

Einstieg in die Objektorientierung mit Unterstützung des Raspberry Pi

Masterarbeit

vorgelegt von

Heiner STROICK

Matrikelnummer:

Mail: heiner.stroick@tu-dortmund.de

Studiengang:

Lehramt an Gymnasien und Gesamtschulen
Lehramt LABG 2009
5. Fachsemester

Erstgutachter:

Prof. Dr. Johannes FISCHER

Zweitgutachter:

Dr. Arno PASTERNAK

Themenausgabe:

18. Oktober 2016

Eingereicht:

30. Januar 2017

Inhaltsverzeichnis

| | |
|--|-----------|
| 1 Einleitung | 1 |
| 1.1 Warum überhaupt Objektorientierung? | 3 |
| 1.2 Aufbau dieser Masterarbeit | 5 |
| 2 Objektorientierung und ihre Didaktik | 7 |
| 2.1 Was ist Objektorientierung? | 7 |
| 2.1.1 Aspekte der Modellierung | 8 |
| 2.1.2 Objektorientierung im Vergleich zu anderen Programmierparadigmen | 10 |
| 2.2 Wie wird Objektorientierung gelehrt? | 11 |
| 2.2.1 »Objects first«-Ansatz | 16 |
| 2.2.2 »Models first«-Ansatz | 20 |
| 2.2.3 »Objects later«-Ansatz | 22 |
| 2.2.4 »Objects first«- und »Objects later«-Ansatz im Vergleich . . | 23 |
| 2.2.5 Weitere Ansätze | 24 |
| 2.3 Probleme beim Verständnis: Fehlvorstellungen | 26 |
| 2.3.1 Was sind Fehlvorstellungen? | 27 |
| 2.3.2 Wie äußern sich Fehlvorstellungen in der OOP? | 27 |
| 2.3.3 Einige Beispiele für Fehlvorstellungen (mit Lösungsansätzen) . | 29 |
| 2.3.4 Fehlvorstellung: Verwechslung »Klasse« und »Objekt« . . . | 32 |
| 3 Die Hardware-Bausteine | 35 |
| 3.1 Idee des Hardwareeinsatzes | 35 |
| 3.1.1 Der Raspberry Pi | 36 |
| 3.1.2 Die Bauteile | 38 |
| 3.2 Erhoffte Vorteile des Hardwareeinsatzes | 40 |
| 3.3 Eventuelle Nachteile des Hardwareeinsatzes | 45 |
| 3.4 Ansteuern der Hardware am Raspberry Pi | 45 |

| | |
|---|------------|
| 4 Unterrichtserfahrungen | 49 |
| 4.1 Zielformulierungen für das Unterrichtsvorhaben | 49 |
| 4.2 Objektverständnis | 51 |
| 4.3 Unterrichtsvorbereitung | 51 |
| 4.4 Welche Projekte sollen mit der Hardware umgesetzt werden? | 52 |
| 4.4.1 Lichtanlage einer Theateraufführung | 52 |
| 4.4.2 Steuerung eines Hafenkrans | 53 |
| 4.4.3 Ausblick auf weitere Projekte | 54 |
| 4.5 Bericht der einzelnen Unterrichtsstunden | 55 |
| 4.6 Anmerkungen zur Durchführung des Unterrichts | 65 |
| 4.7 Auswertung der Unterrichtsziele | 66 |
| 5 Auswertung: Hardwareeinsatz | 67 |
| 5.1 Informationen zu den Interviews | 67 |
| 5.2 Interviewleitfaden | 68 |
| 5.3 Auswertung der Interviews | 69 |
| 5.3.1 Hauptkategorienbildung | 69 |
| 5.3.2 Unterkategorienbildung | 70 |
| 5.3.3 Gezeigtes positives Verständnis von Objektorientierung . . . | 72 |
| 5.3.4 Gezeigtes negatives Verständnis von Objektorientierung . . | 74 |
| 6 Fazit | 79 |
| 6.1 Vergleich zu klassischen Herangehensweisen | 79 |
| 6.2 Probleme und Vorteile des Hardwareeinsatzes | 80 |
| 6.3 Verbindung zu Fehlvorstellungen | 85 |
| 6.4 Weitere Anmerkungen | 86 |
| 6.5 Abschlussfazit | 89 |
| Literaturverzeichnis | 91 |
| Internetquellenverzeichnis | 95 |
| A Unterrichtsmaterialien | 101 |
| A.1 Arbeitsblätter | 105 |
| A.2 Ausgewählte Schülerlösungen | 131 |
| A.2.1 Kandidaten für Objekte zum Theaterprojekt | 132 |
| A.2.2 Objektkarten zum Theaterprojekt | 133 |
| A.2.3 Objektkarten des Objektspiels | 134 |

| | | |
|----------|---|------------|
| A.2.4 | Protokoll zum Objektspiel | 136 |
| A.2.5 | Sequenzdiagramm zum Objektspiel (Besprechung Tafel) | 137 |
| A.2.6 | Sequenzdiagramm zum Objektspiel | 138 |
| A.2.7 | Quelltext Theaterprojekt | 139 |
| A.2.8 | Objektspiel mit Objektbeziehungen | 142 |
| A.2.9 | Übersicht zu Objektbeziehungen | 143 |
| A.2.10 | Sequenzdiagramme zum Schalten der Hintergrundbeleuchtung | 144 |
| A.2.11 | Quelltexte zum Schalten der Hintergrundbeleuchtung | 145 |
| B | Materialen zu den Interviews | 147 |
| B.1 | Leitfaden zum Interview | 151 |
| B.2 | Transkriptionsregeln | 152 |
| B.3 | Interview A | 153 |
| B.4 | Interview B | 160 |
| B.5 | Interview C | 164 |
| B.6 | Interview D | 171 |
| B.7 | Interview E | 177 |
| B.8 | Interview F | 182 |
| B.9 | Auswertung der Interviews nach Haupt- und Unterkategorien | 188 |
| C | Bauteile der rpCollection | 223 |
| C.1 | Bauteil 1: Diode (LED) | 224 |
| C.2 | Bauteil 2: Phototransistor | 225 |
| C.3 | Bauteil 3: RGB-LED | 226 |
| C.4 | Bauteil 4: Taster | 227 |
| C.5 | Bauteil 5: Analog-Digital-Wandler und Regler | 228 |
| C.6 | Bauteil 6: Motor | 232 |
| C.7 | Bauteil 7: Summer | 234 |
| C.8 | Fotos der angeschlossenen Bausteine | 235 |
| D | Verwendete Software | 237 |
| D.1 | BlueJ | 237 |
| D.2 | Geany | 237 |
| D.3 | Pi4J | 238 |
| D.3.1 | Pin-Nummerierungen | 238 |
| D.3.2 | Belegte Pinne freigeben | 239 |
| D.4 | Wiring Pi | 240 |
| D.5 | Groovy | 240 |

| | | |
|----------|---|------------|
| D.5.1 | Die Groovy-Shell (<code>groovysh</code>) | 241 |
| D.5.2 | Die Groovy-Console (<code>groovyConsole</code>) | 241 |
| E | Materialien der rpCollection | 243 |

Danke

Mein Dank gilt Johannes PIEPER für die Betreuung am Joseph-König-Gymnasium in Haltern am See und Schulleiter OStD Ulrich WESSEL für die Erlaubnis, die Interviews durchzuführen. Den Schülerinnen und Schülern danke ich für die Zusammenarbeit und die Teilnahme an den Interviews.

Dortmund, im Januar 2017.

Lizenz

Die Masterarbeit »Einstieg in die Objektorientierung mit Unterstützung des Raspberry Pi« von Heiner STROICK steht unter einer »Namensnennung – Nicht-kommerziell – Weitergabe unter gleichen Bedingungen 4.0 International«-Lizenz.

Die Bedingungen der Lizenz können unter folgendem Link eingesehen werden:

<http://creativecommons.org/licenses/by-nc-sa/4.0/deed.de>



Kapitel 1

Einleitung

In einer Welt, in der die Digitalisierung unaufhaltsam voranschreitet und gar nicht mehr wegzudenken ist, ist auch ein sicherer Umgang mit dem Rechner unerlässlich. Es ist eine Selbstverständlichkeit geworden, ein digitales Leben zu führen und mit der Informatik jeden Tag in Kontakt zu stehen, auch wenn es die meisten Menschen nicht als »Informatik« wahrnehmen.

Neben der Diskussion über die Einführung einer informatischen Grundbildung für alle (Informatik als Pflichtfach) im Hinblick auf gesellschaftlich relevante Themen wie Datenschutz und Post-Privacy (Kontrollverlust im Digitalen), sollten Schülerinnen und Schüler in ihrer Schullaufbahn mit den grundsätzlichen informatischen Konzepten vertraut gemacht werden, um sie für die Arbeits- und Lebenswelt nach der Schule vorzubereiten. Dies wird nicht nur vom Lehrplan, sondern von auch von der Industrie und Gesellschaftsverbänden wie bspw. der GI (Gesellschaft für Informatik) gefordert (vgl. [GI Ziele u. Aufgaben]). Darüber hinaus zählt dies auch zu den Aufgaben und Zielen des Faches Informatik (vgl. [Schulentwicklung NRW 2014, Abs. 3]).

Einen Teil der Informatik macht dabei die Software-Entwicklung aus. Ziel ist die Entwicklung von Programmen / Programmabläufen, die den Rechner anweisen, eine bestimmte Aufgabe zu erfüllen. Die Software-Entwicklung mit ihren Stufen des Entwurfes (Analyse), der Umsetzung (Implementierung und Testen) und der Evaluation (Funktions- und Leistungsüberprüfung) findet in Grundzügen für kleinere Projekte auch in der Schule statt (dieses Modell wird häufig als »Wasserfallmodell« bezeichnet, vgl. [Schubert Schwill 2011, S. 67f]).

Der Einstieg in die objektorientierte Programmierung (OOP) erfolgt in der Schule meist nur am Rechner. Zu Beginn wird für die objektorientierte Modellierung (OOM) oft ein größeres Projekt ausgewählt, wie ein Patienten-System oder ein

Banken-Programm, das zur Verwaltung von Patienten- bzw. Kundendaten dient. Es werden Objekte erstellt, Klassen verändert, Objekte an andere Objekte übergeben, Methoden aufgerufen und Attributwerte geändert. Wenn auch aus dem Alltag mehr oder weniger bekannt, sind sämtliche Szenarien dabei meist nur virtuell erfahrbar.

Diese Masterarbeit geht einen anderen Weg der Einführung der Objektorientierung. Für den Unterricht wurden kleine Hardware-Bausteine vorbereitet, die die Aufgabe haben, ein Objekt zu repräsentieren. Mit diesen einfachen Bausteinen lassen sich komplexere Sachzusammenhänge modellieren, mit dem Unterschied, dass Objekte jetzt tatsächlich angefasst und buchstäblich beobachtet werden können.

Dabei kommt der Raspberry Pi zum Einsatz, der einerseits durch seinen günstigen Preis für die Schule interessant ist, andererseits durch seine vielen Einsatzzwecke zu überzeugen weiß. So bietet er die einfache Möglichkeit, ein Steckbrett anzuschließen und die Pinne eines GPIO (Grand Purpose Input Output) zu schalten. Für den Unterricht bedeutet dies, dass z. B. Bauteile wie LEDs, Taster, Phototransistoren oder Motoren-Steuerungen im Unterricht mit dem Raspberry Pi gesteuert werden können. Damit kann die Objektorientierung wirklich greifbar gemacht werden. Hinzu kommt, dass der Zustand eines Objekts bei den Hardware-Bausteinen auch von außen ersichtlich ist, ohne einen Methodenaufruf zu initiieren. Auch die Auswirkungen bei der Änderung eines Attributwerts sind dabei direkt zu beobachten.

Die Schülerinnen und Schüler der Einführungsphase (10. Klasse) des Joseph-König-Gymnasiums in Haltern am See haben mit ihrem Informatiklehrer Johannes PIEPER erstmals mit den Hardware-Bausteinen gearbeitet. Im Unterricht wird ein Szenario modelliert, das unterschiedliche Hardware-Bausteine einsetzt und dabei die Programmiersprache Java verwendet.¹

Diese Masterarbeit hat das Ziel, neben einer Betrachtung und Darstellung einer klassischen Herangehensweise an die OOM / OOP, den neuen Weg vorzustellen und Stärken und Schwächen aufzuzeigen. Dem Leser² dieser Arbeit soll von den Erfahrungen aus dem Unterricht berichtet und gezeigt werden, wie sich der Ansatz umsetzen und noch verbessern lässt. Dabei sollen drei Hauptfragen beantworten werden:

1. Welche Fehlvorstellungen äußern sich beim klassischen Ansatz und wie können diese Fehlvorstellungen behandelt werden?

¹ Beim INFORMATIK-TAG NRW 2017 wird es einen Workshop zu dem hier beschriebenen Ansatz geben, auch mit der Möglichkeit, selbst einmal mit den Hardware-Bausteinen zu arbeiten.

² Aus Gründen der besseren Lesbarkeit wird an einigen Stellen auf die gleichzeitige Verwendung männlicher und weiblicher Sprachformen verzichtet. Sämtliche Personenbezeichnungen gelten gleichermaßen für beiderlei Geschlecht.

2. Bietet der oben beschriebene Hardwareeinsatz eine praktikable Möglichkeit für den Unterricht, die objektorientierte Modellierung einzuführen und dabei Begriffe wie »Objekt«, »Attribut«, »Attributwert« und »Zustand« (evtl. noch weitere) kennenzulernen und erfahrbar zu machen?
3. Welche Fehlvorstellungen des klassischen Ansatzes lassen sich durch den Hardwareeinsatz vermeiden? Birgt der Ansatz dieser Masterarbeit evtl. auch die Gefahr, andere / neue Fehlvorstellungen zu wecken?

Neben einer Begleitung des Unterrichts werden am Ende des Unterrichtsvorhabens Interviews mit Schülerinnen und Schülern geführt, die dann durch die inhaltlich-strukturierende, qualitative Inhaltsanalyse nach KUCKARTZ ausgewertet werden (vgl. [Kuckartz 2016, Kap. 5, S. 97ff]). Die gewonnenen Ergebnisse werden dazu genutzt, den vorgestellten Ansatz zu überarbeiten und die Fragen zu beantworten.

1.1 Warum überhaupt Objektorientierung?

Eine Möglichkeit, einen Inhalt des Informatikunterrichts zu rechtfertigen, ist die Argumentation als »fundamentale Idee«. Eine fundamentale Idee bzgl. eines Gegenstandsbereichs ist nach [Schubert Schwill 2011, S. 64f] ein „Denk-, Handlungs-, Beschreibungs- oder Erklärungsschema“ (vgl. ebd. S. 64), das fünf Kriterien genügt, die eine Rechtfertigung der Idee begründen.

Dass es sich bei der Objektorientierung um eine fundamentale Idee handelt, gilt heute als unumstritten. Die fünf Kriterien nach [Schubert Schwill 2011, S. 64f], an denen man dieses festmachen kann, sind alle erfüllt.

1. Horizontalkriterium

Die OOP findet heutzutage in vielfältigen Gebieten und Industrien Anwendung. Sie wird als Konzept in verschiedenen Programmiersprachen genutzt (Java, C++) und tritt in mehreren Variationen auf. Neben klassischen Computerprogrammen lassen sich z. B. auch Apps für Smartphones mit objektorientierten Sprachen entwickeln. Heutzutage wird die Objektorientierung als klassisches Programmierparadigma gehandelt.

Schon 1994 schreibt LUKER die folgenden Worte zur Verbreitung des Begriffs »Objektorientierung«:

Today, it is impossible to avoid the term [object-oriented], it lappers in nearly every book, journal and magazine which relates to computer software,

and yet, while they rush to embrace it, many computer professionals and educators still do not understand what object orientation means.

[Luker 1994, S. 56]

Auch wenn er darstellt, dass es viel Unwissenheit um den Begriff gebe, war zum damaligen Zeitpunkt schon die Tragweite der Idee erkennbar.

2. Vertikalkriterium

Die OOP lässt sich auch auf vielfältigen intellektuellen Niveaus erleben und vermitteln. Die Grundkonzepte der Objektorientierung können schon von Programmieranfängern erlernt werden. Dafür gibt es extra Programme, die (u. a. Kindern) den Einstieg erleichtern sollen. Ein solches Programm ist z. B. Greenfoot³, das spielerisch in die Objektorientierung einführt.

3. Zielkriterium

Das (idealisierte) Ziel der Objektorientierung ist natürlich die Erstellung von Software, die im besten Fall qualitativ hochwertig ist und den Punkten der Anforderungsanalyse gerecht wird. Zieht man in Betracht, in welcher Weise sich die Objektorientierung verbreitet hat (jedes Smartphone kann objektorientierte Apps ausführen), lassen sich Teile dieses idealisierten Ziels sicherlich auch im Unterricht erreichen.

4. Zeitkriterium

Die Objektorientierung ist ein altes informatisches Konzept, welches aller Voraussicht nach auch in der Zukunft noch relevant bleiben wird. Die Objektorientierung hat ihre Anfänge in den 60er Jahren und hat sich über die Jahre bewährt (s. Kapitel 2.1).

5. Sinnkriterium

Der „Bezug zu Sprache und Denken des Alltags und der Lebenswelt“ [Schubert Schwill 2011, S. 65] ist bei der Objektorientierung gegeben, da es sich einerseits um eine bestimmte Art und Weise des Aufbaus einer Programmiersprache handelt (Sprachaspekt). Andererseits ist der Blick für Objekte bzw. das Einsteilen in Objekte etwas, dass auch der Alltagswelt entstammt (Zerlegen eines komplexen Sachverhalts in seine Einzelteile).

Eine andere Möglichkeit, die Objektorientierung für die Schule zu rechtfertigen, liefert ein Blick in die gymnasialen Aufgaben und Ziele des Faches Informatik. So

³ Greenfoot – <http://www.greenfoot.org/>.

steht auf der Internetseite des LANDESINSTITUTS FÜR SCHULE Folgendes, das so auch auf die Objektorientierung zutrifft:

Der Informatikunterricht der gymnasialen Oberstufe geht deutlich über eine Grundbildung im Bereich der Informations- und Kommunikationstechnologien in der Sekundarstufe I hinaus. Die Schülerinnen und Schüler erwerben [...] Fähigkeiten zur kritischen und verantwortungsvollen Analyse, Modellierung und Implementierung komplexer Informatiksysteme. Dabei konzentriert sich der Unterricht stets auf fundamentale und zeitbeständige informatische Ideen, Konzepte und Methoden und schließt auch die Auseinandersetzung mit Fragen einer menschengerechten Gestaltung und der Sicherheit von Systemen sowie der Folgen und Wirkungen des Einsatzes von Informatiksystemen ein. Schülerinnen und Schüler werden so befähigt und motiviert, auch zukünftige Entwicklungen zu nutzen, zu verstehen, hinsichtlich ihrer Wirkungen zu beurteilen und sich aktiv an der Fortentwicklung zu beteiligen.

[Schulentwicklung NRW 2014, Abs. 4]

Es ist erkennbar, dass sich viele Punkte der fundamentalen Idee »Objektorientierung« auch in dem Zitat wiederfinden lassen („Modellierung und Implementierung komplexer Informatiksysteme“, „zeitbeständige informatische Ideen“, „menschengerechte[-] Gestaltung“). Neben den klassischen schulischen Aufgaben, die Schülerinnen und Schüler zu selbstständigen und verantwortungsvoll handelnden Menschen zu erziehen (vgl. [SchulG NRW 2016, S. 2]), gibt es darüber hinaus auch noch abseits der Vorgaben für das Abitur (vgl. [Abiturvorgaben NRW]) gute (innerinformatische) Gründe, die Objektorientierung im Unterricht zu behandeln.

1.2 Aufbau dieser Masterarbeit

In Kapitel 2 werden didaktische Theorien zur Lehre der Objektorientierung dargestellt und von den klassischen Ansätzen berichtet, Anfängern das Programmieren zu lehren. Darüber hinaus werden dabei auftretende Fehlvorstellungen ausführlich dargestellt und Lösungsvorschläge unterbreitet.

Die Hardware-Bausteine sind Thema des 3. Kapitels. Es wird erklärt, welche Vorteile der Hardware-Ansatz unter Umständen bietet. Die Bausteine werden kurz dargestellt und es wird erklärt, welche Voraussetzungen an Hard- und Software gestellt werden müssen und wie die Bauteile allgemein zu benutzen sind.

In Kapitel 4 werden dann die Ziele des Hardwareeinsatzes für den Unterricht formuliert und von ebendiesem berichtet. Der Unterricht wird in Kapitel 5 durch die Auswertung von Schülerinterviews bzgl. des Hardwareeinsatzes und des Lernfortschritts evaluiert.

Abschließend werden in Kapitel 6 alle Erkenntnisse aus dem Unterricht und den Interviews aufgegriffen und ein Fazit gezogen. Des Weiteren werden Verbesserungsvorschläge für die Hardware-Bausteine und den neuen Ansatz aufgezeigt, die sich aus der Auswertung ergeben haben.

Sämtliche Unterrichtsmaterialien, Interviewtranskripte, Erklärungen zur Hard- und Software sowie der Zugang zu den Quelltexten sind im Anhang zu finden.

Kapitel 2

Objektorientierung und ihre Didaktik

In diesem Kapitel soll ein kleiner Abriss zur objektorientierten Programmierung (OOP) gegeben werden. Wodurch zeichnet sich die OOP aus und wo sind Unterschiede zur objektorientierten Modellierung (OOM)? Wo liegen für Schülerinnen und Schüler die Schwierigkeiten – und ist OOP überhaupt schwieriger als andere Programmierparadigmen?

Darüber hinaus soll auch auf den weit verbreiteten »Objects first«-Lehransatz nach KÖLLING und ROSENBERG eingegangen werden. Neben diesem Ansatz gibt es auch noch die Idee, Objekte erst später im Lernprozess einzuführen (»Objects later«). Auch dieser Ansatz wird dargestellt und ein Vergleich gezogen. Daneben werden noch andere Ansätze zur Lehre präsentiert.

In diesem Kapitel werden auch einige Fehlvorstellungen der Objektorientierung beleuchtet, die insbesondere bei Programmieranfängern auftreten. Im Fokus steht dabei besonders die Verwechslung bzw. Unkenntnis darüber, was Klassen von Objekten unterscheidet (Kapitel 2.3.4). Ein Anliegen dieser Arbeit ist es auch, Vorschläge zum Auflösen von Fehlvorstellungen zu unterbreiten.

2.1 Was ist Objektorientierung?

Bei der Objektorientierung kommt eine besondere Weltsicht zum Tragen, die die Welt in sog. »Objekte« einteilt. Dabei habe die Objektorientierung in den letzten Jahren vor allem deshalb an Zulauf gewonnen, weil es einer natürlichen Weltsicht entspreche (vgl. [Uysal 2012, S. 816] und bspw. auch [Schmolitzky 2006, S. 32]). UYSAL beschreibt dies mit den folgenden Worten:

It is fair to say that object-oriented programming (OOP) has gained much interest while it is becoming a common programming paradigm. This is partly because it is similar to our view of the real world.

[Uysal 2012, S. 816]

Das Programmierparadigma der Objektorientierung hat seine Ursprünge Anfang der 60er Jahre mit der Entwicklung der Sprache »Simula« (vgl. [Lewis 2000, S. 245]). Ziel war es damals, Modelle für diskrete Simulationen zu programmieren, indem „[d]ie Objekte der realen Welt [...] durch Eigenschaften und Fähigkeiten modelliert“ (vgl. [Engbring 2009, S. 31]) wurden. Schon damals wurde die Grundlage für heutige Konzepte der OOP gelegt (vgl. ebd. S. 31, vgl. auch [Armstrong 2006, S. 123f] [Lewis 2000, S. 245]).

XEROX PARC hat im Anschluss mit der Entwicklung von »Smalltalk« zur Verbreitung der Objektorientierung beigetragen. Smalltalk demonstrierte die Mächtigkeit eines graphischen User-Interfaces (GUI) und ist durchgängig objektorientiert aufgebaut, da sämtliche Kommunikation zwischen Objekten via message-passing funktionierte (vgl. [Lewis 2000, S. 246] [Engbring 2009, S. 31]). Message-passing wird hier als Prozess verstanden, bei dem ein Objekt Daten an ein anderes sendet oder ein anderes Objekt anfragt, eine Methode auszuführen (vgl. [Armstrong 2006, S. 126]). Smalltalk hat damals den Grundstein für heutige Benutzeroberflächen gelegt. In [Engbring 2009] findet man einen kleinen geschichtlichen Abriss, der die Objektorientierung in die Geschichte der Informatik einordnet.

Heutzutage wird die Objektorientierung als mächtiges Programmierparadigma verstanden und ist nicht mehr wegzudenken. Im Folgenden soll kurz auf die wichtigsten Aspekte im Modellierungsprozess eingegangen werden. ARMSTRONG bezeichnet die fundamentalen Konzepte als „Quarks“ der OOP (vgl. [Armstrong 2006, S. 123]). In der Studie werden »Vererbung«, »Objekt«, »Klasse«, »Kapselung«, »Methode«, »message-Passing«, »Polymorphie« und »Abstraktion« als Quarks identifiziert. An dieser Stelle sollen diese Begriffe nicht weiter definiert werden, da eine genaue Definition für diese Arbeit nicht von Relevanz ist. In der Schule werden diese und damit zusammenhängende Begriffe (wie Attribut, Geheimnisprinzip, Beziehung etc.) allerdings besprochen, sodass davon ausgegangen werden kann, dass der Leser diese kennt.

2.1.1 Aspekte der Modellierung

Modellierung heißt nicht, die Programmierumgebung gut zu kennen oder die Programmiersprache zu beherrschen. Vielmehr geht es um das Design bzw. die Modelerstellung einer Software. Dieser anfängliche Schritt in der Software-Entwicklung

ist auch für Studierende mit Schwierigkeiten verbunden (vgl. [Thomasson Ratcliffe et. al. 2006, S. 2]). OR-BACH und LAVY beschreiben das Programmieren an sich wie folgt:

Programming is not an isolated technical skill; it is a complex cognitive activity associated with a programming paradigm.

[Or-Bach Lavy 2004, S. 82]

Ein Aspekt ist die »mentale Abstraktion«, die geleistet werden muss, wenn ein etwas umfangreicheres Programm erstellt werden soll. Dies kommt vor allem bei der Identifikation von Objekten zum Tragen und der daraus folgenden Notwendigkeit, passende Klassen und korrekte Klassenbeziehungen zu modellieren („Defining a class with its relevant attributes and methods is the basic required abstraction.“ [Or-Bach Lavy 2004, S. 83]). THOMASSON et. al. konnten in ihrer Forschung (neben anderen Fehlvorstellungen) z. B. oft fehlende Beziehungen zwischen Klassen identifizieren („non-referenced Class fault“ [Thomasson Ratcliffe et. al. 2006, S. 3, 5]).

ENGBRING beschreibt diesen Prozess wie folgt und misst den algorithmischen Abläufen (sprich den Abläufen der Methoden) erstmal keine Bedeutung bei:

In der Phase der Analyse können Objekte zunächst identifiziert und dann außerdem die Beziehungen der Objekte untereinander modelliert werden. Dies ist eine sehr einfache Abstraktionsleistung, für die man über die Verarbeitungsprozesse im Computer nichts wissen muss. Die algorithmischen Abläufe im Computer können und sollen auf dieser Ebene ein ‚schwarzer Kasten‘ bleiben. Es kann deklarativ gearbeitet werden, ohne dass man sich um die Details der Implementierung kümmert.

[Engbring 2009, S. 30f]

Trotzdem wird in [Kortenkamp Modrow et. al. 2009, S. 44] darauf hingewiesen, dass ein reines »Abilden« von real existierenden Strukturen nicht ausreichend sei, um Problemstellungen zu modellieren. So machen KORTENKAMP et. al. auf die Ablehnung dieser Theorie mit den folgenden Worten aufmerksam:

Grundsätzlich entspricht die Überzeugung, man könne die richtige Klassenstruktur einfach von einer realen Situation ‚ablesen‘, einer Abbildtheorie, [...] die heute auf breiter Front von Kognitivisten bis zu Konstruktivistern abgelehnt wird.

[Kortenkamp Modrow et. al. 2009, S. 44]

Auch das Modellieren von Vererbungsbeziehungen kann Probleme bereiten (vgl. [Or-Bach Lavy 2004, S. 82f]). Stichpunkte sind hier Unterklassen und abstrakte

Klassen. Darüber hinaus muss richtig entschieden werden, bestimmte Anforderungen an die Software entweder als Attribut oder Methode einer Klasse umzusetzen. OR-BACH und LAVY konnten beobachten, dass Studierende die Länge eines Beschäftigungsverhältnisses einer Klasse `employee` als Attribut umgesetzt hatten, wobei eine Methode natürlich die sinnvollere Alternative gewesen wäre.

Ein Ansatz, solche Probleme zu vermeiden oder auch aufzuzeigen, ist natürlich die Wahl geeigneter und damit auch genügend komplexer Beispiele (s. auch Kapitel 2.2.2 und [Alphonse Ventura 2002, S. 73]). Darüber hinaus muss den Lehrenden klar sein, ob der Fokus mehr auf dem Modellieren (dem Design) oder der Umsetzung liegt, um schon gezielt auf Fehler im Modellierungsprozess hinweisen zu können oder dies erst während der Implementierung zu tun. Insgesamt stellen die in diesem Kapitel zitierten Autoren aber alle das fest, was OR-BACH und LAVY auf den Punkt bringen:

Object oriented programming involves object-oriented design, modeling capabilities, identification of relations between the world and the program objects, along with abstraction capabilities. These are always considered higher order cognitive skills.

[Or-Bach Lavy 2004, S. 84]

2.1.2 Objektorientierung im Vergleich zu anderen Programmierparadigmen

Das Paradigma der Objektorientierung ist neben dem imperativen und dem deklarativen Paradigma (v. a. prozedurale bzw. funktionale Programmierung) wohl eines der bekanntesten¹. Wie schon in der Einleitung mit Hilfe der funktionalen Ideen die Bedeutung der Objektorientierung unterstrichen wurde, spricht in der Praxis tatsächlich die weite Verbreitung und potentielle Erreichbarkeit von Milliarden Endgeräten, die Objektorientierung »verstehen« (hauptsächlich Geräte, die Java unterstützen), für dieses Programmierparadigma. „Java ist inzwischen eine relevante Sprache in der kommerziellen Informationsverarbeitung“ [Schmolitzky 2006, S. 32].

Natürlich stellt dies aber noch keine Rechtfertigung für den Einsatz in der Schule dar. BERGIN konstatiert, dass eine prozedurale Sprache nicht unbedingt schwieriger zu erlernen sei als eine objektorientierte:

There is nothing especially complex about OOP, any more than there is anything complex about procedural programming. It's just that the world looks completely different in the two paradigms.

[Bergin 2000]

¹ Darüber hinaus gibt es natürlich noch weitere Paradigmen. Auf eine weitere Darstellung wird hier verzichtet.

HOLLAND, GRIFFITHS und WOODMAN halten dabei zum Thema Objektorientierung fest, dass es gerade am Anfang viele Aspekte bei der Modellierung gebe, die zu diskutieren seien und an denen Studierende (sicherlich auch Schülerinnen und Schüler) Fehlvorstellungen entwickeln könnten. Sie verweisen dabei auf den Arbeitsaufwand zu Beginn, wie folgendes Zitat belegt:

Object concepts are often taught, especially in the first few lessons, with a great deal of practical demonstration during lectures, and with a lot of expert help on hand for lab work. This is not because object concepts are intrinsically difficult, but because the subject does offer many opportunities, especially in the early stages, for students to develop misconceptions, which can be hard to shift later.

[Holland Griffiths et. al. 1997, S. 131]

Dabei halten die Autoren noch einmal fest, wie schwierig es sein kann, diese Fehlvorstellungen im späteren Verlauf des Lernens zu korrigieren. Wie im vorangegangenen Kapitel dargelegt, kann auch der Umfang der nötigen Abstraktion den Lernprozess erschweren.

2.2 Wie wird Objektorientierung gelehrt?

Es gibt verschiedene Ansätze, die sich in der Didaktik etabliert haben, wie Objektorientierung unterrichtet werden soll. Da in den Vorgaben für das Abitur und den Lehrplänen die Objektorientierung mittlerweile vorgeschrieben ist (vgl. [Kernlehrplan Informatik 2014, S. 22, 27f, 33] [Abiturvorgaben NRW]), wird auch diskutiert, welche (prozedurale) Sprache neben der objektorientierten (meist Java) unterrichtet werden soll und in welcher Reihenfolge.

Die hier vorgestellten Ansätze werden in vielen Veröffentlichungen diskutiert und sind weit verbreitet. Forschungsgegenstand vieler wissenschaftlicher Publikationen ist vor allem der »Objects first«-Ansatz, häufig unter Verwendung von BlueJ².

Wozu objektorientiertes Programmieren? Eine Antwort auf diese Frage ist natürlich der Zwang durch den Lehrplan, in dem die Behandlung von Klassen und Objekten im Unterricht vorgeschrieben wird – sowohl in der Einführungs- als auch in der Qualifikationsphase (vgl. [Kernlehrplan Informatik 2014, S. 22, 27f, 33]). Darüber hinaus kann der Unterrichtsgegenstand »Objektorientierung« aber auch wie folgt begründet werden.

² BlueJ – <http://bluej.org/>. Siehe auch Anhang D.1.

Neben der Rechtfertigung in der Einleitung findet ENGBRING in [Engbring 2009] Argumente aus der Geschichte der Informatik für das objektorientierte Programmieren. So bieten objektorientierte Sprachen die Möglichkeit, sich computerunabhängig Problemstellungen zu widmen und sich nicht um die Übersetzung in Maschinensprache zu kümmern. Hierfür stehen Compiler und Interpreter bereit. Die Objektorientierung kann somit als Arbeitserleichterung gesehen werden (vgl. ebd. S. 28).

Eine weitere Begründung für den Vormarsch der OOP sieht ENGBRING in der Möglichkeit, „universelle Bausteine, ähnlich wie z.B. in der Elektrotechnik, zu haben, aus denen Programme zusammengesetzt werden können“ (ebd. S. 31). Daneben sieht er im „Abrücken von der Konsolen-Programmierung und der Hinwendung zu grafischen Benutzungsoberflächen“ (ebd. S. 31) einen weiteren Grund für die OOP, da auch die Benutzungsoberflächen „in Form von Bausteinen“ (ebd. S. 31) aufgebaut seien. Zudem lasse sich ein „nach Objekten zerlegtes Programmsystem“ (ebd. S. 32) einfacher einer Revision unterziehen, da Objekte unabhängig vom Rest des Systems geändert werden könnten, vorausgesetzt, die Beziehungen zwischen den Objekten wurden richtig erkannt (vgl. ebd. S. 32).

Im Folgenden sollen einige Punkte aufgezählt und erläutert werden, die diskutabel erscheinen, wenn in der Schule über OOP und deren Berechtigung für den Unterricht gesprochen wird.

Der Zeitpunkt im Unterricht Nach BERGIN sollte mit der Objektorientierung begonnen und danach (die) prozedurale Sprache(n) unterrichtet werden, da sich in der Software-Industrie gezeigt habe, dass die Zeitspanne, in der sich (erfahrene) prozedurale Programmierer an die Objektorientierung gewöhnt hätten, sehr lang sein könne (vgl. [Bergin 2000]). Dafür verweist er auf STROUSTRUP³, der eine Zeitspanne von 18 Monaten für diesen mentalen Wechsel nennt. Ebenfalls verweist er auf LATTANZI und HENRY⁴, die auch Schwierigkeiten beim Unterrichten von Objektorientierung festgestellt haben, wenn Schülerinnen und Schüler vorher schon Kontakt mit dem prozeduralen Paradigma hatten.

Für die Schule sieht BERGIN damit die Gefahr, dass – wenn einmal prozedurale Programmierung unterrichtet wurde – die Schülerinnen und Schüler immer wieder in diesen Modus wechseln. Er schreibt Folgendes über Schülerinnen und Schüler, die gerade die Objektorientierung lernen:

³ Bjarne STROUSTRUP (1994): *The Design and Evolution of C++*. Addison Wesley. S. 172.

⁴ Mark LATTANZI, Sallie HENRY (1996): *Teaching the Object-Oriented Paradigm and Software Reuse*. Computer Science Education, V7, N1, S. 99.

While the programmers are in this learning mode, they will naturally try to solve problems by decomposing functions and not by discovering objects. Whenever the going gets hard, they will fall back on what they know best—procedural programming.

[Bergin 2000]

Auf der anderen Seite gibt es aber auch Studien, die keinen oder nur einen geringen Unterschied messen. Dazu sei an dieser Stelle auf das Kapitel 2.2.4 verwiesen, in der zwei Studien vorgestellt werden, die den »Objects first«- und »Objects later«-Ansatz vergleichen. Auch wenn der »Objects later«-Ansatz nicht allumfassend dem prozeduralen Paradigma unterzuordnen ist, lassen sich doch gerade am Anfang viele Parallelen ziehen.

Die Wahl der Sprache KÖLLING, KOCH und ROSENBERG haben schon 1995 Anforderungen an eine objektorientierte Sprache aufgestellt (vgl. [Kölling Koch et. al. 1995, S. 2]). Damals war Java noch nicht erfunden, aber die Autoren haben schon damals folgende wichtige Punkte identifiziert (Auswahl):

- Die Sprache sollte klare, einfache und wohldefinierte Konzepte aufweisen (bspw. das Geheimnisprinzip, Vererbung etc. Vgl. [Kölling Koch et. al. 1995, S. 2]).
- Die Sprache sollte *echt* objektorientiert sein (eine objektorientierte Umsetzung eines Programms sollte keine Option einer Sprache sein).

Die Autoren geben trotzdem den Hinweis, dass Kontrollstrukturen nicht als Objekte verstanden werden sollen, wie es bspw. in Smalltalk der Fall ist (vgl. ebd. S. 4).

- Die Sprache sollte statisch getypt sein (einfacher für die Fehlerbehebung im Code, da so unter Umständen schneller die Fehlerquelle gefunden werden kann, vgl. auch [Kölling I 1999, S. 5]).
- Die Sprache sollte keine Konstrukte beinhalten, die sich zwingend auf die ausführende Maschine beziehen (Forderung einer höheren Sprache). KÖLLING schreibt dazu:

Putting the responsibility for storage management into the hands of a programmer (especially if it is a beginning student) is unnecessary and leads to frustrating experiences.

[Kölling I 1999, S. 5]

- Die Sprache sollte eine einfach zu lesende und konsistente Syntax haben. Dies mache das Unterrichten einfacher (vgl. auch [Kölling I 1999, S. 5]).

- Die Sprache sollte eine einfache Programmierumgebung besitzen, in der auch Möglichkeiten zum Debuggen von Code gegeben werden.
- Die Sprache sollte ein Anknüpfen an andere Sprachen ermöglichen.
- Die Sprache kann ruhig langsamer als andere bzw. die Programmiersprachen in der Industrie sein, sofern sich mit ihr die Konzepte der Objektorientierung besser vermitteln lassen.

KÖLLING führt diese Punkte in [Kölling I 1999] weiter aus. Er stellt dabei heraus, dass solch eine Sprache nicht per se gut oder schlecht sei, es ihm aber vor allem um den didaktischen Mehrwert einer solchen Sprache gehe (s. auch letzter Punkt u. vgl. [Kölling I 1999, S. 3]).

Während im obigen Artikel aus dem Jahr 1996 von Java noch keine Rede ist, wird Java in [Kölling I 1999] an obigen Punkten gemessen. KÖLLING stellt dabei fest, dass Java vielen geforderten Punkten gerecht werde (Klarheit, Mehrfach-Vererbung), aber auch einige kleinere Inkonsistenzen aufweise (bspw. den Typ `boolean` und die Klasse `Boolean`, vgl. ebd. S. 13). Ferner sei die Notwendigkeit von Typ-Castings an einigen Stellen in Java ein Kritikpunkt (vgl. ebd. S. 14), ebenso wie die Syntax. Diese erinnere an C bzw. C++ und sei sehr knapp gehalten (die Syntax von C++ wird als „overly terse“ beschrieben, ebd. S. 10), welches die Lesbarkeit erschwere (vgl. ebd. S. 10).

Darüber hinaus stellt die `main`-Methode einen Kritikpunkt dar, da ihre Methodensignatur viele Punkte aufweist, die für Anfänger unverständlich seien (vgl. ebd. S. 13f). Dies wird auch in Kapitel 2.2.1 diskutiert.

Die Programmierumgebung Der Wunsch nach einer einfach zu benutzenden Programmierumgebung wurde schon bei der Auswahl der Sprache deutlich:

A well designed language is only half of what we need and is rendered useless by the absence of a suitable environment.

[Kölling I 1999, S. 14]

KÖLLING fordert von der Programmierumgebung sieben Punkte, die hier kurz erwähnt werden sollen (vgl. [Kölling II 1999, S. 2]). Über allem steht aber die Anforderung, Objektorientierung zu unterstützen und für die Lehre geeignet zu sein.

1. **Ease of use** (die Umgebung muss für Anfänger einfach zu benutzen sein und sollte eine graphische Benutzeroberfläche bieten, vgl. ebd. S. 2f)

2. **Integrated tools** (zum Debuggen von Code oder zur Darstellung einer Konsole, vgl. ebd. S. 3f)

3. **Object-support**

Es soll die Möglichkeit geben, Objekte von Klassen zu erstellen und Methoden aufzurufen:

[...] a user in a true object-oriented development environment should be able to interactively create objects of any available class, manipulate these objects and call their interface routines.

[Kölling II 1999, S. 5]

Interessanterweise wird an dieser Stelle eine Grundidee von BlueJ beschrieben, bei dessen Entwicklung u. a. KÖLLING mitwirkt (s. Anhang D.1):

[...] classes and objects should be the main abstractions used for user-level interaction in the environment. They should be treated as user level objects on which the user can perform operations by interacting with them through their interfaces.

[Kölling II 1999, S. 5]

Nur weil eine Umgebung eine objektorientierte Sprache unterstütze, sei die Umgebung noch nicht objektorientiert (vgl. ebd. S. 11).

4. **Support for code reuse** (Wiederverwendbarkeit von Code ermöglichen – eine der Hauptideen der Objektorientierung, vgl. ebd. S. 5)
5. **Learning support** (die Umgebung sollte Möglichkeiten zur Interaktion geben, zudem muss sie Konzepte (z. B. Klassen oder Vererbungsbeziehungen) graphisch und textuell darstellen können, vgl. ebd. S. 5f)
6. **Group support** (die Umgebung sollte paralleles Arbeiten an verschiedenen Stellen eines Projektes ermöglichen, vgl. ebd. S. 6f)
7. **Availability** (geringe Kosten der Umgebung und geringe Hardware-Anforderungen, vgl. ebd. S. 6)

Mit dem Blue-System haben KÖLLING und ROSENBERG versucht, eine Umgebung zu entwickeln, die den aufgezählten Punkten gerecht wird (vgl. ebd. S. 12). Aus dem Blue-System wurde BlueJ (»J« für Java), das gerade in Schulen eine große Anhängerschaft gefunden hat.

2.2.1 »Objects first«-Ansatz

KÖLLING und ROSENBERG haben mit ihrem Paper „Guidelines for Teaching Object Orientation with Java“ [Kölling Rosenberg 2001] ein richtungsweisendes Werk vorgelegt, wie Objektorientierung mit BlueJ zu unterrichten sei. Dabei haben sie Guidelines formuliert, die die Reihenfolge beim Vorgehen vorschreiben und das Ausbilden von Fehlvorstellungen verhindern sollen. Sie beginnen mit den Objekten (daher der Name) und arbeiten gleich zu Beginn mit mehreren Objekten. Erst später wird der Blick auf die Klassen erweitert und es werden andere Konzepte der Objektorientierung eingeführt.

Kritik für diesen Ansatz findet sich bspw. in [Cooper Dann et. al. 2003], wo vor allem die Komplexität als ein großer Kritikpunkt genannt wird:

An objects-first strategy is intended to have students work immediately with objects. This means students must dive right into classes and objects, their encapsulation (public and private data, etc.) and methods (the constructors, accessors, modifiers, helpers, etc.). All this is in addition to mastering the usual concepts of types, variables, values, and references, as well as with the often-frustrating details of syntax. Now, add event-driven concepts to support interactivity with GUIs!

[Cooper Dann et. al. 2003, S. 1]

Insbesondere der Blick für den Zustand eines Objekts und die Methoden, die aufgerufen werden können, könnten viele Lernenden überfordern (vgl. ebd. S. 1). Deshalb stellen die Autoren in ihrem Paper ein Konzept in der Programmierumgebung Alice⁵ vor, welches Objekte anhand einer 3D-Welt einführt, die durch Methodenaufrufe manipuliert werden. So stehen Käfer und Frösche bereit, die durch Methodenaufrufe fliegen oder hüpfen können. Objekte können durch Vererbung in ihrer Funktionalität erweitert werden. Das Interface stellt dabei einfache Interaktionsmöglichkeiten bereit, Methoden aufzurufen oder selber Methoden zu schreiben (es erinnert dabei stark an Scratch⁶).

Die Autoren sind sich dabei bewusst, dass die besondere GUI bzw. die besonderen Interaktionsmöglichkeiten auch Nachteile bieten können (z. B. wüssten die Studierenden nicht, wie man Code debuggt, oder hätten kaum Fehlermeldungen zu Syntax-Fehlern behoben etc.). Trotzdem ließen sich die Erfolge durch den 3D-Ansatz messen und die Autoren halten den »Objects first«-Ansatz für den besten, Objektorientierung zu lehren (vgl. [Cooper Dann et. al. 2003, S. 4f]).

⁵ Alice – <http://www.alice.org/>.

⁶ Scratch – <https://scratch.mit.edu/>.

Im Folgenden werden die acht Guidelines von KÖLLING und ROSENBERG vorgestellt und diskutiert. Für BlueJ gibt es mit [Barnes Kölling 2002] ein komplettes Lehrbuch, welches den »Objects first«-Ansatz verfolgt.

1. Objects first Die Idee ist, gleich zu Beginn mit Objekten zu arbeiten. BlueJ bietet einfache Möglichkeiten (im Gegensatz zu anderen Programmierumgebungen) Objekte zu erzeugen und zu manipulieren. Die Autoren beschreiben die ersten Schritte wie folgt:

Since objects can be created interactively, the first activity for students should be to open an existing project, create a few objects, make method calls on these objects and inspect the objects' state.

[Kölling Rosenberg 2001, S. 2]

Abbildung 6.3 (S. 87) zeigt, wie einfach sich in BlueJ Objekte erzeugen lassen. Auch wenn es nach den Autoren keine wissenschaftliche Begründung dafür gibt, bei dieser Reihenfolge ein gutes Verständnis von Objektorientierung zu erlangen, sei die anekdotische Evidenz trotzdem groß (vgl. ebd. S. 2). Außerdem können auf diese Art und Weise eine Reihe weiterer Konzepte mit eingeführt werden (ebd. S. 2):

- Auch wenn die Schülerinnen und Schüler nicht mit den Klassen eines Projektes in dem Sinne »arbeiten«, erkennen sie doch, dass es sie gibt, und dass diese die Komponenten eines Programms („the program's components“ [Kölling Rosenberg 2001, S. 2]) repräsentieren.
- Sie erkennen auch, dass sich von den Klassen Objekte erzeugen lassen, und dass viele Objekte von einer Klasse erzeugt werden können.
- Ferner erkennen die Schülerinnen und Schüler Struktureigenschaften von Objekten: Alle Objekte einer Klasse haben dieselben Methoden und Attribute, Objekte können sich aber in ihren Attributwerten unterscheiden, falls sie von derselben Klasse erzeugt worden sind. Objekte unterschiedlicher Klassen haben i. d. R. andere Methoden und Attribute.
- Die Schülerinnen und Schüler erkennen bei der Arbeit mit den Objekten auch, dass es Methoden mit Parametern und Rückgabewerten gibt.

2. Don't start with a blank screen Für Studenten und auch Schülerinnen und Schüler sei es am Anfang schwierig, zu entscheiden, welche Klassen ein Projekt brauche. Deshalb sei die Arbeit *mit* einem Projekt besser, als die Schülerinnen und

Schüler ein Programm von Anfang an schreiben zu lassen (vgl. [Kölling Rosenberg 2001, S. 2]). Die Schülerinnen und Schüler können zu Beginn Änderungen im Quelltext von Methoden vornehmen und Änderungen im Programmverhalten beobachten und erklären.

3. Read code Eine oft nicht wahrgenommene Möglichkeit beim Erlernen des Programmierens sei es, gut geschriebene Programme von anderen Autoren zu lesen (vgl. [Kölling Rosenberg 2001, S. 2]). Es gebe nach Aussage von KÖLLING und ROSENBERG viele Kurse, in denen Lernende nur ihre eigenen Programme lesen. Sie weisen natürlich auch darauf hin, dass der gezeigte Code es wert sein muss, von den Lernenden gelesen zu werden und vielleicht auch einen Stil haben sollte, der es wert ist, kopiert zu werden.

BERGIN fordert dies auch ein (vgl. [Bergin 2000]). Schülerinnen und Schülern sei es so möglich, größere Programme kennenzulernen, die sie mit ihrem anfänglichen Wissensstand noch gar nicht hätten schreiben können. BERGIN hofft hier auch auf Synergieeffekte beim Lesen von „well designed and well written programs“ [Bergin 2000].

4. Use „large“ projects Die Autoren stellen sich gegen die Meinung, dass große Projekte Schülerinnen und Schüler am Anfang überfordern würden und plädieren sogar für große Programme („several classes with a sensible number of methods“ [Kölling Rosenberg 2001, S. 2]). Auf diese Weise besteht die Möglichkeit, dass die Schülerinnen und Schüler den Nutzen von einer Dokumentation und gut lesbarem Code (Guideline 3) erkennen. Des Weiteren können sie evtl. im Team arbeiten, UML-Diagramme für das Projekt nutzen und so »soft skills« ausarbeiten, die bei der Arbeit und Organisation im Team unerlässlich sind (vgl. [Stevens Henry et. al. 2000, S. 399f]). In [Stevens Henry et. al. 2000, S. 399f] wird von vielen (universitären) Kursen berichtet, in denen große Projekte gut etabliert werden konnten und gute Ergebnisse erzielt worden sind. Neben den »soft skills« kann so die komplette Entwicklungsphase eines Projekts beleuchtet werden: Von den Anforderungen an die Software, dem ersten Entwurf, durchzuführenden Änderungen und Tests, bis hin zur Dokumentation und Installation, besteht die Möglichkeit, so einen Einblick in die Arbeitstechniken der „real world“ (ebd. S. 399) zu erlangen. Von KÖLLING und ROSENBERG wird dementsprechend der klare Hinweis gegeben:

This means that showing students programs that consist of a single method is showing them bad examples (and thus contradicts our guideline 3).

[Kölling Rosenberg 2001, S. 2]

5. Don't start with "main" Da die `main`-Methode nichts mit Objektorientierung zu tun hat (vgl. [Kölling Rosenberg 2001, S. 2] u. nächste Guideline für eine andere Meinung) und auch für das Erzeugen von Objekten nicht von Nöten ist (ebd. S. 3), sollte auch nicht mit ihr angefangen werden. Hinter der `main`-Methode stehen die Konzepte der »statischen Methoden«, die für Anfänger zunächst uninteressant sind.

6. Don't use "Hello world" Diese Guideline ergibt sich direkt aus der vorherigen. Ein simples »Hello World«-Programm bricht mit allen vorher aufgestellten Guidelines und sollte nicht am Anfang der Objektorientierung stehen (zumal noch nicht einmal ein Objekt erzeugt oder eine Methode implementiert wird, die Objekte manipuliert).

In [Lewis 2000, S. 247] dagegen wird argumentiert, dass es sich auf bei dem Aufruf der Zeile

```
1 System.out.println("Hello World");
```

um Objektorientierung handle, da beim Objekt `System.out` die Methode `println(...)` aufgerufen werde. Nach wie vor bleibt aber die Kritik, dass der Aufruf mit den vorherigen Guidelines bricht. Außerdem wird nicht der Versuch unternommen, dass die Schülerinnen und Schüler wirklich in selbstständige Interaktion mit Objekten und Methoden treten, sondern einfach fertige Konzepte nutzen, die für sie unsichtbar und wenig nachvollziehbar sind.

7. Show program structure In vielen Programmierumgebungen ist die Klassenstruktur eines Projekts nicht ersichtlich. Für die Schülerinnen und Schüler ist es aber wichtig, ein Verständnis aufzubauen, welche Klassen benötigt werden und wie sie in einem Projekt zusammenspielen. Durch das Studieren des zugehörigen Klassendiagramms, welches BlueJ automatisch für ein Projekt generiert, kann dies gelingen (vgl. [Kölling Rosenberg 2001, S. 3]).

An dieser Stelle sei angemerkt, dass andere Programmierumgebungen dies meist nicht bieten („[BlueJ] offers a unique interaction style“, ebd. S. 1). Dies ist der Tatsache geschuldet, dass BlueJ extra für die Lehre entwickelt wurde. Trotzdem sollen die Schülerinnen und Schüler auch ohne das Klassendiagramm ein Verständnis ausbilden, welche Klassen für ein Projekt nötig sind und wie sie miteinander kommunizieren (auch wenn man die Klassen bspw. nur in einer Liste sieht).

8. Be careful with the user interface Die Idee dieser Guideline ist, dass Schülerinnen und Schüler im Unterricht nicht zu viel Zeit in das Aussehen der Programmoberfläche (GUI / User-Interface) investieren. Zwar ist ein gutes Design wichtig (Steigerung der Funktionalität), aber es besteht die Gefahr, dass Schülerinnen

und Schüler sich in Kleinigkeiten verlieren und zu wenig Zeit für die zugrundeliegenden Programmstrukturen aufwenden. Das User-Interface an sich ist ein schlechtes Beispiel für Objektorientierung und lenkt den Blick zu sehr vom Wesentlichen ab (vgl. [Kölling Rosenberg 2001, S. 3]). Dazu schreiben KÖLLING und ROSENBERG:

GUIs provide a serious distraction from the real issues underlying general programming concepts and do not serve well to illustrate general principles. GUI code is a very specific example of an object structure that has very idiosyncratic characteristics that are not common to OO in general.

[Kölling Rosenberg 2001, S. 3]

Um diesen Problemen aus dem Weg zu gehen, können Beispiele im Unterricht vorgegeben werden. Dabei kann die Erstellung des User-Interfaces in die Hausaufgaben ausgelagert werden, sodass die Schülerinnen und Schüler dies nach ihrem Belieben gestalten können.

2.2.2 »Models first«-Ansatz

In [Diethelm] wird ein Unterrichtskonzept für die OOM vorgestellt, das vier Leitideen verfolgt: models first, strictly objects first, Nachvollziehbarkeit und Ausführbarkeit.

DIETHELM fasst den Begriff »models first« hier so auf, dass „die objektorientierte Modellierung von Anfang an im Informatikunterricht unterrichtet wird, insbesondere, dass nicht nur die Modellierungssprache, sondern auch die Modellierungstechnik von Beginn an gelehrt werden soll“ [Diethelm, S. 46]. So soll die Modellierung als informative Denkweise konsequent von Beginn an gelehrt werden und gewissermaßen zur „Muttersprache des [...] Problemlösens“ [Diethelm, S. 47] werden.

BENNEDSEN und CASPERSEN verfolgen in [Bennedsen Caspersen 2004] denselben Ansatz. Sie haben in CS1-Kursen an Universitäten mit dem »Models first«-Ansatz Erfolge erzielen können, da der Fokus auf die Modellierung eine helfende Struktur gebe, die gerade für Programmieranfänger hilfreich sei. Ferner verfolgen sie einen systematischen Aufbau bzw. eine systematische Herangehensweise an die Programmierung, die auch eine Hilfestellung sei (vgl. [Bennedsen Caspersen 2004, S. 1]).

DIETHELM versteht die OOM nicht als „unnötige[n] Overhead oder als schlicht zu komplex“ [Diethelm, S. 45], macht aber darauf aufmerksam, dass beim Umstieg von einer strukturierten Programmiersprache auf eine objektorientierte Sprache Probleme auftreten können. In der Didaktik herrscht Konsens darüber, dass solche Wechsel Probleme aufwerfen können, da Änderungen in der Denkweise und neue Konzepte verstanden werden müssen (vgl. [Bergin 2000]). Das Objektspiel (vgl. Kapitel 4.5) sei eine Möglichkeit, die Besonderheiten der OOM zu veranschaulichen.

So könnte ein besseres Verständnis vom Zusammenspiel der Objekte erlangt werden (vgl. [Diethelm, S. 52]).

Damit ein Programm den Ansprüchen des »Models first«-Ansatzes gerecht wird, sollte es ein paar besondere Eigenschaften mitbringen: Einerseits sollte es ausreichend komplex sein, sodass die Schülerinnen und Schüler Modellierungsentscheidungen treffen müssen (d. h. mindestens drei Klassen für das Einführungsbeispiel, vgl. [Diethelm, S. 49]). Andererseits sollte das ausgewählte Beispiel aus einem „vertrauten Erfahrungsbereich“ [Diethelm, S. 49] der Schülerinnen und Schüler kommen.

Es wäre wünschenswert, wenn das Anfangsprojekt umfangreich genug wäre, dass die anfängliche Modellierung spätere Änderungen provozieren würde. Darüber hinaus sollte es Möglichkeiten bieten, in Gruppen zu arbeiten und über Modellierungsentscheidungen zu diskutieren, sodass Schülerinnen und Schüler verschiedene Sichtweisen auf eine Problemstellung kennenlernen (vgl. ebd. S. 49).

DIETHELM plädiert für eine Fortführung des Unterrichts nach dem Modellieren durch den »(strictly) Objects first«-Ansatz, sodass die Blickrichtung vom Modell zu den Objekten geht. Erst später sollten sich die Klassen aus den Objekten „ergeben“ [Diethelm, S. 47], ebenso wie die Modellierung des Verhaltens. Auch hier bestünde die Möglichkeit mit dem Objektspiel anzusetzen, um das Zusammenspiel bei Objekten kennenzulernen und Entscheidungen zur Implementierung zu treffen (vgl. [Diethelm, S. 45, 47, 51f]). Einen ausführlichen Vorschlag zur Unterrichtsgestaltung findet sich im Skript von DIETHELM, GEIGER und SCHULTE in [Diethelm Geiger et. al. 2006].

BENNEDSEN und CASPERSEN verstehen den »Models first«-Ansatz so, dass neben einer systematischen Herangehensweise an die Programmiersprache immer auch ein tieferes Verständnis des Programmierprozesses angestrebt werden sollte. Dies geschehe durch den stetigen Wechsel zwischen der Programmierung und der Modellierung („Coding and conceptual modeling is done hand-in-hand“ [Bennedsen Caspersen 2004, S. 2]). Andererseits solle man sich nicht mit Eigenheiten von Programmiersprachen aufhalten, sondern eher den Blick für das hintergründliche Konzept erlangen (vgl. [Bennedsen Caspersen 2004, S. 2]).

Die Autoren fangen damit an, Klassen über UML-Diagramme einzuführen und an diesen die „basic language constructs“ [Bennedsen Caspersen 2004, S. 3] wie Zuweisungen, Parameter oder Kontrollstrukturen zu erläutern (vgl. ebd. S. 3). Später werden Relationen wie *Eine Person ist mit einer anderen Person verheiratet* oder *Eine Person liebt eine andere Person* modelliert und in Klassen übersetzt. Die $0..*$ -Assoziation bietet dabei die Möglichkeit, auf Collections in Java einzugehen, und weitere typische Programmierkonzepte kennenzulernen (Schleifen, Iteratoren).

Sie verfolgen fünf Abstraktionsniveaus, die den Studierenden Hilfestellung im Kreativprozess der Modellierung und der Programmierung geben. Der Fokus wird dabei weniger auf die algorithmische Seite (hier ist bei der Umsetzung kaum Kreativität gefragt), sondern eher auf die Modellierungsseite gelegt (vgl. ebd. S. xxx). Im Speziellen werden die folgenden Blickrichtungen in dem Kurs umgesetzt:

1. **Problem domain** → **model**: UML-Klassendiagramme für das Problem erstellen, Fokus auf die Struktur zwischen Klassen legen
2. **Model** → **Java code**: Standardkonzepte aus dem Klassendiagramm mit coding patters in Java abbilden (bspw. Collections nutzen)
3. **Functionality** → **Java code**: Eigenschaften implementieren und an andere Klassen weitergeben (auch: Beziehungen herstellen, um Verantwortlichkeiten zu verteilen)
4. **Implementation of classes**: Dafür sorgen, dass Methodenaufrufe in den Klassen funktionieren (Bedingungen des Problems in den Klassen repräsentieren)
5. **Implementation of methods**: Mit algorithm patterns für Standardprobleme (suchen, sortieren, Schleifen etc.) die Methoden ausfüllen

(vgl. [Bennedsen Caspersen 2004, S. 4])

BENNEDSEN und CASPERSEN haben damit in der Durchführung eine andere Sichtweise auf den »Models first«-Ansatz als DIETHELM, sind sich bei der Bedeutung des Modellierungsprozesses aber einig und verfolgen beide das Ziel, über Modellierungsentscheidungen nachzudenken und nachvollziehen zu können. Im besten Falle werden die Lernenden angehalten, Standardprobleme durch die Konzepte der Objektorientierung zu lösen (z. B. Modellierungen für $0..*$ -Assoziationen, Methodenaufrufe bei anderen Objekten, Klassen haben Objekte anderer Klassen als Attribut etc.).

2.2.3 »Objects later«-Ansatz

Der »Objects later«-Ansatz⁷ ist die natürliche Gegenposition zum »Objects first«-Ansatz. Namensgebend ist die spätere Einführung objektorientierter Konzepte wie Objekt, Klasse oder Methode (und auch darauf aufbauender Konzepte wie Vererbung oder allgemeiner Modellierungstechniken).

⁷ In manchen Publikationen auch als »Objects late«-Ansatz bezeichnet.

UYSAL fasst die Unterschiede wie folgt zusammen und stellt dabei insbesondere das Einführen von Kontrollstrukturen oder basalen Programmierkonzepten (u. a. Datentypen, Variablen, Konstanten, Schleifen, Iteratoren) an den Anfang des »Objects later«-Ansatzes (vgl. auch [Uysal 2012, Tabelle 2, S. 818]):

The objects-first method puts emphasis on both design and programming principles of OOP from the very beginning. However, the objects-late method initially starts with non-object-oriented concepts, such as the statements and the control structures, and it defers the discussion of classes and objects to later lectures.

[Uysal 2012, S. 817]

In den Studien von [Uysal 2012] und [Ehlert Schulte 2009] kommt als Programmierumgebung auch BlueJ zum Einsatz. Dabei wird nicht auf die visuelle Ebene von BlueJ (die UML-ähnliche Klassenübersicht) eingegangen, sondern auf die textuelle (Quelltexteditor).

2.2.4 »Objects first«- und »Objects later«-Ansatz im Vergleich

UYSAL hat in einer Studie die Effekte des »Objects first«- und »Objects later«-Ansatzes verglichen. Dazu wurden 20 männliche Studierende mit einem B.Sc.-Abschluss in System Engineering in zwei Gruppen aufgeteilt, die mit jeweils anderen Ansätzen unterrichtet wurden. Als Programmierumgebung wurde BlueJ ausgewählt.

Die Nullhypothese, dass kein statistisch signifikanter Unterschied zwischen beiden Gruppen existiere, konnte dabei verworfen werden. UYSAL schreibt dazu, dass es einen statistisch relevanten Unterschied beider Gruppen gegeben habe (die Werte sind z -standardisiert):

[...] as it is seen in Table 5, there is a statistically significant difference between these experimental groups and we reject the null hypothesis ($z = -2.290$, $p < 0.05$). It is possible to say that the learners instructed with objects-first method achieved higher learning outcomes.

[Uysal 2012, S. 820]

Auf der anderen Seite konnte die durchgeführte Studie von EHLERT und SCHULTE in [Ehlert Schulte 2009] keine statistisch relevanten Unterschiede zwischen zwei Lerngruppen messen. Sie haben aber festhalten können, dass schwierige Themen wie Assoziationen und Arrays (vgl. [Ehlert Schulte 2009, Kap. 5.3, S. 20f]) nicht vom Lehransatz beeinflusst werden (vgl. ebd. S. 24). In ihrer Arbeit verweisen sie auch auf andere Studien, in denen sowohl Vor- und Nachteile eines Lehrsatzen gefunden wurden, als auch auf Studien, in denen keine Differenzen festgestellt wurden (vgl. [Ehlert Schulte 2009, S. 15]).

2.2.5 Weitere Ansätze

COOPER, DANN und PAUSCH üben unterdessen Kritik am »Objects first«-Ansatz, indem sie u. a. darauf aufmerksam machen, dass Lernende immer sowohl den Zustand als auch die Methoden im Überblick behalten müssten:

The functional-first strategy initially focuses on functions, deferring a discussion of state until later. The imperative-first strategy initially focuses on state, deferring a discussion of functions until later. The objects-first strategy requires an initial discussion of both state and functions. The challenge of an objects-first strategy is to provide a way to help novice programmers master both of these concepts at once.

[Cooper Dann et. al. 2003, S. 1]

Im Folgenden sollen noch zwei andere Herangehensweisen an die Objektorientierung kurz aufgezeigt werden. Dabei handelt es sich um den »Methodology First and Language Second«-Ansatz und die Möglichkeit, die Objektorientierung mithilfe der »Stifte und Mäuse«-Bibliotheken einzuführen.

