Objektorientierter Anfangsunterricht in der Oberstufe mit der Java-Entwicklungsumgebung BlueJ

Hausarbeit zum Hauptseminar

„Objektorientierte Programmierung im Anfangsunterricht“

von Siegfried Spolwig,

gehalten im Wintersemester 2003/ 04 an der Humboldt-Universität Berlin,

erstellt von Michael Opel und Peter Rüßmann

Inhaltsangabe

1. Einleitung

2. Warum objektorientierter Anfangsunterricht?

Die Rahmenpläne zum Aufbau des Informatikunterrichts haben in den vergangenen zwei Jahrzehnten einen großen Wandel durchgemacht, der sich auch in den Schulbüchern dieser Zeit widerspiegelt. Mitte der achtziger Jahre wurde die zentral auf die Vermittlung von Algorithmen ausgerichtete Schulinformatik überwunden. Beispielhaft sein hier Hui/ Bestmann [1] zitiert, die über ihr Lehrbuch schreiben: „Gewisse Kapitel ...(fehlen), z. B. das Arbeiten mit einem Programmpaket, wie Tabellenrechnen, Graphik oder Textverarbeitung. Hier ist aber weniger Theorie nötig als praktische Übung.“ Hui/ Bestmann haben ein Schulbuch vorgelegt, das als eines der Letzten dem Algorithmus als einzig wesentlichem Lerninhalt verpflichtet war und eine umfängliche Sammlung klassisch-elementarer Verfahren der modernen Mathematik (z.B. zu Sortieralgorithmen und deren Anwendungen) enthält.

Mit H. Balzert ist der Paradigmenwechsel der Schulinformatik hin zum problem- und anwendungsorientierten Informatikunterricht verbunden. Schon 1982 veröffentlicht er mit [2] ein Lehrbuch, in dem die „(m)ethodisch-didaktische Gleichbehandlung von Datenstrukturen und Algorithmen“ durchgängig eingehalten wird. Der Begriff des abstrakten Datentyps tritt bei Balzert noch nicht auf. Doch diesen Schritt geht 1990 das Autorenteam des Metzler-Verlags in [3]. Ziel der Autoren war „... eine Didaktik, die von Anwendungen ausgehend Probleme formuliert, sie algorithmisch löst und diese Lösungen in eine Programmiersprache umsetzt, damit sie auf dem Computer ablauffähig sind ...“. Dieses Lehrbuch setzt bei der Behandlung seiner Inhalte konsequent ein 3-Phasen-Modell zur Darstellung des Prozesses der Softwareentwicklung ein[1]. Grundlegend genutzte Konzepte sind dabei das Modulkonzept und das Konzept der abstrakten Datentypen. Wichtig ist den Autoren auch der Projektunterricht, zu dem der Lehrerband viele nützliche Handreichungen enthält.

Schon aus dem bisher Gesagten ergibt sich eine deutliche Verschiebung der schulischen Ausbildungsschwerpunkte. Die 1980 vorhandene starke Betonung der Programmier- und Algorithmenkompetenz wurde ab 1990 zugunsten der Modellierungskompetenz drastisch reduziert. Dabei spiegelt dieser Wechsel des Kompetenzschwerpunkts nur die Softwarekrise von 1970 in ihrer Auswirkung auf die Didaktik wieder, die in der Informatik auch zu einer Konzentration auf die Neuformulierung der Grundlagen der Systembeschreibung und –entwicklung führte. So wurde bereits 1967 mit SIMULA67 bereits die erste Programmiersprache mit ausgereiftem Klassenkonzept entwickelt, darauf aufbauend ab 1970 die Programmiersprachen Eiffel und SmallTalk, die ausgefeilte Konzepte zum Botschaftenaustausch zwischen Objekten enthalten. Unterstützt durch die bahnbrechenden Leistungssteigerungen der Rechner seit Ende der achtziger Jahre tritt das Paradigma der Objektorientierung seinen Siegeszug an.

---

[1] (I) Problem (Anwendung); (II) Lösung (Algorithmus, Pseudocode); (III) Programm (in Pascal) und Ausführung (Maschinenprogramm)

So ist es auch nur natürlich, dass die Entwicklungen der Informatik der vergangenen dreißig Jahre auch wieder auf die Didaktik zurückwirken und in den neunziger Jahren des vergangenen Jahrhunderts eine breite Diskussion um die richtige Weiterentwicklung der Didaktik der Informatik begann. Nachdem neben dem Paradigma der Objektorientierung auch Programmiersprachen und Rechner zu dessen Anwendung zur Verfügung standen, wurde die Frage nach einem konsequent objektorientierten Anfangsunterricht in der Schulinformatik gestellt. Erste Antworten darauf findet man seit Anfang der neunziger Jahre und es sei hier beispielhaft das Unterrichtskonzept von S. Spolwig [4] erwähnt. Die Seminaraufgabe, nämlich eine konsequent objektorientierte Unterrichtsplanung des Unterrichts im Fach Informatik für das Anfangshalbjahr in der Oberstufe zu erstellen, ist somit eine aktuelle, reizvolle und praxisrelevante Aufgabe.

3. Bedingungsanalyse

3.1. Lehrvoraussetzungen und –erwartungen der Schüler

Die Vorraussetzungen zum Anfangsunterricht in Informatik werden zuerst im Mathematik-Unterricht gelegt. Die grundlegenden begriffe der Mathematik haben allesamt Analoga in der Begriffs- und Methodenwelt der Informatik. Dies gilt auch für die Modellbildung der imperativen, insbesondere der objektorientierten Programmierung. Anders als beim rein funktionalen Programmieransatz tritt bei der objektorientierten Programmierung auch die Modellierung von zeitbehafteten Prozessen in den Fordergrund. Die damit verbundenen Schwierigkeiten fallen jedoch im Anfangsunterricht aufgrund der Einfachheit der genutzten Beispiele nicht besonders ins Gewicht. Die unter dem objektorientierten Paradigma vorgenommene methodische Verschränkung von Attributen und Methoden ist den Schülern schon aus dem Mathematikunterricht bekannt, wobei sie dort offenbar nicht besonders betont wird. Das Beispiel der Menge $\mathrm{P} = \{p | p \text{ ist reelles Polynom}\}$ zusammen mit $(\mathrm{P},+,\cdot)$, dem Ring der Polynome, verdeutlicht den objektorientierten Aspekt mathematischer Begriffsbildung. Ziel des Mathematikunterrichts ist nun aber nicht die Ausarbeitung dieses Aspekts, der seine zentrale Bedeutung dann erhält, wenn eine konkrete Implementierung, also eine technische Realisierung z. B. auf einem Computer, Unterrichtsgegenstand ist. Von daher böte es sich durchaus an, Mathematik und Informatikunterricht stärker miteinander zu verbinden, was nicht zuletzt auch im Sinne des fächerübergreifenden Unterrichts wäre. Dieser Gedanke wird aber von den informatik-didaktischen Fachleuten nicht betont, was seinen Grund auch in den Schülererwartungen an das Fach Informatik haben kann. Die Schülererwartungen sind nach unseren, durchaus beschränkten Erfahrungen im Detail unklar, doch auf praktisches Tun am Rechner fixiert. Fragt man Schüler, welche konkreten Lernziele sie im Informatikunterricht haben, so wird man ohne Frage weit häufiger die Antwort „Ich möchte das Web verstehen und eine Webseite programmieren, zum Beispiel mit XML!“ hören, als „Ich möchte eine Realisierung des Rings der Polynome implementieren!“

Die Betonung des Praxisrelevanz und des Systemgedankens in der Didaktik der Informatik nutzt die Motivationslage der Schüler, doch sind auch hier Fallstricke zu berücksichtigen, denn oftmals sind die Schülerziele technisch zu anspruchsvoll, um als Ansatzpunkt zur Vermittlung methodisch-systematischer Grundlagen zu taugen. Hier wird der Lehrer seine Entscheidungs- und Steuerungskompetenz einsetzen. Unabhängig davon ist der spielerische und praktische Zugang zur Arbeit mit dem Rechner eine so gewaltige didaktische Möglichkeit, dass der Lehrer sie unbedingt bestmöglich nutzen sollte. Das Verständnis der Wirkung einer Methode einer Instanz einer Klasse wird sich im konkreten Tun am Rechner besser vermitteln, als durch eine abstrakte Beschreibung dieses Vorgangs. Dies war einer der Gründe, die Java-Entwicklungsumgebung BlueJ als DV-technische Grundlage für das zu planende Einstiegssemester in die objekt-orientierte Programmierung zu wählen.

BlueJ unterstützt alle Lernebenen, angefangen bei der enaktiven Ebene. Die Feature-Überfrachtung anderer kommerzieller Systeme hat BlueJ nicht, so dass die Konzentration auf das Wesentlich zusammen mit einem deutlich geringeren Einarbeitungsaufwand verbunden ist. Es ist allgemein bekannt, dass kommerzielle Entwicklungssysteme, die dem MVC-Prinzip folgen, in ihren Werkzeugen den Oberflächen-Layout-Aspekt zu sehr betonen. Zum Einstieg in die Objektorientierung ist das aber, wie BlueJ zeigt, gar nicht nötig.

3.2. Der Unterrichtsstoff und seine Kompetenzanforderungen

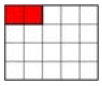
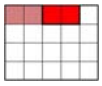
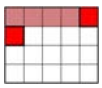
Zentraler Gegenstand des Unterrichts zum Einstieg in die objekt-orientierte Programmierung ist der Klassenbegriff der Informatik. Die Modellierung von Klassen als Gebilden, die aus Attributen und Methoden bestehen, steht dabei unabdingbar im Fordergrund, ebenso die Einsicht in den natürlichen Zusammenhang zwischen den Begriffen Klasse und Objekt. Zur graphisch-ikonischen Darstellung wird die UML-Beschreibung genutzt werden. Dabei wird UML im Anfangsunterricht nicht systematisch als Beschreibungsmöglichkeit eingeführt, sondern einzig in seinen grundlegenden Sprachelementen als Dokumentations- und Modellierungswerkzeug genutzt, um darüber den Transfer von der „quasi-ikonischen“ Ebene auf eine strengere symbolische Ebene zu realisieren.

