Ausarbeitung zu Hauptseminar

'Objektorientierte Programmierung im Anfangsunterricht'

unter der Leitung von

Siegfried Spolwig

im Wintersemster 2003/ 2004 an der Humbold-Universität zu Berlin

Einführung in die Objektorientierte Programmierung

Halbjahresentwurf für das 2. Halbjahr des 11. Jahrgangs

ausgearbeitet von

Annett Schönherr

Imma-Nr.: 165753

Clemens Dubberke

Imma-Nr.:149255

Inhalt

Hinweis:

Zur Kenntlichmachung der bearbeiteten Anteile steht ein @ für Annett Schönherr und ein © für Clemens Dubberke zu Beginn der entsprechenenden Abschnitte hinter dem jeweils obersten Titel. Ausnahme ist die Halbjahresentwurfsübersicht, hier steht die Kennung rechts unter der Tabelle.

Einleitung@

Im 2. Halbjahr der Klasse 11 ist unser Ziel, den SchülerInnen die objektorientierte Programmierung nahezubringen. Dazu wird als Einstieg ein „Ampel- Projekt“ durchgeführt, bei dem das Ziel ist, verschiedenfarbige Felder auf dem Bildschirm erscheinen zu lassen. Dieses wird nicht in Java programmiert, sondern mit Stiften und Mäusen realisiert, da diese Anwendung ohne Programmierkenntnisse auf einfachem Weg durch Vorgabe von Klassen erfolgen kann und der Umstieg auf die eigentliche Programmierung relativ leicht ist. Der Nachteil, dass vorerst ein Stift erzeugt werden muss, ist nicht zu verhindern, lässt sich jedoch im Zusammenhang mit der Unterrichtsstunde zum Vergleich der realen mit der abgebildeten Welt erklären.

Um anschließend ein „Mensch- ärgere- Dich- nicht!“- Spiel zu programmieren, sind die Voraussetzungen der grafischen Oberfläche mit Hilfe des Ampel- Projektes geschaffen. Java dient hierbei als angemessene Programmiersprache, da es als die wohl bedeutendste gehandelt werden kann, was im Zusammenhang mit seinem Einsatz in der Internet- Programmierung steht. Dieser Fakt bringt ihm auch einen hohen Stellenwert bei der Schülerschaft ein.

1. Bedingungsfeldanalyse@

1.1 Institutionelle Voraussetzungen

In den Klassenräumen steht jedem/r SchülerIn ein vernetzter Arbeitsplatz zur Verfügung. Als Werkzeuge sind UMLed- Designer, Turbo- Pascal 7, Delphi, Stifte und Mäuse, Java mit grafischer Oberfläche, BlueJ, Fujaba und Python vorhanden.

Der Stundenplan sieht für die 11. Klasse 3x 45 Minuten Informatikunterricht pro Woche vor, wobei im zweiten Halbjahr mit dem Programmieren begonnen werden sollte.

1.2 Allgemeine Voraussetzungen der Schülerinnen und Schüler

Im 1. Halbjahr wurden behandelt: Einführung in die Informatik, Grundlagen der Rechnerorganisation, der Netze und Geschichte der Informatik, Datenbanken und Datenschutz.

Im Kurs sind knapp 50% Mädchen, vereinzelt und unsystematisch sind Programmierkenntnisse vorhanden. Fast alle SchülerInnen verfügen über einen heimischen PC mit Internetzugang.

2. Stoffverteilungsplan

2.1 Ziele des Halbjahres©

In diesem Halbjahr werden den Schülerinnen und Schülern die wesentlichen Konzepte der Objektorientierten Programmierung, wie Klasse, Objekt, Vererbung und Methoden vermittelt. Durch das Verständnis des Paradigmas der Objektorientierung erlangen die Schülerinnen und Schüler die Beherrschung der Grundlagen von Java als Beispielprogrammiersprache. Die Schülerinnen und Schüler können am Ende des Halbjahres wenigstens triviale Programme lesen, verstehen und anwenden. Sie haben dadurch die Grundlage erworben, weitere Programmiersprachen leichter erlernen und verstehen zu können. Zu den oberen Lernzielen gehören weiterhin das

Verständnis von Dateistrukturen sowie die Kenntnis der elementaren Such- und Sortierverfahren bzw. –algorithmen.

Als soziale Lernziele sind das Erlernen projektbezogenen Arbeitens allein und in der Gruppe sowie das Erwerben von praktischen Tätigkeiten zu nennen.