»Methodology First and Language Second«-Ansatz Dieses von ZHU und ZHOU beschriebene Unterrichtskonzept (für C++) fängt mit einer Erklärung grundlegender, objektorientierter Konzepte an (Objekt, Klasse – in dieser Reihenfolge) und geht dann den Weg zur Vererbung. In [Arif 2000] wird für C++ ein Beispiel gegeben, wie anhand eines Taschenrechner-Projekts solch eine Entwicklung aussehen kann. Auch wenn sich beide Paper auf C++ beziehen, sollte eine Übertragung in andere Programmiersprachen möglich sein.

Genauer betrachtet orientiert sich der »Methodology First and Language Second«-Ansatz an folgenden sechs Schritten (vgl. [Zhu Zhou 2003, S. 2]), die sich so auch im Paper von ARIF nachvollziehen lassen (allerdings nicht in dieser Granularität⁸):

1. Grundlegende Prinzipien der Objektorientierung erklären.

Dabei können die Überlegungen top-down oder bottom-up erfolgen. Entweder wird ein System in immer weitere kleinere Systeme (Objekte) zerlegt bzw. spezialisiert (top-down) oder kleinere Systeme (Objekte) zu größeren zusammengefasst bzw. generalisiert (bottom-up). Beide Sichtweisen lassen sich in der Objektorientierung abbilden (s. auch nächster Punkt).

2. Das Konzept »Objekt« anhand von Beobachtungen aus der realen Welt erklären (orientiert an Smalltalk: (buchstäblich) *alles* ist ein Objekt).

⁸ Die Studierenden in dem Paper von ARIF hatten Vorerfahrungen mit strukturierter Programmierung, sodass der Lernprozess auch mit Erklärungen zur Syntax anfing.

3. Das Konzept »Klasse« durch Abstraktion vieler ähnlicher Objekte kennenlernen (induktives Vorgehen – vom Speziellen zum Abstrakten).
4. Klassen instanziieren, nachdem das Konzept »Klasse« gelehrt und verstanden wurde (eine Klasse versteht sich dabei als „template“ [Zhu Zhou 2003, S. 3], sprich »Vorlage« für Objekte, vgl. ebd. S. 3).
5. Unterklassen als Möglichkeit der Spezialisierung und Oberklassen als Möglichkeit der Generalisierung einführen.
6. (Optional) Metaklassen⁹ einführen (um das Objekt- und Klassenverständnis zu vervollständigen).

Damit greift das Konzept drei wichtige Bereiche der Objektorientierung (»Klasse«, »Objekt« und »Vererbung«) auf (vgl. auch [Arif 2000, S. 30]). Auf die Vererbung kann dabei in besonderer Weise eingegangen werden. Einmal kann sie als Möglichkeit der Spezialisierung verstanden werden, anderseits als bedeutungsunabhängige Arbeitserleichterung, da keine weiteren Klassen geschrieben werden müssen (vgl. [Zhu Zhou 2003, S. 4]). In dem Paper von ZHU und ZHOU wird noch auf weitere Besonderheiten von C++ eingegangen, die hier nicht mehr dargestellt werden.

»Stifte und Mäuse« (SuM) Die Bibliothek »Stifte und Mäuse« (SuM) wurde in den 90er Jahren im Rahmen der Lehrerfortbildung von Ulrich BORGOFF, Dr. Jürgen CZISCHKE, Dr. Georg DICK, Horst HILDEBRECHT, Dr. Ludger HUMBERT und Werner UEDING entwickelt (vgl. [Schriek 2005, S. 9]). Sie wurde zunächst für Pascal und Oberon geschrieben, wurde später aber nach Java portiert (vgl. [Schriek MG Werl SuM 2003]). Sie ist sowohl für Schülerinnen und Schüler als auch für Lehrkräfte oft der erste Kontakt mit der Objektorientierung. Bernard SCHRIEK schrieb daher die Lehrbücher [Schriek 2005, Schriek 2006, Schriek 2007], um im Informatikunterricht der Oberstufe SuM einsetzen zu können.

Der SuM-Kern ist eine Ansammlung von Klassen, die den Computer abbilden. Er umfasst Klassen für Bildschirm, Maus und Tastatur, einen Stift und eine allgemeine Anwendung (vgl. [Spolwig 2006, S. 1]).

Das Problem, welches sich bei der Einführung der Objektorientierung mit der SuM-Bibliothek ergibt, ist Folgendes: In anfänglichen Beispielen werden Zeichnungen am Computer umgesetzt, die abgesehen von der objektorientierten Notation¹⁰ nichts

⁹ Metaklassen sind Klassen, deren Instanzen Klassen sind. Da dies für die Schule nicht von Relevanz ist, soll an dieser Stelle auf eine weitere Ausführung verzichtet werden. Eine Erklärung wird aber in [Zhu Zhou 2003, S. 4] gegeben.

¹⁰ Aufruf von Methoden wie in [Humbert 2006, S. 120] dargestellt:
 <objektbezeichner>.<methodenbezeichner>(<parameterwert>)

mit eigentlicher Objektorientierung zu tun haben. So sollen in einem ersten Beispiel einfache Strichzeichnungen umgesetzt werden (vgl. [Schriek 2005, Übung 3.5, S. 29]). Dieses Programm (genau genommen die Klasse `MeineZeichnungen`) ist dabei aber eher als „anweisungsgesteuertes Malprogramm“ [Spolwig 2006, S. 2] zu verstehen (»Turtle-Graphic«), als ein objektorientiertes Programm, das einer Analyse und Modellierung folgt.

Bei der Einführung abstrakter Klassen und Vererbung schlägt SCHRIEK vor, dies am Beispiel einer sich bewegenden Eisenbahn zu umsetzen (vgl. [Schriek 2005, Kap. 9, S. 96ff]). Dabei erben die Klassen `Lokomotive`, `Personenwagen` und `Gueterwagen` alle von der Klasse `waggon`, erweitern aber die Funktionalität des Waggons nicht. Sie unterscheiden sich lediglich darin, dass sie unterschiedlich auf dem Bildschirm gezeichnet werden. So werden keine Anwendungsaspekte und Ausschnitte aus der realen Welt umgesetzt (vgl. [Spolwig 2006, S. 5]). Dabei wird das oft vorhandene Problem der SuM-Programme deutlich:

SuM bewirkt auf dieser Ebene durch die Restriktionen, dass jedes Objekt prinzipiell Attribute aus der Kern-Bibliothek haben muss, ein eigenartiges Abbild der gefundenen Objekte. Überspitz [sic!] könnte man auch sagen, wer als einziges Werkzeug einen Stift hat, für den besteht die ganze Welt aus Strichen. Damit wird die Forderung nach sauberer Modellierung konterkariert und den Schülern vom ersten Tage an ein falsches Bild von den Entwurfsaufgaben der Informatik und den Möglichkeiten der OOP vermittelt.

[Spolwig 2006, S. 5]

SPOLWIG führt auf, dass die Klassen in den SuM-Beispielen oft über einen Bildschirm und einen Stift als Attribut verfügen, um sich selbst auf dem Bildschirm darzustellen. Dies seien allerdings Attribute, die in einer korrekten Modellierung ausgelagert werden würden (vgl. ebd. S. 5).

2.3 Probleme beim Verständnis: Fehlvorstellungen

Wenn Anfänger das Programmieren lernen, machen sie unweigerlich auch Fehler. Im Folgenden werden Fehlvorstellungen diskutiert, die sich aus anfänglich einfachen Fehlern zu einer falschen Denkweise / Sichtweise auf ein Konzept gewandelt haben. Die Forschung zu Fehlvorstellungen und wie sie aufzulösen oder zu vermeiden sind, steht im Blickpunkt vieler Didaktiker (vgl. [Eckerdal Thuné 2005, S. 89]).

Es ist darauf hinzuweisen, dass im deutschsprachigen Raum keine ausgewiesene Informatikdidaktikforschung zu diesen Fragen stattfindet (vgl. [Humbert 2006,

S. 156]). Deswegen werden im Folgenden vor allem englischsprachige Quellen zitiert, die sich auf universitäre Veranstaltungen (zumeist CS1, CS2¹¹) beziehen.

2.3.1 Was sind Fehlvorstellungen?

Für die Einführung des Begriffs »Fehlvorstellung« soll aus genannten Gründen auf die englischsprachige Literatur verwiesen werden. In [Holland Griffiths et. al. 1997, S. 131] wird darauf hingewiesen, dass Fehlvorstellungen (engl. »misconceptions«) spätere Lernerfahrungen behindern können und so Blickwinkel auf die Objektorientierung verzerren können:

[...] students [can] [...] develop misconceptions, which can be hard to shift later. Such misconceptions can act as barriers through which later all teaching on the subject may be inadvertently filtered and distorted.

[Holland Griffiths et. al. 1997, S. 131]

Diese Definition liegt auch dieser Masterarbeit zugrunde. Hier werden Fehlvorstellungen als falsche gedankliche Vorstellungen bzw. Konstrukte verstanden, die das weitere Lernen zu einem späteren Zeitpunkt hemmen (können) und Fehler produzieren (können), auf jeden Fall aber dem Konsens, wie ein objektorientiertes Konzept zu verstehen ist, widersprechen (die Vorstellungen sind somit objektiv falsch).

Bei den Fehlvorstellungen ist es problematisch, dass sie sich im Nachhinein nur schwierig wieder korrigieren lassen (vgl. [Holland Griffiths et. al. 1997, S. 131]). Falsche gedankliche Konzepte von objektorientierten Prinzipien bereiten bei komplexeren Fragestellungen Probleme, fallen bei kleineren Projekten oder den anfänglichen Aufgaben aber meist nicht auf.

Darüber hinaus wissen Schülerinnen und Schüler i. d. R. nicht um ihre eigenen Fehlvorstellungen. Deswegen ist es umso wichtiger, als Lehrkraft zu versuchen, Fehlvorstellungen aufzudecken und den Unterricht möglichst von Beginn an so zu gestalten, dass das Ausbilden von Fehlvorstellungen vermieden wird. So wird die Unterrichtsqualität gesteigert und es kann ein in sich stimmiges Bild der Objektorientierung gelehrt werden.

2.3.2 Wie äußern sich Fehlvorstellungen in der OOP?

Fehlvorstellungen äußern sich auf vielfältige Art und Weise. HUMBERT erwähnt zum Beispiel die »If-Schleife« oder die Verwechslung von »Vergleich« und »Assignment«

¹¹ CS1 und CS2 (Computer Science 1, 2) gehören zu den Einführungsveranstaltungen im Informatikstudium an englischen und amerikanischen Universitäten (1. und 2. Semester). In Deutschland gibt es entsprechende Vorlesungen im Grundstudium.

von Variablen, die je nach Programmiersprache auch unterschiedlich implementiert sind (vgl. [Humbert 2006, S. 156]).

Die Autoren SANDERS und THOMAS haben in [Sanders Thomas 2007, S. 170] eine Checkliste erstellt, anhand derer sich das Verständnis der Objektorientierung ableiten lasse. Sie enthält z. B. folgende Items, an denen sich ein schlechtes Verständnis der Objektorientierung ablesen lasse:¹²

- Es tauchen identische Klassen auf, die nur einen anderen Klassennamen haben, oder Klassen unterscheiden sich nur geringfügig.
- Es tauchen nicht verwendete Klassen auf, der gesamte Code steht in einer Klasse oder die Arbeit passiert nur über das Assignment (s. Bsp. 7, Kapitel 2.3.3).
- Fehler bei der Vererbung: Verwechslung von Ober- und Unterklasse oder zwei Klassen enthalten dieselbe Methode, die in eine Oberklasse ausgelagert werden könnte (vgl. [Sanders Thomas 2007, S. 167, 168]).
- Verwechslung von Attributwert und Attribut bei Namen von Variablen (s. Bsp. 8, Kapitel 2.3.3) oder Klassen enthalten keine Methoden (bis auf Konstruktoren).

Nachfolgend werden einige Beispiele gegeben. Es sei aber noch einmal darauf hingewiesen, dass nach [Sanders Thomas 2007, S. 169] auch funktionierende bzw. ausführbare Programme Fehlvorstellungen enthalten können. Dies zeigte sich auch bei den Untersuchungen der Experten. So schreiben sie:

What is evidenced in these programs, however, is that working programs can contain subtle errors that suggest a serious misconception. For instance, the variations that students showed of the class-instance-conflation misconception were a real surprise to us.

[Sanders Thomas 2007, S. 169]

Ein grundsätzliches Ziel der Lehre ist es natürlich, die Objektorientierung so zu unterrichten, dass Schülerinnen und Schüler bestenfalls ein Verständnis gänzlich ohne Fehlvorstellungen erlangen. Die nachfolgend aufgeführten Fehlvorstellungen lassen sich häufig durch eine günstige Wahl der anfänglichen Beispiele vermeiden, deren Bedeutsamkeit auch in [Diethelm, Kap. 3.2, S. 49] dargestellt wird.

¹² In [Sanders Thomas 2007, S. 170] gibt es zwei tabellarische Übersichten von gutem und schlechtem Verständnis der Objektorientierung, die wesentlich umfangreicher als die hier aufgeführten Punkte sind. Es wurde versucht, sich hier auf die wesentlichen Punkte zu konzentrieren und wenn möglich Aspekte zusammenzufassen.

2.3.3 Einige Beispiele für Fehlvorstellungen (mit Lösungsansätzen)

In [Schmolitzky 2006] werden sieben Thesen formuliert, die beim Lernen der Objektorientierung auftreten können. Es wird später noch dargestellt, wie diese Aspekte durch das Projekt dieser Masterarbeit verhindert oder angeregt wurden (s. Kapitel 6.3). Darüber hinaus werden noch einige der in [Holland Griffiths et. al. 1997] beschriebenen Fehlvorstellungen vorgestellt.

1. Objekte haben Namen Dass Objekte Namen haben, ist eine sehr weit verbreitete Fehlvorstellung. Darüber hinaus verwechseln viele Lernende, dass der Name eines Objekts etwas anderes ist als der Attributwert des Attributs `name`. Oft werden Klassen modelliert, die solch ein Attribut haben.

Weitere Erläuterungen u. Idee zur Lösung Auch wenn diese Sprechweise umgangssprachlich sehr weit verbreitet ist, sollte erklärt werden, dass der Zugriff auf Objekte durch eine Variable passiert, der »Name« eines Objekts aber nicht bekannt ist. Diese Variable wird oft für den Namen des Objekts gehalten, dabei hat das Objekt keinen Namen und im Rechner werden Objekte auch nur über die Speicheradresse verwaltet (die i. d. R. nicht weiter interessiert). In [Holland Griffiths et. al. 1997, S. 133] werden von dieser Fehlvorstellung weitere falsche Konzepte abgeleitet:

- Nur eine Variable kann auf ein Objekt zu einem Zeitpunkt verweisen.
- Verweist eine Variable auf ein Objekt, kann sie niemals auf ein anderes Objekt verweisen.
- Verweist eine Variable auf ein Objekt, spezifiziert sie es zu jedem Zeitpunkt.
- Zwei Unterschiedliche Variablen müssen auch auf unterschiedliche Objekte verweisen.
- Ein Objekt kann nach den Variablen gefragt werden, die auf das Objekt verweisen.
- Zwei Objekte derselben Klasse mit demselben Zustand sind dieselben Objekte (sind identisch).
- Zwei Objekte mit demselben Attributwert des Attributs `name` sind dieselben Objekte (sind identisch).

Die Autoren schlagen eine Reihe von Gegenbeispielen zum Auflösen dieser Fehlvorstellungen vor. In diesen Beispielen sollen mehrere Variablen auf ein Objekt verweisen, Variablen nach und nach auf verschiedene Objekte verweisen oder versucht werden, Objekte anhand ihres Attributs `name` und ihres Verhaltens auseinander zu halten.

2. Die Zustandsfelder von Objekten definieren ihre Attribute Der Begriff »Attribut« wird auch im alltäglichen Sprachgebrauch genutzt, bspw. für Eigenschaften oder besondere Ausprägungen, die eine Person, ein Gegenstand oder allgemein eine Sache hat. In der Anfängerausbildung kann es zu einer Übergeneralisierung kommen, wenn alles das, was ein Objekt buchstäblich hat, als Attribut definiert wird.

Idee zur Lösung Es ist an vielen Stellen möglich, auf Attribute zu verzichten und eine Realisierung des Modellierziels auf einem anderen Weg zu erreichen. [Schmolitzky 2006, S. 2] nennt als Beispiel ein Konto, das seinen Kontostand mit einem Attribut `kontostand` verwaltet. Dieses Attribut ist aber nicht zwingend erforderlich. So könnte der Kontostand auch als Summe aller Ein- und Abbuchungen implementiert werden.

Weitere Beispiele liefern Situationen, in denen entweder Informationen in Form von Attributen vorgehalten oder bei Bedarf (durch Methodenaufrufe) generiert werden können. Dies hat natürlich Auswirkungen auf das Laufzeitverhalten.

3. Objekte enthalten Objekte Diese Fehlvorstellung kann entstehen, wenn Klassen Objekte anderer Klassen als Attribute haben. Dabei wird vergessen, dass nur Zeiger auf Objekte gesetzt werden und diese auch nur in der Attributliste eines Objekts verwaltet werden. Die Objekte existieren nebeneinander – und nicht ein Objekt *in* einem anderen.

Idee zur Lösung Am einfachsten ist es, eine Situation zu schaffen, in der ein Objekt von mehreren Objekten als Attribut vorgehalten wird. Wird das Objekt, das ein anderes Objekt vermeintlich »enthält«, dereferenziert, müsste die Zugriffsmöglichkeit auf das Attributobjekt eigentlich auch verloren gehen. Trotzdem lassen sich Beispiele schaffen, in denen dies möglich ist.

Stellt man sich eine Patientenverwaltung vor, in der ein Arzt (Klasse `Arzt`) ein Attribut `aktuellerPatient` hat, so muss der Zugriff auf den Patienten auch nach Dereferenzierung des `Arzt`-Objekts möglich sein (z. B. ließe der Feierabend des Arztes

das Objekt dereferenzieren, die Verwaltung hätte aber nach wie vor Zugriff auf den Patienten). Das `Arzt`-Objekt »enthält« also nicht den Patienten, sondern greift wie alle anderen Objekte auf dieses zu.

4. Mit Vererbung werden Begriffshierarchien abgebildet Diese Fehlvorstellung ist weit verbreitet (vgl. [Schmolitzky 2006, S. 34]). Viele Schülerinnen und Schüler verstehen die Vererbung als eine »ist-ein«-Beziehung, wenn bspw. die Klassen `Limousine`, `Sportwagen` und `Gelaendewagen` von der Klasse `Auto` erben. Die Vererbung als reine »ist-ein«-Beziehung zu sehen, reicht aber nicht aus, da auch der Blick für die Erweiterungsmöglichkeiten der Hauptklasse durch die Unterklassen außer Acht gelassen wird. Zudem ist diese Vorstellung nicht vollständig.

Idee zur Lösung Auch wenn die »ist-ein«-Beziehung ist im Auto-Beispiel vielleicht sinnvoll ist, funktioniert sie bei anderen Beispielen nicht. Wenn die Klassen `Rotkehlchen`, `Adler` und `Strauss` von der Klasse `Vogel` erben würden, wäre die Realisierung mittels Vererbung nicht sinnvoll, da der Strauß nicht fliegen kann, aber alle drei Tiere Vögel sind.

Ein anderes problematisches Beispiel ist die Hierarchie in einem Krankenhaus. Die Klassen `Obearzt`, `Arzt` und `Pflegepersonal` stehen nicht notwendigerweise auch in einer Vererbungshierarchie – hier muss darauf geachtet werden, dass nicht automatisch eine Übergeneralisierung stattfindet. Die natürliche Hierarchie der zu modellierenden Situation muss sich nicht im Programm widerspiegeln.

Für die OOP sind die Beziehungen »kann verwendet werden als« und »übernimmt Verhalten und Struktur von« daher auch von großer Bedeutung, mit denen man auch einsieht, dass das Vogel-Beispiel nicht funktioniert. Auch die Vererbung im Krankenhaus-Beispiel kann mit diesen beiden Beziehungen überprüft werden.

5. Objekte sind so etwas wie Variablen Die in [Holland Griffiths et. al. 1997, S. 132] beschriebene Fehlvorstellung tritt häufig dann auf, wenn Objekte mit nur einem Attribut verwendet werden. Haben Schülerinnen und Schüler schon prozedurale Programmiererfahrung, kann die Vorstellung aufgebaut werden, Objekten seien »Hüllen« für Variablen („wrappers for variables“, vgl. ebd. S. 132).

Idee zur Lösung Die Lösung liegt direkt auf der Hand: von Anfang an sollten nur solche Beispiele verwendet werden, die mehr als ein Attribut verwalten. Dabei sollte auch Wert darauf gelegt werden, dass die Typen dieser Variablen unterschiedlich sind, um hier nicht die Vorstellung zu wecken, alle Attribute müssten vom selben

Typ sein (vgl. ebd. S. 132). In der Regel kann ein Attribut `name` eingefügt werden, das aber nicht mit dem Namen des Objekts oder dem Namen der Klasse verwechselt werden darf (s. Bsp. 1).

6. Objekte sind so etwas wie Datenbanken (records) Werden am Anfang häufig nur Getter- und Setter-Methoden programmiert und genutzt, kann schnell der Eindruck entstehen, in Objekten würden Werte gespeichert werden, die dann abgerufen werden können. HOLLAND et. al. sprechen von „simple, inert records“ [Holland Griffiths et. al. 1997, S. 132].

Idee zur Lösung Um diese Fehlvorstellung zu vermeiden, müssen Objekte so eingesetzt werden, dass sie sich je nach Zustand unterschiedlich verhalten (veränderte Rückgaben, verändertes Verhalten bei Methodenaufruf etc.). Um im Beispiel des Kontos (s. o.) zu bleiben, könnte das Konto Abbuchungen zurückweisen, wenn es nicht ausreichend gedeckt wäre oder das Überziehungslimit unterschritten werden würde.

7. In Methoden passiert die Arbeit durch das Assignment Diese Fehlvorstellung tritt auf, wenn Beispiele nur so eingesetzt werden, dass Methodenaufrufe hauptsächlich auf unveränderbaren Objekten arbeiten. Methodenaufrufe, die nur Zahlwerte ändern oder zurückgeben, wecken den Eindruck, dass dies die sei einzige Art von Arbeit sei, die in Methoden passiert (vgl. [Holland Griffiths et. al. 1997, S. 132]).

HOLLAND, GRIFFITHS und WOODMAN haben diese Fehlvorstellung auch bei erfahrenen Studierenden beobachten können. Wenn Schülerinnen und Schüler versuchten, mit Zuweisungen statt mit Objekten zu programmieren, könne diese Fehlvorstellung zu einem prozeduralem Stil führen.

Idee zur Lösung Beispiele sollten demnach so ausgewählt werden, dass auch Attribute vorhanden sind, die einen eigenen Zustand haben. Werden die Attributwerte verändert bzw. wird der Attributzustand durch Methoden von außen verändert, ist erkennbar, dass in Methoden nicht nur Zuweisungen (assignments) ausgeführt werden.

2.3.4 Fehlvorstellung: Verwechslung »Klasse« und »Objekt«

Im Folgenden soll noch eine der klassischen Fehlvorstellungen zur Objektorientierung näher beleuchtet werden: Schülerinnen und Schüler neigen dazu, die beiden

Begriffe »Klasse« und »Objekt« synonym zu gebrauchen, obwohl sie völlig unterschiedliche Dinge meinen. Ein Ziel des Ansatzes dieser Masterarbeit ist es auch, diese Fehlvorstellung zu vermeiden. Sie sei „[e]ine in jüngerer Zeit häufig gefundene Fehlvorstellung“ [Humbert 2006, S. 156].

Diese Fehlvorstellung ergibt sich aus einem verzerrtem Bild der OOM und OOP: In der OOM wird von Objekten gesprochen, wobei dann Klassen programmiert und instanziert werden. Die Akteure (die Instanzen der Klassen, sprich die Objekte) treten so nur noch in zweiter Linie in den Vordergrund, nicht aber bei der eigentlichen Programmierung. „Somit kann leicht Verwirrung bei Schülern bei der Unterscheidung zwischen Klassen und Objekten entstehen“ [Diethelm, S. 47].

ECKERDAL und THUNÉ berichten in [Eckerdal Thuné 2005] von ihrer qualitativen, phänomenologischen Interview-Studie, die sie an der schwedischen Universität zu Uppsala durchgeführt haben. Sie befragten 14 Studierende¹³, die gerade ihren ersten (verpflichtenden) Programmierkurs abgeschlossen hatten (vgl. [Eckerdal Thuné 2005, S. 89, S. 90]), zu den Phänomenen »Klasse« und »Objekt«. So sollten die Studierenden bspw. erklären, was für sie eine Klasse oder ein Objekt ausmache.

In der Auswertung konnten die Autoren zeigen, dass das Verständnis der Studierenden in drei Stufen sowohl beim Phänomen »Klasse« als auch beim Phänomen »Objekt« einzuteilen ist. Diese Einteilung reicht von einem grundlegenden Verständnis rein auf Programmierebene bis zu einem Verständnis darüber, dass Klassen und Objekte in Bezug zur realen Welt stehen. Sie verwendeten dabei die in den Tabellen 2.1 und 2.2 aufgeführte Kategorienbildung.

Objektverständnis

Stufe 1: Objekte werden wie (ein Abschnitt) Code verstanden.

Stufe 2: Wie vor, mit der Ergänzung, dass Objekte als etwas »Aktives« im Programm verstanden werden.

Stufe 3: Wie vor, mit der Ergänzung, dass Objekte auch als Modell eines realen Phänomens verstanden werden.

Tabelle 2.1: Kategorienbildung für gezeigtes Verständnis zum Phänomen »Objekt« (vgl. [Eckerdal Thuné 2005, S. 91]).

Es stellte sich in der Studie heraus, dass die Studierenden oft ein stufen-gleiches Verständnis von Klassen und Objekten zeigten (vgl. [Eckerdal Thuné 2005, S. 92]).

¹³ Die Autoren weisen explizit darauf hin, dass es sich um eine qualitative Studie handle und sie nicht auf statistisch verwertbare Ergebnisse abzielen würden (vgl. [Eckerdal Thuné 2005, S. 90]).

Klassenverständnis

Stufe 1: Klassen werden als Entitäten im Programm verstanden, die zur Struktur des Codes beitragen.

Stufe 2: Wie vor, mit der Ergänzung, dass Klassen eine Beschreibung für Eigenschaften und Verhalten eines Objekts bereitstellen.

Stufe 3: Wie vor, mit der Ergänzung, dass Klassen mit ihrer Beschreibung für Eigenschaften und Verhalten eines Objekts zur Modellierung einer realen Situation beitragen.

Tabelle 2.2: Kategorienbildung für gezeigtes Verständnis zum Phänomen »Klasse« (vgl. [Eckerdal Thuné 2005, S. 91]).

Daraus leiten ECKERDAL und THUNÉ Anforderungen an die Lehre bzw. den Unterricht ab, die wie folgt zusammenzufassen sind:

- Um die in Kapitel 2.3.3 beschriebenen Fehlvorstellungen »Die Zustandsfelder von Objekten definieren ihre Attribute« und »Objekte sind so etwas wie Variablen« zu vermeiden, sei ein Fokus auf die vorgestellten Lösungsansätze sinnvoll. So könne ein Verständnis der zweiten Stufe der Konzepte »Klasse« und »Objekt« bei den Studierenden erreicht werden.
- Die hier beschriebene Fehlvorstellung wird auch in [Holland Griffiths et. al. 1997, S. 132] beschrieben. Sie schlagen zur Lösung stets die Arbeit mit mehreren Instanzen einer Klasse vor. Die so erzeugte Variation mache dabei den Unterschied deutlich (vgl. [Eckerdal Thuné 2005, S. 92]). ECKERDAL und THUNÉ sind derselben Meinung und betonen dies mit folgenden Worten:

[...] variation in dimensions corresponding to critical aspects of the understanding is of great importance [...].

[Eckerdal Thuné 2005, S. 92]

- Es sei generell von großer Bedeutung, die Objektorientierung so zu unterrichten, dass eine Trennung der Konzepte möglich sei (vgl. [Eckerdal Thuné 2005, S. 93]). Dabei bestärken die Autoren die oben dargelegten Forderungen von HOLLAND et. al.

Kapitel 3

Die Hardware-Bausteine

Neben einer Erklärung der Idee des Hardwareeinsatzes werden auch die einzelnen Bauteile vorgestellt. Daneben werden Informationen zum Raspberry Pi gegeben und beispielhaft gezeigt, wie Objekte für die Bauteile zu erzeugen und zu manipulieren sind.

3.1 Idee des Hardwareeinsatzes

In Kapitel 2 wurden u. a. Probleme klassischer Herangehensweisen an die OOP dargestellt. Eine zentrale Rolle spielte dabei immer das Verständnis von Klassen und Objekten. Schülerinnen und Schüler wissen oft nicht um die Unterschiede zwischen den beiden Begriffen und verwechseln diese (vgl. Kapitel 2.3.4). Ein Ziel des hier vorgestellten Ansatzes ist es, eine größere Trennschärfe zu erreichen.

Ein anderes Problem, das sich auch bei verschiedenen Ansätzen ergibt, ist das große Abstraktionsvermögen, das die Schülerinnen und Schüler für die Lösung anfänglicher Probleme aufbringen müssen (vgl. [Or-Bach Lavy 2004, S. 82] und Kapitel 2.1.1). Der Hardware-Ansatz versucht, von Anfang an für Schülerinnen und Schüler durchschaubar zu sein und nicht mit hohen Konzepten anzufangen. Eine langsame Herangehensweise an die Themen der OOP soll gewährleisten, dass sich auch schwächere Schülerinnen und Schüler mit den Themen beschäftigen können.

Die Idee des Hardwareeinsatzes ist, für jedes Objekt auch einen greifbaren Hardware-Baustein bereitzustellen. So interagieren die Schülerinnen und Schüler mit der Objektorientierung nicht nur virtuell, sondern ganz real. Bei dem Ansatz sind Objekte und ihr jeweiliger Zustand in der realen Welt erfahrbar, indem ihr Zustand optisch oder akustisch wahrgenommen werden kann. Auch die Kommunikation zwischen den Objekten kann so sehr gut verdeutlicht werden, da nicht nur am

Bildschirm gearbeitet wird, sondern physikalische Taster gedrückt werden, Regler verdreht werden oder Licht auf Phototransistoren geleuchtet wird.

Die Idee ist in Abbildung 3.1 noch einmal visualisiert. Von einer gegebenen Klasse wurden drei Objekte erzeugt. Die Hardware-Bausteine repräsentieren zu jedem Zeitpunkt die Objekte und ermöglichen dem Beobachter ein Ablesen der Zustände von außen, auch ohne Methoden auf den Objekten aufzurufen.

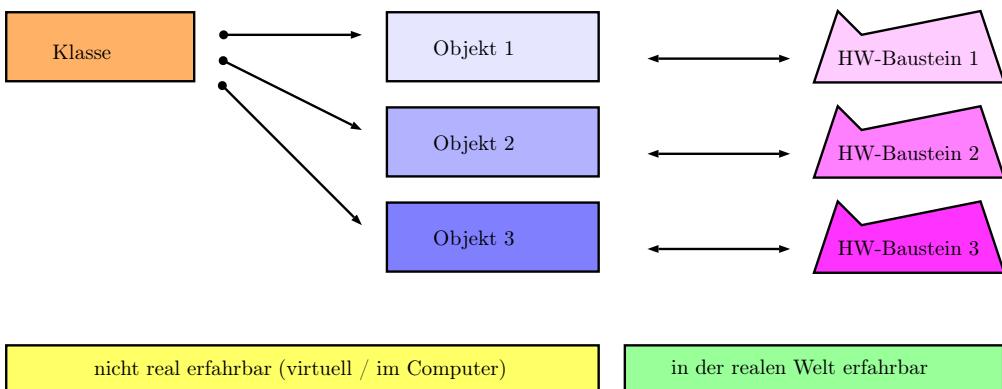


Abbildung 3.1: Idee des Hardwareeinsatzes. Klasse und Objekt sind normalerweise nur virtuell zu erfahren; durch den hier vorgestellten Ansatz ist für jedes Objekt ein Hardware-Baustein vorhanden, mit dem Schülerinnen und Schüler interagieren können, um so die Objektorientierung zu begreifen.

Die Klassen für die Bausteine wurden für den Unterricht vorbereitet und werden in ihrer Gesamtheit als »rpCollection« bezeichnet. Dabei steht »rp« für »Raspberry Pi«.

3.1.1 Der Raspberry Pi

Die Entwicklung des Raspberry Pi wurde ins Leben gerufen, um Studenten der Universität Cambridge die Informatik näher zu bringen, da dort immer geringere Kenntnisse auf Hardware-Ebene festgestellt wurden (vgl. [Dembowski 2015, S. 2]). Der Raspberry Pi besticht durch seine kleine Größe (ca. 9 cm × 6 cm) und seinen geringen Preis (vgl. [Dembowski 2015, S. 1, 3]). Abbildung 3.2 zeigt den Raspberry Pi 3 Model B (die derzeit aktuelle Version).

Der Raspberry Pi wird von der gemeinnützigen Organisation »Raspberry Pi Foundation« entwickelt. Er wird über einen Micro-USB-Anschluss mit Strom versorgt und verfügt (in seiner neuesten Generation) über einen Netzwerkanschluss, W-LAN und vier USB-Anschlüsse. Daneben ist noch ein Audio-Anschluss und ein HDMI-Anschluss vorhanden¹, sodass er sich sowohl an einem moderneren Monitor als auch

¹ Unter [RS Components] ist eine vollständige Liste der Anschlussmöglichkeiten aufgeführt.



Abbildung 3.2: Raspberry Pi 3 Model B. Bildquelle: [RS Components Bild].

an einem Fernseher betreiben lässt (vgl. [Dembowski 2015, Abb. ohne Nr.; S. 4]). Die Kosten für eine lauffähige Erstaustattung belaufen sich somit hauptsächlich auf den Raspberry Pi an sich, der derzeit bei ca. 35 Euro liegt (vgl. [RS Components]; der Vertrieb des Raspberry Pi wird von der Firma RS Components übernommen). Es muss lediglich eine Micro-SD-Karte separat erworben werden, die als Festplatte für den Raspberry Pi dient (vgl. [Dembowski 2015, S. 1]). DEMBOWSKI sieht die geringen Anschaffungskosten dabei als Grund, junge Menschen an die Arbeitsweise von und mit Computern heranzuführen (vgl. [Dembowski 2015, S. 4f]).

Der Name »Raspberry« steht in der (Cambridge-)Tradition, Computer nach Früchten zu benennen. »Pi« steht für »Python interpreter« (und nicht für engl. »Pie«, Kuchen), da das System auf Python als erste Programmiersprache setzt (vgl. [Dembowski 2015, S. 3]). Die Himbeere hat es auch zum Logo der Raspberry Pi Foundation geschafft (Abbildung 3.3).

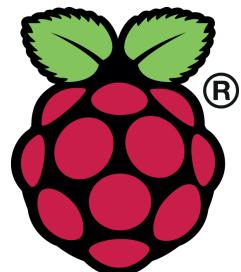


Abbildung 3.3: Das prominente Himbeer-Logo der Raspberry Pi Foundation. Bildquelle: [Raspberry Pi Trademark].

Heutzutage hat sich um den Raspberry Pi eine aktive Community gebildet, die auch viel im Open-Source-Bereich arbeitet. Insgesamt wurden schon über 10 Millionen Exemplare verkauft (vgl. [Raspberry Pi Blog]).

3.1.2 Die Bauteile

Wie schon erwähnt, wurden für den Unterricht Java-Klassen erstellt, mit denen sich die Hardware steuern lässt². Für jedes Bauteil gibt es eine eigene Klasse. Bei der Entwicklung wurde Wert darauf gelegt, dass es wirklich nur *eine* .java-Datei für einen Hardware-Baustein gibt, damit Schülerinnen und Schüler – falls sie sich die Klassen einmal ansehen sollten – die Klasse als Einheit verstehen. Die Entwicklung verzichtet dabei auf Vererbungsstrukturen aus genannten Gründen, auch wenn man Methoden zum Setzen eines Pins oder für Funktionalitäten, die bei mehreren Bauteilen vorkommen, gut in eine Oberklasse hätte auslagern können.

Die Bauteile sind für den Unterricht auf kleine Platinen gelötet worden, um sie den Schülerinnen und Schülern so als eine Einheit übergeben zu können (mehr dazu in Anhang C). Abbildung 3.4 zeigt beispielhaft ein paar am GPIO angeschlossene Bauteile, weitere Fotos befinden sich in Anhang C.8.

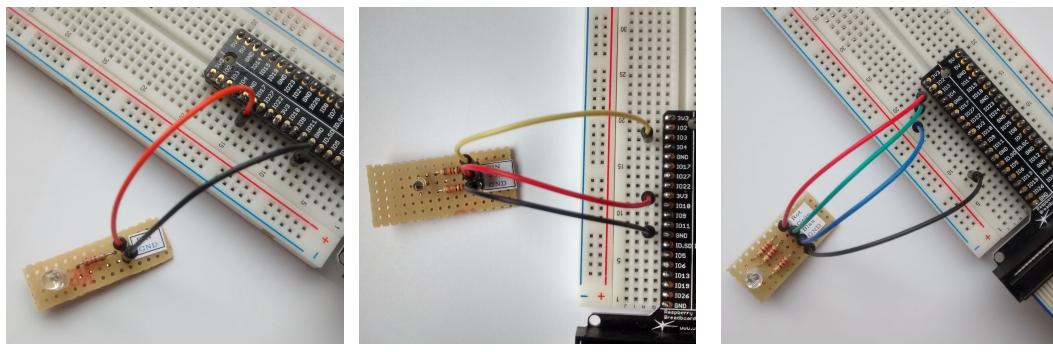


Abbildung 3.4: Angeschlossene Bausteine am Steckbrett des Raspberry Pi.

Beim Anschließen der Bauteile muss auf das korrekte Verbinden der Pinne geachtet werden (vgl. auch das Informationsblatt auf S. 119f). In der Regel müssen zwei oder drei Kabel am Steckbrett des Raspberry Pi angeschlossen werden, um ein Bauteil verwenden zu können. Dabei sind auf den Bauteilen farbige Aufkleber angebracht, um den Schülerinnen und Schülern eine Hilfestellung beim Verbinden zu geben. Tabelle 3.1 zeigt dies beispielhaft für den Anschluss eines Phototransistors.

² Die Quelltexte sind frei zugänglich in einem Git-Repository veröffentlicht: <https://github.com/hnrstrck/rpCollection>. Dort findet sich auch die Dokumentation der Klassen. Nähere Information sind in Anhang C (Bauteile) und Anhang E (Quelltexte u. weitere Materialien) dieser Arbeit vermerkt.

| Phototransistor | | GPIO |
|-----------------|---|------|
| PIN | ↔ | IO11 |
| +3,3V | ↔ | 3V3 |
| GND | ↔ | GND |

Tabelle 3.1: Anschluss eines Phototransistors an Pin 11.

Insgesamt stehen folgende Bauteile für den Unterricht bereit. Tabelle 3.2 zeigt die Anzahl der vorhandenen Bauteile für den Unterricht.

Diode (LED) Die Diode eignet sich für visuelle Rückmeldungen (blitzen, an / aus). Der Zustand eines Objekts ist direkt erkennbar. Die Farbe der Diode ist dabei vorgegeben (durch das bunte Plastik) und kann nicht geändert werden.

RGB-LED Die RGB-LED ist etwas umfangreicher als die Diode, kann aber für sich genommen mehr Farben darstellen. Es wäre denkbar, den Farben der RGB-LED in einem Programm besondere Bedeutungen zuzuordnen (Bestätigung, Ablehnung).

Phototransistor Der Phototransistor reagiert auf Lichtsignale und gibt eine Rückmeldung darüber, ob gerade Licht auf ihn fällt oder nicht. Er eignet sich gut um Lichtschranken zu bauen oder Methoden lichtgesteuert aufzurufen.

Taster Der Taster reagiert ebenfalls auf Einflüsse von außen. Taster eignen sich gut, wenn Aktionen auf Knopfdruck abgebrochen oder gestartet werden sollen (bspw. dreht sich ein Motor, solange ein Taster nicht gedrückt wurde). Bei diesem Bauteil handelt es sich allerdings nicht um einen Schalter (d. h. der Taster bleibt – einmal gedrückt – nicht in der gedrückten Position, sondern springt wieder hoch). Er funktioniert wie eine Türklingel, nicht wie ein Lichtschalter.

Summer Der Summer eignet sich für einfache akustische Signale, ähnlich wie die Diode dies visuell ermöglicht. Es handelt sich hierbei natürlich nicht um einen Lautsprecher, aber das Summen (Beepen) kann als Rückmeldung gedeutet werden.

Analog-Digital-Wandler Der AD-Wandler macht es möglich, analoge Signale am Raspberry Pi auszulesen. Da die GPIO-Pinne nur über die Zustände 0 (aus) und 1 (an) verfügen, ist es nötig, einen speziellen Mikrochip vorzuschalten, der analoge Signale über eine spezielle Schnittstelle umwandeln kann.

Am AD-Wandler hängen jeweils vier drehbare Potentiometer, die – je nach Stand – Werte zwischen 0 und 4095 (bzw. für Prozentwerte zwischen 0 und 100) zurück geben. So lässt sich beispielsweise die Drehgeschwindigkeit eines Motors oder die Blinkfrequenz einer Diode (beinahe) stufenlos regeln.

Motor (Motorsteuerung) Das Bauteil für die Steuerung von Motoren eignet sich für den Anschluss von drei Fischertechnik-Motoren. Motoren lassen sich an- und ausschalten und in ihrer Drehrichtung steuern (vorwärts, rückwärts). Auch Steuerungen für einen bestimmten Zeitraum (in Sekunden) oder eine bestimmte Leistung (in Prozent) sind möglich (realisiert über eine Pulswidenmodulation der entsprechenden Pinne).

| Bauteil | Anzahl |
|--------------------------------------|--------|
| Diode (versch. Farben ³) | 52 |
| RGB-LED | 10 |
| Phototransistor | 10 |
| Taster | 10 |
| Summer | 4 |
| AD-Wandler (mit jew. 4 Reglern) | 4 |
| Motorsteuerung | 2 |

Tabelle 3.2: Anzahl der vorhandenen Bauteile.

3.2 Erhoffte Vorteile des Hardwareeinsatzes

Der Hardwareeinsatz kann gegenüber einem klassischen objektorientierten Unterricht wahrscheinlich in folgenden Punkten überzeugen. Es soll generell versucht werden, den Schülerinnen und Schülern nicht mit abstrakten oder Programmiersprachen-eigenen Problemen zu begegnen, sondern von Anfang an schülerorientierte und alltagsbezogene Beispiele, Vorgehensweisen und Begründungen zu geben.

Konzentration auf das Wesentliche Ein Problem bei der Programmierung von Java-Klassen ist der Umfang an neuen Begriffen, mit dem Anfänger konfrontiert werden. [Kölling Rosenberg 2001, S. 2f] erwähnen, dass man nicht mit `main(...)` anfangen sollte, aber auch wenn man sich nur eine einfache Klasse wie in Quelltext 3.1

³ 10 Rot, 10 Blau, 10 Gelb, 10 Grün, 10 Weiß, 1 Orange, 1 Mintgrün

anguckt, tauchen neben den geschweiften Klammern auch Begriffe wie `public`, `class`, `static` oder `RPDiode` und `RPDiode()` auf.

Würde man komplett auf die `main(...)`-Methode verzichten und alles als Klassenkonstruktor schreiben, würde immer noch die Besonderheit des Konstruktors eine Rolle spielen. Solche Besonderheiten werden aber erst später für die Schülerinnen und Schüler relevant.

Der Ansatz dieser Masterarbeit umgeht die Notwendigkeit, solche Themen evtl. (auf Nachfrage) im Unterricht thematisieren zu müssen. Die Programmiersprache Groovy⁴ erlaubt es, in der gewohnten Java-Syntax zu arbeiten und dabei nicht die `main(...)`-Methode verwenden zu müssen oder Objekte zu typisieren. Programmiert wird in der Groovy-Console (Abbildung 3.6).

Um im Beispiel zu bleiben, sind im vorgestellten Ansatz nur die Zeilen 5–7 in die Groovy-Console einzugeben, um ein Objekt der Klasse `RPDiode` zu erzeugen, einen Pin anzugeben und die Diode blinken zu lassen (s. Kapitel 3.4). Damit wird die »Guideline Nr. 5« von [Kölling Rosenberg 2001] aufgegriffen (vgl. Kapitel 2.2.1), da man sich auf das Wesentliche beschränkt. An den angegebenen Zeilen erkennt man, dass beim Hardware-Ansatz auch der Unterschied zwischen Deklaration und Initialisierung einer Variablen auf einen späteren Unterrichtszeitpunkt verschoben wird (Zeilen 4 u. 5), da die Objekte in Groovy ungetypt sind.

```
1 public class manipulator
2 {
3     public static void main(String[] args){
4         RPDiode meinLicht;
5         meinLicht = new RPDiode();
6         meinLicht.setPin(17);
7         meinLicht.blinke();
8     }
9 }
```

Quelltext 3.1: Eine einfache Klasse.

Die Bedeutung der Programmiersprache wird auch im Ansatz von COOPER, DANN et. al. betont. Sie führen die Objektorientierung mit 3D-Modellen ein (vgl. Kapitel 2.2.1) und nutzen auch einen Editor / eine Programmierumgebung, die nicht sonderlich streng ist im Hinblick auf die Syntax:

Students are able to focus on the concepts of objects and encapsulation, rather than dealing with the frustration of parentheses, commas, and semicolons.

[Cooper Dann et. al. 2003, S. 2]

⁴ Groovy – The Groovy programming language – <http://www.groovy-lang.org/>.

Auch bei dem hier vorgestellten Ansatz sollen positive Erfahrungen mit den Hardware-Bausteinen im Vordergrund stehen. Die Schülerinnen und Schüler sollen sich um das Wesentliche kümmern und ein Verständnis der Objektorientierung erlangen.

Gesteigerte Motivation durch Hardware Schon Friedrich FRÖBEL, deutscher Pädagoge und Schüler PESTALOZZIS, war 1837 der Ansicht, dass das Konkrete vor dem Abstrakten eingeführt werden sollte (vgl. [Pollard Duvall 2006, S. 2]). Die Hardware-Bausteine bieten die Möglichkeit, das Konzept »Objekt« zu begreifen, indem Schülerinnen und Schüler Objekte buchstäblich anfassen und mit ihnen experimentieren können.

POLLARD und DUVALL versprechen sich von anschaulichem Informatikunterricht (umgesetzt durch Spiele, Spielzeuge und Geschichten) auch eine Motivation abseits der Noten und dass auch solche Schülerinnen und Schüler motiviert werden können, die weniger technisch versiert sind. Sie schreiben dazu:

In particular, we advocate incorporating teaching techniques reminiscent of kindergarten: games, toys, stories, and play. These techniques promote an active learning environment, level the playing field for non-technical students, provide motivation beyond grades, and make class time fun.

[Pollard Duvall 2006, S. 1]

Im Informatikunterricht lässt sich eine ganze Reihe von Konzepten durch konkrete, anzufassende Objekte beschreiben. Hierzu einige Beispiele aus [Pollard Duvall 2006, S. 2]:

- Hin- und Herwerfen von Frisbees zur Parameterweitergabe
- Stapelablagen für Arrays (vgl. [Poon 2000, S. 199])
- Gestapelte Becher für Stacks
- Magnetische Buchstaben zur Textdarstellung und Darstellung der Arbeitsweise von String-Funktionen
- Würfel zum Herstellen von Zufall

In den Zielen des Informatikunterrichts wird auch der Motivationswert konkreter, kreativer Informatiksysteme beschrieben, wie das folgende Zitat zeigt:

Die Konstruktion eines abstrakten Modells zu einer anwendungsbezogenen Problemstellung fördert das Abstraktionsvermögen sowie kreatives und strukturelles Denken. Die Umsetzung eines informatischen Modells in ein lauffähiges Informatiksystem hat für Schülerinnen und Schüler nicht nur einen hohen Motivationswert, sondern ermöglicht ihnen auch die eigenständige Überprüfung der Angemessenheit und Wirkung des Modells im Rückbezug auf die Problemstellung.

[Schulentwicklung NRW 2014, Abs. 5]

Die Bauteile eignen sich zur Umsetzung eines informatischen Modells, wie auch das Zitat fordert. Dabei können die Lösungen direkt von den Schülerinnen und Schülern überprüft werden, da die Kommunikation zwischen den Objekten visuell erkennbar abläuft. Natürlich ist ein rein experimentelles Vorgehen ohne Vorüberlegung (auch in der Informatik) nicht sinnvoll, trotzdem sollen die Hardware-Bausteine zum Experimentieren einladen, da sich viele Situationen finden lassen, in denen ein Objekt ein anderes beeinflusst. Die dabei stattfindende Kommunikation kann auch von den Schülerinnen und Schülern provoziert werden. Dies wird im nächsten Paragraphen weiter ausgeführt.

Sichtbare Kommunikation von Objekten Abbildung 3.5 zeigt die Kommunikation (Pfeile) zwischen den Objekten (Kreise), die die Schülerinnen und Schüler erfahren (sehen, hören) können. Durch das Anschließen der Bauteile am Steckbrett lassen sich verschiedene Szenarien modellieren. Die Schülerinnen und Schüler können bspw. Phototransistoren mit Licht bescheinen (LED-Blitz am Handy) oder Taster drücken, um direkt mit Objekten in Interaktion zu treten.

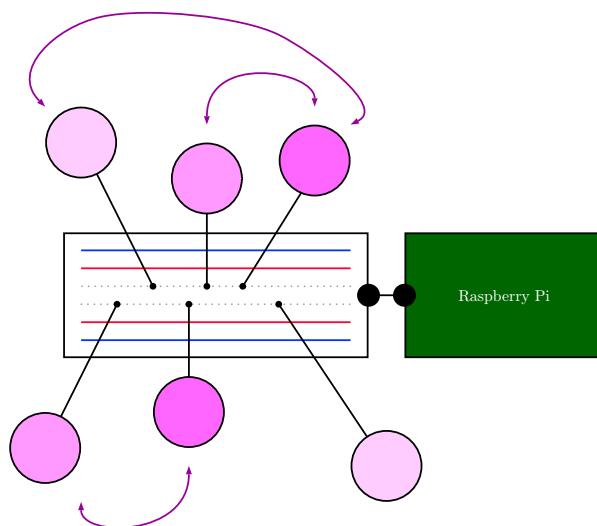


Abbildung 3.5: Angeschlossene Bauteile am Raspberry Pi in einer Stern Topologie. Die Pfeile sollen die sichtbare Kommunikation der Objekte symbolisieren.

Es lassen sich weitere Situationen finden, in denen Kommunikation zwischen Objekten erlebbar ist, wie folgende Aufzählung zeigt:

- Schalten einer Diode durch Betätigung eines Tasters
- Einstellen der Farben einer RGB-LED durch die Regler eines AD-Wandlers (die Farbe ändert sich nicht automatisch beim Drehen der Regler)
- Einstellen der Blinkfrequenz einer LED durch einen Regler eines AD-Wandlers (Stroboskop-Effekt; die Frequenz ändert sich beim Drehen des Reglers)

Allgemein lassen sich in Programmen natürlich direkt die Eingaben der Hardware-Bausteine nutzen, wie Quelltext 3.2 zeigt. In den Zeilen 5–8 wird der Ton eines Summers abgebrochen, wenn ein Taster betätigt wurde. Auch wenn dies kein Beispiel für die Umsetzung einer *rein objektorientierten* Problemsituation ist (da nicht immer wie hier eine Kontrollklasse vorliegt), lässt sich in diesem Fall der Zustand eines Objekts auslesen (Taster) und mit der Eingabe ein anderes schalten (Summer).

```
1 meinSummer = new RPSSummer();
2 meinSummer.setPin(17);
3 meinTaster = new RPTaster();
4 meinTaster.setPin(20);
5 while(meinTaster.istGedrueckt() == false){
6     meinSummer.an();
7 }
8 meinSummer.aus();
```

Quelltext 3.2: Der Summer tönt so lange, bis der Taster betätigt wurde.

Für den Unterricht sollen solche Problemsituationen besprochen werden, die ohne Wiederholungsanweisungen und andere Kontrollstrukturen (bedingte Anweisungen) auskommen (vgl. auch Kapitel 4.4). Den Schülerinnen und Schülern bleibt der Kontrollfluss so erst einmal verborgen und wird zu einem späteren Zeitpunkt eingeführt (vgl. Kapitel 4.1).

Voraussetzungen an Hard- und Software Neben dem Raspberry Pi, einem GPIO-Steckbrett-Adapter mit Flachbandkabel und einem Steckbrett halten sich die Kosten gering, um mit den Hardware-Bausteinen zu arbeiten. Die Software ist frei zugänglich (s. Anhang E bzw. Fußnote 2).

Natürlich bedarf es einiger Zeit, die Hardware vorzubereiten (Komponenten bestellen, Bausteine löten). Es ist aber denkbar, dass interessierte Schülerinnen und Schüler in AGs mithelfen, die Hardware-Bausteine zu erstellen.

3.3 Eventuelle Nachteile des Hardwareeinsatzes

Ein Nachteil könnte sein, dass Schülerinnen und Schüler nach dem Unterrichtsvorhaben eine verzerrte Vorstellung von Objekten haben. So könnte es passieren, dass sie sich zu sehr auf die Bauteile fixieren, sollen allgemein Objekte beschrieben werden. In der Software-Entwicklung kommt man immer wieder in Situationen, Klassen schreiben zu müssen, deren Objekte sich in erster Linie nicht begreifen lassen. Davon ausgehend, dass dies aber der erste Kontakt der Schülerinnen und Schüler mit dem Thema Objektorientierung ist, sollte dies ein eher kleines Problem sein. Der Zugang zu abstrakteren Beispielen sollte für die Schülerinnen und Schüler mehr Erfahrung nachvollziehbar sein.

3.4 Ansteuern der Hardware am Raspberry Pi

Im Folgenden soll beispielhaft gezeigt werden, wie eine Diode am Raspberry Pi angesteuert werden kann. Dies gelingt im Wesentlichen in drei Schritten: Anschließen des Bausteins (s. Kapitel 3.1.2), Kompilieren der .java-Datei und Objekterstellung und -manipulation in Groovy.

Kompilieren der .java-Datei(en) Es ist darauf zu achten, dass das Java-SDK und die Groovy-Version kompatibel sind. Ansonsten muss beim Kompilieren noch die Versionsnummer des Java-SDKs als Ziel angegeben werden (bspw. `-target 1.7`), damit es beim späteren Aufruf in Groovy nicht zu einem `major.minor version 51.0 error` kommt.

Ebenfalls ist darauf zu achten, dass der Java-Compiler Zugriff auf die Pi4J-Klassen hat (`-classpath ...`, s. auch Kapitel D.3). Damit sieht der Aufruf zum Kompilieren der Klasse RPDiode wie folgt aus:

```
1 javac -classpath '.:classes:/opt/pi4j/lib/*' RPDiode.java
```

Sollen gleich *alle* .java-Dateien in einem Verzeichnis kompiliert werden, so ist `RPDiode.java` durch `*.java` zu ersetzen.

Für die Schule wurden obige Befehle in einem Skript zusammengefasst, das auch das Starten der Groovy-Console ermöglicht. Der Aufruf erfolgt im Verzeichnis `rpCollection` durch die folgende Eingabe:

```
1 bash start.sh
```

Sollen die .java-Dateien vorher nicht kompiliert werden (die Groovy-Console startet dann schneller), so ist das Skript `GCstart.sh` auszuführen.

Starten der Groovy-Console Der Aufruf der Groovy-Console muss in dem Ordner geschehen, in dem auch die kompilierten .class-Dateien liegen. Die Groovy-Console muss als `sudo` gestartet werden und es müssen die Pfade für Pi4J übergeben werden⁵ (`-classpath ...`). Dies geschieht mit dem folgenden Aufruf:

```
1 sudo groovyConsole -classpath '.:classes:/opt/pi4j/lib/*'
```

Der Befehl

```
1 sudo groovysh -classpath '.:classes:/opt/pi4j/lib/*'
```

startet die Groovy-Shell, in der sich analog arbeiten lässt.

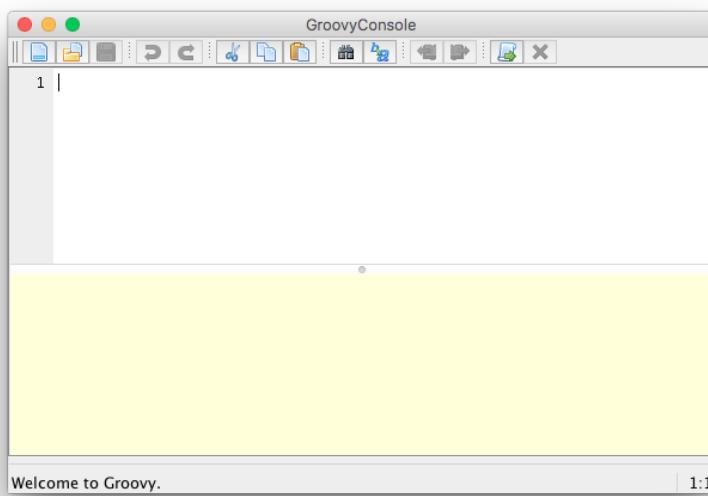


Abbildung 3.6: Startbildschirm der Groovy-Console.

Es kann ein paar Sekunden dauern, bis der Startbildschirm erscheint (Groovy-Console: Abb. 3.6, Groovy-Shell: Abb. 3.7). Eingaben in der Groovy-Console müssen entweder der korrekten Java-Syntax oder der etwas vereinfachten Groovy-Syntax entsprechen (s. auch Kapitel D.5).

Es sei an dieser Stelle angemerkt, dass die angegebenen Pfade je nach System abweichen können. Sie sind dann in den Aufrufen entsprechend zu ersetzen.

⁵ Arbeitet man häufiger in dieser Konfiguration, ist das Starten der Groovy-Console über das Start-Skript natürlich angenehmer.

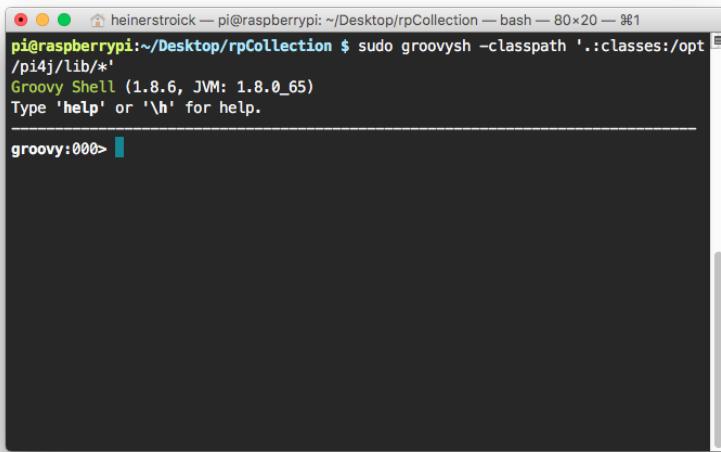


Abbildung 3.7: Startbildschirm der Groovy-Shell.

Hardware steuern Für Befehlseingaben wird die übliche Punktnotation verwendet, d. h. der Aufruf von Methoden erfolgt in dem in [Humbert 2006, S. 120] dargestellten Schema:

<objektbezeichner>.<methodenbezeichner>(<parameterwert>)

Durch die Befehlsfolge

```

1 meineDiode = new RPDiode();
2 meineDiode.setPin(17);
3 meineDiode.blinke();

```

wird ein Objekt der Klasse `RPDiode` erzeugt, welches die Diode blinken lässt. Der Pin kann auch direkt im Konstruktor übergeben werden (nicht dargestellt). Es wird eine Rückmeldung in der Konsole gegeben, ob die Objekterstellung und das Setzen des Pins erfolgreich war (Quelltext 3.3, Zeilen 2 u. 4f).

```

1 groovy:000> meineDiode = new RPDiode()
2 ====> RPDiode@641d23
3 groovy:000> meineDiode.setPin(17)
4 Output-Pin gesetzt:
5 Pin 17 gesetzt
6 ====> null
7 groovy:000> meineDiode.blinke()
8 ====> null

```

Quelltext 3.3: Ausgaben in der Groovy-Shell.

Die Möglichkeit einer Kommunikation zwischen Objekten muss natürlich programmiert werden. Das nächste Beispiel lässt die Diode leuchten, falls kein Licht auf den Phototransistor fällt (sonst bleibt die Diode aus). So ist ersichtlich, dass der Zustand eines Objekts von dem eines anderen abhängt und sich Objekte auch gegenseitig beeinflussen können.

```
1 meineDiode = new RPDiode();
2 meineDiode.setPin(17);
3
4 meinLichtsensor = new RPPhototransistor();
5 meinLichtsensor.setPin(20);
6
7 meineDiode.schalten(meinLichtsensor.hatLichteinfall());
8
9 Helfer.herunterfahren()
```

Quelltext 3.4: Schalten einer LED mit einem Phototransistor.

Damit Skripte in der Groovy-Console mehrfach hintereinander ausgeführt werden können, müssen die Pinne der Bauteile am Ende wieder frei gegeben werden. Dies geschieht über den statischen Methodenaufruf `Helfer.herunterfahren()` bei der Klasse `Helfer` (Quelltext 3.4, Zeile 9). Ohne diesen Aufruf kommt es zu Fehlern, da sonst die Pinne bei jedem Skriptaufruf mehrfach benutzt werden. Der Quelltext 3.4 kann auch mehrmals hintereinander ausgeführt werden, sodass die Schülerinnen und Schüler mit dem Phototransistor experimentieren können.

Kapitel 4

Unterrichtserfahrungen

Der Unterricht fand am Joseph-König-Gymnasium (JKG) in Haltern in einer 10. Klasse (Einführungsphase – EF) statt.¹ Der Kurs umfasste 22 Schülerinnen und Schüler.

4.1 Zielformulierungen für das Unterrichtsvorhaben

Für den Unterricht sind die Vorgaben des Kernlehrplans bindend. Dort wird aus Inhaltsfeld 1 »Daten und ihre Strukturierung«, Unterpunkt »OBJEKTE UND KLASSEN«, folgender Lerninhalt bis zum Ende der Einführungsphase vorgeschrieben:

Die Schülerinnen und Schüler

- ermitteln bei der Analyse einfacher Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M),
- modellieren Klassen mit ihren Attributen, ihren Methoden und Assoziationsbeziehungen (M),
- modellieren Klassen unter Verwendung von Vererbung (M),
- ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen oder lineare Datensammlungen zu (M),
- ordnen Klassen, Attributen und Methoden ihren Sichtbarkeitsbereich zu (M),
- stellen den Zustand eines Objekts dar (D),
- stellen die Kommunikation zwischen Objekten grafisch dar (M),
- stellen Klassen, Assoziations- und Vererbungsbeziehungen in Diagrammen grafisch dar (D),
- dokumentieren Klassen durch Beschreibung der Funktionalität der Methoden (D),

¹ Joseph-König-Gymnasium, Holtwicker Str. 3–5, 45721 Haltern am See, <http://www.joseph-koenig-gymnasium.de/>.

- analysieren und erläutern eine objektorientierte Modellierung (A),
- implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I).

[Kernlehrplan Informatik 2014, S. 22]

Dabei stehen die Buchstaben in den Klammern für die Kompetenzbereiche des Faches Informatik, die in Tabelle 4.1 angegeben sind.

| Abkürzung | Kompetenzbereich |
|------------------|-------------------------------|
| (A) | Argumentieren |
| (M) | Modellieren |
| (I) | Implementieren |
| (D) | Darstellen und Interpretieren |
| (K) | Kommunizieren und Kooperieren |

Tabelle 4.1: Kompetenzbereiche des Kernlehrplans (vgl. [Kernlehrplan Informatik 2014, S. 19–21]).

Im Vorschlag für ein schulinternes Curriculum der GI in [GI Curriculum 2016, S. 7f] werden folgende Unterrichtsvorhaben für die EF erklärt. Auch der schulinterne Lehrplan orientiert sich an dem Vorschlag der GI (vgl. [Schulint. Lehrplan, S. 4–6]).

1. Unterrichtsvorhaben EF-2, »Grundlagen der objektorientierten Analyse und Modellierung anhand von Beispielkontexten«
2. Unterrichtsvorhaben EF-3, »Datenschutz aus informatischer Perspektive«
3. Unterrichtsvorhaben EF-4, »Algorithmen zum Suchen und Sortieren«
4. Unterrichtsvorhaben EF-5, »Klassen und ihre Implementierung durch Umsetzung einer objektorientierten Modellierung«
5. Unterrichtsvorhaben EF-6, »Modellierung von Abläufen – Kontrollstrukturen«

Diesem Vorschlag folgend, wird in Unterrichtsvorhaben (UVH) EF-2 im Kontext von Objekten (Objektkarten, Objektdiagramme) und ihrem Zusammenspiel geblieben, bevor der Blick in UVH EF-5 auf die Klassen erweitert wird. Darin geht es auch um die Beantwortung der Frage, wo Objekte überhaupt herkommen.

Der Vorschlag sieht im UVH EF-2 noch keine Kontrollstrukturen vor, sodass der Quellcode aus linearen Sequenzen besteht, da „auf diese Weise eine Fokussierung auf die Grundlagen der Objektorientierung möglich [ist]“ [GI Curriculum 2016, S. 18]. Dieser Ansatz wird auch in der Masterarbeit verfolgt. Das erste Projekt kommt auch

ohne Kontrollstrukturen aus und ermöglicht die im Zitat angesprochene Fokussierung (s. Kapitel 4.4).

