Vorgelegt an der Humboldt-Universität zu Berlin
Institut für Informatik

14. November 2006

= 2.Unterrichtspraktikum =

der Informatik »im Block«
Zeitraum: 01.09.2006 bis 30.09.2006

Lehrende:	Christine Janischek
Betreuende Lehrkraft:	Herr Lüdtke
Fach:	Informatik
Schule:	Heinrich-Hertz-Oberschule
Strasse:	Rigaer Straße 81-82
Ort:	10247 Berlin

Thema der Unterrichtsreihe:
Programmieren mit Java und Software-Entwicklung

Inhaltsverzeichnis

1 Die Schule

1.1 Auswahl der Schule

Im ersten Praktikum (Wirtschaft) war der Einfluss auf die Wahl der Schule gering. Wir konnten aus einer Liste von Schulen auswählen und Wünsche äußern. Diese wurden dann berücksichtigt. So bekam ich die Möglichkeit, am Oberstufenzentrum Handel I mein Praktikum zu machen. Diesmal war uns die Auswahl der Schule gänzlich überlassen. Vorgabe waren zwei Aspekte, „ein Gymnasium“ musste es sein, und bewerben mussten wir uns selbstständig. Weil ich beim ersten Mal so gute Erfahrungen gemacht hatte, fiel die Wahl wieder auf den Bezirk Friedrichshain, und zwar diesmal auf die Heinrich-Hertz-Oberschule, deren Homepage ich entnehmen konnte, dass hier ein LK Informatik angeboten wird.

1.2 Kontaktaufnahme

In einer kurzen E-Mail an den Informatiklehrer Herrn Lüdtke schilderte ich mein Anliegen und betonte dabei, dass ich eher unterstützen als zur Last fallen wollte. Nachträglich bin ich mir fast sicher, dass ich mein Versprechen auch eingelöst habe. Prompt erhielt ich eine Antwort mit einem Terminvorschlag für ein Treffen in der Schule, das sehr positiv verlief. In einem etwa halbstündigen Gespräch haben wir unsere inhaltlichen Vorstellungen ausgetauscht. Auf Wunsch von Herrn Lüdtke reichte ich nachträglich meine Kurzbewerbung bei der Schulleitung ein. Nach den Sommerferien meldete ich mich dann telefonisch bei Herrn Lüdtke, er hatte mich bereits bei den Schülern angekündigt. Ich sollte mich dem Kurs vorstellen, also machten wir einen Termin aus. So begann mein Praktikum bereits zum 31. August 2006 im Informatik-LK 13 der Heinrich-Hertz-Oberschule mit einer 45-minütigen Vorstellung meiner Person. Ich hatte mir vorgenommen, meinen Lebenslauf ab der 11. Klasse zu schildern und das habe ich dann auch in die Tat umgesetzt.

1.3 Geschichte der Schule

Die Heinrich-Hertz-Oberschule ist ein Gymnasium mit einem mathematisch- naturwissenschaftlichen Profil. Das Gymnasium befindet sich im Berliner Bezirk Friedrichshain in der Rigaer Straße 81 und ist das 3. Gymnasium des Bezirks. Die Schule blickt auf eine 46-jährige Tradition zurück. Kurz nach der Wende, 1990, erhielt die Schule eine mathematisch-naturwissenschaftliche Profilierung. Das dreistöckige Gebäude liegt heute inmitten der alten Friedrichshainer Hausbesetzer-Szene, setzt sich aber durch seine monumentale Bauweise deutlich von der unmittelbaren Umgebung ab. Der quadratische Bau umgibt den einen im Innenhof gelegenen Schulhof. Die Schule ist bekannt für die rege und erfolgreiche Teilnahme von Schülern an diversen Wettbewerben.

1.4 Die Klassenräume

Die Klassenräume sind verteilt auf allen Etagen. Während meines Praktikums hatte ich primär in den Informatik-Räumen zu tun. Diese liegen im linken Quergebäude, im dritten Stock. Generell sind die Klassenräume für bis zu 30 Schüler ausgelegt und frontal ausgerichtet. Einer der zwei Informatik-Räume ist zwar ebenfalls frontal ausgerichtet, die Computerarbeitsplätze sind jedoch in U-Form an der Wand entlang angeordnet. Der LK 13 fand in diesem Raum statt. Der andere ist inklusive der Computerarbeitsplätze ausschließlich frontal ausgerichtet. Dadurch, dass Computerarbeitsplätze ohnehin mehr Platz benötigen, war der Raum extrem groß und hatte zudem eine für den Unterricht sehr ungünstige Akustik.

1.5 Die technische Ausstattung

In beiden Computerkabinetten waren Whiteboard, Beamer und ein Overhaed-Projektor permanent verfügbar. Auf den Arbeitsplatzrechnern waren jeweils die Betriebssysteme Windows 2000 und KUbunto installiert. Das Netzwerk und die DSL-Leitung waren während meiner Anwesenheit zuverlässig verfügbar. Welche Programme unter Windows vorhanden sind, kann ich nicht sagen, da ich mich während meiner Anwesenheit ausschließlich unter Linux aufhielt, aber ich nehme stark an, dass zumindest eines der gängigen Office-Pakete und ein Grafikprogramm installiert sind. Unter Linux war die von mir auch verwendete Entwicklungsumgebung Eclipse 3.1., OpenOffice und alle sonst standardmäßig installierten Programme verfügbar. Was noch fehlte, war eine für den Unterricht geeignete UML-Anwendung. Mein spontaner Vorschlag war, das Linux Tool «dia» zu installieren, das die wichtigsten UML-Shapes für eine angemesse Modellierung zur Verfügung stellt. Für die Planung meines Unterrichts habe ich ein Werkzeug ausgesucht, das den Code-Transfer in Java zulässt und frei verfügbar ist. Ich wusste von einem UML-Eclipseplugin und nutzte die Gelegenheit, die freie Version von Omondo zu testen. Ich war beeinduckt, auf diese Weise war Modellierung richtig sinnvoll. Im nachhinhein betrachtet hätte ich mir einige Probleme während des Studiums sparen können, wenn ich diese Information früher gehabt hätte.(Omondo Eclipse- Plugin, http://www.omondo.com/download/free/ getestet: 16.09.2006)

2 Klassen, Schüler und Lehrer

2.1 Allgemeines

In den Klassen, in denen ich hospitierte und teilweise unterrichtete, saßen insgesamt etwa zehn Mädchen, vier davon in den Informatik-Stunden(Basiskurs 11), die übrigen in den Mathematik-Stunden der unteren Klassenstufe(Klasse 8). Generell ist der Anteil an Mädchen an der Schule mit ca. 20 Prozent ziemlich gering. Das ist wohl hauptsächlich auf das Profil der Schule zurückzuführen. Ich glaube übrigens nicht, dass Mädchen weniger Verständnis oder Talent für das Fach Informatik mitbringen, sondern dass die Unterrichtsmethoden eher den männlichen Denkstrukturen entsprechen als den natürlichen weiblichen. Einige Mädchen können sich leicht, andere weniger leicht oder gar nicht anpassen. Vielleicht sollte man diesbezüglich mal ein Pilotprojekt andenken, das den geschlechtlich getrennten Unterricht vorsieht; nicht weil Mädchen weniger können, sondern weil sie anders denken als Jungen. Ziel könnte das zentrale Abitur sein. Ich bin mir, was die Informatik angeht, fast sicher, dass es da keine Gewinner geben würde.

2.2 Lehrer

Im ersten Praktikum war es üblich, sich morgens mit den Lehrern im Lehrerzimmer zu treffen. Das war diesmal nicht der Fall. Informatik und Physik haben jeweils einen eigenen Vorbereitungsraum, in dem man sich traf und absprach. Ich fand diese Art der Grüppchenbildung auch amüsant, da der praktizierte Humor unter den Kollegen immer auch eine gewisse Fachspezifik inne hatte. Da ich primär in Begleitung von Herrn Lüdtke und Herrn Trotzke war, hielt ich mich fast ausschließlich in den genannten Räumen auf. In das Lehrerzimmer kam ich kaum. Ich habe die beiden Lehrkräfte als sehr motivierte Personen kennengelernt, die mit einem hohen Einsatz, Flexibilität und Freude den Unterricht gestalten. Mit und für die beiden zu arbeiten war eine Bereicherung. Beide Lehrer hatten sehr unterschiedliche Ansätze beim Unterrichten und im Umgang mit den Schülern, was vielleicht auch durch die unterschiedlichen Größen der Klassen bzw. Kurse bedingt war. Herr Trotzke führte die Basiskurse Klasse 11 und den LK 12, relativ große Gruppen. Er praktizierte eine leichte Version des Prinzips «Zuckerbrot und Peitsche», eine Vorgehensweise, die ich unter den gegebenen Umständen als durchaus angebracht empfand. (Basiskurs: 26 Schüler im Alter zwischen 15 und 17 Jahren und eine schlechte Akustik) Der Unterricht war vor allem für die Jungen motivierend, aber auch nicht ganz ohne einen gewissen «Drill». Bei den Mädchen war ich mir nicht sicher, ob sie mit seiner Art zu unterrichten gut zurecht kamen. Herr Trotzke hat mich durch meinen Unterricht begleitet und mir hilfereiche Verbesserungsvorschläge für meinen Unterricht gemacht. Das betraf den

Java-Unterricht genauso wie den Unterricht zur Software-Entwicklung. Herr Lüdtke unterrichtete ausschließlich den Informatik-LK. Er ist im laufenden Schuljahr an das Institut für Mathematik der HU Berlin abgeordnet. Sein Einsatz für die Schüler war grandios. Er hat die außergewöhnliche Fähigkeit, Schüler zu mehr als der normalen Schulleistung zu motivieren. Seine Schüler nahmen zahlreich an außerschulischen Wettbewerben, Vorträgen und Schulungen teil, wie ich es vorher noch an keiner anderen Schule gesehen habe.(ORP: Im OSZ für Datenverarbeitung und Industrie; UP-Wirtschaft: Am OSZ Handel I, Abteilung 2) Während meiner Zeit habe ich versucht, einen Kontakt zwischen Professor Fischer und Herrn Lüdtke aufzubauen, da die Schüler den Wunsch geäußert hatten, sich an einem Simulationsprojekt aus der Praxis zu beteiligen. Leider habe ich noch keine Antwort von ihm oder seiner Sekretärin bekommen, obwohl ich meine Unterstützung bei der Realisierung angeboten habe.

2.3 Schüler der unteren Klassen

Die Schüler, die ich im Unterricht aus den unteren Klassen begleitet habe, waren viel unruhiger als die aus den oberen Klassen. Hier war der Unterricht durch Laustärke etwas beeinträchtigt.(Primär 8.Klasse Mathematik) Mir scheint es so, als ob die Schüler erst mit der Oberstufe und dem sich nähernden Abitur die Notwendigkeit empfinden, aufmerksam am Unterricht teilzunehmen.

2.4 Schüler der Basiskurse Klasse 11

Beide Basiskurse, die ich unterrichtet habe, waren mit je 26 SchülerInnen voll besetzt. In einem der beiden Kurse war das weibliche Geschlecht mit vier Mädchen vertreten. Die Kurseteilnehmer waren motiviert, teilweise kritisch und sehr kommunikativ. Während des Unterrichts war die Beteiligung der Schüler stets vorhanden. Die Teilnehmer verhielten sich ruhig und waren aufmerksam. Viele haben sich zur Lösung der freiwilligen Hausaufgabe via E-Mail an mich gerichtet, und so kam es auch zu einem regen Wissensaustausch auf dieser Ebene. Ich hatte das Gefühl, Interesse mit meinem Unterricht geweckt zu haben.

2.5 Schüler des LK 12 und 13

In beiden Gruppen waren die Teilnehmer ausschließlich männlich. Die Teilnehmerzahlen waren aufgrund des bestehenden Kurssystems erheblich kleiner als die den Basiskursen. Die Schüler der 12.Klasse LK habe ich als sehr motiviert kennengelert; ausschließlich Jungs, die sich prïmär für die Programmierung kleiner Spiele-Applikationen interessierten und schon Merkmale richtiger «Computer Nerds» aufwiesen.(In Java) Der LK 13 war

dagegen ganz anders. Ein kommunikationsfreudiges, motiviertes „Trüppchen'", die alle an einem Strang zogen und immer zur gegenseitigen Hilfestellung bereit waren. Es war sehr motivierend für mich, einen solchen Kurs zu erleben und zu unterrichten.

