfolgt durch die verbale Erklärung des gerade durchgeführten Schrittes durch die Schülerinnen und Schüler.

Das Spiel wird nun kurz für die vier oben genannten Fälle beschrieben. Hierbei wird jedes Objekt von einer Spielerin bzw. einem Spieler verkörpert, der die Aktionen, welche das Objekt macht, im Rollenspiel durchführt.

Fall 1:

Wenn während des Spiels die Nachricht `istGetroffen(50, 60)` von "ObjektA" an "ObjektB" geschickt wird, so schreibt ObjektA den Werte "50" auf ein Blatt und legt das Blatt in den Behälter "hPosition", welcher zur Methode "istGetroffen()" gehört. Ebenso wird mit dem Wert "60" und dem Behälter "vPosition" verfahren. Objekt B nimmt nun diese beiden Blätter und führt damit, je nach Methodeninhalt, eine Aktion durch. Falls es sich bei der Methode um eine Anfrage handelt, kann auch für die Antwort ein Behälter zur Verfügung gestellt werden und die Antwort wird analog zur Parameterübergabe zurückgegeben.

Fall 2:

Angenommen "ObjektA" schickt die Nachricht `setzeRadius(50)` an "ObjektB". "ObjektA" nimmt nun ein Blatt und schreibt den Wert 50 darauf. Anschließend wird das Blatt in den Behälter, welcher zur Methode "setzeRadius()" beim ObjektB gehört, gelegt. "ObjektB" nimmt das Blatt, schreibt den Wert auf ein neues Blatt und legt dieses in den Behälter, der zum Attribut "radius" gehört (s. Abbildung 9).

Fall 3:

Zunächst läuft dieser Fall genauso ab wie Fall 2, anstatt das abgeschriebene Blatt mit dem Wert des Parameters aber in den Behälter für ein Attribut zu legen, legt "ObjektB" das Blatt in einen Behälter, der zu einer Methode eines dritten Objektes gehört. Dieses verfährt dann damit weiter.

Fall 4:

Wie Fall 3, allerdings legt "ObjektB" das Blatt in einen Behälter, der zu einer seiner eigenen

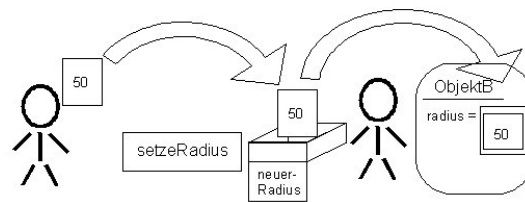


Abbildung 9: Ablauf des Schubladenspiels

Methoden gehört.

Die Darstellung der Parameter als Schubladen, in die Werte gelegt werden, entspricht der Parameterübergabeart call-by-value mit einfachen Datentypen. Falls Referenzen mittels call-by-value übergeben werden, treten Seiteneffekte auf, die mithilfe des Schubladenspiels nicht anschaulich dargestellt werden können. Die Darstellung ist auch für call-by-reference irreführend, da hierbei der Bezeichner des formalen Parameters lediglich zu einem Synonym für den aktuellen Parameter wird. Es existiert keine eigene Schublade für den formalen Parameter.

4.3 Referenzen auf Objekte

Durch das Schubladenspiel und vertiefende Übungen sind die Schülerinnen und Schüler mit dem Ablauf der Parameterübergabe vertraut. Im Umgang mit Referenzen auf Objekte treten allerdings neue Probleme auf, falls mehrere Referenzen auf ein Objekt verweisen. Es wird eine Darstellung benötigt, um zu verdeutlichen, dass nun mehrere Referenzen, d.h. verschiedene Bezeichner, auf dasselbe Objekt verweisen. Dementsprechend kann mit verschiedenen Referenzen dasselbe Objekt manipuliert werden. Werden Referenzen auf Objekte als Parameter übergeben, so tritt dieses Problem zwangsläufig auf.

Die Vorstellung von Behältern mit Werten ist hier nicht zielführend. Hier liegt bei der Einführung von Programmiersprachen wie Java ein Bruch vor. Den Schülerinnen und Schülern sollte zunächst einmal anhand eines Beispiels das Konzept der Verweise klargemacht werden.

Die Klasse Punkt definiert einen Punkt in einer Ebene, hat also zwei Koordinaten. Der Punkt kann in Richtung der beiden Koordinatenachsen verschoben werden, oder an bestimmte Koor-

dinaten gesetzt werden. Zur Klasse siehe Anhang B.

```

1 >>> punkt1 = Punkt(3,5)
2 >>> punkt2 = punkt1
3 >>> punkt2.verschiebePunkt(5,5)

```

Abbildung 10: Verweis auf Objekte (Python)

Da `punkt2` auf dasselbe Objekt verweist wie `punkt1`, wird durch die Anweisung `punkt2.verschiebePunkt(5,5)`; dieses Objekt verändert. `punkt1` hat also nicht mehr die Koordinaten (3,5) sondern (8,10). Die Verweise können graphisch durch Pfeildia-gramme dargestellt werden (Abbildung 11).