4.2 Objektverständnis

Aus den Forderungen des Lehrplans leitet sich ein erstes Objektverständnis ab, das die Schülerinnen und Schüler durch den Ansatz dieser Masterarbeit (und damit nach UVH EF-2) erlangt haben sollten. Dieses Objektverständnis wird auch durch die Interviews im nächsten Kapitel evaluiert. Konkret sollen die Schülerinnen und Schüler über folgende Fertigkeiten und Kompetenzen am Ende des Unterrichtsvorhabens verfügen:

- Die Schülerinnen und Schüler können für eine gegebene Geschichte (Problemstellung) Objekte, ihre Attribute, ihre Methoden und ihre Beziehungen ermitteln. Dazu können sie die Methode von ABBOTT anwenden (vgl. [Abbott 1983, S. 884f]) und das Objektspiel nutzen, um nicht benötigte Objekte oder Fehler in der Modellierung zu identifizieren.
- Sie können Objekte auf verschiedene Arten beschreiben und darstellen (textuell, Objektkarten, Objektdiagramm). Sie können den Zustand von Objekten in der Geschichte ermitteln und begründen.
- Sie kennen verschiedene Arten von Methoden (mit und ohne Parameter, mit und ohne explizite Rückgabe) und können die richtige Art von Methode bei einer Modellierung beschreiben.
- Sie können Unterschiede in verschiedenen Modellierungen analysieren und Vorteile und Nachteile einzelner Modellierungsentscheidungen erläutern.
- Sie können den Programmablauf (in der Groovy-Console) implementieren und nutzen dafür auch die Hardware-Bausteine.

4.3 Unterrichtsvorbereitung

Für den Unterricht sind Arbeitsblätter vorbereitet worden, um mit den Hardware-Bausteinen arbeiten zu können. In der Vorbereitungszeit wurden auch die Bausteine gelötet und die Klassen programmiert.

Zu Beginn des Unterrichtsvorhabens wurde auch die Raspberry-Pi-Bestellung durchgeführt, sodass die Schülerinnen und Schüler danach an *ihrem* Pi arbeiteten

konnten. Für die restliche Ausstattung konnte auf den Informatikraum zurückgegriffen werden (Monitore, Tastaturen, Mäuse, Steckbretter u. GPIO-Steckbrett-Adapter, Jumperkabel).

4.4 Welche Projekte sollen mit der Hardware umgesetzt werden?

Diese Masterarbeit begleitet vor allem das erste Projekt (Lichtanlage einer Theateraufführung). In Zukunft soll im Unterricht auch der Kranwagen mit der Hardware modelliert werden.

4.4.1 Lichtanlage einer Theateraufführung

Bei diesem Projekt soll die Lichtsteuerung für eine Theateraufführung modelliert werden. Da sehr viele Dioden zur Verfügung stehen, können die Schülerinnen und Schüler vielfach das Erstellen von Objekten und Aufrufen von Methoden üben. Die Einstiegsgeschichte orientiert sich nicht an der Nomenklatur der vorbereiteten Klassen, sondern lässt den Schülerinnen und Schülern den Freiraum, selbst Methoden zu benennen und Objekte zu identifizieren. Das Arbeitsblatt ist schon vorbereitet worden (Anhang S. 109).

Die bei diesem Projekt erstellten Objektkarten sind überschaubar (Abbildung 4.1), kommen dafür aber häufig vor. Die Schülerinnen und Schüler arbeiten bei diesem Projekt mit drei verschiedenen Bauteil-Typen (Diode, RGB-LED, Phototransistor).

| roterScheinwerfer |
|-------------------------|
| pin = 12 |
| status = "aus" |
| standort = "bühne" |
| schalten(status) |
| anschalten() |
| ausschalten() |
| setzeStandort(standort) |

Abbildung 4.1: Objektkarte zu einem roten Scheinwerfer.

Dieses Projekt kommt gänzlich ohne Kontroll- und Wiederholungsstrukturen aus, da zunächst die OOM im Mittelpunkt steht. Auch der Begriff »Klasse« muss im Unterricht nicht an dieser Stelle besprochen werden. Damit passt das Projekt in das UVH EF-2 des schulinternen Lehrplans (vgl. [Schulint. Lehrplan, S. 14–17])

und nicht behandelte Themen der Objektorientierung werden auf einen späteren Zeitpunkt des Unterrichts verschoben (UVH EF-5, [Schulint. Lehrplan, S. 22–24]).

4.4.2 Steuerung eines Hafenkrans

Anknüpfend an die gelernten Modellierungstechniken des ersten Projekts soll dann eine umfangreichere Problemstellung angegangen werden. In der Schule sind zwei große Fischertechnik-Hafenkräne vorhanden (Abbildung 4.2), die mit Motoren ausgestattet sind, um sich bspw. zu drehen oder den Arm zu heben und zu senken. Ziel des Projekts ist es, die Modellerungstechniken an einem größeren Projekt zu wiederholen und zu üben. Dabei können immer wieder Rückgriffe auf das erste Projekt gemacht werden (ABBOTT, Beziehungen, Methoden mit und ohne Rückgabewert etc.).



Abbildung 4.2: Hafenkran aus Fischertechnik, ausgestattet mit drei Motoren (drehen, Arm heben und senken, Seilwinde). Bildquelle: privat.

Bei diesem Projekt besteht auch die Möglichkeit, das Konzept »Klasse« einzuführen, sodass die Schülerinnen und Schüler nicht mehr rein auf Objekten arbeiten. Dabei ist zu bedenken, dass „[d]ie Abstraktion von Objekten und ihren Beziehungen zu Klassen [...] eine Metastruktur dar[stellt], die nicht trivial ist“ [Schulint. Lehrplan, S. 23]. Eine Klasse `Kran` könnte beispielsweise über drei Motoren und zwei Signal-Dioden verfügen und Methoden wie `dreheUmGradNachLinks(int pGrad)` haben, die von den Schülerinnen und Schülern selbst geschrieben werden. Vorschläge zur schulgerechten Umsetzung werden in UVH EF-5 des schulinternen Lehrplans gemacht (vgl. [Schulint. Lehrplan, S. 22–24]).

Bei der Arbeit mit dem Kran können die Schülerinnen und Schüler Kontroll- und Wiederholungsstrukturen und den Umgang mit Variablen kennenlernen (UVH EF-6, [Schulint. Lehrplan, S. 25f]). Das anfängliche Arbeitsblatt für dieses Projekt ist schon vorbereitet worden (Anhang S. 127).

4.4.3 Ausblick auf weitere Projekte

Mit der Hardware können viele Projekte bzw. Geschichten modelliert werden. In jedem Fall ist es wichtig, dass ausgewählte Projekte nicht konstruiert wirken. Sie sollten aus Bereichen kommen, die den Schülerinnen und Schülern bekannt sind, und immer auch die Möglichkeit zur Erweiterung bieten. Wenn sie sich mit Themen beschäftigen, die nichts mit der reinen Projektumsetzung an sich zu tun haben, können sie noch eine andere Sichtweise auf das Projekt eröffnen.

Darüber hinaus ist wichtig, dass die Projekte eine gewisse Grundgröße haben (vgl. Kapitel 2.2.2). So kann die Idee der Wiederverwendbarkeit von Klassen diskutiert werden. Einfach nur Lichter zum Blinken zu bringen ist dabei nicht ausreichend. Erst wenn Projekte bearbeitet werden, in denen die Objekte kommunizieren und es mehrere Objekte einer Klasse gibt, sind diese Punkte erfüllt.

Eventuell ließe sich sogar ein LEGO-Radlader mit dem Raspberry Pi und den Hardware-Bauteilen steuern (Abbildung 4.3). Auch hier kommt die Modularität der Objektorientierung insofern ins Spiel, als dass es gleich drei Motoren für den Radlader gibt, die gesteuert werden können (vorwärts fahren², Arm heben und senken, Schaufel kippen). Es könnte sicherlich an die Erfahrungen mit dem Hafenkran angeknüpft werden.



Abbildung 4.3: LEGO-Technic Radlader (Bausatz 42030). Bildquelle: [LEGO].

² Der Radlader kann nicht mit dem Raspberry Pi / den Motoren gelenkt werden, da der LEGO-Bausatz dies nicht vorsieht.

Denkbar wäre auch ein Einsatz der Hardware im Eisenbahnumfeld (Bahn- und Schrankensteuerung). Die Schranken an einer Bahnstrecke senken sich automatisch, wenn der Zug einen Sensor in der Schiene überfährt (Lichtschranke) und der Zug hält an, wenn die Schranken nicht korrekt geschlossen werden konnten (Taster). So kommuniziert der Sensor im Gleisbett mit den Schranken und diese wiederum mit dem Zug. Interessant ist hierbei der Punkt, ob und wie festgestellt werden kann, ob die Schranke wirklich geschlossen ist.

4.5 Bericht der einzelnen Unterrichtsstunden

Im Folgenden wird von den Unterrichtsstunden am Joseph-König-Gymnasium berichtet. Der Unterricht wurde seit Ende der Herbstferien 2016 begleitet und vom Fachlehrer durchgeführt.

| Nr. | Datum | Thema |
|-----|--------------------------|---|
| 1 | Einzelstd. 26.10.2016 | Kennenlernen des Verfahrens von ABBOTT und erste Problemanalyse am Beispiel der Theateraufführung |

Die Schülerinnen und Schüler lasen den Text zum Verfahren von ABBOTT (Anhang S. 107) und begannen dann an der Geschichte zur Theateraufführung (Anhang S. 109) die Substantive, Verben und Adjektive herauszufinden und die Kandidaten für Objekte, Methoden und Attribute in einer Tabelle festzuhalten. Der Lehrer erklärte dabei, dass das Ziel der Arbeit die Erstellung eines Modells sei, das dann am Rechner in ein Programm überführt werden könne. Der Begriff des Modells wurde von den Schülern als eine „Veranschaulichung“ erklärt, die etwas „vereinfacht darstellt“. Einige bezeichneten ein Modell auch als „Abbildung der Realität“.

Die Schülerlösungen wichen dabei stark voneinander ab. Zu Beginn wussten einige Schülerinnen und Schüler nicht mit Begriffen wie »Martin« oder »Zeit« umzugehen, nahmen sie aber dann doch in die Tabelle auf, da es sich nur um Kandidaten handeln sollte.

| Nr. | Datum | Thema |
|-----|--------------------------|---|
| 2 | Doppelstd. 27.10.2016 | Identifizieren von Objekten aus einer gegebenen Problemstellung und Erstellung von Objektkarten |

In der Doppelstunde schrieben die Schülerinnen und Schüler zu Beginn ihre identifizierten Objekte, Methoden und Attribute mit Attributwerten in eine große Tabelle an der Tafel (vgl. Anhang A.2.1), die dann im Plenum besprochen wurde. Dabei

wurden Begriffe wie »Zeit« oder »Szene« gestrichen, da diese Begriffe im Regelfall zu abstrakt für die Modellierung sind. Einige Methoden wurden auch gestrichen, da diese mehr Zustandsbeschreibungen als Handlungsaufforderungen waren (vgl. Anhang A.2.1).

Die Schülerinnen und Schüler suchten sich dann ein Objekt aus und modellierten Objektkarten für dieses Objekt. Ziel war es, das Objekt zum Zeitpunkt am Ende des Textes zu modellieren (die Attributwerte ändern sich während des Textverlaufs). Drei Lösungen wurden vorgestellt (vgl. Anhang A.2.2, Abbildungen A.1–A.3). Einige Schüler diskutierten, ob Methoden wie `blinke(laenge)` wirklich nötig seien, da man dies auch durch wiederholtes ein- und ausschalten erreichen könnte. Es wurde auch vorgeschlagen, Methoden wie `einschalten()` und `ausschalten()` zu einer Methode zusammenzufassen (bspw. `wechseln()`). Trotzdem wurde zunächst die umfangreichere Modellierung gewählt, da das Vorgehen von ABBOTT im Vordergrund stand und nicht die Implementierung.

Es gab Schülerinnen und Schüler, die Probleme beim Identifizieren hatten oder noch nicht trennen konnten, was Objekte und was keine Objekte sind. Diesen wurde der Leitspruch an die Hand gegeben, dass man die identifizierten Objekte nach ABBOTT „anfassen“ könne.

Nach der Besprechung der Objektkarten modellierten die Schülerinnen und Schüler sämtliche Objekte des Textes in Partnerarbeit. Im Text sind insgesamt 10 Objekte erwähnt (5 Scheinwerfer, RGB-Scheinwerfer, Lichtsensor, (spezielle) Hintergrundbeleuchtung, Martin, Bühne). Hausaufgabe war es, diese Arbeit fertig zu stellen.

| Nr. | Datum | Thema |
|-----|--------------------------|---|
| 3 | Einzelstd. 02.11.2016 | Kennenlernen des Objektspiels als Methode zur Überprüfung eines Modellierungsansatzes |

Zu Beginn der Stunde wurden große, lamierte Objektkarten ausgeteilt (Anhang S. 111f), auf die die Schülerinnen und Schüler in Partnerarbeit jeweils ein Objekt aus der Hausaufgabe von der letzten Stunde übertrugen. Auf die Vorderseite schrieben sie den Namen des Objekts und die Methoden, auf die Rückseite Attribute und Attributwerte. Die Aufteilung in Vorder- und Rückseite ist eine Anspielung auf die öffentliche Sichtbarkeit der Methoden nach außen und die private Sichtbarkeit von Attributen (die Seite zeigt auf den Körper, wenn man die Objektkarte vor sich hält).

Es wurde erklärt, dass das Spiel dazu diene, das Objektdiagramm, das in den Hausaufgaben entstanden war, auf Vollständigkeit und Konsistenz zu prüfen. Das

Objektspiel eignet sich prinzipiell dazu, fehlende Methoden zu identifizieren oder fehlende Attribute auszumachen. Auch kann das Spiel dazu dienen, eine einheitliche Nomenklatur für Attribute und Methoden einzuführen. Die Schülerinnen und Schüler wurden auch darauf aufmerksam gemacht, dass das Spiel immer mal wieder im Unterricht an verschiedenen Stellen vorkäme.

Dann begannen die Schülerinnen und Schüler das Spiel durchzuführen. Jedes im Text identifizierte Objekt wurde dabei von einer Schülerin oder einem Schüler gespielt. Dabei wurde auch eine neue Sprechweise eingeführt, die die folgende Form hatte:

- S1: Ich, (das Objekt) `martin`, rufe beim Objekt `blauerScheinwerfer` die Methode `anschalten()` auf.
- S2: Ich, (das Objekt) `blauerScheinwerfer`, ändere meinen Attributwert des Attributs `status` auf `an` und gebe zurück an `martin`.
- S1: Ich, `martin`, rufe beim Objekt...

Dabei waren kleinere Ungenauigkeiten zu Beginn des Spiels zu beobachten. Einige Schülerinnen und Schüler sagten z. B. „Ich rufe beim blauen Scheinwerfer ... auf“, obwohl es kein solches Objekt gab. Während des Spielens stellte sich u. a. heraus, dass das Objekt `martin` die Methode `warten(zeit)` benötigte (vgl. Anhang A.2.3). Auch die Sprechweise

- S1: Ich, `martin`, rufe bei mir selbst die Methode `warten(...)` mit dem Parameterwert `10` für den Parameter `zeit` auf.

wurde an dieser Stelle neu eingeführt.

| Nr. | Datum | Thema |
|-----|--------------------------|---|
| 4 | Doppelstd. 03.11.2016 | Durchführung des Objektspiels und Überführung des Protokolls in ein Sequenzdiagramm |

Anknüpfend an das Objektspiel wurde zu Beginn dieser Stunde das Informationsblatt zum Objektspiel ausgeteilt (vgl. Anhang S. 115). Dann fanden sich die Schülerinnen und Schüler wieder im Stuhlkreis zusammen, um das Objektspiel noch einmal von Anfang an zu spielen. Diesmal führten die Schülerinnen und Schüler, die keine Objektkarte vor sich trugen, ein Protokoll (vgl. Anhang A.2.4).

Gegen Ende des ersten Paragraphen im Text wird das gesamte Licht ausgeschaltet. Die Schülerinnen und Schüler riefen intuitiv die Methode `ausschalten()` bei drei Objekten gleichzeitig auf, was zum Anlass genommen wurde, zu erklären, dass der Rechner keine Aufträge parallel abarbeiten kann, sondern nur sequentiell arbeitet.

Das Schalten der Hintergrundbeleuchtung stellte sich als schwierig heraus. Beim Befragen des Helligkeitssensors wurde zum ersten Mal etwas an das Objekt `martin` zurückgegeben. Der Lehrer erklärte, dass es Methoden gebe, die nichts zurückgeben (bspw. `einschalten()`) und welche, die etwas zurückgeben, wie in diesem Fall. Es wurde `"ist hell"` an das Objekt `martin` zurück gegeben und es stellte sich dann heraus, dass die Hintergrundbeleuchtung keine Methode hatte, die mit dem Parameterwert `"ist hell"` etwas anfangen konnte. Deswegen wurde die Objektkarte um die Methode `schalten(status)` erweitert.

Beim RGB-Scheinwerfer sahen die Schülerinnen und Schüler selbstständig ein, dass man für den Methodenaufruf `einschalten()` drei Parameter benötigte (Farbwerte für Rot, Grün, Blau). An dieser Stelle wurden der Objektkarte auch drei Attribute für die Farbwerte hinzugefügt.

Anschließend wurde das Protokoll in ein Sequenzdiagramm überführt. Dafür bekamen die Schülerinnen und Schüler das entsprechende Informationsblatt (Anhang S. 117f). Hausaufgabe war es, diese Arbeit fertig zu stellen und ggf. das Objektdiagramm aus den Stunden zuvor zu erweitern.

| Nr. | Datum | Thema |
|-----|---------------------------------------|----------------------------------|
| 5 | Einzelstd. ³ 16.11.2016 | Besprechung des Sequenzdiagramms |

Neben einer kurzen Wiederholung der klausurrelevanten Themen wurde in dieser Stunde das Sequenzdiagramm besprochen, für das die Schülerinnen und Schüler ca. zwei Wochen Zeit hatten. Dabei wurde nicht das komplette Sequenzdiagramm betrachtet (vgl. Anhang A.2.6), sondern nur das Diagramm für die ersten vier Objekte, die aktiv wurden.

Dabei kam unter anderem der Vorschlag auf, den Aktivitätsbalken beim Anschalten eines `Scheinwerfer`-Objekts länger zu machen, weil der Scheinwerfer danach die ganze Zeit an sei. Dies wurde aber abgelehnt, weil die eigentliche Aktivität (das »Anschalten« des Scheinwerfers) nur kurz dauere und sich dabei der Zustand ändere. Nach dem Statuswechsel (von `aus` in `an`) ist das Objekt darüber hinaus nicht weiter aktiv.

Für den Tafelanschrieb (vgl. Anhang A.2.5) wurde noch einmal die Namensgebung für Objekte wiederholt. Darüber hinaus wurde auch noch einmal das Konzept eines Parameters und des Parameterwerts erklärt: zum Beispiel ruft das Objekt `Lehrer` beim Objekt `Schueler` die Methode `zuhoeren(inhalt)` auf, wobei der Parameterwert

³ Die Stunden am 09.11.2016 und 10.11.2016 sind ausgefallen (Informatik-Biber und Wandertag).

des Parameters `inhalt` dann eben der Lehrinhalt ist (z. B. binomische Formeln). Eine Rückgabe auf diese Methode vom Objekt `Schueler` könnte dann "`habe verstanden`" lauten.

In diesem Zusammenhang wurde auch über die Methode `befragen()` des Helligkeits-sensors (Phototransistors) und die Rückgabe "`ist hell`" diskutiert. Mit dem Hintergedanken an die Wahrheitswerte `true` und `false` wurde die Methode in `befragenIstHell()` umbenannt. Begründet wurde dies mit der Häufigkeit, mit der solche Fragestellungen auftreten, und dass Programmiersprachen für solche Fragen Standardantworten bzw. Schlüsselwörter (`true`, `false`) bereitstellten. Es wurde von einem Schüler auch der Vorschlag gemacht, die Helligkeit in einen Zahlenwert zu konvertieren und diesen zurück zu geben (bspw. 0 für keinen Lichteinfall, 1 für Lichteinfall).

Die Schülerinnen und Schüler fragten bei der Besprechung auch nach, ob die Klammern am Ende einer Methode immer nötig seien. Dies wurde bestätigt, mit der Begründung, so Methoden schon am Namen erkennen zu können (und sie von Attributen unterscheiden zu können). Die Schülerinnen und Schüler sollten zur nächsten Stunde das Sequenzdiagramm ggfs. korrigieren.

| Nr. | Datum | Thema |
|-----|---------------------------------------|---|
| 6 | Einzelstd. ⁴ 23.11.2016 | Vorstellen der Hardware und Kennenlernen des Raspberry Pi |

In dieser Stunde wurden zunächst die Klausuren zurück gegeben. Dann lernten die Schülerinnen und Schüler den Raspberry Pi und die Bauteile kennen, die „die Objekte in die Wirklichkeit übertragen sollen“ (Zitat Fachlehrer). Die Erklärungen waren dabei die ganze Zeit via Beamer zu verfolgen.

| Eigener Name (im Unterricht gewählt) | | Klasse im Hintergrund (aus der rpCollection) |
|---|---|---|
| Scheinwerfer | → | RPDiode |
| Hintergrundbeleuchtung | → | RPDiode |
| RGBScheinwerfer | → | RPRGB |
| Helligkeitssensor | → | RPPhototransistor |

Tabelle 4.2: Modellierte Klassen aus dem Hintergrund (für die Schülerinnen und Schüler noch nicht als »Klassen« bekannt).

Die Schülerinnen und Schüler bekamen ein Arbeitsblatt zum Anschluss der Bauteile an das Steckbrett des Raspberry Pi (Anhang S. 119f). Auf diesem Blatt

⁴ Die Stunde am 17.11.2016 wurde nicht begleitet (Informatik-Klausur).

wurden noch einmal alle wichtigen Punkte wiederholt. Die Schülerinnen und Schüler hatten dann die Möglichkeit, ihre Raspberry Pis zu starten und die rpCollection zu aktualisieren. Die Schülerinnen und Schüler waren bis zum Ende der Stunde damit beschäftigt, die Pis kennenzulernen und einzurichten.

Für den Unterricht mit der rpCollection wurden die Klassennamen aus der Modellierungsphase (dem Objektspiel) übernommen. So hatten die Schülerinnen und Schüler die Möglichkeit mit exakt den Klassen / Objekten und Methoden zu arbeiten, für die sie sich in der Modellierung entschieden haben (Tabelle 4.2) und die sie schon aus dem Objektspiel kannten.

| Nr. | Datum | Thema |
|-----|--------------------------|---|
| 7 | Doppelstd. 24.11.2016 | Installation der rpCollection und Übung zur OOM |

Neben anfänglichen Problemen die korrekte `$JAVA_HOME`-Variable zu setzen um die Groovy-Console starten zu können, schlossen die Schülerinnen und Schüler ihre Pis an und brachten die rpCollection auf den neusten Stand. Bedauerlicherweise stellten sich die Java-Probleme als größer heraus, sodass die Arbeit abgebrochen wurde, und die Schülerinnen und Schüler mit einer Partnerarbeit weitermachten.

Dafür wurde für die letzte halbe Stunde ein Arbeitsblatt zum Würfelspiel »Meiern« ausgegeben (Anhang S. 121), in dem auch ein linearer Ablauf ähnlich zum Theaterprojekt mit dem Verfahren von ABBOTT modelliert werden sollte. Hausaufgabe war es, die Objektkarten für die Modellierung fertig zu stellen.

| Nr. | Datum | Thema |
|-----|--------------------------|--|
| 8 | Einzelstd. 30.11.2016 | Einsatz der Hardware-Bausteine zur Umsetzung des Theaterprojekts |

In dieser Stunde konnte zum ersten Mal richtig mit den Bauteilen gearbeitet werden. Die Schülerinnen und Schüler hatten keine Probleme, die Bauteile korrekt anzuschließen. Sie nahmen dabei das Arbeitsblatt aus der letzten Woche zu Hilfe (Anhang S. 119f).

Vor den Schülerinnen und Schülern wurde an dieser Stelle nicht von Bauteilen, sondern von Objekten gesprochen (diese Nomenklatur wird auch auf dem Arbeitsblatt verwendet). Diese Entscheidung wurde bewusst getroffen, da auf Objekten Methoden aufgerufen werden (und nicht auf Bauteilen) und so ein direkter Bezug zu den modellierten Objekten hergestellt werden konnte.

Während des Unterrichts konnte großes Interesse an der Arbeit mit der Hardware beobachtet werden. Die Schülerinnen und Schüler experimentierten von sich aus mit der Steuerung der Dioden und der RGB-LED, sodass sie erinnert werden mussten, den Theater-Text umzusetzen. Dafür hatte ca. die Hälfte der Schülerinnen und Schüler das Protokoll aus dem Objektspiel zur Hand, in welchem festgehalten wurde, welches Objekt wann auftritt und mit welchen anderen Objekten es interagiert.

| Nr. | Datum | Thema |
|-----|--------------------------|---|
| 9 | Doppelstd. 01.12.2016 | Fortsetzung: Einsatz der Hardware-Bausteine zur Umsetzung des Theaterprojekts |

In dieser Doppelstunde wurde an der Programmierung des Theater-Textes weitergearbeitet. Die Schülerinnen und Schüler hatten dafür wieder das Protokoll als Hilfestellung vorliegen.

Bei der Programmierung wurden insbes. Fehler beim Aufruf von Methoden gemacht, die in Quelltext 4.1 aufgeführt sind. Hauptsächliche Fehlerquellen waren falsche Objekte / Objektbezeichner, bei denen Methoden aufgerufen wurden. Oder es wurden Methoden aufgerufen, die ein Objekt gar nicht hatte (hier wurde dann nicht die Methode aus der Modellierung verwendet, sondern geraten, wie die Methode heißen könnte). Dies war insbes. beim Befragen des Helligkeitssensors der Fall. Die entsprechende Methode `befragen()` wurde in der Woche zuvor in `befragenIstHell()` umbenannt, um auf den booleschen Rückgabewert hinzuweisen.

```

1 // Falsche Objekterstellung
2 rgbscheinwerfer = new RGBScheinwerfer(11)(12)(13)
3
4 rgbscheinwerfer = new RGBScheinwerfer(rot 11, gruen 12, blau 13)
5
6 // Falsche Aufrufe von Methoden: falsche Punktnotation
7 roterScheinwerfer . anschalten()
8
9 // Falsche Aufrufe von Methoden: falscher Objektbezeichner
10 rgbscheinwerfer = new RGBScheinwerfer(15)
11 RGBScheinwerfer.einschalten(100, 100, 100)
12
13 // Fehlende Semiklare am Ende einer Zeile sorgen in Groovy fuer keine
   Fehlermeldung (und wurden von den SuS mal gemacht, mal nicht)

```

Quelltext 4.1: Falsche Aufrufe von Methoden.

Die Schülerinnen und Schüler kamen bis zu dem Abschnitt mit dem Helligkeitssensor gut und zügig voran. Die Rückgabe des Helligkeitssensors und das Schalten

der Hintergrundbeleuchtung mit der Rückgabe bereiteten Probleme, sodass unter Rückgriff auf das Protokoll und das Sequenzdiagramm das Vorgehen erläutert werden musste. Die Schülerinnen und Schüler speicherten die Rückgabe in einer Variable (`helligkeitswert` oder `antwort`), die als Parameter an die Hintergrundbeleuchtung übergeben wurde. Eine einzeilige Lösung wie in Zeile 5 wurde nicht beobachtet.

```

1 antwort = helligkeitssensor.befragenIstHell()
2 hintergrundbeleuchtung.schalten(antwort)
3
4 // nicht beobachtet:
5 hintergrundbeleuchtung.schalten(helligkeitssensor.befragenIstHell())

```

Quelltext 4.2: Hintergrundbeleuchtung und Helligkeitssensor.

Der Hinweis, dass am Ende (oder zu Beginn) des Skripts der GPIO einmal heruntergefahren werden muss, wurde mehrfach gegeben. So ist eine erneute Ausführung des Skripts ohne Fehlermeldungen möglich (Pinne werden freigegeben und stehen beim erneuten Aufruf zur Verfügung, vgl. Kapitel 3.4 und Anhang D.3.2). Einige Schülerinnen und Schüler programmierten in externen Editoren und führten den Quelltext dann in der Groovy-Console aus, da diese gerade auf älteren Raspberry Pis fast nicht flüssig zu bedienen ist.

| Nr. | Datum | Thema |
|-----|--------------------------|--|
| 10 | Einzelstd. 07.12.2016 | Besprechung des Quelltextes zum Theaterprojekt |

Zu Beginn dieser Stunde wurde eine Schülerlösung mit dem Beamer projiziert und besprochen. Ein Schüler las dabei den Text Satz für Satz vor und der Kurs erklärte die Umsetzung im Quelltext (vgl. Anhang A.2.7).

Bei der Besprechung wurde besonders auf die Interaktion zwischen Hintergrundbeleuchtung und Helligkeitssensor Wert gelegt. Dabei wurde von den Schülerinnen und Schülern auch ein Aufruf wie in Zeile 5 in Quelltext 4.2 vorgeschlagen. Darüber hinaus wurden die Vor- und Nachteile der beiden Möglichkeiten besprochen (Quelltext 4.2, Zeile 5 im Vergleich zu Zeilen 1 u. 2).

Im letzten Drittel der Stunde sollten die Schülerinnen und Schüler das Besprochene in ihre Lösungen aufnehmen und Fehler korrigieren. An dieser Stelle sei angemerkt, dass der Informatik-Raum die Möglichkeit bot, die Steckbretter mit den angeschlossenen Bauteilen bis zur darauf folgenden Informatikstunde einzuschließen.

| Nr. | Datum | Thema |
|-----|--------------------------|--|
| 11 | Doppelstd. 08.12.2016 | Kennenlernen von Objektbeziehungen durch das Objektspiel am Beispiel des Theaterprojekts |

Diese Stunde begann mit dem Objektspiel. Dabei wurde der Fokus auf die Beziehungen zwischen den Objekten gelegt. Es wurden Garn und Pfeile benutzt, um gerichtete Objektbeziehungen zu kennzeichnen. Die Schülerinnen und Schüler gingen dabei wieder Schritt für Schritt vor, indem die Geschichte satzweise vorgelesen wurde. Das Garn wurde an die Objektkarten geklebt, sodass ein Netz der Beziehungen entstand (vgl. Anhang A.2.8, Abb. A.6). Die meisten Beziehungen gingen dabei von Martin aus und führten zu den verschiedenen Scheinwerfern und dem anderen Bühnenequipment. Die Pfeile wurden mit den Namen beschriftet, unter welchen das Objekt `martin` die anderen Objekte kennt.

Auf die Beziehung zwischen Hintergrundbeleuchtung und Helligkeitssensor wurde besonders eingegangen, da es hier zwei Möglichkeiten gibt, die Situation zu modellieren. Dabei wurde von den Schülerinnen und Schülern die Modellierung wie in Abbildung 4.4 dargestellt selbst vorgeschlagen, da „Martin den Helligkeitssensor gar nicht kennen muss“. Genauere Erklärungen finden sich noch in Anhang A.2.8.

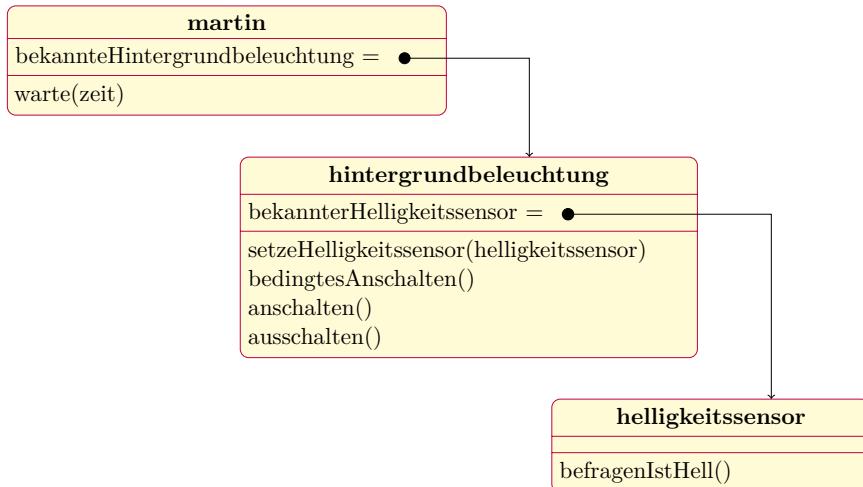


Abbildung 4.4: Martin kennt nur die Hintergrundbeleuchtung und diese den Helligkeitssensor. Es werden nur die wichtigsten Attribute und Methoden dargestellt.

Es wurden auch Vor- und Nachteile dieser Modellierung angesprochen. Kennt nur die Hintergrundbeleuchtung den Helligkeitssensor, kann Martin mit der Antwort des Helligkeitssensors keine anderen Scheinwerfer schalten. Die Klasse `Hintergrundbeleuchtung` wurde um die Methode `bedingtesAnschalten()` erweitert, sodass die Schülerinnen und Schüler die Situation umsetzen konnten.

Am Ende der Stunde wurde an der Tafel die Notation der Beziehungen im Objektdiagramm besprochen (vgl. Anhang A.2.9). Hausaufgabe war es, das Objektdiagramm des Theater-Textes um die Beziehungen zu erweitern.

| Nr. | Datum | Thema |
|-----|--------------------------|--|
| 12 | Einzelstd. 14.12.2016 | Besprechung der Modellierung für das Spiel »Meiern« ⁵ |

Da die Besprechung des Arbeitsblattes zum Spiel »Meiern« noch ausstand, wurde dies in dieser Stunde nachgeholt. Die Schülerinnen und Schüler verglichen ihre Modellierungen und erstellten aus den Objektkarten ein Objektdiagramm. Dabei wurden auch die Beziehungen modelliert. Die Begriffe der »gerichteten« und »ungerichteten« Beziehung wurden dabei wiederholt.

| Nr. | Datum | Thema |
|-----|--------------------------|---|
| 13 | Doppelstd. 15.12.2016 | Fortsetzung: Besprechung der Modellierung (Fokus Beziehungen) für das Spiel »Meiern« ⁶ |

Für die Arbeit mit dem Theater-Text erstellten die Schülerinnen und Schüler Sequenzdiagramme, die die zwei Varianten des Schaltens der Hintergrundbeleuchtung abbildeten (vgl. Anhang A.2.10). Als Hausaufgabe sollte das Objektdiagramm zum Spiel »Meiern« fertig gestellt werden.

| Nr. | Datum | Thema |
|-----|--------------------------|--|
| 14 | Einzelstd. 21.12.2016 | Möglichkeiten der Modellierung der Beziehung zwischen Hintergrundbeleuchtung und Helligkeitssensor und Umsetzung in der Groovy-Console |

Zu Beginn dieser Stunde wurde ein Informationsblatt zur Modellierung der Schaltung der Hintergrundbeleuchtung ausgeteilt (Anhang S. 123f). Auf dem Blatt wurde eine neue Notation im Objektdiagramm für den Fall genutzt, dass ein Objekt ein anderes kennt (und es deswegen als Attribut hat, vgl. auch Abb. 4.4). Die Schülerinnen und Schüler konnten die Notation von sich aus erklären.

Die Sequenzdiagramme wurden von den Schülerinnen und Schülern schon selber erarbeitet und wurden auf dem Informationsblatt nur der Vollständigkeit halber angegeben. Für den Rest der Stunde wurde damit begonnen, beide Versionen in der

⁵ Da das Spiel »Meiern« für diese Masterarbeit nicht relevant ist, fällt der Bericht von dieser Stunde entsprechend kurz aus.

⁶ Vgl. Fußnote 5.

Groovy-Console zu programmieren (vgl. Anhang A.2.11). Dabei ist die Variante A (Quelltext A.2, S. 145) die bekannte und Variante B I die neue Modellierung (Quelltext A.3, S. 145). Bei einigen Schülern wurde auch die Variante B II beobachtet (Quelltext A.4, S. 146), die noch konsequenter als Variante B I ist: In Variante B II hat Martin *gar keine* Verbindung zum Helligkeitssensor.

| Nr. | Datum | Thema |
|-----|--------------------------|---|
| 15 | Doppelstd. 22.12.2016 | Fortsetzung: Möglichkeiten der Modellierung der Beziehung zwischen Hintergrundbeleuchtung und Helligkeitssensor und Umsetzung in der Groovy-Console |

In dieser letzten Stunde vor den Weihnachtsferien wurde die Arbeit aus der letzten Stunde beendet. Die Schülerinnen und Schüler wurden noch einmal besonders aufgefordert, wirklich nur die auf dem Informationsblatt beschriebene Situation umzusetzen, da sie sich teilweise nicht mit dem Schalten der Hintergrundbeleuchtung beschäftigten. Unter Rückgriff auf das Sequenzdiagramm konnte die Variante A von allen umgesetzt werden (diese war schon länger bekannt).

Die Methode `setzeHelligkeitssensor(helligkeitssensor)` wurde nach der ersten Stunde im Plenum diskutiert, da mehrere Schülerinnen und Schüler zu diesem Zeitpunkt auf das Problem stießen, keine Verbindung zwischen Hintergrundbeleuchtung und Helligkeitssensor herstellen zu können. Dafür wurde der Ausschnitt des Objektdiagramms an die Tafel gezeichnet und mit den Schülerinnen und Schülern die Beziehung besprochen.

Am Ende der Stunde hatten alle Schülerinnen und Schüler beide Varianten erfolgreich umgesetzt. Diese Stunde beendete vorerst die Arbeit mit der rpCollection. Nach den Weihnachtsferien ging es mit dem Unterrichtsvorhaben EF-3 weiter (vgl. [Schulint. Lehrplan, S. 17–20]).

4.6 Anmerkungen zur Durchführung des Unterrichts

Der Unterricht in der EF wurde seit den Herbstferien durchgehend begleitet. Dabei wurden die Beobachtungen in entsprechenden Bögen festgehalten (Anhang S. 103f).

Die Schülerinnen und Schüler kritisierten beim Theater-Text, dass die Scheinwerfer in einem Theater nicht *auf* der Bühne stünden, sondern an der Decke montiert wären. Für ein wiederholtes Arbeiten mit dem Text sollte dies geändert werden.

Das Objektspiel wurde von den Schülerinnen und Schülern anfangs belächelt. Sie sahen es dann aber als eine gute Möglichkeit, die Modellierung zu überprüfen und um bspw. überflüssige Objekte zu entfernen. Das Kennzeichnen von Beziehungen durch

Garn und Pfeile war eine gute Möglichkeit, diese einzuführen und auf Unterschiede der gerichteten und ungerichteten Beziehung einzugehen. Auch das Schalten der Hintergrundbeleuchtung konnte so von den Schülerinnen und Schülern weitestgehend selbst erarbeitet werden.

Die Arbeit mit Objekt- bzw. Sequenzdiagrammen hat den Schülerinnen und Schülern keine Probleme bereitet. Sie konnten mit den Informationsblättern die Inhalte auf den Theater-Text übertragen und für diese Problemsituation die Diagramme zeichnen.

Die Arbeit mit den Hardware-Bausteinen hat dabei gut geklappt, auch wenn angemerkt werden muss, dass die Einrichtung von Java auf dem Raspberry Pi mehr Zeit als geplant in Anspruch nahm. Das Anschließen der Bauteile gelang ohne Probleme und es wurde kein Bauteil durch falsches Anschließen zerstört. Die Schülerinnen und Schüler mussten teilweise sogar dazu aufgerufen werden, sich mit dem eigentlichen Text zu beschäftigen und nicht andere Programme für die Hardware zu schreiben. Sie fragten interessiert nach, was die einzelnen Bauteile seien und was man mit ihnen machen könne, sodass sie sich wahrscheinlich auf die Umsetzung eines zweiten Projekts freuen (Hafenkran).

Insgesamt konnte (rein subjektiv festgestellt) kein großer Unterschied beim Interesse der Schülerinnen und Schüler während der Modellierungs- und Implementationsphase festgestellt werden. Auch als sie in der Groovy-Console arbeiteten, haben sie interessiert am Unterricht teilgenommen.

4.7 Auswertung der Unterrichtsziele

Im Unterricht wurden alle Punkte behandelt, die in der Definition eines guten Objektverständnisses verwendet wurden (Kapitel 4.2). Die Schülerinnen und Schüler haben intensiv verschiedene Situationen modelliert und mit Objekten gearbeitet. Dabei haben sie Objekte auf diverse Weisen dargestellt und konnten Unterschiede in verschiedenen Modellierungen erklären. Sie haben unterschiedliche Arten von Methoden kennen gelernt und konnten die richtige Art für die Modellierung auswählen (dies ist für das Programmieren von Klassen noch wichtig).

Die Schülerinnen und Schüler haben während der Beobachtung ein gutes Verständnis objektorientierter Zusammenhänge gezeigt. Da dieses Verständnis bis jetzt nur subjektiv begründet werden kann, soll im nächsten Kapitel die Arbeit mit der rpCollection durch die Schülerinterviews ausgewertet werden.

Kapitel 5

Auswertung: Hardwareeinsatz

In diesem Kapitel werden die Interviews ausgewertet und es wird aufgezeigt, welches objektorientierte Verständnis die Schülerinnen und Schüler durch den Einsatz der Hardware erlangen konnten. Dafür wird die inhaltlich-strukturierende, qualitative Inhaltsanalyse nach KUCKARTZ angewendet (vgl. [Kuckartz 2016, Kap. 5, S. 97ff]). Es werden auch die Themen aufgezeigt, bei denen die Schülerinnen und Schüler noch Schwächen haben.

5.1 Informationen zu den Interviews

Mit den Schülerinnen und Schülern wurden leitfadengestützte, problemzentrierte Interviews geführt¹ (vgl. [Helfferich 2011, S. 36]). Der Leitfaden kann „als Hintergrundskontrolle [sic!]“ (vgl. ebd. S. 36) dienen und es gibt die Möglichkeit „spontane[r] Fragen durch Interviewende“ (vgl. ebd. S. 36). Generell ist die Formulierung und Reihenfolge der Fragen während des Interviews anpassbar (vgl. ebd. S. 36), um bspw. den Gesprächsfluss nicht abbrechen zu lassen oder alle Themen aufzugreifen, die im Leitfaden festgehalten sind. „[D]as Eingehen auf die InterviewpartnerInnen und die spezielle Situation [kann] andere Fragestrategien nahe legen“ [Altrichter Posch 2007, S. 152], als im Leitfaden festgehalten sind.

Die Interviews wurden in der vorletzten Dezemberwoche vor den Weihnachtsferien geführt (am 14.12.2016 u. 15.12.2016). Auch wenn nicht alle interviewten Schülerinnen und Schüler männlich waren, wird im Folgenden nur die männliche Form verwendet. Die Schüler wurden in Absprache mit dem Fachlehrer so ausgewählt, dass ein möglichst durchschnittlicher Leistungsstand in den Interviews beobachtet

¹ Das Einverständnis der Erziehungsberechtigten für die Durchführung der Interviews wurde im Vorhinein eingeholt (Anhang S. 149f). Mit dem Schulleiter wurde die Durchführung während der Unterrichtszeit abgesprochen.

werden kann. Es wurden sechs Interviews geführt, die alle ca. 15 Minuten gedauert haben. Sie sind transkribiert worden und im Anhang zu finden (Anhang B.3–B.8, Anhang B.2 für die Transkriptionsregeln).

5.2 Interviewleitfaden

Der Leitfaden ist so aufgebaut, dass er vom allgemeinen Begriff der »Objektorientierung« zu konkreteren Begriffen (»Objekt«, »Methode« etc.) übergeht. Zunächst sollen die Schüler aus ihrer Sicht erklären, was die zentralen Punkte bei der Objektorientierung seien. So soll festgestellt werden, welchem Thema die Schüler die größte Relevanz zuordnen. Danach wird der Blick auf Objekte gerichtet. Die Schüler sollen selbst ein Beispiel nennen und erklären, warum es sich dabei in ihren Augen um ein Objekt handle.

Anschließend soll auf Objektkommunikation und -beziehungen eingegangen werden. Dies wurde ausführlich im Theaterprojekt behandelt. Damit hängt natürlich auch das Verhalten bei Methodenaufrufen zusammen (Frage 4), sodass sich die Antworten zu diesen Fragen sehr wahrscheinlich nicht gänzlich trennen lassen. Bei den Methodenaufrufen sollen auch Begriffe wie »Parameter« und »Rückgabewert« angesprochen werden.

Frage 5 ist optional, falls die Objekte aus dem Unterricht noch nicht angesprochen wurden. Es ist denkbar, dass dieses Thema immer mal wieder in Antworten angeschnitten wird. Abschließend sollen die Schüler die Arbeit mit dem Pi und der Hardware aus ihrer Sicht bewerten.

Die Fragen sind im Folgenden noch einmal aufgeführt. Dabei wurde die Frage nach einem bisherigen Kontakt mit der Informatik als erzählgenerierender Einstieg gewählt, der eine „angenehme, entspannte Gesprächsatmosphäre“ [Selter Spiegel 1997, S. 107] vermitteln soll. Den Schülern soll der Druck der Interventionsituations genommen werden und es soll auch deutlich werden, dass es sich nicht um eine Prüfungssituation handelt (Stichwort Transparenz, [Selter Spiegel 1997, S. 107]). Der Leitfaden kann in Anhang B.1 eingesehen werden.

1. Erzähle, welchen Kontakt du bisher zur Informatik hattest. Welche Vorerfahrungen konntest du evtl. schon sammeln?
2. Erkläre die im Unterricht kennengelernten Begriffe der Objektorientierung.
3. Nenne ein Objekt als Beispiel und erkläre, warum das für dich ein Objekt ist.

4. Erkläre, wie Objekte untereinander Informationen austauschen können. Kannst du ein Beispiel geben?
5. Wie verhält sich ein Objekt, wenn eine Methode auf diesem aufgerufen wird?
6. Wie haben wir im Unterricht mit den Objekten in der Groovy-Console gearbeitet? (Falls noch nicht angesprochen bisher)
7. Bewerte die Arbeit mit der Hardware (den Objekten) und dem Raspberry Pi. Inwiefern war diese Art von Arbeit für dich hilfreich für das Verständnis?

Neben den Fragen enthält der Leitfaden Stichpunkte für erzählgenerierende Nachfragen, falls die Schüler eine Frage nicht beantworten können. Durch die aufgeführten Stichpunkte ist oft eine andere Blickrichtung auf den Kern der Fragen möglich. Der Leitfaden deckt alle in Kapitel 4.2 aufgeführten Punkte eines guten Objektverständnisses ab, sodass final festgestellt werden kann, inwiefern die Schüler dieses verinnerlicht haben.

5.3 Auswertung der Interviews

Die inhaltlich-strukturierende, qualitative Inhaltsanalyse nach KUCKARTZ läuft in sieben Schritten ab (vgl. [Kuckartz 2016, S. 97]). Nach einer Sichtung des Materials und der Hauptkategorienbildung wird das Material mit ebendiesen codiert. Danach werden alle mit der gleichen Hauptkategorie gekennzeichneten Stellen zusammenge stellt und dabei induktiv Unterkategorien abgeleitet, mit denen das Material erneut codiert wird. So entsteht eine noch feinere Kategorisierung. Daraufhin können das Material analysiert, visuell aufbereitet, präsentiert und die Forschungsfragen beantwortet werden. In der Übersicht in Anhang B.9 findet man einen schnellen Zugriff auf die mit den Haupt- und Unterkategorien codierten Textstellen. Für das Codieren der Interviews wurde eine QDA-Software benutzt.

5.3.1 Hauptkategorienbildung

Es sind die in Tabelle 5.1 aufgeführten deduktiven Hauptkategorien für die qualitative Analyse der Interviews gebildet worden. Diese aus der Theorie abgeleiteten Kategorien (vgl. [Kuckartz 2016, Kap. 4.1, S. 64ff]) werden bei einer zweiten Sichtung des Materials um die Unterkategorien erweitert (s. Kapitel 5.3.2). Auch wenn sich i. d. R. nicht für jede Frage des Leitfadens eine eigene Hauptkategorie ableiten lässt, gelingt dies hier doch, da der Leitfaden versucht, die Themen möglichst unabhängig

voneinander aufzugreifen. Zudem lassen dies die Interviews aufgrund der geringen Länge zu.

| Hauptkategorie | Name (<i>kursiv</i>) und Definition |
|----------------|--|
| 0 (VOR) | <i>Vorerfahrungen mit Informatik</i> Textstellen, die auf Vorerfahrungen mit Informatik hindeuten. |
| 1 (OOP) | <i>Objektorientierung als Konzept</i> Textstellen, die die zentralen Aspekte der Objektorientierung aus Sicht der Schüler beinhalten. |
| 2 (OBJ) | <i>Verständnis Objekt</i> Textstellen, die auf das Verständnis von Objekten hindeuten. |
| 3 (MET) | <i>Verständnis Methode</i> Textstellen, die auf das Verständnis von Methoden hindeuten. |
| 4 (ATT) | <i>Verständnis Attribut</i> Textstellen, die auf das Verständnis von Attributen hindeuten. |
| 5 (KOM) | <i>Verständnis Objektkommunikation</i> Textstellen, die auf das Verständnis von Objektbeziehungen und Objektkommunikation hindeuten. |
| 6 (GRO) | <i>Arbeit mit Groovy</i> Textstellen, die von der Arbeit mit Groovy berichten. |
| 7 (BEW) | <i>Bewertung der Arbeit</i> Textstellen, die aus einer Bewertung der Arbeit mit dem Raspberry Pi und der rpCollection bestehen. |

Tabelle 5.1: Hauptkategorien für die qualitative Analyse nach KUCKARTZ.

5.3.2 Unterkategorienbildung

Die Unterkategorien sind induktiv am Material gebildet worden (vgl. [Kuckartz 2016, Kap. 4.2, S. 72ff]) und fächern die Hauptkategorien weiter aus (s. Tabelle 5.2). Sie erweitern somit die Kategorienbildung und ermöglichen eine bessere Trennung von Textstellen innerhalb einer Hauptkategorie. Dabei gehören die Textstellen aus einer Unterkategorie auch alle zur entsprechenden Hauptkategorie. Auf der anderen Seite gibt es auch Textstellen, die sowohl einer Haupt- und Unterkategorie zugeordnet werden (wenn sie z. B. der allgemeinen Sichtweise der Hauptkategorie entsprechen).

| Hauptkategorie | Unterkategorie (<i>kursiv</i>) und Definition |
|----------------|--|
| 0 (VOR) | <p>0.1 — <i>Vorerfahrungen intrinsisch motiviert</i> Textstellen, die auf Vorerfahrungen mit Informatik aus einer intrinsischen Motivation hindeuten.</p> <p>0.2 — <i>Vorerfahrungen extrinsisch motiviert</i> Textstellen, die auf Vorerfahrungen mit Informatik aus einer extrinsischen Motivation hindeuten.</p> |
| 1 (OOP) | <p>1.1 — <i>Objektspiel</i> Textstellen, die von dem Objektspiel berichten. Diese Textstellen können sich auch auf das Verfahren von ABBOTT beziehen, da im Unterricht die Ergebnisse meist mit dem Objektspiel auf Konsistenz geprüft wurden.</p> |
| 2 (OBJ) | <p>2.1 — <i>Beispiele für Objekte</i> Textstellen, die von Schülern gewählte Beispiele für Objekte beinhalten.</p> |
| 3 (MET) | <p>3.1 — <i>Umgang mit Rückgabewerten</i> Textstellen, die von dem Umgang mit Methoden mit Rückgaben berichten.</p> <p>3.2 — <i>Umgang mit Parameterwerten</i> Textstellen, die von dem Umgang mit Methoden mit Parameterwerten berichten.</p> <p>3.3 — <i>Beispiele für Methoden</i> Textstellen, die von Schülern gewählte Beispiele für Methoden beinhalten.</p> |
| 4 (ATT) | <p>4.1 — <i>Beispiele für Attribute</i> Textstellen, die von Schülern gewählte Beispiele für Attribute beinhalten.</p> |
| 5 (KOM) | <p>5.1 — <i>Hintergrundbeleuchtung und Helligkeitssensor</i> Textstellen, die von der Objektkommunikation bei der Hintergrundbeleuchtung und dem Helligkeitssensor berichten.</p> <p>5.5 — <i>Beispiele für Objektkommunikation</i> Textstellen, die von Schülern gewählte Beispiele für Objektkommunikation beinhalten.</p> |
| 6 (GRO) | <p>6.1 — <i>Objekterstellung in Groovy</i> Textstellen, die von der Objekterstellung in Groovy berichten.</p> |
| 7 (BEW) | <p>7.1 — <i>Bewertung der Arbeit</i> Textstellen, die explizit die Arbeit mit den Hardware-Bausteinen bewerten.</p> |

Tabelle 5.2: Unterkategorien für die qualitative Analyse nach KUCKARTZ.

5.3.3 Gezeigtes positives Verständnis von Objektorientierung

In Kapitel 4.2 wurde ein erstes Objektverständnis definiert, welches die Schüler nach dem Unterrichtsvorhaben erreicht haben sollten. Die interviewten Schüler haben alle ein implizites Verständnis, was als Objekt zu bezeichnen ist, und nennen dafür Dinge, die man anfassen könne (vgl. Schüler D, Z. 38–39). Schüler C hat es besonders treffend formuliert: „also mir fällt jetzt gerade nichts ein, was kein Objekt ist“ (Schüler C, Z. 61), wobei dies natürlich auch aufgrund fehlenden Abgrenzungswissens als Unwissenheit interpretiert werden kann. Wenn Beispiele vorliegen (Kaffeemaschine, Auto), können die Schüler Methoden und Attribute für diese Objekte finden.

Das Verfahren von ABBOTT und das Objektspiel können die Schüler erklären. Der direkte Zusammenhang des Objektspiels zur Arbeit mit Groovy wird besonders in den Aussagen von Schüler A deutlich, der die besondere Sprechweise („Phrasen“, Z. 60) beim Objektspiel als Grundlage für die Eingaben in der Groovy-Console („das Programm“, Z. 62) sieht:

Ja wir haben das halt erst theoretisch gemacht, wie gesagt dieses Objektkartenpiel, haben dann versucht, das quasi in so (.) ein Programm einzubringen, genau das was wir halt gesagt haben, mit diesen bestimmten (.) Phrasen, also, bestimmte Sätze, die wir dann eingebracht haben, und dann quasi genau das was auf den Objektkarten drauf stand, wurde dann genau so in das Programm eingebracht, was wir später mit den Raspberry Pis gemacht haben

Schüler A, Z. 58–63

Mit den Phrasen meint Schüler A natürlich die Methodenaufrufe. Allgemein können die Schüler alle erklären, welchen Zweck Methoden im Allgemeinen erfüllen, und sind in der Lage, verschiedene Arten von Methoden zu unterscheiden und die richtige Art von Methode für verschiedene Einsatzzwecke auszuwählen. Im Unterricht wurden durch das Objektspiel und auch beim Würfelspiel »Meiern« immer wieder verschiedene Arten von Methoden wiederholt, sodass die Schülerinnen und Schüler damit viel Kontakt hatten. Auch wenn an einigen Stellen die Begriffe »Attribut« und »Parameter« verwechselt werden (vgl. Schüler F, s. Kapitel 5.3.4), verwenden die meisten Schüler diese Begriffe korrekt. Schüler A versteht Methoden auch als Möglichkeit, anderen Objekten Befehle zu erteilen (vgl. Schüler A, Z. 119–127).

Beim Thema Objektkommunikation ist allen Schülern klar, dass Objekte durch Aufrufe von Methoden bei anderen Objekten mit diesen interagieren können. Im Unterricht haben sie dies beim Schalten der Hintergrundbeleuchtung gesehen. Die Schüler haben diese Problemsituation aus der Geschichte korrekt wieder gegeben und können erklären, wie die einzelnen Schritte ablaufen, und inwiefern die Antwort

des Helligkeitssensors einen Einfluss auf den Zustand der Hintergrundbeleuchtung hat. Im Folgenden erklärt Schüler C die beiden Modellierungen aus dem Unterricht:

- S: Ähm, ja also die Person, was ja bei dem Objektspiel Martin war, der hat ja die Hintergrundbeleuchtung angeschaltet mit der Antwort vom Helligkeitssensor. Und dann hat die Hintergrundbeleuchtung ja den Sensor befragt und wenn die Antwort halt *hell* war, dann ist es ausgeblieben, und wenn die Antwort *dunkel* war, dann ist es angegangen
- I: Ja, hat da jetzt die Hintergrundbeleuchtung den Helligkeitssensor gefragt oder wie hatten wir das am Anfang gemacht?
- S: Ne, also am Anfang hatten wir, dass die Person den Helligkeitssensor befragt und mit der Antwort, die er kriegt, die Beleuchtung schaltet

Schüler C, Z. 118–126

Ebenfalls erklärt Schüler B die Situation treffend. Er geht noch einmal auf den Unterschied der beiden Varianten ein:

- S: Ja wir hatten ja mehre Möglichkeiten. Entweder man speichert den [Helligkeitssensor], also, Martin fragt erst den Helligkeitssensor ob es hell ist und speichert die Antwort dann unter einer Variablen, mit der Variablen, die gibt er dann als Parameterwert weiter an die Hintergrundbeleuchtung oder wir hatten, dass der Martin nur die Hintergrundbeleuchtung kennt, die anschaltet und die dann halt erst den Helligkeitssensor fragt und dann halt je nach dem anschaltet oder nicht
- I: Wo ist da der Unterschied?
- S: Wir müssen nicht so viele Beziehungen herstellen. Also Martin muss halt zum Beispiel bei der zweiten Methode nicht den Helligkeitssensor kennen, sondern nur die Hintergrundbeleuchtung

Schüler B, Z. 53–62

Darüber hinaus sind die Schüler in der Lage, andere Beispiele für Objektkommunikation zu finden. Schüler E überträgt die Kommunikation in den Auto-Kontext:

[...] wenn das Auto jetzt einen Regensensor hat, dann befragt man den Regensensor ob es regnet, und dann könnte der eben den Wert in einer Variable abspeichern, wenn es zum Beispiel regnet könnte man mit dem Wert, ob der jetzt *ja* oder *nein* ist oder eben *true* oder *false*, damit dann den Scheibenwischer anschalten [...]

Schüler E, Z. 154–158

Schüler D nennt die alltägliche Kommunikation als Beispiel (vgl. Schüler D, Z. 120) und Schüler A führt ein Beispiel an, bei dem zwei Personen an der Ampel stehen und sich die eine bei der anderen erkundigt, ob es noch rot sei (vgl. Schüler A, Z. 199–214). Im Unterricht wurde als Beispiel auch die Kommunikation zwischen Schüler und Lehrer genannt (auch explizit als Beispiel für einen Methodenaufruf mit Parameter).

Die Schüler haben ein Verständnis dafür, dass Objekte zunächst erstellt werden müssen, bevor mit den Bauteilen interagiert wird. Auch wenn dies im Unterricht mehr implizit als explizit vermittelt wurde, haben die Schüler dies verinnerlicht. Sie sprechen von „anmelden“ (Schüler F, Z. 103), „registrieren“ (Schüler B, Z. 100) oder eintragen (vgl. Schüler D, Z. 160) der Objekte, damit der Computer diese kenne (vgl. UK 6.1).

Dass somit eine Beziehung zwischen Martin (dem Programm) und den verschiedenen Objekten entsteht, haben sie verstanden. Die Schüler begründen die explizite Anmeldung überwiegend damit, dass Martin / das Programm / der Computer die Objekte dann kenne und dann mit diesen gearbeitet werden könne.

Schüler A macht sogar gesondert darauf aufmerksam, dass beim Aufruf einer Methode erst einmal überprüft werden muss, ob das Objekt die entsprechende Methode überhaupt hat. Auch Schüler E geht auf diesen Punkt ein (vgl. Schüler E, Z. 162–163). Dieser Aspekt wurde im Unterricht vor allem beim Objektspiel angemerkt, wenn Schülerinnen und Schüler bspw. Methoden aufriefen, die ein Objekt gar nicht anbot oder zu ungenau in der Sprechweise waren (Aufruf von z. B. `einschalten()` statt `anschalten()`). Schüler A beschreibt die Situation treffend:

Ja, das [Objekt, das die Methode ausführen soll,] guckt halt erstmal nach, ob es diese Methode gibt, und wenn die Methode vorhanden ist, ja führt das die einfach aus und guckt nach, ob danach noch irgendetwas passieren muss, ob es den Status geändert werden muss, oder so etwas

Schüler A, Z. 135–138

In den Interviews wurden nicht die verschiedenen Darstellungsformen für Objekte oder Objektkommunikation abgefragt (Objektdiagramm mit Beziehungen, Sequenzdiagramm). Schüler B erwähnt als einziger eine der beiden Diagrammformen und nennt den „Aktivitätskasten“ (ebd. Z. 75) als Antwort auf die Frage, wie sich ein Objekt beim Methodenaufruf verhalte (vgl. Schüler B, Z. 72–75).

5.3.4 Gezeigtes negatives Verständnis von Objektorientierung

Auch wenn die Schüler Begriffe wie »Objekt« oder »Methode« richtig verwenden und auch eine Vorstellung zu diesen aufgebaut haben, können sie nichts mit dem Begriff der »Objektorientierung« an sich anfangen. Sie greifen dabei oft auf das Objektspiel bzw. das Verfahren von ABBOTT zurück, soll der Begriff erklärt werden. Die Objektorientierung wird von ihnen nicht als ein Weg gesehen, Software zu schreiben. Für sie ist Objektorientierung mehr ein Sammelbegriff, als ein Programmierparadigma bzw. eine besondere Sichtweise auf die Welt.

Für die Frage nach Objekten greifen alle Schüler auf die Arbeit mit der Hardware im Unterricht zurück und nennen Scheinwerfer oder Lampen für Beispiele. Dies war durch den Unterricht zu erwarten (s. Kapitel 3.3). Allerdings haben die Schüler dann oft Probleme, sich Objekte abseits der Hardware-Bausteine zu überlegen. Die Schüler können den Begriff überwiegend richtig erklären, sind aber spontan selten kreativ genug, den Objektbegriff auf andere Situationen zu erweitern.

Das Unvermögen, sich selbst Objekte auszudenken, kann daher röhren, dass im Unterricht hauptsächlich Scheinwerfer (und andere Bauteile) als Objekte genannt wurden. Auf der anderen Seite erklären die Schüler auch, dass im Prinzip *alles* ein Objekt sein kann. Vielleicht wissen die Schüler sich im Interview nicht festzulegen oder sind nicht spontan und / oder kreativ genug, sich ein eigenes Beispiel auszudenken.

In der Hauptkategorie »Verständnis Methode« lassen sich einige kleine Ungenauigkeiten beobachten. Interessant ist die Aussage von Schüler C, welche eine mögliche Fehlvorstellung zum Aufruf von Methoden enthält:

Ja, wenn da eine Methode aufgerufen wird, verändert das immer seinen Status oder halt, kommt halt auf die Methode drauf an, was es verändert, aber irgendein Attributwert wird dann halt immer verändert.

Schüler C, Z. 156–158

Natürlich muss ein Methodenaufruf nicht immer ein Attribut verändern. Diese Fehlvorstellung könnte sich durch das Objektspiel gebildet haben, in der beim An- und Ausschalten von Lampen immer auch der Status der Lampe auf `an` oder `aus` gesetzt wurde.

In der Geschichte wurde jedoch auch ein Beispiel für eine Methode ohne Attributwertänderung gegeben. Das Befragen des Helligkeitssensors läuft in der durch die Schülerinnen und Schüler modellierten Lösung ohne eine Attributwertänderung ab.

Schüler F hat an einer Stelle ein falsches Verständnis davon, was beim Methodenaufruf den Farbwert einer RGB-LED ändert. Er versteht die Attribute so, dass diese den Farbwert ändern und die Methode ausführen:

- I: [...] Wofür sind die Attribute da, und was passiert mit denen, wenn man mit so einem Objekt arbeitet?
- S: [...] Die Attribute führen das dann halt eben aus, beim RGB Scheinwerfer hatten wir jetzt so mischen und mit den unterschiedlichen Prozenten, dass die das dann so eingestellt haben

Schüler F, Z. 133–137

Hier scheint es sich nicht um ein versehentliches Verwechseln beider Begriffe zu handeln. Auf die Nachfrage, was dann Methoden und Attribute unterscheide, geht

Schüler F die Begründung über das Objektspiel, indem er Verben als Grundlage von Methoden nennt und dann auch noch Parameter- und Attributwerte durcheinander bringt, wie das folgende Zitat zeigt:

(3) Ja, im Wesentlichen sind Attribute ja erstmal (2) Verben, glaube ich (3), und Methoden (2) Ad-, ne, Methoden sind Verben und Attribute sind Adjektive, joar, und Methoden das macht der dann genau und Attribute sind dann immer die Parameterwerte dazu (2)

Schüler F, Z. 139–142

Auch Schüler A verwechselt Attribute und Methoden. Im Interview wird an folgender Stelle über einen Monitor gesprochen (Objekte im Computer-Kontext, wie Maus, Drucker etc.). Es ist natürlich nur sinnvoll, die Farbe eines Pixels als Attribut zu speichern.