3 Mein Stundenplan

3.1 Informatik-Unterricht in der Heinrich-Hertz-Oberschule

4 Der Stoff

4.1 Allgemeines

Hier möchte ich hauptsächlich auf den von mir vermittelten Stoff eingehen und eine Einordnung der Unterrichtseinheiten in den Berliner Rahmenlehrplan vornehmen.

4.2 Java als Programmiersprache

Mit diesem Thema befinden wir uns im Berliner Rahmenlehrplan für die Sekundarstufe II in der Einführungsphase. Diese lief bereits in den Kursen der Basis 11 in Form von Wiederholungen und Vertiefungen der in der Sekundarstufe I erworbenen Kompetenzen. Herr Trotzke hatte bereits mit dem Lernabschnitt 2, „Grundlagen der Programmentwicklung“, des Berliner Rahmenlehrplans begonnen und hatte die Kennnisse der SchülerInnen über “Programmieren im Kleinen“ aufgefrischt.(Berliner Rahmenlehrplan - Sek2 Informatik, S.8, Einführungsphase, Lernabschnitt 2) Hierzu wurde unter anderem der Algorithmusbegriff wiederholt und die Notation für Struktogramme eingeführt. Diese hatte ich seit meiner eigenen Schulzeit nicht mehr gesehen (scheint ein schulisches Rudiment zu sein). So waren die Schüler schon in der Lage kleine Programme anhand einfacher Aufgabenstellungen in Form von Struktogrammen umzusetzen. Daran sollte mein Unterricht „Einführung in die Java-Programmierung“ anschließen.

4.3 Software-Entwicklung

Der LK 13 hatte mit abiturrelevanten Themen für die Softwareentwicklung begonnen. Diese sind Bestandteil des Kapitels 4, „Kompetenzen und Inhalte“, des Berliner Rahmenlehrplans. Durch den Fachlehrer waren bereits die gängigen Vorgehensmodelle eingeführt und mit der Modellierung in UML begonnen worden. Die Notation für die Modellierung von Klassen- und Objektdiagrammen kannten die Schüler. Meine Aufgabe sollte es sein, die notwendige Praxisrelevanz zu schaffen. Hierzu wurde von der Tatsache ausgegangen, dass die Schule ein System zur Fehlzeitenverwaltung der Schüler benötigt. Die Realisierung mit einem Softwaresystem sollte praxisnah im Rahmen eines Kursprojektes erfolgen.

5 Zusammenfassung und Bewertung der Praktikumserfahrung

5.1 Allgemeines

Im allgemeinen war meine Erfahrung in diesem Unterrichtspraktikum positiv. Natürlich gab es Momente, in denen ich mir gewüschnt hätte, ich hätte geschickter auf Schülerfragen reagiert oder die Stundenziele nicht ganz so hoch gesteckt beziehungsweise die Zeit besser im Auge behalten. Aber alles, was ich im Vorfeld über das Image der Schule erfahren hatte, hat sich bestätigt. Die Schüler besitzen überdurchschnittlich gute Voraussetzungen im mathematisch-naturwissenschaftlichen Bereich. Deshalb hatte ich mich natürlich auch mit entsprechend hohen Erwartungen an die Unterrichtsvorbereitung gemacht und wollte die Schüler auf keinem Fall unterfordern.

5.2 Persönliche Einordnung

Hier möchte ich kurz einen Vergleich zu meinem ersten Praktikum ziehen. In den Wirtschaftswissenschaften besitze ich durch mein bereits abgeschlossenes Studium und anschließende berufliche Tätigkeiten eine gewisse Praxiserfahrung. Entsprechend locker bin ich im Rahmen meines ersten Praktikums in den Unterricht gegangen. Das sind die besten Voraussetzungen für den Unterricht in der Sekundarstufe II, sowohl für die Schüler als auch für eine Lehrkraft. Diese Voraussetzung brachte ich für die Informatik nicht in dem Maße mit. Ich habe zwar aufgrund meiner Tätigkeit an der Uni recht umfangreiche Erfahrung in der Systemadministration gesammelt und auch einige Webapplikationen geschrieben. Aber große Projekte im Rahmen der Softwareentwicklung und der objektorientierten Programmierung fehlten bisher gänzlich. Entsprechend umfangreich war die Vorbereitung auf den Unterricht. Auch die Unsicherheit bei dem Versuch, praxisrelevant zu unterrichten, war groß. Ich würde mir wünschen, dass das Referendariat auf ein Jahr verkürzt und dafür um ein verpflichtendes praktisches Jahr in der freien Wirtschaft ergänzt wird. Unter diesen Umständen wäre man als Lehrer(in) bestens für den Unterricht in der Sekundarstufe II gewappnet. Da mir zum Zeitpunkt des Praktikums genau das fehlte, versuchte ich diese Tatsache wettzumachen. Dazu habe ich alle Register gezogen und alle meine Kontakte genutzt, um praktische Tipps und Korrekturen zu meinen Unterrichtsvorhaben zu erhalten.(Vielen Dank an Steffen Eckardt (Technischer Projektleiter der SerCon) und Katrin Miosga (Technische Projektleiterin Ebay Deutschland)) Wie der Unterricht in den einzelnen Fällen verlaufen ist, kann man in den Erfolgskontrollen der jeweiligen Unterrichtseinheit nachlesen. Insgesamt war das Unterrichtspraktikum eine tolle Erfahrung. Aus meiner Sicht kann ich nur sagen, wenn es Berlin schafft, flächendeckend solche engagierten LehrerInnen anzustellen, wie ich Sie kennengelernt habe, dann hätte

die Stadt kein Problem in Sachen Bildung.

6 Erfolgskontrollen der einzelnen Unterrichtseinheiten

6.1 Java-Unterrichtsentwurf 1

Die Unterrichtsstunde begann, wie geplant, mit einer Begrüßung und einer kurzen Vorstellung meiner Person. Damit die Schüler mich ansprechen können, habe ich meinen Namen gut leserlich am rechten oberen Rand des Whiteboards notiert. Da mein Unterricht zwei Unterrichtseinheiten umfasst, wollte ich zu Anfang einen umfassenden Überblick zum folgenden Stoff vermitteln. Zu diesem Zweck kam die von mir erstellte Mindmap zum Einsatz.(Siehe Kapitel 7.6.1 Mindmap) Mit einer kurzen Erläuterung der aufgeführten Punkte sollte dann die Überleitung zum Thema der Stunde erfolgen. Vom Ansatz her war das sicher eine sinnvolle Vorgehensweise. Selbstkritisch möchte ich jedoch anführen, dass der Schriftgrad zu klein gewählt war. Die Schüler waren zu freundlich, um mich darauf aufmerksam zu machen, ansonsten hätte sich das Problem mit einem Klick lösen lassen. Wie Herr Trotzke im nachträglichen Gespräch mitteilte, konnte man die Schrift aus der fünften Reihe kaum lesen. Da die Klasse mit 26 Schülern voll besetzt war, waren mindestens acht Schüler davon betroffen. Auch akustisch hatte der Raum den Nachteil, dass Frontalunterricht meist nur zu realisieren war, wenn man sich als Lehrkraft am linken Rand des Klassenraumes platzierte oder alternativ die eigene Lautstärke entsprechend anpasste. Auf diesen Aspekt wurde ich aber im Vorfeld von Herrn Trotzke aufmerksam gemacht, sodass dies kein Problem war, das mich betraf. Die Erarbeitungsphase teilt sich in meinem Ansatz in zwei Bestandteile: Zum einen die Einführung des «Einfüge-Sortieralgorithmus», zum anderen die Vermittlung der zur Umsetzung benötigten Java-Syntax. Im ersten Teil übernahm ich den Vorschlag der Univertität Wuppertal.(Nachzulesen unter http://www.matheprisma.uni-wuppertal.de/(letzter Aufruf: 30.10.2006)) Auf deren Empfehlung wird ein einfaches Kartenspiel eingesetzt.(Ein Skatkartensatz (ohne Joker)) Da auch ein Skatkartensatz nur beschränkte Kapazitäten hat, wurden von mir drei Schüler ausgewählt. Deren Aufgabe war es, die Karten nacheinander aufzunehmen und gleichzeitig (auf der Hand) der Größe nach aufsteigend zu sortieren. Die zentrale Problemstellung war an die ganze Klasse gerichtet: Welche Frage stellen sich die Probanden mit der Aufnahme jeder einzelnen Karte immer wieder? Die Antworten waren zahlreich, aber nur teilweise enthielten sie Relevanz für die von mir angestrebte algorithmische Problemlösung. Ich hatte zum Beispiel nicht bedacht, die Karten vorab zur Vereinfachung nach Herz, Karo, Kreuz und Pik zu trennen, sodass dies zu Unklarheiten in der Aussagekraft des Ergebnisses führte. Außerdem sprengte das Spiel den zeitlichen Rahmen und viele SchülerInnen waren unbeteiligt. Da ich im Praktikumsverlauf die Chance bekam, den Unterricht in einer weiteren Klasse gleicher Schüleranzahl zu wiederholen, habe ich dieses Spiel gestrichen und bin letztlich wesentlich besser damit

gefahren. Trotzdem hatte ich Interesse auf «mehr» geweckt. Eine von mir erstellte Folienanimation sollte die Vorgehensweise des Algorithmus anhand einer einfachen unsortieren Liste von Integer-Zahlen simulieren.(Siehe Kapitel 7.6.3 Prasentation einfuege.odp/ .pdf) Dies gelang sehr gut, der Pseudocode, den die SchülerInnen im Anschluß produzierten, war in beiden Unterrichtsversuchen überwiegend korrekt und enthielt alle Aspekte des Sortiervorgangs.(Siehe Kapitel 7.6.5 Beispiel für erwarteten Pseudocode) Somit waren die Grundlagen für den weiterführenden Unterricht gelegt. Die Unterrichtsziele, den Algorithmus «Sortieren durch Einfügen» zu erkennen und mit eigenen Worten zu beschreiben sowie angemessenen Pseudocode zu schreiben und Wiederholstrukturen zu erkennen, wurden erreicht. Geplant war, die benötigte Java-Syntax mit Hilfe der fortführenden Präsentation in den Abschnitten Grundstruktur von Javaprogrammen, Zeilenkomentare und Klammerkomentare, ganzzahlige Datentypen und Wertebereiche, Variablendeklaration und Initialisierung, nummerische Listen deklarieren und initialisieren, einfache Java-Operatoren, Anweisungen, Vertauschen zweier Variableninhalte und einfache Ausgaben zu vermitteln und unmittelbar in das Java-Programm «EinfuegeSortieren.java» zu übertragen.(Siehe Kapitel 7.6.3 Präsentation - einfuege.odp/ .pdf) Gleich zu Beginn wurde deutlich, dass das Wechseln zwischen zwei Anwendungen mit dem Beamer die Schüler verwirrt.(Präsentation und Editor) Deshalb bin ich schon während des ersten Unterrichtsverlaufes dazu übergegangen, erst die grundlegende Syntax zu vermitteln und dann den Transfer in Java zu realisieren. Zum Glück hatte ich sowieso angedacht, die Syntax-Bestandteile als Skript an die SchülerInnen auszuteilen, sodass sich jeder während des Transfers daran orientieren konnte.(Siehe Kapitel 7.6.3 Präsentation - einfuege.odp/ .pdf) Die Zeit war knapp bemessen und so endete mein erster Unterrichtsveruch bereits bei dem Transfer der Deklarierung und Initialisierung nummerischer Listen. Hier wurden die letzten drei Ziele nicht erreicht. Im zweiten Unterrichtsversuch dachte ich mit der Streichung des Kartenspiels den zeitlichen Rahmen ausreichend reguliert zu haben. Das war aber nicht der Fall; dieser Versuch endete aufgrund von Zeitknappheit bei dem Abschnitt Vertauschen zweier Veriablen. Es wurde das letzte Ziel nicht realisiert. Trotzdem scheine ich bei einigen Intesse geweckt zu haben, denn fünf Schüler haben einen Beitrag zur Lösung der freiwilligen Hausaufgaben geleistet.(Siehe Kapitel 7.6.4 Präsentation - Aufgabenstellung) Die folgenden Unterrichtseinheiten sollten unmittelbar an den jeweiligen Unterrichtstand anknüpfen.