2.2 Begründung der Halbjahresziele©

Der Einstieg in die Programmierung mit Objektorientierter Programmierung hat den Vorteil, dass es bei den Schülerinnen und Schülern nicht zu einer „Verbildung“ kommt. Sie gehen nicht erst den imperativen Weg der Programmierung und werden dann gezwungen mühsam umzulernen, sondern sie beginnen von vornherein mit der Objektorientierung und haben somit einen Anschluss an den aktuellen Stand der Technik.

Durch die objektorientierte Herangehensweise lernen die Schülerinnen und Schüler, ein zu entwerfendes System in seine Einzelteile zu zerlegen. Jedes Einzelteil kann (falls es noch zu komplex ist) in weitere Einzelteile zerlegt werden. Aus der Komplexität des Gesamten wird so eine einfach überschau- und beherrschbare Struktur (u.a. Stichwort UML). Zur Lösung können alle Komponenten zusammengesetzt werden. Durch die überwiegende Unabhängigkeit der Einzelteile vom Gesamten sind die Schülerinnen und Schüler in der Lage sich allein oder in Kleingruppen mit den Komponenten zu beschäftigen und ohne große Kenntnis der anderen Einzelteile mit diesen ein lauffähiges System herzustellen. Dabei lernen die Schülerinnen und Schüler, sich in eine Gruppe einzugliedern genauso, wie sich im bzw. vor dem Team zu behaupten, eigene Lösungen zu erklären und für die eigene Arbeit sowie das Weiter-/ Mitkommen der Komilitonen Verantwortung zu übernehmen. Die Wichtigkeit des Erlernens solcher Kompetenzen reicht weit über den Informatikunterricht hinaus, in jeden Bereich unserer Gesellschaft.

2.3 Sach- und Zielanalyse©

Im folgenden werden nur die Inhalte beschrieben, auf welche in den ausgearbeiteten Unterrichtsstundenentwürfen Bezug genommen wird.

2.3.1 Klasse

Unter einer Klasse ist ein selbstdefinierter Datentyp zu verstehen, der zur Modellierung neuer Strukturen verwendet wird. Eine Klasse ist eine Sammlung von Variablen und Methoden. Klassen sind in eigenständigen Programmdateien oder in inneren Klassen definiert. Klassen bilden die oberste Struktureinheit in einem objektorienterten Programm.

2.3.2 Objekt

Allgemein ist unter einem Objekt eine Größe zu verstehen, die durch einen bezeichner benannt oder in Form von Daten bei der Programmierung auftreten kann.

Speziell sind unter Objekten Einheiten zu verstehen, die innere Einheiten (Variablen) besitzen und diese durch bestimmte Nachrichten (Methoden) verändern bzw. auf diese Nachrichten reagieren können. Der Informationsaustausch erfolgt durch Senden und Empfangen dieser Nachrichten. Ein Objekt kann sich (wie in der realen Welt) aus anderen Objekten zusammensetzten, die ebenfalls aus anderen Objekten zusammengesetzt sind.

2.3.3 Methode

Durch Methoden wird ausführbarer Code unter einem Namen zusammengefasst. Dieser Code kann unter Verwendung von Parametern formuliert sein, denen beim Aufrufen der Methode Werte übergeben werden. Neben Variablen gehören Methoden in der Regel zu festen Bestandteilen von Klassen.

2.3.4 Konstruktoren

Hierbei handelt es sich um eine Art Methode. Konstruktoren sind jedoch keine Methoden im eigentlichen Sinn, da sie nicht wie Klassen- oder Instanzmethoden explizit aufgerufen werden. Außerdem enthalten sie keinen Rükgabewert.

2.3.5 Variable

Eine Variable ist ein Speicherplatz, auf den nur mit einem eindeutigen Schlüssel (z. B. Variablenname = eindeutige Adresse) zugegriffen werden kann. Variablen können bestimmte Inhalte zugeordnet und später wieder ausgelesen werden. Dazu müssen sie deklariert und initaliesiert werden.

2.3.6 UML-Diagramme

Unter dem Begriff UML-Diagramme werden die folgenden drei Diagramme zusammengefasst: 1. Das *Use-Case-Diagramm* mit dessen Hilfe sich Anwendungsfälle in einem Bild skizzieren lassen. 2. Das *Sequenz-Diagramm* mit dem sich Abläufe innerhalb von Klassen darstellen lassen, und 3. Das Verteilungsdiagramm, mit dem sich darstellen lässt, welche Komponente auf welchem Rechner (z. B. Web-Server-Anwendung) angesiedelt ist. In einer visuellen Entwicklungsumgebung werden Klassen mit UML-Diagrammen entworfen und aus diesen Diagrammen kann Java-Code generiert werden. So ist es möglich erste kleine Programme zu schreiben, ohne die Programmiersprache beherrschen zu müssen.