I: Was hättest du da vielleicht noch für Methoden?
S: Beim Bildschirm?
I: Ja
S: Dass der und der Pixel vielleicht diese Farbe hat, und, ja

Schüler A, Z. 52–55

Darüber hinaus kann in den Interviews bei drei Schülern ein falscher Umgang mit Parametern beobachtet werden. Schüler C verwechselt Attribut und Parameter, wie das folgende Zitat zeigt. Diese Verwechslungen könnten daher kommen, dass der Begriff »Parameter« in der Mathematik häufig für eine unbekannte Zahl genutzt wird.

(1) Wenn man jetzt zum Beispiel einen Scheinwerfer hat, hat man das Attribut Standort zum Beispiel mit dem Para– Der Parameterwert ist auch irgendwie so Status ob *an* oder *aus*, und der Attributwert das andere

Schüler C, Z. 41–43

Schüler F verwechselt auch die beiden Begriffe. Er beschreibt ein Beispiel, bei dem ein Wecker auf dem Computer eingestellt wird, der längere Zeit inaktiv ist und „sich dann wieder melden muss“ (Interview F, Z. 60). Die Zeiterfassung erfolgt dabei wie folgt:

[...] da ist ja auch so ein Attributwert, so ein Parameter, nach einer bestimmten Zeit schaltet er sich dann wieder aus

Schüler F, Z. 62–63

Interessant ist auch die Stelle in Interview A, als über den Helligkeitssensor gesprochen wird. Schüler A sieht das Befragen des Helligkeitssensors nicht als Methodenaufruf beim entsprechenden Objekt des Helligkeitssensors, sondern als Handlung des Programms:

[...] und beim Helligkeitssensor war es so, dass das Programm sich selbst quasi einmal abfragt, welcher Parameter da [beim Aufruf der Methode *schalten* der Hintergrundbeleuchtung] rein muss

Schüler A, Z. 100–102

Dass „das Programm sich selbst [-] einmal abfragt“ (Schüler A, 100f), ist so natürlich nicht richtig. Um in der Sprechweise der Schüler zu bleiben, fragt Martin, bzw. der Computer, den Helligkeitssensor, aber weder das Programm noch Martin fragen sich selbst. Schüler B versteht den Computer als Martin, was korrekt ist: „Martin war ja jetzt sozusagen der Computer“ (Schüler B, Z. 110).

Kapitel 6

Fazit

Im letzten Kapitel wurden die Interviews ausgewertet und es hat sich gezeigt, dass die vorher festgelegten Lernziele für das Unterrichtsvorhaben erreicht werden konnten. Die Interviews haben die Schülerinnen und Schüler auch für eine Bewertung der Arbeit zu Wort kommen lassen, sodass auch ihre Meinung zu dem Ansatz dieser Masterarbeit vorliegt. In diesem Kapitel werden nun die Erkenntnisse aus den Unterrichtsbeobachtungen und Interviews zusammengeführt, sodass ein Fazit gezogen werden kann.

6.1 Vergleich zu klassischen Herangehensweisen

Klassische Ansätze behandeln das Thema »Objektorientierung« meist nur theoretisch und verzichten somit auf den Hardware-Anteil, der den hier vorgestellten Ansatz ausmacht. Gerade der praktische Anteil gefällt Schüler A, der auch den gelungenen Wechsel von der Theorie zur Praxis betont:

[...] ich finde es eigentlich super, weil wir vorher halt auch erstmal theoretisch alles gemacht haben, damit wir halt auch erstmal einen groben Überblick haben, wie das überhaupt geht, und dass wir danach (.) an die Pis gegangen sind, wir haben ja auch ein Arbeitsblatt bekommen, und das konnten wir dann quasi Schritt für Schritt durchgehen

Schüler A, Z. 168–172

Besonders treffend bringt es Schüler C an einer Stelle auf den Punkt. Für ihn hat der Einsatz der Hardware und die Arbeit mit der Groovy-Console zum Verständnis beigetragen (er fand die Arbeit mit dem Raspberry Pi interessant und „ganz cool“, vgl. Schüler C, Z. 181–182), da das Thema damit nicht nur theoretisch behandelt wurde:

Ich glaub schon, dass es mehr Vorteile gibt, weil man halt dadurch selbst auch Sachen ausprobieren kann. Und da viele Leute die Sachen halt besser verstehen, wenn man sie selbst in der Praxis umsetzt, als wenn man sie nur in der Theorie durchnimmt. Und ich persönlich fand das Objektspiel mit den Karten immer sehr verwirrend und hab das nie so ganz verstanden. Aber wenn man das dann halt in diese Groovy Konsole eingegeben hat, wurde es für mich halt immer deutlicher und ich habe es halt besser verstanden. Und da glaub ich halt schon, dass diese (.) Praxisarbeit zum Verständnis beiträgt

Schüler C, Z. 202–209

Schüler F betont die Möglichkeiten, selbstständig mit den Hardware-Bausteinen zu arbeiten. Die Geschichte mit den vielen Dioden war gerade so ausgelegt, dass eine eigenständige Überprüfung des Quelltextes möglich war (s. auch Kapitel 3.2), worauf auch Schüler F aufmerksam macht:

[...] und außerdem konnte man dann eben auch gucken, sich selbst überprüfen, also ob man es richtig gemacht hat, indem man das dann halt ablaufen lassen hat, und das musste nicht vom Lehrer überprüft werden

Schüler F, Z. 155–157

Es ist natürlich Grundvoraussetzung, dass die Hard- und Software ohne Probleme funktionieren. Leider gab es an dieser Stelle anfangs Probleme, da Schülerinnen und Schüler unterschiedliche Versionen des Betriebssystems verwendeten (Raspbian¹) und auch unterschiedliche Java-Versionen installiert waren. Schüler F beschreibt diese Phase als „chaotisch“ (vgl. Schüler F, Z. 145–147).

Für die Zukunft kann sich wie folgt geholfen werden. Da das Betriebssystem des Raspberry Pi auf einer SD-Karte installiert ist, können SD-Karten vorbereitet werden, die alle über dieselbe Java-Version verfügen, die mit der rpCollection und Groovy funktioniert. Dann wird keine Unterrichtszeit für die Installation aufgewendet, da alles schon vorbereitet ist.

6.2 Probleme und Vorteile des Hardwareeinsatzes

Nachdem nun dargestellt wurde, wie die Schülerinnen und Schüler die Arbeit mit dem Raspberry Pi und den Hardware-Bausteinen bewerten, soll nun noch einmal auf Vor- und Nachteile dieses Ansatzes eingegangen werden. Es wird auch berichtet, an welchen Stellen noch Verbesserungspotential besteht, und was zu beachten sind, soll der Ansatz in anderen Schulen verfolgt werden.

¹ Raspbian – <https://www.raspberrypi.org/downloads/raspbian/>. Der Name ist eine Mischung aus »Raspberry« und »Debian«.

Unterricht Im Unterricht wurden die Sozialformen immer wieder gewechselt. Wie oben erwähnt, gefällt den Schülerinnen und Schülern dabei die zunächst theoretische Behandlung des Themas und der darauf folgende Wechsel an die Raspberry Pis. Das Objektspiel hat dazu beigetragen, Methodenaufrufe für den ganzen Kurs ersichtlich darzustellen und auch Themen wie »Beziehungen« zu erklären.

In den Interviews haben die Schüler natürlich keine Definition der Begriffe »Objekt«, »Methode« oder »Beziehung« genannt, was durchaus zu erwarten und natürlich auch kein Lernziel des Unterrichts war, aber ihr Benutzen der Begriffe hat gezeigt, dass sie diese Begriffe verstehen. Eine Definition solcher Begriffe wird z. B. in [Armstrong 2006, S. 124ff] gegeben, ist aber allgemein schwierig. Es gibt nicht *die* Definition dieser Begriffe.

Die Schülerinnen und Schüler erkennen die Objektorientierung nicht als Programmierparadigma. Dies ist aber auch nicht verwunderlich, da sie kein anderes kennen, und die Rechtfertigung der Objektorientierung als Paradigma erscheint erst dann sinnvoll, wenn sie über Abgrenzungswissen zu anderen Programmieransätzen verfügen.

Kontexte Wie in Kapitel 4.6 beschrieben, haben die Schülerinnen und Schüler schon während der Theater-Geschichte angemerkt, dass die Positionierung der Scheinwerfer *auf* der Bühne untypisch sei (Attributwert »Bühne«). Darüber hinaus wäre in der Geschichte eine zweite Stelle gut gewesen, an der die Schülerinnen und Schüler eine komplexere Objektkommunikation hätten modellieren müssen. Im Unterricht wurde somit nur das Schalten der Hintergrundbeleuchtung behandelt, wenn dafür auch sehr ausführlich. In diesem Zusammenhang wurde im Unterricht auch diskutiert, welche Vorteile die verschiedenen Modellierungen haben. Eine andere Situation hätte sich gut geeignet, die Begriffe »Parameter« und »Rückgabe« zu wiederholen.

Diese Überlegungen sind in das überarbeitete Arbeitsblatt der Geschichte eingeflossen (Anhang S. 129). Dort wurde die Anzahl der anfänglich an- und auszuschaltenden Scheinwerfer reduziert, da in den Interviews u. a. kritisiert wurde, dass am Anfang oft dasselbe eingeben werden musste (vgl. Schüler C, Z. 172–175).

Für eine weitere Stelle komplexerer Objektkommunikation wurden in der überarbeiteten Geschichte zwei Deckenscheinwerfer eingeführt, die nur zusammen ein- und ausgeschaltet werden können. Diese hängen an einer Deckenkonstruktion, sodass ein Einschalten der Deckenbeleuchtung wie in Sequenzdiagramm 6.1 abläuft².

² Ein Schalten der Deckenbeleuchtung wäre natürlich auch mit dem Helligkeitssensor möglich. Auf eine Darstellung wird hier verzichtet.

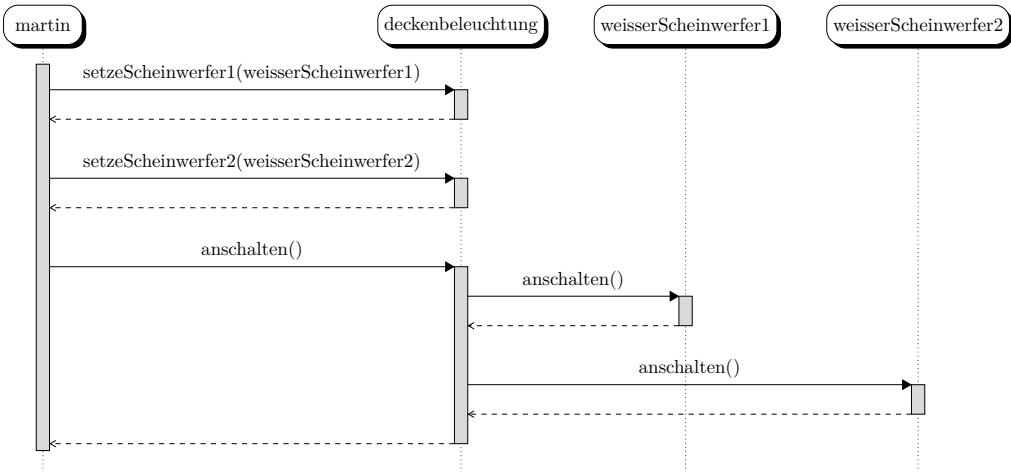


Abbildung 6.1: Das Anschalten des Deckenlichts zieht das Anschalten der beiden weißen Scheinwerfer nach sich.

Die übrigen Scheinwerfer sind in der Geschichte links und rechts der Bühne positioniert. Durch die weiteren Informationen könnte das Theaterprojekt auch tatsächlich realisiert werden. Motivierte Schülerinnen und Schüler könnten dies in einem Schuhkarton nachbauen; kleine Löcher in den Seitenwänden bzw. dem Deckel könnten dabei als Fassungen für die Dioden dienen.

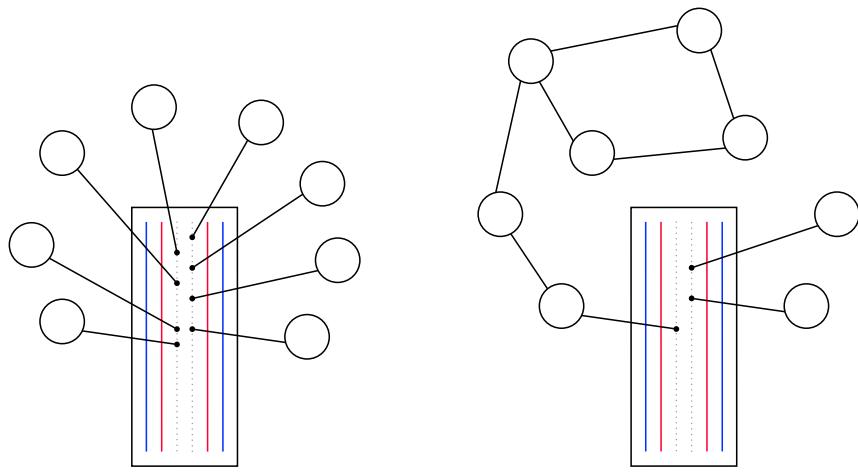
Der Text hat sich auch dazu geeignet, über ein paralleles Ausführen von Methoden zu sprechen. So wollten die Schülerinnen und Schüler beim Objektspiel bei mehreren Objekten gleichzeitig eine Methode aufrufen (beim Ausschalten der Scheinwerfer). Da dies aber ohne weiteres nicht möglich ist, müssen die Methodenaufrufe nacheinander erfolgen.

Hardware Auf Seiten der Hardware muss man natürlich in Betracht ziehen, dass die Schule entsprechend ausgestattet sein muss (Raspberry Pis, HDMI-Monitore, USB-Tastaturen u. Mäuse etc.) bzw. Möglichkeiten zur Anschaffung bereitstellen kann. Diese Investitionen sind natürlich unumgänglich, fallen dafür aber vergleichsweise gering aus, da i. d. R. nur die Raspberry Pis und die Hardware für die Steckbretter angeschafft werden müssen (GPIO-Steckbrett-Adapter, Flachbandkabel, Jumperkabel etc.).

Der Arbeitsaufwand für die Vorbereitung muss natürlich auch eingeplant werden. Dabei nimmt das Löten der Bauteile sicherlich die meiste Zeit in Anspruch. Die Klassen zum Ansteuern der Hardware sind bereits vorbereitet, sodass lediglich die Klassen- und Methodennamen für die Modellierung der Schülerinnen und Schüler angepasst werden müssen. In Zukunft könnte das Erstellen der Hardware-Bausteine evtl. durch Ätz-Vorlagen für die Platinen erleichtert werden. Dann bräuchten nur

noch die Bauteile aufgelötet werden. Dies sind Aufgaben, die auch interessierte Schülerinnen und Schüler übernehmen können.

Das Verbinden der Hardware mit dem Steckbrett des Raspberry Pi hat sehr gut geklappt. Es wurde kein Bauteil durch falsches Anschließen zerstört. Durch den Anschluss der Hardware-Bausteine wird eine Sterntopologie am Steckbrett erzeugt (Abbildung 6.2 (a), vgl. auch Abbildung 3.5). Ein Anschluss wie in Abbildung 6.2 (b) ist hingegen nicht möglich, sodass eine »Kette« an Objekten (ein Objekt fragt ein anderes, welches ein anderes fragt u. s. w.) nicht Eins-zu-Eins durch die Hardware-Bausteine abgebildet werden kann³. Dies liegt hauptsächlich daran, dass der Pin-Status für jedes Glied der Kette einzeln ausgelesen muss, wofür eine direkte Verbindung zum Steckbrett notwendig ist.



(a) Objekte in einer Stern topologie.

(b) Objekte in einem Netzwerk.

Abbildung 6.2: Anschlussarten von Bauteilen am Steckbrett des Raspberry Pi.

Interessant wäre noch die Möglichkeit, beim Hafenkran-Projekt die Drehung des Krans über ein Gyroskop steuern zu können. Dann könnte man durch Neigen der Platine den Kran drehen. Da die Drehrichtung aber auch erstmal anders beeinflusst werden kann (AD-Wandler, Taster), wurde das Thema für diese Masterarbeit ausgelassen.

Umgang mit Vorerfahrungen In den Interviews wurde deutlich, dass nahezu alle interviewten Schüler keine Vorerfahrungen mit Informatik hatten (auch nicht aus der Schule, vgl. HK 0). Interessant ist hier die Trennung zwischen dem alltäglichen

³ Am ehesten kommt der AD-Wandler dieser Topologie nach: Der Raspberry Pi fragt den AD-Wandler, welcher wiederum einen Regler fragt. Trotzdem sind die Regler auf der Platine des AD-Wandlers festgelötet worden (s. Abbildung C.10(e)) und können nicht separat an- oder abgesteckt werden. Dies wäre evtl. eine Option für die Zukunft.

Arbeiten mit dem Computer und der Informatik bzw. dem informatischen Verständnis der Schüler (vgl. Schüler D, Z. 18–22). Für sie hat die tägliche Arbeit mit dem Computer nichts mit Informatik zu tun (was durchaus korrekt ist).

Die Vorerfahrungen (Hauptkategorie 0) sind noch einmal nach in- und extrinsisch motivierte Vorerfahrungen unterteilt. Schüler E beschreibt, wie er schon sehr früh begonnen hat, sich für den Computer zu interessieren. Er hat auch schon Vorerfahrungen im Bereich der Spieleprogrammierung sammeln können:

Ja, also ich hab relativ früh angefangen, mich dafür zu interessieren. So bestimmt mit sieben, acht habe ich dann irgendwann angefangen eben mit dem Rechner von meinen Eltern damals so ein bisschen rum zu spielen, alles auszuprobieren und dann irgendwann hab ich mir ein Buch ausgeliehen, so ein QBasic-Buch, da hab ich dann so ganz, aber auch nur wirklich so vier, fünf Befehle gelernt, da war ich höchstens acht oder neun, und wirklich angefangen mit Java, also da hab ich jetzt auch schon wieder das meiste verlernt, aber wirklich angefangen habe ich dann... da haben wir Minecraft gespielt und da haben wir Plug-Ins angefangen zu programmieren dafür. Für Server so eigene Game-Modi und so, da hab ich dann wirklich damit angefangen und ja, ich weiß nicht, ich interessiere mich dafür, auch für Hardware, ich habe auch meinen PC zum Beispiel selber zusammen gestellt und so, und ja, also (1)

Schüler E, Z. 7–19

Schüler C wurde von seinem Onkel angeregt, sich mit der Programmierung eines LEGO-Roboters zu beschäftigen, nutzte diese Möglichkeit aber kaum (vgl. Schüler C, Z. 10–24). Schüler E betont die Vorteile des Hardwareeinsatzes und die Möglichkeit, auch ohne Vorerfahrungen in die Programmierung in Groovy einzusteigen:

Ich finde das hat auch wirklich Sinn gemacht, vor allen Dingen ich hab ja jetzt auch schon ein bisschen was an Programmiererfahrung, aber ich glaube, wenn man da keine hat, dann war das auf jeden Fall sinnvoll, so einzusteigen, weil wirklich, also auch den Pi hat man ja was, was man wirklich anfassen kann, angucken, man sieht vielleicht so ein bisschen sogar, was da passiert, und auch durch die Methoden, dass die dann was ausführen, dass dann wirklich was passiert, also das fand ich auf jeden Fall, ich fand das gut. Auch dass man nicht auf so viel achten musste, auf irgendwelche Klammern oder so und dabei wirklich jetzt, ich fand das schön, dass das jetzt so vereinfacht war jetzt

Schüler E, Z. 107–115

Zusammengefasst lässt sich sagen, dass die Schüler teilweise schon Vorerfahrungen mit der Programmierung hatten (vgl. Schüler B, Schüler F), aber die Objektorientierung für sie neu war. Schüler B betont, dass es durch die unterschiedlichen Vorerfahrungen teilweise zu Wartezeiten im Unterricht kam (vgl. Schüler B, Z. 119–121). Dies ist aber gerade im Fach Informatik, mit dem man sich auch sehr gut alleine zuhause beschäftigen und Vorerfahrungen sammeln kann, fast nicht zu vermeiden.

6.3 Verbindung zu Fehlvorstellungen

In Kapitel 2.3.3 wurden einige Beispiele für Fehlvorstellungen diskutiert, die hier noch einmal explizit erwähnt werden sollen. Im Folgenden werden diese Fehlvorstellungen bzgl. des Hardwareeinsatzes diskutiert und es wird erklärt, wie sie mit dem Ansatz dieser Masterarbeit vermieden oder provoziert werden.

Objekte haben Namen Diese Fehlvorstellung wurde im Unterricht natürlich provoziert. Einerseits wurde im Objektspiel immer vom »Objekt mit Namen ...« gesprochen, andererseits wurde bei der Arbeit mit Groovy auch der eigentliche Variablenname bei der Instanziierung des Objekts für das Objekt gehalten. Da dies aber der allererste Kontakt mit Objekten war, ist ein anderes Vorgehen hier kaum denkbar. Die genaueren Zusammenhänge lassen sich mit ein wenig mehr Erfahrung (und dem Wissen um Klassen) besser diskutieren. In der Theater-Geschichte kam kein Attribut `name` vor, sodass hier Verwechslungen mit dem Namen des Objekts ausgeschlossen waren.

Die Zustandsfelder von Objekten definieren ihre Attribute Durch das Objektspiel und das Identifizieren von Attributen durch das Verfahren von ABBOTT konnten die Schülerinnen und Schüler überwiegend gut zwischen Attributen und Methoden trennen. In den Interviews kam es zu Verwechslungen der Begriffe »Attributwert« / »Attribut« und »Parameterwert« / »Parameter«.

Objekte enthalten Objekte Diese Fehlvorstellung könnte durch die zweite Modellierung zum Schalten der Hintergrundbeleuchtung hervorgerufen worden sein; konnte jedoch nicht beobachtet werden. Die Darstellung im Objektdiagramm (wie in Abbildung 4.4) und die Modellierung im Objektspiel als gerichtete Beziehung (mit Garn u. Pfeil) verdeutlichen aber, dass es sich nicht um eine »enthält«-Beziehung handelt.

Dieser Aspekt wurde indirekt auch bei den Varianten zum Schalten der Hintergrundbeleuchtung diskutiert (Anhang A.2.11). Durch Dereferenzieren der Hintergrundbeleuchtung entfällt evtl. die Zugriffsmöglichkeit auf den Helligkeitssensor (Variante B I), aber das entsprechende Objekt ist nach wie vor existent.

Mit Vererbung werden Begriffshierarchien abgebildet Diese Fehlvorstellung ist für den vorstellten Ansatz zu diesem Zeitpunkt noch nicht relevant, da die Vererbung noch nicht diskutiert wurde. Bei der Erstellung der Klassen der rpCollec-

tion wurde darauf geachtet, die Klassen möglichst ohne Vererbung zu realisieren, um so *eine* Klasse für einen Scheinwerfer zu haben. Auf der anderen Seite kann man auch für die Vorteile der Vererbung als klassisches Konzept der Software-Entwicklung argumentieren.

Objekte sind so etwas wie Variablen Da überwiegend alle Objekte mehrere Attribute hatten (`status`, `standort` etc.), konnte diese Fehlvorstellung nicht beobachtet werden. Zudem war es für die meisten Schülerinnen und Schüler der erste Kontakt mit der OOP (keine Vorerfahrungen mit prozeduraler Programmierung, s. oben).

Objekte sind so etwas wie Datenbanken (records) Aufgrund der Tatsache, dass die Objekte nicht nur über Getter- und Setter-Methoden verfügten und durch die Hardware-Bausteine auch buchstäblich anzufassen waren, konnte diese Fehlvorstellung nicht beobachtet werden (und wurde auch gar nicht erst provoziert).

In Methoden passiert die Arbeit durch das Assignment Diese Fehlvorstellung konnte bei Schüler C beobachtet werden (vgl. Schüler C, Z. 156–158, vgl. Kapitel 5.3.4). In der Geschichte gibt es aber auch Methodenaufrufe, die ohne ein Assignment auskommen (Befragen des Helligkeitssensors). Auf der anderen Seite muss festgehalten werden, dass diese Fehlvorstellung evtl. dadurch provoziert wurde, dass auch beim Ausschalten schon ausgeschalteter Scheinwerfer der Attributwert des Attributs `status` von `aus` auf `aus` geändert wurde (vgl. Schüler C, Z. 146–148).

Verwechslung »Klasse« und »Objekt« Da die Schülerinnen und Schüler den Begriff »Klasse« noch nicht kennen, konnte dieser Fehlvorstellung nur vorgebeugt werden. Die Schülerinnen und Schüler haben selbst angemerkt, dass man am Anfang eine Vielzahl von Scheinwerfern hatte (genau genommen verschiedene Objekte der Klasse `scheinwerfer`). So ist der fundamentale Unterschied klar, dass es zu einer Klasse mehrere Objekte geben kann (s. auch Kapitel 2.3.4). Auch dass die Klassen eine zentrale Rolle spielen, ist den Schülerinnen und Schülern an dieser Stelle schon klar geworden.

6.4 Weitere Anmerkungen

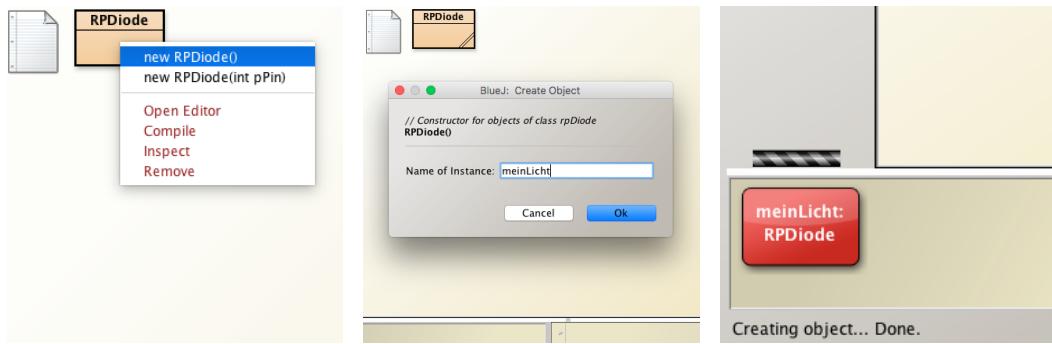
Dem »Objects first«-Ansatz folgend, fand die Arbeit zu Beginn des Unterrichtsvorhabens allein auf Basis von Objekten statt. Es wurden Objekte erzeugt, Attributwerte

gesetzt und abgefragt und Methoden aufgerufen. Im Unterricht wurde mit der Groovy-Console gearbeitet und Objekte wie in Quelltext 6.1 erstellt.

```
1 meinLicht = new Scheinwerfer(12);
```

Quelltext 6.1: Objekterstellung in Groovy.

An dieser Stelle sei angemerkt, dass ein Erzeugen von Objekten prinzipiell auch direkt in BlueJ geschehen kann. Dort wird für die Instanziierung und den Aufruf einer Methode allerdings nicht die Java-typische Notation gewählt, sondern sämtliche Aktionen sind über das BlueJ-Interface auszuführen (Abbildung 6.3).



(a) Neues Objekt erzeugen. (b) Name des Objekts eingeben (»Name of Instance«). (c) Das Objekt ist erzeugt.

Abbildung 6.3: Ein Objekt der Klasse `RPDiode` wird über die GUI erzeugt (BlueJ).

Durch den Ansatz mit Groovy wird dem Problem eines teilweise verwirrenden User-Interfaces aus dem Weg gegangen (vgl. „Guideline 8: Be careful with the user interface.“ [Kölling Rosenberg 2001, S. 3]). Auch wenn sich BlueJ durch die Möglichkeit auszeichnet, Klassen sehr einfach im User-Interface instanziieren zu können, geht das Programm damit einen anderen Weg als andere Entwicklungsumgebungen. KÖLLING und ROSENBERG schreiben dazu Folgendes:

BlueJ has some capabilities that are not commonly available in other Java development environments which lead to a different approach to software development.

[Kölling Rosenberg 2001, S. 1]

Am Ende ist es also eine Entscheidung der Lehrenden, welchen Weg sie gehen wollen. Der Weg über Groovy ist näher an der klassischen Java-Programmierung, der Weg mittels BlueJ dafür vielleicht übersichtlicher und weniger fehleranfällig (es können z. B. kaum Syntaxfehler beim Methodenaufruf gemacht werden, vgl. Quelltext 4.1). Diese Masterarbeit setzte aber Groovy ein, um einerseits den Besonderheiten von BlueJ aus dem Weg zu gehen und andererseits dem klassischen Programmieren nachzukommen.

Sind in BlueJ Objekte erzeugt worden, werden diese übersichtlich am unteren Rand des Fensters gruppiert und können inspiziert werden, um bspw. Attributwerte auszulesen. Methoden können durch einen Rechtsklick auf das Objekt ausgeführt werden (vgl. Abbildung 6.4).

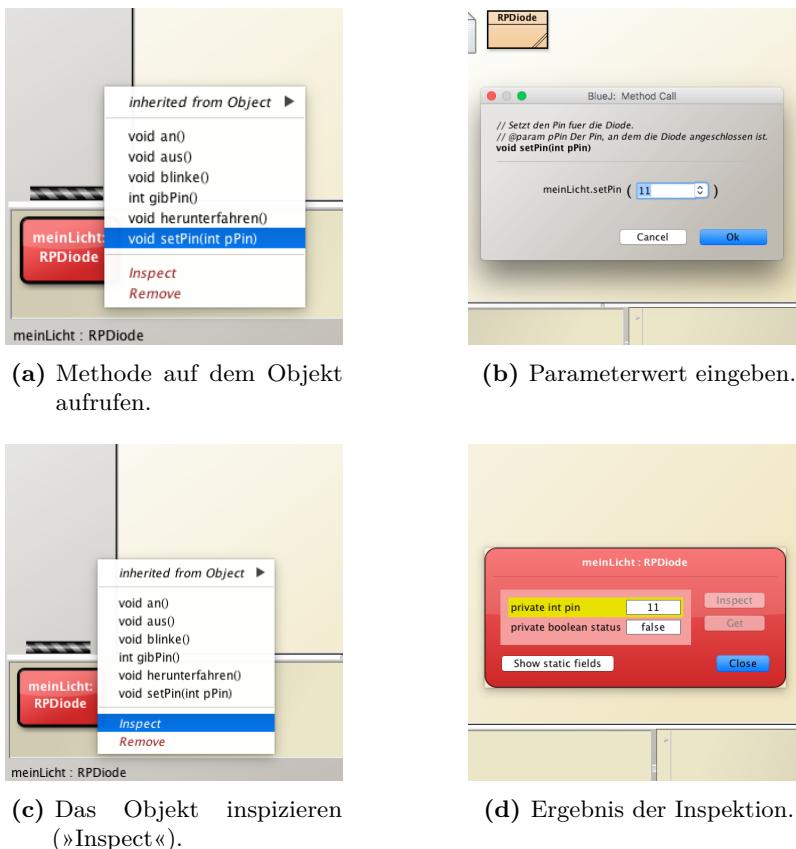


Abbildung 6.4: Methode auf einem Objekt aufrufen und Objekt inspizieren (BlueJ).

Die Hardware-Bausteine sind auch in anderen Jahrgangsstufen einsetzbar. In der Klasse 5 haben die Schülerinnen und Schüler einer freiwilligen Computer-AG mit Scratch⁴ gearbeitet. Dort kann ein sog. »GPIO-Server« gestartet werden, der dann ein Ansprechen der Pinne wie in Abbildung 6.5 ermöglicht. Die Schülerinnen und Schüler hatten besonders Freude daran, die Dioden blinken zu lassen oder Töne mit dem Summer zu erzeugen. Die in Kapitel 3.2 beschriebene Motivation durch die Hardware konnte an dieser Stelle beobachtet werden, auch wenn ihr Arbeiten mit den Hardware-Bausteinen nicht mit dem Unterricht in Verbindung stand.

⁴ Scratch – <https://scratch.mit.edu/>.



Abbildung 6.5: Blinkende LED in Scratch: Die LED an Pin »IO02« blinkt fünf Mal, wenn die Taste »b« gedrückt wurde.

6.5 Abschlussfazit

Der in dieser Arbeit vorgestellte Ansatz stellt einen praktikablen Weg dar, Schülerinnen und Schülern die anfänglichen Konzepte der Objektorientierung zu vermitteln. Die Hardware-Bausteine wirken dabei motivierend und der praktische Anteil ist eine Abwechslung vom theoretischen Unterricht. Auch wenn die Vorbereitung einiger Zeit bedarf und die technische Ausstattung auf Seiten der Schule vorhanden sein muss, sind sie eine gute Möglichkeit, einen Einstieg in die Objektorientierung zu bieten. Dies wurde durch die Auswertung der Interviews im vorangegangenen Kapitel gezeigt. Der Theater-Kontext wurde noch einmal überarbeitet, als die Schwachpunkte der Geschichte im Unterricht und in den Interviews identifiziert wurden. So soll die Einführung der Anfangskonzepte der Objektorientierung gelingen, ohne die aufgezeigten Fehlvorstellungen zu stark zu provozieren.

Die in der Einleitung formulierten Hauptfragen wurden beantwortet. Nach einer Darstellung der Theorie zur Einführung in die OOP wurden auch die wichtigsten Fehlvorstellungen dargelegt. In Kapitel 6.2 wurden die Fehlvorstellungen in Bezug auf die Hardware-Bausteine diskutiert und u. a. gezeigt, welche sich durch den vorgestellten Ansatz vermeiden lassen. Die Interviews wurden auch dazu genutzt, um die Praxistauglichkeit des vorgestellten Ansatzes zu überprüfen.

Abschließend lässt sich sagen, dass die Vorteile durch den Einsatz der Hardware auf der Hand liegen und die Hardware-Bausteine eine gute Möglichkeit bieten, einen abwechslungsreichen Unterricht zu gestalten. Er ist weder zu theoretisch, noch zu computerorientiert, und fokussiert sich auch nicht nur auf eine Programmiersprache. Der Wechsel von der Theorie (Modellierung) zur Programmierpraxis kann

durch die Bausteine begleitet werden. Sie unterstützen das Lernen, sodass auch leistungsschwächeren Schülerinnen und Schülern schnell Erfolgserlebnisse ermöglicht werden können. Trotz unterschiedlicher Vorerfahrungen konnte bei allen interviewten Schülerinnen und Schülern das geforderte Objektverständnis festgestellt werden, wenn auch teilweise mit kleineren Schwächen. Da der hier vorgestellte Ansatz aber lediglich den Einstieg in die Objektorientierung begleitete, wird es im Unterricht noch viele Möglichkeiten geben, das Objektverständnis zu vertiefen und weitere Erfahrungen zu sammeln.

Literaturverzeichnis

- [Abbott 1983] Russel J. ABBOTT (1983): *Programm Design by Informal English Descriptions*. In: *Communications of the ACM*, Ausgabe 26 (November 1983), Nr. 11, S. 882–894.
- [Alphonse Ventura 2002] Carl ALPHONSE, Phil VENTURA (2002): *Object Orientation in CS1-CS2 by Design*. In: *ITiCSE 2002*, Aarhus, Dänemark.
- [Altrichter Posch 2007] Herbert ALTRICHTER, Peter POSCH (2007): *Lehrer erforschen ihren Unterricht: Unterrichtsentwicklung und Unterrichtsevaluation durch Aktionsforschung*. Klinkhardt. Bad Heilbrunn.
- [Arif 2000] Essam M. ARIF (2000): *A Methodology for Teaching Object-Oriented Programming Concepts in an Advanced Programming Course*. In: *SiGCSE 2000*, Austin, Texas, USA.
- [Armstrong 2006] Deborah J. ARMSTRONG (2006): *The Quarks of Object-Oriented Development* In: *Communications of the ACM*. Februar 2006 / Vol. 49, No. 2.
- [Barnes Kölling 2002] David BARNES, Michael KÖLLING (2002): *Objects First with Java: A Practical Introduction using BlueJ*. 1. Auflage. Pearson Education. Harlow, England.
- [Bennedsen Caspersen 2004] Jens BENNEDSEN, Michael E. CASPERSEN (2004): *Programming in Context – A Model-First Approach to CS1* In: *SiGCSE 2004*, Norfolk, Virginia, USA.
- [Cooper Dann et. al. 2003] Stephen COOPER, Wanda DANN, Randy PAUSCH (2003): *Teaching Objects-first In Introductory Computer Science*. In: *SiGCSE 2003*, Reno, Nevada, USA.
- [Dembowski 2015] Klaus DEMBOWSKI (2015): *Raspberry Pi – Das technische Handbuch. Konfiguration, Hardware, Applikationserstellung*. 2., erweiterte und überarbeitete Auflage. Springer Vieweg. Wiesbaden.

[Eckerdal Thuné 2005] Anna ECKERDAL, Michael THUNÉ (2005): *Novice Java programmers' conceptions of "object" and "class", and variation theory*. In: *ITiCSE 2005*, Monte de Caparica, Portugal.

[Ehlert Schulte 2009] Albrecht EHLERT, Carsten SCHULTE (2009): *Empirical comparison of objects-first and objects-later*. In: *Proceedings of the fifth international workshop on Computing education research workshop – ICER 2009*.

[Engbring 2009] Dieter ENGBRING (2009): *Wozu objektorientiertes Programmieren? Versuch einer Begründung aus der Informatik-Geschichte*. In: *LOG IN*, Heft Nr. 157 / 158 (2009).

[Helfferich 2011] Cornelia HELFFERICH (2011): *Die Qualität qualitativer Daten – Manual für die Durchführung qualitativer Interviews*. 4. Auflage. VS Verlag.

[Holland Griffiths et. al. 1997] Simon HOLLAND, Robert GRIFFITHS, Mark WOODMAN (1997): *Avoiding Object Misconceptions*. In: *SiGCSE 1997*, San Jose, California, USA.

[Humbert 2006] Ludger HUMBERT (2006): *Didaktik der Informatik: mit praxiserprobtem Unterrichtsmaterial*. 2., überarbeitete und erweiterte Auflage. Teubner-Verlag. Wiesbaden.

[Kernlehrplan Informatik 2014] MINISTERIUM FÜR SCHULE UND WEITERBILDUNG DES LANDES NORDRHEIN-WESTFALEN (Hrsg.) (2014): *Kernlehrplan für die Sekundarstufe II Gymnasium / Gesamtschule in Nordrhein-Westfalen. Informatik*. 1. Auflage. Düsseldorf.

[Koller 2014] Hans-Christoph KOLLER, Gereon WULFTANGE (Hrsg.) (2014): *Lebensgeschichte als Bildungsprozess? Perspektiven bildungstheoretischer Biographieforschung*. transcript-Verlag. Bielefeld.

[Kölling I 1999] Michael KÖLLING (1999): *The Problem of Teaching Object-Oriented Programming, Part I: Languages* In: *Journal of Object-Oriented Programming*, 11(8): 8–15, 1999.

[Kölling II 1999] Michael KÖLLING (1999): *The Problem of Teaching Object-Oriented Programming, Part II: Environments* In: *Journal of Object-Oriented Programming*, 11(9): 6–12, 1999.

- [Kölling Koch et. al. 1995] Michael KÖLLING, Bett KOCH, John ROSENBERG (1995): *Requirements For A First Year Object-Oriented Teaching Language*. In: *SiGCSE 1995*, Nashville, Tennessee, USA. S. 173–177.
- [Kölling Rosenberg 2001] Michael KÖLLING, John ROSENBERG (2001): *Guidelines for Teaching Object Orientation with Java*. In: *The Proceedings of the 6th conference*. In: *ITiCSE 2001*, Canterbury, Großbritannien.
- [Kortenkamp Modrow et. al. 2009] Ulrich KORTENKAMP, Eckart MODROW, Reinhard OLDENBURG, Jürgen POLOCZEK, Magnus RABEL (2009): *Objektorientierte Modellierung — aber wann und wie? Zur Bedeutung der OOM im Informatikunterricht*. In: *LOG IN*, Heft Nr. 160 / 161 (2009).
- [Kuckartz 2016] Udo KUCKARTZ (2016): *Qualitative Inhaltsanalyse. Methoden, Praxis, Computerunterstützung*. 3., überarbeitete Auflage. Beltz Juventa. Weinheim.
- [Lewis 2000] John LEWIS (2000): *Myths about Object-Orientation and Its Pedagogy*. In: *SiGCSE 2000*, Austin, Texas, USA.
- [Luker 1994] Paul A. LUKER (1994): *There's More to OOP Than Syntax!* In: *SiGCSE 1994*, Phoenix, Arizona, USA.
- [Or-Bach Lavy 2004] Rachel OR-BACH, Ilana LAVY (2004): *Cognitive Activities of Abstraction in Object Orientation: An Empirical Study* In: *SiGCSE 2004*, Norfolk, Virginia, USA.
- [Pollard Duvall 2006] Shannon POLLARD, Robert C. DUVALL (2006): *Everything I Needed to Know About Teaching I Learned in Kindergarten: Bringing Elementary Education Techniques to Undergraduate Computer Science Classes*. In: *SiGCSE 2006*, Houston, Texas, USA.
- [Poon 2000] Josiah POON (2000): *Java meets teletubbies: an interaction between program codes and physical props*. In: *ACE 2000* 12/00 Melbourne, Australia.
- [Sanders Thomas 2007] Kate SANDERS, Lynda THOMAS (2007): *Checklists for Grading Object-Oriented CS1 Programs: concepts and misconceptions*. In: *ITiCSE 2007*, Dundee, Großbritannien.
- [Schmolitzky 2006] Axel SCHMOLITZKY (2006): *Sieben Thesen zur erfolgreichen Verwirrung von Anfängen der objektorientierten Programmierung*. Springer-Verlag.

- [Schriek 2005] Bernard SCHRIEK (2005): *Informatik mit Java – Eine Einführung mit BlueJ und der Bibliothek Stifte und Mäuse – Band I*. Nili-Verlag. Werl.
- [Schriek 2006] Bernard SCHRIEK (2006): *Informatik mit Java – Eine Einführung mit BlueJ und der Bibliothek Stifte und Mäuse – Band II*. Nili-Verlag. Werl.
- [Schriek 2007] Bernard SCHRIEK (2007): *Informatik mit Java – Eine Einführung mit BlueJ und der Bibliothek Stifte und Mäuse – Band III*. Nili-Verlag. Werl.
- [Schubert Schwill 2011] Sigrid SCHUBERT, Andreas SCHWILL (2011): *Didaktik der Informatik*. 2. Auflage. Spektrum Akademischer Verlag. Heidelberg.
- [Selter Spiegel 1997] Christoph SELTER, Hartmut SPIEGEL (1997): *Wie Kinder rechnen*. 1. Auflage. Klett-Grundschulverlag. Leipzig, Stuttgart, Düsseldorf.
- [Stevens Henry et. al. 2000] K. Todd STEVENS, Joel HENRY, Pamela B. LAWHEAD, John LEWIS, Constance BLAND, Mary Jane PETERS (2000): *Using Large Projects in a Computer Science Curriculum*. In: *SiGCSE 2000*, Austin, Texas, USA.
- [Thomasson Ratcliffe et. al. 2006] Benjy THOMASSON, Mark RATCLIFFE, Lynda THOMAS (2006): *Identifying novice difficulties in object oriented design*. In: *SiGCSE 2006*, Houston, Texas, USA.
- [Uysal 2012] Murat Pasa UYSAL (2012): *The Effects of Objects-First and Objects-Late Methods on Achievements of OOP Learners*. In: *Journal of Software Engineering and Applications*, 2012, 5, S. 816–822.
- [Zhu Zhou 2003] Haibin ZHU, Mengchu ZHOU (2003): *Methodology First and Language Second: a Way to Teach Object-Oriented Programming* In: *Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications – OOPSLA 2003*.

Internetquellenverzeichnis

[Abiturvorgaben NRW] QUALITÄTS- UND UNTERSTÜTZUNGSAGENTUR – LANDESINSTITUT FÜR SCHULE (Hrsg.) (2014): *Zentralabitur in der gymnasialen Oberstufe. Informatik.* <https://www.standardsicherung.schulministerium.nrw.de/cms/zentralabitur-gost/faecher/fach.php?fach=15>, aufgerufen am 29.11.2016.

[Adafruit] Michael SKLAR (2012): *Analog Inputs for Raspberry Pi Using the MCP3008*, <https://learn.adafruit.com/reading-a-analog-in-and-controlling-audio-volume-with-the-raspberry-pi/connecting-the-cobbler-to-a-mcp3008>, aufgerufen am 27.09.2016.

[Bergin 2000] Joseph BERGIN (2000): *Why Procedural is the Wrong First Paradigm if OOP is the Goal*, <https://csis.pace.edu/~bergin/papers/Whynotproceduralfirst.html>, aufgerufen am 07.11.2016.

[BlueJ Adjustable LED] BlueJ: *Adjustable LED Tutorial*, <http://www.bluej.org/raspberrypi/PWMLed.html>, aufgerufen am 27.09.2016. Direktlink: <http://www.bluej.org/raspberrypi/projects/AdjustableLED.zip>, aufgerufen am 27.09.2016.

[BlueJ Button] BlueJ: *Button Tutorial*, <http://www.bluej.org/raspberrypi/button.html>, aufgerufen am 27.09.2016.

[BlueJ LED] BlueJ: *LED Tutorial*, <http://www.bluej.org/raspberrypi/led.html>, aufgerufen am 27.09.2016. Direktlink: <http://www.bluej.org/raspberrypi/projects/LEDButton.zip>, aufgerufen am 27.09.2016.

[DDI Wuppertal Abbott] Didaktik der Informatik an der Bergischen Universität Wuppertal: *Abbott*, <http://ddi.uni-wuppertal.de/material/materialsammlung/oberstufe/allgemein/abbott.html>, aufgerufen am 27.10.2016. Direktlink: http://ddi.uni-wuppertal.de/material/materialsammlung/oberstufe/allgemein/abbott/info_01_verfahren_abbott.pdf, aufgerufen am 27.10.2016.

[DDI Wuppertal Meiern] Didaktik der Informatik an der Bergischen Universität Wuppertal: *Meiern*, <http://ddi.uni-wuppertal.de/material/materialsammlung/oberstufe/oom/meiern.html>, aufgerufen am 26.11.2016. Direktlink: http://ddi.uni-wuppertal.de/material/materialsammlung/oberstufe/oom/meiern/ab_01_beschreibung_meiern.pdf, aufgerufen am 26.11.2016.

[DDI Wuppertal Objektspiel] Didaktik der Informatik an der Bergischen Universität Wuppertal: *Objektspiel*, <http://ddi.uni-wuppertal.de/material/materialsammlung/oberstufe/allgemein/objektspiel.html>, aufgerufen am 04.11.2016. Direktlink: http://ddi.uni-wuppertal.de/material/materialsammlung/oberstufe/allgemein/objektspiel/info_01_anleitung_objektspiel.pdf, aufgerufen am 04.11.2016; Spielmaterialien für das Objektspiel: *Vorderseite*. Direktlink: http://ddi.uni-wuppertal.de/material/materialsammlung/oberstufe/allgemein/objektspiel/objektspielkarte_vorderseite.pdf, aufgerufen am 04.11.2016. *Rückseite*. Direktlink: http://ddi.uni-wuppertal.de/material/materialsammlung/oberstufe/allgemein/objektspiel/objektspielkarte_rueckseite.pdf, aufgerufen am 04.11.2016. *Beziehungspfeil*. Direktlink: http://ddi.uni-wuppertal.de/material/materialsammlung/oberstufe/allgemein/objektspiel/objektspiel_beziehungspfeil.pdf, aufgerufen am 04.11.2016.

[DDI Wuppertal OOM] Didaktik der Informatik an der Bergischen Universität Wuppertal: *Material für die objektorientierte Modellierung in der Einführungsphase*, <http://ddi.uni-wuppertal.de/material/materialsammlung/oberstufe/oom.html>, aufgerufen am 27.09.2016.

[DDI Wuppertal Sequenzdiagramm] Didaktik der Informatik an der Bergischen Universität Wuppertal: *Sequenzdiagramme*, <http://ddi.uni-wuppertal.de/material/materialsammlung/oberstufe/allgemein/sequenz.html>, aufgerufen am 04.11.2016. Direktlink: http://ddi.uni-wuppertal.de/material/materialsammlung/oberstufe/allgemein/sequenz/info_01_sequenzdiagramm.pdf, aufgerufen am 04.11.2016.

[DDI Wuppertal Übersicht] Didaktik der Informatik an der Bergischen Universität Wuppertal, *Aktuelles*, <http://ddi.uni-wuppertal.de/>, aufgerufen am 27.09.2016.

[Diethelm] Ira DIETHELM: *Strictly models and objects first — Unterrichtskonzept für objektorientierte Modellierung*, <http://cs.emis.de/LNI/Proceedings/Proceedings112/gi-proc-112-004.pdf>, aufgerufen am 14.11.2016.

[Diethelm Geiger et.al. 2006] Ira DIETHELM, Leif GEIGER, Carsten SCHULTE (2006): *Einführung in die Objektorientierung im Informatik-Anfangsunterricht*,

<http://docplayer.org/7533868-Einfuehrung-in-die-objektorientierung-im-informatik-anfangsunterricht.html>, aufgerufen am 16.11.2016.

[GI Curriculum 2016] Gesellschaft für Informatik (Dorothee MÜLLER / Johannes PIEPER et. al.) (2016): *Informatik: Schulinterner Lehrplan zum Kernlehrplan für die gymnasiale Oberstufe an der Mustergesamtschule in Musterstadt*, <http://ddi.uni-wuppertal.de/material/materialsammlung/klp.html>, aufgerufen am 25.07.2016. Direktlink: http://ddi.uni-wuppertal.de/material/materialsammlung/oberstufe/KERNLEHRPLAN/SILP_GoSt_Informatik_Muster.pdf, aufgerufen am 06.11.2016.

[GI Ziele u. Aufgaben] Gesellschaft für Informatik *Was ist die GI?*, <https://www.gi.de/wir-ueber-uns/ziele-und-aufgaben.html>, aufgerufen am 14.01.2017.

[GitHub markuskeller1960] Markus KELLAR – GitHub-Profil: <https://gist.github.com/markuskeller1960>, `Led.java`, <https://gist.github.com/markuskeller1960/76239e945d5fc7111ec>, aufgerufen am 24.09.2016; `ButtonLedController.java`, <https://gist.github.com/markuskeller1960/86efe131cb8a19f28dc8>, aufgerufen am 24.09.2016; `ButtonLedController.java`, <https://gist.github.com/markuskeller1960/08215c184a9ba916f6b9>, aufgerufen am 24.09.2016; `BlinkingLED.java`, <https://gist.github.com/markuskeller1960/ad03baacc530da1768ed>, aufgerufen am 24.09.2016.

[GitHub oksbwn] Bikash Narayan PANDA – GitHub-Profil: <https://github.com/oksbwn>, `MCP3208_raspberryPi.java`, https://github.com/oksbwn/MCP3208_Raspberry-Pi/blob/master/MCP3208_raspberryPi.java, aufgerufen am 21.09.2016.

[GitHub OlivierLD] Olivier LEDIOURIS – GitHub-Profil: <https://github.com/OlivierLD/>, `Real4PWMLed.java`, <https://github.com/OlivierLD/raspberry-pi4j-samples/blob/master/RasPISamples/src/raspisamples/Real4PWMLed.java>, aufgerufen am 24.09.2016.

[Knight of Pi] *Reading an analog potentiometer with the A/D converter MCP3008 and SPI*, <http://www.knight-of-pi.org/reading-an-analog-potentiometer-with-the-ad-converter-mcp3008-and-spi/>, aufgerufen am 27.09.2016.

[LeDiouris] Olivier LEDIOURIS: *Read analog data, in Java Raspberry PI*, <http://www.lediouris.net/RaspberryPI/ADC/readme.html>, aufgerufen am 27.09.2016.

[LEGO] LEGO SHOP: *VOLVO L350F Radlader*, <https://shop.lego.com/de-DE/VOLVO-L350F-Radlader-42030>, aufgerufen am 10.11.2016. Direktlink: [https://sh-s7-live-s.legocdn.com/is/image/LEGO/42030?\\$PDPDefault\\$](https://sh-s7-live-s.legocdn.com/is/image/LEGO/42030?$PDPDefault$), aufgerufen am 10.11.2016.

[Pi4J Pins] The Pi4J Project: *Pin Numbering – Raspberry Pi 3 Model B*, <http://pi4j.com/pins/model-3b-rev1.html>. Direktlink: <http://pi4j.com/images/j8header-3b.png>, aufgerufen am 01.09.2016.

[Pi4J Team List] The Pi4J Project: *The Team*, <http://pi4j.com/pins/model-3b-rev1.html>, aufgerufen am 07.11.2016.

[Pieper Raspi Skriptum] Johannes PIEPER (2016): *Skriptum zum Raspberry Pi*, Stand: 04. Oktober 2016, http://info.johpie.de/stufe_09/raspi_skriptum.pdf, aufgerufen am 07.11.2016.

[Raspberry Pi Blog] RASPBERRY PI FOUNDATION, Eben UPTON (2016): *Ten millionth Raspberry Pi, and a new kit*, <https://www.raspberrypi.org/blog/ten-millionth-raspberry-pi-new-kit/>, aufgerufen am 16.01.2017.

[Raspberry Pi Trademark] RASPBERRY PI FOUNDATION: *Trademark rules and brand guidelines*, <https://www.raspberrypi.org/trademark-rules/>, aufgerufen am 10.12.2016.

[RS Components] RS Components: *Evaluierungsplatine Computer-Board Raspberry Pi 3 Modell B, ARM Cortex-A53, BCM2837*, <http://de.rs-online.com/web/p/entwicklungskits-prozessor-mikrocontroller/8968660/>, aufgerufen am 10.01.2017.

[RS Components Bild] RS Components: *Evaluierungsplatine Computer-Board Raspberry Pi 3 Modell B, ARM Cortex-A53, BCM2837*, <http://de.rs-online.com/web/p/entwicklungskits-prozessor-mikrocontroller/8968660/>, aufgerufen am 10.01.2017. Direktlink: <http://de.rs-online.com/largeimages/F8968660-01.jpg>, aufgerufen am 10.01.2017.

[Schriek MG Werl SuM 2003] Bernard SCHRIEK (2013): *Informatiklehrbuch für die Schule (SII) – OOP mit SuM, BlueJ und Java und SuM-Bibliotheken für BlueJ*, <http://www.mg-werl.de/sum/>, aufgerufen am 06.12.2016.

[Schulentwicklung NRW 2014] QUALITÄTS- UND UNTERSTÜTZUNGSAGENTUR – LANDESINSTITUT FÜR SCHULE (Hrsg.) (2014): *Aufgaben und Ziele des Faches*, <http://www.schulentwicklung.nrw.de/lehrplaene/lehrplannavigator-s-ii/gymnasiale-oberstufe/informatik/informatik-klp/aufgaben-ziele/aufgaben-und-ziele-des-faches.html>, aufgerufen am 22.11.2016.

[SchulG NRW 2016] SCHULGESETZ FÜR DAS LAND NORDRHEIN-WESTFALEN (Schulgesetz NRW – SchulG). Vom 15. Februar 2005 (GV. NRW. S. 102) zuletzt geändert durch Gesetz vom 14. Juni 2016 (GV. NRW. S. 442). <https://www.schulministerium.nrw.de/docs/Recht/Schulrecht/Schulgesetz/Schulgesetz.pdf>, aufgerufen am 20.12.2016.

[Schulint. Lehrplan] Joseph-König-Gymnasium in Haltern am See (2014): *Informatik: Schulinterner Lehrplan zum Kernlehrplan für die gymnasiale Oberstufe an dem Joseph-König-Gymnasium in Haltern am See*, <http://www.joseph-koenig-gymnasium.de/wp-content/uploads/Informatik-Schulinternes-Curriculum-Sek-II-2014-11-12.pdf>, aufgerufen am 20.09.2016.

[Sorensen 2016] Neal SORENSEN (2016): *Pi4J – ADC MCP3008 – SPI – Sensor Reader Example*, <https://nealvs.wordpress.com/2016/02/19/pi4j-adc-mcp3008-spi-sensor-reader-example/>, aufgerufen am 27.09.2016.

[Spolwig 2006] Siegfried SPOLWIG (2006): “*Von Stiften und Mäusen*” oder *Was heißt objektorientierte Modellierung?*, <http://www.oszhandel.de/gymnasium/faecher/informatik/didaktik/sum/sum-kritik.htm>, aufgerufen am 07.09.2016.

Anhang A

Unterrichtsmaterialien

Die Unterrichtsmaterialien umfassen:

- Beobachtungsbogen für den Unterricht (S. 103f)
- Arbeitsblätter (Anhang A.1, S. 105)
- Ausgewählte Schülerlösungen (Anhang A.2, S. 131)

Beobachtungsbogen

| | |
|--|---------|
| Datum: Thema: | Stunde: |
| Einstieg in die Stunde <i>Einstieg und Reaktion der Schülerinnen und Schüler</i> | |
| Stundengliederung | |

Erarbeitung

Material, Einleitungen in Arbeitsphasen, Sozialform, Reaktionen der Schülerinnen und Schüler, Ergebniskontrolle und Sicherung der Ergebnisse

Impulse

1)

2)

3)

4)

5)

Besonderheiten und Notizen

A.1 Arbeitsblätter

Die Arbeitsblätter umfassen:

- **Projekt 1:** Theateraufführung¹
 - *Das Verfahren nach Abbott; Objektkarten erstellen*
Einführungstext in das Verfahren von ABBOTT und wie man Objektkarten erstellt [DDI Wuppertal Abbott].
 - *Die Lichtanlage für eine Theateraufführung*
Einführungstext in die Problemsituation »Theateraufführung« für die OOM.
 - *Karten des Objektspiels (Vorder- und Rückseite), Beziehungspfeile*
Karten, die die Schülerinnen und Schüler beim Spielen des Objektsspiels vor sich tragen. Die Karten sind zu laminieren – dann können mit Fölienstift Objektnamen und Methoden (Vorderseite), sowie Attribute und Attributwerte (auf die Rückseite) eingetragen werden. Das Zeigen der Attributseite auf den eigenen Körper soll die private Sicht eines Objekts verdeutlichen (nicht jedes Attribut ist von außen einsehbar). Die Pfeile symbolisieren gerichtete Beziehungen (s. Erläuterungen auf dem Informationsblatt) [DDI Wuppertal Objektspiel].
 - *Objektspiel*
Anleitung zum Objektspiel [DDI Wuppertal Objektspiel].
 - *Sequenzdiagramm*
Informationsblatt zum Sequenzdiagramm [DDI Wuppertal Sequenzdiagramm].
 - *Anschluss von Objekten an den Raspberry Pi*
Informationsblatt zum Anschluss der Bauteile der rpCollection an den Raspberry Pi.
 - *Das Würfelspiel Meiern*
Einführungstext in die Problemsituation »Meiern« für die OOM [DDI Wuppertal Meiern].

¹ Unter [DDI Wuppertal Übersicht] findet sich die Übersichtsseite zu allgemeinen Informatik-Didaktik-Materialien der Universität Wuppertal, von der viele der hier aufgeführten Materialien stammen. Die Übersichtsseite für Materialien zur Objektorientierung ist unter [DDI Wuppertal OOM] zu finden.

- *Schalten der Hintergrundbeleuchtung*
Informationsblatt zu möglichen Varianten die Hintergrundbeleuchtung zu schalten.
- *Die Lichtanlage für eine Theateraufführung – Teil 2*
Teil 2 des Textes zur Theateraufführung (mit anderen Bauteilen).

- **Projekt 2:** Hafenkran

- *Kranwagen-Projekt*
Einführungstext in die Problemsituation »Hafenkran« für die OOM.

- **Überarbeitungen:** Theateraufführung

- *Die Lichtanlage für eine Theateraufführung*
Einführungstext in die Problemsituation »Theateraufführung« für die OOM.

Das Verfahren nach Abbott; Objektkarten erstellen

Auf RUSSELL J. ABBOTT geht ein Verfahren zurück, das für die objektorientierte Analyse (OOA) bzw. objektorientierte Modellierung (OOM) hilfreich sein kann. Es folgt eine Zusammenfassung der drei erforderlichen Schritte¹.

Um aus einer umgangssprachlich formulierten Problembeschreibung die Objekte mit den zugehörigen Objektkarten (s. u.) zu erarbeiten, geht man wie folgt vor:

1. Substantive (Hauptwörter) und Eigennamen herausfiltern

Die Hauptwörter sind mögliche *Objekte*. Meist nicht beachtet werden allerdings Mengen- und Größenangaben (»Kilogramm«), Sammelnamen (»Regierung«), Materialbezeichnungen (»Plastik«) und abstrakte Begriffe (»Liebe«, »Arbeit«). Zeitwörter (Verben), die als Hauptwörter benutzt werden (»das Betrachten eines Bildes«) werden behandelt wie die zugehörigen Zeitwörter. Gattungsnamen wie z. B. »Kraftfahrzeug«, »Säugetier« und »Einwohner« sind ebenfalls meist keine Objekte.

2. Verben (Zeitwörter) herausfiltern

Sie bezeichnen häufig die Aktionen, welche von Objekten ausgeführt werden können (die sogenannten *Methoden* der Objekte). Es ist festzustellen, welchem Objekt die Methode zugeordnet werden kann.

3. Adjektive (Eigenschaftswörter) herausfiltern

Sie bezeichnen häufig die »Ausprägungen« (*Attributwerte*), welche bestimmte Eigenschaften (die *Attributes*) von Objekten annehmen können. Beispielsweise wäre »ledig« ein Attributwert zum Attribut »Familienstand« oder »1216« der Attributwert des Attributs »Seitenzahl« des aktuellen Dudens. Auch hier ist wieder festzustellen, welchem Objekt der Attributwert zugeordnet und wie das zugehörige Attribut bezeichnet werden kann.

Die grafische Darstellung von Objekten erfolgt durch *Objektkarten*. Eine Objektkarte wird als Rechteck mit abgerundeten Ecken gezeichnet. Die erste Zeile beinhaltet dabei den eindeutigen Namen des Objekts. Nach einer horizontalen Trennlinie folgen zeilenweise die Attribute mit ihren jeweiligen Attributwerten. Die Methoden der Objekte sind (sofern vorhanden) von den Attributen und Attributwerten wiederum durch eine horizontale Linie getrennt.

Per Konvention beginnen **Bezeichner** für Objekte, Attribute und Methoden mit einem Kleinbuchstaben. Attributwerte werden häufig als sogenannte Zeichenketten dargestellt, die in »...« eingeschlossen sind. Es ist üblich, bei zusammengesetzten Bezeichnern neu einsetzende Worte durch Großbuchstaben hervorzuheben (siehe Beispiel).

Umlaute, Sonderzeichen und Leerzeichen sind in Bezeichnern grundsätzlich verboten. Neben den im Englischen verwendeten Groß- und Kleinbuchstaben und den Ziffern 0 bis 9 ist allenfalls noch der Unterstrich (»_«) erlaubt.

gustavsRadioWecker

standort = "Gustavs Zimmer"

weckzeit = "06:30"

weckzeitAktiv = "AN"

einschalten()

ausschalten()

alarmAusloesen()

¹Urheberin der Zusammenfassung unbekannt, hier in leicht veränderter Form wiedergegeben



Die Lichtanlage für eine Theateraufführung

Für die Schultheater-Aufführung des Stücks „Mac Bath“ von Shakespeare soll mit Hilfe des Raspberry Pis die Lichtanlage gesteuert werden. Alle Scheinwerfer sind über der Bühne an einem Gerüst angebracht und können ihre Position nicht verändern. Martin ist für das Licht verantwortlich.

Hier ist ein Ausschnitt seiner Arbeit:

Martin schaltet zunächst den grünen, blauen und roten Scheinwerfer an. Nachdem er etwa 20 Sekunden gewartet hat, schaltet er auch den weißen Scheinwerfer an und schaltet den grünen aus. Nach weiteren 10 Sekunden lässt er die gelbe Lampe kurz blinken und schaltet dann das gesamte Licht aus. Für die nächste Szene wird die spezielle Hintergrundbeleuchtung aktiviert. Dazu wird ein Helligkeitssenor befragt und mit der Antwort die Hintergrundbeleuchtung geschaltet.