6.2 Java-Unterrichtsentwurf 2

Mit der Begrüßung begann auch diese Unterrichtsstunde. Da in der letzten Unterrichtseinheit einige Ziele auf der Strecke geblieben waren, war der Einstieg der Stunde klar vorgegeben. Um die noch ausstehende Java-Syntax zu vermittelt, diente meine Präsentation als roter Faden, an dem ich mich zügig und erfolgreich entlanghangeln konnte. (Siehe Kapitel

7.6.3 Präsentation - einfuege.odp/ .pdf) Schnell war die Java-Syntax für Anweisungen, Operatoren und einfache Ausgaben vermittelt. Da der Tausch zweier Variableninhalte in der Praxis recht häufig zu finden ist, sollte dieser Vorgang auf besondere Weise geübt werden. Hier kamen die bereits oben beschriebenen Karten zum Einsatz.(Siehe Kapitel 8.4.2 sowie 8.6.2) Der von mir vorgefertigte Anschrieb am Whiteboard wurde von den SchülerInnen eifrig ergänzt. In einem Nebensatz hatte ich in beiden Unterrichtseinheiten erwähnt, dass diese Art des Tauschens zweier Werte häufig vorkommt und in nahezu jeder Programmiersprache mit Hilfe eines Zwischenspeichers gelöst wird. Um so mehr habe ich mich gefreut, dass ich genau in diesem Unterrichtspart das Gefühl hatte, dass mir am meisten Aufmerksamkeit entgegengebracht wurde. Daraus lässt sich wohl unter anderem schließen, dass die Karten-Methode ihren Zweck erfüllte.(Siehe Kapitel 8.4.2 sowie 8.6.2) Die Vervollständigung des Java-Programms «EinfuegeSortieren.java» erfolgte in beiden Fällen durch die rege Teilnahme der SchülerInnen am Unterrichtsgeschehen. Nun sollte das fertige Programm von den SchülerInnen selbst getestet werden. Dazu hatte ich die Datei via FTP ins Netz hochgeladen. Damit die Schüler die Internetadresse abschreiben konnten, hatte ich diese gut leserlich auf dem Whiteboard notiert und den Vorgang des Downloads exemplarisch via Beamer und Browser demonstriert. Alle Schüler haben darauf hin den Download eigenständig, teilweise mit meiner Hilfe, durchgeführt und die Datei auf den Desktop gespeichert.(Siehe Kapitel 8.6.5 «EinfuegeSortieren.java») Abschließend wurde von mir das Öffnen der Linux-Konsole/ Terminal und die Befehle: «cd» und «..» für das Bewegen im Datei- System, «pwd» zur Bestimmung der aktuellen Position im Dateisystem, «ls» für das Listen eines Ordnerinhalts, «javac» für das Kompilieren und «java» für das Ausführen eines Java-Programms demonstriert.(Siehe Kapitel 8.6.4 Linux-Konsolenbefehle) Auch diesen Vorgang sollten die Schüler eingenständig nachvollziehen. Dies gelang mit meiner Hilfe allen SchülerInnen. Der Eifer war groß, sodass einige SchülerInnen sich gleich an die größtenteils noch offene, freiwillige Hausaufgabe machten oder die weiteren Java-Programme, die ich zum Download bereitgestellt hatte, ausprobierten. Bis zum Stundenabschluß beantwortete ich die individuellen Fragen der Schüler an den jeweiligen Arbeitsplatzrechnern. Ich war erstaunt, wie viel Interesse ich letztlich geweckt hatte. Gefreut habe ich mich besonderst über die rege Teilnahme der vier Schülerinnen, die am zweiten der beiden Basiskurse teilnahmen. Der erste Basiskurs bestand ausschließlich aus männlichen Teilnehmern. Die Unterrichtsziele wurden erreicht.

6.3 UML-Unterrichtsentwurf 1

Die Unterrichtsstunde begann wie geplant mit einer Begrüßung und einer kurzen Vorstellung meiner Person. Damit die Schüler mich namentlich ansprechen konnten, habe meinen ich Namen gut leserlich am rechten oberen Rand des Whiteboards notiert. Da

mit meiner Unterrichtsstunde eine neue Unterrichtseinheit eingeleitet wurde, wollte ich zu Anfang einen umfassenden Überblick zu den folgenden zwei Unterrichtseinheiten schaffen. Für die kommenden zwei Unterrichtseinheitenen sollte das Kursprojekt anhand eines Praxisbeispiels eingeührt werden. Die Idee war mit der Projektdefinition „Fehlzeitenverwaltung der Schüler“ vorgegeben. Die Kenntnisse zur Notation in Klassen- und Objektdiagrammen sowie die grundlegenden Modelle der Softwareentwicklung konnten in meinen Unterrichtsstunden vorausgesetzt werden, da dies bereits Inhalt der letzten Klassenarbeit waren. Da ein Klassendiagramm alle systemrelevanten Klassen in Beziehung setzt, das Gesamtsystem abbildet und außerdem noch Bestandteil der Analyse/Definitionsphase ist, entschied ich mich für diesen Einstiegspunkt. Das Praxisbeispiel „Restaurant“ (Christoph Kercher, 1.Ausgabe 2005), sollte als Vorlage dienen.(Siehe Kapitel 9.6.2 Notation in UML - Klassendiagramme) Die Schüler erhielten eine Kopie. Ein weiterer Auszug enthielt eine Erläuterung zum Lesen des Klassendiagramms. Ich lotste die Schüler durch die wichtigsten Abschnitte und zog Parallelen zum aktuellen Kursprojekt. Das Beispiel zeigt alle Akteure (Gast, Kellner, Koch,...) zusammengefasst in den jeweiligen Klassen und deren Beziehungen untereinander. Im Beispiel waren dazu alle verwendeten Notationselemente beschriftet. Das animiert dazu, selbst kreativ zu werden. Auch ohne mein Eingreifen waren die zwei Teams schnell gebildet. Die Gruppen bekamen die Aufgabe, „Die Schule als System“ in einem Klassendiagramm darzustellen. Zur Dokumentation der Gruppenlösung erhielt jede Gruppe eine Overheadfolie und einen Folienstift. Für die folgenden 45 min war ich vom Frontalunterricht befreit und konnte mich den individuellen Fragen der Gruppen widmen. In regem Austausch untereinander und über die Gruppen hinaus wurden Tipps gegeben und die ersten Skizzen angefertigt. Unklarheiten wurden mit meiner Hilfe, ggf. mit einem Whiteboard, verdeutlicht und mit der ergänzenden Fachliteratur (Christoph Kercher, 1.Ausgabe 2005) geklärt. Die geschlossene Dynamik, Motivation und Begeisterung in der Klasse erfreute mich. An dieser Stelle des Unterrichtsverlaufes wurde mir erstmals klar, welche enorme Auswirkung die Schüleranzahl auf die Lehrer-Schüler-Beziehung hat. Während ich in den großen Basiskursen vor 26 Schülern agierte, war dieser LK Informatik mit 14 Schülern merklich kleiner. Der Unterricht war wesentlich einfacher, flexibler und persönlicher, was mir sehr entgegenkam. Die Gruppenaufgabe erforderte von allen Beteiligten das Lesen und Verstehen eines UML-Klassendiagramms aus der Praxis. Dazu benötigten die Schüler alle wichtigen Notationselemente von Klassendiagrammen und mussten diese im Rahmen der Aufgabenstellung definieren und anwenden können. Auch die Definition des Begriffs Klasse mussten sich die Schüler bei der Umsetzung ständig vor Augen führen. Die Darstellung der Beziehungen zwischen einzelnen Klassen sollten die Schüler anhand der Definition relevanter Methoden, Attribute und deren Typen überprüfen. So sollte laut Aufgabenstellung jede Verknüpfung/ Beziehung von den jeweiligen Gruppen begründet werden können. Die Wertebereiche, Kardinalitäten/ Multiplizitäten

sollten von den Schülern ebenfalls in das System integriert werden. Fragen dazu konnten individuell in den Lehrer-Gruppen-Gesprächen geklärt werden, da auch diese Kenntnisse nur aufgefrischt und mit einem Praxisbezug versehen werden sollten. Da in der Praxis oft die Assiziationsklasse zum Einsatz kommt, sollte dieser Begriff an einem praktischen Beispiel mit einem Whiteboard-Anschrieb erklärt werden. (Siehe Kapitel 9.6.4 Whiteboard - Assoziationsklassen) Ein taktisch guter guter Zug, wie sich nachträglich herausstellte. Denn einige der Gruppen versuchten, das Gelernte gleich in Ihre Gruppenlösung zu integrieren - teilweise korrekt, wie die Gruppenlösungen in der folgenden Unterrichtseiheit zeigten. Abschließend erhielten die Gruppen die Hausaufgabe, das erstellte Klassendiagramm zu vervollständigen. Die Präsentation der einzelnen Gruppenlösungen sollte in der nächsten Unterrichtseinheit erfolgen. Die Ziele für diesen Unterricht wurden vollständig erreicht.

6.4 UML-Unterrichtsentwurf 2

Die Unterrichtsstunde begann wie immer mit der Begrüßung. Wie geplant sollten die Schüler diesmal hauptsächlich Zeit bekommen, ihre Gruppenlösungen zu präsentieren. Um einführend nochmals auf die wichtigen Aspekte beim Lesen eines Klassendiagramms aufmerksam zu machen, wurde von mir nochmals auf den Auszug der letzten Stunde zum „Lesen eines Klassendiagramms'"hingewiesen und wichtige Fakten angesprochen.(Siehe Kapitel 10.6.3 Lesen eines Klassendiagramms) Im Anschluß daran erhielt jede Gruppe etwa 5 Minuten Zeit, um die Ergebnisse zu erläutern. Dabei sollte ein Grupenteilnehmer die genannten systemrelevanten Klassen an der Tafel notieren, der zweite Grupenteilnehmer sollte begrnden, warum diese Klassen notwendig und sinnvoll sind. Erstaunlicherweise hatten einige Gruppen ihre Diagramme bereits mit Dia, einem Modellierungsprogramm unter Linux, realisiert und somit die Hausaufgaben bestmöglichst bewältigt. Auch die Redegewandheit einzelner Schülern verblüffte mich. Die Liste systemrelevanter Klassen wurde über alle Gruppen hinweg stetig durch die Schüler ergänzt und so vervollständigt. Zum Schluß war das Gesamtsystem nahezu komplett mit allen Klassen auf dem Whiteboard gelistet (ist auf die Klasse des schulischen Koordinators, den jedoch selbst ich bei meiner Modellierung im Vorfeld übersehen hatte). Die Schüler waren danach in der Lage, Einzelergebnisse zu einem Geamtergebnis zusammenzuführen, einzelne Komponenten zu identifizieren und zu definieren. Mit der Ausgabe einer von mir erstellten Musterlösung als Kopie wurden die Ergebnisse der Schüler größtenteils bestätigt. (Siehe Kapitel 10.6.1 Mögliche Gruppenlösung) Im weiteren Unterrichtsverlauf sollte nun die Überprüfung anhand eines Objektdiagramms erfolgen. Zur Einführung erfragte ich bei den Schülern die Definition für den Objekt-Begriff. Da die Beantwortung promt erfolgte, war klar, dass die Kenntnisse noch vorhanden waren. Ein weiterer Auszug aus Christoph Kercher, 1.Ausgabe

2005 sollte hier helfen. Der Auszug schildert aufbauend zum bisherigen Beispiel (Restaurant) ein Objektdiagramm und gibt Tipps zum „Lesen eines Objektdiagramms'".[1] Da sich das Vorgehen bereits in der letzten Unterrichtseinheit bewährt hatte, lotste ich die Schüler durch die wichtigsten Abschnitte und zog Parallelen zum aktuellen Kursprojekt. Der Auszug geht ebenfalls auf die Anwendungsbereiche von Objektdiagrammen ein, sie dienen u.a. der Überprüfung des Klassendiagramms auf Korrektheit. So war klar, welchen Zweck ich mit der Erstellung eines Objektdiagramms für das Kursprojekt verfolgte. Die Schüler erhielten die Aufgabenstellung, ein von mir vormodelliertes Objektdiagramm in Bezug auf das bereits vorliegende Klassendiagramm zu überprüfen und ggf. zu ergänzen. Fragen zur Notation in Objektdiagrammen konnten individuell in den Lehrer-Schüler-Gesprächen geklärt werden, da auch diese Kenntnisse nur aufgefrischt und mit einem Praxisbezug versehen werden sollten. Abschließend wurden die Ergebnisse in der Klasse besprochen. Die Ziele für diesen Unterricht wurden vollständig erreicht. In der folgenden Unterrichtseinheit sollte eine geeignete Software-Anwendung zur Modellierung eingesetzt werden, ich berichtete von meiner Erfahrung. Die Vorstellung der einzelnen Software-Anwendungen wurde aus zeitlichen Gründen auf eine der folgenden Unterrichtseinheiten vertagt.