2.3.7 Assoziation

Assoziation beschreibt die Beziehung zweier Objekte so, dass ein Objekt das andere und dessen Methoden kennt und diese benutzen kann. Die Realisierung findet durch Referenzierung der Objekte statt. Somit ist die Assoziation gerichtet, denn die gegenseitige Kenntnis zweier Objekte voneinander ist erst mit der Referenzierung beider Objekte auf das jeweils andere gegeben.

2.3.8 Aggregation

Aggregation beschreibt die Beziehung zwischen zwei Objekten so, dass eines der Objekte als Teil des anderen, umgebenden Objektes aufgefasst werden kann ohne von diesem zu erben. Das umgebende Objekt kennt sein Teilobjekt und kann dieses benutzen. Wie bei der Assoziation stehen die beiden Objekte in einer Bezieung.

Diese ist jedoch wesentlich fester. So ist das umgebene (Teil-) Objekt abhängig vom umgebenden Objekt, es wird mit ihm erzeugt und gelöscht. Diese enge Verbindung wird durch eine statische Referenz (wie der Name schon sagt: fest, nicht mehr veränderbar) erzeugt. Sie verweist auf das Objekt, das ihr bei der Initialisierung zugewiesen wurde.

2.3.9 Generalisierung

Generalisierung ist die Erstellung einer Superklasse von Klassen. Das bedeutet: Wenn mit Hilfe des Computers z. B. verschiedene Tiere simuliert werden sollen, wird jede Tierart durch eine Klasse realisiert. Durch die Instantiierung werden Objekte erzeugt, die die einzelnen Tiere darstellen. Es werden z. B. die Klassen *Katze* und *Maus* entworfen, die die entsprechenden Tierarten realisieren sollen. Nun haben diese beiden Tierarten die Gemeinsamkeit, dass es sich hierbei jeweils um Säugetiere handelt. Es ist also möglich zu generalisieren und eine Superklasse *Säugetiere* zu entwerfen. Objekte mit gleichen Eigenschaften werden in einer Superklasse zusammengefasst, generalisiert. Die Umkehrung der Generalisierung wird Spezialisierung oder Erweiterung genannt. Hierbei ist die bestehende Klasse *Maus* Superklasse. Zu dieser Superklasse werden die Subklassen *Feldmaus* , *Spitzmaus* und *Wüstenrennmaus* entworfen. Alle drei Subklassen haben die Gemeinsamkeit Mausarten zu realisieren. Die Spezialisierung/ Erweiterung beruht auf dem gleichen Konzept wie die Generalisierung. Beide unterscheiden sich nur in der Richtung (von unten nach oben = Generalisierung; von oben nach unten = Spezialisierung). Aus diesem Grund erfolgt die Ansiedlung der Spezialisierung in der gleichen Kategorie.

2.3.10 Vererbung

Aufgrund ihrer Beziehung zueinander können Subklassen als Ausprägung ihrer Superklassen betrachtet werden. Das bedeutet, das die Superklasse ihre Eigenschaften an die Subklassen vererbt. Durch Vererbung können die gleichen Eigenschaften verschiedener Klassen gleichzeitig und einmalig durch die Definition in der Superklasse modelliert werden. Nach obigem Beispiel sind die Eigenschaften der Lebendgeburt (statt Eiablage) und der Säugung der Jungtiere in der Superklasse

Säugetiere definiert und werden an die Subklassen *Katze* und *Maus* und natürlich auch an deren Subklassen *Feldmaus* , *Spitzmaus* und *Wüstenrennmaus* vererbt.

2.3.11 Kontrollstrukturen@

if- Anweisung: Die wohl grundlegendste Entscheidungsanweisung vieler Programmiersprachen ist die so genannte **if- else**- Anweisung. Während der Programmausführung wird zunächst der Ausdruck ausgewertet, dessen Ergebnis **boolean** sein muss. Ist das Ergebnis **true**, so wird die unmittelbar nachfolgende Anweisung ausgeführt, ist es **false**, so kommt die Anweisung **else** zur Ausführung. Danach wird mit der nächsten Anweisung fortgefahren. Es wird jedoch immer genau eine der beiden Anweisungen ausgeführt- die Anweisung nach **if** und die Anweisung nach **else** können also in einem Durchlauf der **if- else**- Anweisung niemals beide zur Ausführung kommen. Will man keine Anweisung durchführen, wenn der Ausdruck das Ergebnis **false** liefert, so kann man den **else-** Teil auch komplett weglassen.