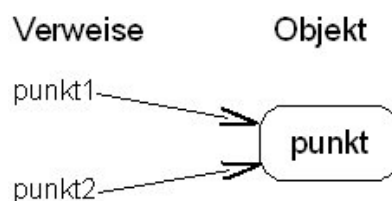


Abbildung 11: Graphische Darstellung von Verweisen

Wird ein Verweis auf ein Objekt als Parameter übergeben, so ist der Parameter lediglich ein weiterer Verweis und somit ein zusätzlicher Name für das Objekt, kann also ebenso graphisch dargestellt werden wie in Abbildung 11. Diese Darstellung ist zwar abstrakter als die Vorstellung von Behältern, aber an dieser Stelle für eine korrekte Vorstellung notwendig. Diese Darstellung kann mithilfe einer Aufgabe vertieft werden. (s. Anhang C).

Diese Darstellung lässt sich auch auf die Parameterübergabe anwenden. Ein formaler Parameter verweist bei der Wertübergabe einer Referenz auf dasselbe Objekt wie der aktuelle Parameter.

4.4 Vertiefung der Lerninhalte

Eine Festigung und Vertiefung der gelernten Prinzipien der Parameterübergabe kann durch mehrfache Wiederholung gewährleistet werden (vgl. [Bov04], S.239f). Neben vertiefenden

schriftlichen Übungen (auch an der Tafel), bietet sich hier die Durchführung eines Projektes an, das auf mehrere Unterrichtseinheiten verteilt wird. Bei der Auswahl eines Projektes ist darauf zu achten, dass den Schülerinnen und Schülern zum einen bereits alle darin enthaltenen Konzepte der Objektorientierung bekannt sind. Zum anderen sollte zum Üben der Parameterübergabe darauf Wert gelegt werden, dass die verschiedenen Objekte des Programms viel miteinander kommunizieren. Ein Beispiel für eine sich über mehrere Unterrichtsstunden erstreckendes Projekt ist die Implementierung einer Kasse eines Multiplexkinos mit mehreren Kinosälen und Anzeigetafeln. Hier wird der Lebenskontext der Schülerinnen und Schüler miteinbezogen, da der Besuch und die Funktionsweise eines Multiplex den Schülerinnen und Schüler aus ihrem Privatleben vertraut ist. Eine Steuerkonsole übernimmt durch Methodenaufrufe die Steuerung der Anzeigetafeln. Dieser Teilaspekt kann in einer vorgeschalteten Unterrichtseinheit implementiert werden. Ein weiteres motivationsförderndes Beispiel, bei der die Schülerinnen und Schüler auch ihre literarische Kreativität einbringen können, ist die Entwicklung eines eigenen Textadventures. Hier werden alle begehbaren Räume von einer Klasse "Raum" erzeugt. Im Verlauf des Unterrichts kann dieses Beispiel so weiterentwickelt werden, dass die Klasse "Gegenstand" eingeführt wird, von der alle auffindbaren Gegenstände abgeleitet sind. Diese Gegenstände können über ihre Methoden benutzt (aufgerufen) werden und treten so in Interaktion mit der sie umgebenden Umwelt, die wiederum durch Objekte realisiert ist. Am Beispiel des Textadventures können im weiteren Unterrichtsverlauf neu hinzugekommene Prinzipien der Objektorientierung, wie etwa Vererbung⁶, geübt und alte Lerninhalte wiederholt werden. Vorschläge zur Umsetzung der genannten Projekte und weitere Vorschläge finden sich unter [Jak06].

⁶Eine Klasse, die von einer Vaterklasse abgeleitet ist, besitzt die Attribute und Methoden der Vaterklasse, sie "erbt" sie.

5 Umsetzung

5.1 Umsetzung des Schubladenspiels

Im Folgenden wird eine Einzelstunde aus einem Grundkurs im Fach Informatik in der Jahrgangsstufe 11 beschrieben. Die Stunde fand in der 19. Unterrichtswoche des laufenden Schuljahres statt.

Die Schülerinnen und Schüler waren zum Zeitpunkt der Durchführung des Schubladenspiels bereits mit dem Objektspiel vertraut. Sie kannten Objekt-, Klassen- und Sequenzdiagramme. Weiterhin hatten sie bereits einige Programme selbst geschrieben. Das letzte Programm war ein vereinfachtes Billardspiel⁷. Da die Schülerinnen und Schüler nach wie vor Probleme im Umgang mit Parametern hatten, sollte das Schubladenspiel die Funktionsweise von Parametern verdeutlichen.