[1] Siehe Kapitel «Modellierung in UML 2.0 - Aggregation der Gruppenergebnisse und Überprüfung der Ergebnisse anhand geeigneter Objektdiagramme» im Anhang: Mögliche Einzellösung

Vorgelegt an der Humboldt-Universität zu Berlin
Institut für Informatik

14. November 2006

= Unterrichtsentwurf =

im Rahmen des 2.Unterrichtspraktikums »im Block«
Zeitraum: 01.09.2006 bis 30.09.2006
Beisitzende Lehrkraft: Herr Trotzke

Lehrende:	Christine Janischek
Schule:	Heinrich-Herz-Oberschule
Klasse/Kurs:	11 Basis
Raum:	Rigaer Straße 81-82
Fach:	Informatik
Datum:	21.09.2006
Zeit:	10:00 bis 11:50
Betreuende Lehrkraft:	Herr Lüdtke

Thema der Unterrichtsreihe:
Einführung in Java

Thema der letzten Unterrichtsstunde:
Kurztest: Algorithmen und Struktogramme

Thema der heutigen Unterrichtsstunde:
Java Syntax - Einführung

Freiwillige Hausaufgaben zur heutigen Unterrichtsstunde:
Erstellen der Rechnen.java und der Erweiterung der EinfuegeSortieren.java laut Aufgabenstellung

Thema der folgenden Unterrichtsstunde:
Erste Programmierschritte in Java

7 Java Syntax - Einführung

7.1 Voraussetzungen

Die Schüler:

- wissen, was ein Struktogramm ist und können anhand konkreter Aufgabenstellungen die Lösung in einem Struktogramm darstellen.
- wissen, was mit mit der Abkürzung EVA-Prinzip gemeint ist und können es erklären .
- wissen, was Datentypen sind und können Beispiele nennen.
- wissen, was ein Algorithmus ist und können den Begriff definieren.
- können die Eigenschaften eines Algorithmus nennen und Beispiele aus dem Unterricht formulieren.

7.2 Unterrichtsziele

Die Schüler:

- können den Algorithmus Sortieren durch Einfügen erkennen und mit eigenen Worten beschreiben.
- können angemessenen Pseudocode schreiben und Wiederholstrukturen erkennen.
- können die Grundstruktur von Java-Programmen erkennen, formulieren und notieren.
- können Zeilenkommentare und Klammerkommentare erstellen.
- können den ganzzahligen Datentypen (byte, short, int, long) einen Wertebereich zuordnen.
- können Variablendeklarationen und Initialisierungen in Java durchführen.
- können nummerische Listen deklarieren und initialisieren.
- können einfache Java-Operatoren definieren.
- können einfache Java-Anweisungen formulieren und in Java-Synatx übersetzen.
- können das Vertauschen zweier Variableninhalte in Java realisieren.
- können einige wichtige Java Operatoren nennen und anwenden.

7.3 Hilfsmittel

Als Hilfsmittel werden benötigt:

- Laptop, Beamer, Präsentationsfolien
- Arbeitsblatt ist in der Präsentation enthalten und wiederholt die praktische Anwendung der gelernten Syntax in Java (Grundstruktur eines Java Programms, Kommentare, Variablendeklaration, Operatoren, Anweisungen, Eingaben-/Ausgaben, kompilieren und ausführen von Programmcode)
- Editor
- Konsole/Terminal
- 1 Skat Kartenspiel
- Overheadfolien und Folienstifte
- Laptop, Beamer, Präsentationsfolien

7.4 Didaktisch-Methodische Reflexion

7.4.1 Basaltext

„Sortieren durch Einfügen“ ist ein spezieller Sortieralgorithmus. Er kann Werte, z.B. Zahlen, in aufsteigender Reihenfolge ordnen. Der Verlauf ist von links nach rechts und sieht so aus:
Je zwei Elemente werden verglichen. Ist der linke Wert größer als der rechte Wert, werden die Elemente vertauscht. Kommt es zum Austausch zweier Elemente, wird das größere Element auch mit den davor liegenden Elementen verglichen. Jedesmal, wenn die Bedingung „größer als“ zutrifft, wird ein Tausch vollzogen. Ist der Algorithmus am Ende der Wertfolge angekommen, ist die Liste (Menge) sortiert. Dieser Algorithmus soll innerhalb dieser Unterrichtsstunde in einem Java-Programm realisiert werden.

7.4.2 Rechtfertigung der didaktisch-methodischen Entscheidungen

Da in der letzten Stunde bereits der Algorithmusbegriff und dessen beispielhafte Veranschaulichung mit Hilfe von Struktogrammen im Rahmen eines Kurztests abgefragt wurde, verfügen die Schüler über eine rudimente Vorstellung von Algorithmen, die in der Praxis angewandt werden. Im Unterrichtsversuch soll praxisorientiert vorgegangen werde. Hierzu wird das Sortierverfahren «Sortieren durch Einfügen» gewählt. Dieses Verfahren soll

exemplarisch in Java-Syntax umgesetzt werden. Um grundlegend in diesen Sortieralgorithmus einzuführen, kommt zu Beginn der Erarbeitungsphase ein Kartenspiel zum Einsatz. Im Rahmen dieses Spiels bekommen drei ausgewählte Schüler nacheinander zehn Skat-Karten auf die Hand. Die Aufgabenstellung besteht darin, die aufgenommenen Karten nacheinander der Größe nach zu sortieren. Die zentrale Fragestellung richtet sich hierbei an die gesamte Klasse: Welche Fragen stellen sich die Probanden mit der Aufnahme jeder ausgeteilten Karte immer wieder? Aufgrund ihrer Vorkenntnisse aus der vergangenen Unterrichtsstunde werden die Schüler wahrscheinlich gleich versuchen, die entsprechenden Bedingungen bzw. Anweisungen (u.a. größer als, vergleiche, tausche) zu formulieren. Je nach bisherigem zeitlichen Verlauf und den noch zur Verfügung stehenden zeitlichen Reserven kann dieser erste Versuch abgebrochen werden oder auch nicht. Durch die nun folgende Präsentation wird der Algorithmus vertieft. (Siehe Kapitel 7.6.3 Präsentation - einfuege.odp/ .pdf) Um den exakten Sortierprozess zu erfassen werden die Schüler nun aufgefordert, mit den bereits genannten möglichen Bedingungen und Anweisungen, Pseudocode zu formulieren. An diesem Punkt soll darauf hingewiesen werden, dass die Effzienz eines Sortierverfahrens daran gemessen werden kann, wie viele Vergleiche und Vertauschungen bestenfalls bzw. schlechtestenfalls erfolgen müssen, um eine Liste zu sortieren. Eine tiefer gehendere Erläuterung soll erst mit der Bekanntheit weiterer Sortierverfahren folgen. Die gemeinsame Erstellung eines Struktogramms soll die Verinnerlichung des Algorithmus bewirken. Im zweiten Teil der Erarbeitungsphase soll nun die Umsetzung in Java-Syntax erfolgen. In immer abwechselden Phasen soll Syntax erlernt und dann auf den umzusetzenden «Sortieren durch Einfüge-Algorithmus» angewandt werden. Schritt für Schritt wächst so ein kleines Javaprgramm. Ziel ist, ein korrektes Java-Programm, welches in der Lage ist, eine vorgegebene Liste aus sieben Integer-Werten der Größe nach aufsteigend mit dem «Sortieren-durch-Einfügen-Algorithmus» zu sortieren. Die in diesem Vorgehen immante praktische Tendenz führt nach meiner Aufassung zu einer nachhaltigen Verankerung des Wissen. Der Ablauf des fertigen Programms via Beamer soll in der Klasse den Anreiz schaffen, den Code mit Hilfe des Rechners zu realisieren, also das Programmieren zu erlernen. Die anschließende Aufgabe (oder freiwillige Hausaufgabe, siehe Folienpräsetation) setzt die neu erworbenen Kompetenzen voraus und soll diese nachhaltig festigen.

7.5 Verlaufsplan

Verlaufsplan der Stunde am 21.09.2006, Klasse 11 Basis Informatik, Heinrich-Herz-Oberschule

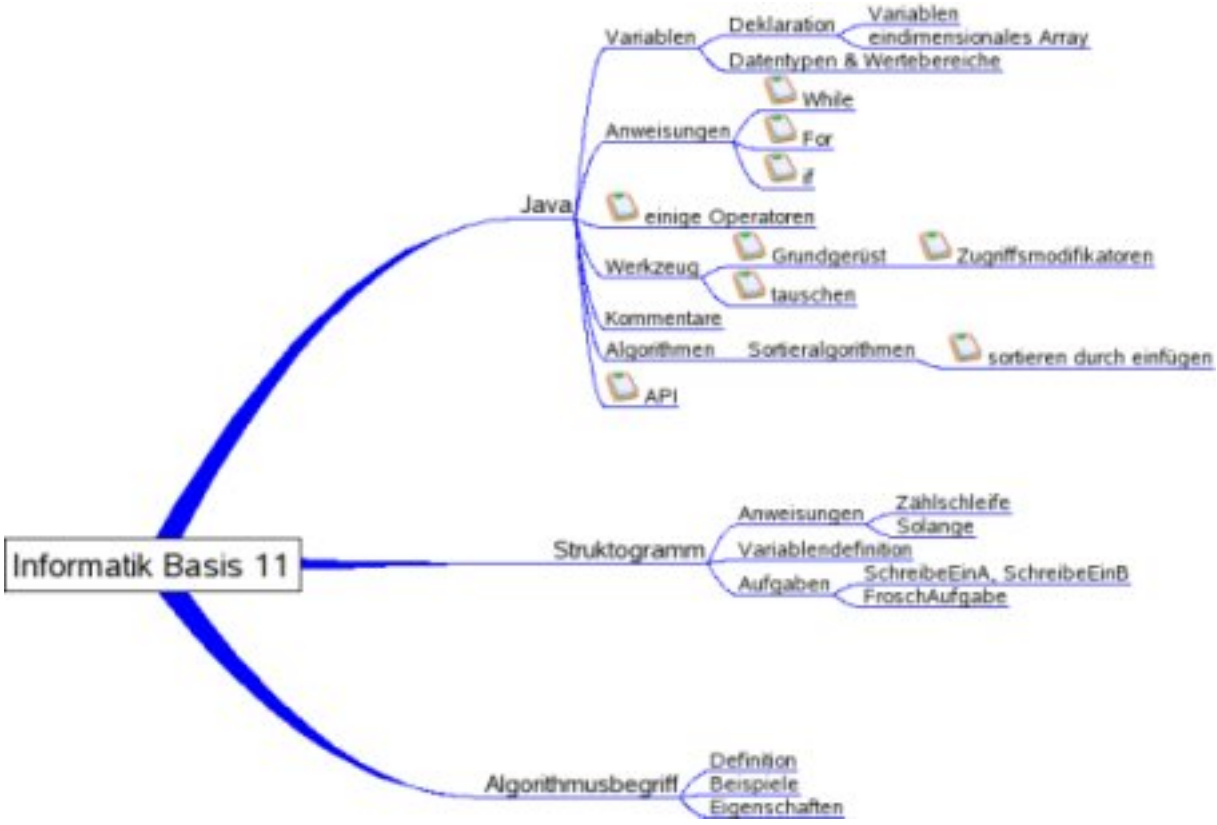
Phase (Beginn, Dauer)	**Inhalt**	**Unterrichtsform**	**Medien**
Einführungsphase 10.00 Uhr 20 Minuten	Begrüßung, Einbettung des Themas in die Stunde Präsentationsfolie (html Dokument) mit der Mindmap zur Einbettng des Themas in den bisherigen Unterrichtsverlauf Die Grafik beinhaltet die exakte Positionierung der kommenden zwei Unterrichtsstunden und gibt gleichzeitig einen Gesamtüberblick über Algorithmen und einfache Java Syntax.	Lehrer-Schüler-Gespräch	Mindmap, Tafel
Erarbeitungsphase 10.20 Uhr 25 Minuten	Kartenspiel: 10 Karten der Größe nach ordnen Forlienpräsentation zum Sortieren durch Einfügen einfügen.odp Erarbeitung der Eigenschaften des Algorithmus anhand der Präsentation Die Schüler haben 5 min Zeit sich Gedanken zu machen und Pseudocode für den Einfüge-Algorithmus zu entwickeln und weitere 5 min um eine ausgewählte Schülerlösung vorzutragen.»Vorgabe:„3 45 6 7 23 89 5“.«	Lehrer-Schüler-Gespräch	Beamer, Präsentation (.odp), Aufgabestellung