for- Anweisung: Zunächst werden im Teil <INITIALISIERUNG> eine oder mehrere (typgleiche) Variablen vereinbart und initialisiert und daraufhin der Ausdruck, dessen Ergebnistyp wiederum vom Typ **boolean** sein muss, ausgewertet. Ist sein Wert **true,** so werden die Anweisung bzw. der Anweisungsblock ausgeführt und danach noch zusätzlich die Anweisungen in der Update- Liste ausgeführt. Dies wird solange wiederholt, bis der Ausdruck den Wert **false** liefert.

while- Anweisung: Die **while-** Schleife wird als abweisend bezeichnet, weil hier, bevor irgendwelche Anweisungen zur Ausführung kommen, zunächst ein logischer Ausdruck geprüft wird. Solange dieser den Wert true liefert, wird die Anweisung bzw. der Anweisungsblock ausgeführt und der Ausdruck erneut berechnet.

do- Anweisung: Die **do-** Schleife ist nicht- abweisend, da hier die Anweisungen auf jeden Fall zur Ausführung kommen, bevor ein logischer Ausdruck geprüft wird. Solange dieser dann den Wert **true** liefert, wird das Ganze wiederholt. Es kommt also auf jeden Fall zu mindestens einer Ausführung der Anweisung bzw. des Anweisungsblockes.

Es empfiehlt sich, die Anweisungen oder Anweisungsblöcke *immer* in Klammern zu setzen, um Verwechslungen vorzubeugen.

2.4 Methodisches Vorgehen@

Da fast alle SchülerInnen noch keine Erfahrungen mit Programmierung haben, wird der Einstieg in diese Thematik über vereinfachte Beispiele aus dem Leben der SchülerInnen vollzogen. Dabei sollen sich Lehrervortrag, Unterrichtsgespräch und selbständiges Arbeiten der SchülerInnen allein oder in Gruppen abwechseln, um einen variantenreichen Unterricht zu gestalten. Beim Lehrervortrag erfolgt ein deduktives Vorgehen, da dies eine schnelle Vermittlung der Themen sichert. Die Einzel- und Gruppenarbeit gewährleistet eine induktive Lehrweise, da die SchülerInnen selbständig Probleme erkennen und lösen sollen. Hierbei steht die Anwendung und Festigung des theoretisch Erlernten im Vordergrund. Bei den Projektentwicklungen werden den SchülerInnen kleine Programmteile vorgegeben, da die zur Verfügung stehende Zeit sonst nicht ausreichen würde. Während der gesamten Projektzeit steht die Lehrkraft den SchülerInnen helfend zur Seite. Es können Einzel- oder Gruppenbesprechungen vereinbart werden. Hilfestellungen über E- Mail gehören ebenfalls zum Betreuungsumfang.

2.5 Gliederung des Stoffverteilungsplans@

Im Folgenden wird die Unterrichtsverteilung über das Halbjahr in tabellarischer Form dargestellt. Hier wird davon ausgegangen, dass am Dienstag eine Doppelstunde und freitags zu fortgeschrittener Stunde eine Einzelstunde des Informatikunterrichts stattfindet. Im Entwurf wurde versucht, die meist niedrigere Konzentrationsfähigkeit der SchülerInnen am Freitag zu berücksichtigen.

Vorerst folgt eine kurze Erläuterung der ersten Stunden, die als Einführung in die Programmierung gestaltet werden sollen. Die erste Stunde dient der Mitteilung über die Zielstellung des Semesters, indem die zu bearbeitenden Projekte vorgestellt werden. Die SchülerInnen sollen über ein Ampelprogramm, das gemeinsam bearbeitet wird zur Programmierung eines „Mensch- ärgere- dich- nicht!“- Spiels