Als Beispielprogramm wurde ein Programm gewählt, welches ähnlich dem Billardspiel funktionierte. Den Schülerinnen und Schülern wurde kurz die Funktionsweise des Programms erläutert, anschließend wurde die Lerngruppe in vier Gruppen eingeteilt, wobei jede Gruppe einen Arbeitsauftrag⁸ bekam. Drei Gruppen erhielten den Auftrag, jeweils einige Methoden des Programms zu implementieren. Eine Gruppe sollte aus einem vorgegebenen Storyboard ein Sequenzdiagramm erstellen. Sämtliche Ergebnisse wurden auf Plakate geschrieben. Währenddessen wurden von der Lehrperson die notwendigen Vorbereitungen für das Schubladenspiel getroffen. Die Schubladen und Objektkarten wurden im Unterrichtsraum aufgestellt. Beim Erstellen der einzelnen Programmteile hatten einige Gruppen Probleme, hauptsächlich im Umgang mit Parametern. Zudem waren die mathematischen Grundlagen zur Implementierung der Methode `istGetroffen()` nicht klar. Hier musste die Lehrperson unterstützend eingreifen. Das Sequenzdiagramm (s. Abbildung 12) enthielt einige Fehler. Anstatt der Methode `neuePosition(neuehPosition, neuevPosition)` haben die Schülerinnen und Schüler `zeichneKreis(neuehPosition, neuevPosition)` geschrieben. Der erste Balken auf der Lebenslinie des Objektes "kreis" ist zu lang. Diese Fehler führten während

⁷Für eine kurze Beschreibung des Programms Billardspiel s. Beispiel 1 S 12

⁸Für die Arbeitsblätter s. Anhang A

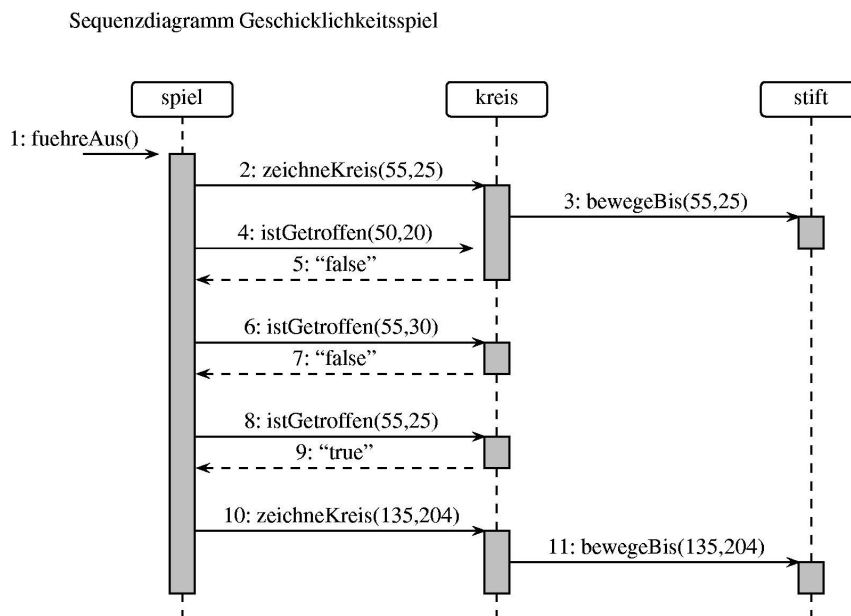


Abbildung 12: Sequenzdiagramm der Schüler

der Durchführung des Spiels jedoch zu keinen Problemen.

Anschließend stellten die einzelnen Gruppen ihre Ergebnisse kurz vor, wobei die Plakate im Klassenraum aufgehängt wurden. Anhand des Sequenzdiagramms hat nun ein Schüler vorbereitete Blätter mit den dem Sequenzdiagramm entsprechenden Werten in die Schubladen zur Methode `zeichneKreis()` gelegt. Eine Schülerin nahm diese Blätter und legte sie in die Schubladen zur Methode `bewegeBis`. Von dort wurden die Blätter an der Objektkarte - diese war ebenfalls vorbereitet - in eine Klarsichtfolie gelegt. Auf diese Weise wurde das Spiel weiterspielt. Aus Zeitgründen wurde das Sequenzdiagramm allerdings nicht bis zu Ende nachgespielt. Abschließend wurde der Stundeninhalt von einer Schülerin wiederholt.

6 Reflexion und Ausblick

6.1 Reflexion der Umsetzung des Schubladenspiels

Allgemein ist festzuhalten, dass das Rollenspiel zu einem früheren Zeitpunkt hätte stattfinden sollen. Eine erste Durchführung des Spiels ist auch schon vor den ersten Programmiererfahrungen möglich. Die Schülerinnen und Schüler benötigen lediglich Objektkarten und ein Sequenzdiagramm um den zeitlichen Ablauf im Schubladenspiel nachzuspielen. Sobald sie dann mit einfachen Sequenzen und Methodendeklarationen vertraut sind, kann das Spiel erneut durchgeführt werden. Später kann das Spiel mit etwas komplexeren Programmen wiederholt werden.