Phase (Beginn, Dauer)	**Inhalt**	**Unterrichtsform**	**Medien**
Erarbeitungsphase II 10.55 Uhr 30 Minuten	Einführung von Java: Grundstruktur von Java-Programmen, Zeilenkommentare und Klammerkommentare erstellen. ganzzahlige Datentypen und Wertebereiche, Variablendeklaration und Initialisierung, Nummerische Listen deklarieren und initialisieren, Einfache Java- Operatoren, Vertauschen zweier Variableninhalte, Einfache Ausgaben	Lehrervortrag	Folienprentation

Phase (Beginn, Dauer)	**Inhalt**	**Unterrichtsform**	**Medien**
Auswertungsphase 11.25 Uhr 15 Minuten	Testen des Algorithmus via Beamer. Ausgabe der Aufgabenstellung in Form der Folienpräsentation: (Verändern Sie das Programm so, dass die Eingabe der Werte über die Konsole erfolgen kann und lassen Sie sich die zwischensortierte Liste immer dann ausgeben, wenn ein Wert eingefügt wurde. Außerdem Soll ein Java-Programm erstellt werden, das jeweils zwei Werte addieren, subtrahieren, multiplizieren und ganzzahlig dividieren kann.)	Einzelarbeit	Arbeitsblätter
Aggregation 11.40 Uhr 10 Minuten	Vortrag von zwei Einzellösungen Ausgabe der Mustersterlösung Ausgabe einer Kopie der Forlienpräsentation mit integrierter Aufgabenstellung für die freiwillige Hausaufgabe.	Einzellösung	Kopien

7.6 Anhang

7.6.1 Mindmap



7.6.2 Sortieraufgabe

Sortieren Sie die Zahlenfolge „3 45 6 7 23 89 5“. nach dem Einfügeprinzip! Verwenden Sie dazu einen verständlichen Pseudocode.»

7.6.3 Präsentation - einfuege.odp/ .pdf

Anbei auf dem Datenträger.

7.6.4 Präsentation - Aufgabenstellung

kommt noch.

7.6.5 Beispiel für erwarteten Pseudocode

/*Erste Zahlen: „3 45 “. */

vergleiche 3 und 45
3 größer als 45?
nein - weiter

/* nächste Zahlen: 45 6*/

vergleiche 45 und 6
45 größer als 6?
ja - austauschen

/* nächste Zahlen: 3 6*/

vergleiche 3 und 6
3 größer als 6?
nein - weiter

/* nächste Zahlen: 45 7*/

vergleiche 45 und 7
45 größer als 7?
ja - austauschen

3 6 7 45 23 89 5 /* nächste Zahlen: 6 7*/

vergleiche 6 und 7
6 größer als 7?
nein - weiter

/* nächste Zahlen: 7 45*/

vergleiche 7 und 45
7 größer als 45? nein - weiter

/* nächste Zahlen: 45 23*/

vergleiche 45 und 23
45 größer als 23? ja - austauschen

3 6 7 23 45 89 5 /* nächste Zahlen: 7 23*/

vergleiche 7 und 23
7 größer als 23? nein - weiter

/* nächste Zahlen: 23 45*/

vergleiche 23 und 45
23 größer als 45? nein - weiter

/* nächste Zahlen: 45 89*/

vergleiche 45 und 89
45 größer als 89? nein - weiter

/* nächste Zahlen: 89 5*/

vergleiche 89 und 5
89 größer als 5? ja - austauschen

3 6 7 23 45 5 89 /* nächste Zahlen: 45 5*/

vergleiche 45 und 5
45 größer als 5? ja - austauschen

3 6 7 23 5 45 89 /* nächste Zahlen: 23 5*/

vergleiche 23 und 5
23 größer als 5? ja - austauschen

3 6 7 5 23 45 89 /* nächste Zahlen: 7 5*/

vergleiche 7 und 5
7 größer als 5? ja - austauschen

3 6 5 7 23 45 89 /* nächste Zahlen: 6 5*/

vergleiche 6 und 5
6 größer als 5? ja - austauschen

3 5 6 7 23 45 89 /* nächste Zahlen: 3 5*/

vergleiche 3 und 5
3 größer als 5? nein - weiter

kein weiterer Austausch -> Stop.

„3 5 6 7 23 47 89 “*/

/* mögliche Verkürzung: */

für die Anzahl der Zahlen
vergleiche a und b
wenn a größer tausche zahlen und prüfe nach vorne.
wenn b größer gehe weiter.

Am Ende der Liste angekommen ist die Menge sortiert -> Stop

Vorgelegt an der Humboldt-Universität zu Berlin
Institut für Informatik

14. November 2006

= Unterrichtsentwurf =

im Rahmen des 2.Unterrichtspraktikums »im Block«
Zeitraum: 01.09.2006 bis 30.09.2006
Beisitzende Lehrkraft: Herr Trotzke

Lehrende:	Christine Janischek
Schule:	Heinrich-Herz-Oberschule
Klasse/Kurs:	11 Basis
Raum:	Rigaer Straße 81-82
Fach:	Informatik
Datum:	29.09.2006
Zeit:	10:00 bis 11:50
Betreuende Lehrkraft:	Herr Lüdke

Thema der Unterrichtsreihe:
Einführung in Java

Thema der letzten Unterrichtsstunde:
Java- Syntax - Einführung

Thema der heutigen Unterrichtsstunde:
Erste Programmierschritte in Java Teil 1

freiwillige Hausaufgaben zur heutigen Unterrichtsstunde:
Erstellen der Rechnen.java und der Erweiterung der EinfuegeSortieren.java laut Aufgabenstellung

Thema der folgenden Unterrichtsstunde:
Erste Programmierschritte in Java Teil 2

8 Erste Programmierschritte in Java

8.1 Voraussetzungen

Die Schüler:

- wissen, was ein Sortieralgorithmus ist und können den Sortieralgorithmus «Sortieren durch Einfügen» erkennen und beschreiben.
- wissen, was Pseudocode ist, können ihn eigenständig formulieren und Wiederholstrukturen erkennen.
- wissen, die Grundstruktur von Java-Programmen, können diese formulieren und notieren.
- wissen, was Zeilenkommentare und Klammerkommentare sind und können diese selbst erstellen.
- wissen, die ganzzahligen Datentypen (byte, short, int, long) und können diesen den entsprechenden Wertebereich zuordnen.
- wissen, was mit Variablendeklaration und Initialisierung gemeint ist, können diese unterscheiden und durchführen.
- wissen, was mit der Deklaration und Initialisierung nummerischer Listen gemeint ist, können diese unterscheiden und durchführen.
- wissen, was einfache Java-Operatoren sind und können diese definieren und anwenden.
- wissen, was einfache Anweisungen sind und können Beispiele nennen und anwenden.
- wissen, wie Sie das Vertauschen zweier Variableninhalte realisieren können.

8.2 Unterrichtsziele

Die Schüler:

- können einfache Anweisungen (aus dem Struktogramm, Pseudocode) in Java- Syntax übersetzen und in ein Java- Programm implementieren.
- können einige wichtige Java-Operatoren nennen und anwenden.
- können das Vertauschen zweier Variableninhalte in Java realisieren.
- können einfache Ausgaben in Java realisieren.

- können mit dem Browser eine Java-Datei aus dem Internet downloaden.
- können die Linux-Konsole/ Terminal starten.
- können sich anzeigen lassen, wo Sie sich auf der Linux-Konsole befinden.(pwd)
- können sich auf der Linux-Konsole die Ordnerinhalte anzeigen lassen.(ls)
- können sich auf der Linux-Konsole bewegen.(cd, ..)
- können Java-Code auf der Linux-Konsole compilieren.(javac)
- können Java-Code auf der Linux-Konsole ausführen.(java)

8.3 Hilfsmittel

Als Hilfsmittel werden benötigt:

- Laptop, Beamer, Präsentationsfolien
- Arbeitsblatt ist in der Präsentation enthalten und wiederholt die praktische Anwendung der gelernten Syntax in Java (Grundstruktur eines Java Programms, Kommentare, Variablendeklaration, Operatoren, Anweisungen, Eingaben-/Ausgaben, kompilieren und ausführen von Programmcode)
- Editor
- Konsole

8.4 Didaktisch-Methodische Reflexion

8.4.1 Basaltext

„Sortieren durch Einfügen“ ist ein spezieller Sortieralgorithmus. Er kann Werte, z.B. Zahlen, in aufsteigender Reihenfolge ordnen. Der Verlauf ist von links nach rechts und sieht so aus:
Je zwei Elemente werden verglichen. Ist der linke Wert größer als der rechte Wert, werden die Elemente vertauscht. Kommt es zum Austausch zweier Elemente, wird das größere Element auch mit den davor liegenden Elementen (links davon), verglichen. Jedesmal, wenn die Bedingung „größer als“ zutrifft, wird ein Tausch vollzogen. Ist der Algorithmus am Ende der Wertfolge angekommen, ist die Liste (Menge) sortiert. Dieser Algorithmus soll heute in Java fertiggestellt werden.

8.4.2 Rechtfertigung der didaktisch-methodischen Entscheidungen

Da bereits eine Unterrichtsstunde zum Thema „Java Syntax - Einführung“ stattgefunden hat, verfügen die Schüler über eine gewisse Vorstellung über das Grundgerüst, ganzzahlige Datentypen, nummerische Listen, Kommentare, Variablendeklaration mit Initialisierung und Anweisungen in Java. Fortführend zur letzten Unterrichtseinheit soll heute das gemeinsam erstellte Java-Programm in einem ersten selbstständigen Versuch der Schüler getestet werden. Hierzu werden die grundlegenden Befehle der Linux-Konsole vermittelt.