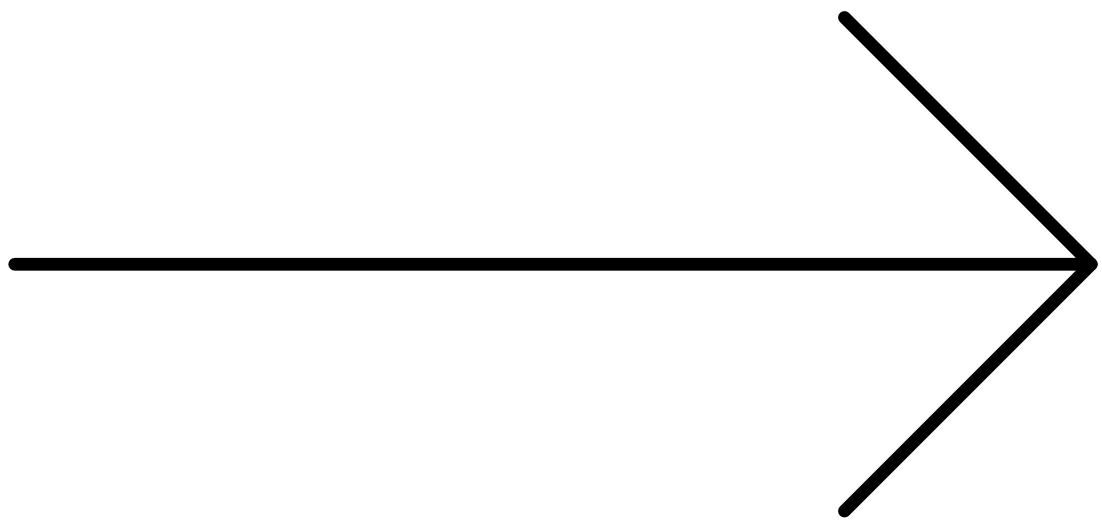
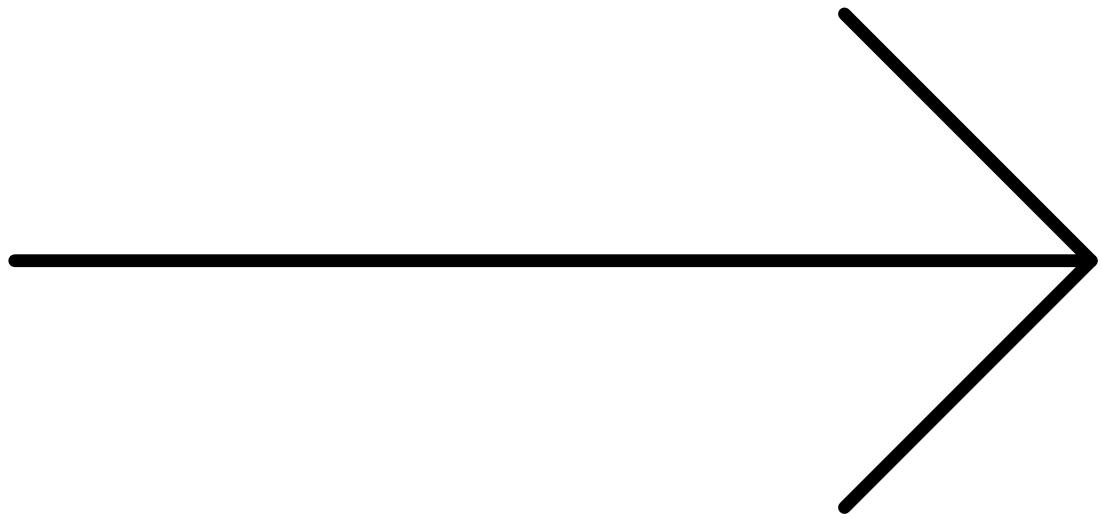
Dann arbeitet Martin mit dem RGB-Scheinwerfer^a: Er stellt eine Farbe von 50% für grün, 20% für rot und 25% für blau ein. Nach 10 Sekunden stellt er eine neue Farbe ein: 10% für grün, 70% für rot und 65% für blau. Dann wartet er nochmal 10 Sekunden und schaltet das Licht ab.

Um einen Farbwechsel zu erzielen, lässt Martin dann die Bühne mehrere Male für jeweils fünf Sekunden in weißes, rotes und blaues Licht hüllen. Die Gesamtdauer des Farbwechsels dauert ca. 45 Sekunden.

^aDer RGB-Scheinwerfer kann – anders als die anderen – eine beliebige Farbe annehmen. Dazu stellt man den Anteil an Rot, Grün und Blauwert ein

Aufgaben:

1. Entwerfen Sie zu der gegebenen Problembeschreibung mit Hilfe des Verfahrens von ABBOTT ein objektorientiertes Modell.
2. Notieren Sie die Objekte als Objektkarten.



Objektspiel

Das Objektspiel kann für verschiedene Dinge im Bereich der objektorientierten Modellierung eingesetzt werden. So können mit ihm z. B. die Grundlagen für ein Sequenzdiagramm gelegt werden oder eine Modellierung überprüft werden.

Beim Spiel übernehmen Personen die Rolle von Objekten aus der Modellierung. Im Verlauf des Spiels wird eine Situation aus der Modellierung Schritt für Schritt durchgegangen, wobei die Objekte entsprechend agieren müssen. Zur Verdeutlichung erhalten die Objekte eine beschriftete Objektkarte. Auf der Vorderseite sind der Bezeichner des Objekts und die Bezeichner für die Methoden notiert. Auf der Rückseite ist Platz für die Attributbezeichner mit ihren Attributwerten.

Regeln

- Je nach Vereinbarung reagieren die Objekte selbstständig oder ein »Erzähler« liest schrittweise die Abfolge vor.
- Objekte können nur verbal mit Hilfe ihrer Methoden miteinander kommunizieren. Dabei ist z. B. folgende Form zu wählen: »Ich ... rufe bei ... die Methode ... auf.«
- Änderungen an Attributwerten werden so notiert, dass der vorherige Wert durchgestrichen und der neue Wert dahinter geschrieben wird.
- Es kann nur auf Grundlage der Attributwerte gehandelt und geantwortet werden.
- Beziehungen zwischen Objekten werden mit Hilfe von Bändern verdeutlicht, die an den Objektkarten angeklebt werden. Bei gerichteten Beziehungen wird an das Band noch zusätzlich ein Pfeil gehängt.
- Objekte können nur mit Objekten kommunizieren, zu denen sie eine direkte Beziehung haben oder sie müssen über andere Objekte kommunizieren.

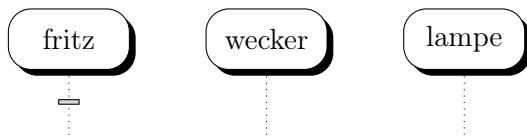
Alle nicht am Objektspiel beteiligten Personen haben die Aufgabe, den Ablauf genau zu protokollieren. Dazu füllen sie eine Tabelle mit folgenden Spalten aus:

| Aktion von | an wen | Aktion | Parameter | Rückgabe | Bemerkung |
|------------|--------|--------|-----------|----------|-----------|
| | | | | | |



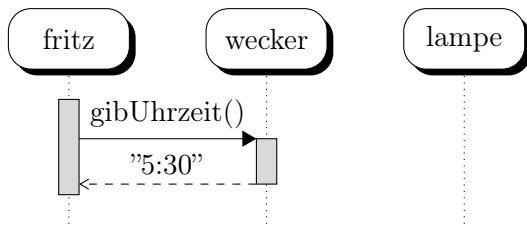
Sequenzdiagramm

Zur Modellierung der zeitlichen Abfolge von Interaktionen zwischen Objekten werden sogenannte Sequenzdiagramme verwendet. Die an der Interaktion beteiligten Objekte werden dabei wie bisher als abgerundete Rechtecke in der obersten Zeile dargestellt. Allerdings werden weder die Attribute (und ihre Werte) noch die Methoden in das abgerundete Rechteck eingetragen – ausschließlich der Objektbezeichner. Zu jedem Objekt gehört eine *Lebenslinie*, die vertikal nach unten verläuft und gestrichelt dargestellt wird.



Die Interaktion der Objekte wird nun durch waagerechte Pfeile zwischen den Lebenslinien der Objekte dargestellt. Dabei wird jede *Anfrage* und jeder *Auftrag* über dem Pfeil notiert.

Da es sich bei Anfragen und Aufträgen um den Aufruf einer *Methode* des Zielobjektes handelt, signalisiert ein Balken über der Lebenslinie, wie lange die entsprechende Methode abläuft. Ein gestrichelter Pfeil zurück zum Startobjekt gibt bei einer Anfrage einen Wert zurück. Dieser wird ebenfalls oberhalb des Pfeils notiert.

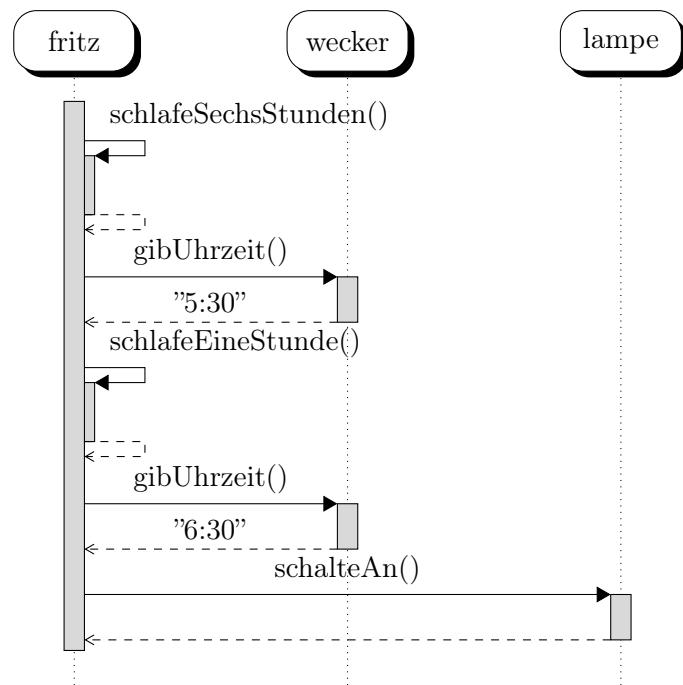


Gelegentlich gibt es auch die Möglichkeit, dass ein Objekt eine Anfrage oder Anweisung an sich selbst stellt. Dies wird durch einen Pfeil zurück auf das Objekt und einen zusätzlichen Balken für die entsprechend aktive Methode dargestellt.

Beispiel: Schlaf des Fritz

Fritz schläft sechs Stunden. Fritz fragt den Wecker, wie spät es ist. Der Wecker gibt das Ergebnis „5:30“ zurück. Fritz schläft eine weitere Stunde. Fritz fragt den Wecker, wie spät es ist. Der Wecker gibt das Ergebnis „6:30“ zurück. Fritz schaltet die Lampe an.





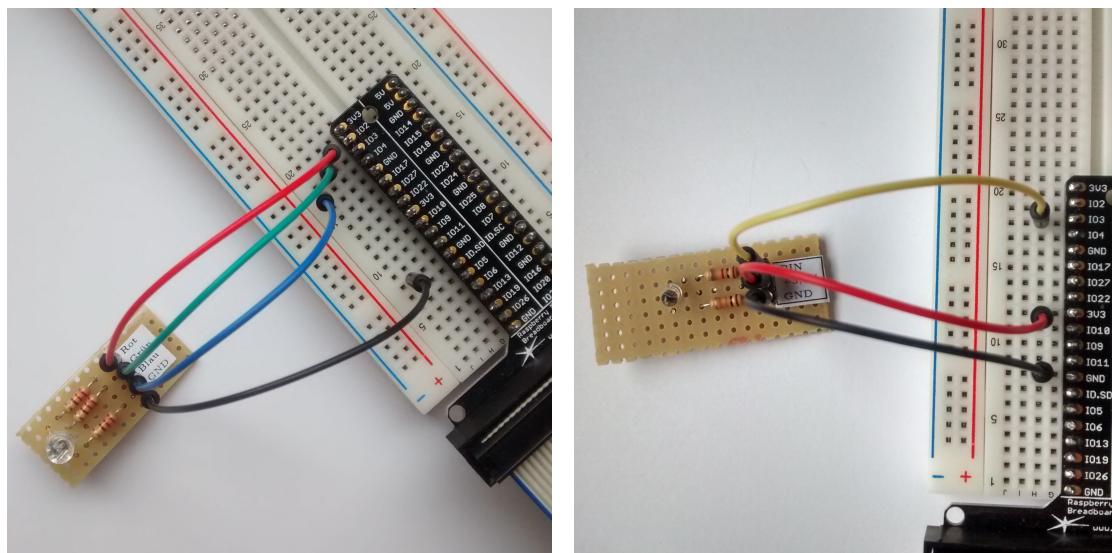
Anschluss von Objekten an den Raspberry Pi

Am Raspberry Pi können verschiedene Objekte angeschlossen werden. Dazu verfügt jedes Objekt über Anschlüsse, die auf dem Steckbrett mit dem GPIO des Raspberry Pi verbunden werden müssen. Dabei ist es wichtig, die Kabel so zu stecken, wie es die Anschlüsse auf dem Objekt vorschreiben.

Bei Anschlüssen mit der Aufschrift »3,3V« oder »GND« kann ein entsprechender ausgewählt werden. Dabei können an diese beiden Pinne auch mehrere Objekte angeschlossen werden.

Bei den anderen Aufdrucken, in den meisten Fällen »PIN«, kann am GPIO ein beliebiger »IO«-Pin ausgewählt werden. Bei diesem darf aber jeweils nur ein Objekt angeschlossen werden. Dieser Pin muss dann später im Programm angegeben werden, um z. B. das Objekt zu steuern oder den Zustand auszulesen.

In Abbildung 1 werden zwei Beispiele gegeben, wie die Objekte anzuschließen sind.



(a) Anschluss des RGB-Scheinwerfers (Anschluss an »IO02«, »IO03«, »IO04« und »GND«). (b) Anschluss eines Helligkeitssensors (Anschluss an »IO3«, »3V3« und »GND«).

Abbildung 1: Angeschlossene Objekte am Steckbrett.

Abfolge zum Anschließen und Ansteuern eines Objektes

Hinweis: Änderungen an der Hardware sollten nur dann erfolgen, wenn der Raspberry Pi ausgeschaltet ist.

1. Die Objekte mit Jumper-Kabeln mit dem GPIO des Raspberry Pi verbinden, die blaue Seite am Flachbandkabel muss dabei zum Displayanschluss zeigen. Anschließend die Objekte verkabeln.

2. Am eingeschalteten Raspberry Pi die GroovyConsole starten: Dazu in einem geöffneten Terminalfenster folgendes eingeben:

```
1 cd rpCollection  
2 git pull  
3 ./start.sh
```

Der erste Befehl wechselt in das passende Verzeichnis und der zweite aktualisiert die Daten, wenn eine Verbindung mit dem Internet besteht. Die Aktualisierung ist nur bei Bedarf notwendig.

3. In der GroovyConsole können die Objekte direkt angesprochen werden. Im Quelltext 1 ist dazu ein Beispiel zum Schalten eines Scheinwerfers (einer LED) am Pin 11 angeben.

Listing 1: Steuerung eines Scheinwerfers.

```
1 gelbeLampe = new Scheinwerfer(11);  
2 gelbeLampe.anSchalten();  
3 gelbeLampe.setzeStandort("Martins_Buehne");  
4 sleep(400); //Warten von 400 Millisekunden = 0,4 Sekunden  
5  
6 Helfer.herunterfahren(); //Pinne wieder frei geben (damit das  
Skript neu gestartet werden kann)
```

4. Die weiteren Methoden der Objekte stimmen mit denen aus dem Unterricht überein.

Aufgabe 1

Modelliere den Ablauf aus der Problembeschreibung aus der Sicht von Martin.

Das Würfelspiel Meiern

Lesen Sie sich die nachfolgende Situationsbeschreibung durch und bearbeiten Sie anschließend die darunter stehenden Aufgaben.

Eine Szene aus einer *Meiern*-Runde

Peter, Doris und Siglinde spielen das Spiel *Meiern*¹. Gespielt wird immer reihum: Peters Nachfolger ist Doris, Peters Vorgänger Siglinde. Entsprechend ist Doris Nachfolger Siglinde und ihr Vorgänger Peter. Siglindes Nachfolger ist schließlich Peter, ihr Vorgänger Doris. (*Wir befinden uns mitten im Spiel.*)

Peter hatte sich die Zahl gemerkt, die seine Vorgängerin Siglinde ihm gesagt hatte. Peters gemerkte Zahl der Vorgängerin ist 54.

Peter führt einen Zug aus, indem er den Würfelbecher schüttelt. Die zwei Würfel, welche sich im Würfelbecher befinden, nehmen dabei eine neue Augenzahl an, jeweils eine 3.

Durch das Anheben des Würfelbechers kann Peter nun die beiden Würfel sehen. Er liest die Augenzahl vom ersten und zweiten Würfel ab: Vom ersten Würfel eine 3 und vom zweiten Würfel ebenfalls eine 3. Er merkt sich seine eigens gewürfelte Zahl 33 (*Pasch 3*)².

Peter lässt Doris den Würfelbecher nehmen. Sie fragt ihn, was er gewürfelt habe. Peter antwortet ihr darauf „*Pasch 3*“. Sie merkt sich die Zahl und überlegt...

Aufgaben

1. Entwerfen Sie zu der gegebenen Problembeschreibung mit Hilfe des Verfahrens von Abbott ein objektorientiertes Modell, indem Sie die relevanten Objekte mit ihren Attributen und Methoden identifizieren. Notieren Sie die Objekte als Objektkarten.
2. Erstellen Sie ein Objektdiagramm, in dem alle Beziehungen zwischen den Objekten so dargestellt sind, wie sie am Ende der Situationsbeschreibung vorliegen.
3. Erstellen Sie ein Sequenzdiagramm, welches die Interaktion zwischen den identifizierten Objekten gemäß der Problembeschreibung beschreibt.

¹Kurze Spielbeschreibung: Ein Würfelspiel mit 2 Würfeln und einem Würfelbecher für mehrere Spieler. Einer fängt an zu würfeln und schaut so unter den Würfelbecher, dass niemand sonst die Zahlen erkennen kann, die oben liegen. Dieser Spieler sagt dem nächsten nun eine Zahl. Ob er die gewürfelte Zahl nimmt oder lügt bleibt dem Spieler überlassen. Er muss aber eine Zahl nennen, die höher ist, als die Zahl, die er selbst gesagt bekommen hat, bevor er den Würfelbecher überreicht bekommen hat. Die nachfolgende Person muss nun entscheiden, ob sie die Aussage für wahr hält, oder ob der Vorgänger gelogen hat. Dann deckt man die Würfel auf. Für nähere Erläuterungen recherchieren Sie bitte im Internet oder fragen Bekannte.

²33 steht für einen Pasch 3, der höher ist als 54.



Schalten der Hintergrundbeleuchtung

Wir haben im Unterricht gesehen, dass es zwei verschiedene Varianten gibt, folgende Stelle aus der Geschichte zu modellieren:

»Für die nächste Szene wird die spezielle Hintergrundbeleuchtung aktiviert. Dazu wird ein Helligkeitssensor befragt und mit der Antwort die Hintergrundbeleuchtung geschaltet.«

Variante A: Martin kennt Hintergrundbeleuchtung und Helligkeitssensor

In dieser Variante kennt Martin die Hintergrundbeleuchtung und den Helligkeitssensor. Das heißt, Martin hat zwei Attribute, die auf das Objekt der Hintergrundbeleuchtung und das Objekt des Helligkeitssensors verweisen (dies hatten wir mit zwei Fäden im Objektspiel von Martin zur Hintergrundbeleuchtung und zum Helligkeitssensor dargestellt, s. Objektdiagramm in Abbildung 1).

Das Sequenzdiagramm ist in Abbildung 2 dargestellt. Man erkennt sehr schön, wie Martin selbstständig beim Helligkeitssensor nachfragt, ob es gerade hell ist oder nicht und dann mit der Antwort die Hintergrundbeleuchtung schaltet.

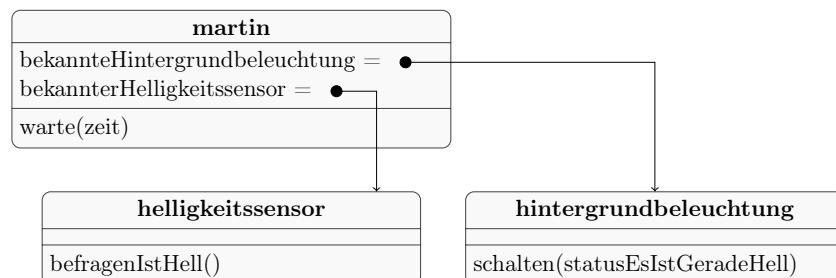


Abbildung 1: Martin kennt den Helligkeitssensor und die Hintergrundbeleuchtung.

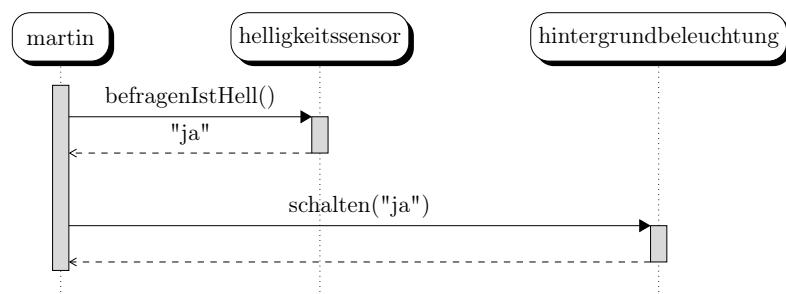


Abbildung 2: Martin fragt selbstständig beim Helligkeitssensor nach (im Unterschied zu Variante B).

Variante B: Nur die Hintergrundbeleuchtung kennt den Helligkeitssensor

In dieser Variante kennt Martin nur die Hintergrundbeleuchtung, da er diese zu einem bestimmten Zeitpunkt im Theaterstück (bedingt) einschalten muss. Den Helligkeitssensor kennt er nicht. Nur die Hintergrundbeleuchtung kennt den Helligkeitssensor (Objektdiagramm in Abbildung 3).

Das bedeutet: Martin kann bei der Hintergrundbeleuchtung die Methode zum bedingten Anschalten aufrufen und diese fragt dann *selbstständig* bei dem Helligkeitssensor nach, ob es gerade hell ist oder nicht und schaltet sich dann *selbstständig* ein oder aus (Sequenzdiagramm in Abbildung 4).

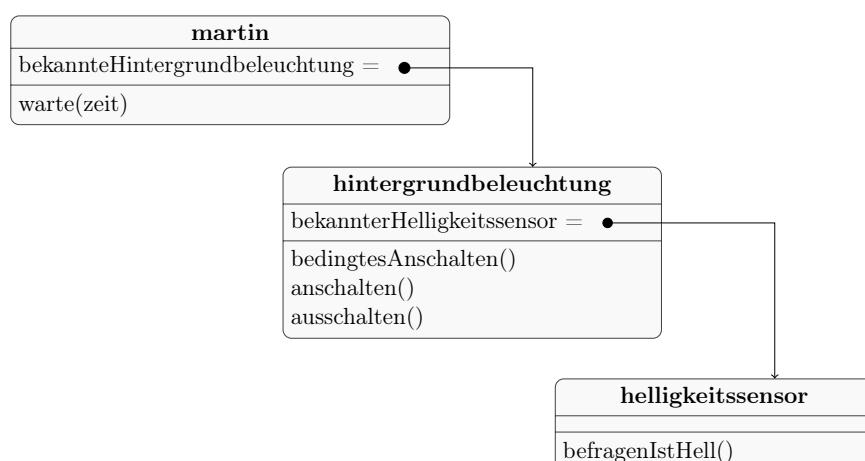


Abbildung 3: Martin kennt nur die Hintergrundbeleuchtung und diese den Helligkeitssensor.

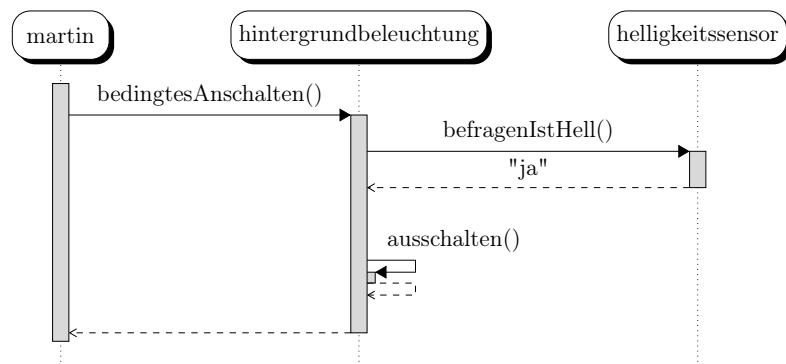


Abbildung 4: Martin ruft das bedingte Anschalten, die Hintergrundbeleuchtung befragt den Helligkeitssensor und schaltet sich dann selbst an oder aus.

Die Lichtanlage einer Theater-Aufführung – Teil 2

Hier siehst du, wie Martin weiterarbeitet:

Für die nächsten Szene trägt Martin den RGB-Scheinwerfer auf die Empore und Martin das Licht gerne flexibel regeln. Dazu stehen ihm drei Regler zur Verfügung, um die Farbe zu steuern. Er hat für jeden Farbkanal einen Regler.

Zunächst dreht er alle drei Regler voll auf. Dann gibt er den Befehl an den RGB-Scheinwerfer, die eingestellten Werte anzunehmen. Während auf der Bühne das Theaterstück seinen Lauf nimmt, verändert er die Regler. Er verstellt den grünen auf ca. 80%, den roten auf ca 50% und dreht den blauen auf 0%. Nach 10 Sekunden nimmt dann der RGB-Scheinwerfer die eingestellte Farbe an.

Dann schleicht sich Martin hinter die Bühne und baut die Hintergrundbeleuchtung ab (er überprüft zuvor noch einmal, ob die Hintergrundbeleuchtung auch wirklich ausgeschaltet ist).

(Objekt auflösen?)

Für die kommende Szene arbeitet Martin mit den Schauspielern auf der Bühne zusammen. Beim RGB-Scheinwerfer hat er mittlerweile ein leichtes rosa eingestellt. Die Schauspieler haben eine kleine Taschenlampe in der Hand und wollen den roten Scheinwerfer immer dann anschalten, wenn sie mit der Taschenlampe auf den Helligkeitssensor scheinen.

Für das Finale wartet Martin dann auf das Signal der Schauspieler, alle Lampen auf einmal anzuschalten. Dafür haben die Schauspieler an der Bühnendekoration einen geheimen Taster angebracht, der gedrückt werden muss, um das Signal zum Einschalten zu geben. Martin fragt also beim Taster nach und schaltet abhängig von der Antwort dann alle Scheinwerfer. Nach 5 Sekunden schaltet er alle Lichter aus. Dann räumt Martin alle Scheinwerfer wieder in den Medienraum. Das Theaterstück ist zu Ende.

Aufgabe 1

Führen Sie das schon bekannte Modellierungsschema von ABBOTT für diese neue Situation durch. Schon modellierte Objekte brauchen Sie natürlich nicht noch einmal zu modellieren, sondern können auf Ihre Unterlagen zurückgreifen.

Kranwagen-Projekt

Es soll ein Hafenkran modelliert werden. Der Kran soll sich frei bewegen können (360° drehen) und den Arm heben können. Auch die Seilwinde soll sich automatisch drehen.

Es gibt verschiedene Leuchten am Kran, die einzuschalten sind, wenn dieser arbeitet (Arbeitsschutz).

Damit der Kran keine Arbeiter verletzt, soll beim Drehen ein Geräusch ertönen, wie man es von großen Fahrzeugen kennt, wenn diese rückwärts fahren. Zudem sollen die Lampen blitzen.

Der Kran soll etwas anheben und muss gesteuert werden:

Aus Sicherheitsgründen soll, bevor irgendetwas anders passiert, die Kranbeleuchtung angeschaltet werden. Dann soll der Kran zweimal kurz hupen, um auch akustisch auf sich aufmerksam zu machen. Danach kann die Arbeit beginnen!

Der Kran soll einen Container anheben, steht aber noch in der falschen Position. Zunächst muss er sich für 5 Sekunden nach links drehen. Dazu muss der entsprechende Motor angeschaltet werden und dann für 5 Sekunden laufen. Danach muss der Kran den Arm etwas senken, um den Container erreichen zu können. Der entsprechende Motor muss zunächst wieder angeschaltet werden und dann für 2 Sekunden laufen. Dann kann der Motor für die Seilwinde angeschaltet werden und ...

Aufgabe 1

Entwerfen Sie zu der gegebenen Problembeschreibung mit Hilfe des Verfahrens von Abbott ein objektorientiertes Modell, indem Sie die relevanten Objekte mit ihren Attributen und Methoden identifizieren. Notieren Sie die Objekte als Objektkarten.

Aufgabe 2

Überlegen Sie sich eine Komfortable Steuerung für die Anforderungen.

Aufgabe 3

Für Fortgeschrittene: Ließe sich auch ein Not-Aus-Schalter realisieren? Wenn ja, wie?

Die Lichtanlage einer Theater-Aufführung

Für die Schultheater-Aufführung des Stückes »Mac Bath« von Shakespeare soll mit Hilfe des Raspberry Pis die Lichtanlage gesteuert werden. Die meisten Scheinwerfer sind links und rechts der Bühne an großen, senkrechten Gerüsten angebracht. Martin ist für das Licht verantwortlich.

Hier ist ein Ausschnitt seiner Arbeit:

Martin schaltet zunächst den grünen und roten Scheinwerfer an. Nachdem er etwa 20 Sekunden gewartet hat, schaltet er auch den blauen Scheinwerfer an und schaltet den grünen aus. Nach weiteren 10 Sekunden lässt er die gelbe Lampe kurz blinken und schaltet dann das gesamte Licht aus.

Für die nächste Szene wird die spezielle Hintergrundbeleuchtung aktiviert. Dazu wird ein Helligkeitssenor befragt und mit der Antwort die Hintergrundbeleuchtung geschaltet.

Für die nächste Szene sind über der Bühne zwei weiße Scheinwerfer an einer Deckenkonstruktion angebracht worden, die Tageslicht auf der Bühne erzeugen sollen. Martin kann die Scheinwerfer nicht einzeln an- und ausschalten, erzeugt aber für die Schauspieler Tageslicht auf der Bühne.

Dann arbeitet Martin mit dem RGB-Scheinwerfer^a: Er stellt eine Farbe von 50% für grün, 20% für rot und 25% für blau ein. Nach 10 Sekunden stellt er eine neue Farbe ein: 10% für grün, 70% für rot und 65% für blau. Dann wartet er nochmal 10 Sekunden und schaltet das gesamte Licht ab.

^aDer RGB-Scheinwerfer kann – anders als die anderen – eine beliebige Farbe annehmen. Dazu stellt man den Anteil an Rot, Grün und Blau ein

Aufgabe 1

Entwerfen Sie zu der gegebenen Problembeschreibung mit Hilfe des Verfahrens von AB-BOTT ein objektorientiertes Modell.

Aufgabe 2

Notieren Sie die Objekte als Objektkarten.

A.2 Ausgewählte Schülerlösungen

Die Schülerlösungen umfassen:

- **Projekt 1:** Theateraufführung
 - Kandidaten für Objekte zum Theaterprojekt (Anhang A.2.1, S. 132)
 - Objektkarten zum Theaterprojekt (Anhang A.2.2, S. 133)
 - Objektkarten des Objektspiels (Anhang A.2.3, S. 134)
 - Protokoll zum Objektspiel (Anhang A.2.4, S. 136)
 - Sequenzdiagramm zum Objektspiel (Besprechung Tafel, Anhang A.2.5, S. 137)
 - Sequenzdiagramm zum Objektspiel (vollständig, Anhang A.2.6, S. 138)
 - Quelltext Theaterprojekt (Anhang A.2.7, S. 139)
 - Objektspiel mit Objektbeziehungen (Anhang A.2.8, S. 142)
 - Übersicht zu Objektbeziehungen (Besprechung Tafel, Anhang A.2.9, S. 143)
 - Sequenzdiagramme zum Schalten der Hintergrundbeleuchtung (Anhang A.2.10, S. 144)
 - Quelltexte zum Schalten der Hintergrundbeleuchtung (Anhang A.2.11, S. 145)

A.2.1 Kandidaten für Objekte zum Theaterprojekt

| Objekte | Kandidaten für | |
|-------------------|----------------|-----------------------------|
| | Methoden | Attribute und Attributwerte |
| Martin | blinken() | grün |
| Scheinwerfer | einschalten() | blau |
| rgbScheinwerfer | ausschalten() | rot |
| Zeit * | rot() * | weiß |
| Lampe | gruen() * | speziell |
| Helligkeitssensor | blau() * | 50% |
| Bühne | befragen() | 20% |
| Szene * | warten() | 25% |
| | | 60% |
| | | 75% |
| | | 10% |
| | | 45 sec. |
| | | 5 sec. |

Tabelle A.1: Identifizierte Kandidaten für Objekte des ersten Theater-Textes. Mit * gekennzeichnete Einträge wurden in der Besprechung entfernt.

A.2.2 Objektkarten zum Theaterprojekt

Diese Objektkarten wurden im Unterricht besprochen.

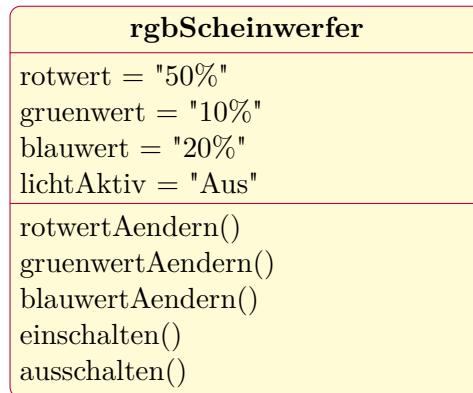


Abbildung A.1: Objektkarte zum Theaterprojekt – Lösungsvorschlag 1.

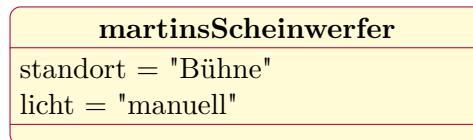


Abbildung A.2: Objektkarte zum Theaterprojekt – Lösungsvorschlag 2.

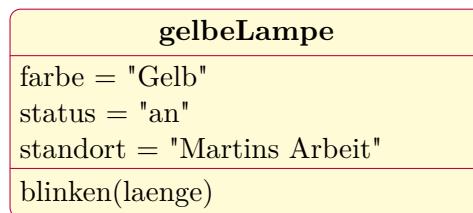


Abbildung A.3: Objektkarte zum Theaterprojekt – Lösungsvorschlag 3.

A.2.3 Objektkarten des Objektspiels

Die Objektkarten vor und nach dem Objektspiel.

| |
|--------------------|
| martin |
| Standort = "Bühne" |

| |
|--------------------|
| martin |
| Standort = "Bühne" |
| warten(zeit) |

| |
|---------------|
| buehne |
| |

Es hat sich im Spiel herausgestellt, dass das Objekt **buehne** nicht gebraucht wird.

| |
|--------------------------|
| roterScheinwerfer |
| farbe = "rot" |
| status = "aus" |
| schalten(Status) |

| |
|--------------------------|
| roterScheinwerfer |
| farbe = "rot" |
| status = "aus" |
| schalten(Status) |
| anschalten() |
| ausschalten() |

| |
|---------------------------|
| grünerScheinwerfer |
| standort = "Bühne" |
| status = "aus" |
| farbe = "grün" |
| anschalten() |
| ausschalten() |

| |
|----------------------------|
| gruenerScheinwerfer |
| standort = "Bühne" |
| status = "aus" |
| farbe = "grün" |
| anschalten() |
| ausschalten() |

| |
|--|
| gelbeLampe |
| !Farbe = GELB |
| aktiviert = FALSCH |
| farbe = "gelb" |
| schalten(Wahrheitswert anschaltStatus) |

| |
|--|
| gelbeLampe |
| !Farbe = GELB |
| aktiviert = FALSCH |
| farbe = "grün" |
| schalten(Wahrheitswert anschaltStatus) |
| anschalten(){schalten(WAHR)} |
| blinken() |

| |
|---------------------------|
| blauerScheinwerfer |
| farbe = "blau" |
| zustand = aus |
| anschalten() |
| ausschalten() |

| |
|---------------------------|
| blauerScheinwerfer |
| farbe = "blau" |
| zustand = aus |
| anschalten() |
| ausschalten() |

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|-------------------------------|--|---------------------|--|---------------------------------------|--|----------------|--|---------------|--|------------------------|--|---|-------------------------------|--------------------|---------------------|--|---------------------------------------|---------------|---------------|-----------|---------------|---------------|------------------|---|----------------------------|--|----------------|--|----------------|--|--------------|--|---------------|--|------------------------|--|---------------|--|-----------------|--|----------------|--|--------------------|--|---------------------------|--|-------------------------------|--|-----------|--|---------------|--|
| <table border="1"> <tr><td colspan="2">weisserScheinwerfer</td></tr> <tr><td>farbe = "weiß"</td><td></td></tr> <tr><td>status = null</td><td></td></tr> <tr><td>anschalten()</td><td></td></tr> <tr><td>ausschalten()</td><td></td></tr> </table> <table border="1"> <tr><td colspan="2">rgbScheinwerfer</td></tr> <tr><td>farbe = "gruen", "blau", "gelb", "rot", "weiß"</td><td></td></tr> <tr><td>lichtAktiv = "AUS"</td><td></td></tr> <tr><td>farbenmischen()(unterschiedlicherAnteil)</td><td></td></tr> <tr><td>einschalten()</td><td></td></tr> <tr><td>blinken()</td><td></td></tr> <tr><td>ausschalten()</td><td></td></tr> </table> | weisserScheinwerfer | | farbe = "weiß" | | status = null | | anschalten() | | ausschalten() | | rgbScheinwerfer | | farbe = "gruen", "blau", "gelb", "rot", "weiß" | | lichtAktiv = "AUS" | | farbenmischen()(unterschiedlicherAnteil) | | einschalten() | | blinken() | | ausschalten() | | <table border="1"> <tr><td colspan="2">weisserScheinwerfer</td></tr> <tr><td>farbe = "weiß"</td><td></td></tr> <tr><td>status = "aus"</td><td></td></tr> <tr><td>anschalten()</td><td></td></tr> <tr><td>ausschalten()</td><td></td></tr> </table> <table border="1"> <tr><td colspan="2">rgbScheinwerfer</td></tr> <tr><td>farberot = 20</td><td></td></tr> <tr><td>farbegruen = 50</td><td></td></tr> <tr><td>farbeblau = 25</td><td></td></tr> <tr><td>lichtAktiv = "AUS"</td><td></td></tr> <tr><td>mischen(rot, gruen, blau)</td><td></td></tr> <tr><td>einschalten(rot, gruen, blau)</td><td></td></tr> <tr><td>blinken()</td><td></td></tr> <tr><td>ausschalten()</td><td></td></tr> </table> | weisserScheinwerfer | | farbe = "weiß" | | status = "aus" | | anschalten() | | ausschalten() | | rgbScheinwerfer | | farberot = 20 | | farbegruen = 50 | | farbeblau = 25 | | lichtAktiv = "AUS" | | mischen(rot, gruen, blau) | | einschalten(rot, gruen, blau) | | blinken() | | ausschalten() | |
| weisserScheinwerfer | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| farbe = "weiß" | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| status = null | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| anschalten() | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ausschalten() | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| rgbScheinwerfer | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| farbe = "gruen", "blau", "gelb", "rot", "weiß" | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| lichtAktiv = "AUS" | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| farbenmischen()(unterschiedlicherAnteil) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| einschalten() | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| blinken() | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ausschalten() | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| weisserScheinwerfer | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| farbe = "weiß" | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| status = "aus" | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| anschalten() | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ausschalten() | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| rgbScheinwerfer | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| farberot = 20 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| farbegruen = 50 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| farbeblau = 25 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| lichtAktiv = "AUS" | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| mischen(rot, gruen, blau) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| einschalten(rot, gruen, blau) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| blinken() | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ausschalten() | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <table border="1"> <tr><td colspan="2">hintergrundbeleuchtung</td></tr> <tr><td>Standort = "Buehne"</td><td></td></tr> <tr><td>status = "an"</td><td></td></tr> <tr><td>status = "aus"</td><td></td></tr> <tr><td>anschalten()</td><td></td></tr> <tr><td>ausschalten()</td><td></td></tr> </table> | hintergrundbeleuchtung | | Standort = "Buehne" | | status = "an" | | status = "aus" | | anschalten() | | ausschalten() | | <table border="1"> <tr><td colspan="2">hintergrundbeleuchtung</td></tr> <tr><td>Standort = "Buehne"</td><td></td></tr> <tr><td>status = "aus"</td><td></td></tr> <tr><td>anschalten()</td><td></td></tr> <tr><td>ausschalten()</td><td></td></tr> <tr><td>schalten(status)</td><td></td></tr> </table> | hintergrundbeleuchtung | | Standort = "Buehne" | | status = "aus" | | anschalten() | | ausschalten() | | schalten(status) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| hintergrundbeleuchtung | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Standort = "Buehne" | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| status = "an" | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| status = "aus" | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| anschalten() | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ausschalten() | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| hintergrundbeleuchtung | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Standort = "Buehne" | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| status = "aus" | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| anschalten() | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ausschalten() | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| schalten(status) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <table border="1"> <tr><td colspan="2">helligkeitssensor</td></tr> <tr><td>Standort = "Bühne"</td><td></td></tr> <tr><td>hintergrundbeleuchtung = "geschaltet"</td><td></td></tr> <tr><td>einschalten()</td><td></td></tr> <tr><td>ausschalten()</td><td></td></tr> <tr><td>befragen()</td><td></td></tr> </table> | helligkeitssensor | | Standort = "Bühne" | | hintergrundbeleuchtung = "geschaltet" | | einschalten() | | ausschalten() | | befragen() | | <table border="1"> <tr><td colspan="2">helligkeitssensor</td></tr> <tr><td>Standort = "Bühne"</td><td></td></tr> <tr><td>hintergrundbeleuchtung = "geschaltet"</td><td></td></tr> <tr><td>einschalten()</td><td></td></tr> <tr><td>ausschalten()</td><td></td></tr> <tr><td>befragen()</td><td></td></tr> </table> | helligkeitssensor | | Standort = "Bühne" | | hintergrundbeleuchtung = "geschaltet" | | einschalten() | | ausschalten() | | befragen() | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| helligkeitssensor | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Standort = "Bühne" | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| hintergrundbeleuchtung = "geschaltet" | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| einschalten() | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ausschalten() | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| befragen() | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| helligkeitssensor | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Standort = "Bühne" | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| hintergrundbeleuchtung = "geschaltet" | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| einschalten() | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ausschalten() | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| befragen() | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Tabelle A.2: Objektkarten der Schülerinnen und Schüler vor (links) und nach (rechts) dem Objektspiel.

A.2.4 Protokoll zum Objektspiel

Das zum Objektspiel gehörende Protokoll.

| Aktion von | an wen | Aktion | Parameter | Rückgabe | Bemerkung |
|------------|------------------------|-------------------------------|--|----------|-----------------------------------|
| martin | guenerScheinwerfer | anschalten() | — | — | — |
| martin | blauerScheinwerfer | anschalten() | — | — | — |
| martin | roterScheinwerfer | anschalten() | — | — | — |
| martin | martin | warten(zeit) | zeit = 20 | — | — |
| martin | weisserScheinwerfer | anschalten() | — | — | — |
| martin | gruenScheinwerfer | ausschalten() | — | — | — |
| martin | martin | warten(zeit) | zeit = 10 | — | — |
| martin | gelbeLampe | blinken() | — | — | evt. Parameter: "kurz blinken" |
| martin | roterScheinwerfer | ausschalten() | — | — | — |
| martin | blauerScheinwerfer | ausschalten() | — | — | — |
| martin | weisserScheinwerfer | ausschalten() | — | — | — |
| martin | helligkeitssensor | befragen() | — | — | — |
| martin | hintergrundbeleuchtung | schalten() | "ist hell" | — | — |
| martin | rgbScheinwerfer | einschalten(rot, gruen, blau) | rot = 20%, gruen = 50%, blau = 25% | — | — |
| martin | martin | warten(zeit) | zeit = 10 | — | — |
| martin | rgbScheinwerfer | mischen(rot, gruen, blau) | rot = 70%, gruen = 10%, blau = 65% | — | — |
| martin | martin | warten(zeit) | zeit = 10 | — | — |
| martin | rgbScheinwerfer | ausschalten() | — | — | — |

Tabelle A.3: Protokoll des Objektspiels für die ersten zwei Paragraphen. Der Trennstrich in der Mitte zeigt den Paragraphenwechsel im Text an.

A.2.5 Sequenzdiagramm zum Objektspiel (Besprechung Tafel)

An der Tafel besprochener Ausschnitt des Sequenzdiagramms.

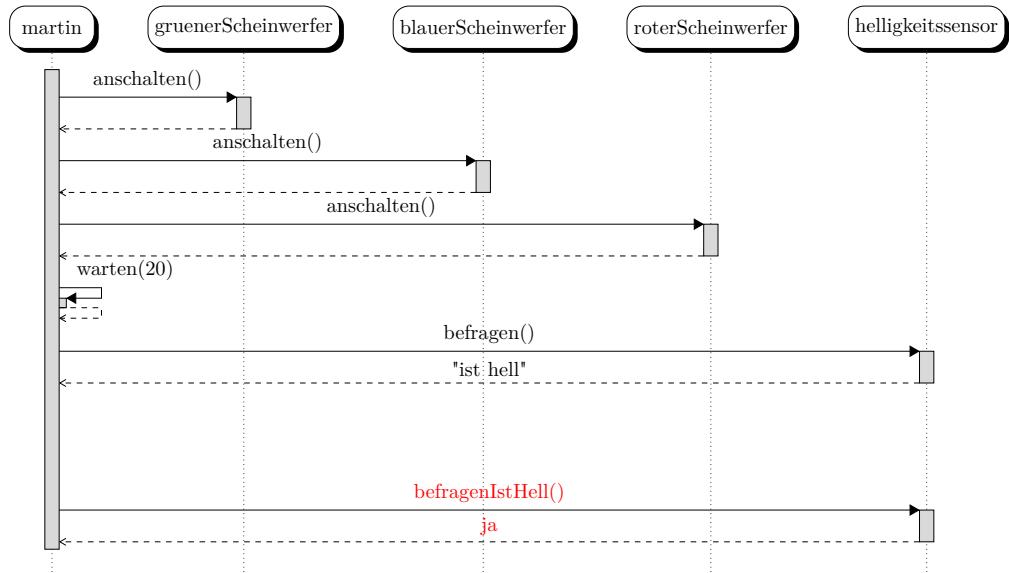


Abbildung A.4: Sequenzdiagramm zum Objektspiel (Besprechung an der Tafel). Der Ausschnitt enthält alle wichtigen Aspekte: Methodenaufruf bei anderen Objekten ohne und mit Parameter, Methodenaufruf mit und ohne Rückgabewert, Methodenaufruf beim Objekt selbst.

Der abgesetzte rote Aufruf stellt die beschriebene Überarbeitung dar (Einführung der Wahrheitswerte `ja` und `nein`, bzw. `true` und `false` in Java).

A.2.6 Sequenzdiagramm zum Objektspiel

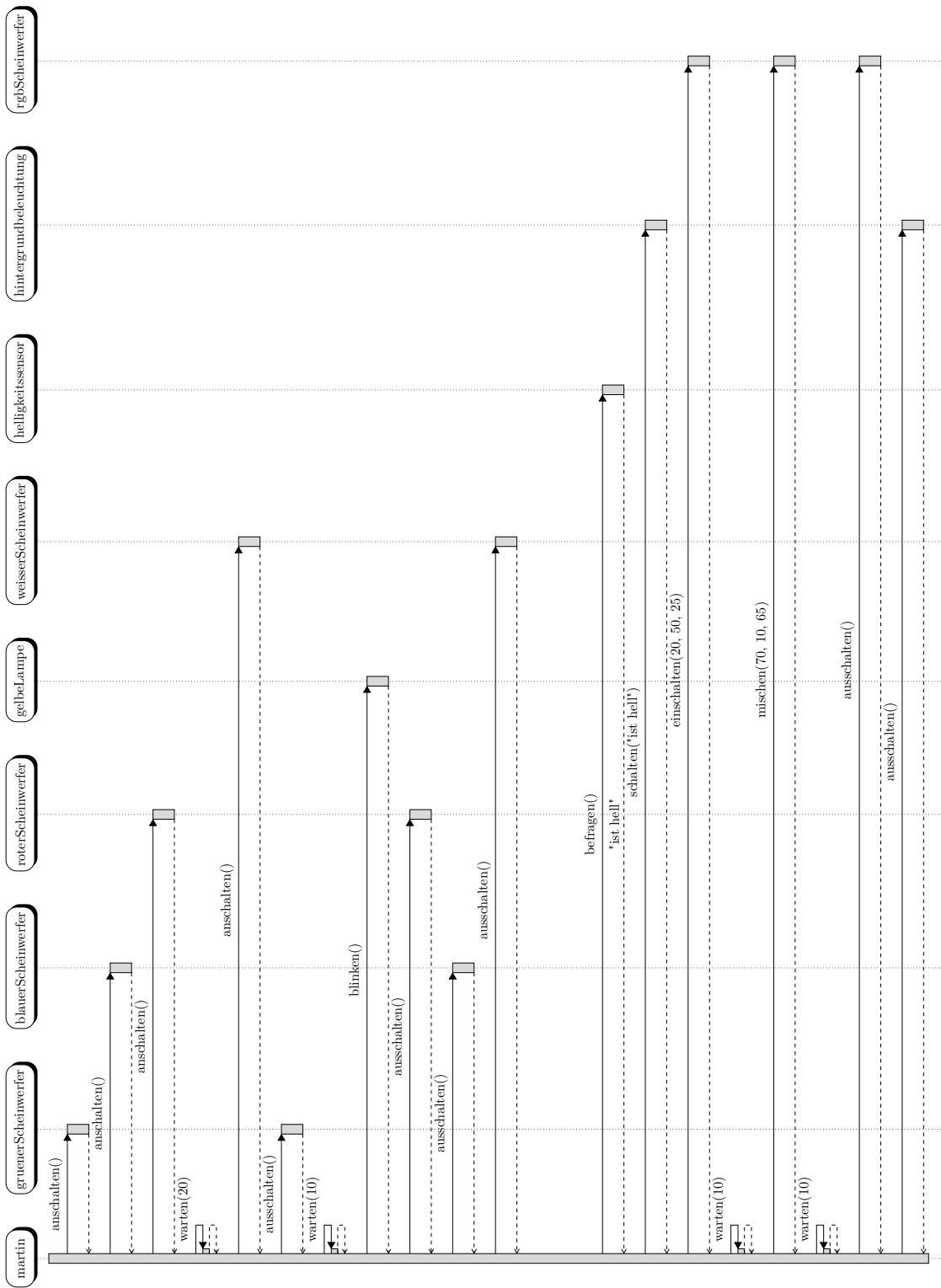


Abbildung A.5: Vollständiges Sequenzdiagramm zum Objektspiel (für die ersten zwei Paragraphen). Der vertikale Abstand markiert den Paragraphenwechsel im Text.

A.2.7 Quelltext Theaterprojekt

Die Befehlsfolge kann in einer Datei `theater.groovy` gespeichert werden und dann in der Groovy-Shell aufgerufen werden. Dazu ist ein Aufruf mit

```
1 :load theater.groovy
```

erfolgen. Alternativ kann die Datei in der Groovy-Console geöffnet und ausgeführt werden.

Im Folgenden Quelltext A.1 sind die Paragraphen des Theater-Textes kenntlich gemacht (s. auch Trennung in Tabelle A.3).

```
1 // Erstelle die Objekte
2 gruenerScheinwerfer = new Scheinwerfer(2);
3 blauerScheinwerfer = new Scheinwerfer(3);
4 roterScheinwerfer = new Scheinwerfer(4);
5 gelbeLampe = new Scheinwerfer(5);
6 weisserScheinwerfer = new Scheinwerfer(6);
7 helligkeitssensor = new Helligkeitssensor(15);
8 hintergrundbeleuchtung = new Hintergrundbeleuchtung(11);
9 rgbscheinwerfer = new RGBScheinwerfer(16, 17, 18);
10
11 // 1. Paragraph
12 // *****
13 gruenerScheinwerfer.anschalten();
14 blauerScheinwerfer.anschalten();
15 roterScheinwerfer.anschalten();
16
17 // Warte 20 Sek
18 // Zur Erklärung:
19 // Thread.sleep(milliseconds) wartet die angegebene Zeit
20 // Ebenfalls funktioniert:
21 // sleep(milliseconds) oder Helfer.warte(seconds)
22 Thread.sleep(20000);
23
24 weisserScheinwerfer.anschalten();
25 gruenerScheinwerfer.ausschalten();
26
27 // Warte 10 Sek
28 Thread.sleep(10000);
29
30 gelbeLampe.blinken();
31 weisserScheinwerfer.ausschalten();
32 roterScheinwerfer.ausschalten();
33 blauerScheinwerfer.ausschalten();
34
```

```

35 // 2. Paragraph
36 // ****
37 antwort = helligkeitssensor.befragenIstHell();
38 hintergrundbeleuchtung.schalten(antwort);
39
40 rgbScheinwerfer.einschalten(20, 50, 25);
41
42 // Warte 10 Sek
43 Thread.sleep(10000);
44
45 rgbScheinwerfer.mischen(70, 10, 65);
46
47 // Warte 10 Sek
48 Thread.sleep(10000);
49
50 rgbScheinwerfer.ausschalten();
51 hintergrundbeleuchtung.ausschalten();
52
53 // 3. Paragraph
54 // ****
55
56 // Wiederhole 45 / 5 = 9 mal:
57
58 // ---- Beginn Wiederholung ----
59 weisserScheinwerfer.anschalten();
60
61 // Warte 5 Sek
62 Thread.sleep(5000);
63
64 weisserScheinwerfer.ausschalten();
65 roterScheinwerfer.anschalten();
66
67 // Warte 5 Sek
68 Thread.sleep(5000);
69
70 roterScheinwerfer.ausschalten();
71 blauerScheinwerfer.anschalten();
72
73 // Warte 5 Sek
74 Thread.sleep(5000);
75
76 blauerScheinwerfer.ausschalten();
77 // ---- Ende Wiederholung ----
78
79

```

```
80 // Dereferenzieren der Objekte
81 // (Die Pinne fuer eine erneute Ausfuehrung wieder frei geben)
82 gruenerScheinwerfer.herunterfahren();
83 blauerScheinwerfer.herunterfahren();
84 roterScheinwerfer.herunterfahren();
85 gelbeLampe.herunterfahren();
86 weisserScheinwerfer.herunterfahren();
87 helligkeitssensor.herunterfahren();
88 hintergrundbeleuchtung.herunterfahren();
89 rgbScheinwerfer.herunterfahren();
90
91 // oder kuerzer: alles herunterfahren mit
92 // Helfer.herunterfahren()
```

Quelltext A.1: Quelltext des Theaterprojekts. Die Paragraphenwechsel der Geschichte sind im Quelltext gekennzeichnet.

A.2.8 Objektspiel mit Objektbeziehungen

Abbildung A.6 zeigt das Ende des Objektspiels mit den Beziehungen zwischen den Objekten. Dabei werden zwei Varianten gezeigt, wie das Schalten der Hintergrundbeleuchtung in Abhängigkeit von dem Helligkeitssensor realisiert (bzw. modelliert) werden kann. Einerseits könnte Martin beide Objekte kennen, andererseits würde es ausreichen, wenn nur die Hintergrundbeleuchtung über ein privates Attribut zum Ansprechen des Helligkeitssensors verfügen würde. Dies hat direkte Auswirkungen auf das Aussehen der entsprechenden Sequenzdiagramme (Anhang A.2.10).

Es wurde auch überlegt, welche Objekte die Bühne kennen muss. Es ist leicht einzusehen, dass die Bühne die Scheinwerfer kennen muss (zur besseren Übersichtlichkeit sind diese Pfeile etwas heller gehalten). Ob die Bühne den Helligkeitssensor kennen muss, ist fraglich. Einerseits, weil in der Geschichte nicht genau spezifiziert wird, wo genau sich der Helligkeitssensor befindet, andererseits bringt es keinen Mehrwert, würde die Bühne diesen kennen. Deswegen ist der Pfeil von der Bühne zum Helligkeitssensor noch heller gefärbt und gestrichelt.

Im Unterricht wurden für gerichtete Beziehungen Garn und entsprechende Pfeile benutzt (s. Anhang S. 113). Zum Beispiel kennt das Objekt `martin` das Objekt `roteLampe` als `roteLampe` (könnte es z. B. aber auch als `meinRoterScheinwerfer` kennen). Entsprechend wurde dann `roteLampe` / `meinRoterScheinwerfer` über den Pfeil geschrieben.

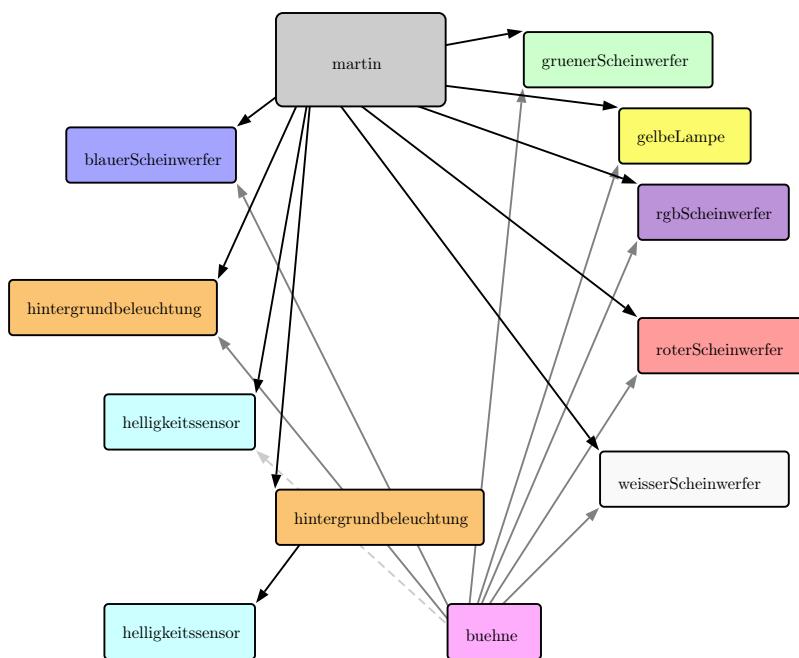


Abbildung A.6: Objektspiel mit Objektbeziehungen.

A.2.9 Übersicht zu Objektbeziehungen

Folgendes Beispiel wurde ausgewählt, um gerichtete und ungerichtete Objektbeziehungen im Unterricht zu erklären. Kommentare sind in grün verfasst.

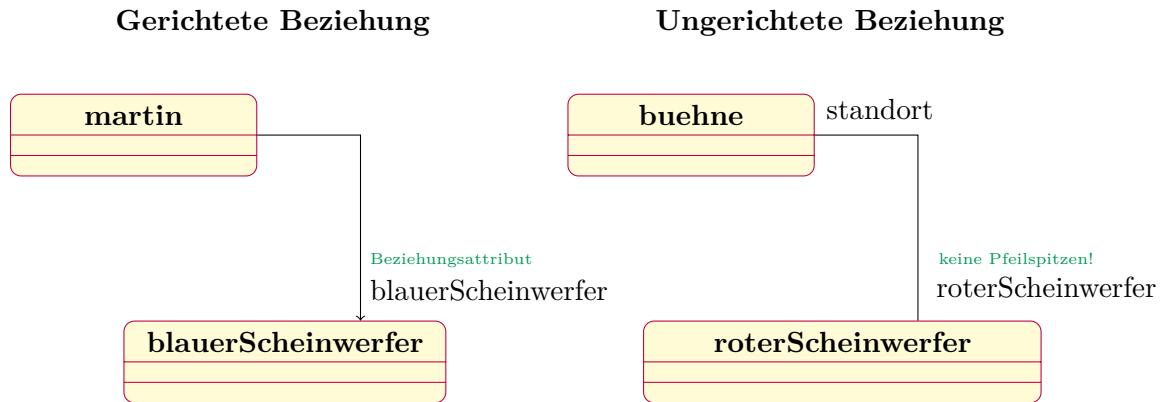


Abbildung A.7: Gerichtete und ungerichtete Objektbeziehung. Kommentare sind in grün gehalten.

A.2.10 Sequenzdiagramme zum Schalten der Hintergrundbeleuchtung

Die Schülerinnen und Schüler programmierten die beiden Varianten, die auf dem Informationsblatt besprochen werden (Anhang S. 123f). Zu beachten sind auch die Quelltexte in Anhang A.2.11.

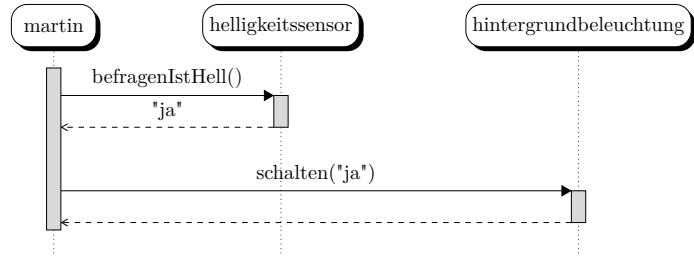


Abbildung A.8: Martin kennt den Helligkeitssensor und auch die Hintergrundbeleuchtung (klassische Situation).

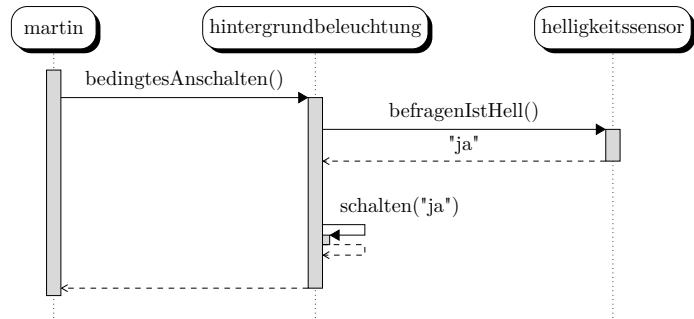


Abbildung A.9: Die Hintergrundbeleuchtung kennt den Helligkeitssensor und ruft bei diesem die Methode `befragenIstHell()` auf. Man beachte die andere Sortierung der Objekte im Vergleich zum Sequenzdiagramm in Abbildung A.8.

A.2.11 Quelltexte zum Schalten der Hintergrundbeleuchtung

Die Schülerinnen und Schüler programmierten die beiden Varianten, die auf dem Informationsblatt besprochen werden (Anhang S. 123f). Zu beachten sind auch die Sequenzdiagramme in Anhang A.2.10.

Variante A

Martin kennt die anderen Objekte.

```
1 Helfer.herunterfahren();  
2  
3 hgb = new Hintergrundbeleuchtung(10);  
4 hs = new Helligkeitssensor(7);  
5  
6 antwort = hs.befragenIstHell();  
7 hgb.schalten(antwort);
```

Quelltext A.2: Martin kennt sowohl Hintergrundbeleuchtung und Helligkeitssensor.

Variante B I

Die Hintergrundbeleuchtung kennt den Helligkeitssensor.

```
1 Helfer.herunterfahren();  
2  
3 hgb = new Hintergrundbeleuchtung(10);  
4 hs = new Helligkeitssensor(7);  
5  
6 // Herstellen der Beziehung  
7 hgb.setzeHelligkeitssensor(hs);  
8 hgb.bedingtesAnschalten();
```

Quelltext A.3: Martin kennt nur die Hintergrundbeleuchtung. Diese muss den Helligkeitssensor erst kennenlernen (Zeile 7), bevor die Hintergrundbeleuchtung geschaltet werden kann. (Genaugenommen kennt Martin, wenn man das Programm als Martin sieht, im Unterschied zur Variante B II in Quelltext A.4 auch den Helligkeitssensor, aber diese kennen sich zusätzlich untereinander).

Variante B II

In dieser Variante wird das Objekt für den Helligkeitssensor nicht benannt, d.h. nur die Hintergrundbeleuchtung kennt es (und keine anderen Objekte können dieses kennen).

```
1 Helfer.herunterfahren();  
2  
3 hgb = new Hintergrundbeleuchtung(10);  
4  
5 // Herstellen der Beziehung  
6 hgb.setzeHelligkeitssensor(new Helligkeitssensor(7));  
7 hgb.bedingtesAnschalten();
```

Quelltext A.4: In dieser Version kennt Martin den Helligkeitssensor gar nicht. Die Hintergrundbeleuchtung lernt einen neuen Helligkeitssensor kennen (Zeile 6), indem sie ein komplett „neues“ Objekt übergeben bekommt.

Anhang B

Materialen zu den Interviews

Die Materialien zu den Interviews umfassen:

- Informationsblatt zu den Interviews (S. 149)
- Einverständniserklärung der Erziehungsberechtigten zum Interview (S. 150)
- Leitfaden zu den Interviews (Anhang B.1, S. 151)
- Transkriptionsregeln (Anhang B.2, S. 152)
- Transkript Interview A (Anhang B.3, S. 153)
- Transkript Interview B (Anhang B.4, S. 160)
- Transkript Interview C (Anhang B.5, S. 164)
- Transkript Interview D (Anhang B.6, S. 171)
- Transkript Interview E (Anhang B.7, S. 177)
- Transkript Interview F (Anhang B.8, S. 182)
- Auswertung der Interviews nach Haupt- und Unterkategorien (Anhang B.9, S. 188)

Dortmund, den 04. Dezember 2016

Informationsblatt zum Interview

Liebe Schülerinnen und Schüler,
Sehr geehrte Eltern,

mein Name ist Heiner Stroick und ich studiere Mathematik und Informatik im fünften Mastersemester an der TU Dortmund. Im Rahmen meiner Masterarbeit begleite ich den Informatik-Kurs in der 10. Klasse seit den Herbstferien.

Für meine Masterarbeit im Fach Informatik untersuche ich den Einstieg in die objektorientierte Modellierung. Für mich ist es dabei interessant zu beobachten, inwiefern sich der Unterricht mit dem Raspberry Pi positiv oder negativ auf das Programmierverständnis und Verständnis objektorientierter Zusammenhänge ausgewirkt hat.

Ich bitte Sie, mir Ihr Einverständnis für die Durchführung eines Interviews mit Ihrer Tochter / Ihrem Sohn zu geben, welches das Verständnis objektorientierter Zusammenhänge evaluieren wird.

Dazu nähere Informationen:

Die Teilnahme der Schülerinnen und Schüler ist vollkommen freiwillig. Alle Daten, die ich während des Projekts erhalte, werden von mir vertraulich behandelt und dienen ausschließlich der wissenschaftlichen Arbeit. Dabei ist das Bundesdatenschutzgesetz bindend. Während des Interviews wird nur der Ton aufgezeichnet — es werden keine Video- oder Fotoaufnahmen gemacht.

Das Interview wird nach Durchführung transkribiert und anonymisiert. Die Tonaufnahme wird danach gelöscht, sodass keine Rückschlüsse auf Ihre Tochter / Ihren Sohn möglich sein werden.