Auswahl des Beispielprogramms

Das Beispielprogramm für das Schubladenspiel sollte die vier in Kapitel 4.2.2 beschriebenen Fälle beinhalten. Weiterhin sollte das Programm relativ einfach aufgebaut sein. Diese Voraussetzungen sind beim Programm "Geschicklichkeitsspiel" erfüllt. Es ist generell wichtig, dass bei Beispielen von objektorientierter Modellierung zu jeder Klasse mehrere Objekte existieren. Dies ist beim Geschicklichkeitsspiel nicht erfüllt. Hier wäre eine Erweiterung denkbar. Es könnten sich mehrere Kreise auf einmal auf dem Bildschirm befinden. So würden mehrere Instanzen der Klasse Kreis und somit auch der Klasse Stift existieren.

Gruppeneinteilung

Beim Einstieg in die Programmierung, also nach der Einführung in die objektorientierte Modellierung, ist es möglich, mit den Schülerinnen und Schülern zunächst das Lesen einfacher Programme zu üben und sie auf diese Weise in die Syntax und Struktur von Programmcode einzuführen. Wenn dies bereits geschehen ist, aber die Schülerinnen und Schüler noch nicht selbstständig programmiert haben, kann das Schubladenspiel mit einem vorgegebenen Programm durchgeführt werden. Dies führt jedoch zwangsweise zu einer anderen Gruppeneinteilung als in Kapitel 5.1 beschrieben ist. Die einzelnen Gruppen könnten verschiedene Storyboards erhalten, um daraus Sequenzdiagramme zu erstellen.

Falls zu einem späteren Zeitpunkt das Spiel wiederholt wird und diesmal das Programm selbstständig implementiert werden soll, erscheint die arbeitsteilige Gruppenarbeit sinnvoll. Jeder Lernende hat sich so mit einem Teil des Programms auseinandergesetzt und jede Gruppe steuert einen wichtigen Teil zum Unterricht bei.

Durchführung des Schubladenspiels

Die Schülerinnen und Schüler haben ihre Aufgaben schnell und eigenständig bearbeitet. Es gab einige kleinere Probleme bei der richtigen Methodendeklaration. Größere Schwierigkeiten traten beim Berechnen des Abstandes der Mausposition vom Kreismittelpunkt in der Methode `istGetroffen` aus der Klasse `Kreis` auf. Diese Methode war vor allem deshalb schwierig, weil die Schülerinnen und Schüler mit Umgang mit Wahrheitswerten noch nicht vertraut genug waren. Nachdem die Vorgehensweise mit den Schubladen an der ersten Methode vom Lehrer erläutert wurde, konnten die Schülerinnen und Schüler das Gruppenspiel weitgehend selbstständig fortführen. Der Ablauf des Spiels war klar und bereitete den Teilnehmern keine großen Schwierigkeiten. Da die Schülerinnen und Schüler das Objektspiel bereits kannten, war nur das genaue Vorgehen mit den Parametern neu. Auch wenn an der Durchführung des Spiels nicht alle Schülerinnen und Schüler beteiligt waren, so waren doch alle durch die vorherige arbeitsteilige Gruppenarbeit mit dem Programm vertraut.

Bei der Durchführung ist wichtig, immer genau deutlich zu machen, welches Objekt gerade aktiv ist. Falls man das Schubladenspiel, wie in diesem Fall, anhand eines konkreten Programms durchführt, kann man zusätzlich zum Sequenzdiagramm an den Plakaten mit den einzelnen Methoden zeigen, an welcher Stelle im Programm sich das Spielgeschehen gerade abspielt. Dies kann durch die Schülerinnen und Schüler selbst geschehen und sollte nicht vorgegeben werden. Hierbei kann die Lehrperson beobachten wie gut die Schülerinnen und Schüler bereits Programmcode lesen und verstehen können.

Nach der Durchführung des Schubladenspiels verbesserte sich der Umgang mit Parametern im Unterricht. Lediglich die Häufigkeit des Vergessens der Typdeklaration oder das Einfügen der Datentypen beim Methodenaufruf wurde durch das Spiel nicht verändert. Da die Syntax einer

Programmiersprache letztlich eine feste Vorgabe ist, an die die Schülerinnen und Schüler sich gewöhnen müssen, erscheint hier vor allem das Üben und Verbessern durch die Lehrperson notwendig.

Zur genauen Überprüfung des Lernerfolges bietet sich kurz nach der Einführung in eine Programmiersprache eine Lernzielkontrolle an, die einige typische Situationen im Umgang mit Parametern abdeckt.