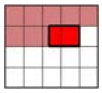
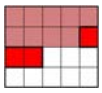
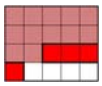
Grundlegend für die Beschäftigung mit Programmierung ist, unter welcher Paradigmen-Sammlung sie auch immer geschieht, ist eine ausreichende Kenntnis und Erfahrung mit den grundlegenden syntaktischen Konstrukten einer Programmiersprache und ein gesichertes Wissen ihrer Semantik. Dies ist wohl allgemein anerkannt und daher ist es klar, dass dieser Aspekt wichtig für die Planung der Reihe war. Nicht zuletzt auch, weil dadurch bei der konkreten Programmierung Vorwissen zur Interpretation von Fehlermeldungen des Compilers (die natürlich auch unter Java durchaus kryptisch sein können) bereit steht. Die Erweiterung der syntaktischen Möglichkeiten ist dabei nicht Schwerpunkt der Semesterplanung. Sie soll parallel zur Vertiefung des objekt-orientierten Ansatzes geschehen. Dabei sind natürlich neben den primitiven Datentypen, dem Variablen-Begriff und den Elementaroperationen auch die grundlegenden Kontrollstrukturen (if-then-else, for, while) zu behandeln. Hier, aber auch zur Darstellung von einfachen Algorithmen, werden Nassi-Sneiderman-Diagramme zur Formalisierung eingesetzt werden. Deren Nutzung geschieht dabei analog zur Nutzung von UML.

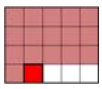
Die vorgenannten Inhalte sind allesamt notwendig beim Einstieg in die objekt-orientierte Programmierung. Der Vererbungsbegriff ist zentral für die gesamte Methode und muss daher behandelt werden. Für das Einstiegssemester ist eine Reduktion auf die elementaren Vererbungsbeziehungen zur Herausarbeitung der Idee vorgesehen. Dies vermeidet Schwierigkeiten, die notwendig mit der Behandlung komplexerer Vererbungskonzepte verbunden sind und trägt dem Spiralansatz des Lehrplansaufbaus Rechnung. So kann der Interface-Begriff in einem höheren Semester behandelt werden.

Die Semesterplanung fußt dabei in vielen Teilen auf der Einführung von D. J. Barnes und M. Kölling [6], deren Buch ja nicht nur für Studierende, sondern aufgrund der praktischen Elternerfahrung der Autoren mit dem Informatikunterricht ihrer Kinder, vor allem für Schüler und ihre Lehrer geschrieben wurde.

4. Grobplanung der gesamten Unterrichtsreihe

Woche	Thema	Behandelte Konzepte	Inhalte/ Bemerkungen/ Materialien
 1 - 2	**Objekte und Klassen I**	Objekt und Klasse Attribut und Methode Java Sprachelemente: Primitive Typen Zuweisung =	**Begleitende(s) Aufgabenstellung (Java Programm):** Zeichenebene mit Kreis, Rechteck und Dreieck Beschreibung von Objekten (Zeichenebene zusammen mit Quadrat, Dreieck und Kreis) Klassifizierung Attribut/ Methode Transfer: I) Vorstellung von einem Objekt und konkretes Objekt in der Welt, II) Klasse und Objekt Java-Objekte erzeugen und ihre Methoden aufrufen mit BlueJ **Erstelltes Programm:** Zeichenebene I (parameterlose Methoden)
 3 - 4	**Objekte und Klassen II**	Aufbau der Java Klassendefinition I Klassendefinitionen erweitern Methoden mit Parametern Singleton-Pattern Java-Sprachelemente: if ..else; for; +=, -= Kommentare	**Begleitende(s) Aufgabenstellung (Java Programm):** Zeichenebene mit Kreis, Rechteck und Dreieck Attribute, Methoden und Codierungs-Standards Parameter einer Methode Quellcode erstellen und testen (ausprobieren) Mehrere Instanzen einer Klasse Rolle der Zeichenebene (singleton) **Erstelltes Programm:** Zeichenebene II (parameterbehaftete Methoden)
 5 - 6	**Hilfsmittel der BlueJ Entwicklungs-umgebung**	Arrays (primitive Typen, Objekten) BlueJ Debugger Java-Sprachelemente: object array while (block) Java Docu: System.out	**Begleitende(s) Aufgabenstellung (Java Programm):** Verwaltung einer Bücherliste (Array) Ausgabe: System.out.println Ablaufdiagramme (Nassi-Sneiderman) Nutzung des Debuggers (step, step into, contiune, halt, terminate, break point, object inspector) **Erstelltes Programm:** Bücherliste

Woche	Thema	Behandelte Konzepte	Inhalte/ Bemerkungen/ Materialien
 7 – 9 (1. Teil)  7 – 9 (2. Teil)	**Problemanalyse und Modellierung**	Konstruktoren mit und ohne Parametern Signatur einer Methode Lokale Variablen Java Bibliotheken get- und set-Methoden	**Begleitende(s) Aufgabenstellung (Java Programm):** Bestandteile und Arbeitsweise eines Fahrkarten-automaten in zwei Detaillierungstiefen (Prototyp, Release) beschreiben Objekt –und Klassendiagramme (UML) Rückgabe einer Methode Information Hiding **Erstelltes Programm:** Fahrkarten-automat I (Prototyp)+ II (Release)
 10 - 12	**Beziehungen zwischen Klassen**	Abstraktion/ Vererbung Aufruf von Konstruktoren Java-Sprachelemente: new	**Begleitende(s) Aufgabenstellung (Java Programm):** Bestandteile und Funktionsweise der Anzeige einer Computeruhr beschreiben Problemanalyse und Festlegung der Aufgaben in den einzelnen Klassen Darstellung der Abstraktion/ Vererbung in Objekt- und Klassendiagrammen **Erstelltes Programm:** Simulation der Anzeige einer Computeruhr
 13 - 16	**Projekt**	Sammlungen von Objekten Klassenbibliotheken nutzen Java-Sprachelemente: java.util.ArrayList import statement instanceOf, cast final	**Aufgabenstellung (Java Programm):** Modellierung eines computergestützten Auktionssystems für Fahrräder, Mofas und Motorrädern Objekt- und Klassendiagramm entwerfen Strukturierung mit Sammelklassen Methoden der Klasse ArrayList (add(Object), size(), get(Object)) Dokumentation von Bibliotheks-klassen lesen Implementierungsaufgaben in Subteams lösen Präsentation und Diskussion der Ergebnisse **Erstelltes Programm:** Bike-Auction-System

Woche	Thema	Behandelte Konzepte	Inhalte/ Bemerkungen/ Materialien
 17 - 17	Klausur		

5. Beispiele für Unterrichtsentwürfe

In diesem Kapitel werden die ersten beiden Unterrichtssequenzen eines Semesters in der Oberstufe eines objektorientierten Anfangsunterrichts mit der Java-Entwicklungsumgebung BlueJ beschrieben.

Im ersten Teil werden die zentralen Begriffe „Klasse, Objekt, Instanz, Attribut, parameterlose Methode“ der OOP eingeführt und die Java-Entwicklungsumgebung BlueJ erläutert.

Im zweiten Teil werden die Schwächen parameterloser Methoden diskutiert und darauf aufbauend Methoden mit Parameter eingeführt.

In beiden Teilen werden benötigte Java-Sprachelemente erläutert.

5.1. Einführung des Objekt- und Klassenbegriffs

Um sich in der Welt der objektorientierten Programmierung zurechtzufinden ist die Kenntnis der grundlegenden Begriffe und Konzepte notwendig: Objekte und Klassen. Am Ende des Unterrichtsblockes sollen Schüler ein Grundverständnis davon haben, was Objekte und Klassen sind und wie sie miteinander in Beziehung stehen.

Die Abbildung der realen Welt in einem Programm wird durch Objekte realisiert und diese müssen in einem Modell im Rechner geeignet repräsentiert werden. Objekte können kategorisiert werden und Klassen beschreiben alle Objekte einer solchen Kategorie.

Die Einführung dieser abstrakten Begriffe im Anfängerunterricht überfordert viele Schüler. Daher müssen wir sie behutsam heranführen, indem wir (nicht nur) ein Beispiel betrachten.

5.1.1. Einstieg in die objektorientierte Programmierung

Der Entwurf einer Verkehrssimulation scheint in diesem Zusammenhang ein besonders komplexer Ansatz zu sein, aber um die Begriffe Klasse und Objekt gegeneinander abzugrenzen scheint er bestens geeignet. Jeder Schüler kann sich eine solche Simulation vorstellen. Ein Begriff der darin mit Sicherheit verwendet wird ist „Auto“. Aber was meinen wir mit dem Begriff Auto, was stellt in unserem Entwurf ein Auto dar? Bezeichnet es eine Klasse oder ein Objekt? Die Schüler können dies aktuell nicht entscheiden, da sie weder den Unterschied kennen, noch verstanden haben, was die Begriffe Klasse und Objekt beinhalten.

Einige Fragen können uns aber dabei helfen eine Entscheidung trotzdem zu treffen und die Begriffe voneinander zu trennen.

Fragen wir z.B.

- Welche Farbe hat ein Auto?
- Welche Höchstgeschwindigkeit, kw-Zahl, Reifengröße, ... hat ein Auto?
- Wo befindet sich ein Auto?

so bemerken die Schüler sofort, dass diese Fragen im Allgemeinen nicht beantwortet werden können. Sie machen erst Sinn, wenn wir sie zu einem konkreten Auto stellen, z.B. das Auto des Lehrers. Der Grund ist, das wir von der Klasse „Auto“ ausgegangen sind. In unserer Simulation werden wir von Autos im allgemeinen sprechen und keine Aussagen über ein konkretes Auto machen.

Das konkrete Auto des Lehrers ist ein Objekt (besser: eine Instanz - also ein Vertreter) der Klasse „Auto“. Wir können die Farbe, die Höchstgeschwindigkeit und die Position des Autos nennen.