befähigt werden. Dazu werden in der ersten und zweiten Stunde Einführungen zu Programmiersprachen und Compilern gegeben. Die SchülerInnen werden aufgefordert, einen möglichst detaillierten Tagesablauf zu schreiben, der von anderen SchülerInnen problemlos nachahmbar sein soll. Über die hierbei auftretenden Probleme kann der Algorithmus- Begriff eingeführt und ein erster Hinweis auf die Schwierigkeit der Umsetzung der realen Welt in einen Code gegeben werden. Beim Vorstellen von Compilern soll der Bezug zur Rechnerorganisation im vergangenen Schulhalbjahr hergestellt werden, indem man darauf aufmerksam macht, dass das auf dem Bildschirm ablaufende Programm in anderer Form im Rechner abläuft. Zur Einführung der grundlegenden Begriffe der objektorientierten Programmierung werden die SchülerInnen aufgefordert, anzugeben, welche Bausteine nötig sind, um eine Ampel darzustellen. Die Antworten werden auf der Tafel notiert und im Anschluss mit einem Beispielprogrammcode, der mittels eines Beamers an die Wand projiziert wird, verglichen. Damit kann man zusammenfassen, welche Elemente nötig sind und die Einteilung in Klassen, Objekte usw. vornehmen. Daran schließt die Darstellung in UML- Klassendiagrammen an, es folgen Beziehungen zwischen Klassen mit Konzentration auf Vererbung, Assoziationen und Aggregationen. Nach der Übersetzung der entstandenen UML- Codes in Stifte und Mäuse wird zur Behandlung der Teilthematik „Kontrollstrukturen“ übergegangen.

Dem folgt in der achten Woche direkt der Einstieg in das Spiel- Projekt, bei dem die SchülerInnen selbständig Gruppen bilden, ein Regelwerk aufstellen und beginnen, ihr Spiel in UML- Diagrammen darzustellen und zu programmieren. Ihnen wird in der zweiten Woche dieser Projektarbeit das Programm für den Würfel vorgegeben, wobei die Funktionsweise besprochen werden soll. Für die 11. Bis 14. Woche werden den SchülerInnen Meilensteine, die es wöchentlich bis zum Freitag zu erreichen gilt, vorgegeben. Das erste Ziel ist es, das Spielfeld grafisch dargestellt zu haben, dem folgt der Farbwechsel der Felder (hier kommt das Würfel- Programm zum Einsatz) und die beiden folgenden Freitage dienen der Zwischenpräsentation des Spielfeldes mit Farbwechsel der Felder nach den von den SchülerInnen aufgestellten Regeln. Betreut wird das Projekt ständig durch die Lehrkraft, welche Vorgaben, Hilfen und E-Mail- Betreuung anbietet.

Thema	Inhalte	Kommentar	Woche	h
Einführung	Halbjahresverlauf, Unterrichtsziele, Organisatorisches Grundlagen der Programmierung	Projektvorstellung Was ist Programmierung? Programmiersprachen und Compiler Algorithmusbegriff	1	1,2 3
OOA **OOD**	Ampelprojekt	Problemanalyse der Ampelrealisierung, Begriffseinführung: Klasse, Objekt, Methode, Variable, UML- Diagramme, Beziehungen zwischen Klassen: Vererbung, Assoziation, Aggregation, Kardinalitäten -> Bezug zu DB	2 - 4	1 - 9
OOA, OOD	Wiederholung, Test, UML- Code	Fragen mittels Bankrutschen von den SchülerInnen erklären lassen	5	1 2 3
OOP	SuM- Ampel-Programm erstellen, Einführung in Anweisungen und Ablaufsteuerung	Anschließende Kontrolle mit UML- Diagramm	6	1,2 3
	Kontrollstrukturen vorstellen	Entscheidungs- und Wiederholungsanweisungen	7	1,2,3

	Üben der Kontrollstrukturen	(if, while, do, for)	8	1,2,3
Einführung	Projektbeginn	Gruppenbildung (6 x 4); Projektvorstellung (L)	9	1
OOA	Spielregeln	Ausarbeitung einfacher Spielregeln praktisches Brettspiel mitbringen und durchführen	9	2 3
OOD	UML-Diagramm		10	1 - 3
OOP	Projektbearbeitung	Gruppenarbeit Brett m. Feldern (graf.) Gruppenarbeit Würfel (L), Farbwechsel (MS) Gruppenarbeit Farbwechsel n. Regeln (MS)	11 12 13,14	1,2 3 1,2 3 1,2 3
	Klausur		15	1,2
Wiederholung	Präsentation	Gruppen stellen ihre Ergebnisse beim gemeinsamen Spielen vor	15	3

@,©

L – LehrerIn MS – Meilenstein

3. Unterrichtsplanungen

3.1 Planung der 13. Unterrichtsstunde©

3.1.1 Lehreinheit

Einführung in die Programmierung: Objektorientierte Analyse, Objektorientiertes Design

3.1.2 Unterrichtsthema

13. Stunde → Wiederholung

3.1.3 Einordnung der Stunde im Halbjahr

Diese Unterrichtsstunde wird zum Beenden des ersten Blocks, Einführung in die Objektorientierte Programmierung ohne Programmierung, in die Unterrichtszeit des Halbjahres eingeordnet. Bevor der nächste Schritt, die OOP gegangen wird, mit der das erste Projekt (Ampel) abgeschlossen wird, ist diese Stunde zur Wiederholung und Festigung gedacht. Gleichzeitig bereiten sich die Schülerinnen und Schüler intensiv auf die direkt folgende Stunde vor, in der ein Test geschrieben wird.