Wie in Kapitel 4.3 beschrieben, eignet sich das Schubladenspiel nur für die Parameterübergabe mit call-by-value mit einfachen Datentypen. Werden Referenzen auf Objekte übergeben, so kann dieses Problem durch eine weitere Darstellungsform gelöst werden (vgl. Kapitel 4.3). Das Schubladenspiel erfüllt dennoch seinen Zweck, die Funktionsweise der Parameterübergabe zu verstehen. Da sich einfache Datentypen und Objekte auch tatsächlich unterschiedlich verhalten, scheint eine unterschiedliche Repräsentation nicht nachteilig.

Der Unterschied zwischen call-by-reference und der Übergabe von Referenzen mittels call-by-value taucht im Anfangsunterricht in Regel nicht auf. Es ist fraglich, ob sie im Unterricht der gymnasialen Oberstufe befriedigend behandelt werden können, da hier auch bei Studierenden höheren Semesters und relativ erfahrenen Informatikern Probleme auftauchen. Für die grundlegenden Probleme im Umgang mit Parametern, ist eine enaktive Repräsentation, mithilfe von Objektspielen, ein wichtiges Element des Anfangsunterricht im Fach Informatik, auf welches, aufgrund der abstrakten Gestalt vieler Gegenstände der Informatik, nicht verzichtet werden sollte.

Abbildungsverzeichnis

1	Parameterübergabe (Java)	4
2	Parameterübergabe (mittels Call by value (Java))	5
3	Call by reference (PASCAL)	5
4	Call by value mit Referenzen auf Objekte (Java)	6
5	Objektkarten	12
6	Methode zeichne()	14
7	Objektkarte elkesAuto	18
8	Objektkarten Wartezimmer	19
9	Ablauf des Schubladenspiels	22
10	Verweis auf Objekte (Python)	23
11	Graphische Darstellung von Verweisen	23
12	Sequenzdiagramm der Schüler	26

Literatur

- [Bov04] BOVET G, HUWENDIEK(Hrsg.) (2004) Leitfaden Schulpraxis, Pädagogik und Psychologie für den Lehrberuf, Cornelsen, Berlin
- [Cla06] CLAUS V, SCHWILL A (2006) Duden, Informatik A-Z, Fachlexikon für Studium, Ausbildung und Beruf, Dudenverlag, Mannheim Leipzig Wien Zürich, 4. Aufl.
- [Har06] HARTMANN W, NÄF M, REICHERT R (2006) Informatikunterricht planen und durchführen, Springer, Berlin Heidelberg
- [Hum05] HUMBERT L (2005) Didaktik der Informatik mit praxiserprobtem Unterrichtsmaterial, Teubner, Wiesbaden
- [Jak06] JAKOBS M (2006) Material und Unterrichtseinheiten für den Informatik-Unterricht. Internet: <http://www.martinjakobs.de/index.php>
(Zugriff: 28.05.07 18:00 MEZ)
- [Kas05] KASTENS U (2007) Grundlagen der Programmiersprachen SS07. Internet: <http://ag-kastens.uni-paderborn.de/lehre/material/gps/fohlen/Folie601.html>
(Zugriff: 28.05.07 18:00 MEZ)
- [Lif07] LIFE³ (2007) life³ - Lernwerkzeuge für den Informatikunterricht: Einsetzen, Evaluieren und (Weiter)Entwickeln. Internet: <http://life.upb.de/>
(Zugriff: 28.05.07 18:00 MEZ)
- [Ull06] ULLENBOOM C (2006) Java ist auch eine Insel, Programmieren mit der Java Standard Edition Version 6. Internet: <http://www.galileocomputing.de/openbook/javainsel6/>
(Zugriff: 28.05.07 18:00 MEZ)
- [Wei06] WEIGEND M (2006) Objektorientierte Programmierung mit Python, mitp, Heidelberg, 3. aktualisierte und erweiterte Aufl.