Das Interview soll etwa eine 20 Minuten dauern und ist mit der Schulleitung abgesprochen. **Für die Teilnahme an dem Interview muss die unterschriebene Einverständniserklärung vorliegen.**

Wenn Sie Fragen zum Interview haben, können Sie diese gerne per E-Mail stellen.

Vielen Dank für Ihre Unterstützung.

Mit freundlichen Grüßen



Heiner Stroick
heiner.stroick@tu-dortmund.de

Einverständniserklärung
zum Interview „Verständnis objektorientierter Zusammenhänge“

- Die Teilnahme ist vollkommen freiwillig.
- Für den Umgang mit persönlichen Daten ist das Bundesdatenschutzgesetz bindend. Alle Daten und Informationen werden vertraulich behandelt und dienen allein wissenschaftlichen Zwecken.
- Während des Interviews wird nur der Ton aufgezeichnet. Es werden keine Video- oder Fotoaufnahmen gemacht.
- Das Interview wird nach Durchführung transkribiert und anonymisiert. Die Tonaufnahme wird danach gelöscht, sodass keine Personenrückschlüsse möglich sind.
- Nach Abgabe der Masterarbeit werden alle Kontaktdaten und Tonaufnahmen gelöscht.

Ich erkläre mich hiermit einverstanden, dass mit meiner Tochter / meinem Sohn das Interview unter oben stehenden Bedingungen durchgeführt werden darf. Das Informationsblatt zum Interview habe ich bekommen und akzeptiere die Bedingungen.

Name der Schülerin / des Schülers

Datum und Unterschrift des / der Erziehungsberechtigten

Kontakt für Nachfragen: Heiner Stroick
 heiner.stroick@tu-dortmund.de

B.1 Leitfaden zum Interview

| Nr. | Fragen und Aufforderungen | Themen für mögliche erzählgerierende Nachfragen |
|-----------|--|--|
| Begrüßung | Begrüßung und Vorstellung der Schülerin / des Schülers | |
| Einstieg | Erzähle, welchen Kontakt du bisher zur Informatik hattest. Welche Vorerfahrungen konntest du evtl. schon sammeln? | Privat? Informatik in der Unter- oder Mittelstufe? |
| 1 | Erkläre die im Unterricht kennengelernten Begriffe der Objektorientierung. | »Objekt« / »Methode« / »Attribut« / »Attributwert« / »Parameter« / »Rückgabewert« / (»Klasse«) |
| 2 | Nenne ein Objekt als Beispiel und erkläre, warum das für dich ein Objekt ist. | Was macht ein Objekt aus? Wie wurde im Unterricht mit Objekten gearbeitet? |
| 3 | Erkläre, wie Objekte untereinander Informationen austauschen können. Kannst du ein Beispiel geben? | »Methode«, »Parameter«, »Rückgabewert« |
| 4 | Wie verhält sich ein Objekt, wenn eine Methode auf diesem aufgerufen wird? | Änderung der Attributwerte / unterschiedliches Verhalten bei verschiedenen Methoden bzw. Parameterwerten |
| 5 | evtl. (falls noch nicht angesprochen): Wie haben wir im Unterricht mit den Objekten in der Groovy-Console gearbeitet? | Objekterstellung / Methodenaufrufe? |
| Ausstieg | Bewerte die Arbeit mit der Hardware (den Objekten) und dem Raspberry Pi. Inwiefern war diese Art von Arbeit für dich hilfreich für das Verständnis? | |

Tabelle B.1: Leitfaden zum Interview.

B.2 Transkriptionsregeln

Die Transkriptionsregeln orientieren sich an den in [Koller 2014, S. 314] verwendeten Regeln. Diese sind in Tabelle B.2 aufgeführt.

| | |
|--------------------|---|
| nein | betont / laut |
| <u>nein</u> | gedehnt |
| nei- | Abbruch eines Wortes |
| (.) | kurze Pause |
| (3) | Dauer längerer Pausen in Sekunden |
| (?) | unverständlich |
| /lacht/, /freudig/ | Gestik, Mimik, non-verbale Äußerungen des Sprechenden |
| [mh] | kurze Äußerung des gerade nicht Sprechenden |

Tabelle B.2: Transkriptionsregeln (vgl. [Koller 2014, S. 314]).

B.3 Interview A

| Interview A | |
|-----------------|-------------------|
| Datum / Uhrzeit | 15. Dezember 2016 |
| Dauer | 15:20 (mm:ss) |
| Interviewer | Heiner Stroick |
| Schüler/in | Schüler A |

Tabelle B.3: Informationen zu Interview A.

Transkript Interview A

- 1 I: Ja, vielleicht stellst du dich einmal kurz vor, dass ich deinen Namen habe, und
 2 dann... #00:00:04#
- 3 S: Also ich bin Schüler A, bin 15 Jahre alt und bin jetzt halt in 11 [sic!] auf dem
 4 Joseph-König-Gymnasium #00:00:13#
- 5 I: Hast du schon irgendwelchen Vorkontakt mit Informatik gehabt? #00:00:20#
- 6 S: Nein #00:00:20#
- 7 I: Privat oder so, oder? #00:00:22#
- 8 S: Äh, ein wenig. Ich begeistere mich für die Luftfahrt und versuche, mir zurzeit
 9 ein Home Cockpit aufzubauen und da habe ich schon ein bisschen versucht,
 10 ein paar Knöpfe anzuschließen und so, aber das hat bisher noch nicht ganz so
 11 gut geklappt #00:00:40#
- 12 I: Was ist das, was hast du gesagt? #00:00:40#
- 13 S: Ein Home Cockpit #00:00:44#
- 14 I: Ah, okay. (.) Ist das für so Flugsimulatoren? #00:00:45#
- 15 S: Ja, genau #00:00:50#
- 16 I: Und das ist auch eine Richtung, in der du später studieren willst? Luft- und
 17 Raumfahrttechnik? #00:00:53#
- 18 S: Ja, also, ich versuche es mal, bei der Lufthansa Pilot zu werden, wenn das
 19 nicht klappt schaue ich mal, ob das bei der AirBerlin geht oder wenn nicht,
 20 bei der Flugsicherung, vielleicht als Lotse, so in Richtung Luftfahrt #00:01:07#
- 21 I: Ja, das ist ein spannender Bereich. Und Informatik, wenn du sagst, du hast
 22 vorher noch gar keinen Kontakt habt. Wir haben ja jetzt mit dieser Objek-
 23 torientierung angefangen. Was fällt dir da spontan zu ein, wenn ich einfach
 24 mal so ganz allgemein frage. Welche Begriffe haben wir kennengelernt und wie
 25 würdest du die erklären? #00:01:25#

26 S: Ja, wir haben ja jetzt dieses Objektspiel gemacht, und da haben wir halt diese
27 Objektkarten erstellt, die quasi ein Objekt, wie zum Beispiel einen Scheinwerfer,
28 darstellen und wo dann drin steht, was der machen kann und welchen Status
29 der gerade hat, also wie der gerade drauf ist, und das haben wir dann halt
30 mit dem Objektkartenspiel durchgeführt, und mal versucht, diese Objekte
31 miteinander zu verbinden, dass die auch miteinander interagieren, ja #00:02:04#

32 I: Was macht für dich so ein Objekt aus? Oder was ist für dich ein Objekt?
33 #00:02:08#

34 S: Sowas wie ein Scheinwerfer zum Beispiel. Einfach etwas, womit man interagieren
35 kann #00:02:18#

36 I: (1) Ist das die einzige Begründung für dich, weil ich damit interagieren kann?
37 #00:02:25#

38 S: Ja #00:02:29#

39 I: Kannst du ein anderes Beispiel nennen für ein Objekt? #00:02:31#

40 S: Ich denke mal so, wahrscheinlich ein Schalter oder so, ein Knopf vielleicht
41 #00:02:39#

42 I: Hast du auch Beispiele unabhängig von diesen Bausteinen, mit denen wir am
43 Raspberry Pi gearbeitet haben, von diesen kleinen Platinen? Jetzt aus der
44 richtigen Welt? #00:02:53#

45 S: Müsste ich jetzt überlegen, aber, ich weiß nicht, (2), keine Ahnung #00:03:10#

46 I: Du kannst dir ruhig irgendetwas überlegen, da gibt es jetzt kein richtig und
47 kein falsch #00:03:11#

48 S: Vielleicht ein Bildschirm oder so #00:03:12#

49 I: Und warum wäre das ein Objekt für dich? #00:03:15#

50 S: Weil du den ja auch an anderen Objekten anschließen kannst und dem dann
51 halt sagen kannst, dass er angehen soll, dass er ausgehen soll, sowas #00:03:25#

52 I: Was hättest du da vielleicht noch für Methoden? #00:03:26#

53 S: Beim Bildschirm? #00:03:30#

54 I: Ja #00:03:30#

55 S: Dass der und der Pixel vielleicht diese Farbe hat, und, ja #00:03:36#

56 I: Ja, könnte man sich als Objekt vorstellen, würde ich auch sagen. Wie haben
57 im Unterricht mit den Objekten so insgesamt immer gearbeitet? #00:03:45#

58 S: Ja wir haben das halt erst theoretisch gemacht, wie gesagt dieses Objektkar-
59 tenspiel, haben dann versucht, das quasi in so (.) ein Programm einzubringen,
60 genau das was wir halt gesagt haben, mit diesen bestimmten (.) Phrasen, also,
61 bestimmte Sätze, die wir dann eingebracht haben, und dann quasi genau das

62 was auf den Objektkarten drauf stand, wurde dann genau so in das Programm
63 eingebracht, was wir später mit den Raspberry Pis gemacht haben #00:04:21#

64 I: Jetzt hast du gesagt, genau das, was da drauf stand. Was stand da so drauf?
65 #00:04:23#

66 S: Zum Beispiel *blinken*, oder so. Oder *anschalten* #00:04:28#

67 I: Warum war das wichtig, dass wir das genau so machen müssen? #00:04:30#

68 S: Weil man das genau so ins Programm reinschreiben konnte (.) und ich dachte
69 zum Beispiel vorher, dass man da einfach nur *Eins* oder *Null* reinschreibt oder
70 so, aber ich wusste nicht, dass man da wirklich reinschreiben kann *einschalten*
71 oder *blinken* oder so #00:04:44#

72 I: Wie meinst du, *Eins* oder *Null* rein schreiben? #00:04:45#

73 S: Ich weiß nicht, ich wusste ja vorher nicht, wie man so ein Programm program-
74 miert. Und da dachte ich, dass man irgendwie reinschreibt *Scheinwerfer Eins*
75 und dass der dann an ist, aber ich wusste nicht, dass es auch mit *einschalten*
76 geht zum Beispiel #00:05:00#

77 I: Okay, vielleicht kannst du mal erklären, wie jetzt, wenn du jetzt ein Programm
78 hast und du hast verschiedene Objekte, wie die miteinander, du hast vorhin
79 schon interagieren genannt, war es glaube ich, wie die interagieren können?
80 #00:05:17#

81 S: Wir hatten ja zum Beispiel das Beispiel mit der Hintergrundbeleuchtung und
82 dem Helligkeitssensor, dass der, dass die Hintergrundbeleuchtung quasi direkt
83 auf den Helligkeitssensor zugreift und halt guckt, wie hell es ist, und je nach
84 dem dann halt die Hintergrundbeleuchtung schaltet #00:05:31#

85 I: Ja, wie funktioniert das genau? #00:05:34#

86 S: Ja, da konnte man ins Programm reinschreiben, dass die Hintergrundbeleuch-
87 tung den Helligkeitssensor befragt, ob es hell ist, und je nach dem was dann
88 für eine Antwort kam, hat die Hintergrundbeleuchtung sich dann an- oder
89 ausgeschaltet #00:05:51#

90 I: Okay, das ist diese eine Form von Methoden, es gibt auch noch andere Metho-
91 den. Du kannst ja entweder den Helligkeitssensor befragen, *Helligkeitssensor,*
92 *ist es hell gerade*, oder die hatten wir glaube ich *befragen ist hell* genannt,
93 die Methode, und es gibt noch andere Methoden, die hatten wir zum Beispiel
94 beim RGB Scheinwerfer einschalten gesehen oder beim Farben mischen. Was
95 war da der zentrale Unterschied? Wenn du sagst, RGB Scheinwerfer oder mein
96 RGB Scheinwerfer Punkt mischen? Was ist da der Unterschied zwischen, oder
97 zum Helligkeitssensor, den du befragst? #00:06:39#

98 S: Ach so, dass man da die, also beim RGB Scheinwerfer konnte man die Parameter
99 direkt eingeben, dass rot zum Beispiel 40 Prozent hat oder so, das muss man
100 direkt ins Programm rein schreiben und beim Helligkeitssensor war es so, dass
101 das Programm sich selbst quasi einmal abfragt, welcher Parameter da rein
102 muss #00:07:03#

- 103 I: Ja, dieses sich selber einmal abfragen, wie war das in der Geschichte beschrieben? #00:07:12#
- 105 S: Äh, ich weiß, ich glaube, Martin befragt den Helligkeitssensor, ob es hell ist und schaltet je nach dem die Hintergrundbeleuchtung an #00:07:18#
- 107 I: Ok, und in der letzten Stunde hatten wir dann auch noch über eine andere Möglichkeit gesprochen, wie man das auch realisieren kann, dass man diese Hintergrundbeleuchtung schalten kann mit dem (.), mit der Antwort des Helligkeitssensor. Weißt du wie die aussah? #00:07:35#
- 111 S: Ja, das wurde dann in eine Zeile geschrieben, aber ganz genau weiß ich das auch nicht mehr #00:07:42#
- 113 I: Bei dem Objektspiel haben wir ja auch vor zwei Stunden oder mit so einem Garn gearbeitet, mit so einem Band praktisch. Was sollte das symbolisieren? #00:07:53#
- 116 S: Die Verbindung zwischen den Objekten, dass sie miteinander interagieren können, dass die sich quasi gegenseitig kennen #00:08:03#
- 118 I: Warum ist das wichtig, dass die sich kennen? #00:08:04#
- 119 S: Weil die sonst nicht wissen, wo, wie, wo die das hin schicken können. Zum Beispiel dass Martin weiß, dass es den grünen Scheinwerfer gibt und dass der das dann da auch direkt hin schicken kann #00:08:27#
- 122 I: Okay, was meinst du mit Befehl hin schicken? #00:08:30#
- 123 S: Ja zum Beispiel *anschalten* oder *ausschalten* #00:08:35#
- 124 I: Und wie verhält sich jetzt so ein Objekt, wenn darauf eine Methode aufgerufen wird? Also du hast jetzt vorhin Befehl gesagt, aber du meinst ja denke ich Methode #00:08:43#
- 127 S: Ja #00:08:44#
- 128 I: Wie verhält sich so ein Objekt, wenn da jetzt wirklich eine Methode aufgerufen wird? #00:08:50#
- 130 S: Das führt diese Methode aus und ändert dann den Status noch, dass es auch weiß, dass es an ist zum Beispiel #00:09:04#
- 132 I: (1) Ja, kannst du noch ein bisschen mehr dazu erzählen? Vielleicht auch bei einem anderen Beispiel, was da so für dich passiert, wenn da eine Methode aufgerufen wird? #00:09:12#
- 135 S: Ja, das guckt halt erstmal nach, ob es diese Methode gibt, und wenn die Methode vorhanden ist, ja führt das die einfach aus und guckt nach, ob danach noch irgendetwas passieren muss, ob es den Status geändert werden muss, oder so etwas #00:09:29#
- 139 I: Und jetzt beim Helligkeitssensor? Wenn du den fragst, *bist du gerade hell*, sozusagen, was passiert da noch bei diesem Methodenaufruf? #00:09:41#

- 141 S: Dann gibt es eine (.) Rückgabe, und dann sagt der Helligkeitssensor zum
142 Beispiel, dass es hell ist, und schickt das dann wieder zurück #00:09:50#
- 143 I: Hast du ein Beispiel aus der realen Welt, sag ich mal, wo so eine Interaktion
144 passiert? #00:10:02#
- 145 S: Ich könnte jetzt den Helligkeitssensor vom Handy vielleicht nehmen, aber... /lacht/
146 #00:10:10#
- 147 I: /lacht/ Vielleicht fällt dir ja irgendetwas ein, morgens auf dem Schulweg zum
148 Beispiel, wo würden da vielleicht Objekte auftreten und wo du sagst, ich rufe
149 bei dem Objekt eine Methode auf, die gibt mir was zurück, und... #00:10:24#
- 150 S: Ja, äh, vielleicht wenn man wen anders grüßt, zum Beispiel. Dass der dann
151 auch zurück grüßt #00:10:28#
- 152 I: Wie würde das da so konkret aussehen? Vielleicht kannst du mal zwei Objekte
153 nennen mit Namen und mal so einen Methodenaufruf dafür formulieren? #00:10:37#
- 154 S: Wenn ich das wieder mit Menschen machen kann, dass ich und ein Freund von
155 mir, dass ich ihm zum Beispiel rufe, zurufe, Hallo, und er dann zurück ruft,
156 hallo #00:10:53#
- 157 I: Nenne es mal so, wie du es aufschreiben würdest in der Groovy Konsole, wo
158 wir mit gearbeitet haben #00:11:01#
- 159 S: Ja, ich würde den Freund halt erstmal hinzufügen, dass ich den auch kenne,
160 und (2) weiß nicht, dann würde ich hinschreiben, *rufen Hallo* oder so #00:11:17#
- 161 I: Wenn du willst, kannst du auch gerne was aufschreiben dazu #00:11:22#
- 162 S: Naja ich weiß es jetzt nicht, wie ich das machen kann #00:11:25#
- 163 I: Okay, aber dieses hinzufügen, wofür war das nochmal wichtig? #00:11:31#
- 164 S: Dass die sich gegenseitig kennen, das war auch das mit den (1) Fäden, die das
165 auch im Objektkartenspiel gemacht haben #00:11:39#
- 166 I: Wie würdest du die Arbeit insgesamt bewerten, die wir da jetzt mit dem
167 Raspberry Pi durchgeführt haben und der Geschichte? #00:11:50#
- 168 S: Mh, ich finde es eigentlich super, weil wir vorher halt auch erstmal theoretisch
169 alles gemacht haben, damit wir halt auch erstmal einen groben Überblick
170 haben, wie das überhaupt geht, und dass wir danach (.) an die Pis gegangen
171 sind, wir haben ja auch ein Arbeitsblatt bekommen, und das konnten wir dann
172 quasi Schritt für Schritt durchgehen #00:12:07#
- 173 I: War das für dich hilfreich für das Verständnis? #00:12:11#
- 174 S: Ja #00:12:12#
- 175 I: Weil? #00:12:14#
- 176 S: Weil man direkt wusste, was man machen muss #00:12:16#

- 177 I: Ach so, ich meine jetzt nicht das Arbeitsblatt, sondern (.) mit diesen kleinen
178 Bauteilen, mit denen wir da gearbeitet haben, die wir angeschlossen haben.
179 Hat das irgendwie für dich zum Verständnis beigetragen? #00:12:31#
- 180 S: Ja, also, wir haben das einfach gemacht (1) zu welchem Verständnis? #00:12:43#
- 181 I: Zum Objektverständnis, kannst du dir jetzt vorstellen, was ein Objekt ist? Oder
182 wie sich zwei Objekte vielleicht unterscheiden oder was ein Objekt auszeichnet?
183 #00:12:53#
- 184 S: Ja, dass die unterschiedliche Methoden haben und dass, ja, dadurch wurde
185 einem das noch ein bisschen näher gebracht, finde ich #00:13:07#
- 186 I: Haben wir noch irgendetwas vergessen? Oder würdest du sagen, ach, das ist
187 mir jetzt noch eben eingefallen, das könnte ich auch noch so und so erklären?
188 #00:13:10#
- 189 S: (5) Ne #00:13:13#
- 190 I: Dann lass uns noch mal ganz kurz, dann gehen wir nochmal eben einen Schritt
191 zurück, wir haben vorhin darüber gesprochen, wie sich Objekte verhalten,
192 wenn man Methoden auf denen aufruft. Da hatten wir (.) das Beispiel mit
193 dem Helligkeitssensor auch [mh], dass du den fragst und dass dann immer eine
194 unterschiedliche Antwort kommt [ja], du hast gesagt, je nach dem, wird das
195 so und so geschaltet, entweder aus gelassen oder angeschaltet. Wenn wir jetzt
196 nochmal in die reale Welt gehen, vielleicht kannst du dir da wirklich noch
197 einmal ein Beispiel überlegen, wo so etwas auch vorkommt, wo je nach dem
198 was für eine Antwort du kriegst, unterschiedlich dich verhältst #00:14:08#
- 199 S: Ja, ich weiß, wenn einem..., wir hatten das Beispiel auch, wenn einem zum
200 Beispiel jemand zuruft, ob die Ampel rot ist, dann muss man ja auch je
201 nach dem antworten, ob die rot oder ob die grün ist, da muss man halt
202 unterschiedlich reagieren #00:14:24#
- 203 I: Ja, wann war das? War das letzte Stunde? #00:14:37#
- 204 S: Ne, ist schon ein bisschen länger her #00:14:39#
- 205 I: Vielleicht kannst du das noch einmal ein bisschen genauer erklären weil das
206 hatte ich jetzt nicht mtibekommen #00:14:39#
- 207 S: Ja, es gab halt das Beispiel, dass man vorne an einer Ampel steht und von
208 hinten ruft einem jemand zu, ob es noch rot ist [ach so], und dann muss man
209 halt antworten, auch wenn man die Person nicht kennt, ob es rot ist oder grün
210 #00:14:54#
- 211 I: Ja, was antwortest du dann? #00:14:53#
- 212 S: Ja, zum Beispiel *ist noch rot* oder so #00:14:58#
- 213 I: Okay, und was würde die Person dann mit dieser Antwort machen? #00:15:05#
- 214 S: Ja, sich bedanken und dann weiß sie es halt und bleibt halt noch stehen #00:15:12#

215 I: Dann haben wir eigentlich über alles gesprochen glaube ich. Dann sage ich
216 erstmal danke, dann hätten wir das auch geschafft #00:15:19#

217 S: Gerne #00:15:20#

B.4 Interview B

| Interview B | |
|-----------------|-------------------|
| Datum / Uhrzeit | 15. Dezember 2016 |
| Dauer | 10:01 (mm:ss) |
| Interviewer | Heiner Stroick |
| Schüler/in | Schüler B |

Tabelle B.4: Informationen zu Interview B.

Transkript Interview B

- 1 I: Also nochmal danke, dass du mitmachst. Vielleicht stellst du dich einmal kurz
 2 vor, dass ich deinen Namen auf der Aufnahme habe, und dann schreibe ich
 3 später Schüler A oder sowas #00:00:09#
- 4 S: Ich bin Schüler B /lacht/ #00:00:13#
- 5 I: /lacht/ Okay, ich würde am Anfang gerne wissen, ob du schon irgendwie
 6 vielleicht Vorerfahrungen, oder vorher schon mal Kontakt zur Informatik
 7 hattest? #00:00:19#
- 8 S: Ich hatte in der Neunten schon Informatik und mein bester Freund studiert
 9 Wirtschaftsinformatik und mein Bruder hat halt auch Informatik und dann...
 10 #00:00:35#
- 11 I: Dein Bruder ist, geht hier auch zur Schule? #00:00:35#
- 12 S: Der ist in der Q2 #00:00:36#
- 13 I: Okay, ansonsten privat schon mal irgendetwas gemacht? #00:00:45#
- 14 S: Nö, (1) ich glaub mir fällt nichts ein #00:00:48#
- 15 I: Okay, wenn wir jetzt so über den Unterricht sprechen, dann haben wir ja mit
 16 dieser Objektorientierung angefangen. Kannst du erklären, was du dir darunter
 17 vorstellst und welche Begriffe da aufgetaucht sind? #00:01:05#
- 18 S: Abbott #00:01:05#
- 19 I: Was fällt dir dazu ein? #00:01:11#
- 20 S: Das ist glaube ich der Mensch, der das mit den Objekten, Objektnamen, also
 21 den Kandidaten für Objekte, Methoden und Attribut mit Attributwerten
 22 gemacht hat #00:01:25#
- 23 I: Ja, was stellst du dir unter so einem Objekt vor? #00:01:26#
- 24 S: Einen Gegenstand, den man berühren kann (1) /lacht/ #00:01:36#

- 25 I: Kannst du ein Beispiel nennen und erklären, warum das für dich ein Objekt
26 ist? #00:01:41#
- 27 S: Ein Scheinwerfer, ja, weil den kann man halt steuern und den kann man auch
28 berühren, der ist ja jetzt nicht so wie Luft, also Luft berührt man zwar immer
29 /lacht/, aber, ja #00:01:57#
- 30 I: Ein anderes Beispiel? #00:01:57#
- 31 S: Der Mensch, irgend ein Name, Dieter, Martin #00:02:05#
- 32 I: Warum wären das dann Objekte? #00:02:08#
- 33 S: Ja, je nach dem, also wir haben das ja jetzt mit Geschichten gemacht, da
34 machen die ja auch bestimmte Sachen. Also die führen ja Sachen aus, zum
35 Beispiel bei anderen Objekten, wie jetzt zum Beispiel dem Scheinwerfer #00:02:22#
- 36 I: Ja, kannst du erklären, wie das funktioniert, wenn du sagst, die führen Sachen
37 aus? #00:02:31#
- 38 S: Ja, also, jedes Objekt hat ja zugehörige Methoden, und die kann man halt
39 aufrufen. Zum Beispiel den Scheinwerfer, den kann man ja auf jeden Fall schon
40 mal *anschalten* und *ausschalten* (.) und das muss ja irgendjemand machen,
41 der kann sich ja nicht von alleine anschalten #00:02:55#
- 42 I: Wie funktioniert das genau, wenn du jetzt mehrere Objekte hast? Wenn die
43 miteinander kommunizieren wollen? Wenn du jetzt ein größeres Programm
44 hast, hast du ja denke ich mal nicht nur ein Objekt, sondern mehrere. Wie
45 kommunizieren die untereinander? #00:03:11#
- 46 S: Man stellt Beziehungen zwischen denen her, gerichtete oder eine ungerichtete,
47 ja, und dann, also, wir hatten das ja mit Martin und dem Scheinwerfer. Da
48 muss der Scheinwerfer ja nicht unbedingt Martin kennen, sondern nur Martin
49 den Scheinwerfer, damit er bei dem Scheinwerfer dann eine Methode aufrufen
50 kann #00:03:33#
- 51 I: Und dann hatten wir auch noch mit diesem Helligkeitssensor uns beschäftigt.
52 Wie funktionierte das da? #00:03:45#
- 53 S: Ja wir hatten ja mehre Möglichkeiten. Entweder man speichert den, also, Martin
54 fragt erst den Helligkeitssensor ob es hell ist und speichert die Antwort dann
55 unter einer Variablen, mit der Variablen, die gibt er dann als Parameterwert
56 weiter an die Hintergrundbeleuchtung oder wir hatten, dass der Martin nur
57 die Hintergrundbeleuchtung kennt, die anschaltet und die dann halt erst den
58 Helligkeitssensor fragt und dann halt je nach dem anschaltet oder nicht #00:04:19#
- 59 I: Wo ist da der Unterschied? #00:04:23#
- 60 S: Wir müssen nicht so viele Beziehungen herstellen. Also Martin muss halt zum
61 Beispiel bei der zweiten Methode nicht den Helligkeitssensor kennen, sondern
62 nur die Hintergrundbeleuchtung #00:04:36#

- 63 I: Ja, ich hab vorhin vergessen zu sagen, wenn du etwas aufschreiben möchtest,
64 kannst du das gerne tun, um das ein bisschen besser zu strukturieren vielleicht,
65 wenn dir das dann einfacher fällt. Ja, dann haben wir jetzt über diese Methoden
66 gesprochen, du sagst die schaltet je nach dem die Hintergrundbeleuchtung an,
67 wie funktioniert das mit diesem je nach dem nochmal? #00:04:58#
- 68 S: Ja, also ich glaube, die Methode hieß *bedingtes Anschalten* und dann befragt
69 die Hintergrundbeleuchtung erst den Helligkeitssensor mit *befragen ist hell*,
70 wenn die Antwort dann *ja* ist, dann schaltet die sich nicht an und wenn die
71 Antwort *nein* ist, dann schaltet die sich an #00:05:15#
- 72 I: Ok, und wie verhält sich allgemein so ein Objekt, wenn man darauf eine
73 Methode aufruft. Was passiert da? #00:05:23#
- 74 S: Das kriegt im Sequenzdiagramm so einen Kasten, also eine Lebenslinie hat es
75 ja schon, aber so einen Aktivitätskasten (1) /lacht/ #00:05:37#
- 76 I: Und wenn du dir, wenn wir jetzt auf Attribute zu sprechen kommen... #00:05:41#
- 77 S: Die ändern sich, zum Beispiel, wenn man den Scheinwerfer jetzt anschaltet,
78 dann ändert sich der Status auf an und nicht mehr auf aus #00:05:55#
- 79 I: Okay. Ist dir vielleicht inzwischen ein Beispiel eingefallen für ein Objekt aus
80 der realen Welt? Du hattest jetzt vorhin nur Dieter genannt, also Menschen,
81 aber vielleicht fällt dir noch ein anderes Beispiel ein? #00:06:10#
- 82 S: (5) /lacht/ #00:06:15#
- 83 I: Sonst gebe ich dir mal ein Beispiel. Stell dir zum Beispiel mal ein Objekt Auto
84 vor. Was würdest du da für Methoden vielleicht haben? #00:06:28#
- 85 S: Erst mal anfahren, bremsen, (3) anschalten, ausschalten, also den Motor,
86 vielleicht auch die Lampen oder das Navigationssystem noch irgendwie oder
87 Musik noch irgendwie da reinbringen #00:06:41#
- 88 I: Was für Attribute hättest du da vielleicht? #00:06:44#
- 89 S: Vielleicht die Farbe (2) oder den Status #00:06:51#
- 90 I: Was meinst du mit Status? #00:06:52#
- 91 S: Ob das gerade an ist oder nicht. Oder auch – wobei das macht glaube ich nicht
92 so viel Sinn –, wenn man noch macht, ob das gerade auf der Autobahn ist oder
93 auf der Straße oder so, also den Standort #00:07:10#
- 94 I: Ja, wäre fürs Navigationssystem ja schon wichtig #00:07:13#
- 95 S: Ja /lacht/ #00:07:16#
- 96 I: Ja, da gibt es kein richtig und kein falsch. Da gibt es verschiedene Möglichkeiten.
97 Dann haben wir im Unterricht ja mit dieser Groovy Konsole gearbeitet, kannst
98 du da erklären, wie das mit den Objekten da drin funktioniert hat? #00:07:27#

- 99 S: Also erstmal musste man glaube ich irgendwie *roter Scheinwerfer ist gleich new Scheinwerfer*, also erstmal die ganzen Objekte registrieren sage ich mal, dann
100 musste man halt immer Objekt und dann Punkt und dann die Methode je nach
101 dem Klammer auf, Klammer zu oder dann noch irgendwie einen Parameterwert
102 und dann so ein Semikolon und dann in der nächsten Zeile weiter. Und ich
103 glaube am Ende musste man noch irgendwas mit *Helper herunterfahren* oder
104 so oder am Anfang, je nach dem #00:08:04#
105
- 106 I: Ja, ja das lag jetzt am Raspberry Pi, dass diese Pinne wieder frei gegeben
107 werden, dass man dann sozusagen das ganze Programm nochmal durchlaufen
108 lassen kann. Aber du hast vorhin am Anfang gesagt, diese Objekte muss man
109 registrieren. Warum ist das wichtig? #00:08:20#
- 110 S: Weil das vielleicht, Martin war ja jetzt sozusagen der Computer, dass der die
111 auch kennt, und nicht einfach irgendein oder dass der auch weiß zum Beispiel,
112 Scheinwerfer haben ja jetzt bestimmte Methoden, wenn man das dann unter
113 Scheinwerfer registriert hat, dass der dann nur bestimmte Methoden offen
114 hat, als wenn man das unter RGB Scheinwerfer oder Helligkeitssensor macht
115 #00:08:44#
- 116 I: Ja, wenn du dir die Arbeit jetzt insgesamt anguckst, wie wir da mit der
117 Groovy Konsole gearbeitet haben und dem Raspberry Pi. Wie würdest du das
118 bewerten? #00:08:57#
- 119 S: Ja, an sich fand ich das ganz gut, nur ich fand es halt schade, dass halt, es gibt
120 halt Leute, die hatten schon Informatik vorher, die waren dann immer schneller
121 als die anderen und dann mussten die halt immer öfters warten #00:09:12#
- 122 I: Hat das für dich zum Verständnis beigetragen, dass wir das so gemacht haben,
123 wie wir es jetzt gemacht haben? #00:09:17#
- 124 S: Ich weiß nicht, weil eigentlich konnte ich ja schon vorher programmieren,
125 also... #00:09:27#
- 126 I: Auch die Objektorientierung insbesondere? #00:09:26#
- 127 S: Nö, aber das war jetzt nicht so schwer. /lacht/ #00:09:31#
- 128 I: Haben wir noch irgendwas vergessen, oder würdest du sagen, das würde ich
129 vielleicht noch mal so oder so erklären? #00:09:39#
- 130 S: (6) Ich glaube nicht. (5) #00:09:53#
- 131 I: Ok, dann haben wir auch über alles gesprochen, also nochmal danke #00:10:01#

B.5 Interview C

| Interview C | |
|-----------------|-------------------|
| Datum / Uhrzeit | 14. Dezember 2016 |
| Dauer | 19:51 (mm:ss) |
| Interviewer | Heiner Stroick |
| Schüler/in | Schüler C |

Tabelle B.5: Informationen zu Interview C.

Transkript Interview C

- 1 I
- 2 I: [...] Wie gesagt nochmal danke, dass du mitmachst. Das soll jetzt auch nur so
3 kurz – 20 Minuten – dauern oder so, und es wäre ganz cool, wenn du einmal
4 noch deinen Namen sagst, damit ich das auf der Aufnahme habe, aber den
5 schwärze ich dann nachher, also eben sagst, wer du bist #00:00:24#
- 6 S: /lacht/ Ich bin Schüler C #00:00:27#
- 7 I: Super, danke. Dann würde ich als erstes gerne mal wissen, welchen Kontakt
8 du bisher schon mal zur Informatik hattest. Welche Vorerfahrungen konntest
9 du schon sammeln? #00:00:37#
- 10 S: Eigentlich, also, so vorm Informatikunterricht gar nicht. Mein Onkel der
11 macht gerne was mit Technik und so und hat mir deswegen schon relativ früh
12 verschiedene Sachen geschenkt mit denen ich dann aber auch nichts anfangen
13 konnte. Und der hat mir zum Beispiel auch von Lego Technik und so auch
14 verschiedene Roboter gezeigt, die man programmieren kann, und mir auch
15 was erklärt, aber da weiß ich eigentlich gar nichts mehr von, weil das mich
16 halt damals noch nicht so wirklich interessiert hat. Und, ja, dann konnte man
17 ja in der Achten glaube ich nochmal wählen, da hab ich mich dann nicht für
18 Informatik entschieden und hab dann halt jetzt gedacht zur Oberstufe, dass
19 ich es ja mal probieren kann #00:01:31#
- 20 I: Ja, was hattest du davor gewählt? #00:01:31#
- 21 S: Kunst #00:01:39#
- 22 I: Okay, aber sonst mit den Lego Robotern auch nicht viel zuhause dann auch
23 gemacht, sondern eher so im Regal stehen lassen? #00:01:42#
- 24 S: Ja, genau /lacht/ #00:01:45#
- 25 I: /lacht/ Vielleicht kannst du mal ein bisschen erklären, welche Begriffe wir im
26 Unterricht jetzt kennengelernt haben zur Objektorientierung und was da so
27 hinter steckt? Wenn du mal einfach so aufzählst und erklären kannst, was dir
28 da so spontan alles zu einfällt? #00:02:02#

- 29 S: Ähm, (.) weiß ich nicht. Ich kann nicht (.) #00:02:15#
- 30 I: Ist ja vielleicht auch ein bisschen allgemein gefragt. Aber wenn du dir jetzt
31 ein Objekt vorstellst – was kannst du dazu sagen? #00:02:23#
- 32 S: Ja, man, kann das ja programmieren. Wenn man jetzt zum Beispiel einen
33 Scheinwerfer nimmt, dass man anhand eines Programms sagt, wann der angehen
34 soll, wann der ausgeht, wie lange er anbleiben soll und so, und man kann halt
35 da ja verschiedene Funktionen einstellen und auch gleichzeitig verschiedene
36 Sachen schalten #00:02:48#
- 37 I: Welche Begriffe fallen dir da noch zu ein? Du hast jetzt gesagt Funktionen?
38 #00:02:55#
- 39 S: Methoden mit Attributwerten und Parameterwerten und so #00:03:03#
- 40 I: Wo ist da der Unterschied? #00:03:07#
- 41 S: (1) Wenn man jetzt zum Beispiel einen Scheinwerfer hat, hat man das Attribut
42 Standort zum Beispiel mit dem Para– Der Parameterwert ist auch irgendwie
43 so Status ob *an* oder *aus*, und der Attributwert das andere #00:03:41#
- 44 I: Der Attributwert gehört ja schon zum Attribut. Für den Standort, für das
45 Attribut Standort, wäre #00:03:45#
- 46 S: Wäre dann Bühne zum Beispiel #00:03:45#
- 47 I: Genau. Über Parameter sprechen wir gleich vielleicht noch drüber und über
48 Methodenaufrufe und so. Aber wenn du dir jetzt ein anderes Objekt vorstellst.
49 Kannst du mir da irgendwie ein Beispiel für nennen? #00:04:03#
- 50 S: (1) weiß ich nicht. Alles mögliche, eigentlich, oder? #00:04:12#
- 51 I: Ja, gib ruhig mal ein Beispiel und dann erkläre mal, warum das für dich ein
52 Objekt ausmacht #00:04:14#
- 53 S: Also, ich finde ein Objekt kann eigentlich alles sein, weil, entweder wird dieses
54 Objekt angeschaltet oder ausgeschaltet und man macht irgendeine Methode
55 mit dem, sag ich jetzt mal. Oder wir hatten ja auch zum Beispiel Würfel, der
56 wird ja, also der Würfel macht ja nichts, aber der Würfel ist ja trotzdem ein
57 Objekt, mit dem was gemacht wird #00:04:49#
- 58 I: Und wenn du jetzt so erklären müsstest, was ein Objekt insgesamt ist? Da
59 hast du gesagt, dass kann alles sein. Was zeichnet es aus? Wie sagst du, oder
60 wie kannst du trennen, das wäre ein Objekt und das wäre kein Objekt? #00:05:10#
- 61 S: (.) weiß ich nicht, also mir fällt jetzt gerade nichts ein, was kein Objekt ist
62 #00:05:24#
- 63 I: Fällt dir denn noch ein Beispiel ein für was, was ein Objekt wäre? #00:05:25#
- 64 S: Ja, äh, Kaffeemaschine (1) weiß ich nicht #00:05:41#

- 65 I: Ja, doch, das könnte man sich durchaus als Objekt vorstellen. Was wären da
66 so für Methoden von dem Objekt? #00:05:46#
- 67 S: *Anschalten, ausschalten*, dann halt *verschiedene Sorten Kaffee machen, Bohnen mahlen* oder sowas... #00:06:01#
- 69 I: Sowas würde mir auch einfallen, oder *reinigen* oder so. Genau, sowas. Und
70 Attribute von einer Kaffeemaschine? Was würdest du da sagen? #00:06:08#
- 71 S: Standort Küche, oder so. Weiß ich nicht #00:06:27#
- 72 I: Da gibt es schon noch mehr #00:06:27#
- 73 S: (lachend) Ja, klar, aber... #00:06:22#
- 74 I: Was interessiert dich denn von einer Kaffeemaschine, wenn du Kaffee machen
75 willst? #00:06:26#
- 76 S: Ja der Kaffee /lacht/ #00:06:26#
- 77 I: /lacht/ Du sagtest ja vorhin so Bohnen mahlen, wie könnte man das vielleicht
78 auch mit einem Attribut in Verbindung bringen? Dass man vorher
79 fragt... #00:06:43#
- 80 S: Ob man die grob oder fein... Ich hab keinen Plan... /lacht/ #00:06:45#
- 81 I: Ne, ich mein ob noch Bohnen da sind oder so (.) #00:06:47#
- 82 S: (2) Joar, vielleicht halt ob noch was in der Kaffeemaschine drin ist oder so
83 zum Beispiel auch Wasser, muss man ja vorher auch einfüllen. Dass das selbst
84 sag ich jetzt mal misst, ob da noch Wasser drin ist oder nicht und dann zeigt
85 der halt an, *Wasser nachfüllen*, oder so #00:07:14#
- 86 I: Okay, was könnte man noch als Attribut nehmen? Fällt dir noch was ein?
87 #00:07:26#
- 88 S: Ne #00:07:26#
- 89 I: Wenn du sagst mit diesem Wasser nachfüllen, was wäre dann das Attribut
90 und wäre der Attributwert? Kannst du da ein Beispiel für geben, wie du das
91 Attribut nennen würdest und was das für einen Wert zum Beispiel haben
92 könnte? #00:07:44#
- 93 S: Vielleicht irgendwie *Füllmenge Wasser* und dann *voll* oder *leer* oder irgendwie
94 sowas. Oder wenn das halt irgendwie *WasserVoll ja nein* irgendwie #00:08:10#
- 95 I: Ja, okay, es gibt da ja kein richtig und kein falsch. Also es gibt da verschiedene
96 Möglichkeiten und man kann sich das durchaus so vorstellen (.) Wie können
97 denn Objekte jetzt untereinander kommunizieren? Wie kann denn jetzt ein
98 Objekt mit dem anderen in Interaktion treten? Das hatten wir ja auch gesehen
99 in diesem Theaterprojekt. Vielleicht kannst du da mal erklären, wie das
100 funktioniert? Musst du dir jetzt nicht am Kaffeemaschinen-Beispiel überlegen,
101 ist vielleicht auch schwierig, da jetzt spontan was zu finden, aber vielleicht
102 kannst du so grob erklären, wie Objekte Informationen austauschen? #00:08:49#

- 103 S: Ja, wir hatten das ja mit dem Scheinwerfer und dem Helligkeitssensor. Und
 104 dass man da halt, also, man möchte ja das eine Gerät, bei dem einen Gerät was
 105 auslösen oder so, und das wird dann ja mit der Antwort des anderen Geräts
 106 geschaltet und damit das halt an gehen kann, fragt das Gerät sozusagen das
 107 andere Gerät #00:09:23#
- 108 I: Wie passiert dieses fragen? Wenn man da jetzt die Begriffe benutzt, die in der
 109 Objektorientierung eine Rolle spielen? #00:09:35#
- 110 S: (6) Also die brauchen ja irgendwas, was sie verbindet, sag ich jetzt mal. Wir
 111 hatten da ja so Fäden gespannt, so Beziehungslinien, und das, weiß ich nicht,
 112 muss halt in irgendeiner Verbindung stehen #00:10:07#
- 113 I: Sonst lass uns dass doch mal ganz konkret machen. Wenn du dir vorstellst
 114 (.) Oder vielleicht kannst du nochmal die Situation beschreiben mit dem
 115 Helligkeitssensor und dem Scheinwerfer, wie das da ablieft. Was hat das eine
 116 Objekt mit dem anderen gemacht? Du hast jetzt gesagt, das eine Objekt hat
 117 das andere gefragt. Wie sah das da genau aus? #00:10:22#
- 118 S: Ähm, ja also die Person, was ja bei dem Objektspiel Martin war, der hat ja die
 119 Hintergrundbeleuchtung angeschaltet mit der Antwort vom Helligkeitssensor.
 120 Und dann hat die Hintergrundbeleuchtung ja den Sensor befragt und wenn
 121 die Antwort halt *hell* war, dann ist es ausgeblieben, und wenn die Antwort
 122 *dunkel* war, dann ist es angegangen #00:11:03#
- 123 I: Ja, hat da jetzt die Hintergrundbeleuchtung den Helligkeitssensor gefragt oder
 124 wie hatten wir das am Anfang gemacht? #00:11:14#
- 125 S: Ne, also am Anfang hatten wir, dass die Person den Helligkeitssensor befragt
 126 und mit der Antwort, die er kriegt, die Beleuchtung schaltet #00:11:25#
- 127 I: Und die Situation, die du jetzt vorhin beschrieben hast, dass wirklich die
 128 Hintergrundbeleuchtung den Helligkeitssensor fragt, wo ist da der Unterschied?
 129 #00:11:33#
- 130 S: Dass wenn der Hellig— also wenn die Hintergrundbeleuchtung den Sensor
 131 befragt, dann braucht die Person halt keine Beziehung zum Helligkeitssensor
 132 und andersherum genauso #00:11:57#
- 133 I: Wenn du willst, kannst du dazu auch was aufmalen. Sonst mal eben auf, wie
 134 das mit dem Befragen da läuft #00:12:15#
- 135 S: /malt Abbildung B.1/ #00:12:22#

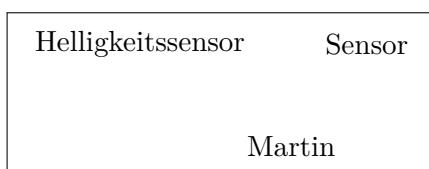


Abbildung B.1: Interview C – Aufschrieb.

- 136 S: Dann würde ja, wenn die Hintergrundbeleuchtung den Sensor befragen würde,
137 wäre hier ja keine Verbindungsline /deutet eine Verbindung von Martin
138 zum Sensor an/ und wenn Martin erst den Sensor befragen würde und mit
139 der Antwort die Hintergrundbeleuchtung schalten würde, wäre hier keine
140 Verbindungsline /deutet eine Verbindung von Hintergrundbeleuchtung zum
141 Helligkeitssensor an/, glaube ich #00:12:45#
- 142 I: Und wie ruft Martin die Hintergrundbeleuchtung auf? Du sagst ja, mit der
143 Antwort schalten, wie sieht da so ein Aufruf aus? #00:12:55#
- 144 S: Ja, es kommt halt darauf an, was der Sensor dann sagt #00:13:02#
- 145 I: Der sagt zum Beispiel *hell* #00:13:05#
- 146 S: Ja, dann, wenn es schon hell ist, dann muss die Hintergrundbeleuchtung gar
147 nicht mehr angehen und dann bleibt sie sozusagen aus, aber sie ändert trotzdem
148 ihren Status von *aus* auf *aus*, wenn sie vorher schon aus war #00:13:24#
- 149 I: Und wie gibst du diese Information *hell*, die der Sensor zurückgibt, an die
150 Hintergrundbeleuchtung weiter? #00:13:26#
- 151 S: Entweder über Martin oder halt direkt vom Sensor zur Beleuchtung #00:13:33#
- 152 I: Dann hast du gesagt, der ändert seinen Status, also die Hintergrundbeleuchtung
153 ändert seinen Status, auch wenn sie schon aus war. Wie kann man das vielleicht
154 irgendwie allgemein beschreiben, wie sich so ein Objekt verhält, wenn da eine
155 Methode drauf aufgerufen wird? #00:13:49#
- 156 S: Ja, wenn da eine Methode aufgerufen wird, verändert das immer seinen Status
157 oder halt, kommt halt auf die Methode drauf an, was es verändert, aber
158 irgendein Attributwert wird dann halt immer verändert. Und ja (1), deswegen
159 muss es dann auch ändern, auch wenn er aus bleibt #00:14:14#
- 160 I: Und vielleicht kannst du noch einmal kurz aufschreiben, wie so ein Metho-
161 denaufruf jetzt mit dieser Antwort *hell* jetzt bei der Hintergrundbeleuchtung
162 aussähe? Wir hatten das ja in dieser Groovy Konsole geschrieben. Kannst du
163 dich ungefähr daran erinnern, wie wir das aufgeschrieben hatten? #00:14:35#
- 164 S: Muss ich nicht erstmal erstellen, sag ich jetzt mal, dass der da ist? #00:14:42#
- 165 I: Dass was da ist? #00:14:43#
- 166 S: Dass der Computer weiß, dass es das Objekt gibt, und dann kann man halt
167 den Standort setzen oder so, muss man aber nicht, und dann kann man das
168 halt mit den Pins, wo man das eingetragen hat, schalten #00:15:11#
- 169 I: Wie fandest du das denn mit der Hardware, wie wir das gemacht haben? Hat
170 das für dich zum Verständnis beigetragen, was ein Objekt ist und wie das mit
171 den Methoden funktioniert? #00:15:19#
- 172 S: Joar, also es war halt dadurch, dass wir glaube ich fünf Scheinwerfer hatten,
173 war halt, dass man sehr oft Sachen neu eingeben musste und so, da musste
174 man halt immer oft das gleiche machen, aber eigentlich hat es schon zum
175 Verständnis geholfen #00:15:42#

- 176 I: Da hatten wir dann natürlich auch mehrere Objekte für die ganzen Scheinwerfer und für die Hintergrundbeleuchtung. Wie konntest du da unterscheiden zwischen den Objekten, welches Objekt du jetzt ansprechen willst oder auf welchem du jetzt eine Methode aufrufst? #00:15:58#
- 180 S: Man hat die ja benannt danach. Also entweder ein grüner Scheinwerfer oder ein blauer Scheinwerfer #00:16:08#
- 182 I: Okay, dann möchte ich gerne nochmal fragen. Wenn du jetzt eine Methode auf diesem grünen Scheinwerfer aufrufst, wie sieht da so ein Beispiel für aus? Also wie schreibt man das, wie schreibst du das auf? #00:16:21#
- 185 S: Also, je nach dem wie ich den grünen Scheinwerfer, wie ich den abgespeichert habe, schreibe ich dann *grüner Scheinwerfer, Doppelpunkt, anschalten* und dann dahinter zwei Klammern, damit der Computer erkennt, dass es eine Methode ist und in die Klammern kann dann halt noch, wenn das jetzt zum Beispiel *warten* wäre oder so, dann kann dahinter halt noch das Parameterwert dann, glaube ich, zum Beispiel wie lange der warten muss #00:16:54#
- 191 I: Mit diesem Parameterwert. Vielleicht kannst du es jetzt nochmal für diesen Sensor erklären, mit der Hintergrundbeleuchtung schalten. Das war vorhin die Frage mit der Antwort *hell*. Wie würde man da die Hintergrundbeleuchtung schalten mit der Antwort #00:17:09#
- 195 S: Ach so, da würde man dann der Hintergrundbeleuchtung sagen *anschalten*, und dann dahinter mit dem Parameterwert, entweder schreibt man dann da rein schon direkt *ist hell* oder man schreibt da halt rein *Antwort des Helligkeitssensors* oder halt das, wie man das definiert hat vorher #00:17:31#
- 199 I: War denn jetzt diese Arbeit, die wir jetzt ein paar Stunden gemacht haben mit der Hardware, wenn du da jetzt ein Fazit ziehen müsstest. Siehst du da irgendwo Vorteile? Oder siehst du irgendwo Nachteile? #00:17:56#
- 202 S: Ich glaub schon, dass es mehr Vorteile gibt, weil man halt dadurch selbst auch Sachen ausprobieren kann. Und da viele Leute die Sachen halt besser verstehen, wenn man sie selbst in der Praxis umsetzt, als wenn man sie nur in der Theorie durchnimmt. Und ich persönlich fand das Objektspiel mit den Karten immer sehr verwirrend und hab das nie so ganz verstanden. Aber wenn man das dann halt in diese Groovy Konsole eingegeben hat, wurde es für mich halt immer deutlicher und ich habe es halt besser verstanden. Und da glaub ich halt schon, dass diese (.) Praxisarbeit zum Verständnis beiträgt #00:18:44#
- 210 I: Okay, kannst du mir erklären, was du da verwirrend fandest bei dem Objektspiel? #00:18:49#
- 212 S: Also ich finde man musste ja immer so bestimmt reden... *Ich, Martin, rufe bei dem Objekt so und so die Methode so und so hervor*, und das muss man zwar auch genau so in den Computer eingegeben, aber ich fand das dann immer (.) ich hab das nie so ganz verstanden, warum man das jetzt genau so machen muss, weil ich (1) also ich weiß ja, dass ich dem Computer nicht sagen, ja dass ich nicht sagen kann, *jetzt schalte den grünen Scheinwerfer an*. Das versteht

218 der ja nicht. Ich weiß schon, dass man da bestimmte Befehle eingeben muss,
219 aber ich habe es halt im Objektspiel nicht so ganz verstanden, warum man
220 dann so diese Befehle so sagen musste #00:19:32#

221 I: Ja, interessant. Diese Sprechweise zu dem eigentlichen Quelltext, diese Verbin-
222 dung ist dir nicht so ganz klar geworden? #00:19:38#

223 S: Genau #00:19:39#

224 I: Ja super, ich glaube dann sind wir schon fertig, ja #00:19:51#

B.6 Interview D

| Interview D | |
|-----------------|-------------------|
| Datum / Uhrzeit | 15. Dezember 2016 |
| Dauer | 13:43 (mm:ss) |
| Interviewer | Heiner Stroick |
| Schüler/in | Schüler D |

Tabelle B.6: Informationen zu Interview D.

Transkript Interview D

- 1 I: Vielleicht stellst du dich einmal kurz vor mit deinem Namen, und dann schreib
2 ich später Schüler A, Schüler B oder so #00:00:05#
- 3 S: Okay, ich bin Schüler D #00:00:07#
- 4 I: Alles klar, vielleicht kannst du mir am Anfang erzählen, welchen Kontakt du
5 bisher schon zur Informatik hattest? Und welche Vorerfahrungen du vielleicht
6 schon sammeln konntest? #00:00:16#
- 7 S: Ich hatte eigentlich gar keinen Kontakt vorher mit Informatik, ich habe mich
8 nur dafür interessiert, weil ich dachte, das ist so mit Computern halt und die
9 ein bisschen besser verstehen und so, da hatte ich Lust drauf, und dann dachte
10 ich, ich probiere es mal aus. Aber, naja, es hat sich rausgestellt, wir arbeiten
11 mit Objektkarten und sowas, das finde ich jetzt nicht so geil #00:00:36#
- 12 I: Was hast du dir unter Informatik vorgestellt? #00:00:37#
- 13 S: Ähm, ja, so, programmieren halt irgendwie sowas und am Computer arbeiten
14 und Programme vielleicht entwickeln, weil ich glaube mein Bruder hatte das
15 auch, und der hat eigentlich Programme geschrieben und sowas. Also mir ist
16 klar, dass man das nicht direkt lernt oder direkt machen kann, aber, naja
17 #00:00:58#
- 18 I: Und sonst privat Kontakt? #00:01:02#
- 19 S: Ne, eigentlich nicht #00:01:02#
- 20 I: Also irgendwas am Computer vorher schon gemacht? #00:01:03# #00:01:03#
- 21 S: Also ich arbeite schon seit vielen Jahren mit dem Computer, aber ich mach
22 da nicht wirklich etwas informatisches mit, glaube ich #00:01:10#
- 23 I: Jetzt sagst du gerade, wir machen jetzt Objektkarten und so, und das ist ja jetzt
24 auch das Thema, Thema Objektorientierung. Wir haben da ja verschiedene
25 Begriffe kennengelernt, vielleicht kannst du da erzählen, was dir da spontan
26 zu einfällt? #00:01:25#

- 27 S: Zu Objektkarten? #00:01:26#
- 28 I: Ne, zu den Begriffen der Objektorientierung. Was kommt dir da als erstes so
29 in den Sinn und wie würdest du das erklären? #00:01:33#
- 30 S: Also erstmal dieses, dass man eine Geschichte, glaube ich, so mit dem Computer
31 verwirklicht, irgendwie. Dass man diese Befehle aufschreibt und die dann
32 ausgeführt werden vom Computer #00:01:48#
- 33 I: Kannst du den Prozess ein bisschen genauer erklären? #00:01:51#
- 34 S: Ja, man muss aus der Geschichte irgendwie die Objekte, die Methoden und die
35 Attributwerte oder so herausfinden (.) und das dann alles irgendwie zuordnen
36 und am Computer halt eintippen /lacht/ #00:02:05#
- 37 I: Wie findest du die Objekte? #00:02:06#
- 38 S: Ja ich gucke mir die ganzen Substantive an. Alles, was man anfassen kann, ist
39 ein Objekt #00:02:16#
- 40 I: Kannst du ein Beispiel dafür geben? Und erklären, warum das für dich ein
41 Objekt ist? #00:02:20#
- 42 S: Scheinwerfer, zum Beispiel #00:02:24#
- 43 I: Ja, und warum? #00:02:25#
- 44 S: Ja, weil gerade in der Geschichte, die wir auch im Unterricht hatten, konnte
45 man ja damit die Sachen durchführen, zum Beispiel anschalten, ausschalten
46 und das war ja auch wichtig für die Geschichte und von daher ist das Objekt
47 auch wichtig und das brauchte man dann. #00:02:40#
- 48 I: Ja, okay, kannst du dir ein anderes Objekt vorstellen, jetzt unabhängig von
49 der Geschichte? #00:02:47#
- 50 S: (2) Ich denke mal alles, womit interagiert wird irgendwie. Also es würde jetzt
51 nicht Sinn machen, wenn man irgendwie Tisch sagt oder so, wenn damit nichts
52 gemacht wird, oder wenn man jetzt eine Geschichte hätte, wo generell Objekte
53 bewegt werden, das sind dann alles Objekte, schätze ich, irgendwie (1) das
54 kann ja eigentlich alles sein, was man halt anfassen kann, außer sowas wie (.)
55 irgendwelche Begriffe, die man eben, ja, wie heißen die nochmal die Begriffe?
56 Abstrakte Begriffe kann man das...? Das sind ja keine Objekte dann #00:03:19#
- 57 I: Kannst du das mal richtig konkret machen, dass wir darüber sprechen? Denk
58 dir mal was aus, wenn du sagst, alles, was man anfassen kann (.) #00:03:31#
- 59 S: Ja, keine Ahnung, Apfel, Tisch, Stuhl, sowas alles kann ein Objekt sein und
60 Freundschaft oder so halt nicht #00:03:40#
- 61 I: Ja, und wie würdest mit so einem Apfel oder so einem Tisch interagieren?
62 #00:03:42#
- 63 S: Ich kann sagen, *Martin (.) setzt (.) Apfel (.) auf (.) keine Ahnung (.) /lacht/*
64 irgendwohin #00:03:53#

- 65 I: (2) Diese Interaktion, die ist ja wichtig, haben wir ja auch in der Geschichte
66 gesehen. Vielleicht kannst du erklären, wie so Objekte miteinander interagieren?
67 Wie das funktioniert? #00:04:13#
- 68 S: Also zwei Objekte miteinander? #00:04:18#
- 69 I: Es geht ja um den Informationsaustausch, weil wenn du jetzt eine größere
70 Geschichte hast, sag ich mal, dann hast du ja denke ich mal nicht unbedingt
71 nur ein Objekt, sondern mehrere, [ja] und die sind alle für verschiedene Be-
72 reiche zuständig, wenn man das vielleicht so unterteilen möchte, und, ja, wie
73 kommt es da jetzt zu Interaktion zwischen verschiedenen Bereichen oder (1)
74 zwischen verschiedenen (2) Abschnitten, (.) oder einfach zwischen verschiedenen
75 Objekten? #00:04:53#
- 76 S: Ja, die Interaktion passiert ja, indem eine Methode aufgerufen wird von einem
77 Objekt zum anderen. Und wir hatten ja in der letzten Stunde auch diese
78 Beziehungspfeile, dass die sich alle erstmal kennen müssen, und dann werden ja
79 Methoden aufgerufen, und dann werden die halt ausgeführt vom bestimmten
80 Objekt #00:05:11#
- 81 I: Kannst du dafür mal ein Beispiel geben? #00:05:14#
- 82 S: (2) Zum Beispiel jetzt aus der Geschichte wieder, wenn der Scheinwerfer halt
83 angemacht wird. Dass man dann die Methode *anschalten* aufruft und der
84 Scheinwerfer führt die dann aus, das Licht ist an, und es wird wieder zurück
85 gegeben #00:05:31#
- 86 I: Was passiert intern beim Scheinwerfer, wenn die Methode aufgerufen wird?
87 #00:05:33#
- 88 S: Ja der Status ändert sich auf *an* #00:05:36#
- 89 I: Okay, das ist ja jetzt so eine einfache Methode, wo man jetzt sagt, schalte
90 dich an, schalte dich aus. Da gibt es ja jetzt auch ein bisschen komplizierte
91 Methoden... #00:05:44#
- 92 S: Ja, zum Beispiel hatten wir bei dem RGB Scheinwerfer dieses *Licht mischen*
93 und so, dass man noch mit den Parametern arbeitet und sagt, dass grün 70
94 Prozent, rot 30 und so weiter #00:05:55#
- 95 I: Okay, kannst du dir vorstellen, was da intern passiert oder ist das nicht groß
96 unterschiedlich zum normalen Scheinwerfer? #00:06:03#
- 97 S: Ja, ich denke das Prinzip ist dasselbe. Es wird auch halt Licht angemacht, nur
98 halt in einer bestimmten (1) Menge, sozusagen. Also bestimmter Lichtanteil
99 #00:06:15#
- 100 I: Dann gibt es ja noch eine andere Klasse von Methoden, fällt dir da spontan
101 was zu ein? Ich will die jetzt nicht nennen... #00:06:23#
- 102 S: Also außer *anschalten*, *ausschalten*, *blinken* und sowas? #00:06:24#
- 103 I: Ja, genau #00:06:25#

- 104 S: *warten*, zum Beispiel #00:06:31#
- 105 I: Ja, das sind alles Methoden, wo du einen Parameterwert für halt einen Para-
106 meter, zum Beispiel diese Zeiteinheit, die du warten willst, mitgibst, aber (.)
107 wir haben auch noch andere Methoden kennengelernt #00:06:46#
- 108 S: Ja /lacht/ #00:06:49#
- 109 I: Stichwort Fragen #00:06:50#
- 110 S: Ach ja, befragen! #00:06:53#
- 111 I: Ja, was waren das für besondere Methoden? #00:06:53#
- 112 S: Dass man halt... Dass ein Objekt gefragt wird und das eine Antwort zurück
113 gibt. Und also, es in einem Binärkode oder so antwortet, dass man nur *ja* oder
114 *nein* oder *Null* oder *Eins* zurück bekommt #00:07:12#
- 115 I: Ja, kannst du dir dafür mal ein anderes Beispiel aus der echten Welt, also jetzt
116 unabhängig von diesen Bausteinen irgendwie überlegen, wo sowas auftritt?
117 #00:07:23#
- 118 S: (2) Also jetzt unabhängig von so Informatik oder #00:07:27#
- 119 I: Ne, ne, also von mir aus gerne im Informatik-Kontext, aber... #00:07:32#
- 120 S: Da würde mir jetzt halt Kommunikation halt einfallen, das ist ja alles das
121 #00:07:39#
- 122 I: Ja, nenne mal zwei Objekte und wie würdest du die Methode benennen? #00:07:38#
- 123 S: Ähm (5) wenn man... (1) also das erste Objekt ist eine Person, die was kauft
124 und die fragt das andere Objekt, *wie viel* und der gibt zurück, irgendeinen
125 Wert, also wie es halt kostet #00:08:01#
- 126 I: Okay, und die Methode würde heißen? Wie würdest du die nennen? #00:08:06#
- 127 S: *Kostet wie viel* (.) Klammer auf, Klammer zu? #00:08:09#
- 128 I: Und dann gibt der zurück: *2 Euro*, oder so? [Ja] Ah okay, so stellst du dir das
129 also vor. Ja jetzt hast du schon so ein bisschen darüber gesprochen, wie sich
130 so ein Objekt verhält, wenn eine Methode darauf aufgerufen wird, vielleicht
131 sprechen wir da noch einmal darüber. (1) Wie hängt dieses Verhalten von dem
132 Parameter ab, den ich da beim Methodenaufruf mitgebe? #00:08:35#
- 133 S: (2) Wenn ich jetzt einen Parameter bei der Methode einsetzte? #00:08:42#
- 134 I: Genau, ja #00:08:44#
- 135 S: (1) Die Methode wird dann irgendwie näher bestimmt. Also, weiß ich nicht
136 #00:08:54#

- 137 I: Also, wenn ich dich jetzt (.) also, wenn wir beim Beispiel bleiben mit diesen
138 zwei Objekten, diesen zwei Personen, diesen zwei Personenobjekten, und das eine
139 Objekt fragt das andere, was kostet das? Der sagt *2 Euro*. Oder der sagt *200*
140 *Euro*. Was passiert dann mit vielleicht mit der Antwort? #00:09:16#
- 141 S: Ja bei dem Beispiel kann das eine Objekt mit der Antwort ja nicht viel machen,
142 das will ja nur die Information haben, und bei anderen Objekten könnte das so
143 sein, dass wenn der die Antwort kriegt, dass dann irgend eine andere Methode
144 ausgeführt wird, aber das ist ja in dem Beispiel nicht der Fall #00:09:35#
- 145 I: Ja, das hatten wir auch in der Geschichte gesehen #00:09:38#
- 146 S: Ja, das (1) war dann dieses bedingte Anschalten von der Hintergrundbeleuchtung
147 und da hing, da hängt das ja dann von der Antwort ab, ob es an geht
148 oder nicht und das ist ja bei diesem Beispiel, was ich gerade genannt habe,
149 nicht der Fall #00:09:50#
- 150 I: Genau. Willst du dazu was aufschreiben? (1) Sonst lass uns noch einmal ganz
151 kurz darüber sprechen. Du hast vorhin gesagt, die müssen sich kennen, kannst
152 du dazu noch was erzählen? Was verstehst du darunter, wenn die sich kennen
153 müssen? #00:10:06#
- 154 S: (2) Ja, wenn ein Objekt ein anderes befragt oder eine Methode daran ausführt,
155 dann muss das Objekt das andere ja kennen, um das halt zu machen und die
156 Methoden auch zu kennen #00:10:23#
- 157 I: (2) Okay, jetzt haben wir in der Geschichte ja, oder in der Groovy Konsole ja
158 immer so geschrieben *roter Scheinwerfer Punkt anschalten, grüner Scheinwerfer*
159 *Punkt ausschalten* #00:10:43#
- 160 S: Ja, und vorher hatten wir das ja (1) mit *new Lampe* oder so eingetragen,
161 also *new Scheinwerfer Gleich irgendwas* und dann hat der das ja gekannt, der
162 Computer /lacht/ #00:10:56#
- 163 I: Okay, ja, warum haben wir das gemacht? #00:10:58#
- 164 S: Ja, damit der Computer die Objekte halt kennt. Weil die Methoden oder so
165 waren ja schon eingespeichert #00:11:08#
- 166 I: Und wenn jetzt so wirklich zwei Objekte, wenn du jetzt wirklich zwei Objekte
167 hast, die miteinander kommunizieren wollen, und das eine Objekt will bei dem
168 anderen etwas aufrufen. Wie stellst du dir da vor, was da im Hintergrund
169 passiert? #00:11:24#
- 170 S: Was da im Hintergrund passiert? Im Computer dann oder wie? #00:11:30#
- 171 I: Ne, ich mein jetzt bei den Objekten selbst #00:11:32#
- 172 S: Ach so, (2) also die müssen halt erstmal das irgendwie empfangen das Signal,
173 also, die Methode, die daran ausgeführt wird, das müssen die irgendwie empfange
174 n schätzen ich, und dann verarbeiten und dann durchführen. Ich weiß auch
175 nicht, wie das dann im Einzelnen passiert, aber, ja (2) Wir haben darüber

- 176 ja auch nicht wirklich gesprochen, wir haben nur gesagt, Methoden werden
177 durchgeführt und es wird zurück gegeben und... fertig. Und Beziehungen
178 #00:12:07#
- 179 I: Wie würdest du insgesamt die Arbeit damit bewerten, die wir da jetzt durch-
180 geführt haben mit der Geschichte und diesen Bausteinen #00:12:16#
- 181 S: Also als es dann zum Raspberry Pi kam fand ich es dann schon interessant
182 und ja, aber, ja doch, war schon ganz cool /lacht/ #00:12:26#
- 183 I: Ging das dann mehr in die Richtung Informatik, die du dir am Anfang so
184 vorgestellt hast? #00:12:28#
- 185 S: Ja, ja, also ich kannte nur irgendwie diesen, diese Groovy Konsole, wo man so
186 Befehle eintippt und dann passiert halt irgendwas und das haben wir dann
187 halt gemacht #00:12:38#
- 188 I: Das kanntest du schon vorher oder? #00:12:40#
- 189 S: Ja, ich wusste nicht, was man da alles so machen kann, aber ich wusste, dass
190 das irgendwie geht, ja #00:12:47#
- 191 I: Würdest du sagen, das war hilfreich für das Verständnis? #00:12:47#
- 192 S: Ja, jetzt kann mir auf jeden Fall besser vorstellen, was man so, was Informatik
193 so am Computer heißt, was man da so macht #00:12:57#
- 194 I: Und fürs Verständnis der Objektorientierung? So insgesamt? #00:13:00#
- 195 S: Ja, geht. Das was wir gemacht haben, habe ich jetzt so halbwegs verstanden,
196 denke ich. (5) #00:13:15#
- 197 I: Okay, möchtest du noch einmal irgendetwas erklären, was, (1) wo du sagst, da
198 fällt mir noch was zu ein? #00:13:20#
- 199 S: (5) Ne, eigentlich nicht #00:13:38#
- 200 I: Ja gut, dann sind wir fertig. Dann danke ich dir #00:13:43#

B.7 Interview E

| Interview E | |
|-----------------|-------------------|
| Datum / Uhrzeit | 14. Dezember 2016 |
| Dauer | 10:54 (mm:ss) |
| Interviewer | Heiner Stroick |
| Schüler/in | Schüler E |

Tabelle B.7: Informationen zu Interview E.