Im weiteren Verlauf wollen wir die Begriffe vertiefen. Ausgehend von einem Bild, das wir aus einem BluJ –Projekt entnommen haben. Dieses Bild wird den Schülern über Beamer bzw. OH-Projektor vorgestellt. Es wird ein Kreis, zwei Quadrate und ein Dreieck in verschiedenen Farben und Größen angezeigt, die an unterschiedlichen Positionen gezeichnet sind. Die Figuren sind so angeordnet, das sie zu einem Bild mit dem Titel „Haus mit Sonne“ zusammengefaßt werden können (s. Abb. 1).

Abb. 1 Material für den Anfangsunterricht

Werden die Schüler aufgefordert das Bild zu beschreiben, erhalten wir ziemlich sicher die Beschreibung „Haus mit Sonne“. Für eine geometrische Interpretation des gezeigten Bildes sind Hinweise des Lehrers notwendig (die Impulsfrage könnte lauten: Beschreibt die Figuren, aus denen das Bild zusammengesetzt ist). Bestandteile der „realen Welt“ auf eine abstrakte Repräsentation zurückzuführen ist für Schüler anfangs problematisch. Der Lehrer muß an

dieser Stelle geeignet bei der Lösungsfindung unterstützen, ohne die Motivation der Schüler zu beeinträchtigen.

Erkennen die Schüler bereits auf dieser Ebene eine Beziehung zwischen dem großen roten und dem kleinen schwarzen Quadrat, dann scheinen die eingangs vermittelten Begriffe verstanden und verinnerlicht worden zu sein. Die Schüler sollten jetzt bereits entscheiden können, ob es sich bei den gezeigten geometrischen Figuren um Klassen oder um Objekte handelt.

Eine weitere Abstraktion erreichen wir durch nähere Betrachtung einer speziellen Klasse. Die Schüler beschreiben die Eigenschaften der Objekte, wie Farbe und Größe und damit bestimmte Attribute einer Klasse. Jedes Objekt hat hierbei eine bestimmte Ausprägung, z.B. das Auto des Lehrers ist rot, das Quadrat hat die Seitenlänge 1cm. Das Ziel der ersten Sequenz ist erreicht, wenn die Schüler von den Ausprägungen einzelner Objekte auf die Struktur der Klasse zurück schließen können. Aussagen über die Struktur einer Klasse zu machen erlauben uns die Objekte der Wirklichkeit am Rechner genau zu beschreiben (Stichwort Modellierung).

Die Struktur einer Klasse anhand ihrer Attribute und deren Ausprägungen zu beschreiben erfordert etwas tieferliegendes Programmier-Know-How. Wir müssen uns mit Attributen und deren Ausprägungen näher beschäftigen. Hier kommen wir auf die Grundlagen jeder Programmiersprache zurück, den elementaren Datentypen und dem Variablenkonzept (das Variablenkonzept wird hier nicht in voller Tiefe behandelt, sondern es werden nur die Begriffe gebraucht, ohne nähere Erläuterungen).

Ein elementarer Datentyp gibt an, welche Art von Information eine Variable annehmen kann. So bezeichnet der Typ „int“ (Abkürzung von Integer) zum Beispiel eine ganze Zahl und der Datentyp zum Attribut Farbe des Autos ist String (Zeichenkette). Fragen die Schüler die Begriffe „Variable“ und „elementarer Datentyp“ nach, erläutert der Lehrer diese. Er kann hier bereits ausholen und dieses Thema aus der 2. Unterrichtssequenz vorziehen. Eine knappe Erläuterung, wie

> „eine Variable steht für einen Platzhalter. Die Klasse reserviert hier (technisch gesehen) Platz im Speicher, zum Ablegen der Information (die Ausprägung der Eigenschaft eines Objekts). Der Name der Variablen dient uns als Hinweis auf diesen Speicherort“

genügt, den Schülern meist, um mit dem Begriffen arbeiten zu können.

Weitere elementare Datentypen werden zeitnah bei Bedarf eingeführt.

5.1.2. Konkretisierung der Begriffe

Einen tieferen Einblick erhalten wir erst bei Betrachtung eines Projekts in BlueJ. Wir wählen für den Einstieg ein sehr simples Projekt, mit dem zentrale Eigenschaften von Objekten demonstriert werden sollen. Es wurde aus einer Vorlage von der Homepage des

BlueJ-Projekts (www.bluej.com) entnommen und wesentlich für die erste Unterrichts-Sequenz vereinfacht (es handelt sich hier ursprünglich um das Projekt „Figuren“).

Auf Übungsblatt 1 werden den Schülern Aufgaben gestellt, die sie mit dem Programmiersystem BlueJ am Rechner lösen sollen. Sie haben bisher keine Einweisung in BlueJ erhalten und sollen in dieser Stunde das System inspizieren, die Funktionalität kennen lernen und spielerisch mit Klassen und Objekten umgehen. Ergebnisse zu den gestellten Aufgaben sind auf dem Arbeitbogen festzuhalten. Dieser wird anschließend eingesammelt und gibt dem Lehrer einen Überblick über den erreichten Kenntnisstand der Schüler im Umgang mit dem System.

Die interaktive Entwicklungsumgebung BlueJ ist speziell für den Anfangsunterricht konzipiert und macht es selbst blutigen Anfängern leicht, auch ohne Kenntnisse der objektorientierten Programmierung fertig erstellte Programme auf ihre Struktur hin zu untersuchen und diese auszuführen (d.h. Objekte beliebiger Klassen erzeugen und diese geeignet zu modifizieren). Schülern wird gezeigt wie Klassen bzw. wie Objekte „aussehen“. Sie haben den Eindruck, Objekte werden greifbar (und damit auch be-greifbar). Langwierige Einweisungsphasen können somit entfallen.

In Aufgabe 1 des Aufgabenblattes werden die Schüler aufgefordert das Projekt Figur.I zu öffnen (siehe Anhang Projekt Figur.I). Die Schüler sehen eine Klasse Kreis und eine Klasse Leinwand am Bildschirm, die durch einen gestrichelten Pfeil verbunden sind. Durch einen Rechtsklick auf die Klasse Kreis, sehen die Schüler die Aktionen, die mit der Klasse Kreis durchgeführt werden können (s. Abb. 2).

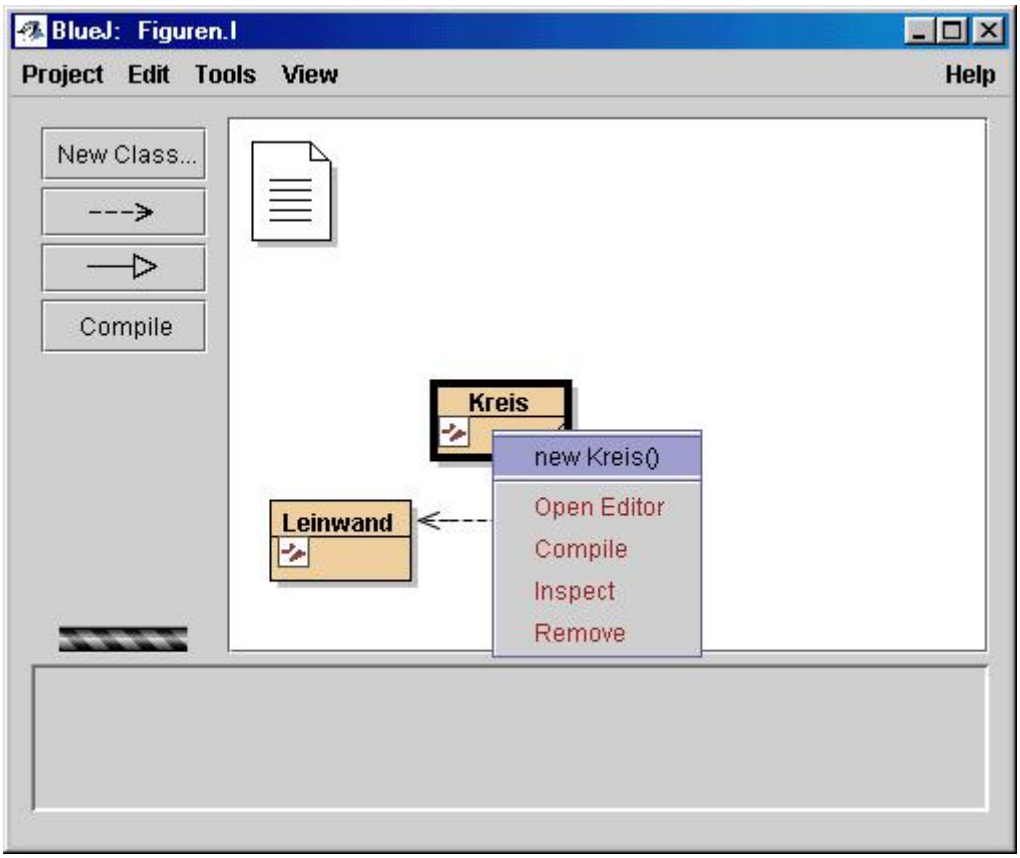


Abb. 2 Kontext-Menu einer Klasse

„new Kreis()“ aus dem Kontext-Menü erzeugt eine Instanz der Klasse Kreis – unser erstes Objekt. In der Objektleiste werden alle erzeugten Instanzen angezeigt.

Wir haben nun die Möglichkeit, verschiedene Instanzen der Klasse Kreis zu erzeugen, die anschließend auf dem Bildschirm angezeigt werden können.

In Aufgabe 2 werden die Schüler aufgefordert die Figuren anzuzeigen (in einem Fenster, das wir hier die "Leinwand" nennen). Dies geschieht durch einen Rechtsklick auf die Instanzen in der Objektleiste und Auswahl des Eintrags „void sichtbarMachen()“(s. Abb. 3). Sie können die Figuren dann interaktiv manipulieren: ihre Position verändern, sie auf der Leinwand anzeigen und wieder von der Leinwand löschen. Im folgenden sollen die Schüler versuchen spielerisch mit dem Programm weitere Kreise zu erzeugen und diese ebenfalls zu manipulieren.