A Verlaufsplan und Arbeitsblätter zum Schubladenspiel

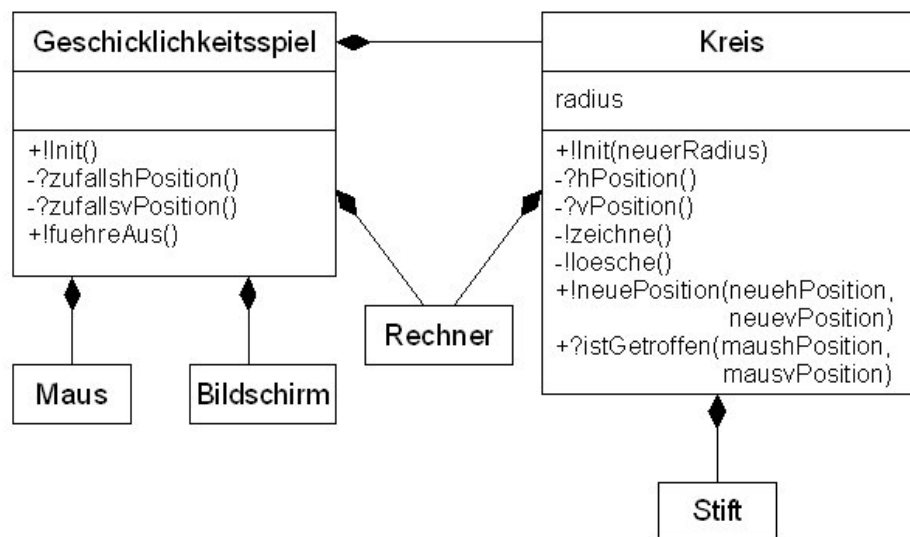
Geplanter Unterrichtsverlauf

Unterrichtsphase	Operationen/Sachaspekte	Aktions- /Sozialform	Medien
Einstieg	Erläuterung der Problemstellung und des Arbeitsauftrages. Klärung von Fragen.	LV	Arbeitsblatt
	Einteilung in Gruppen. Jede Gruppe erhält den Auftrag eine vorgegebene Methode auf ein Plakat zu schreiben	LV	Arbeitsblatt
Arbeitsphase	Die SuS arbeiten in Gruppen an den Arbeitsaufträgen.	Gruppenarbeit	Plakate
Präsentation	Jede Gruppe stellt ihr Plakat vor	SV	
Sicherung	Der Aufruf jeder Methode wird mit vorgeführt. Der Ablauf wird durch das Sequenzdiagramm vorgegeben.	Rollenspiel	Schubladenspiel
	Ausblick auf eine Erweiterung des Programms	UG	

Anlagen: Arbeitsblätter 1-4

Geschicklichkeitsspiel

Es soll ein einfach Geschicklichkeits bzw. Reaktionsspiel programmiert werden. In ein Fenster wird ein Kreis gezeichnet. Der Spieler muss nun den Mauszeiger in diesen Kreis bewegen. Hat er das geschafft, wird der Kreis gelöscht und an einer anderen Stelle neu gezeichnet. Nun muss der Spieler wieder den Mauszeiger in den Kreis bewegen usw. Das Programm wird durch einen Doppelklick mit der Maus beendet. Das UML-Klassendiagramm beschreibt die Klassen des Programms.



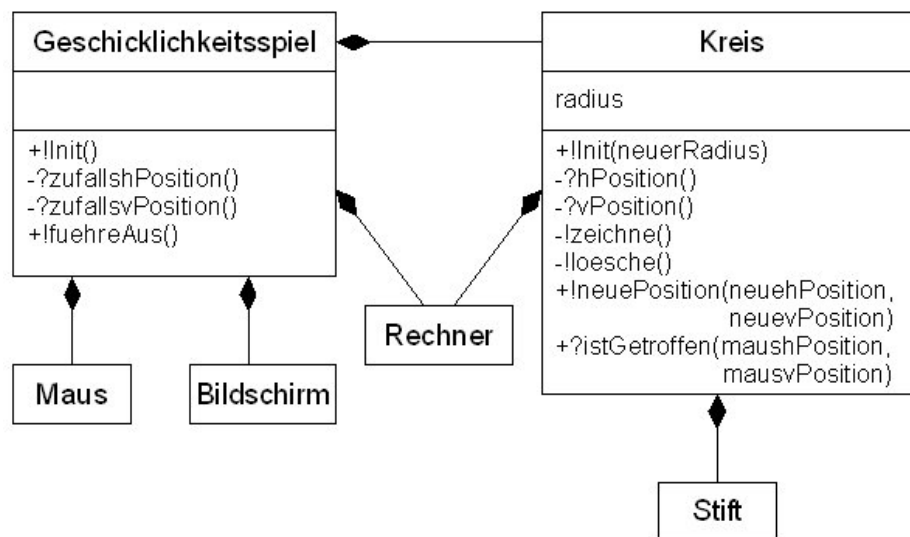
Arbeitsauftrag (Gruppe 1):

Programmieren Sie die Klasse "Geschicklichkeitsspiel". Im Konstruktor soll unter anderem ein Objekt der Klasse Kreis mit einem Radius von 10 Pixeln erzeugt werden. Die Methoden "neuehPosition" und "neuevPosition" geben eine zufällige Ganze Zahl zwischen 0 und der Bildschirmbreite bzw. der Bildschirmhöhe zurück. Die Methode "fuehreAus" führt das oben beschriebene Spiel aus. Über die Methode "istGetroffen" in der Klasse Kreis erfährt das Geschicklichkeitsspiel ob die Maus im Kreis ist.

Schreiben Sie die Methode "fuehreAus" auf ein Plakat.

Geschicklichkeitsspiel

Es soll ein einfach Geschicklichkeits bzw. Reaktionsspiel programmiert werden. In ein Fenster wird ein Kreis gezeichnet. Der Spieler muss nun den Mauszeiger in diesen Kreis bewegen. Hat er das geschafft, wird der Kreis gelöscht und an einer anderen Stelle neu gezeichnet. Nun muss der Spieler wieder den Mauszeiger in den Kreis bewegen usw. Das Programm wird durch einen Doppelklick mit der Maus beendet. Das UML-Klassendiagramm beschreibt die Klassen des Programms.