Da in der letzten Stunde aus Zeitknappheit der Unterricht nicht zum geplanten Ziel geführt hat, soll heute der Abschluss erfolgen. Zu Beginn wird der «Einfüge-Algorithmus» erfragend erarbeitet. In einem Fall der beiden Unterrichtsversuche zum ersten Thema war der aktuelle Stand die Listen-Deklaration und Initialisierung. Genau an diesem Punkt wurde angeknüpft. Zuerst wurde die noch fehlende Java- Syntax anhand der Präsentation eingeführt. Unmittelbar danach wurde der Transfer vollzogen, indem das Java-Programm «EinfuegeSortieren.java» vervollständigt wurde. Als Editor diente Eclipse 3.1.. Um den Tauschprozess innerhalb des Algorithmus zu festigen, werden von mir vorgefertigte Karten an der Tafel befestigt. Drei rote Karten stellen Speicherplätze dar. Diese sind mit der bereits erlernten Java-Syntax beschriftet. Drei weitere, allerdings grüne Karten sind mit Werten aus dem Beispiel der letzten Stunde beschriftet. Da sich bei dem Austausch zweier Werte immer zwei Lösungsmöglichkeiten ergeben, werden diese von mir im Vorfeld an der Tafel angedeutet. Die Lösung selbst sollen jeweils 2 Schüler an der Tafel erarbeiten. Ein Schüler/ eine Schülerin soll die Karten schrittweise versetzen, der/ die andere soll die jeweilige Java-Syntax dazu auf dem Whiteboard notieren. Eine der Lösungen soll von mir direkt in das Programm übernommen werden. Das Java-Programm «EinfuegeSortieren.java» wurde im anderen Fall bereits in der letzten Stunde bis zu genau diesem Tauschprozess gemeinsam mit den Schülern erstellt. Ist der Tauschprozess umgesetzt, wird eine einfache Ausgabe syntaktisch ergänzt. Um die Motivation zu steigern, wird das Programm gestartet und ausgeführt. Da die Schüler noch keine Kenntnisse zur Benutzung der Konsole besitzen, werden die benötigten Befehle eingeführt.(pwd/ls/cd/javac/java) Ergänzend wird von mir erläutert, wo und wie Java-Testprogramme aus dem Internet heruntergeladen werden können. Die Schüler erhalten dann den Arbeitsauftrag, die Dateien eigenständig aus dem Netz herunterzuladen, zu kompilieren und auszuführen. Meine Aufgabe besteht darin, individuelle Hilfestellung zu leisten. Schnellere Schüler erhalten die Aufgabe, den Teil 1 der Hausaufgabe zu lösen.

Die anschließende Aufgabe (oder Hausaufgabe, siehe Präsentation - Hausaufgabe Teil2) ist freiwillig, setzt die neuen Kompetenzen voraus und wiederholt den iterativen Vorgang, mittels Pseudocode ein Struktogramm zu erzeugen und dann zur Codegenerierung über-

zugehen. Die Aufgabe dient der Festigung und Anwendung der bis zu diesem Zeitpunkt erarbeiteten Syntax in Java. Darüber hinaus garantiert dieses Vorgehen, dass sich die einzelnen SchülerInnen mit neu erlerntem Wissen auseinandersetzen sollen. Die SchülerInnen bekommen die Aufgabenstellung in schriftlicher Form, versehen mit einem Abgabezeitpunkt und meiner Emailadresse für eventuelle Klärung von Fragen.

8.5 Verlaufsplan

Verlaufsplan der Stunde am 29.09.2006, Klasse 11 Basis Informatik, Heinrich-Herz-Oberschule

Phase (Beginn, Dauer)	**Inhalt**	**Unterrichtsform**	**Medien**
Einführungsphase 10.00 Uhr 20 Minuten	Begrüßung, Fortführung der letzten Stunde Rekonstruktion des erlernten Einfüge-Algorithmus Rekonstruktion des nun benötigten Tauschprozesses (vorgefertigte Karten) Jeweils 2 SchülerInnen setzten die 2 besehenden Lösungsmöglichkeiten in Teamarbeit an der Tafel um	Lehrer-Schüler-Gespräch	Tafel, Karten
Erarbeitungsphase 10.20 Uhr 25 Minuten	Eine Lösung wird in das bereits bestehende Programm «EinfuegeSortieren.java» übernommen Eine einfache Ausgabe der Liste wird syntaktisch definiert und ergänzt einfügen.odp Die Erarbeitung erfolgt paralell anhand der Präsentation »Das fertige Programm «EinfuegeSortieren.java» wird von mir, auf der Konsole, kompiliert und ausgeführt; dabei werden die benötigten Konsolenbefehle eingeführt (ls,cd,javac,java –> siehe Anhang)«	Lehrer-Schüler-Gespräch	Beamer, Präsentation (.odp), Aufgabestellung, Editor (Eclipse 3.1)

Phase (Beginn, Dauer)	**Inhalt**	**Unterrichtsform**	**Medien**
Auswertungsphase 10.55 Uhr 30 Minuten	Ausgabe einer Übersicht zu den benötigten Konsolen-Befehle. Die Schüler erhalten die Arbeitsanweisung das Testprogramm herunterzuladen, zu kompilieren und auszuführen. Dazu wird die Vorgehensweise exemplarisch via Beamer vorgeführt. Die Schüler bekommen die Aufgabe die gezeigte Vorgehensweise an Ihren Arbeitsplatzrechner eigenständig nachzuvollziehen Jeder Schüler soll das Programm downloaden, auf der Konsole kompilieren und ausführen	Gruppenarbeit	Arbeitsblätter
Aggregation 11.25 Uhr 15 Minuten	Ausgabe der freiwilligen Haushaufgabe Teil 1 und Teil 2 Ausgabe der Folienpräsentation - Java- Syntax Klärung eventueller Fragen zu den Hausaufgaben Wenn noch Zeit bleibt können die SchülerInnen mit der Bearbeitung beginnen	Lehrer-Schüler-Gespräch	Kopien

8.6 Anhang

8.6.1 Präsentation - einfuege.odp

Anbei auf dem Datenträger.

8.6.2 Karten

Karten

8.6.3 Beispiel für Tafelanschrieb

8.6.4 Linux- Konsolenbefehle

Linux- Konsole/ Terminal und die Befehle: «cd» und «..» für das Bewegen im Datei-System,

«pwd» zur Bestimmung der aktuellen Position im Dateisystem,

«ls» für das Listen eines Ordnerinhalts,

«javac» für das Kompilieren eines Java- Programms,

«java» für das Ausführen eines Java- Programms.

8.6.5 Code der «EinfuegeSortieren.java»

```
public class EinfuegeSortieren {
        public static void main(String[] args){

                int[] Liste = {3, 45, 6, 7, 23, 89, 5};
                sortieren(Liste);
                System.out.println("'" + Liste[0] + "'" + "'" + Liste[1] + "'"+ "'" + Liste[2] + "'" + "'" +
                     Liste[3] + "'" + Liste[4]+ "'" + Liste[5] + "'" + Liste[6] + "'" + "␣sortiert!");
        }
        static void sortieren(int[] Liste){
                int i,k; // Variablendeklarationen
                int temp;
                System.out.println("sortiere␣folgende␣Liste:␣'" + Liste[0] + "'" +"'" + Liste[1] + "'"+ "'" +
                     Liste[2] + "'" + "'" + Liste[3] + "'" + Liste[4]+ "'" + Liste[5] + "'" + Liste[6] + "'" );

                /*Alternative: elegantere loesung:
                System.out.println("sortiere folgende Liste: ");
                for(i = 0; i < Liste.length; i++){
                        System.out.print("'" + Liste[i] + "'");
                }
                System.out.println("los gehts...");
                */

                for(i = 1; i < Liste.length; i++) {// Anweisung

                        if(Liste[i - 1] > Liste[i]){
                                temp = Liste[i];
                                k=i;

                                while((k > 0) && Liste[k - 1] > temp){
                                        Liste[k] = Liste[k - 1];
                                        k--;
                                        Liste[k] = temp;
                                        System.out.print("'" + Liste[0] + "'" + "'" + Liste[1] +"'"+ "'" +
                                             Liste[2] + "'" + "'" + Liste[3] + "'" + Liste[4]+"'" + Liste[5] +
                                             "'" + Liste[6] + "'");
                                        System.out.println("___'" + Liste[k] + "'␣eingefuegt." );

                                }

                        }
                }

        }
}
```

8.6.6 Code der «EinfuegeSortierenAusgabe.java»

```
import java.io.*;

public class EinfuegeSortierenAusgabe {

        public static int[] Liste = new int[7];
        public static String rein;

        public static void main(String[] args)
        throws IOException
        {

                eingabe(Liste);
                // clear();
                sortieren(Liste);
                ausgabe(Liste);
                System.out.println("sortiert!!␣:)");
        }
        static void sortieren(int[] Liste){
                int i,k; // Variablendeklarationen
                int temp;

                for(i = 1; i < Liste.length; i++) {// Anweisung

                        if(Liste[i - 1] > Liste[i]){
                                temp = Liste[i];
                                k=i;
                                while((k > 0) && Liste[k - 1] > temp){
                                        Liste[k] = Liste[k - 1];
                                        k--;
                                        Liste[k] = temp;
                                        ausgabe(Liste);
                                        System.out.println("___'" + Liste[k] + "'␣eingefuegt." );

                                }
```

```
                }
        }

}

public static void clear () {
        int i = 0;

        for(i=0;i<300;i++){
                System.out.println("\n");
        }

}

public static void eingabe(int[] Liste){

        String rein;
        System.out.println("Sieben Zahlen eingeben - jeweils <return>:");
        for(int a = 0;a < Liste.length; a++){
                rein ="";
                try {
                        BufferedReader eingabe = new BufferedReader(new InputStreamReader(System.in));
                        rein = eingabe.readLine();
                }
                catch (IOException eingabeFehler){
                        System.out.println("Fehler");
                }
                int raus = Integer.parseInt(rein);
                Liste[a] = raus;

        } //clear();
        for(int i = 0; i < Liste.length; i++){
                System.out.print("'" + Liste[i] + "'");
        }
        System.out.println("los gehts...");

}

public static void ausgabe(int[] Liste){
        //Alternative: elegantere loesung:
        for(int i = 0; i < Liste.length; i++){
                System.out.print("'" + Liste[i] + "'");
        }


}

}
```

8.6.7 Code der «Rechnen.java»

```
public class Rechnen {

        /**
         * @param args
         */
        public static void main(String[] args) {
                int[] Liste = {18, 6};
            int i = Liste[0];
            int j = Liste[1];
            System.out.println("Rechnen mit " + i + " und "+ j );
            System.out.println("#####################################################");

                // verweis auf die methoden die mit der Liste rechnen sollen
                addieren(Liste);
                subtrahieren(Liste);
                multiplizieren(Liste);
                dividieren(Liste);



        }

        static void addieren(int[] Liste){
            int i = Liste[0];
            int j = Liste[1];
            int summe = i + j;

            System.out.println("Ergebnis der Addition: " + summe);
            System.out.println("#####################################################");

        }
        static void subtrahieren(int[] Liste){
            int i = Liste[0];
            int j = Liste[1];
```

```
            int summe = i - j;

            System.out.println("Ergebnis der Subtraktion: " + summe);
            System.out.println("#######################################################");


        }
        static void multiplizieren(int[] Liste){
            int i = Liste[0];
            int j = Liste[1];
            int summe = i * j;
            System.out.println("Ergebnis der Multiplikation: " + summe);
            System.out.println("#######################################################");


        }
        static void dividieren(int[] Liste){
            int i = Liste[0];
            int j = Liste[1];
            if (j == 0)
            System.out.println("Division durch 0 nicht moeglich:");
            int summe = i / j;
            System.out.println("Ergebnis der Division: " + summe);
            System.out.println("#######################################################");
        }
}
```

8.6.8 Code der «RechnenErweitert.java»

```
import java.io.*;

public class RechnenErweitert {

        /**
         * @param args
         */
        public static void main(String[] args)
        throws IOException{
                // eine liste mit 2 zahlen vom typ integer
                int []Liste = new int[2];
                String rein;
                System.out.println("Wir wollen nun mit zwei Zahlen rechnen");
                for (int a=0; a < Liste.length; a++)
                {
                        int nr = a+1;
                        System.out.println("Eingabe Zahl " + nr + ":" );
                        rein ="";
                        BufferedReader eingabe= new BufferedReader(
                                        new InputStreamReader(System.in));
                        rein = eingabe.readLine();
                        int raus = Integer.parseInt(rein);
                        Liste[a] = raus;
                }

            int i = Liste[0];
            int j = Liste[1];
            System.out.println("Rechnen mit " + i + " und "+ j );
            System.out.println("#######################################################");

                // verweis auf die methoden die Liste rechnen sollen
                addieren(Liste);
                subtrahieren(Liste);
                multiplizieren(Liste);
                dividieren(Liste);



        }

        static void addieren(int[] Liste){
            int i = Liste[0];
            int j = Liste[1];
            int summe = i + j;

            System.out.println("Ergebnis der Addition: " + summe);
            System.out.println("#######################################################");

        }
        static void subtrahieren(int[] Liste){
            int i = Liste[0];
            int j = Liste[1];
            int summe = i - j;

            System.out.println("Ergebnis der Subtraktion: " + summe);
            System.out.println("#######################################################");
```

```
        }
        static void multiplizieren(int[] Liste){
            int i = Liste[0];
            int j = Liste[1];
            int summe = i * j;
            System.out.println("Ergebnis der Multiplikation: " + summe);
            System.out.println("#######################################################");


        }
        static void dividieren(int[] Liste){
            int i = Liste[0];
            int j = Liste[1];
            if (j == 0)
            System.out.println("Division durch 0 nicht moeglich:");
            int summe = i / j;
            System.out.println("Ergebnis der Division: " + summe);
            System.out.println("#######################################################");
        }
}
```