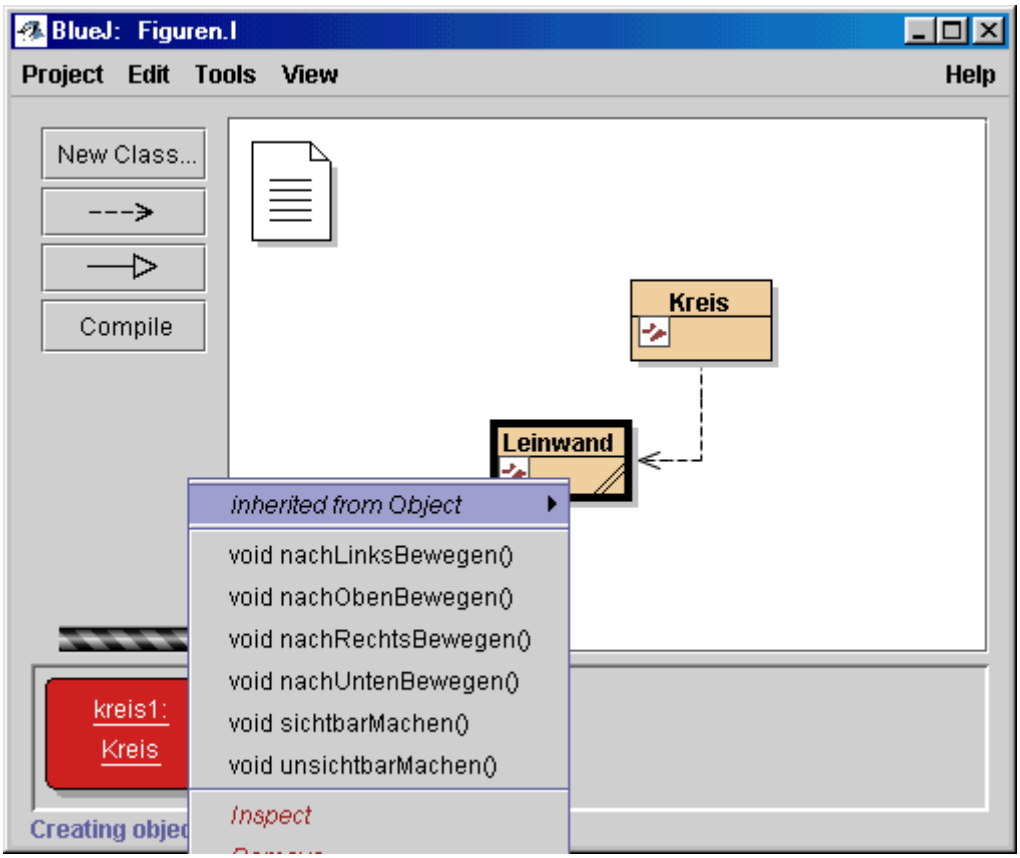


Abb. 3 Kontext-Menü eines Objekts

Die in Abb. 3 gezeigten Aktionen, die an einem Objekt durchgeführt werden können, werden in der objektorientierten Programmierung „Methoden“ genannt. Sie werden in der Klasse analog den Attributen definiert.

Der Lehrer muss die Bearbeitungsphase der Aufgaben mehrmals unterbrechen und für die Schüler sogenannte Ruhepunkte einziehen. Hierbei handelt es sich um Stationen der Zusammenfassung und der Erläuterung, wobei die unterschiedliche Stärke der Gruppen ausgeglichen werden kann und um die Anpassung von unterschiedlichen Arbeitsständen zu ermöglichen. Hier ist es wichtig Ergebnisse und Teile der Lösung bzw. Lösungswege an der Tafel festzuhalten. Aber auch neu aufgeworfene Fragen können in einem Plenum diskutiert werden. Der Lehrer sollte hierbei die Rolle des Moderators übernehmen.

In dieser Art werden alle Begriffe der zweiten Unterrichtsstunde eingeführt. Der Begriff der Methode kann für Schüler in der Form einer Aktion festgelegt werden. Diese Aktion bewirkt eine Änderung der Attribute eines Objekts (Änderungsmethode) bzw. können Methoden dazu genutzt werden, Attribute eines Objekts auszulesen (sog. sondierende Methoden). Schüler finden diese Funktionstrennung allein durch Beobachtung und Analyse der Objektmethoden. Unsere Klasse Kreis besitzt in unserer vereinfachten Version nur Methoden, die Attribute verändern können.

Um sich über die aktuellen Ausprägungen aller Attribute zu informieren wird die Klassenfunktion „inspect“ erläutert. Die Schüler werden aufgefordert mit der Klasse Kreis, den erzeugten Objekten und der Entwicklungsumgebung zu experimentieren.

In diesem (spielerischen) Sinne haben die Schüler jetzt bereits die Unterrichtsziele der zweiten Stunde erreicht.

Arbeitsblatt Nr. ... Klasse: ... Datum: ... Name: ...

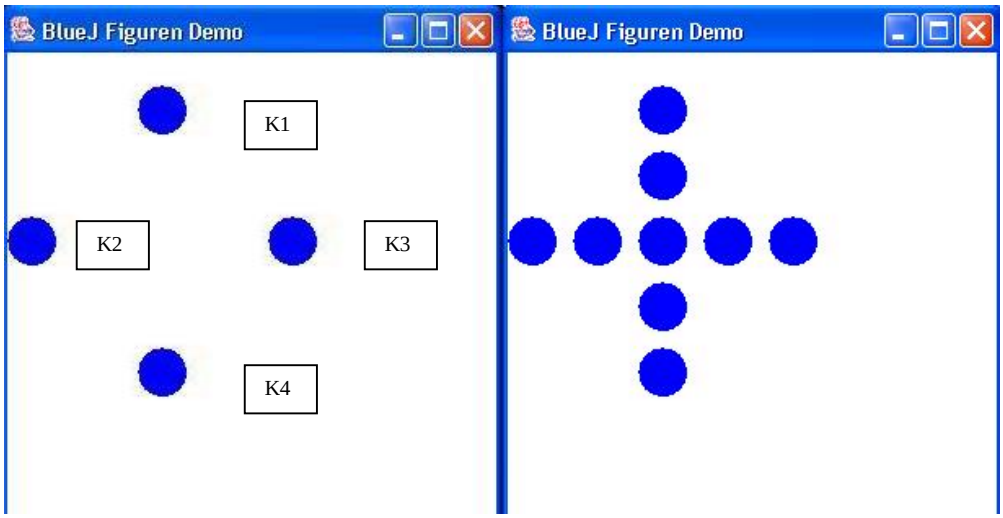
Thema: Klasse und Objekt

Aufgabe 1. Starte die Java-Entwicklungsumgebung BlueJ (Icon am Desktop) und öffne das Projekt Figur.I (es befindet sich im Verzeichnis ...).

Aufgabe 2. Erzeuge eine Instanz der Klasse Kreis, blende die Leinwand ein und bewege den Kreis auf der Leinwand.

Beschreibe deine Vorgehensweise:

Aufgabe 3. Gegeben sind die folgenden Bilder:



Durch die Methoden der Klasse Kreis wird ein Objekt dieser Klasse auf der Leinwand bewegt.

Stelle die Anordnung im 1. Bild (vier Kreise) her und fülle die folgende Tabelle aus.

Methode:	Anzahl Aufrufe K1	Anzahl Aufrufe K2	Anzahl Aufrufe K3	Anzahl Aufrufe K4
nachRechtsBewegen()				
nachLinksBewegen()				
nachObenBewegen()				
nachUntenBewegen()				

Versuche das 2. Bild nachzubilden. Überlege, wie du die Aufgabe schneller lösen kannst (Was fällt dir beim Betrachten der Methode „inspect“ auf). Beschreibe.

Abb. 4 Arbeitsblatt „Klassen und Objekte. Einführung in BlueJ“

5.1.3. Vertiefungsphase

Die Schüler haben in den letzten beiden Stunden den Unterschied zwischen Klasse und Objekt kennen gelernt. Sie können das Entwicklungssystem BlueJ nutzen um Objekte bestehender Klassen zu erzeugen und diese zu modifizieren.

Der Unterricht beginnt mit einer Zusammenfassung der Ereignisse und Ergebnisse der letzten beiden Unterrichtsstunden. Die Schüler versuchen eine geeignete informatische Definition der bisher kennen gelernten Begriffe zu finden. Der Lehrer muss diesen Prozess unterstützen, indem er die Wortbeiträge der Schüler hinterfragt, um Fehlinterpretationen und Unklarheiten zu verhindern. Ziel dieser Unterrichtsphase ist es ein Vokabular zu entwickeln, mit dem im folgenden Semester (und natürlich darüber hinaus) gearbeitet werden kann.

In zweiten Teil der Stunde soll der Begriff Klasse substantiiert, d.h. programmtechnisch bearbeitet werden. Hierzu öffnen die Schüler das in der vorhergehenden Stunde behandelte BlueJ-Projekt Figur.I. Im Kontex-Menü der Klasse Kreis wird der Eintrag „Open Editor“ ausgewählt. Es erscheint das BlueJ-Editor-Fenster mit dem Quellcode der Klasse Kreis. Die Schüler werden aufgefordert den Code zu analysieren und den Aufbau bzw. die Struktur des „Kreis-Programms“ wiederzugeben (s. Abb. 5).

Erwartet werden das Entdecken der Methodenimplementationen: „Das sind ja die Aktionen/Methoden, die wir im Kontext-Menü der Objekte aufgerufen haben!“. Mit gespannter Erwartung sollte der Lehrer die Erläuterungen zum Konstruktor eines Objekts erwarten: „da ist eine Funktion, die genauso heißt wie die Klasse!“. Nach der Code-Analyse wird eine Diskussionsrunde eingelegt, in der die Schüler sich über die Entdeckungen Gedanken machen. Gut ist, wenn es einen Schüler gibt, der bereits Erfahrungen mit der Programmierung besitzt. Aber dies kann i.a. nicht vorausgesetzt werden. Dadurch wird die Rolle des Lehrers in dieser Runde mehr zum aktiven Partner, der das nötige Know-How mitbringt.