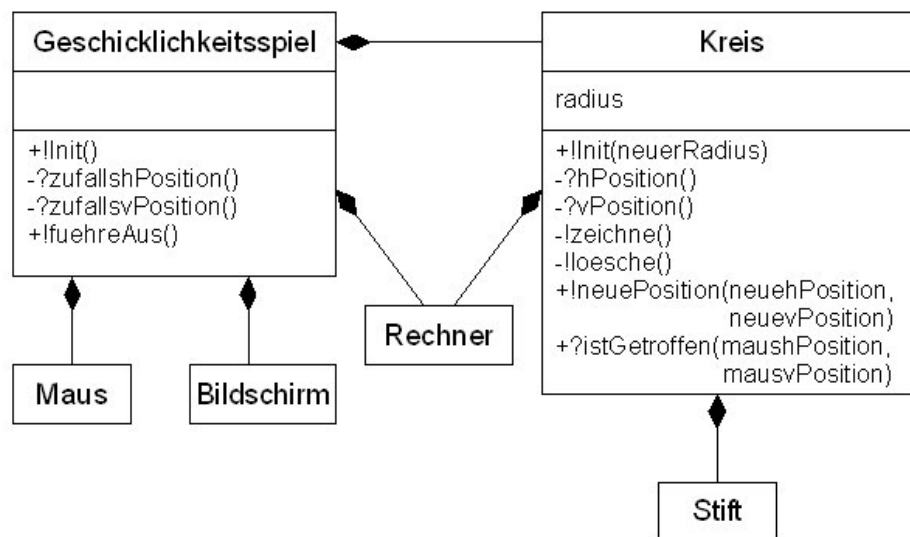
Arbeitsauftrag (Gruppe 2):

Programmieren Sie aus der Klasse "Kreis" den Konstruktor und die Methode "neuePosition". Der Kreis soll einen beliebigen Radius haben können der durch den Aufruf des Konstruktors festgelegt wird. In der Methode "neuePosition" wird der Kreis gelöscht und an der neuen Position neu gezeichnet.

Schreiben Sie den Konstruktor und die Methode "neuePosition" auf je ein Plakat.

Geschicklichkeitsspiel

Es soll ein einfach Geschicklichkeits bzw. Reaktionsspiel programmiert werden. In ein Fenster wird ein Kreis gezeichnet. Der Spieler muss nun den Mauszeiger in diesen Kreis bewegen. Hat er das geschafft, wird der Kreis gelöscht und an einer anderen Stelle neu gezeichnet. Nun muss der Spieler wieder den Mauszeiger in den Kreis bewegen usw. Das Programm wird durch einen Doppelklick mit der Maus beendet. Das UML-Klassendiagramm beschreibt die Klassen des Programms.



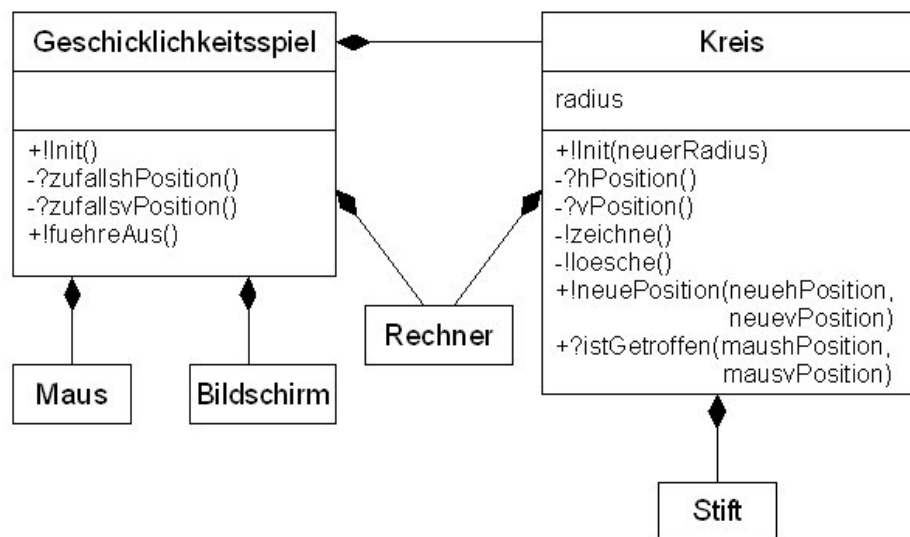
Arbeitsauftrag (Gruppe 3):

Programmieren Sie aus der Klasse “Kreis” die Methoden “istGetroffen”. Die Methode “istGetroffen” gibt den Wert “true” zurück, wenn der Abstand zwischen der Mausposition, welche durch die Parameter angegeben ist, und der aktuellen Stiftposition kleiner als der Radius ist. Sonst gibt sie “false” zurück.