Vorgelegt an der Humboldt-Universität zu Berlin
Institut für Informatik

14. November 2006

= Unterrichtsentwurf =

im Rahmen des 2.Unterrichtspraktikums »im Block«
Zeitraum: 01.09.2006 bis 30.09.2006
Beisitzende Lehrkraft: Herr Lüdke

Lehrende:	Christine Janischek
Betreuende Lehrkraft:	Herr Lüdtke
Schule:	Heinrich-Herz-Oberschule
Fach:	Informatik
Klasse/Kurs:	13 LK Informatik
Raum:	302
Datum:	11.09.2006
Zeit:	10:00 bis 11:50 h

Thema der Unterrichtsreihe:
Software-Entwicklung

Thema der letzten Unterrichtsstunde:
Klausur: Vorgehensmodelle der Software-Entwicklung und Modellierung (Klassen- und Objektdiagramme in UML)

Thema der heutigen Unterrichtsstunde:
Modellierung in UML 2.0 - Klassendiagramme - ein Beispiel aus der Praxis zur Einführung des Kursprojektes

Hausaufgaben zur heutigen Unterrichtsstunde:
Aufgabenstellung: „Modellierung der Schule als Klassendiagramm"

Thema der folgenden Unterrichtsstunde:
Modellierung in UML 2.0 - Aggregation der Gruppenergebnisse und Überprüfung der Ergebnisse anhand geeigneter Objektdiagramme

9 Modellierung in UML 2.0 - Klassendiagramme - ein Beispiel aus der Praxis

9.1 Voraussetzungen

Die Schüler:

- wissen, was Vorgehensmodelle sind, können diese nennen (Wasserfallmmodell, V-Modell, Spiral-Modell, Rational Unified Process, Rapid Application Development, Extreme Programming) und definieren.
- kennen die Definition der fünf grundlegenden Phasen der Softwareentwicklung: Idee, Analyse, Design, Implemetierungs- und Testphase
- und können Klassendiagramme in den Phasenablauf einordnen.
- wissen, was ein Klassendiagramm ist.
- kennen die grundlegenden Notationen in Klasssen- und Objektdiagrammen.
- besitzen grundlegende Java-Programmierkenntnisse (Grundstruktur eines Java Programms, Kommentare, Variablendeklaration, Operatoren, Anweisungen, Eingaben-/Ausgaben, Kompilieren und Ausführen von Programmcode).
- kennen das Ziel des Kursprojektes: Ein Javaprogramm, das die Fehlzeiten der Schüler erfasst.

9.2 Unterrichtsziele

Die Schüler:

- können einfache Klassendiagramme aus der Praxis lesen, verstehen und erläutern.
- können wichtige Notationselemente von Klassendiagrammen definieren und Beispiele zu deren praktischen Einsatz nennen.
- können den Begriff der Klasse definieren.
- können anhand einer Aufgabenstellung ein Klassendiagramm erstellen.
- können im Klassendiagramm Attribute und Methoden definieren.
- können die Begriffe Multiplizität bzw. Kardinalität nennen, definieren und Beispiele zu ihrem Einsatz nennen.
- können alle wichtigen Sichtbarkeitsattribute nennen und definieren.

- können den Attributtyp definieren.
- können den Begriff der Assoziationsklasse erläutern und
- können anhand eines Beispiels deren Funktion erläutern.

9.3 Hilfsmittel

Als Hilfsmittel werden benötigt:

- Auszüge eines Klassendiagramms aus dem Handbuch der UML 2.0 von Christoph Kercher, 1.Ausgabe 2005, S.31
- Overheadfolien und Folienstifte
- Whiteboard

9.4 Didaktisch-Methodische Reflexion

9.4.1 Basaltext

Klassendiagramme werden üblicherweise in den ersten beiden Phasen der Softwareentwicklung erstellt. In den meisten Vorgehensmodellen sind dies die Analyse-/Definitions- und Entwurfs-/Designphase. Klassendiagramme werden jedoch während des gesamten Softwareentwicklungsprozesses gewinnbringend eingesetzt. Anhand der Theorie soll die Praxis Gelerntes festigen. Das Kursprojekt „Fehlzeitenverwaltung der Schüler'ßoll umgesetzt werden. Dabei beginnen wir mit der Modellierung eines Klassendiagramms und befinden uns dabei in der Analyse-/Definitionsphase.

9.4.2 Rechtfertigung der didaktisch-methodischen Entscheidungen

Da bereits Grundkenntnisse der Java Programmierung vermittelt wurden, verfügen die Schüler über eine rudimentï¿½e Vorstellung bezüglich des Umfanges des Kursprojektes. Die Modellierung des Systems soll Klarheit schaffen und die benötigten Resourcen abstecken. Sie soll die wichtigen Aspekte der ganzheitlichen Systembetrachtung vermitteln und den notwendigen Praxisbezug herstellen.

Um grundlegend in die Notation einzuführen, kommt zu Beginn der Erarbeitungsphase der Auszug eines Klassendiagramms aus dem Handbuch der UML 2.0 von Christoph Kercher, 1.Ausgabe 2005, S.31, zum Einsatz. Der Auszug schildert zur Wiederholung die grundlegende Phaseneinteilung der meisten Vorgehensmodelle der Softwareentwicklung und bettet das Klassendiagramm sinnvoll in den Gesamtzusammenhang ein. Das Beispiel

aus dem Handbuch beinhaltet alle wichtigen Notationselemente in beschrifteter Form, sodass es sich insgesamt als wenig erklärungsbedürftig darstellt. Die Schüler sind nun in der Lage, mit Hilfe dieser „Vorlage“ einen Transfer zu vollziehen. Die Aufgabenstellung besteht darin, die Schule als System in einem Klassendiagramm zu erfassen. Die Eingrenzung mit Fokus auf das Kursprojekt (Fehlzeitenverwaltung) soll erst dann erfolgen, wenn ein ganzheitlicher Überblick geschaffen wurde. Die Aufgabenstellung soll in Zweiergruppen bearbeitet werden. Die Schüler erhalten pro Gruppe eine Overheadfolie und einen Folienstift. Mit diesen Utensilien sollen die Gruppenergebnisse dokumentieren werden. Je nach bisherigem zeitlichen Verlauf und den noch zur Verfügung stehenden zeitlichen Reserven kann dieser erste Modellierungsversuch um eine weitere Unterrichtsstunde erweitert werden. Die Schüler haben so genug Zeit, das Feintuning des entwickelten Klassendiagramms zu vollziehen. Hierzu sollen Sie alle notwendigen Attribute und Methoden definieren. Die Gruppenergebnisse werden in einer darauf folgenden Unterrichtseinheit präsentiert. Danach soll die Aggregation der Gruppenergebnisse zu einem Gesamtergebnis/Gesamtsystem erfolgen. Zur Überprüfung des Gesamtkonzeptes wird gemeinsam mit den Schülern ein Objektdiagramm erstellt. Geplant ist in den folgenden Unterrichtsstunden der Einsatz eines Software-Werkzeuges, das die Modellierung in UML und den Java-Code-Transfer ermöglicht. Sind alle systemrelevanten Komponenten detailgenau im Klassendiagramm enthalten, soll das Pflichtenheft (Baseline) für das Kursprojekt abgeleitet werden.

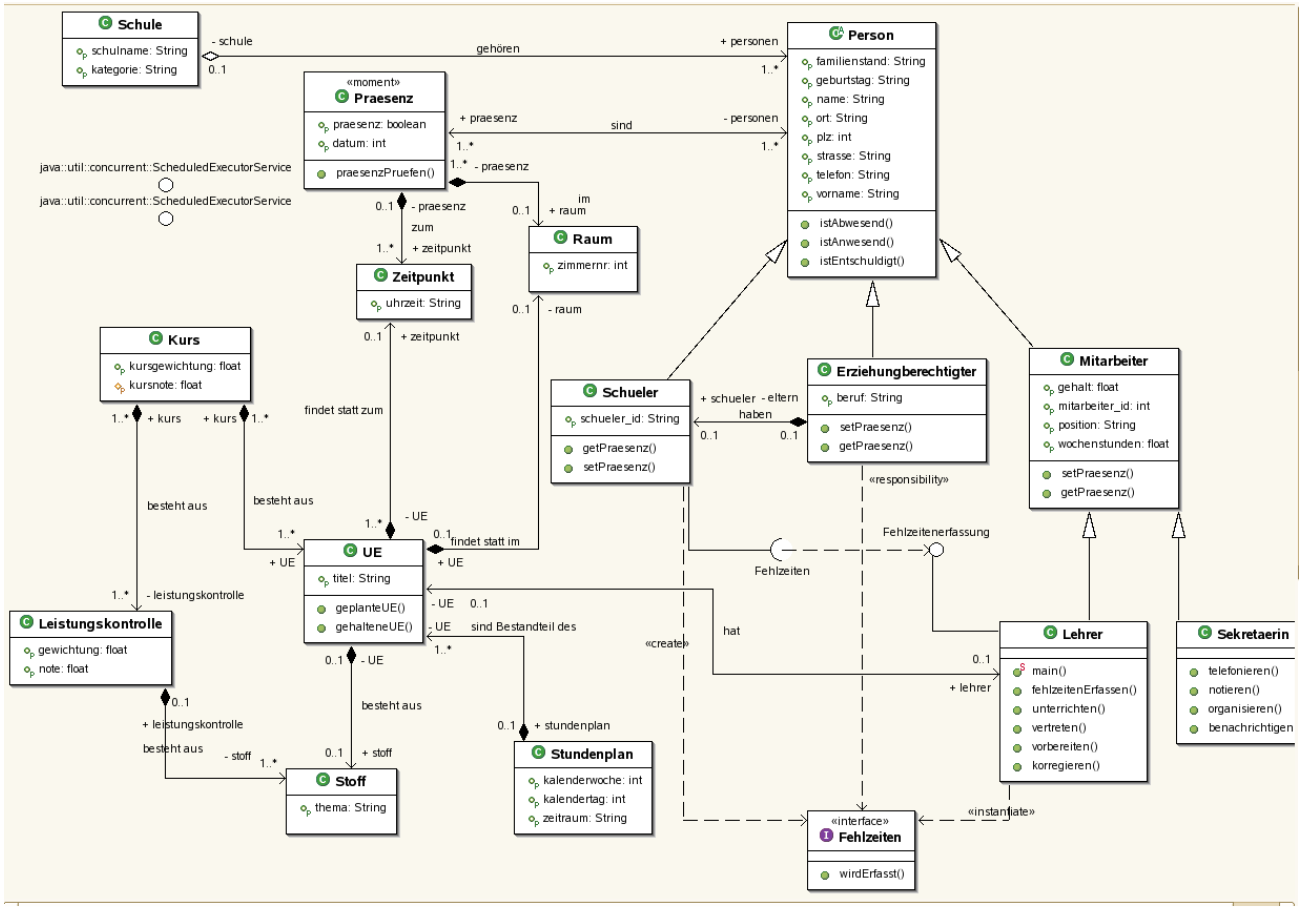
9.5 Verlaufsplan

Verlaufsplan der Stunde am 14.09.2006, Informatik 13 Klasse LK, Heinrich-Herz-Oberschule