Der Blick in das Editorfenster ist meist nicht so sehr spektakulär, da

1. viele Schüler Programmier-Know-How von zu Hause mitbringen oder
2. der Code sehr ausführlich dokumentiert ist (wie in unserem Beispiel).

Daher ist es eher leicht die unterschiedlichen Teile des Codes zu verstehen und den bisher gelernten Begriffen (z.B. elementarer Datentyp, Implementation einer Methode, Konstruktor einer Klasse) zuzuordnen.

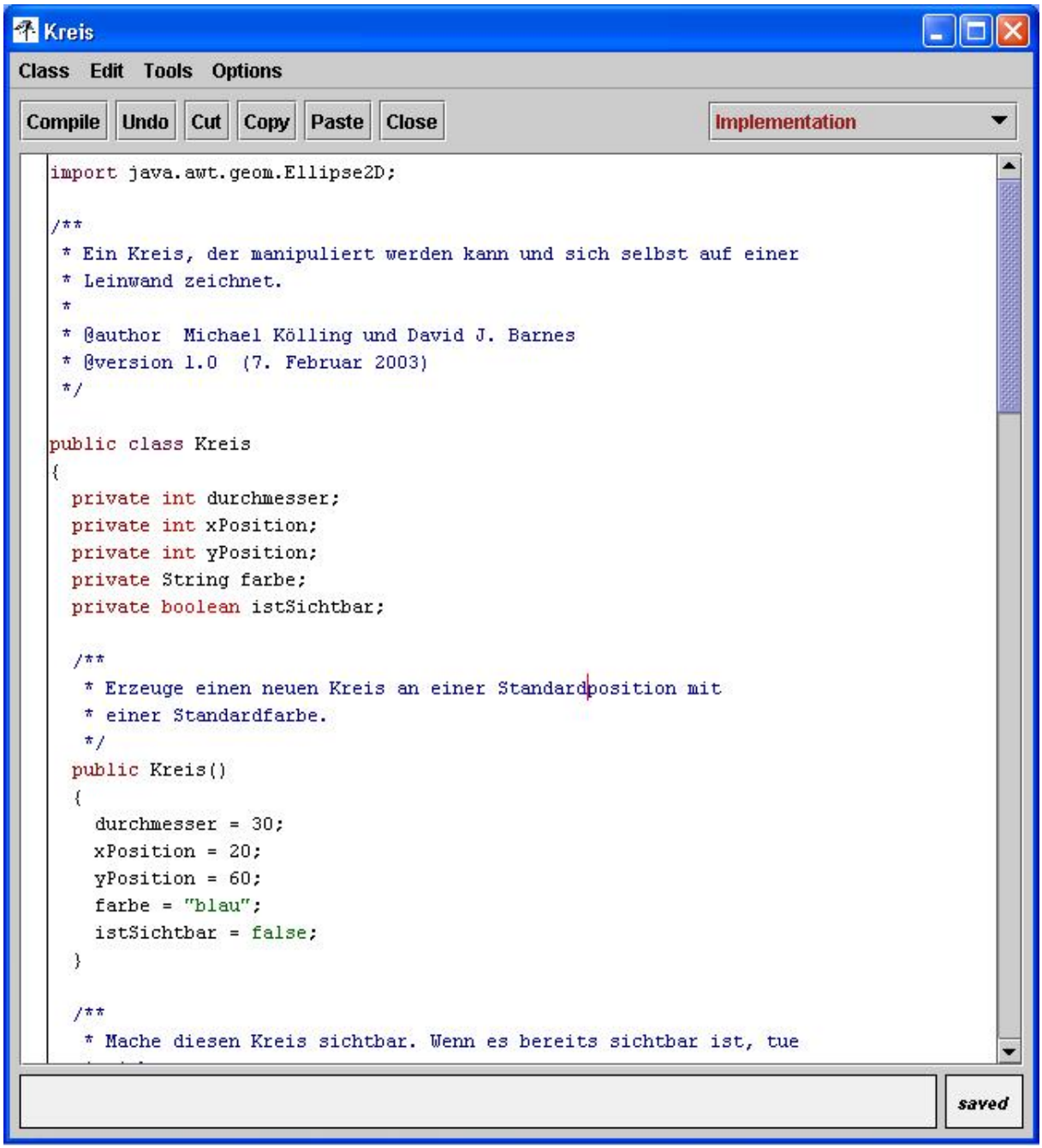


Abb. 5 Editor-Fenster Klasse Kreis

Auch hier ist es wieder erlaubt Änderungen am Code durchzuführen, die zu unterschiedlichen Ergebnissen führt:

1. einfache Änderungen am Konstruktor vornehmen (z.B. durchmesser vergrößern bzw. verkleinern oder Farbe auf „rot“ umsetzen), die ein ablauffähiges Programm hinterlassen.

2. Änderungen in der Methodendefinition, z.B. die Methode „loeschen“ umbenennen zu „loesche“ führt zwangsläufig auf eine Fehlermeldung, die im folgenden analysiert wird.
3. Änderungen im Methodenrumpf (erhöhen der Schrittweite).

Die Schüler erwarten nach beenden des Editors im Kontext-Menü den Eintrag „new Kreis()“ der aber erst nach einem „compile“, bzw. nach einem „rebuild package“ zur Verfügung steht. Auch hier steht der Lehrer als Wissender zur Seite und liefert die nötigen Impulse, falls notwendig.

Während dieser Phase werden spielerisch Java-Sprachelemente erlernt. Es scheint keine Schwierigkeiten zu bereiten, wenn im Code von der *Anweisung* „durchmesser = 30“ gesprochen wird, bzw. von „farbe = ‚blau‘“. Verstärkt wird dies durch den direkten Bezug zu den realen Objekten. Ein Kreis besitzt einen Durchmesser und (wenn er gefärbt ist) eine Farbe. Einzig die Operatoren „+=“ und „-=“ bereiten etwas Kopfschmerzen und müssen näher erläutert werden. Hierbei wird der Lehrer die Langform einführen und den Hintergrund (Speicheroperationen) zitieren.

Nach dieser sogenannten „Spielphase“ formuliert der Lehrer die Frage nach dem allgemeinen Aufbau einer Klasse in Java als Hausaufgabe und verweist auf diverse Java-Dokumente im Internet, die zur Lösung herangezogen werden dürfen.

5.1.4. Fazit der ersten Unterrichtssequenz

Dieses Projekt ist als ein einführendes Beispiel in die objektorientierte Programmierung gedacht. Es illustriert einige Kernkonzepte:

- ein Java-Projekt (eine Anwendung) ist eine Sammlung von Klassen (hier eine Klasse)
- Objekte werden mit Hilfe von Klassen erzeugt (sie sind Instanzen ihrer Klassen)
- von jeder Klassen können beliebig viele Objekte (Instanzen) erzeugt werden
- Objekte haben Operationen (Methoden)
- Objekte speichern Zustände (in ihren Feldern)
- alle Exemplare einer Klasse haben die gleichen Operationen und Felder
- die Werte, die in den Feldern gespeichert sind, können für jedes Objekt unterschiedlich sein

Dieses Projekt demonstriert außerdem einige Aspekte von BlueJ:

- Objekterzeugung

- interaktive Aufrufe von Operationen

Als Fortführung der Übung kann das Projekt „Zeichnung" angesehen werden, das dieses Projekt um drei Klassen erweitert. Diese Klassen (wie das Projekt „Zeichnung" genannt) benutzt die Figuren, um einen Zeichnung zu erstellen. Wir haben das Ergebnis des Projekts „Zeichnung“ bereits in der ersten Stunde diskutiert, es handelte sich um das Bild „Haus mit Sonne“.

Zu den drei Unterrichtsstunden zur Einführung des Klassenbegriffs wurden die nachfolgenden Verlaufsplanungen aufgestellt.

Phase/ Ziel/ Zeit (1. Std.)	**Unterrichts-gegenstand**	**geplantes Lehrerverhalten**	**erwartetes Schülerverhalten**	**Sozialform/ Medien**
Einführung 10‘ (10‘)	Verkehrssimulation	L. leitet Unterrichtsgespräch zur Vorbereitung in das Thema und stellt Fragen zu der Klasse Auto	Sch. Versuchen diese zu beantworten und bemerken, dass diese nicht allgemein zu beantworten sind.	UG
Einstieg/ Problemstellung 5‘ Phase der Schwierigkeiten 30‘ (45‘)	Klasse und Objekt	L. erläutert Unterschied am Beispiel Auto und zeigt Bild „Haus mit Sonne“. L. fordert auf, das Bild zu beschreiben. Impulsfrage: beschreibt aus welchen Figuren das Bild zusammengesetzt ist. L. deutet Gemeinsamkeit bei den beiden Quadraten an und fordert auf den Unterschied zu nennen. L. erläutert den Zusammenhang Klasse/Objekt (auch mit Rückgriff auf Auto) und fragt nach der Bedeutung der Farbe/Größe. L. fragt nach Ausprägungen der Attribute und ihren Wertebereich. L. fragt nach Beispielen für elementare Datentypen	Sch. Benennen die geometrischen Objekte, die auf dem Bild zu sehen sind: Dreieck, Kreis, Quadrat. Sch. Nennen die unterschiedliche Größe der Quadrate. Sch. Erkennen diesen Zusammenhang und nennen Farbe und Größe der Objekte ihre Eigenschaften/Ausprägungen/Attribute. Sch. Nennen mögliche Ausprägungen und finden für den Begriff Wertebereich den Begriff Typ. Sch. Geben Integer, Float, Char, Bit ... mit Beispielen an.	UG Beamer Stift und Papier