3.1.4 Voraussetzungen der Schülerinnen und Schüler

Die Stunde schließt direkt an einen 9-stündigen OOA/ OOD-Block an, in dem die Schülerinnen und Schüler von der Problemanalyse ihres ersten Projektes zum Design desselben mit UML-Diagrammen gelangen. In dieser Zeit haben die Schülerinnen und Schüler grundlegende Begriffe der OOP, wie z. B. Objekte, Klassen, Methoden, Variablen und Konstruktoren kennengelernt und haben bereits mit diesen gearbeitet, d. h. sie können Beziehungen analysieren, selbständig aufstellen usw. Desweiteren kennen sie Generalisierung, Vererbung und Assoziationen bzw. Aggregationen.

3.1.5 Lernziele

Die Schülerinnen und Schüler sind sicher im Umgang mit Klassen, Objekten, Methoden und Variablen. Sie können ihren Mitschülerinnen und Mitschülern eine Definition der Begriffe nennen und diese erklären.

Die Schülerinnen und Schüler sind sicher im Erkennen von Beziehungen zwischen Klassen in UML-Diagrammen.

Alle Schülerinnen und Schüler wissen, wie sie einen Sachverhalt/ ein Problem zerlegen und können diesen/ dieses mit Hilfe von Klassen und Methoden im UML-Diagramm zu einer Lösung zusammensetzen. Dabei müssen sie selbständig arbeiten und sich in der Gruppe integrieren können.

Die Schülerinnen und Schüler üben das gegenseitige Zuhören und Erklären.

3.1.6 Sachanalyse

siehe Stoffverteilungsplan, Sach- und Zielanalyse (2.3)

3.1.7 Methodik

Zur Begrüßung der Schülerinnen und Schüler und zur Vorstellung des Stundenthemas wird die Unterrichtsstunde mit einem kurzen Lehrervortrag begonnen. Hierbei erhalten die Schülerinnen einen Überblick über das Soll ihres Wissensstandes und einen Ausblick auf die nächsten Stunden (Test, Beginn der Programmierung in Java usw.). Im Anschluss haben die Schülerinnen und Schüler in einer Art Unterrichtsgespräch die Möglichkeit, Fragen zu stellen, die vorrangig von ihren Mitschülerinnen und Mitschülern zu beantworten sind. Da die Hausaufgabe zu dieser Stunde die Vorbereitung von mindestens zwei Fragen zum bisherigen Unterrichtsinhalt war, kann das Unterrichtsgespräch zur knappen Hälfte der Unterrichtsstunde beendet werden. Es folgt ein Block von Übungsaufgaben, die erst in Einzelstillarbeit und dann in der Gruppe gelöst werden können. Lestungsstärkere Schülerinnen und Schüler unterstützen den Lehrer bei der Hilfestellung schwächer Schüler. Dies hat den Vorteil, dass alle eventuell auftretenden Fragen/ Hilferufe beantwortet werden können und die Lernergruppe gestärkt wird. Im letzten Drittel der

Stunde stellen sich die Schülerinnen und Schüler im Rahmen eines Gruppenspiels untereinander Aufgaben, dafür erhalten sie Punkte.

3.1.8 Geräte/ Medien

Verwendete Materialien/ Medien werden sein: Whiteboard/ Flipchart und Filzstift und Arbeitsblatt und Stift für die Stillarbeit der Schülerinnen und Schüler.

3.1.9 Verlaufsplanung

Zeit	Unterrichtsablauf	SF	didaktischer Kommentar
5 Min.	1. Begrüßung und Bekanntgabe der Lehreinheits- und Unterrichtsthemen	LV	• Einordnung in den Schulhalbjahresverlauf
15 Min.	2. Fragen und Wiederholung zur Thematik der vergangenen Stunden	UG	• Wiederholen und Festigen des Gelernten, • durch UG werden alle S. mit einbezogen
12 Min.	3. Übungsblock: a) S. lösen Aufgaben in Stillarbeit b) S. lösen Aufgaben in Kleingruppen c) Vorstellen/ Erklären der Lösungen an der Tafel	SA GA UG	• Stillarbeit bringt Ruhe zur Konzentration nach dem UG
10 Min.	4. Spielerisches Üben und Wiederholen im Gruppenspiel, indem sich die S. (einer Gruppe) Anweisungen ausdenken und S. (einer anderen Gruppe) diese lösen	UG, GA	• Spielerisches Üben und Wiederholen mit Bonussystem unterstützt Eigeninitiative, Antrieb und Nachdenken
3 Min.	5. Vorbereitung zum Schreiben des Tests 6. Verabschiedung in die Pause	LV	• S. stellen sich auf die bevorstehende Lernzielkontrolle ein