Schreiben Sie die Methode “istGetroffen” auf ein Plakat.

Geschicklichkeitsspiel

Es soll ein einfach Geschicklichkeits bzw. Reaktionsspiel programmiert werden. In ein Fenster wird ein Kreis gezeichnet. Der Spieler muss nun den Mauszeiger in diesen Kreis bewegen. Hat er das geschafft, wird der Kreis gelöscht und an einer anderen Stelle neu gezeichnet. Nun muss der Spieler wieder den Mauszeiger in den Kreis bewegen usw. Das Programm wird durch einen Doppelklick mit der Maus beendet. Das UML-Klassendiagramm beschreibt die Klassen des Programms.



Arbeitsauftrag (Gruppe 4):

Zeichnen Sie ein Sequenzdiagramm zu einem Programmablauf. Das Sequenzdiagramm enthält die Objekte "geschicklichkeitsspiel", "kreis" und "stift". Zeichnen Sie das Sequenzdiagramm auf ein Plakat.

Das Storyboard finden Sie auf der Rückseite.

Storyboard:

1. Jemand weist das geschicklichkeitsspiel an das Spiel auszuführen.
2. geschicklichkeitsspiel weist den "kreis" an, sich an der Position(55,25) zu bewegen.
3. geschicklichkeitsspiel fragt "kreis" ob er von der Maus"getroffen" wurde.(istGetroffen(50,30))
4. "kreis" antwortet "false"
5. geschicklichkeitsspiel fragt den "kreis" ob er getroffen wurde. Mausposition: (55,30)
6. "kreis" antwortet "false"
7. geschicklichkeitsspiel fragt den "kreis " ob er getroffen wurde. Mausposition: (55,25)
8. "kreis" antwortet "true"
9. geschicklichkeitsspiel weist den "kreis" an, sich an die Position(135,204) zu bewegen.
10. der "kreis" weist den Stift an sich an Position(135,204) zu bewegen

B Klasse Punkt

```
1 class Punkt(object):
2     def __init__(self , xKoordinate , yKoordinate):
3         self.__xKoordinate = xKoordinate
4         self.__yKoordinate = yKoordinate
5
6     def xKoordinate(self):
7         return self.xKoordinate
8
9     def yKoordinate(self):
10        return self.yKoordinate
11
12    def verschiebePunkt(self , xDistanz , yDistanz):
13        self.xKoordinate = self.xKoordinate+xDistanz
14        self.yKoordinate = self.yKoordinate+yDistanz
15
16    def setzePunkt(self , xKoordinate , yKoordinate):
17        self.xKoordinate = xKoordinate
18        self.yKoordinate = yKoordinate
```

C Aufgabe: Verweise auf Objekte

Stellen Sie die Objekte und Verweise zu folgenden Sequenzen schrittweise graphisch dar. Zeichnen Sie also zunächst die Situation nach der ersten, anschliessend nach der zweiten Anweisung usw.

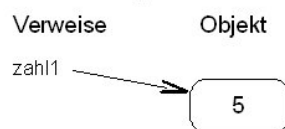
```

1 >>> zahl1 = 5
2 >>> zahl2 = 7
3 >>> zahl3 = 1
4 >>> zahl2 = zahl1
5 >>> zahl1 = zahl3
6 >>> zahl1 = zahl2+zahl3

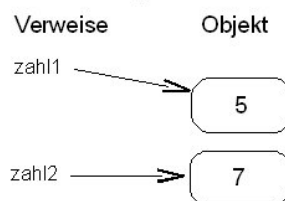
```

Lösung:

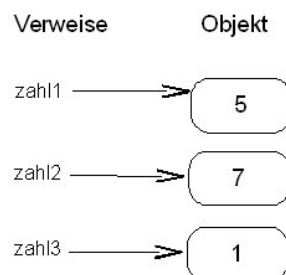
1. Anweisung



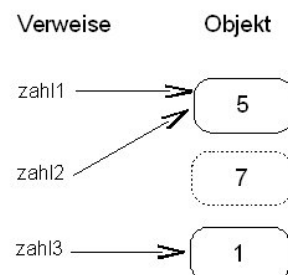
2. Anweisung



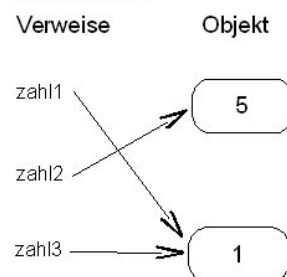
3. Anweisung



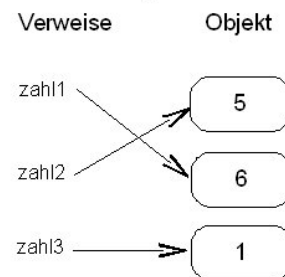
4. Anweisung



5. Anweisung



6. Anweisung



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.

Bönen, 28. Mai 2007