Phase/ Ziel/ Zeit (2.Std.)	**Unterrichts-gegenstand**	**geplantes Lehrerverhalten**	**erwartetes Schülerverhalten**	**Sozialform/ Medien**
Einstieg 10’ (10‘)	Rechner Projekt in BlueJ	L. erläutert das Ziel der Unterrichtsstunde: eigenverantwortliches Einarbeiten in die Entwicklungsumgebung BlueJ mit Hilfe eines Aufgabenblattes.	Sch. Übernehmen die Erläuterungen	UG PC
Problematisie-rungsphase 25‘ (35‘)	Aufgabenblatt	L. strukturiert Unterrichtsgeschehen durch Vorgabe von a) Arbeitsauftrag (Arbeitsblatt) b) Arbeitsform (Gruppenarbeit) c) Zeitplan (30 min.) d) erwarteten Resultaten e) Weiteres Vorgehen (Besprechung im Plenum) f) Konkrete Gruppeneinteilung L. beobachtet in zurückhaltender Moderatorenrolle Arbeitsfortschritte der Sch. L. und geeignete Sch. geben ggf. Hilfen bei der Bearbeitung der Aufgaben. L. beendet Gruppenarbeitsphase.	Sch. finden sich in ihren Arbeitsgruppen zusammen und bearbeiten das Arbeitsblatt Sch. haben Aufgaben1 und 2 (erste Hälfte von 3) mindestens grundlegend bearbeitet und diskutieren Ansätze für mögliche Lösungen des zweiten Teils von Aufgabe 3 Sch. tragen Ergebnisse zu den Aufgaben 1 bis 3 vor und führen ggf. Korrekturen in ihren Arbeitsblättern durch	UG Tafel Arbeitsheft Gruppenarbeit (Beamer)
Auflösung 10’ (45‘)	Aufzeichnungen im Arbeitsheft	L. zeigt Lösung des Aufgabenblatts am Beamer (Ausfüllen nach Meldungen der Sch.) L. leitet Besprechung zu Lösungsansätzen der Sch. zur zweiten Hälfte der Aufgabe 3	Sch. tragen Ergebnisse zu den Aufgaben 1 bis 2 (3) vor und führen ggf. Korrekturen in ihren Arbeitsblättern durch Sch. stellen Lösungsansätze zur zweiten Hälfte der Aufgabe 3 vor und beantworten Nachfragen	UG Beamer Stift und Papier

Phase/ Ziel/ Zeit (3.Std.)	**Unterrichts-gegenstand**	**geplantes Lehrerverhalten**	**erwartetes Schülerverhalten**	**Sozialform/ Medien**
Wiederholung 10’ (10’)	Wdh. der bisherigen Ergebnisse	L. führt Kontrolle durch und leitet Unterrichtsgespräch	Sch. kontrollieren ihre Aufzeichnungen und ergänzen ggf.	UG Arbeitsheft
Einstieg und Durchführung 30’ (40‘)	Java-Code der Implementierungen der Klasse Kreis	L. präsentiert am Beamer das Code-Fenster der Klasse Kreis. L. gibt Arbeitsauftrag für den Rest der Stunde an, nämlich die Analyse und erste Code-Änderungen durchzuführen und strukturiert den weiteren Ablauf durch Vorgabe von a) Arbeitsauftrag b) Arbeitsform (3er-Gruppen) c) Zeitplan (15 min.) L. beendet diese Arbeitsphase L. fordert Sch. auf, ihre Ergebnisse an der Tafel zu notieren. Ihre Code-Änderungen sollen am Beamer nachgestellt und erläutert werden (mit Test)	Sch. nehmen ihre neuen Arbeitsplätze ein und beginnen mit der Analyse Sch. nehmen Code-Änderungen vor und testen diese Sch. nehmen ihre gewöhnliche Sitzordnung ein Sch. schreiben wesentliche Ergebnisse der Analyse und der Code-Änderungen eigenverantwortlich mit	UG Beamer Arbeitsheft Rechner 3er-Gruppen am Rechner
Hausaufgabe 5’ (45‘)		L. stellt die Hausaufgabe und erläutert ggf. Nachfragen der Sch.	Sch. schreiben die Hausaufgabe in ihr Arbeitsheft	Arbeitsheft Stift und Papier

5.2. Erweiterung des Klassenbegriffs[2]

Mit dem Ende der ersten Unterrichtssequenz haben die Schüler grundlegende Vorstellungen von den Begriffen

- Klasse und Objekt
- Attribut und Methoden einer Klasse

Ebenso sind ihnen elementare syntaktische Konstrukte (primitive Datentypen, Attribut- und Methoden-Deklaration, Zuweisung) und deren Semantik bekannt. Hierauf basierend soll in dieser Unterrichtssequenz, die wieder über zwei Wochen mit jeweils einem dreistündigen Unterrichtsblock geplant ist, zweierlei erreicht werden:

- Erweiterung des Methodenbegriffs auf parametrisierte Methoden
- Erweiterung der Java-Sprachelemente durch elementare Kontrollstrukturen

Die Nutzung des Objekts der Klasse Leinwand im Rahmen Figuren-Projekts wird zum Abschluss des Blocks thematisiert. Dies bedeutet, dass sich die Schüler mit der Idee des Singleton-Pattern vertraut machen müssen. Ziel ist dabei, dass die Schüler als eine bewusste, hier von der Modellierungssituation her passende Designmöglichkeit erkennen.

Im folgenden wird der Aufbau des ersten dreistündigen Unterrichtsblocks dieser Sequenz skizziert.

5.2.1. Einstieg und Problematisierungsphase

Zu Beginn der Reihe soll schülerzentriert die Fragestellung nach den Mängeln der ersten Version des Figuren-Programms bearbeitet werden. Dadurch stehen zuerst Aspekte der Sozial- und Methodenkompetenz im Fordergrund des Unterrichtsgeschehens. Die Schüler sollen die Mängel der ersten Version benennen und selbsttätig dahingehend klassifizieren, dass sie zu einer Taxometrie der Mängel gelangen. Diese Taxometrie soll zum einen Abhängigkeiten zwischen einzelnen Mängeln aufzeigen, zum anderen eine Festlegung des erwarteten Aufwands zur Behebung der Mängel beinhalten. Die Arbeitsformen während dieses Abschnitts der Reihe werden dabei vornehmlich die Gruppenarbeit und die Diskussion im Plenum sein. Der Lehrer wird aus einer zurückgenommenen Moderatorenrolle heraus steuernde Impulse setzen. Dabei bearbeiten die Schüler folgende zwei Arbeitsaufträge während der Gruppenarbeitsphase:

I) Stellen sie Mängel der ersten Modellierung der Kreis-Klasse zusammen und ordnen sie diese Mängel hinsichtlich

   a. ihres Gewichts für einen Nutzer der Klasse Kreis,

   b. ihrer Abhängigkeiten untereinander in einem Pfeil-Diagramm

II) Formulieren sie möglichst präzise Systemanforderungen an ein erweitertes zwei Model der Klasse Kreis und klassifizieren sie diese Anforderungen hinsichtlich des von ihnen erwarteten Realisierungsaufwands (gering, mittel, hoch).

Abb. 6 Arbeitsaufträge der Problematisierungsphase

[2] erstellt von Peter Rüßmann

Ziel dieser Phase ist, dass die Schüler auf der Grundlage der festgestellten Mängel eine Abarbeitungsliste von zu lösenden Problemen erstellen, die zum Teil in den nachfolgenden Einheiten der Reihe bearbeitet werden. Dabei werden die Ergebnisse der einzelnen Gruppen im Rahmen einer Besprechung im Plenum konsolidiert und in eine quasi „verbindliche“ Liste der zu realisierenden Systemanforderungen überführt. Schon hier kann auf das dazu analoge Vorgehen z. B. bei Kundenprojekten in Informationsindustrie hinweisen.

Die folgende Liste enthält die hier mindestens aufzulistenden Mängel bzw. Probleme:

Nr.	**Mangel der ersten Version**	**Resultierende Systemanforderung**
1	Objekte bewegen sich stets nur um genau eine Distanz, dabei sollte man sie auch (bequem) über größere Entfernungen bewegen können.	Implementiere eine Methode (bzw. Methodenerweiterung), mit der die Distanz, um die sich ein Objekt bewegt, vom Nutzer „frei“ gewählt werden kann. *Geschätzter Aufwand: gering*
2	Objekte haben stets nur genau eine Farbe, dabei sollten sie beliebige Farben annehmen können.	Implementiere eine Methode (bzw. Methodenerweiterung), mit der die Farbe eines Objekts vom Nutzer „frei“ gewählt werden kann. *Geschätzter Aufwand: gering, vergleichbar zu Nr.1*
3	Objekte bewegen sich stets nur in eine „Elementarrichtung“ (rechts, links, oben, unten), dabei sollten auch „kompliziertere“ Bewegungen möglich sein („Kreisbewegung“).	Implementiere eine Methode (bzw. Methodenerweiterung), mit der der Nutzer eine „Kreisbewegung“ mit einem Objekt durchführen kann. *Geschätzter Aufwand: hoch, noch nicht geeignet genau spezifiziert:* *a) was genau ist eine „Kreisbewegung“?* *b) abhängig von der Realisierung von Nr. 1*
..	..	..

Abb. 7 Mängel der ersten Programmversion und resultierende Systemanforderungen

Die in Abb. 7 genannten Anforderungen erheben keinen Anspruch auf Vollständigkeit, doch soll über die Beschäftigung mit diesen von den Schülern identifizierten Problemen die Erweiterung des Methodenbegriffs motiviert werden. Für die Erarbeitung der Taxometrie und der Anforderungsliste ist ein Zeitrahmen von zirka einer Unterrichtsstunde geplant. Dies sollte, da hier nur Konzeptpapier (während der Gruppenarbeit) und die Tafel (während der Plenumdiskussion und der Zusammenstellung der Resultate) als Medien eingesetzt werden, aufgrund der offensichtlichen Mängel der ersten Programmversion, ein praktikabler Zeitrahmen sein.

5.2.2. Erarbeitungsphase

Mit der zweiten Unterrichtsstunde beginnt die Erarbeitungsphase zum Problem Nr. 1 (parametrisierte Methode zur Definition Distanz bei einer Bewegung). Auch diese Phase wird wieder als Gruppenarbeitsphase begonnen und endet mit der Präsentation und Diskussion der Ergebnisse im Plenum. Neben den schon genannten Kompetenzaspekten tritt hier der Aspekt der fachlichen Kompetenz deutlicher als bisher in den Fordergrund. Je nach

fachlicher Kreativität bzw. fachlichem Vorwissen der Schüler werden die Resultate dieser Phase ausfallen. Dabei ist mit einer größeren Bandbreite der Resultate als in der Problematisierungsphase zu rechnen.