Phase (Beginn, Dauer)	**Inhalt**	**Unterrichtsform**	**Medien**
Einführungsphase 10.00 Uhr 20 Minuten	Kurze Wiederholung zu grundlegenden Phasen der Vorghensmodelle Begrüßung, Einbettung des Themas in die Stunde Besprochen wird ein Auszug eines Klassendiagramms aus dem Handbuch der UML 2.0 von Christoph Kercher, 1.Ausgabe 2005, S.31	Lehrer-Schüler-Gespräch	Kopien des Auszugs
Erarbeitungsphase 10.20 Uhr 35 Minuten	Gruppenaufgabe (zweier Gruppen) Modellierung der Schule (als System) in einem Klassendiagramm Die Schüler haben den den Rest des angebrochenen Unterrichtsblockes Zeit, sich mit der Augabenstellung zu beschäftigen und eine präsentable Lösung zu konstruieren. Die Gruppenpräsentationen sind für den folgenden Unterrichtsblock geplant.	Gruppenarbeit	Overhaedfolie und Folienstifte
Auswertungsphase 11.15 Uhr 35 Minuten	Zusammenfassend sollen letzte Fragen zur Notation in Klassendiagrammen geklärt werden. Erläuterung der bisherigen Gruppenlösungen im Lehrer-Gruppen-Gespräch Definition einer Assotiationsklasse in UML 2.0	Lehrer-Schüler-Gespräch	Overhaedfolie und Folienstifte, Whiteboard
Aggregation 11.50 Uhr 10 Minuten	Als Hausaufgabe sollen die Schüler ggf. das Feintuning des Klassendiagramms fertig stellen.	Lehrer-Schüler-Gespräch	7 Overheadfolien

9.6 Anhang

9.6.1 Mögliche Gruppenlösung



9.6.2 Notation in UML - Klassendiagramme

Auszug aus dem Handbuch der UML 2.0 von Christoph Kercher, 1.Ausgabe 2005, S.31

9.6.3 Lesen eines Klassendiagramms

Auszüg aus dem Handbuch der UML 2.0 von Christoph Kercher, 1.Ausgabe 2005, S.103-106

9.6.4 Tafelbild - Assoziationsklassen

Auszug aus dem Handbuch der UML 2.0 von Christoph Kercher, 1.Ausgabe 2005, S.64f

Vorgelegt an der Humboldt-Universität zu Berlin
Institut für Informatik

14. November 2006

= Unterrichtsentwurf=

im Rahmen des 2.Unterrichtspraktikums »im Block«
Zeitraum: 01.09.2006 bis 30.09.2006
Beisitzende Lehrkraft: Herr Lüdke

Lehrende:	Christine Janischek
Betreuende Lehrkraft:	Herr Lüdke
Schule:	Heinrich-Herz-Oberschule
Fach:	Informatik
Klasse/Kurs:	13 LK Informatik
Raum:	302
Datum:	13.09.2006
Zeit:	12:05 bis 14:00 h

Thema der Unterrichtsreihe:
Software-Entwicklung

Thema der letzten Unterrichtsstunde:
Modellierung in UML 2.0 - Klassendiagramme - ein Beispiel aus der Praxis!

Thema der heutigen Unterrichtsstunde:
Modellierung in UML 2.0 - Aggregation der Gruppenergebnisse und Überprüfung der Ergebnisse anhand geeigneter Objektdiagramme

Hausaufgaben zur heutigen Unterrichtsstunde:
Aufgabenstellung: „Einrichtung eines UML-Kurs-CVS "

Thema der folgenden Unterrichtsstunde:
Einsatz eines Software-Werkzeugs, das die Modellierung in UML und den Java-Code-Transfer ermöglicht

10 Modellierung in UML 2.0 - Aggregation der Gruppenergebnisse und Überprüfung der Ergebnisse anhand geeigneter Objektdiagramme

10.1 Voraussetzungen

Die Schüler:

- wissen, was ein Klassendiagramm ist und können Klassendiagramme aus der Praxis lesen und erläutern.
- kennen wichtige Notationselemente von Klassendiagrammen und können diese anwenden.
- wissen, was eine Klasse ist und können den Begriff der Klasse definieren.
- wissen, wie ein Klassendiagramm anhand einer Aufgabenstellung erstellt wird.
- wissen, was Methoden, Attribute und deren Typen sind und können diese definieren.
- wissen, was mit den Begriffen Kardinalität/ Multiplizität gemeint ist und können diese erläutern und definieren.
- wissen, was die wichtigen Sichtbarkeitsattribute sind und können diese in Beispielen anwenden.
- wissen, was eine Assozationsklasse ist, können den Begriff erläutern und Beispiele aus der Praxis nennen.

10.2 Unterrichtsziele

Die Schüler:

- können Klassendiagramme in einem Gesamtsystem zusammenführen.
- können systemrelevante Komponenten identifizieren und definieren.
- können Schnittstellen identifizieren und in UML 2.0 modellieren.
- können den Begriff „Objekt“ definieren und Beispiele aus der Praxis formulieren.
- können die Anwendungsbereiche von Objektdiagrammen nennen.
- können Objektdiagramme lesen und erklären.
- können Objektdiagramme selbst erstellen und definieren.
- können das modellierte Gesamtsystem mit Hilfe von Objektdiagrammen überprüfen und ggf. optimieren.

10.3 Hilfsmittel

Als Hilfsmittel werden benötigt:

- Auszug aus dem Handbuch der UML 2.0 von Christoph Kercher, 1.Auflage 2005, S.121-123
- Arbeitsblatt

10.4 Didaktisch-Methodische Reflexion

10.4.1 Didaktisch-Methodische Reflexion

Klassendiagramme werden üblicherweise in den ersten beiden Phasen der Softwareentwicklung erstellt. In den meisten Vorgehensmodellen sind dies die Analyse-/Definition- und Entwurfs-/Designphase. Klassendiagramme werden jedoch während des gesamten Softwareentwicklungsprozesses gewinnbringend eingesetzt. Anhand der Theorie soll die Praxis Gerlerntes festigen. Das Kursprojekt „Fehlzeitenverwaltung der Schüler“ soll umgesetzt werden. Fortführend zur letzten Unterrichteinheit sollen die Schüler primär die Zeit bekommen Ihre Gruppenlösungen zu präsentieren. Paralell soll dabei eine Liste aller systemrelevanten Klassen an der Tafel notiert werden. Anschließend soll Überprüfung durch ein Klassendiagramm erfolgen.

10.4.2 Rechtfertigung der didaktisch-methodischen Entscheidungen

Da bereits Grundkenntnisse der Java-Programmierung vermittelt wurden, verfügen die Schüler über eine rudimentäre Vorstellung bezüglich des Umfangs des Kursprojekts. Die Modellierung des Systems soll Klarheit schaffen und die benötigten Resourcen abstecken. Sie soll die wichtigen Aspekte der ganzheitlichen Systembetrachtung vermitteln und den notwendigen Praxisbezug herstellen.

Zur Wiederholung wird einführend der bereits in der letzten Unterrichteinheit erarbeitete Auszug(Christoph Kercher, 1.Auflage 2005). Rückblickend auf die Aufgabenstellung der letzten Stunde gibt dieser Auszug einen kurzen, umfassenden Überblick zum „Lesen und Interpretieren von Klassendiagrammen“. Gemeinsam werden die wichtigsten Punkte des Auszugs nochmals angesprochen. Um weiterführend die Zusammenfassung der unterschiedlichen Gruppenergebnisse der letzten Unterrichtseinheit zu ermöglichen, bekommen die Schüler Zeit, ihre Ergebnisse zu präsentieren und vor den Mitschülern zu begründen. Die Eingrenzung mit Fokus auf das Kursprojekt (Fehlzeitenverwaltung der Schüler) erfolgt erst jetzt, denn ein ganzheitlicher Überblick soll zuerst geschaffen werden. Nachdem alle

Gruppen im Einsatz waren, werden alle systemrelevanten Klassen aus den modellierten Gruppenergebnisse zusammen mit den Schülern extrahiert. Dabei entsteht eine Liste systemrelevanter Klassen. (Whiteboard) Um in die Notation von Objektdiagrammen einzuführen, kommt im Verlauf der Erarbeitungsphase ein weiterer Auszug (Christoph Kercher, 1.Auflage 2005) zum Einsatz. Der Auszug schildert ein Beispiel-Objektdiagramm und bietet eine Anleitung zum „Lesen eines Objektdiagramms“. Anhand des Auszugs lassen sich die Anwendungsbereiche für Objektdiagramme gemeinsam mit den Schülern ermitteln und notieren. Alle wichtigen Notationselemente sind im Beispiel enthalten, sodass gleich im Anschluss der Transfer auf das Kursprojekt erfolgen kann. Die Aufgabenstellung besteht darin, systemrelevante Objekte für das „System Schule“ zu definieren und zu modifizieren. Die Aufgabenstellung soll in Einzelarbeit bearbeitet werden. Die Schüler arbeiten dazu mit dem Arbeitblatt und einem Bleistift. Mit diesen Utensilien sollen die Einzelergebnisse dokumentiert werden. Je nach bisherigem zeitlichen Verlauf und den noch zur Verfügung stehenden zeitlichen Reserven kann dieser erste Modellierungsversuch um eine weitere Unterrichtsstunde erweitert werden. Die Schüler haben so genug Zeit herauszufinden, ob wichtige systemrelevante Klassen übersehen wurden. Diese werden ggf. in der Liste ergänzt. Am Ende der Unterrichtseinheit sollte die Aggregation der Gruppenergebnisse und Einzelergebnisse vorliegen, sodass alle Komponenten für die Modellierung des Gesamtsystems vorhanden sind. Geplant ist in den folgenden Unterrichtsstunden der Einsatz eines Software-Werkzeuges, das die Modellierung in UML und den Java-Code-Transfer ermöglicht. Sind alle systemrelevanten Komponenten detailgenau im Klassendiagramm enthalten, soll daraus das Pflichtenheft (Baseline) für das Kursprojekt abgeleitet werden.

10.5 Verlaufsplan

Verlaufsplan der Stunde am 13.09.2006, Klasse 13 LK Informatik , Heinrich-Herz-Oberschule

Phase (Beginn, Dauer)	**Inhalt**	**Unterrichtsform**	**Medien**
Einführungsphase 12.05 Uhr 35 Minuten	Begrüßung, Einbettung des Themas in die Stunde Kurze Wiederholung zu Klassendiagrammen. Dazu wird ein Auszug aus dem Handbuch der UML 2.0 von Christoph Kercher, 1.Auflage 2005, S.103-106, kurz besprochen. Präsentation der Gruppenergebnisse aus der letzten Unterrichtseinheit	Lehrer-Schüler-Gespräch	Kopien des Auszugs
Erarbeitungsphase 12.40 Uhr 30 Minuten	Gemeinsame Erstellung einer Liste aller systemrelevanten Klassen. Auszug aus dem Handbuch der UML 2.0 von Christoph Kercher, 1.Auflage 2005, S.121-123 Anwendungsbereiche von Objektdiagrammen Notation von Objektdiagrammen Einzelarbeit: Erstellung von Objektdiagrammen	Lehrer-Schüler-Gespräch und Einzelarbeit	Papier und Bleistift
Auswertungsphase 13.35 Uhr 15 Minuten	Zusammenfassend sollen ggf. die für das Gesamtsystem erstellte Klassen-Liste ergänzt werden. Die Relevanz bestimmen die Schüler demokratisch innerhalb der Kursgemeinschaft. Entscheidend ist dabei die Begründung, die der Einzelne liefert.	Lehrer-Schüler-Gespräch	Tafel
Aggregation 13.50 Uhr 10 Minuten	Abschließend sollen einige mögliche UML-Werkzeuge (Programme) mit dem Beamer vorgestellt werden. Anhand der Abwägung von Vor- und Nachteilen soll gemeinsam mit den Schülern ein geeignetes Werkzeug für die Modellierung des Kursprojektes ausgewählt werden.	Lehrer-Schüler-Gespräch	Beamer

10.6 Anhang

10.6.1 Mögliche Einzellösung

kommt noch»

10.6.2 Notation in UML - Objektdiagramme

Auszug aus dem Handbuch der UML 2.0 von Christoph Kercher, 1.Auflage 2005, S.103-106

Auszug aus dem Handbuch der UML 2.0 von Christoph Kercher, 1.Auflage 2005, S.121-123