S – SchülerInnen, SF – Sozialform, UG – Unterrichtsgespräch,
LV – Lehrervortrag, SA – SchülerInnenarbeit

3.1.10 Aufgabenblatt für die Stillarbeit

1. Sie haben folgende vier Klassen zur Verfügung: *Auto*, *Motor, Fahrzeug* und *Fahrrad*. Bitte erstellen Sie ein UML-Klassendiagramm mit zwei Eigenschaften pro Klasse!

2. Vervollständigen Sie bitte den nachvollgenden Lückentext mit Angaben, die sich auf die Klassen Ihres UML-Diagramms beziehen!

a) Die Klasse ..... ist Superklasse der Klasse .....

b) Die Klasse ..... ist Subklasse der Klasse Auto.

c) Die Klasse ..... erbt von der Klasse ..... die Eigenschaft(en) .....

3. Spezialisieren Sie bitte eine Klasse Ihrer Wahl im UML-Diagramm!

4. Was verstehen Sie unter dem Begriff Generalisierung?

5. Benennen Sie bitte den/ die Unterschied(e) zwischen Aggregation und Assoziation, falls es diese(n) gibt!

3.2 Planung der 21. Unterrichtsstunde@

3.2.1 Lehreinheit

Objektorientierte Programmierung

3.2.2 Unterrichtsthema

21. Stunde → Kontrollstrukturen

3.2.3 Einordnung der Stunde im Halbjahr

Diese Unterrichtsstunde wird nach der knappen Hälfte der Unterrichtszeit des Halbjahres durchgeführt. Sie findet zeitlich nach dem Ampel- Projekt und noch vor der Durchführung des Spiel- Projekts statt. Da es bei erst genanntem Projekt nur darum ging, eine Ampel grafisch darzustellen, ohne dass die Lampen ihre Farbe nacheinander wechseln, genügt es, erst jetzt Kontrollstrukturen benutzen zu lernen.

3.2.4 Voraussetzungen der SchülerInnen

Die SchülerInnen kennen grundlegende Begriffe der OOP, wie z. B. Objekte, Klassen, Methoden, Variablen und Konstruktoren und können mit diesen arbeiten, d. h. sie können Beziehungen analysieren, selbständig aufstellen usw. Desweiteren sind sie mit Generalisierung, Vererbung und der Struktur von Blöcken vertraut.

Die Stunde schließt direkt an eine theoretische Einführung von Kontrollstrukturen an, bei der eine kurze Vorstellung der Thematik erfolgte. Es wird der im ersten Drittel des Halbjahres erlernte Umgang mit Deklarations- und Ausdrucksanweisungen aufgegriffen.

3.2.5 Lernziele

Die SchülerInnen kennen mindestens je eine Entscheidungs- und eine Wiederholungsanweisung und können diese anwenden, indem sie gestellte Aufgaben in kurzen Programmanweisungen bearbeiten.

Es soll hierbei vorerst genügen, mit der if- Anweisung und der while- Schleife arbeiten zu können. In selbsttätiger Arbeit werden sich die SchülerInnen außerschulisch mit while- und for- Schleifen vertraut machen, wobei sie das im Unterricht Gelernte in eben solche Anweisungen umwandeln sollen.

3.2.6 Sachanalyse

siehe Stoffverteilungsplan

3.2.7 Methodik

Zur Begrüßung der SchülerInnen und zur Wiederholung der in der vergangenen Stunde behandelten Themen wird mit einem Lehrervortrag begonnen, der sich in einem Unterrichtsgespräch fortsetzen soll. Dieses wird fragender und selbstentwickelnder Art sein, was den Vorteil hat, dass die SchülerInnen zum Mitdenken angeregt werden.