Als Arbeitsmaterial dient in dieser Phase der Programmcode der Klasse Kreis zusammen mit einem Arbeitsblatt (Abb. 8), mit dessen Hilfe die Methoden zur Realisierung der Bewegungen von Kreis-Objekten untersucht werden sollen. Ziel ist dabei, dass sich die Schüler der programmtechnischen Ursache der mangelnden Beweglichkeit der Kreis-Objekte klar bewusst werden und darauf basierend eigene Vorschläge zu deren Behebung soweit als möglich entwickeln und diskutieren. Sie agieren dabei in der Rolle von Programmiersprachen-Entwicklern, die dieses Manko auch behoben haben.

Auch diese Unterrichtsphase sollte in einer Unterrichtsstunde abgeschlossen werden, wobei genügend Zeit für die Diskussion von Problemlösungen bereitstehen sollte. Dazu sind die hinführenden Aufgaben des Arbeitsblatts zielgerichtet und so kleinschrittig formuliert, dass sie auch ohne fachliches Vorwissen lösbar sind. Unabdingbar vorausgesetzt wird dabei natürlich, dass ein grundlegendes Vorwissen und Verständnis für die symbolische Notation von Prozessbeschreibungen, etwa aus dem Physik- oder Mathematikunterricht vorhanden ist.

Ziel dieser Unterrichtsphase ist dabei explizit nicht, dass die Schüler selbsttätig die Idee der parametrisierten Methoden entwickeln. Sollte diese Idee entwickelt werden, weil sie eventuell einem Schüler bekannt ist, so wäre ihre Diskussion ein besonders gelungenes, auf das nachfolgende elegant überleitendes, aber nicht notwendiges Ende dieser Unterrichtsphase.

Folgende Ideen und Lösungsansätze zur Realisierung der „variablen“ Beweglichkeit von Kreis-Objekten sollten jedoch während dieser Phase entwickelt worden sein:

1. Formulierung einer „Parametrisierungs“-Idee

Die Schüler formulieren hierbei während der Diskussion im Plenum Ideen bzw. Anforderungen der Form: „Man muss dem Methodenaufruf die Distanz des Positionswechsels bei jeder Ausführung vorgeben können.“ In diesem Fall ist das Ziel der Unterrichtsphase voll erreicht, denn in der nachfolgenden Problemlösungsphase kann die tatsächliche Umsetzung dieser Anforderung in der Programmiersprache Java von einem soliden Vorverständnis aus gut motiviert werden.

2. Formulierung einer „Bypass“-Idee

Die Schüler formulieren die Lösungsstrategie, die bei einem Aufruf einer Bewegungsmethode zurückgelegte Distanz geeignet klein zu wählen, da dadurch eine „kontinuierliche“ Bewegung auf der Leinwand gesichert werden könne. Größere Distanzen müssen dabei durch geeignet oft wiederholte Aufrufe der Bewegungs-Methoden überbrückt werden. Auch dies stellt ein zufriedenstellendes Ergebnis dieser Phase dar, wobei hier dann noch die Nachteile des Verfahrens herausgearbeitet werden müssen. Dabei ist er „ineffizient“ häufige Aufruf der Methoden löschen() und zeichnen() bei der Ausführung der Bewegungs-Methoden den Schülern zu verdeutlichen.

Zu erwarten steht, dass man mit Ideen vom Typ 1. und 2. die Diskussion beendet und das Interesse der Schüler an der „in der Praxis“ vorgenommenen Problemlösung geweckt ist.

Arbeitsblatt Nr. ... Klasse: ... Datum: ... Name: ...

Thema: Methoden zur Bewegung von Kreis-Objekten

Durch die Anweisungen im Programm-Code der Methoden der Klasse Kreis wird ein Objekt dieser Klasse auf der Leinwand bewegt.

a) Untersuchen sie den Programmcode zu den angegebenen „Bewegungs“-Methoden der kreis-Klasse und beschreiben Sie in der nachfolgenden Tabelle, welche Art von Wertänderungen in den Attributen der Klasse Kreis zu diesen Bewegungen führen.

Attribut:	Wert des Attributs **durchmesser (d)**		Wert des Attributs **xPosition (xP)**		Wert des Attributs **yPosition (yP)**	
Methode:	Vor..	..nach Aufruf von ..	vor..	nach Aufruf von ..	vor..	nach Aufruf von ..
nachRechtsBewegen()	d_{vor}	d_{nach}=	xP_{vor}	xP_{nach}=	yP_{vor}	yP_{nach}=
kritische Zuweisung						
nachLinksBewegen()	d_{vor}	d_{nach}=	xP_{vor}	xP_{nach}=	yP_{vor}	yP_{nach}=
kritische Zuweisung						
nachObenBewegen()	d_{vor}	d_{nach}=	xP_{vor}	xP_{nach}=	yP_{vor}	yP_{nach}=
kritische Zuweisung						
nachUntenBewegen()	d_{vor}	d_{nach}=	xP_{vor}	xP_{nach}=	yP_{vor}	yP_{nach}=
kritische Zuweisung						

b) Woran liegt es, dass die durch die Methoden-Aufrufe bewirkte Bewegung des Kreis-Objekts „starr“ ist?

c) Schreiben sie eine Folge von Methodenaufrufen auf, mit der ein Kreisobjekt zuerst um 80 Pixel nach rechts, dann um 40 Pixel nach unten bewegt wird?

d) Geben sie Beispiele für zwei Nachteile an, die aus der „starren Modellierung“ der Bewegung für Kreis-Objekte resultieren?

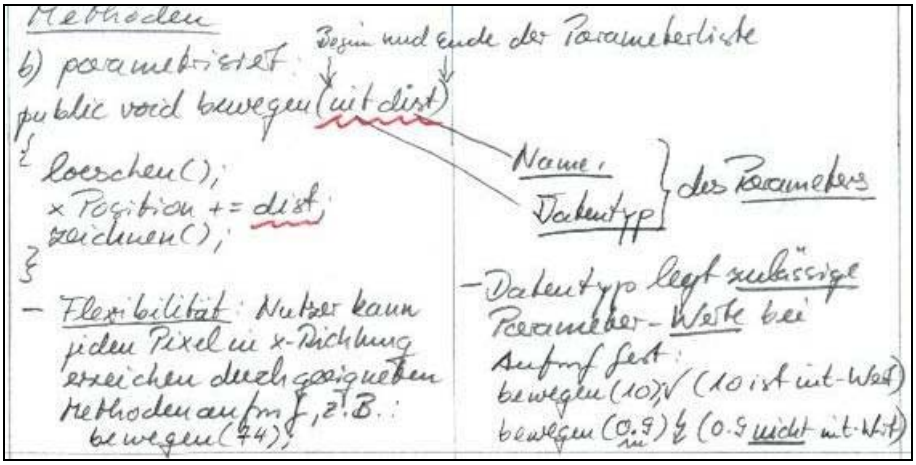
e) Wie kann man die Beweglichkeit der Kreis-Objekte „variabler“ gestalten? Überlegen sie „Spracherweiterungen“ und/ oder andere Methoden, mit denen „variabel“ bewegliche Kreis-Objekte erzeugt werden können!

Abb. 8 Arbeitsblatt zur Systemanforderung „Frei bewegliche Kreis-Objekte realisieren“

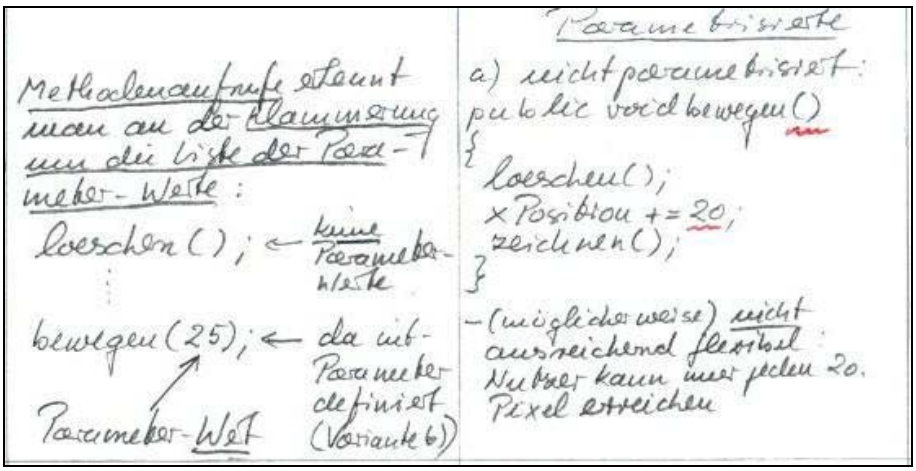
5.2.3. Lösungsphase

Neben der reinen Problemlösung steht während dieser letzten Unterrichtsstunde dieses Blocks auch die Sicherung des Verständnisses vom erweiterten Methodenbegriff im Mittelpunkt. Daher wird das Unterrichtsgeschehen bei der Präsentation und Analyse der Lösung stärker lehrergesteuert als in den vorangegangenen, eigentlich kreativen Phasen dieses Unterrichtsblocks. Ziel ist, dass die Problemlösung knapp und geeignet präzise anhand des übersichtlichen Codebeispiels der Methode nachRechtsBewegen() der Kreis-Klasse den Schülern verständlich wird.

Dazu werden die Schüler aufgefordert, die elementare und die parametrisierte Bewegungs-Methode, die ihnen parallel am Beamer präsentiert werden, zu vergleichen, die Differenzen im Code anzugeben und die von ihnen vermuteten Unterschiede bei der Ausführung der Methoden anzugeben. Den „Korrektheitstest“ für die Vermutungen der Schüler führt der Lehrer in der Entwicklungsumgebung BlueJ durch. Das Tafelbild wird parallel zu den einzelnen Schritten der Präsentation und Analyse der Lösung aufgebaut. Für diese Abschnitt der Unterrichtsstunde sollten bis zu 20 Minuten ausreichen. Das angestrebte Tafelbild verdeutlicht die zentralen Lerninhalte dieses Unterrichtsblocks und gibt eine Anleitung zur Codierung eigener parametrisierter Methoden. Das Tafelbild ist auf eine Tafelfläche von vier mal 1 m² ausgerichtet.