Transkript Interview E

- 1 I: Fangen wir an, würde ich direkt sagen. Es wäre ganz nett, wenn du einmal
 2 deinen Namen sagst, damit ich das einmal in der Aufnahme hab, dann schwärze
 3 ich das nachher #00:00:11#
- 4 S: Schüler E #00:00:11#
- 5 I: Ich würde gerne wissen zu Beginn, welche Vorerfahrungen du schon mit der
 6 Informatik hast? #00:00:18#
- 7 S: Ja, also ich hab relativ früh angefangen, mich dafür zu interessieren. So
 8 bestimmt mit sieben, acht habe ich dann irgendwann angefangen eben mit
 9 dem Rechner von meinen Eltern damals so ein bisschen rum zu spielen, alles
 10 auszuprobieren und dann irgendwann hab ich mir ein Buch ausgeliehen, so ein
 11 Q-Basic-Buch, da hab ich dann so ganz, aber auch nur wirklich so vier, fünf
 12 Befehle gelernt, da war ich höchstens acht oder neun, und wirklich angefangen
 13 mit Java, also da hab ich jetzt auch schon wieder das meiste verlernt, aber
 14 wirklich angefangen habe ich dann...da haben wir Minecraft gespielt und
 15 da haben wir Plug-Ins angefangen zu programmieren dafür. Für Server so
 16 eigene Game-Modi und so, da hab ich dann wirklich damit angefangen und
 17 ja, ich weiß nicht, ich interessiere mich dafür, auch für Hardware, ich habe
 18 auch meinen PC zum Beispiel selber zusammen gestellt und so, und ja, also
 19 (1) #00:01:10#
- 20 I: Nur aus Interesse: Worin programmierst du das für Minecraft? #00:01:11#
- 21 S: In Eclipse, also das normale Java. Da gibt es eine API für #00:01:18#
- 22 I: Ah, okay. Jetzt haben wir ja auch im Unterricht mit Java gearbeitet bzw.
 23 in Groovy und uns das Thema Objektorientierung angeguckt und vielleicht
 24 kannst du da ein paar Begriffe erklären? Erst mal so ganz grob gefragt, was
 25 das ausmacht, Objektorientierung? #00:01:35# #00:01:37#
- 26 S: Objektorientierung ist ja, dass man sich vorgibt, also ich weiß es gar nicht
 27 genau, um ehrlich zu sein, aber man hat ja auf jeden Fall schon mal eine
 28 Schnittstelle sozusagen, so würde ich das jetzt nennen, auf die man sozusagen

29 drauf programmiert, und vorgegeben eben Objekte, die man mit Methoden
30 ansprechen kann und man hat eben (2) ich weiß nicht (2). Man schreibt ja
31 irgendwie was, was sich direkt auf was bezieht, auf verschiedene wirkliche
32 Sachen #00:02:17#

33 I: Kannst du ein Beispiel für ein Objekt nennen? #00:02:17#

34 S: Ja, zum Beispiel so eine Lampe oder das kann ja auch irgendwas anderes sein.
35 Keine Ahnung, in dem Spiel, zum Beispiel der Spieler kann ja auch ein Objekt
36 sein #00:02:27#

37 I: Und warum ist das für ich ein Objekt? #00:02:27#

38 S: Wir haben ja diese Methode gehabt und da hat man ja eben die Nomen zu
39 Objekten gemacht bis auf verschiedene Wörter wie zum Beispiel irgendwelche
40 Mengenangaben oder sowas #00:02:45#

41 I: Ja, und abseits von diesem Algorithmus, den wir da kennengelernt haben.
42 Warum ist Spieler für dich ein Objekt? #00:02:50#

43 S: Spieler, ja, (10). Ich weiß nicht, also Spieler wäre jetzt für mich das Objekt, weil
44 ich da jetzt irgendwelche Methoden dran ausführen würde sozusagen #00:03:13#

45 I: Kannst du vielleicht auch noch ein anderes Beispiel nennen, was vielleicht ein
46 bisschen weniger abstrakt ist? #00:03:17#

47 S: Ja, also zum Beispiel der rote Scheinwerfer oder sowas #00:03:17#

48 I: Was für Methoden könnte man da nehmen für den? #00:03:23#

49 S: Ja man könnte den anschalten, man könnte den ausschalten, vielleicht noch
50 *blinken* als Methode #00:03:31#

51 I: Wie können denn Objekte untereinander Informationen austauschen? Das ist
52 ja eigentlich auch eine Sache, die dann immer von Interesse ist, dass du nicht
53 nur ein Objekt hast, sondern mehrere, die müssen dann ja, wenn du jetzt
54 eine komplexere Problemstellung hast, Informationen untereinander irgendwie
55 austauschen. Wie funktioniert das? #00:03:48#

56 S: Wir haben ja jetzt diese Beziehung gehabt, dass die einander kennen und dann
57 können die ja über die Methode sozusagen immer wieder was zurück geben
58 noch #00:03:58#

59 I: Kannst du es ein bisschen konkreter machen? #00:04:00#

60 S: Man kann ja die Methode auch mit Parameter ausführen. Und über Variablen
61 natürlich #00:04:05#

62 I: Hast du ein Beispiel, wo du das mal so durchspielen kannst? #00:04:09#

63 S: Ja zum Beispiel wenn man jetzt den Helligkeitssensor befragt und dann die
64 Antwort, hat man eine Variable, und setzt eben die Variable auf den Wert, der
65 die Antwort ist von der Methode, die man beim Helligkeitssensor ausgeführt

66 hat und kann dann mit der Variable oder dem Wert, den man gespeichert hat,
67 eben die Hintergrundbeleuchtung natürlich schalten #00:04:29#

68 I: Und wie verhält sich jetzt so ein Objekt, wenn darauf eine Methode aufgerufen
69 wird? #00:04:31#

70 S: Ja das Objekt führt dann eben die Methode aus und gibt eben wieder zurück
71 zu dem, dem (.) ja, einfach zurück #00:04:42#

72 I: Was passiert dabei, wenn du sagst, die Methode wird aufgerufen. Was läuft
73 da sozusagen in Anführungsstrichen intern ab? #00:04:49#

74 S: Ja also, jetzt auf das Beispiel bezogen was wir hatten, oder? #00:04:53#

75 I: Ja, von mir aus #00:04:53#

76 S: Ja da würde dann ja wahrscheinlich, wahrscheinlich würde eben der Raspberry
77 Pi die jeweiligen Pins ansteuern auf seinem Steckbrett und da eben Strom
78 drauf geben auf die Lampe, wenn man die anschaltet #00:05:10#

79 I: Ja, und wenn wir uns jetzt ein bisschen anderes Beispiel überlegen? Wenn du
80 so aus dem Fenster guckst, wo siehst du da, oder was wäre da für dich ein
81 Objekt? #00:05:18#

82 S: Zum Beispiel das Auto oder die Straße oder sowas, oder so eine Straßenlaterne
83 /lacht/ Das wäre jetzt wieder... #00:05:29#

84 I: Das wäre jetzt wieder die Lampe... Ich will so ein bisschen von der Lampe weg.
85 Dass wir uns vielleicht mal etwas aus der realen Welt als Objekt vorstellen.
86 Jetzt hast du Auto gesagt, was wären so Methoden von so einem Auto? #00:05:39#

87 S: Ja das könnte natürlich vorwärts fahren, das könnte rückwärts fahren, das
88 könne meinewegen hupen, vielleicht noch das Licht einschalten, die Blinker
89 einschalten, bremsen natürlich, lenken natürlich auch #00:05:52#

90 I: Ja, das sind ja jetzt Methoden, und eine Sache neben den Methoden interessiert
91 ja auch noch bei Objekten #00:05:58#

92 S: Die Attributwerte #00:06:02#

93 I: Genau, was für Attribute würdest du da haben? #00:06:09#

94 S: Ja man könnte zum Beispiel die Farbe haben, oder irgendwelche Zustände
95 von verschiedenen Teilen des Autos, zum Beispiel ob jetzt gerade irgendwelche
96 Blinker aktiviert sind oder sowas, oder den Standort des Autos #00:06:27#

97 I: Ja, okay. Und kannst du mal ein Beispiel für einen Attributwert nennen? Oder
98 für ein Attribut und ein Attributwert vom Auto? #00:06:35#

99 S: Zum Beispiel die Farbe und die könnte *blau* sein #00:06:38#

100 I: Okay, jetzt hast du vorhin gesagt Standort. Und wenn das Auto jetzt fährt,
101 was würde dann mit dem Attribut passieren? #00:06:45#

- 102 S: Ja, das würde wahrscheinlich immer wieder geupdated werden #00:06:49#
- 103 I: Dann haben wir jetzt im Unterricht mit dieser Groovy Konsole gearbeitet und
104 uns da Objekte erstellt und Methoden darauf aufgerufen. Wie würdest du die
105 Arbeit insgesamt bewerten? Oder auch mit dieser Hardware, mit der wir da
106 gearbeitet haben? #00:07:07#
- 107 S: Ich finde das hat auch wirklich Sinn gemacht, vor allen Dingen ich hab ja jetzt
108 auch schon ein bisschen was an Programmiererfahrung, aber ich glaube, wenn
109 man da keine hat, dann war das auf jeden Fall sinnvoll, so einzusteigen, weil
110 wirklich, also auch den Pi hat man ja was, was man wirklich anfassen kann,
111 angucken, man sieht vielleicht so ein bisschen sogar, was da passiert, und auch
112 durch die Methoden, dass die dann was ausführen, dass dann wirklich was
113 passiert, also das fand ich auf jeden Fall, ich fand das gut. Auch dass man
114 nicht auf so viel achten musste, auf irgendwelche Klammern oder so und dabei
115 wirklich jetzt, ich fand das schön, dass das jetzt so vereinfacht war jetzt #00:07:39#
- 116 I: Ja, die Groovy Konsole ist ja da nicht so streng mit den Klammern, sonst
117 muss man da ja sehr aufpassen beim Programmieren #00:07:43#
- 118 S: Jaja, klar #00:07:43#
- 119 I: Das fandest du gut, dass die da so ein bisschen lascher war? #00:07:47#
- 120 S: Ja, das war ja ganz gut, also ich glaube vor allen Dingen, wenn man halt eben
121 das zum ersten Mal macht #00:07:50#
- 122 I: Weswegen? #00:07:56#
- 123 S: Damit man erstmal den Grund sozusagen, den Hintergrund könnte man auch–,
124 die Basis verstehen kann, und sich dann später was komplexeres erarbeiten
125 kann #00:08:03#
- 126 I: Also würdest du das sozusagen gerne weiter so machen? Damit arbeiten?
127 #00:08:09#
- 128 S: Ja, die Möglichkeiten sind natürlich beschränkt, aber, also ich halte zumindest
129 für den Einstieg als sehr sinnvoll #00:08:13#
- 130 I: Jetzt hast du gesagt, dass man diese Grundidee, dieses Grundkonzept erstmal
131 verstehen muss, vielleicht kannst du jetzt noch mal erklären, was so das
132 Grundkonzept von dieser Objektorientierung ist? Da wusstest du ja vorhin am
133 Anfang nicht so recht, wie du es ausdrücken solltest #00:08:33#
- 134 S: Ja, (4), dass man immer sich direkt auf andere Objekte bezieht und nicht nur,
135 also noch andere Sachen sozusagen steuert, außer dass man nur irgendwelche
136 Eingaben bekommt und die verarbeitet und wieder ausgibt. Also naja, ich
137 weiß es tatsächlich gar nicht genau #00:08:58#
- 138 I: Worüber wir jetzt noch nicht so wirklich gesprochen haben sind Parameter,
139 oder Rückgabewerte, wie das jetzt beim Methodenaufruf funktioniert. Vielleicht
140 kannst du da nochmal was– #00:09:20#

- 141 S: Ja da kann man ja in die Klammer hinter der Methode, die ja immer hingehört,
142 kann man ja noch einen Parameter mitgeben, und das Objekt eben, das ist
143 dann ja in der Schnittstelle, in der API, hinterlegt und das weiß dann was mit
144 dem Wert anzufangen und führt die Methode eben mit verschiedenen Werten
145 dann eben anders aus #00:09:37#
- 146 I: Ja und jetzt auf das Auto bezogen. Was wäre da eine Methode mit einem
147 Parameter? #00:09:43#
- 148 S: Wenn man jetzt zum Beispiel sagt *fahre vorwärts*, und dann sagt man *20*
149 *Meter* oder wie auch immer, kommt jetzt auf die Maßeinheiten an, dann würde
150 das Auto nur 20 Meter vorwärts fahren, wenn da ja jetzt *100* eingibt, würde
151 das eben 100 Meter vorwärts fahren #00:09:55#
- 152 I: Und wenn du dir eine Methode vorstellst, die auch wirklich einen Rückgabewert
153 hat? #00:10:01#
- 154 S: Zum Beispiel, wenn man jetzt, ich weiß nicht, wenn das Auto jetzt einen
155 Regensensor hat, dann fragt man den Regensensor ob es regnet, und dann
156 könnte der eben den Wert in einer Variable abspeichern, wenn es zum Beispiel
157 regnet könnte man mit dem Wert, ob der jetzt *ja* oder *nein* ist oder eben *true*
158 oder *false*, damit dann den Scheibenwischer anschalten zum Beispiel #00:10:25#
- 159 I: Okay und wenn du jetzt den Scheibenwischer einschalten solltest, was brauchst
160 du dann vom Scheibenwischer? Also wie funktioniert das genau? Du kannst ja
161 jetzt nicht einfach sagen *Scheibenwischer wische*, oder so (1) #00:10:39#
- 162 S: Ja, man müsste dem ja, also die Methode *wische*, die müsste der haben, und
163 dann müsste man die Methode mit dem Parameterwert aufrufen, den man
164 vom Regensensor hat #00:10:47#
- 165 I: Ich glaub dann haben wir es. Oder möchtest du noch etwas loswerden? #00:10:53#
- 166 S: Ne #00:10:55#
- 167 I: Ne, alles klar. Super #00:10:54#

B.8 Interview F

| Interview F | |
|-----------------|-------------------|
| Datum / Uhrzeit | 15. Dezember 2016 |
| Dauer | 14:51 (mm:ss) |
| Interviewer | Heiner Stroick |
| Schüler/in | Schüler F |

Tabelle B.8: Informationen zu Interview F.

Transkript Interview F

- 1 I: Okay, vielleicht stellst du dich einmal kurz vor, dass ich deinen Namen habe
 2 und ich schreibe dann später Schüler A oder Schüler B, also wie gesagt, das
 3 wird alles anonymisiert #00:00:30#
- 4 S: Ich bin Schüler F, oder was? #00:00:30#
- 5 I: Genau, das reicht mir schon. Dann kann ich nämlich später Schüler A oder so
 6 schreiben, und dann kann man dir das nicht mehr zuordnen. Vielleicht erzählst
 7 du mal zu Beginn, welchen Kontakt du bisher schon zur Informatik hattest?
 8 #00:00:30#
- 9 S: Eigentlich nicht ganz so viel. Mein Vater arbeitet in die Richtung und ich
 10 wollte später vielleicht auch ein bisschen was mit Computern machen, joar,
 11 aber sonst... #00:00:42#
- 12 I: Privat schon mal irgendetwas gemacht? #00:00:46#
- 13 S: Nein, eigentlich nicht. Ich hab nur meinem Vater manchmal über die Schulter
 14 geschaut #00:00:52#
- 15 I: Ja, und sonst hier in der Schule auch noch nicht in der Mittelstufe oder so?
 16 #00:00:56#
- 17 S: Ne, ich hab das nicht angewählt gehabt, erst jetzt #00:00:58#
- 18 I: Also hast du vorher schon Programmiererfahrungen gesammelt oder eher nicht?
 19 #00:01:06#
- 20 S: Eher nicht #00:01:06#
- 21 I: Okay, jetzt haben wir ja über diese Objektorientierung gesprochen im Unter-
 22 rich, jetzt neu angefangen. Was fällt dir da spontan zu ein, und wie würdest
 23 du vielleicht die Begriffe, die dir da einfallen, erklären? Wenn ich jetzt mal so
 24 ganz grob frage? #00:01:23#
- 25 S: Zur Objektorientierung fällt mir erstmal im Wesentlichen Objektkarten ein
 26 [ja] und die Beziehungen zwischen all den Objekten und was die Objekte alles
 27 überhaupt machen, das man in die Objektkarte schreibt (4) joar #00:01:44#

- 28 I: Was wäre so ein Objekt für dich? Kannst du ein Beispiel dafür nennen? #00:01:47#
- 29 S: Ja, wir hatten das ja jetzt ganz oft im Unterricht, so Scheinwerfer oder sowsas
30 #00:01:50#
- 31 I: Und warum wäre das ein Objekt? #00:01:53#
- 32 S: Ja, weil das eben ein Gegenstand ist, der unterschiedliche Sachen ausführen
33 muss und ja, das gesagt bekommt und auch wieder zurück gibt #00:02:05#
- 34 I: Hättest du ein anderes Beispiel noch? #00:02:07#
- 35 S: (1) Ja (1) Menschen zum Beispiel, wie dieser Martin da jetzt (5) Eventuell so
36 die Bühne hatten wir jetzt auch als Objekt, aber das war ja eher ein Standort
37 #00:02:26#
- 38 I: Okay, und vielleicht unabhängig von der Geschichte? So aus dem realen Leben?
39 (1) Aus dem echten Leben? #00:02:31#
- 40 S: (3) Entweder so Menschen untereinander könnte man auch als Objekte bezeichnen
41 oder recht viele elektronische Gegenstände natürlich, aber auch generell,
42 wenn man sich so einen ganz normalen Gegenstand nimmt und wieder zurück
43 stellt oder so, dann ist das ja auch so eine Tätigkeit, die man ausgeführt hat
44 #00:02:58#
- 45 I: Nenn mal ruhig ein Beispiel und dann nenn mal, oder sag mal, was könnte
46 dieser Gegenstand, du hast vorhin elektronische Geräte gesagt, damit geht das
47 glaube ich ganz gut #00:03:09#
- 48 S: Ja, wenn du jetzt an den Computer gehst und den erstmal anschaltest. Dann
49 muss der Computer sich anschalten und dann hochfahren und dann muss der
50 ja irgendwie fragen, dass du das Passwort eingibst und so weiter #00:03:26#
- 51 I: Was könnte der noch? Also du kannst den anschalten... #00:03:29#
- 52 S: Ja, du kannst den natürlich wieder ausschalten und es gibt natürlich unterschiedliche Funktionen. Du kannst Internet aufrufen und dann innerhalb des
53 Internets noch andere Sachen aufrufen, das kann ja (.) sehr weit gehen #00:03:41#
- 55 I: Okay, und was wären so Attribute von so einem Computer? Das haben wir ja
56 auch auf den Objektkarten aufgeschrieben, einerseits was das kann, andererseits
57 die Attribute? #00:03:56#
- 58 S: (9) Fällt mir spontan nicht ein, mir würde jetzt nur der Wecker einfallen, so
59 quasi so, aber der ist ja nicht unbedingt auf dem Computer, so dass du dem
60 eine bestimmte Zeit gibst, wann er sich dann wieder melden muss, oder auch,
61 wenn er sich selbst wieder ausschaltet, weil der zu lange nicht benutzt wurde,
62 da ist ja auch so ein Attributwert, so ein Parameter, nach einer bestimmten
63 Zeit schaltet er sich dann wieder aus #00:04:41#
- 64 I: Ok, also da gibt es jetzt kein richtig und kein falsch und man könnte das
65 wahrscheinlich auf hunderttausend verschiedene Arten und Weisen machen,

66 wenn dir gleich noch was einfällt, kannst du ja noch was nennen. Jetzt hast du
67 in so einem Programm natürlich nicht immer nur ein Objekt, sondern mehrere,
68 und die müssen ja auch irgendwie miteinander kommunizieren, das hatten wir
69 ja auch im Unterricht gesehen. Vielleicht kannst du da was zu sagen, wie das
70 funktioniert hat? #00:05:05#

71 S: Das mit den Beziehungen was wir jetzt hatten, oder? #00:05:08#

72 I: Genau, die Beziehungen spielen da eine große Rolle #00:05:10#

73 S: Naja, dass die entweder von (...) es gibt so ein Leitobjekt eventuell, das ruft
74 dann die anderen Objekte, immer, fragt die an oder befragt die oder sowas,
75 und die geben dann meistens etwas zurück oder es kann auch sein, dass so ein
76 Objekt eben befragt wird und das wiederum befragt ein anderes, was das dann
77 erst an das eine zurückgibt und dann wieder zurück, also, das kann sich dann
78 auch länger ziehen und dann gehen die immer eine Beziehung miteinander ein,
79 je nach dem, mit welchem Objekt die dann kommunizieren #00:05:48#

80 I: Wo haben wir das gesehen in der Geschichte? #00:05:51#

81 S: Vor allen Dingen hatten wir jetzt das mit, von Martin zum Hell-, ne, zur
82 Hintergrundbeleuchtung und dann musste die Hintergrundbeleuchtung den
83 Helligkeitssensor befragen, dieser Helligkeitssensor hat dann eben was zurück
84 gegeben und darauf hat dann die Hintergrundbeleuchtung reagiert #00:06:15#

85 I: Also das ist so eine Möglichkeit, das sozusagen zu programmieren, da gab es
86 dann ja noch diese andere Möglichkeit, mit der wir auch angefangen haben.
87 Weißt du noch, oder kannst du erklären, wo da die Unterschiede waren? #00:06:29#

88 S: (4) #00:06:34#

89 I: Also am Anfang hatten wir es nicht so verschachtelt gemacht, wie du es vorhin
90 erklärt hast, sondern am Anfang hatten wir es uns anders überlegt #00:06:40#

91 S: Ja, einfach dass ein Objekt immer die anderen fragt und dann wieder was
92 zurück gibt oder wie? Ach so, ne, die erste Methode die wir da hatten war,
93 dass Martin direkt den Helligkeitssensor befragt #00:06:53#

94 I: Und dann? #00:06:52#

95 S: Der Helligkeitssensor was zurück gibt und dann Martin daraufhin die Hinter-
96 grundbeleuchtung (...) über das zurückgegebene informiert und dann reagiert
97 die Hintergrundbeleuchtung #00:07:08#

98 I: Ja, wie würde man sowas aufschreiben? Wir haben ja in dieser Groovy Konsole
99 gearbeitet mit diesen Objekten. Du sagst jetzt, mit dieser Information würde
100 der die weitergeben an die Hintergrundbeleuchtung und die würde sich dann
101 anschalten oder nicht anschalten. Wie sieht das aus? Oder wie programmiert
102 man sowas? #00:07:31#

- 103 S: Ja, erstmal musst du diese ganzen natürlich anmelden, also dass diese Groovy
 104 Konsole das alles überhaupt hat und dann musst du da eben mit Parameter-
 105 werten arbeiten und eben, erstmal muss die, muss man den Helligkeitssensor
 106 anfragen, wir hatten da glaube ich (2) anfr–, einfach anfragen, und dann musste
 107 der was zurückgeben, das war dann in dem Fall *ist hell*, und (1) dann hat
 108 Martin (3) ich weiß gar nicht genau, welche Methode wir da hatten (2) #00:08:16#
- 109 I: Um mit der Antwort jetzt weiterzumachen, sozusagen? #00:08:21#
- 110 S: Ja #00:08:21#
- 111 I: Die hatten wir *schalten* genannt #00:08:25#
- 112 S: Ach so, dann hatten wir *schalten* mit dem Parameterwert *ist hell* und darauf
 113 hat dann zur Hintergrundbeleuchtung (1) und die hat dann nicht reagiert, weil
 114 es war ja schon hell #00:08:40#
- 115 I: Ja, und wie würde sich die Hintergrundbeleuchtung verhalten, wenn es jetzt
 116 dunkel gewesen wäre? #00:08:45#
- 117 S: Dann hätte die sich angeschaltet #00:08:48#
- 118 I: Und woran liegt das? #00:08:50#
- 119 S: Äh, (3) ja weil die Hintergrundbeleuchtung ja nur in Kraft tritt, wenn es
 120 dunkel ist #00:09:00#
- 121 I: Und wo kannst du das bei dem Methodenaufruf bei der Hintergrundbeleuchtung
 122 festmachen, ob die jetzt an- oder ausgeht? #00:09:07#
- 123 S: Ja, das kommt je nach dem, welchen Parameterwert die dann gesagt bekommt,
 124 wenn gesagt wird *ist dunkel*, dann reagiert die eben darauf, dass die sich
 125 anschaltet, *ist hell*, dann reagiert die darauf, dass die sich eben nicht anschaltet
 126 #00:09:23#
- 127 I: Ok, und wir haben ja auch so andere Objekte gehabt, zum Beispiel diesen
 128 RGB Scheinwerfer oder so einen grünen Scheinwerfer oder einen blauen Schein-
 129 werfer. Und wenn du da die Methode *anschalten* aufrufst, was passierte da im
 130 Hintergrund? Weißt du das noch? #00:09:42#
- 131 S: (3) Ne /lacht/ #00:09:44#
- 132 I: Du hast bei diesen Scheinwerfern ja auch nicht nur Methoden, sondern auch
 133 Attribute. Wofür sind die Attribute da, und was passiert mit denen, wenn man
 134 mit so einem Objekt arbeitet? #00:10:02#
- 135 S: (4) Die Attribute führen das dann halt eben aus, beim RGB Scheinwerfer
 136 hatten wir jetzt so mischen und mit den unterschiedlichen Prozenten, dass die
 137 das dann so eingestellt haben #00:10:22#
- 138 I: Und was unterscheidet jetzt ein Attribut von einer Methode? #00:10:23#

- 139 S: (3) Ja, im Wesentlichen sind Attribute ja erstmal (2) Verben, glaube ich
140 (3), und Methoden (2) Ad-, ne, Methoden sind Verben und Attribute sind
141 Adjektive, joar, und Methoden das macht der dann genau und Attribute sind
142 dann immer die Parameterwerte dazu (2) #00:11:03#
- 143 I: Ok, wie würdest du die Arbeit insgesamt so bewerten? Wie wir da jetzt im
144 Unterricht vorgegangen sind? #00:11:10#
- 145 S: Ja, eigentlich ganz ok, zwischendurch war es vielleicht ein bisschen chaotisch
146 /lacht/ #00:11:22#
- 147 I: Das stimmt, (1) /lacht/ das muss halt sozusagen erstmal alles laufen [ja] Hat
148 das denn für dich zum Verständnis beigetragen, dass wir das, dass wir damit
149 gearbeitet haben mit dem Raspberry Pi und dem, und den kleinen Lampen
150 und dem Helligkeitssensor? #00:11:41#
- 151 S: Ja, das auf jeden Fall. Weil da musste man erstmal gucken, was man überhaupt
152 alles in diesen Raspberry Pi eingeben muss, weil vorher hatten wir das ja
153 auch aufgeschrieben, aber das war dann eben nicht ganz so detailliert und da
154 mussten wir dann eben noch alles einzeln anmelden und so weiter und so fort,
155 ja, und außerdem konnte man dann eben auch gucken, sich selbst überprüfen,
156 also ob man es richtig gemacht hat, indem man das dann halt ablaufen lassen
157 hat, und das musste nicht vom Lehrer überprüft werden #00:12:10#
- 158 I: Ja, das fandest du gut? #00:12:11#
- 159 S: Ja #00:12:14#
- 160 I: Okay, jetzt haben wir vorhin über den Computer gesprochen und du hast
161 schon ein paar Methoden genannt, so anschalten, ausschalten (.) oder Internet
162 aufrufen, fallen dir jetzt noch ein paar Attribute vielleicht dazu ein? #00:12:26#
- 163 S: (2) #00:12:29#
- 164 I: Attribute hast du nämlich vorhin auch genannt. Da sagtest du das seien
165 Adjektive... #00:12:33#
- 166 S: (5) Ja, ob (2) vielleicht auch (4) ich weiß nicht genau, wie man das, ob das
167 auch so geht, was weiß ich, ob man das schnell oder langsam auf-, ob sich das
168 so ganz schnell öffnet oder eher langsamer oder wenn man so mit der Maus
169 runterscrollt, ob das dann eben sich sehr schnell bewegt oder langsam, also das
170 kann man natürlich selbst auch variieren, aber muss ja relativ auch eingestellt
171 werden #00:13:13#
- 172 I: Du meinst so die Maus, die Mauszeigergeschwindigkeit sozusagen? #00:13:14#
- 173 S: Ja #00:13:15#
- 174 I: Ok #00:13:17#
- 175 S: Ähm, wie groß vielleicht alles auch im Wesentlichen alles abgebildet wird
176 #00:13:29#

177 I: Ja, das würde wohl gehen #00:13:31#
178 S: Und die Farben, die abgebildet werden, so, die müssen ja auch an den Rechner
179 weitergegeben werden, was der überhaupt abbilden muss #00:13:45#
180 I: Ok. Was meinst du jetzt mit Farben? Nur aus Interesse jetzt... #00:13:53#
181 S: Ja, das könnte man eventuell auch als ganzes Bild nehmen #00:14:03#
182 I: Ach so, jetzt so den Monitor an sich? #00:14:05#
183 S: Ja #00:14:09#
184 I: Und mit welchen Objekten würden so ein Computer dann interagieren? #00:14:11#
185 S: Ja, wenn man jetzt die Schnelle oder Langsamkeit der Maus nehmen würde,
186 dann mit diesem Mauspad, also mit der Maus, und dann müsste man mit dem
187 Zeiger eben gucken, ja, und mit der Größe und so variieren, da müsste man
188 dann glaube ich im Wesentlichen die Einstellung vom Computer nehmen, ja
189 #00:14:43#
190 I: Ja, super, dann sind wir fertig. Dann sage ich danke #00:14:51#

B.9 Auswertung der Interviews nach Haupt- und Unterkategorien

Um die codierten Textstellen nach den Haupt- und Unterkategorien übersichtlicher aufführen zu können, finden sich hier nochmal alle Kategorien unter Angabe der Seitenzahlen.

- Vorerfahrungen mit Informatik S. 189
 - Private Vorerfahrungen intrinsisch motiviert S. 189
 - Private Vorerfahrungen extrinsisch motiviert S. 191
- Objektorientierung als Konzept S. 193
 - Objektspiel S. 194
- Verständnis Objekt S. 196
 - Beispiele für Objekte S. 198
- Verständnis Methode S. 201
 - Umgang mit Rückgabewerten S. 202
 - Umgang mit Parameterwerten S. 204
 - Beispiele für Methoden S. 205
- Verständnis Attribut S. 207
 - Beispiele für Attribute S. 207
- Verständnis Objektkommunikation S. 209
 - Hintergrundbeleuchtung und Helligkeitssensor S. 211
 - Beispiele für Objektkommunikation S. 213
- Arbeit mit Groovy S. 215
 - Objekterstellung in Groovy S. 217
- Bewertung der Arbeit S. 219
 - Bewertung der Hardware S. 221

HK 0: Vorerfahrungen mit Informatik

Schüler B

- Z. 8–9:

S: Ich hatte in der Neunten schon Informatik und mein bester Freund studiert Wirtschaftsinformatik und mein Bruder hat halt auch Informatik und dann...

- Z. 124–127:

S: Ich weiß nicht, weil eigentlich konnte ich ja schon vorher programmieren, also...

I: Auch die Objektorientierung insbesondere?

S: Nö, aber das war jetzt nicht so schwer. /lacht/

Schüler C

- Z. 10:

S: Eigentlich, also, so vorm Informatikunterricht gar nicht. [...]

- Z. 16–19:

S: [...] Und, ja, dann konnte man ja in der Achten glaube ich nochmal wählen, da hab ich mich dann nicht für Informatik entschieden und hab dann halt jetzt gedacht zur Oberstufe, dass ich es ja mal probieren kann

Schüler D

- Z. 10–11:

S: [...] Aber, naja, es hat sich rausgestellt, wir arbeiten mit Objektkarten und sowas, das finde ich jetzt nicht so geil

Schüler F

- Z. 15–17:

I: Ja, und sonst hier in der Schule auch noch nicht in der Mittelstufe oder so?

S: Ne, ich hab das nicht angewählt gehabt, erst jetzt

UK 0.1: Private Vorerfahrungen intrinsisch motiviert

Schüler A

- Z. 5–11:

I: Hast du schon irgendwelchen Vorkontakt mit Informatik gehabt?

S: Nein

I: Privat oder so, oder?

S: Äh, ein wenig. Ich begeistere mich für die Luftfahrt und versuche, mir zurzeit ein Home Cockpit aufzubauen und da habe ich schon ein bisschen versucht, ein paar Knöpfe anzuschließen und so, aber das hat bisher noch nicht ganz so gut geklappt

Schüler B

- Z. 13–14:

I: Okay, ansonsten privat schon mal irgendetwas gemacht?

S: Nö, (1) ich glaub mir fällt nichts ein

- Z. 124–127:

S: Ich weiß nicht, weil eigentlich konnte ich ja schon vorher programmieren, also...

I: Auch die Objektorientierung insbesondere?

S: Nö, aber das war jetzt nicht so schwer. /lacht/

Schüler C

- Z. 22–24:

I: Okay, aber sonst mit den Lego Robotern auch nicht viel zuhause dann auch gemacht, sondern eher so im Regal stehen lassen?

S: Ja, genau /lacht/

Schüler D

- Z. 7–11:

S: Ich hatte eigentlich gar keinen Kontakt vorher mit Informatik, ich habe mich nur dafür interessiert, weil ich dachte, das ist so mit Computern halt und die ein bisschen besser verstehen und so, da hatte ich Lust drauf, und dann dachte ich, ich probiere es mal aus. Aber, naja, es hat sich rausgestellt, wir arbeiten mit Objektkarten und sowas, das finde ich jetzt nicht so geil

- Z. 18–22:

I: Und sonst privat Kontakt?

S: Ne, eigentlich nicht

I: Also irgendwas am Computer vorher schon gemacht?

S: Also ich arbeite schon seit vielen Jahren mit dem Computer, aber ich mach da nicht wirklich etwas informatisches mit, glaube ich

Schüler E

- Z. 7–21:

S: Ja, also ich hab relativ früh angefangen, mich dafür zu interessieren. So bestimmt mit sieben, acht habe ich dann irgendwann angefangen eben mit dem Rechner von meinen Eltern damals so ein bisschen rum zu spielen, alles auszuprobieren und dann irgendwann hab ich mir ein Buch ausgeliehen, so ein Q-Basic-Buch, da hab ich dann so ganz, aber auch nur wirklich so vier, fünf Befehle gelernt, da war ich höchstens acht oder neun, und wirklich angefangen mit Java, also da hab ich jetzt auch schon wieder das meiste verlernt, aber wirklich angefangen habe ich dann... da haben wir Minecraft gespielt und da haben wir Plug-Ins angefangen zu programmieren dafür. Für Server so eigene Game-Modi und so, da hab ich dann wirklich damit angefangen und ja, ich weiß nicht, ich interessiere mich dafür, auch für Hardware, ich habe auch meinen PC zum Beispiel selber zusammen gestellt und so, und ja, also (1)

I: Nur aus Interesse: Worin programmierst du das für Minecraft?

S: In Eclipse, also das normale Java.

Schüler F

- Z. 9–14:

S: Eigentlich nicht ganz so viel. Mein Vater arbeitet in die Richtung und ich wollte später vielleicht auch ein bisschen was mit Computern machen, joar, aber sonst...

I: Privat schon mal irgendetwas gemacht?

S: Nein, eigentlich nicht. Ich hab nur meinem Vater manchmal über die Schulter geschaut

- Z. 18–20:

I: Also hast du vorher schon Programmiererfahrungen gesammelt oder eher nicht?

S: Eher nicht

UK 0.2: Private Vorerfahrungen extrinsisch motiviert

Schüler C

- Z. 10–16:

S: [...] Mein Onkel der macht gerne was mit Technik und so und hat mir deswegen schon relativ früh verschiedene Sachen geschenkt mit denen ich dann aber auch nichts anfangen konnte. Und der hat mir zum Beispiel auch von Lego Technik und so auch verschiedene Roboter gezeigt, die man programmieren kann, und mir auch was erklärt, aber da weiß ich eigentlich gar nichts mehr von, weil das mich halt damals noch nicht so wirklich interessiert hat. [...]

Schüler F

- Z. 9:

S: [...] Mein Vater arbeitet in die Richtung [...]

HK 1: Objektorientierung als Konzept

Schüler B

- Z. 18:

S: Abbott

Schüler C

- Z. 29:

S: Ähm, (.) weiß ich nicht. Ich kann nicht (.)

Schüler D

- Z. 24–27:

I: [...] Thema Objektorientierung. Wir haben da ja verschiedene Begriffe kennengelernt, vielleicht kannst du da erzählen, was dir da spontan zu einfällt?

S: Zu Objektkarten?

- Z. 30–32:

S: Also erstmal dieses, dass man eine Geschichte, glaube ich, so mit dem Computer verwirklicht, irgendwie. Dass man diese Befehle aufschreibt und die dann ausgeführt werden vom Computer

Schüler E

- Z. 26–32:

S: Objektorientierung ist ja, dass man sich vorgibt, also ich weiß es gar nicht genau, um ehrlich zu sein, aber man hat ja auf jeden Fall schon mal eine Schnittstelle sozusagen, so würde ich das jetzt nennen, auf die man sozusagen drauf programmiert, und vorgegeben eben Objekte, die man mit Methoden ansprechen kann und man hat eben (2) ich weiß nicht (2). Man schreibt ja irgendwie was, was sich direkt auf was bezieht, auf verschiedene wirkliche Sachen

- Z. 134–137:

S: Ja, (4), dass man immer sich direkt auf andere Objekte bezieht und nicht nur, also noch andere Sachen sozusagen steuert, außer dass man nur irgendwelche Eingaben bekommt und die verarbeitet und wieder ausgibt. Also naja, ich weiß es tatsächlich gar nicht genau

Schüler F

- Z. 25–27:

S: Zur Objektorientierung fällt mir erstmal im Wesentlichen Objektkarten ein [ja] und die Beziehungen zwischen all den Objekten und was die Objekte alles überhaupt machen, das man in die Objektkarte schreibt (4) joar

UK 1.1: Objektspiel

Schüler A

- Z. 26–31:

S: Ja, wir haben ja jetzt dieses Objektspiel gemacht, und da haben wir halt diese Objektkarten erstellt, die quasi ein Objekt, wie zum Beispiel einen Scheinwerfer, darstellen und wo dann drin steht, was der machen kann und welchen Status der gerade hat, also wie der gerade drauf ist, und das haben wir dann halt mit dem Objektkartenspiel durchgeführt, und mal versucht, diese Objekte miteinander zu verbinden, dass die auch miteinander interagieren, ja

- Z. 58–63:

S: Ja wir haben das halt erst theoretisch gemacht, wie gesagt dieses Objektkartenspiel, haben dann versucht, das quasi in so (.) ein Programm einzubringen, genau das was wir halt gesagt haben, mit diesen bestimmten (.) Phrasen, also, bestimmte Sätze, die wir dann eingebracht haben, und dann quasi genau das was auf den Objektkarten drauf stand, wurde dann genau so in das Programm eingebracht, was wir später mit den Raspberry Pis gemacht haben

- Z. 164–165:

S: Dass die sich gegenseitig kennen, das war auch das mit den (1) Fäden, die das auch im Objektkartenspiel gemacht haben

Schüler C

- Z. 110–112:

S: [...] Wir hatten da ja so Fäden gespannt, so Beziehungslien, und das, weiß ich nicht, muss halt in irgendeiner Verbindung stehen

- Z. 205–209:

S: [...] Und ich persönlich fand das Objektspiel mit den Karten immer sehr verwirrend und hab das nie so ganz verstanden. Aber wenn man das dann halt in diese Groovy Konsole eingegeben hat, wurde es für mich halt immer deutlicher und ich habe es halt besser verstanden. Und da glaub ich halt schon, dass diese (.) Praxisarbeit zum Verständnis beiträgt

- Z. 212–223:

S: Also ich finde man musste ja immer so bestimmt reden... *Ich, Martin, rufe bei dem Objekt so und so die Methode so und so hervor*, und das muss man zwar auch genau so in den Computer eingeben, aber ich fand das dann immer (.) ich hab das nie so ganz verstanden, warum man das jetzt genau so machen muss, weil ich (1) also ich weiß ja, dass ich dem Computer nicht sagen, ja dass ich nicht sagen kann, *jetzt schalte*

den grünen Scheinwerfer an. Das versteht der ja nicht. Ich weiß schon, dass man da bestimmte Befehle eingeben muss, aber ich habe es halt im Objektspiel nicht so ganz verstanden, warum man dann so diese Befehle so sagen musste

I: Ja, interessant. Diese Sprechweise zu dem eigentlichen Quelltext, diese Verbindung ist dir nicht so ganz klar geworden?

S: Genau

Schüler D

- Z. 34–36:

S: Ja, man muss aus der Geschichte irgendwie die Objekte, die Methoden und die Attributwerte oder so herausfinden (.) und das dann alles irgendwie zuordnen und am Computer halt eintippen /lacht/

- Z. 37–38:

I: Wie findest du die Objekte?

S: Ja ich gucke mir die ganzen Substantive an. [...]

- Z. 77–78:

S: [...] wir hatten ja in der letzten Stunde auch diese Beziehungspfeile [...]

Schüler E

- Z. 38–40:

S: [...] und da hat man ja eben die Nomen zu Objekten gemacht bis auf verschiedene Wörter wie zum Beispiel irgendwelche Mengenangaben oder sowas

Schüler F

- Z. 139–142:

S: (3) Ja, im Wesentlichen sind Attribute ja erstmal (2) Verben, glaube ich (3), und Methoden (2) Ad–, ne, Methoden sind Verben und Attribute sind Adjektive, joar, und Methoden das macht der dann genau und Attribute sind dann immer die Parameterwerte dazu (2)

HK 2: Verständnis Objekt

Schüler A

- Z. 34–38:

S: [...] Einfach etwas, womit man interagieren kann

I: (1) Ist das die einzige Begründung für dich, weil ich damit interagieren kann?

S: Ja

- Z. 130–131:

S: Das führt diese Methode aus und ändert dann den Status noch, dass es auch weiß, dass es an ist zum Beispiel

- Z. 181–184:

I: [...] kannst du dir jetzt vorstellen, was ein Objekt ist? Oder wie sich zwei Objekte vielleicht unterscheiden oder was ein Objekt auszeichnet?

S: Ja, dass die unterschiedliche Methoden haben [...]

Schüler B

- Z. 24:

S: Einen Gegenstand, den man berühren kann (1) /lacht/

- Z. 27–29:

S: Ein Scheinwerfer, ja, weil den kann man halt steuern und den kann man auch berühren, der ist ja jetzt nicht so wie Luft, also Luft berührt man zwar immer /lacht/, aber, ja

- Z. 34–35:

S: [...] Also die führen ja Sachen aus, zum Beispiel bei anderen Objekten, wie jetzt zum Beispiel dem Scheinwerfer

- Z. 38–39:

S: Ja, also, jedes Objekt hat ja zugehörige Methoden, und die kann man halt aufrufen. [...]

Schüler C

- Z. 49–50:

I: [...] Kannst du mir da irgendwie ein Beispiel für nennen?

S: (1) weiß ich nicht. Alles mögliche, eigentlich, oder?

- Z. 53–55:

S: Also, ich finde ein Objekt kann eigentlich alles sein, weil, entweder wird dieses Objekt angeschaltet oder ausgeschaltet und man macht irgendeine Methode mit dem, sag ich jetzt mal. [...]

- Z. 61–61:

S: (.) weiß ich nicht, also mir fällt jetzt gerade nichts ein, was kein Objekt ist

- Z. 177–181:

I: [...] Wie konntest du da unterscheiden zwischen den Objekten, welches Objekt du jetzt ansprechen willst oder auf welchem du jetzt eine Methode aufrufst?

S: Man hat die ja benannt danach. Also entweder ein grüner Scheinwerfer oder ein blauer Scheinwerfer

Schüler D

- Z. 38–39:

S: [...] Alles, was man anfassen kann, ist ein Objekt

- Z. 44–47:

S: [...] weil gerade in der Geschichte, die wir auch im Unterricht hatten, konnte man ja damit die Sachen durchführen, zum Beispiel anschalten, ausschalten und das war ja auch wichtig für die Geschichte und von daher ist das Objekt auch wichtig und das brauchte man dann.

- Z. 50–52:

S: [...] Ich denke mal alles, womit interagiert wird irgendwie. Also es würde jetzt nicht Sinn machen, wenn man irgendwie Tisch sagt oder so, wenn damit nichts gemacht wird [...]

- Z. 59–60:

S: Ja, keine Ahnung, Apfel, Tisch, Stuhl, sowas alles kann ein Objekt sein und Freundschaft oder so halt nicht

Schüler E

- Z. 31–32:

S: [...] was sich direkt auf was bezieht, auf verschiedene wirkliche Sachen

- Z. 43–44:

S: [...] Spieler wäre jetzt für mich das Objekt, weil ich da jetzt irgendwelche Methoden dran ausführen würde sozusagen

Schüler F

- Z. 32–33:

S: Ja, weil das eben ein Gegenstand ist, der unterschiedliche Sachen ausführen muss und ja, das gesagt bekommt und auch wieder zurückgibt

- Z. 41–42:

S: [...] aber auch generell, wenn man sich so einen ganz normalen Gegenstand nimmt [...]

UK 2.1: Beispiele für Objekte

Schüler A

- Z. 34:

S: [...] Sowas wie ein Scheinwerfer zum Beispiel. [...]

- Z. 40:

S: [...] ein Schalter oder so, ein Knopf vielleicht

- Z. 42–45:

I: [...] Beispiele unabhängig von diesen Bausteinen, mit denen wir am Raspberry Pi gearbeitet haben, von diesen kleinen Platinen? Jetzt aus der richtigen Welt?

S: Müsste ich jetzt überlegen, aber, ich weiß nicht, (2), keine Ahnung

- Z. 48:

S: [...] Bildschirm [...]

- Z. 145:

S: [...] Helligkeitssensor vom Handy [...]

Schüler B

- Z. 27:

S: Ein Scheinwerfer [...]

- Z. 31:

S: Der Mensch, irgend ein Name, Dieter, Martin

- Z. 79–82:

I: Okay. Ist dir vielleicht inzwischen ein Beispiel eingefallen für ein Objekt aus der realen Welt? Du hattest jetzt vorhin nur Dieter genannt, also Menschen, aber vielleicht fällt dir noch ein anderes Beispiel ein?

S: (5) /lacht/

Schüler C

- Z. 55–57:

S: [...] Oder wir hatten ja auch zum Beispiel Würfel, der wird ja, also der Würfel macht ja nichts, aber der Würfel ist ja trotzdem ein Objekt, mit dem was gemacht wird

- Z. 64:

S: Ja, äh, Kaffeemaschine (1) [...]

Schüler D

- Z. 59–60:

S: Ja, keine Ahnung, Apfel, Tisch, Stuhl, sowas alles kann ein Objekt sein und Freundschaft oder so halt nicht

Schüler E

- Z. 34–36:

S: Ja, zum Beispiel so eine Lampe oder das kann ja auch irgendwas anderes sein. Keine Ahnung, in dem Spiel, zum Beispiel der Spieler kann ja auch ein Objekt sein

- Z. 43–44:

S: Spieler, ja, (10). Ich weiß nicht, also Spieler wäre jetzt für mich das Objekt, weil ich da jetzt irgendwelche Methoden dran ausführen würde sozusagen

- Z. 47:

S: Ja, also zum Beispiel der rote Scheinwerfer oder sowas

- Z. 82:

S: Zum Beispiel das Auto oder die Straße oder sowas, oder so eine Straßenlaterne [...]

Schüler F

- Z. 29:

S: [...] so Scheinwerfer oder sowas

- Z. 35–36:

S: (1) Ja (1) Menschen zum Beispiel, wie dieser Martin da jetzt (5) Eventuell so die Bühne hatten wir jetzt auch als Objekt, aber das war ja eher ein Standort

- Z. 40–43:

S: (3) Entweder so Menschen untereinander könnte man auch als Objekte bezeichnen oder recht viele elektronische Gegenstände natürlich, aber auch generell, wenn man sich so einen ganz normalen Gegenstand nimmt und wieder zurück stellt oder so, dann ist das ja auch so eine Tätigkeit, die man ausgeführt hat

- Z. 58:

S: [...] mir würde jetzt nur der Wecker einfallen [...]

HK 3: Verständnis Methoden

Schüler A

- Z. 100–102:

S: [...] und beim Helligkeitssensor war es so, dass das Programm sich selbst quasi einmal abfragt, welcher Parameter da rein muss

- Z. 130–131:

S: Das führt diese Methode aus und ändert dann den Status noch, dass es auch weiß, dass es an ist zum Beispiel

- Z. 135–138:

S: Ja, das guckt halt erstmal nach, ob es diese Methode gibt, und wenn die Methode vorhanden ist, ja führt das die einfach aus und guckt nach, ob danach noch irgendetwas passieren muss, ob es den Status geändert werden muss, oder so etwas

Schüler B

- Z. 72–75:

I: Ok, und wie verhält sich allgemein so ein Objekt, wenn man darauf eine Methode aufruft. Was passiert da?

S: Das kriegt im Sequenzdiagramm so einen Kasten, also eine Lebenslinie hat es ja schon, aber so einen Aktivitätskasten (1) /lacht/

- Z. 77–78:

S: Die ändern sich, zum Beispiel, wenn man den Scheinwerfer jetzt anschaltet, dann ändert sich der Status auf an und nicht mehr auf aus

Schüler C

- Z. 156–158:

S: Ja, wenn da eine Methode aufgerufen wird, verändert das immer seinen Status oder halt, kommt halt auf die Methode drauf an, was es verändert, aber irgendein Attributwert wird dann halt immer verändert. [...]

Schüler D

- Z. 97–98:

S: Ja, ich denke das Prinzip ist dasselbe. Es wird auch halt Licht angemacht, nur halt in einer bestimmten (1) Menge, sozusagen. [...]

Schüler E

- Z. 60–61:

S: Man kann ja die Methode auch mit Parameter ausführen. Und über Variablen natürlich

- Z. 70–71:

S: Ja das Objekt führt dann eben die Methode aus und gibt eben wieder zurück zu dem, dem (.) ja, einfach zurück

- Z. 141–145:

S: [...] in die Klammer hinter der Methode, die ja immer hingehört, kann man ja noch einen Parameter mitgeben, und das Objekt eben, das ist dann ja in der Schnittstelle, in der API, hinterlegt und das weiß dann was mit dem Wert anzufangen und führt die Methode eben mit verschiedenen Werten dann eben anders aus

UK 3.1: Umgang mit Rückgabewerten

Schüler A

- Z. 141–142:

S: Dann gibt es eine (.) Rückgabe, und dann sagt der Helligkeitssensor zum Beispiel, dass es hell ist, und schickt das dann wieder zurück

Schüler B

- Z. 53–56:

S: [...] Martin fragt erst den Helligkeitssensor ob es hell ist und speichert die Antwort dann unter einer Variablen, mit der Variablen, die gibt er dann als Parameterwert weiter an die Hintergrundbeleuchtung [...]

Schüler D

- Z. 83–85:

S: [...] der Scheinwerfer führt die dann aus, das Licht ist an, und es wird wieder zurück gegeben

- Z. 112–114:

S: Dass man halt... Dass ein Objekt gefragt wird und das eine Antwort zurück gibt. Und also, es in einem Binärcode oder so antwortet, dass man nur *ja* oder *nein* oder *Null* oder *Eins* zurück bekommt

- Z. 123–129:

S: Ähm (5) wenn man... (1) also das erste Objekt ist eine Person, die was kauft und die fragt das andere Objekt, *wie viel* und der gibt zurück, irgendeinen Wert, also wie es halt kostet

I: Okay, und die Methode würde heißen? Wie würdest du die nennen?

S: *Kostet wie viel* (.) Klammer auf, Klammer zu?

I: Und dann gibt der zurück: *2 Euro*, oder so? [Ja] Ah okay, so stellst du dir das also vor. [...]

Schüler E

- Z. 56–58:

S: [...] und dann können die ja über die Methode sozusagen immer wieder was zurück geben [...]

- Z. 70–71:

S: Ja das Objekt führt dann eben die Methode aus und gibt eben wieder zurück zu dem, dem (.) ja, einfach zurück

- Z. 70–158:

S: Zum Beispiel, wenn man jetzt, ich weiß nicht, wenn das Auto jetzt einen Regensensor hat, dann befragt man den Regensensor ob es regnet, und dann könnte der eben den Wert in einer Variable abspeichern, wenn es zum Beispiel regnet könnte man mit dem Wert, ob der jetzt *ja* oder *nein* ist oder eben *true* oder *false*, damit dann den Scheibenwischer anschalten [...]

Schüler F

- Z. 32–33:

S: [...] der unterschiedliche Sachen ausführen muss und ja, das gesagt bekommt und auch wieder zurück gibt

- Z. 105–107:

S: [...] erstmal muss die, muss man den Helligkeitssensor anfragen, wir hatten da glaube ich (2) anfr–, einfach anfragen, und dann musste der was zurückgeben, das war dann in dem Fall *ist hell* [...]

UK 3.2: Umgang mit Parameterwerten

Schüler A

- Z. 98–100:

S: [...] also beim RGB Scheinwerfer konnte man die Parameter direkt eingeben, dass rot zum Beispiel 40 Prozent hat oder so, das muss man direkt ins Programm rein schreiben [...]

Schüler B

- Z. 53–56:

S: [...] Martin fragt erst den Helligkeitssensor ob es hell ist und speichert die Antwort dann unter einer Variablen, mit der Variablen, die gibt er dann als Parameterwert weiter an die Hintergrundbeleuchtung [...]

Schüler C

- Z. 33–35:

S: [...] wann der angehen soll, wann der ausgeht, wie lange er anbleiben soll und so, und man kann halt da ja verschiedene Funktionen einstellen [...]

- Z. 188–190:

S: [...] wenn das jetzt zum Beispiel *warten* wäre oder so, dann kann dahinter halt noch das Parameterwert dann, glaube ich, zum Beispiel wie lange der warten muss

- Z. 195–198:

S: Ach so, da würde man dann der Hintergrundbeleuchtung sagen *anschalten*, und dann dahinter mit dem Parameterwert, entweder schreibt man dann da rein schon direkt *ist hell* oder man schreibt da halt rein *Antwort des Helligkeitssensors* oder halt das, wie man das definiert hat vorher

Schüler D

- Z. 92–94:

S: Ja, zum Beispiel hatten wir bei dem RGB Scheinwerfer dieses *Licht mischen* und so, dass man noch mit den Parametern arbeitet und sagt, dass grün 70 Prozent, rot 30 und so weiter

Schüler E

- Z. 148–151:

S: Wenn man jetzt zum Beispiel sagt *fahre vorwärts*, und dann sagt man *20 Meter* oder wie auch immer, kommt jetzt auf die Maßeinheiten an, dann würde das Auto nur 20 Meter vorwärts fahren, wenn da ja jetzt *100* eingibt, würde das eben 100 Meter vorwärts fahren

- Z. 162–164:

S: [...] die Methode *wische*, die müsste der haben, und dann müsste man die Methode mit dem Parameterwert aufrufen, den man vom Regensensor hat

Schüler F

- Z. 112–114:

S: Ach so, dann hatten wir *schalten* mit dem Parameterwert *ist hell* und darauf hat dann zur Hintergrundbeleuchtung (1) und die hat dann nicht reagiert, weil es war ja schon hell

- Z. 121–125:

I: Und wo kannst du das bei dem Methodenaufruf bei der Hintergrundbeleuchtung festmachen, ob die jetzt an- oder ausgeht?

S: Ja, das kommt je nach dem, welchen Parameterwert die dann gesagt bekommt, wenn gesagt wird *ist dunkel*, dann reagiert die eben darauf, dass die sich anschaltet, *ist hell*, dann reagiert die darauf, dass die sich eben nicht anschaltet

- Z. 139–142:

S: (3) Ja, im Wesentlichen sind Attribute ja erstmal (2) Verben, glaube ich (3), und Methoden (2) Ad–, ne, Methoden sind Verben und Attribute sind Adjektive, joar, und Methoden das macht der dann genau und Attribute sind dann immer die Parameterwerte dazu (2)

UK 3.3: Beispiele für Methoden

Schüler A

- Z. 123:

S: Ja zum Beispiel *anschalten* oder *ausschalten*

- Z. 52–55:

I: Was hättest du da vielleicht noch für Methoden?

S: Beim Bildschirm?

I: Ja

S: Dass der und der Pixel vielleicht diese Farbe hat, und, ja

Schüler B

- Z. 39–41:

S: [...] den Scheinwerfer, den kann man ja auf jeden Fall schon mal *anschalten* und *ausschalten* (.) und das muss ja irgendjemand machen, der kann sich ja nicht von alleine anschalten

- Z. 85–87:

S: Erst mal anfahren, bremsen, (3) anschalten, ausschalten, also den Motor, vielleicht auch die Lampen oder das Navigationssystem noch irgendwie oder Musik noch irgendwie da reinbringen

Schüler C

- Z. 67–68:

S: *Anschalten, ausschalten, dann halt verschiedene Sorten Kaffee machen, Bohnen mahlen oder sowas...*

Schüler D

- Z. 83–85:

S: [...] Dass man dann die Methode *anschalten* aufruft und der Scheinwerfer führt die dann aus, das Licht ist an, und es wird wieder zurück gegeben

- Z. 164–165:

S: [...] Weil die Methoden oder so waren ja schon eingespeichert

Schüler E

- Z. 49–50:

S: Ja man könnte den anschalten, man könnte den ausschalten, vielleicht noch *blinken* als Methode

- Z. 87–89:

S: Ja das könnte natürlich vorwärts fahren, das könnte rückwärts fahren, das könnte meinetwegen hupen, vielleicht noch das Licht einschalten, die Blinker einschalten, bremsen natürlich, lenken natürlich auch

- Z. 162:

S: [...] also die Methode *wische*, die müsste der haben, [...]

Schüler F

- Z. 48–50:

S: Ja, wenn du jetzt an den Computer gehst und den erstmal anschaltest. Dann muss der Computer sich anschalten und dann hochfahren und dann muss der ja irgendwie fragen, dass du das Passwort eingibst und so weiter

- Z. 52–54:

S: Ja, du kannst den natürlich wieder ausschalten und es gibt natürlich unterschiedliche Funktionen. Du kannst Internet aufrufen und dann innerhalb des Internets noch andere Sachen aufrufen, das kann ja (.) sehr weit gehen

HK 4: Verständnis Attribute

Schüler A

- Z. 130–131:

S: Das führt diese Methode aus und ändert dann den Status noch, dass es auch weiß, dass es an ist zum Beispiel

Schüler E

- Z. 100–102:

I: [...] Und wenn das Auto jetzt fährt, was würde dann mit dem Attribut passieren?

S: Ja, das würde wahrscheinlich immer wieder geupdated werden

Schüler F

- Z. 133–137:

I: [...] Wofür sind die Attribute da, und was passiert mit denen, wenn man mit so einem Objekt arbeitet?

S: (4) Die Attribute führen das dann halt eben aus, beim RGB Scheinwerfer hatten wir jetzt so mischen und mit den unterschiedlichen Prozenten, dass die das dann so eingestellt haben

- Z. 139–142:

S: (3) Ja, im Wesentlichen sind Attribute ja erstmal (2) Verben, glaube ich (3), und Methoden (2) Ad–, ne, Methoden sind Verben und Attribute sind Adjektive, joar, und Methoden das macht der dann genau und Attribute sind dann immer die Parameterwerte dazu (2)

UK 4.1: Beispiele für Attribute

Schüler B

- Z. 89:

S: Vielleicht die Farbe (2) oder den Status

Schüler C

- Z. 44–46:

S: [...] Für den Standort, für das Attribut Standort, wäre

S: Wäre dann Bühne zum Beispiel

- Z. 77–85:

I: /lacht/ Du sagtest ja vorhin so Bohnen mahlen, wie könnte man das vielleicht auch mit einem Attribut in Verbindung bringen? Dass man vorher fragt...

S: Ob man die grob oder fein... Ich hab keinen Plan... /lacht/

I: Ne, ich mein ob noch Bohnen da sind oder so (.)