Die Definition der Anweisungen erfolgt deduktiv, da hier eine reine Wissensvermittlung stattfinden soll, auf die induktiv - bei der folgenden Entwicklung des Programmes - aufgebaut wird. Ebenso wird die folgende Aufgabe induktiv durchgeführt, da die SchülerInnen selbständig den Umgang mit Kontrollstrukturen üben sollen, wobei sie ggf. auf Probleme stoßen und versuchen sollen, diese selbst zu lösen. Hier steht die Lehrkraft helfend zur Seite.

3.2.8 Geräte/ Medien

Verwendete Materialien/ Medien werden sein: die Tafel und Kreide und der Rechner als Arbeitsplatz der SchülerInnen.

3.2.9 Verlaufsplanung

Zeit	Unterrichtsablauf	SF	didaktischer Kommentar
2 Min.	1. Begrüßung und Bekanntgabe der Lehreinheits- und Unterrichtsthemen	UG	
10 Min.	2. Fragen und Wiederholung zur Thematik der vergangenen Stunde	UG	• Auffrischen des Gelernten
20 Min.	3. Syntax und Funktionsweise von Entscheidungsanweisungen 4. Übung zur Entscheidungsanweisung, indem die Aufgabenstellung in Java-Syntax an die Tafel gebracht wird	LV UG	• gemeinsames Erarbeiten einer Anweisung, bezieht alle S. ein; der Unterricht wird aufgelockert
	5. Syntax und Funktionsweise von Wiederholungsanweisungen 6. Übung zur Wiederholungsanweisung, indem die Aufgabenstellung in Java-Syntax an die Tafel gebracht wird	LV UG	
10 Min.	7. Selbstständiges Üben am Rechner, indem sich die S. Anweisungen ausdenken und versuchen, diese zu realisieren	SA	• induktives Lernen fördert die Selbstständigkeit und unterstützt Eigeninitiative, Antrieb und Nachdenken
3 Min.	8. Hausaufgabenstellung (die Ergebnisse der Übungsphase sind Grundlage der folgenden Stunden)	LV	• S. sollen sich weiterführend mit dem Thema beschäftigen, um im Umgang damit sicherer zu werden
	9. Verabschiedung, Vorschau auf das weitere Unterrichtsvorgehen		

S – SchülerInnen, SF – Sozialform, UG – Unterrichtsgespräch,
LV – Lehrervortrag, SA – SchülerInnenarbeit

Fortgeschrittene Schüler können bei der Einzelarbeit die if- Anweisung durch einen else- Teil erweitern und allgemein komplexere Anweisungen programmieren.

3.2.10 Aufgabenstellung für die Tafelarbeit:

Für die zu erzeugende if- Anweisung sei gegeben:

```
int x = IOTools.readInteger( ) ;
```

Wenn x = = 0 ist , soll auf dem Bildschirm erscheinen: „ x ist gleich 0.“,

Zusatz: anderenfalls soll auf dem Bildschirm „ x ist ungleich 0.“ stehen.

Lösung:

```
int x = IOTools.readInteger ( ) ;
if (x = = 0)    {
        System.out.println („x ist gleich 0.“) ;    }
else     {
        System.out.println („x ist ungleich 0.“)        }
```

Die Aufgabenstellung für die while- Schleife lautet wie folgt:

gegeben sei int i = 0;

solange i kleiner als 10 ist, soll auf dem Bildschirm der Wert von i erscheinen, danach wird i inkrementiert.

Lösung:

```
int i = 0;
while (i < 10)          {
System.out.println (i);
        i++;                    }
```

Literatur

Dobberkat E.-E., Dißmann S.: Einführung in die objektorientierte Programmierung mit Java. 2., überarbeitete Auflage, Oldenbourg, München 2002

Ratz, D., Scheffler, J., Seese, D.: Grundkurs Programmieren in Java, Bd.1: Der Einstieg in die Programmierung und Objektorientierung, Hanser Verlag, München, 2001

Goll, J., Weiß, C., Müller, F.: Java als erste Programmiersprache. 3., durchgesehene und erweiterte Auflage, B. G. Teubner Stuttgart/ Leipzig/ Wiesbaden 2001

Echtle, K., Goedicke, M.: Lehrbuch der Programmierung mit Java, dpunkt Verlag, Heidelberg 2000

Baumann, R.: Didaktik der Informatik. 2. vollständig neu bearbeitet Auflage, Klett, Leipzig/ Stuttgart/ München/ Düsseldorf 1996.

Meyer, H.: Leitfaden zur Unterrichtsvorbereitung. 12. Auflage, Cornelsen Scriptor, Frankfurt am Main 1993

http://golem14.informatik.hu-berlin.de/spolwig/hsem_02/dokumente/jannasch2.pdf