Die mittleren Abschnitte (siehe auch Bild unten) und der rechte Abschnitt des Tafelbildes werden zuerst angeschrieben



Der linke Abschnitt des Tafelbildes wird zuletzt angeschrieben

Anschließend erweitern die Schüler die anderen Bewegungs-Methoden der Klasse Kreis am Computer. Die entstehenden Programme sollen sie testen, indem sie ihre erweiterten Programme übersetzen und ausführen.

Die Hausaufgaben beschäftigen sich mit den analogen Erweiterungen einer elementaren Version der Klasse Dreieck, die den Schülern zur Verfügung gestellt wird. Durch die Bearbeitung dieser Hausaufgabe wird zumindest auf formalem Niveau eine Festigung des erweiterten Methodenbegriffs erzielt. Weiterführend soll als Hausaufgabe eine Konzeption für eine neue Methode kreisBewegung() ausgearbeitet werden, zu der auch freiwillig schon ein Implementierungsversuch vorgenommen werden kann.

Zu den drei Unterrichtsstunden zur Erweiterung des Klassenbegriffs wurden die nachfolgenden Verlaufsplanungen aufgestellt.

Phase/ Ziel/ Zeit (1. Std.)	**Unterrichts-gegenstand**	**geplantes Lehrerverhalten**	**erwartetes Schülerverhalten**	**Sozialform/ Medien**
Wiederholung 10‘ (10‘)	Hausaufgaben	L. führt Kontrolle durch und leitet Unterrichtsgespräch zur Präsentation der Resultate durch einzelne Schüler	Sch. tragen ihre Resultate vor und beantworten Rückfragen anderer Sch. oder des L.	UG
Einstieg/ Problemstellung 5‘ Phase der Schwierigkeiten 20‘ (35‘)	Klasse Kreis	L. initiiert kurz Meldungen der Sch. zum Thema „Mängel der 1. Version der Kreis-Klasse“ und systematisiert das Unterrichtsgeschehen durch Vorgabe von a) Arbeitsauftrag b) Arbeitsform (Gruppenarbeit) c) Zeitplan (20 min.) d) erwarteten Resultaten e) Weiteres Vorgehen (Besprechung im Plenum) f) Konkrete Gruppeneinteilung L. in zurückhaltender Moderatorenrolle L. setzt Impuls zur Beschäftigung mit dem zweiten Teil des Arbeitsauftrags (nach zirka 7 min) L. beobachtet Arbeitsfortschritte und Qualität der einzelnen Resultate, um Besprechung im Plenum effizient moderieren zu können	Sch. greifen „Mängel-Diskussion auf und folgen der Vorstellung des Arbeitsauftrags und finden sich in ihren Arbeitsgruppen zusammen Sch. bearbeiten die Arbeitsaufträge	UG Beamer Stift und Papier Gruppenarbeit
Auflösung 10’ (45‘)	Klasse Kreis	L. hebt Gruppenarbeitsphase auf, leitet Besprechung im Plenum, erstellt Stichwortliste an der Tafel	Sch. stellen Ihre Resultate vor und beantworten Nachfragen der Sch.	UG Tafel

Phase/ Ziel/ Zeit (2.Std.)	**Unterrichts-gegenstand**	**geplantes Lehrerverhalten**	**erwartetes Schülerverhalten**	**Sozialform/ Medien**
Einstieg 10’ (10‘)	Anforderungsliste für die Klasse Kreis („variable“ Beweg-lichkeit von Kreis-Objekten realisie-ren)	L. gibt Tabellenstruktur der zu erstellenden Anforderungsliste vor und fordert Sch. auf, konkrete Anforderungen zu ausgewählten Punkten zu formulieren (ggf. bei Formulierung lenkend eingreifen)	Sch. formulieren Anforderungen Sch. übernehmen Anforderungsliste in ihre Aufzeichnungen	UG Tafel Arbeitsheft
Problematisie-rungsphase 20‘ (30‘)	Anforderungsliste für die Klasse Kreis („variable“ Beweg-lichkeit von Kreis-Objekten realisie-ren)	L. strukturiert Unterrichtsgeschehen durch Vorgabe von g) Arbeitsauftrag (Arbeitsblatt) h) Arbeitsform (Gruppenarbeit) i) Zeitplan (20 min.) j) erwarteten Resultaten k) Weiteres Vorgehen (Besprechung im Plenum) l) Konkrete Gruppeneinteilung L. beobachtet in zurückhaltender Moderatorenrolle Arbeitsfortschritte der Sch. L. und geeignete Sch. geben ggf. Hilfen bei der Bearbeitung der Aufgabenteile a) –d), damit geeignet viel Zeit zur Bearbeitung von Aufgabenteil e) zur Verfügung steht (8 min). L. beendet Gruppenarbeitsphase.	 Sch. finden sich in ihren Arbeitsgruppen zusammen und bearbeiten das Arbeitsblatt Sch. haben Aufgabenteile a) –d) mindestens grundlegend bearbeitet und diskutieren Ansätze für mögliche „Spracherweiterungen“ zur Lösung des Problems Nr. 1 Sch. tragen Ergebnisse zu den Teilaufgaben a) – d) vor und führen ggf. Korrekturen in ihren Arbeitsblättern durch	UG Tafel Arbeitsheft Gruppenarbeit Beamer (Code der Bewegungs-Methoden der Klasse Kreis)
Konsolidierung/ Auflösung 10’ 5’ (45‘)	Anforderungsliste für die Klasse Kreis („variable“ Beweg-lichkeit von Kreis-Objekten realisie-ren)	L. leitet Lösung des Aufgabenblatts (Teilaufgaben a) – d)) am OH-Projektor (Ausfüllen nach Meldungen der Sch.) L. leitet Besprechung zu Lösungsansätzen der Sch. zu Aufgabenteil e) L. kündigt Lösung der „professionellen“ Programmiersprachen-Entwickler an, Vergleich mit den Ansätzen der Sch.	Sch. tragen Ergebnisse zu den Teilaufgaben a) – d) vor und führen ggf. Korrekturen in ihren Arbeitsblättern durch Sch. stellen Lösungsansätze zu Aufgabenteil e) vor und beantworten Nachfragen	UG OH Projektor Stift und Papier

Phase/ Ziel/ Zeit (3.Std.)	**Unterrichts-gegenstand**	**geplantes Lehrerverhalten**	**erwartetes Schülerverhalten**	**Sozialform/ Medien**
Einstieg 6’ (6‘)	Java-Code der zwei Implementierungen der Klasse Kreis	L. präsentiert am Beamer synoptische Darstellung der nicht-parametrisierten und der parametrisierten Bewegungs-Methode. L. fordert Sch. auf, Unterschiede zu benennen L. markiert Unterschiede am Beamer	Sch. benennen die Unterschiede der zwei Methoden	UG Beamer
Lösung 14’ (20’)	Java-Syntax und –Semantik von parametrisierten Methoden	L. fertigt Grundstruktur des Tafelbildes an und fordert Sch. auf, die programmtechnischen Konsequenzen der erweiterten Syntax zu benennen L. ergänzt Fachbegriffe (Datentyp und Name eines Parameters, Parameterliste)	Sch. erkennen das syntaktische Konstrukt, dass die „variable“ Übergabe von Parametern an eine Methode realisiert und erarbeiten sich dessen Semantik Sch. übernehmen Tafelbild in ihre Aufzeichnungen	UG Tafel/ Beamer Arbeitsheft, Stift und Papier
Sicherung 20’ (40‘) Hausaufgabe 5’ (45‘)	Java-Syntax und –Semantik von parametrisierten Methoden, Klasse Kreis	L. gibt Arbeitsauftrag für den Rest der Stunde an, nämlich die Erweiterung und erste Ablauf-Test für eine andere Bewegungsmethode der Klasse Kreis mit BlueJ und strukturiert den weiteren Ablauf durch Vorgabe von d) Arbeitsauftrag (Arbeitsblatt) e) Arbeitsform (2er-Gruppen) f) Zeitplan (15 min.) L. beendet diese Arbeitsphase L. fordert Sch. auf, ihre Implementierung am Beamer nachzustellen und zu erläutern L. stellt die Hausaufgabe und erläutert ggf. Nachfragen der Sch.	Sch. nehmen ihre neuen Arbeitsplätze ein und beginnen mit der Implementierung Sch. testen ihre Methodenerweiterung Sch. nehmen ihre gewöhnliche Sitzordnung ein Sch. schreiben wesentliche Implementierungsschritte eigenverantwortlich mit Sch. schreiben die Hausaufgabe in ihr Arbeitsheft	Rechner Zweiergruppen am Rechner Arbeitsheft Stift und Papier Arbeitsheft Stift und Papier Arbeitsheft Stift und Papier

6. Literatur

Die Fehler auf den vorangegangen Seiten gehen auf das Konto der Verfasser. Das, was richtig ist, haben sie im Seminar von Herrn Spolwig oder bei anderen Gelegenheiten aufgeschnappt.

[1] E. Hui, U. Bestmann: Informatik für Gymnasien (Lehr- und Arbeitsbuch), Frankfurt, 1986 (Diesterweg)

[2] H. Balzert: Informatik 1 + 2 (jeweils mit zugehörigem Lösungsband), München, 1982 (Hueber-Holzmann)

[3] J. B. Metzler[3]: Metzler-Informatik (mit zugehörigem Lehrerband), Stuttgart, 1990 (Metzler)

[4] S. Spolwig: Objektbasierte Programmierung im Anfangsunterricht, LOG IN 15/3, 43 – 49, Berlin (1995)

[5] R. Baumann: Didaktik der Informatik (2. Aufl.), Stuttgart, 1996 (Klett)

[6] D. J. Barnes, M. Kölling: Object first with Java (A practical introduction using BlueJ), Harlow, 2003 (Pearson Education Limited)

[3] Autorenteam