S: (2) Joar, vielleicht halt ob noch was in der Kaffeemaschine drin ist oder so zum Beispiel auch Wasser, muss man ja vorher auch einfüllen. Dass das selbst sag ich jetzt mal misst, ob da noch Wasser drin ist oder nicht und dann zeigt der halt an, *Wasser nachfüllen*, oder so
- Z. 93–94:

S: Vielleicht irgendwie *Füllmenge Wasser* und dann *voll* oder *leer* oder irgendwie sowas. Oder wenn das halt irgendwie *WasserVoll ja nein* irgendwie

Schüler E

- Z. 94–96:

S: Ja man könnte zum Beispiel die Farbe haben, oder irgendwelche Zustände von verschiedenen Teilen des Autos, zum Beispiel ob jetzt gerade irgendwelche Blinker aktiviert sind oder sowas, oder den Standort des Autos

Schüler F

- Z. 58–63:

S: [...] mir würde jetzt nur der Wecker einfallen, so quasi so, aber der ist ja nicht unbedingt auf dem Computer, so dass du dem eine bestimmte Zeit gibst, wann er sich dann wieder melden muss, oder auch, wenn er sich selbst wieder ausschaltet, weil der zu lange nicht benutzt wurde, da ist ja auch so ein Attributwert, so ein Parameter, nach einer bestimmten Zeit schaltet er sich dann wieder aus
- Z. 167–171:

S: [...] ob man das schnell oder langsam auf-, ob sich das so ganz schnell öffnet oder eher langsamer oder wenn man so mit der Maus runterscrollt, ob das dann eben sich sehr schnell bewegt oder langsam, also das kann man natürlich selbst auch variieren, aber muss ja relativ auch eingestellt werden
- Z. 175:

S: Ähm, wie groß vielleicht alles auch im Wesentlichen alles abgebildet wird

HK 5: Verständnis Objektkommunikation

Schüler A

- Z. 29–31:

S: [...] und das haben wir dann halt mit dem Objektkartenspiel durchgeführt, und mal versucht, diese Objekte miteinander zu verbinden, dass die auch miteinander interagieren, ja

- Z. 119–121:

S: Weil die sonst nicht wissen, wo, wie, wo die das hin schicken können. Zum Beispiel dass Martin weiß, dass es den grünen Scheinwerfer gibt und dass der das dann da auch direkt hin schicken kann

- Z. 150–151:

S: Ja, äh, vielleicht wenn man wen anders grüßt, zum Beispiel. Dass der dann auch zurück grüßt

- Z. 154–156:

S: Wenn ich das wieder mit Menschen machen kann, dass ich und ein Freund von mir, dass ich ihm zum Beispiel rufe, zurufe, Hallo, und er dann zurück ruft, hallo

- Z. 163–165:

I: Okay, aber dieses hinzufügen, wofür war das nochmal wichtig?

S: Dass die sich gegenseitig kennen, das war auch das mit den (1) Fäden, die das auch im Objektkartenspiel gemacht haben

Schüler B

- Z. 46:

S: Man stellt Beziehungen zwischen denen her, gerichtete oder eine ungerichtete [...]

Schüler C

- Z. 103–107:

S: [...] Und dass man da halt, also, man möchte ja das eine Gerät, bei dem einen Gerät was auslösen oder so, und das wird dann ja mit der Antwort des anderen Geräts geschaltet und damit das halt an gehen kann, fragt das Gerät sozusagen das andere Gerät

- Z. 110–112:

S: (6) Also die brauchen ja irgendwas, was sie verbindet, sag ich jetzt mal. Wir hatten da ja so Fäden gespannt, so Beziehungslinien, und das, weiß ich nicht, muss halt in irgendeiner Verbindung stehen

Schüler D

- Z. 76–80:

S: Ja, die Interaktion passiert ja, indem eine Methode aufgerufen wird von einem Objekt zum anderen. Und wir hatten ja in der letzten Stunde auch diese Beziehungsfeile, dass die sich alle erstmal kennen müssen, und dann werden ja Methoden aufgerufen, und dann werden die halt ausgeführt vom bestimmten Objekt

- Z. 141–144:

S: Ja bei dem Beispiel kann das eine Objekt mit der Antwort ja nicht viel machen, das will ja nur die Information haben, und bei anderen Objekten könnte das so sein, dass wenn der die Antwort kriegt, dass dann irgend eine andere Methode ausgeführt wird, aber das ist ja in dem Beispiel nicht der Fall

- Z. 154–156:

S: (2) Ja, wenn ein Objekt ein anderes befragt oder eine Methode daran ausführt, dann muss das Objekt das andere ja kennen, um das halt zu machen und die Methoden auch zu kennen

- Z. 172–177:

S: Ach so, (2) also die müssen halt erstmal das irgendwie empfangen das Signal, also, die Methode, die daran ausgeführt wird, das müssen die irgendwie empfangen schätzen ich, und dann verarbeiten und dann durchführen. Ich weiß auch nicht, wie das dann im Einzelnen passiert, aber, ja (2) Wir haben darüber ja auch nicht wirklich gesprochen, wir haben nur gesagt, Methoden werden durchgeführt und es wird zurück gegeben und... fertig. Und Beziehungen

Schüler E

- Z. 56–58:

S: Wir haben ja jetzt diese Beziehung gehabt, dass die einander kennen und dann können die ja über die Methode sozusagen immer wieder was zurück geben noch

Schüler F

- Z. 71–79:

S: Das mit den Beziehungen was wir jetzt hatten, oder?

I: Genau, die Beziehungen spielen da eine große Rolle

S: Naja, dass die entweder von (.) es gibt so ein Leitobjekt eventuell, das ruft dann die anderen Objekte, immer, fragt die an oder befragt die oder sowas, und die geben dann meistens etwas zurück oder es kann auch sein, dass so ein Objekt eben befragt wird und das wiederum befragt ein anderes, was das dann erst an das eine zurückgibt und dann wieder zurück, also, das kann sich dann auch länger ziehen und dann gehen die immer eine Beziehung miteinander ein, je nach dem, mit welchem Objekt die dann kommunizieren

UK 5.1: Hintergrundbeleuchtung und Helligkeitssensor

Schüler A

- Z. 81–84:

S: Wir hatten ja zum Beispiel das Beispiel mit der Hintergrundbeleuchtung und dem Helligkeitssensor, dass der, dass die Hintergrundbeleuchtung quasi direkt auf den Helligkeitssensor zugreift und halt guckt, wie hell es ist, und je nach dem dann halt die Hintergrundbeleuchtung schaltet

- Z. 86–89:

S: Ja, da konnte man ins Programm reinschreiben, dass die Hintergrundbeleuchtung den Helligkeitssensor befragt, ob es hell ist, und je nach dem was dann für eine Antwort kam, hat die Hintergrundbeleuchtung sich dann an- oder ausgeschaltet

- Z. 107–112:

I: Ok, und in der letzten Stunde hatten wir dann auch noch über eine andere Möglichkeit gesprochen, wie man das auch realisieren kann, dass man diese Hintergrundbeleuchtung schalten kann mit dem (.), mit der Antwort des Helligkeitssensor. Weißt du wie die aussah?

S: Ja, das wurde dann in eine Zeile geschrieben, aber ganz genau weiß ich das auch nicht mehr

Schüler B

- Z. 47–50:

S: [...] wir hatten das ja mit Martin und dem Scheinwerfer. Da muss der Scheinwerfer ja nicht unbedingt Martin kennen, sondern nur Martin den Scheinwerfer, damit er bei dem Scheinwerfer dann eine Methode aufrufen kann

- Z. 53–62:

S: [...] Entweder man speichert den, also, Martin fragt erst den Helligkeitssensor ob es hell ist und speichert die Antwort dann unter einer Variablen, mit der Variablen, die gibt er dann als Parameterwert weiter an die Hintergrundbeleuchtung oder wir hatten, dass der Martin nur die

Hintergrundbeleuchtung kennt, die anschaltet und die dann hält erst den Helligkeitssensor fragt und dann hält je nach dem anschaltet oder nicht

I: Wo ist da der Unterschied?

S: Wir müssen nicht so viele Beziehungen herstellen. Also Martin muss hält zum Beispiel bei der zweiten Methode nicht den Helligkeitssensor kennen, sondern nur die Hintergrundbeleuchtung

Schüler C

- Z. 118–126:

S: Ähm, ja also die Person, was ja bei dem Objektspiel Martin war, der hat ja die Hintergrundbeleuchtung angeschaltet mit der Antwort vom Helligkeitssensor. Und dann hat die Hintergrundbeleuchtung ja den Sensor befragt und wenn die Antwort hält *hell* war, dann ist es ausgeblieben, und wenn die Antwort *dunkel* war, dann ist es angegangen

I: Ja, hat da jetzt die Hintergrundbeleuchtung den Helligkeitssensor gefragt oder wie hatten wir das am Anfang gemacht?

S: Ne, also am Anfang hatten wir, dass die Person den Helligkeitssensor befragt und mit der Antwort, die er kriegt, die Beleuchtung schaltet

- Z. 136–141 (s. auch Abbildung B.1, S. 167):

S: Dann würde ja, wenn die Hintergrundbeleuchtung den Sensor befragen würde, wäre hier ja keine Verbindungsleitung /deutet eine Verbindung von Martin zum Sensor an/ und wenn Martin erst den Sensor befragen würde und mit der Antwort die Hintergrundbeleuchtung schalten würde, wäre hier keine Verbindungsleitung /deutet eine Verbindung von Hintergrundbeleuchtung zum Helligkeitssensor an/, glaube ich

Schüler D

- Z. 146–149:

S: Ja, das (1) war dann dieses bedingte Anschalten von der Hintergrundbeleuchtung und da hängt, da hängt das ja dann von der Antwort ab, ob es an geht oder nicht und das ist ja bei diesem Beispiel, was ich gerade genannt habe, nicht der Fall

Schüler E

- Z. 63–67:

S: Ja zum Beispiel wenn man jetzt den Helligkeitssensor befragt und dann die Antwort, hat man eine Variable, und setzt eben die Variable auf den Wert, der die Antwort ist von der Methode, die man beim Helligkeitssensor ausgeführt hat und kann dann mit der Variable oder dem Wert, den man gespeichert hat, eben die Hintergrundbeleuchtung natürlich schalten

Schüler F

- Z. 81–84:
S: Vor allen Dingen hatten wir jetzt das mit, von Martin zum Hell–, ne, zur Hintergrundbeleuchtung und dann musste die Hintergrundbeleuchtung den Helligkeitssensor befragen, dieser Helligkeitssensor hat dann eben was zurück gegeben und darauf hat dann die Hintergrundbeleuchtung reagiert
- Z. 91–97:
S: Ja, einfach dass ein Objekt immer die anderen fragt und dann wieder was zurück gibt oder wie? Ach so, ne, die erste Methode die wir da hatten war, dass Martin direkt den Helligkeitssensor befragt
I: Und dann?
S: Der Helligkeitssensor was zurück gibt und dann Martin daraufhin die Hintergrundbeleuchtung (.) über das zurückgegebene informiert und dann reagiert die Hintergrundbeleuchtung

UK 5.2: Beispiele für Objektkommunikation

Schüler A

- Z. 199–202:
S: [...] wenn einem zum Beispiel jemand zuruft, ob die Ampel rot ist, dann muss man ja auch je nach dem antworten, ob die rot oder ob die grün ist, da muss man halt unterschiedlich reagieren
- Z. 207–214:
S: [...] dass man vorne an einer Ampel steht und von hinten ruft einem jemand zu, ob es noch rot ist [ach so], und dann muss man halt antworten, auch wenn man die Person nicht kennt, ob es rot ist oder grün
I: Ja, was antwortest du dann?
S: Ja, zum Beispiel *ist noch rot* oder so
I: Okay, und was würde die Person dann mit dieser Antwort machen?
S: Ja, sich bedanken und dann weiß sie es halt und bleibt halt noch stehen

Schüler D

- Z. 120:
S: Da würde mir jetzt halt Kommunikation halt einfallen, das ist ja alles das
- Z. 123–125:
S: Ähm (5) wenn man... (1) also das erste Objekt ist eine Person, die was kauft und die fragt das andere Objekt, *wie viel* und der gibt zurück, irgendeinen Wert, also wie es halt kostet

Schüler E

- Z. 154–158:

S: Zum Beispiel, wenn man jetzt, ich weiß nicht, wenn das Auto jetzt einen Regensensor hat, dann befragt man den Regensensor ob es regnet, und dann könnte der eben den Wert in einer Variable abspeichern, wenn es zum Beispiel regnet könnte man mit dem Wert, ob der jetzt *ja* oder *nein* ist oder eben *true* oder *false*, damit dann den Scheibenwischer anschalten [...]

Schüler F

- Z. 184–187:

I: Und mit welchen Objekten würden so ein Computer dann interagieren?

S: Ja, wenn man jetzt die Schnelle oder Langsamkeit der Maus nehmen würde, dann mit diesem Mauspad, also mit der Maus, und dann müsste man mit dem Zeiger eben gucken, ja [...]

HK 6: Arbeit mit Groovy

Schüler A

- Z. 58–63:

S: Ja wir haben das halt erst theoretisch gemacht, wie gesagt dieses Objektkartenspiel, haben dann versucht, das quasi in so (.) ein Programm einzubringen, genau das was wir halt gesagt haben, mit diesen bestimmten (.) Phrasen, also, bestimmte Sätze, die wir dann eingebracht haben, und dann quasi genau das was auf den Objektkarten drauf stand, wurde dann genau so in das Programm eingebracht, was wir später mit den Raspberry Pis gemacht haben

- Z. 68–76:

S: Weil man das genau so ins Programm reinschreiben konnte (.) und ich dachte zum Beispiel vorher, dass man da einfach nur *Eins* oder *Null* reinschreibt oder so, aber ich wusste nicht, dass man da wirklich reinschreiben kann *einschalten* oder *blinken* oder so

I: Wie meinst du, *Eins* oder *Null* rein schreiben?

S: Ich weiß nicht, ich wusste ja vorher nicht, wie man so ein Programm programmiert. Und da dachte ich, dass man irgendwie reinschreibt *Scheinwerfer Eins* und dass der dann an ist, aber ich wusste nicht, dass es auch mit *einschalten* geht zum Beispiel

- Z. 86–89:

S: Ja, da konnte man ins Programm reinschreiben, dass die Hintergrundbeleuchtung den Helligkeitssensor befragt, ob es hell ist, und je nach dem was dann für eine Antwort kam, hat die Hintergrundbeleuchtung sich dann an- oder ausgeschaltet

- Z. 98–100:

S: [...] also beim RGB Scheinwerfer konnte man die Parameter direkt eingeben, dass rot zum Beispiel 40 Prozent hat oder so, das muss man direkt ins Programm rein schreiben [...]

- Z. 111–112:

S: [...] Ja, das wurde dann in eine Zeile geschrieben, aber ganz genau weiß ich das auch nicht mehr

- Z. 159–160:

S: Ja, ich würde den Freund halt erstmal hinzufügen, dass ich den auch kenne, und (2) weiß nicht, dann würde ich hinschreiben, *rufen Hallo* oder so

Schüler B

- Z. 99–105:

S: Also erstmal musste man glaube ich irgendwie *roter Scheinwerfer ist gleich new Scheinwerfer*, also erstmal die ganzen Objekte registrieren sage ich mal, dann musste man halt immer Objekt und dann Punkt und dann die Methode je nach dem Klammer auf, Klammer zu oder dann noch irgendwie einen Parameterwert und dann so ein Semikolon und dann in der nächsten Zeile weiter. Und ich glaube am Ende musste man noch irgendwas mit *Helper herunterfahren* oder so oder am Anfang, je nach dem

Schüler C

- Z. 32–36:

S: Wenn man jetzt zum Beispiel einen Scheinwerfer nimmt, dass man anhand eines Programms sagt, wann der angehen soll, wann der ausgeht, wie lange er anbleiben soll und so, und man kann halt da ja verschiedene Funktionen einstellen und auch gleichzeitig verschiedene Sachen schalten

- Z. 185–190:

S: Also, je nach dem wie ich den grünen Scheinwerfer, wie ich den abgespeichert habe, schreibe ich dann *grüner Scheinwerfer, Doppelpunkt, anschalten* und dann dahinter zwei Klammern, damit der Computer erkennt, dass es eine Methode ist und in die Klammern kann dann halt noch, wenn was jetzt zum Beispiel *warten* wäre oder so, dann kann dahinter halt noch das Parameterwert dann, glaube ich, zum Beispiel wie lange der warten muss

- Z. 205–209:

S: [...] Und ich persönlich fand das Objektspiel mit den Karten immer sehr verwirrend und hab das nie so ganz verstanden. Aber wenn man das dann halt in diese Groovy Konsole eingegeben hat, wurde es für mich halt immer deutlicher und ich habe es halt besser verstanden. Und da glaub ich halt schon, dass diese (.) Praxisarbeit zum Verständnis beiträgt

Schüler D

- Z. 31–32:

S: [...] Dass man diese Befehle aufschreibt und die dann ausgeführt werden vom Computer

- Z. 36:

S: [...] und am Computer halt eintippen /lacht/

- Z. 63–64:

S: Ich kann sagen, *Martin (.) setzt (.) Apfel (.) auf (.) keine Ahnung (.) /lacht/ irgendwohin*

- Z. 185–186:

S: [...] diese Groovy Konsole, wo man so Befehle eintippt und dann passiert halt irgendwas [...]

Schüler E

- Z. 30–32:

S: [...] Man schreibt ja irgendwie was, was sich direkt auf was bezieht, auf verschiedene wirkliche Sachen

UK 6.1 Oerstellung in Groovy

Schüler B

- Z. 99–100:

S: Also erstmal musste man glaube ich irgendwie *roter Scheinwerfer ist gleich new Scheinwerfer*, also erstmal die ganzen Objekte registrieren sage ich mal, [...]

- Z. 110–114:

S: Weil das vielleicht, Martin war ja jetzt sozusagen der Computer, dass der die auch kennt, und nicht einfach irgendein oder dass der auch weiß zum Beispiel, Scheinwerfer haben ja jetzt bestimmte Methoden, wenn man das dann unter Scheinwerfer registriert hat, dass der dann nur bestimmte Methoden offen hat, als wenn man das unter RGB Scheinwerfer oder Helligkeitssensor macht

Schüler C

- Z. 162–168:

I: [...] Kannst du dich ungefähr daran erinnern, wie wir das aufgeschrieben hatten?

S: Muss ich nicht erstmal erstellen, sag ich jetzt mal, dass der da ist?

I: Dass was da ist?

S: Dass der Computer weiß, dass es das Objekt gibt, und dann kann man halt den Standort setzen oder so, muss man aber nicht, und dann kann man das halt mit den Pins, wo man das eingetragen hat, schalten

Schüler D

- Z. 160–165:

S: Ja, und vorher hatten wir das ja (1) mit *new Lampe* oder so eingetragen, also *new Scheinwerfer Gleich irgendwas* und dann hat der das ja gekannt, der Computer /lacht/

I: Okay, ja, warum haben wir das gemacht?

S: Ja, damit der Computer die Objekte halt kennt. Weil die Methoden oder so waren ja schon eingespeichert

Schüler F

- Z. 103–104:

S: Ja, erstmal musst du diese ganzen natürlich anmelden, also dass diese Groovy Konsole das alles überhaupt hat [...]

HK 7: Bewertung der Arbeit

Schüler A

- Z. 168–172:

S: Mh, ich finde es eigentlich super, weil wir vorher halt auch erstmal theoretisch alles gemacht haben, damit wir halt auch erstmal einen groben Überblick haben, wie das überhaupt geht, und dass wir danach (.) an die Pis gegangen sind, wir haben ja auch ein Arbeitsblatt bekommen, und das konnten wir dann quasi Schritt für Schritt durchgehen

- Z. 180:

S: Ja, also, wir haben das einfach gemacht (1)

- Z. 184–185:

S: Ja, dass die unterschiedliche Methoden haben und dass, ja, dadurch wurde einem das noch ein bisschen näher gebracht, finde ich

Schüler B

- Z. 119–121:

S: Ja, an sich fand ich das ganz gut, nur ich fand es halt schade, dass halt, es gibt halt Leute, die hatten schon Informatik vorher, die waren dann immer schneller als die anderen und dann mussten die halt immer öfters warten

- Z. 122–125:

I: Hat das für dich zum Verständnis beigetragen, dass wir das so gemacht haben, wie wir es jetzt gemacht haben?

S: Ich weiß nicht, weil eigentlich konnte ich ja schon vorher programmieren, also...

Schüler C

- Z. 202–205:

S: Ich glaub schon, dass es mehr Vorteile gibt, weil man halt dadurch selbst auch Sachen ausprobieren kann. Und da viele Leute die Sachen halt besser verstehen, wenn man sie selbst in der Praxis umsetzt, als wenn man sie nur in der Theorie durchnimmt.

Schüler D

- Z. 181–182:

S: Also als es dann zum Raspberry Pi kam fand ich es dann schon interessant und ja, aber, ja doch, war schon ganz cool /lacht/

- Z. 191–196:

I: Würdest du sagen, das war hilfreich für das Verständnis?
 S: Ja, jetzt kann mir auf jeden Fall besser vorstellen, was man so, was Informatik so am Computer heißt, was man da so macht
 I: Und fürs Verständnis der Objektorientierung? So insgesamt?
 S: Ja, geht. Das was wir gemacht haben, habe ich jetzt so halbwegs verstanden, denke ich. (5)

Schüler E

- Z. 107–125:

S: Ich finde das hat auch wirklich Sinn gemacht, vor allen Dingen ich hab ja jetzt auch schon ein bisschen was an Programmiererfahrung, aber ich glaube, wenn man da keine hat, dann war das auf jeden Fall sinnvoll, so einzusteigen, weil wirklich, also auch den Pi hat man ja was, was man wirklich anfassen kann, angucken, man sieht vielleicht so ein bisschen sogar, was da passiert, und auch durch die Methoden, dass die dann was ausführen, dass dann wirklich was passiert, also das fand ich auf jeden Fall, ich fand das gut. Auch dass man nicht auf so viel achten musste, auf irgendwelche Klammern oder so und dabei wirklich jetzt, ich fand das schön, dass das jetzt so vereinfacht war jetzt
 I: Ja, die Groovy Konsole ist ja da nicht so streng mit den Klammern, sonst muss man da ja sehr aufpassen beim Programmieren
 S: Jaja, klar
 I: Das fandest du gut, dass die da so ein bisschen lascher war?
 S: Ja, das war ja ganz gut, also ich glaube vor allen Dingen, wenn man halt eben das zum ersten Mal macht
 I: Weswegen?
 S: Damit man erstmal den Grund sozusagen, den Hintergrund könnte man auch–, die Basis verstehen kann, und sich dann später was komplexeres erarbeiten kann

Schüler F

- Z. 145–146:

S: Ja, eigentlich ganz ok, zwischendurch war es vielleicht ein bisschen chaotisch /lacht/

UK 7.1: Bewertung der Hardware

Schüler C

- Z. 172–175:

S: Joar, also es war halt dadurch, dass wir glaube ich fünf Scheinwerfer hatten, war halt, dass man sehr oft Sachen neu eingeben musste und so, da musste man halt immer oft das gleiche machen, aber eigentlich hat es schon zum Verständnis geholfen

- Z. 208–209:

S: [...] Und da glaub ich halt schon, dass diese (.) Praxisarbeit zum Verständnis beiträgt

Schüler E

- Z. 126–129:

I: Also würdest du das sozusagen gerne weiter so machen? Damit arbeiten?

S: Ja, die Möglichkeiten sind natürlich beschränkt, aber, also ich halte zumindest für den Einstieg als sehr sinnvoll

Schüler F

- Z. 147–159:

I: [...] Hat das denn für dich zum Verständnis beigetragen, dass wir das, dass wir damit gearbeitet haben mit dem Raspberry Pi und dem, und den kleinen Lampen und dem Helligkeitssensor?

S: Ja, das auf jeden Fall. Weil da musste man erstmal gucken, was man überhaupt alles in diesen Raspberry Pi eingeben muss, weil vorher hatten wir das ja auch aufgeschrieben, aber das war dann eben nicht ganz so detailliert und da mussten wir dann eben noch alles einzeln anmelden und so weiter und so fort, ja, und außerdem konnte man dann eben auch gucken, sich selbst überprüfen, also ob man es richtig gemacht hat, indem man das dann halt ablaufen lassen hat, und das musste nicht vom Lehrer überprüft werden

I: Ja, das fandest du gut?

S: Ja

Anhang C

Bauteile der rpCollection

Da die Bauteile später u. a. über die Java-Bibliothek Pi4J angesteuert werden, ist eine spezielle Nummerierung der Pinne zu beachten. Informationen dazu (und ein Bild der Nummerierung) finden sich in Kapitel D.3.

Für die Entwicklung der Hardware-Bausteine wurde u. a. auf das Skript von Johannes PIEPER (vgl. [Pieper Raspi Skriptum, Kap. 5]) zurückgegriffen. Dort wird auch erklärt, wie die Komponenten (Diode, Phototransistor, Taster etc.) an den Pi anzuschließen sind.

- Diode (Anhang C.1, S. 224)
- Phototransistor (Anhang C.2, S. 225)
- RGB-LED (Anhang C.3, S. 226)
- Taster (Anhang C.4, S. 227)
- AD-Wandler (Anhang C.5, S. 228)
- Motor (Anhang C.6, S. 232)
- Summer (Anhang C.7, S. 234)

C.1 Bauteil 1: Diode (LED)

Dieses Bauteil ist eine einfache Diode, die unter anderem an- und ausgeschaltet werden kann. Zudem kann die Diode blinken, ihren Zustand wechseln und gefragt werden, ob sie gerade leuchtet oder nicht. Die Diode kann gut als visuelle Veranschaulichung genutzt werden.

Schaltplan Neben der Diode an sich wird lediglich ein Vorwiderstand von 330Ω benötigt. Das Bauteil wird mit zwei Kabeln an den Raspberry Pi angeschlossen. Die Diode wird im Schaltplan mit Pin 17 geschaltet.

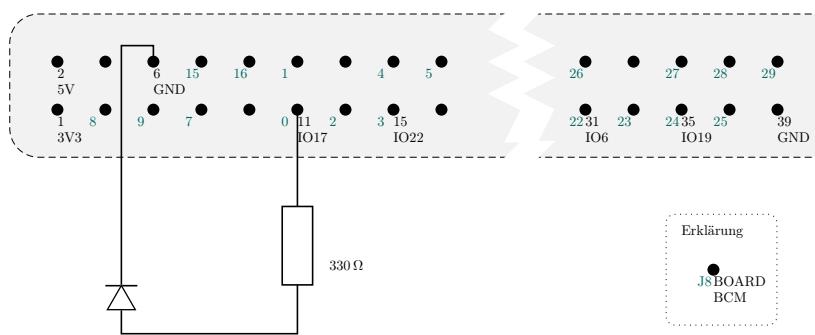


Abbildung C.1: Schaltplan der Diode am GPIO.

Verwendete Quellen: [BlueJ LED, GitHub markuskeller1960].

C.2 Bauteil 2: Phototransistor

Der Phototransistor kann gefragt werden, ob gerade Licht auf ihn scheint oder nicht. So können Lichtschranken gebaut werden oder Prozesse bei Lichteinfall gestartet oder gestoppt werden.

Schaltplan Der Anschluss eines Phototransistors erfolgt analog zum Anschluss eines Tasters (Anhang C.4). Im Schaltplan liegt das Signal an Pin 22 an.

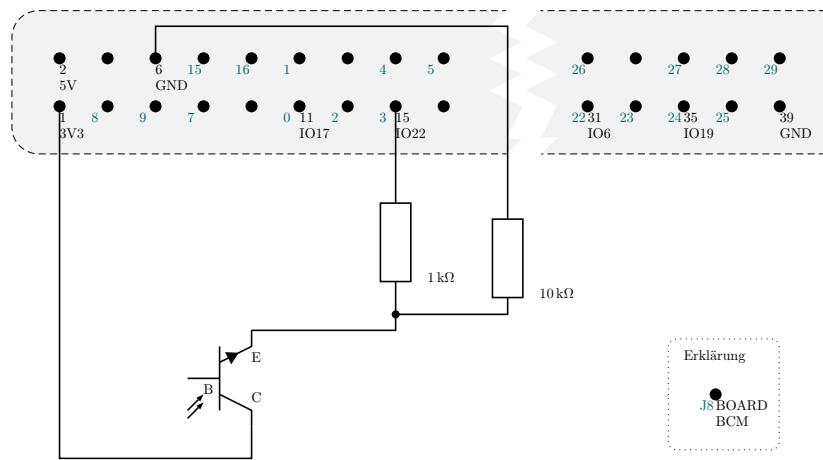


Abbildung C.2: Schaltplan des Phototransistors am GPIO.

C.3 Bauteil 3: RGB-LED

Die RGB-LED hat prinzipiell die gleichen Methoden wie die LED (anschalten, ausschalten, blinken). Hinzu kommt die Möglichkeit, eine Farbe einzustellen. In dem Bauteil der RGB-LED sind drei LEDs für die Farben »rot«, »grün« und »blau« verbaut. Da jedem Farbkanal eine Intensität von 0 (aus) bis 255 (volle Leuchtkraft) zugeordnet werden kann, können ca. 16,7 Millionen Farben dargestellt werden.

Softwareseitig geschieht die Darstellung der Farben über eine Pulsweitenmodulation der einzelnen LEDs. An sich kann der Raspberry Pi nur ein binäres Signal (an / aus) an die Diode senden. Wird aber bspw. die grüne LED wiederholt für einen kurzen Zeitraum eingeschaltet, entsteht für das Auge der Effekt, die LED leuchte nur mit halber Kraft. Der Zeitraum, in dem die LED an ist, wird als »Phase« bezeichnet. Über eine Pulsweitenmodulation der drei Farbkanäle lässt sich die Farbe beliebig einstellen.

Schaltplan Die RGB-LED wird mit drei Kabeln für die Farbkanäle und einem Kabel für Ground am Raspberry Pi angeschlossen. Es wurden 220Ω Widerstände benutzt, damit die Farben etwas mehr Leuchtkraft besitzen (im Gegensatz zur normalen Diode). Die Diode wird im Schaltplan über die Pinne 17, 27 und 22 angesprochen.

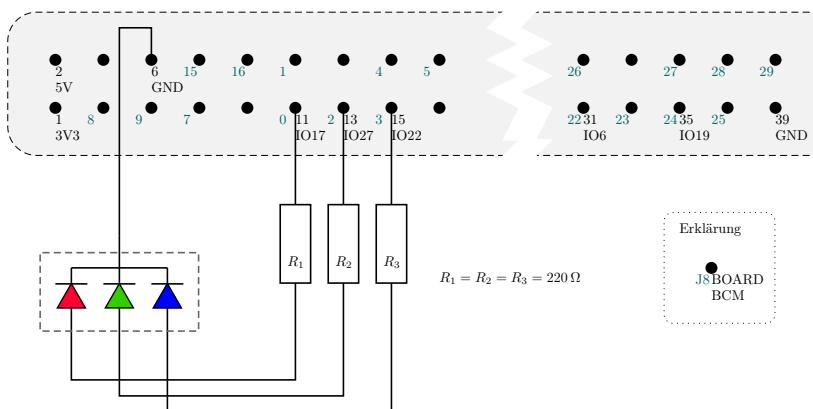


Abbildung C.3: Schaltplan der RGB-LED am GPIO.

Verwendete Quellen: [BlueJ Adjustable LED, GitHub OlivierLD].

C.4 Bauteil 4: Taster

Der Taster kann gefragt werden, ob er gerade gedrückt ist oder nicht. Es handelt sich nicht um einen Schalter. Das heißt, dass der Taster nach Betätigung (bzw. Loslassen) wieder in seinen Ausgangszustand zurückkehrt und nicht in dem Zustand »gedrückt« verweilt. Am einfachsten ist der Vergleich zu einer Klingel, die so lange klingelt, wie sie gedrückt ist. Im Vergleich dazu hat der klassische Lichtschalter zwei Zustände, in denen er jeweils nach Betätigung verweilt.

Schaltplan Der Taster funktioniert ähnlich wie der Phototransistor (Anhang C.2). Das Signal liegt im Schaltplan an Pin 27 an.

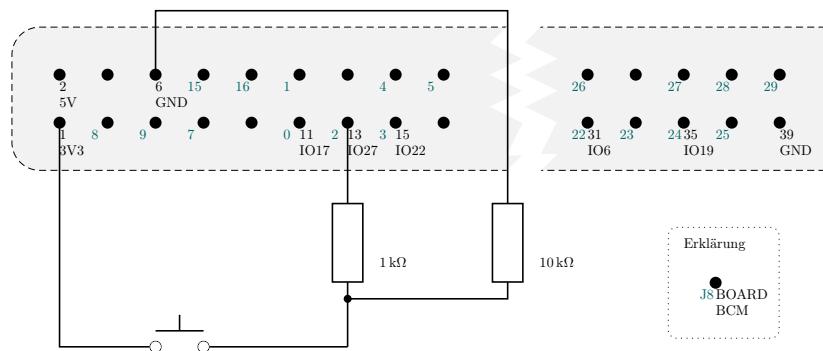


Abbildung C.4: Schaltplan des Tasters am GPIO.

Verwendete Quellen: [BlueJ Button, GitHub markuskeller1960].

C.5 Bauteil 5: Analog-Digital-Wandler und Regler

Da der Raspberry Pi eigentlich nur digitale Signale über die Pinne auslesen kann, ist ein AD-Wandler notwendig, um auch »Zwischenwerte« auslesen zu können. Der verwendete AD-Wandler ist ein »MCP3208« mit einer Auflösung von 12 Bit (Abbildung C.5), d. h. er kann $2^{12} = 4096$ Werte an den Raspberry Pi übermitteln.

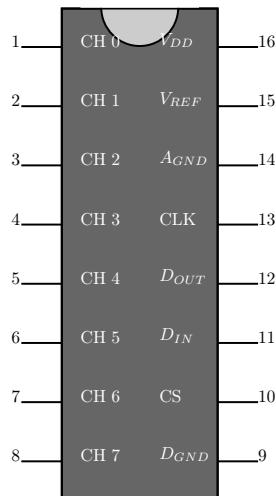


Abbildung C.5: Anschlüsse des AD-Wandlers MCP3008 bzw. MCP3208.

AD-Wandler kommen zum Beispiel dann zum Einsatz, wenn die Stellung eines Reglers ausgelesen werden soll oder ein Thermometer angeschlossen wird, da sich diese Signale nicht als ausschließlich 0 oder 1 übertragen lassen.

Es ist ebenfalls möglich, einen »MCP3200« mit einer Auflösung von 10 Bit mit den Klassen der `rpCollection` zu benutzen. Dazu muss bei der Klasse `Helper` die (statische) Methode `setzeADWandler("MCP3200")` aufgerufen werden. Standardmäßig wird aber der MCP3208 genutzt.

Softwareanpassungen Für den Raspberry Pi muss das »SPI-Protokoll« aktiviert werden, damit der AD-Wandler genutzt werden kann. Dies sorgt dafür, dass der AD-Wandler über vordefinierte Pinne ausgelesen werden kann (Pinne 10, 9, 11 und 8), die z. B. den Takt für den AD-Wandler vorgeben. Mehr Informationen findet man in [Pieper Raspi Skriptum, S. 35f]. Das Aktivieren des SPI-Protokolls erfolgt durch Anhängen der Zeile

```
1 dtparam=spi=on
```

in der Datei `/boot/config.txt` oder kann über die graphische Benutzeroberfläche über die Raspberry-Pi-Konfiguration (»raspi-config«) eingestellt werden. Nach einem Neustart kann der AD-Wandler dann verwendet werden.

Schaltplan Der AD-Wandler wird an den Pinnen 3V3, GND, 10, 9, 11 und 8 angeschlossen. Eine genauere, tabellarische Zuordnung der anzuschließenden Pinne findet man in [Pieper Raspi Skriptum, S. 36]. Da auf den Bauteilen die anzuschließen Pinne aber aufgedruckt sind, soll auf dieses Thema an dieser Stelle nicht weiter eingegangen werden.

Insgesamt lassen sich an beide AD-Wandler 8 Bauteile anschließen (an die Channel 0 bis 7), deren Stellung / Zustand sich dann über das SPI-Protokoll auslesen lässt. Für die Schülerinnen und Schüler wurden vier Potentiometer an einen AD-Wandler angeschlossen. Potentiometer sind kleine, einstellbare Widerstände, die mit einem Drehregler ausgestattet sind (s. Abbildung C.10 (e)). Die Stellung des Reglers lässt sich für jeden Regler einzeln auslesen. Die Auflösung des AD-Wandlers gibt dabei die Genauigkeit vor, mit der dies geschehen kann.

- Stellung des Potentiometers auf 0%: Der AD-Wandler liefert 0 zurück.
- Stellung des Potentiometers auf 100%: Der AD-Wandler liefert $2^{12} - 1 = 4095$ zurück¹.

Im Schaltplan ist ein Potentiometer an den Channel 0 des MCP3208 angeschlossen.

¹ Bei einem AD-Wandler des Typs MCP3208.

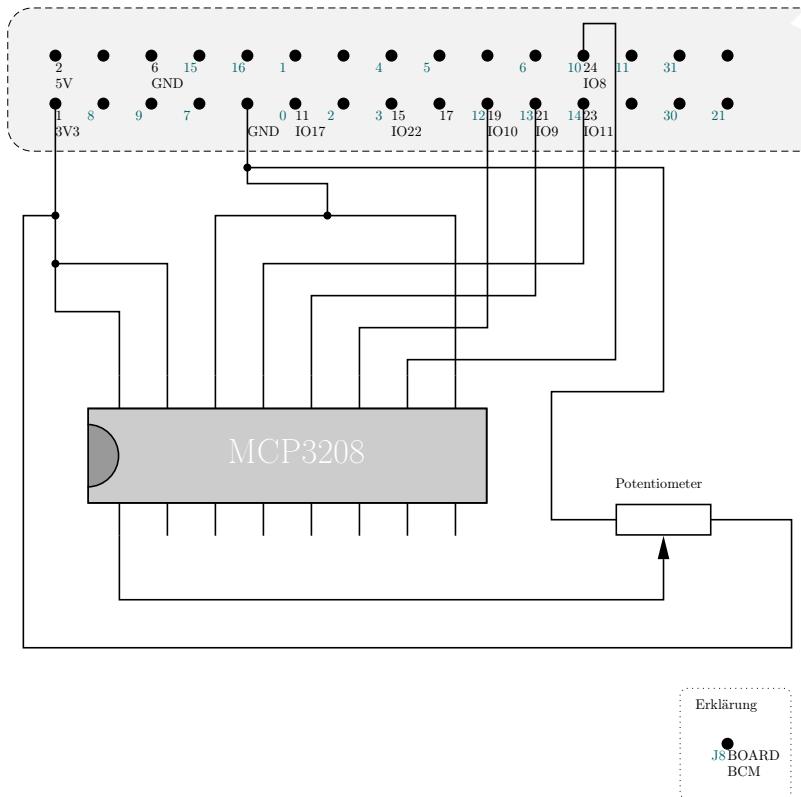


Abbildung C.6: Schaltplan des AD-Wandlers mit Potentiometer am GPIO.

Die Klasse RPRegler Um die Stellung eines Reglers auslesen zu können, muss ein Objekt der Klasse `RPRegler` erstellt werden. Bei diesem Objekt wird entweder über den Konstruktor oder über eine separate Methode der Wert für den Channel gesetzt, an welchem der Regler am AD-Wandler angeschlossen ist.

Dieses Objekt muss dann dem Objekt der Klasse `RPADWandler` beim Auslesen (also dem Aufruf der Methode `gibWertVonRegler(RPRegler pRegler)`) übergeben werden. Dadurch ist sichergestellt, dass nicht bloß eine Zahl für den Channel übergeben wird (Quelltext C.1, Zeile 3), sondern ein Objekt. So wird auch beim Auslesen ein konsequent objektorientierter Ansatz verfolgt.

Auch wenn sich alle vier Regler auf derselben Platine befinden (und damit gewissermaßen eine Einheit bilden), wird die Idee, dass ein virtuelles Objekt real erfahrbar ist, weiter verfolgt. Es interessiert nicht so sehr der AD-Wandler, sondern viel mehr die Stellung eines Reglers. Objekte der Klasse `RPRegler` stellen also den Zugang zum Auslesen der Analogwerte dar (Quelltext C.1, Zeile 8f). Dass dafür ein AD-Wandler benötigt wird, hat physikalische Gründe und ist auch die Begründung

für die zwingende Existenz eines RPADWandler-Objekts. Dies ist aber von geringerem Interesse.

```
1 // Auslesen von Channel 0 ohne die Klasse RPRegler
2 meinADW = new RPADWandler();
3 meinADW.gibWertVonChannel(0);
4
5
6 // Auslesen von Channel 0 mit der Klasse RPRegler
7 meinZweiterADW = new RPADWandler();
8 geschwindigkeitsregler = new RPRegler(0);
9 meinZweiterADW.gibWertVonRegler(geschwindigkeitsregler);
```

Quelltext C.1: Auslesen des AD-Wandlers mit und ohne Einsatz der Klasse RPRegler.

Verwendete Quellen (Auslesen des Potentiometers): [Adafruit, GitHub oksbwn, Knight of Pi, Sorensen 2016, LeDiouris].

C.6 Bauteil 6: Motor

Am Raspberry Pi können auch Fischertechnik-Motoren angeschlossen werden. Dies sind kleine Elektromotoren, die sich in zwei Richtungen drehen können. Der Raspberry Pi übernimmt dabei die Steuerung der Drehrichtung und Drehgeschwindigkeit.

Motorenchip Motoren können am Raspberry Pi über den Steuerchip »L293D« gesteuert werden (Abbildung C.7). Dieser Chip kann zwei Motoren gleichzeitig bedienen.

Motoren müssen zunächst eingeschaltet und können dann durch Anlegen einer Steuerspannung aktiviert werden, sodass sich der Motor tatsächlich dreht. Dafür stehen die Pinne `In 1.1` bzw. `In 1.2` am Chip zur Verfügung, wovon immer nur an ausschließlich einem von beiden Strom anliegen darf. Die beiden Pinne bestimmen dann die Drehrichtung des Motors (deswegen ist auch nicht sinnvoll, an beiden Pinnen eine Spannung anzulegen).

Ausführlichere Erklärungen zum Anschluss eines Motors finden sich in [Pieper Raspi Skriptum, S. 32f]. Die Drehgeschwindigkeit eines Motors kann über eine Pulsweitenmodulation der Steuerpinne `In 1.1` bzw. `In 1.2` geschehen.

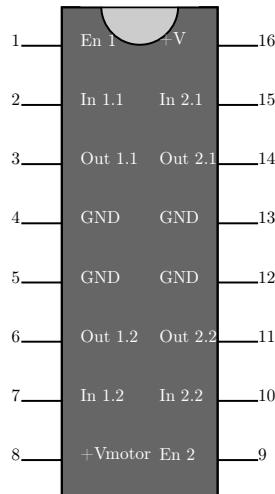


Abbildung C.7: Motorenchip L293D.

Schaltplan Im Schaltplan ist ein Motor an den L293D angeschlossen. Dabei ist eine externe Stromquelle erforderlich, da die Motoren eine höhere Betriebsspannung voraussetzen, als der Raspberry Pi liefern kann.

Über den Pin 18 wird der Motor eingeschaltet und durch Anlegen von Spannung an den Pinnen 23 oder 24 dann aktiviert (der Elektromotor dreht sich).

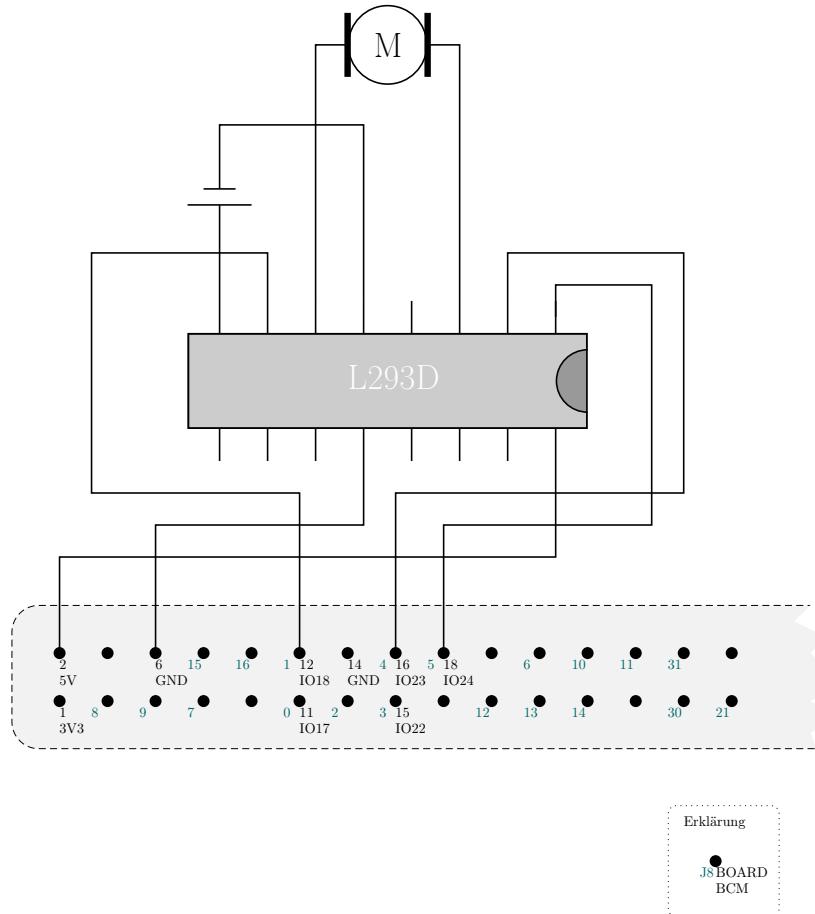


Abbildung C.8: Schaltplan des Motors am GPIO.

C.7 Bauteil 7: Summer

An den Raspberry Pi kann auch direkt ein Summer angeschlossen werden. Dieser wird für akustische Rückmeldungen genutzt.

Schaltplan Der Summer wird direkt mit einem Pin und Ground verbunden und kann dann wie eine LED geschaltet werden. Im Schaltplan ist er an Pin 17 angeschlossen.

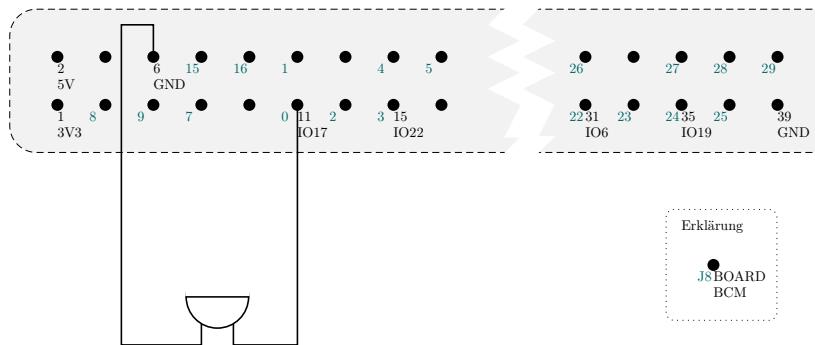
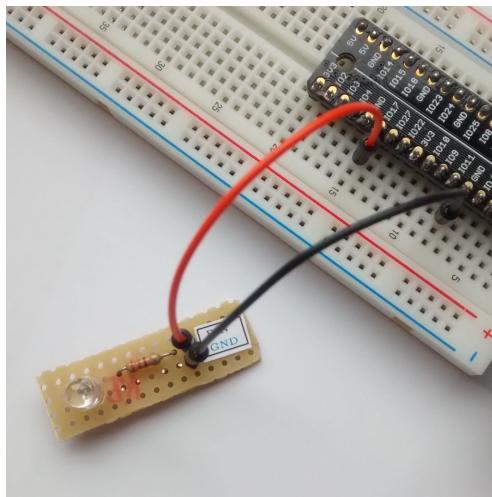
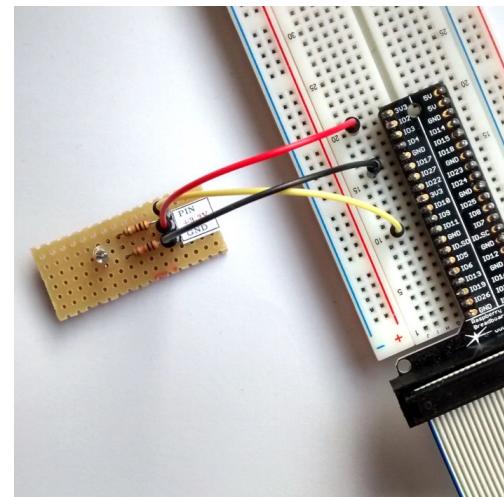


Abbildung C.9: Schaltplan des Summers am GPIO.

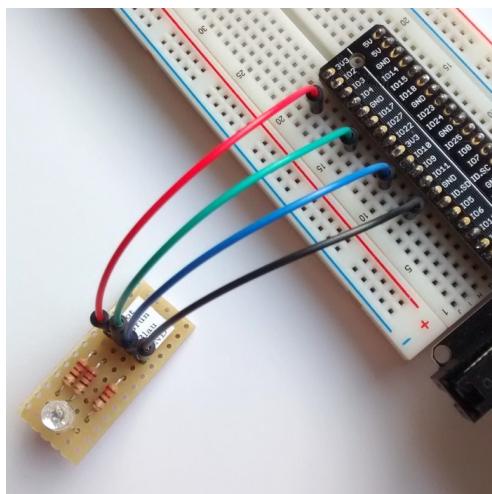
C.8 Fotos der angeschlossenen Bausteine



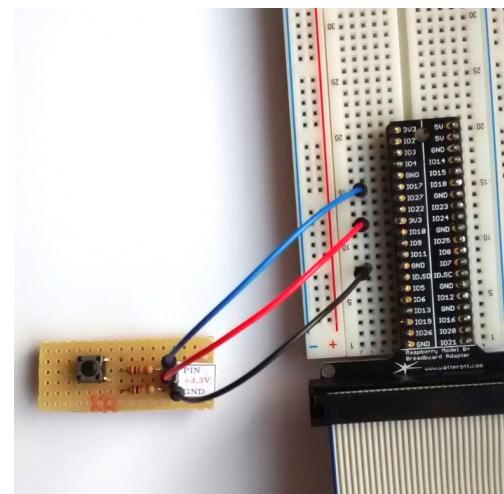
(a) Diode.



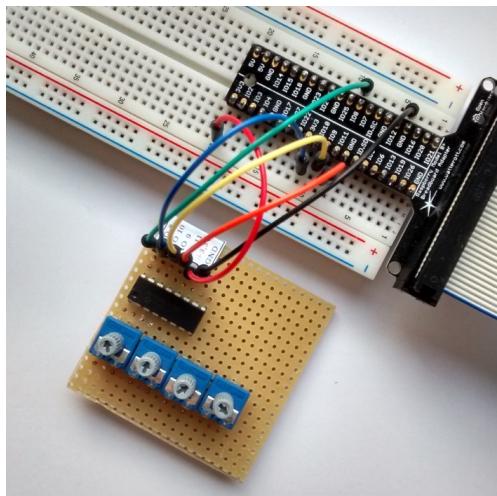
(b) Phototransistor.



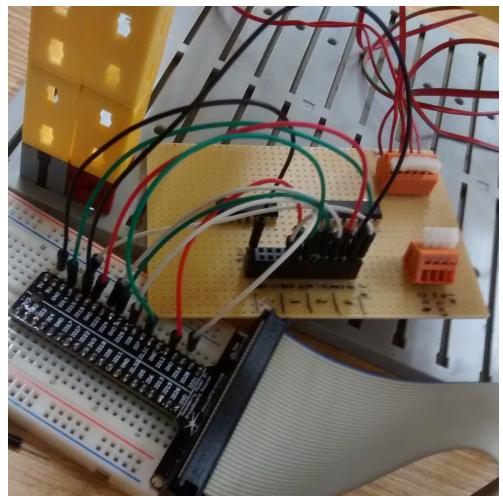
(c) RGB-LED.



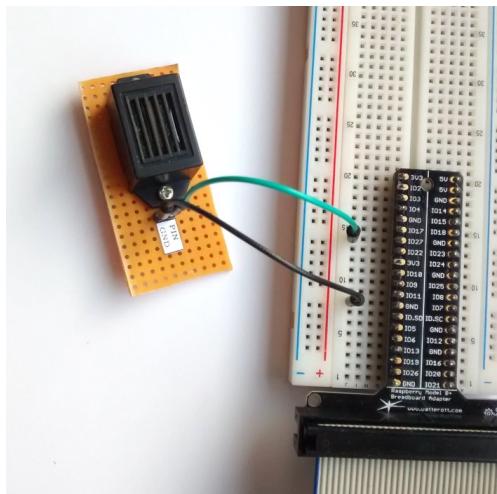
(d) Taster.



(e) AD-Wandler mit vier Reglern.



(f) Motor (Fischer-Technik).



(g) Summer.

Abbildung C.10: Fotos der am Raspberry Pi angeschlossenen Bauteile.

Anhang D

Verwendete Software

D.1 BlueJ

BlueJ¹ ist eine leichtgewichtige Programmierumgebung, die heutzutage in viele Schulen Einzug erhalten hat. Die Software wird u. a. von Michael KÖLLING (University of Kent, Großbritannien) und John ROSENBERG (La Trobe University Australien) entwickelt.

Mit [Barnes Kölling 2002] haben die BlueJ-Entwickler ein Lehrbuch vorgelegt, das auch auf den »Objects First«-Ansatz setzt. BlueJ ist auf dem Raspberry Pi bereits vorinstalliert.

D.2 Geany

Geany² ist eine im Vergleich zu BlueJ umfangreichere IDE, mit der sich komfortabel programmieren lässt. Neben einer übersichtlichen Tab-Darstellung offener Dateien hat Geany auch ein integriertes Terminal, sodass direkt im Verzeichnis auf der Kommandozeile gearbeitet werden kann.

Geany hat den Vorteil, dass man den *Compile*-Befehl anpassen kann, sodass die Java-Dateien direkt für die zur Groovy-Console passenden Java-Version übersetzt werden. Die Einstellung dazu findet man im Menü unter *Build → Set Build Commands*.

Dort ist unter den *Java commands* ein neuer Command einzutragen, bspw. *Pi4J Groovy Compile*. Der Befehl lautet dann wie folgt:

```
1  javac -classpath '.:classes:/opt/pi4j/lib/*' "\%f"
```

¹ BlueJ – <http://bluej.org/>.

² Geany – <http://geany.org/>.

Weitere Informationen dazu sind auch in Kapitel 3.4 zu finden. Geany ist beim Raspberry Pi bereits vorinstalliert. Weitere Informationen finden sich auf der Projekt-Website.

D.3 Pi4J

Installation von Pi4J Pi4J³ ist eine Bibliothek, die es ermöglicht, am Raspberry Pi mittels Java Pinne am GPIO anzusteuern. Sie ist quelloffen und wird derzeit unter der Federführung von Robert SAVAGE und Daniel SENDULA entwickelt [Pi4J Team List]. Standardmäßig sind die (neueren) Raspbian-Versionen (das Betriebssystem des Raspberry Pi) mit Pi4J ausgestattet, sodass eine Installation nicht nötig ist. Sollte Pi4J noch installiert werden müssen, finden sich die nötigen Informationen zur Installation auf der Projekt-Website⁴.

D.3.1 Pin-Nummerierungen

Für die Arbeit mit den Baustein ist es essentiell, sie mit richtigen Pin-Nummern anzusprechen. Es gibt leider keine einheitliche Nummerierung oder Zählweise der Pinne, sodass es hier schnell zu Verwechslungen kommen kann. Insgesamt werden drei Nummerierungen unterschieden:

- **BOARD** Bei der BOARD-Nummerierung sind die Pinne von 1 bis 40 durchnummert (für einen Raspberry Pi 3B). Diese Nummerierung wird von der rpCollection nicht unterstützt.
- **BCM** Die BCM-Nummerierung ist die auf dem GPIO-Controller aufgedruckte Nummerierung. Dieses Belegungsmuster wird standardmäßig bei der rpCollection benutzt, da dies das einfache Anschließen und Adressieren von Bauteilen ermöglicht.
- **J8** Die J8-Nummerierung ist die Standardnummerierung für die Pi4J-Bibliothek und wird in Abbildung D.1 für den Raspberry Pi 3B dargestellt. Sie kann auch bei der rpCollection genutzt werden. Eine Tabelle, die zwischen BCM- und J8-Nummerierung übersetzt, findet sich in Tabelle D.1. Um in der rpCollection die Standardbelegung zu wechseln, muss bei der Klasse `Helper` die (statische) Methode `setzePinLayout("J8")` aufgerufen werden.

³ The Pi4J Project – Java I/O library for the Raspberry Pi – <http://pi4j.com/>.

⁴ Pi4J Installation – <http://pi4j.com/install.html>.

| J8 | BCM | | BCM | J8 |
|----|-------|-----|-------|----|
| – | 3V3 | • • | 5V | – |
| 8 | IO2 | • • | 5V | – |
| 9 | IO3 | • • | GND | – |
| 7 | IO4 | • • | IO14 | 15 |
| – | GND | • • | IO15 | 16 |
| 0 | IO17 | • • | IO18 | 1 |
| 2 | IO27 | • • | GND | – |
| 3 | IO22 | • • | IO23 | 4 |
| – | 3V3 | • • | IO24 | 5 |
| 12 | IO10 | • • | GND | – |
| 13 | IO9 | • • | IO25 | 6 |
| 14 | IO11 | • • | IO8 | 10 |
| – | GND | • • | IO7 | 11 |
| 30 | ID.SD | • • | ID.SC | 31 |
| 21 | IO5 | • • | GND | – |
| 22 | IO6 | • • | IO12 | 26 |
| 23 | IO13 | • • | GND | – |
| 24 | IO19 | • • | IO16 | 27 |
| 25 | IO26 | • • | IO20 | 28 |
| – | GND | • • | IO21 | 29 |

Tabelle D.1: Zählweisen der Pinne nach J8- und BCM-Nummerierung.

D.3.2 Belegte Pinne freigeben

Um gespeicherte Skripte in Groovy auch mehrfach aufrufen zu können, ist es nötig, die belegten Pinne am Ende des Skripts frei zu geben, sodass sie beim erneuten Aufruf wieder zur Verfügung stehen.

Pinne können freigegeben werden, wenn bei erstellten Objekten die Methode `herunterfahren()` aufgerufen wird⁵. Wird

```
1 Helfer.herunterfahren();
```

aufgerufen, werden *alle* vergebenen Pinne freigegeben. Dabei handelt es sich um eine statische Methode, sodass kein Objekt der Klasse `Helfer` nötig ist.

⁵ Nicht jedes Objekt muss unbedingt heruntergefahren werden. Eine Ausnahme bildet bspw. der AD-Wandler.

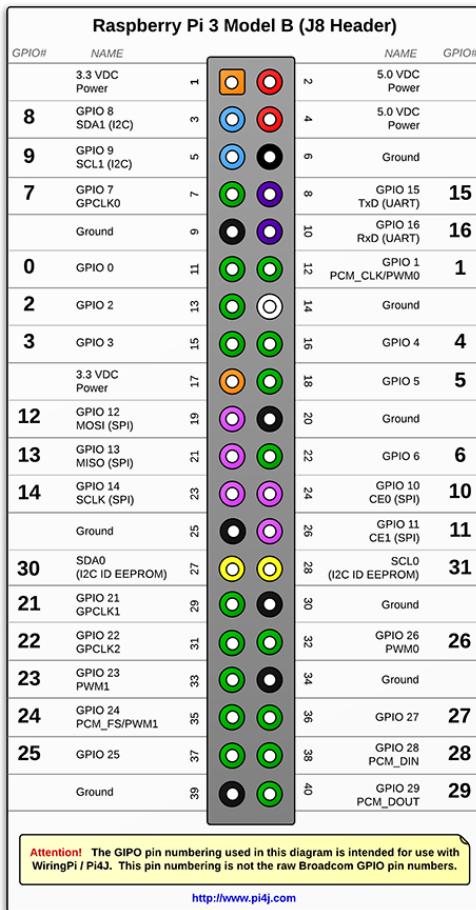


Abbildung D.1: J8-Nummerierung der Pinne am Raspberry Pi 3B. Die Pin-Belegung für andere Raspberry-Pi-Versionen finden sich auf der Projekt-Website. Bildquelle: [Pi4J Pins].

D.4 Wiring Pi

Wiring Pi⁶ kommt für die Pulsweitenmodulation der RGB-LEDs und das Steuern der Motoren zum Einsatz. In Wiring Pi kann man sehr einfach Pinne modulieren (s. Quelltexte der Klassen `RPRGB` und `RPMotor` im Git-Repository, vgl. Anhang E).

D.5 Groovy

Groovy⁷ ist eine eigene Programmiersprache, die eine nahtlose Java-Einbindung ermöglicht. Sie hat eine leicht vereinfachte Java-Syntax und ist ungetypt. Das hat den Vorteil, dass Variablen einfach mit `var = 12` deklariert und initialisiert werden können. Zeilen müssen zudem nicht unbedingt mit einem Semikolon beendet werden.

⁶ Wiring Pi – GPIO Interface library for the Raspberry Pi – <http://wiringpi.com/>.

⁷ Groovy – The Groovy programming language – <http://www.groovy-lang.org/>.

Für diese Masterarbeit sollen die Groovy-Syntax oder Vor- und Nachteile von Groovy nicht weiter interessieren, sondern nur die Tatsache, dass innerhalb von Groovy wie mit Java programmiert werden kann.

D.5.1 Die Groovy-Shell (`groovysh`)

Die Groovy-Shell bietet die Möglichkeit, mit Groovy auf der Kommandozeile zu arbeiten. Das User-Interface ist entsprechend einfach gestaltet (vgl. Abbildung 3.7, S. 47). Eingegebene Befehle werden direkt übersetzt und ausgeführt; Fehlermeldungen werden direkt angezeigt.

Starten der Groovy-Shell Die Groovy-Shell wird durch den Aufruf von

```
1 groovysh
```

gestartet. Für die Verwendung mit der rpCollection muss der Aufruf aber wie in Kapitel 3.4 geschehen.

Laden von Groovy-Dateien In der Groovy-Shell können .groovy-Dateien geladen werden. Dies sind einfache Text-Dateien, die mit jedem Editor bearbeitet werden können. Befinden sich die .groovy-Dateien im selben Verzeichnis, in dem auch die Groovy-Shell gestartet wurde, lädt der Befehl

```
1 :load my_groovy_script.groovy
```

ein Skript und führt es sofort aus.

Hilfe Über den Groovy-Shell-Befehl

```
1 :help
```

kann die Groovy-Shell-Hilfe eingeblendet werden, die einen Überblick über weitere Möglichkeiten bietet.

Beenden der Groovy-Shell Mit dem Groovy-Shell-Befehl

```
1 :exit
```

wird die Groovy-Shell beendet.

D.5.2 Die Groovy-Console (`groovyConsole`)

Die Groovy-Console bietet im Prinzip die gleichen Möglichkeiten wie die Groovy-Shell, allerdings mit dem Unterschied, dass sie auch ein ansprechendes User-Interface

bereitstellt. Über das User-Interface (vgl. Abbildung 3.6, S. 46) können dann .groovy-Dateien geladen, ausgeführt und gespeichert werden. Dabei ist das Fenster zweigeteilt: im oberen Bereich erfolgt die Eingabe von Befehlen, im unteren Teil (gelb) erfolgt die Ausgabe der Konsole.

Starten der Groovy-Console Die Groovy-Console wird durch den einfachen Aufruf von

```
1 groovyConsole
```

gestartet.

Für die Verwendung mit der rpCollection muss der Aufruf aber wie in Kapitel 3.4 durch Einbinden der Pi4J-Bibliothek geschehen:

```
1 sudo groovyConsole -classpath ':./classes:/opt/pi4j/lib/*'
```

Anhang E

Materialien der rpCollection

Die Quelltexte sind in einem Git-Repository unter der Adresse

<https://github.com/hnrstrck/rpCollection>

veröffentlicht. Ebenfalls finden sich dort die Dateien zum Etikettendruck sowie die verwendeten Arbeitsblätter für die Schule.

Bitte beachten Sie dabei die Lizenzvereinbarungen: Sofern nicht anders angegeben, stehen sämtliche Materialien der »rpCollection« unter einer »Namensnennung – Nicht-kommerziell – Weitergabe unter gleichen Bedingungen 4.0 International«-Lizenz. Maßgeblich für die Lizenzangaben sind aber die genaueren Hinweise im Git-Repository.

Die Bedingungen der Lizenz können unter folgendem Link eingesehen werden:

<http://creativecommons.org/licenses/by-nc-sa/4.0/deed.de>

Installation auf dem Raspberry Pi Die Installation geschieht mit folgendem Befehl:

```
1 git clone https://github.com/hnrstrck/rpCollection
```

Dabei wird ein Verzeichnis `rpCollection` erstellt, in das alle Dateien geladen werden. Dafür ist natürlich eine Internetverbindung erforderlich.

Aktualisierung der Installation Der Befehl

```
1 git pull
```

aktualisiert die Installation (Internetverbindung erforderlich).

Eidesstattliche Versicherung

Name, Vorname

Matr.-Nr.

Ich versichere hiermit an Eides statt, dass ich die vorliegende Bachelorarbeit/Masterarbeit* mit dem Titel

selbstständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Ort, Datum

Unterschrift

*Nichtzutreffendes bitte streichen

Belehrung:

Wer vorsätzlich gegen eine die Täuschung über Prüfungsleistungen betreffende Regelung einer Hochschulprüfungsordnung verstößt, handelt ordnungswidrig. Die Ordnungswidrigkeit kann mit einer Geldbuße von bis zu 50.000,00 € geahndet werden. Zuständige Verwaltungsbehörde für die Verfolgung und Ahndung von Ordnungswidrigkeiten ist der Kanzler/die Kanzlerin der Technischen Universität Dortmund. Im Falle eines mehrfachen oder sonstigen schwerwiegenden Täuschungsversuches kann der Prüfling zudem exmatrikuliert werden. (§ 63 Abs. 5 Hochschulgesetz - HG -)

Die Abgabe einer falschen Versicherung an Eides statt wird mit Freiheitsstrafe bis zu 3 Jahren oder mit Geldstrafe bestraft.

Die Technische Universität Dortmund wird gfls. elektronische Vergleichswerkzeuge (wie z.B. die Software „turnitin“) zur Überprüfung von Ordnungswidrigkeiten in Prüfungsverfahren nutzen.

Die oben stehende Belehrung habe ich zur Kenntnis genommen:

Ort, Datum

Unterschrift

Dieses PDF wurde unter der Adresse https://github.com/hnrstrck/Masterarbeit_00M veröffentlicht.
PDF gesetzt:20. April 2017