Android-Workshop zur Vertiefung von Datenstrukturen in der Studieneingangsphase

Schriftliche Hausarbeit im Rahmen der Ersten Staatsprüfung, dem Landesprüfungsamt für Erste Staatsprüfungen für Lehrämter an Schulen vorgelegt von:

Manuel Froitzheim

Aachen, den 14. August 2013

Gutachter: Univ.-Prof. Dr.-Ing. Ulrik Schroeder
Betreuer: Dipl.-Gyml. Nadine Bergner
Dipl.-Inform. Hendrik Thüs
Institut: Lehr- und Forschungsgebiet Informatik 9
RWTH Aachen University

Inhaltsverzeichnis

1 Einleitung **1**

2 Motivation **4**

2.1 Komponenten der Motivationsförderung 4

2.1.1 Computerspiel 5

2.1.2 Mobiles Endgerät 6

2.2 Kompetenzvermittlung in der Studieneingangsphase 7

2.3 Wissensrepräsentation 9

3 Theoretischer Hintergrund **11**

3.1 Ausgangslage und Ziele 11

3.2 Vorwissen 13

3.2.1 Formale Kenntnisse 13

3.2.2 Praktische Fähigkeiten 18

3.3 Thesen 18

4 Konzeption und Analyse **20**

4.1 Voraussetzungen 20

4.2 Hintergrund zur Auswahl des Inhalts der App . . . 21

4.3 Aufbau des Workshops 24

4.3.1 SEE-Phase 26
4.3.2 TRY-Phase 28
4.3.3 DO-Phase 30
4.3.4 EXPLAIN-Phase 33
4.4 Implementierung der Android-App 34
4.4.1 Aufbau des Programms 34
4.4.2 Spiellogik 38
4.5 Beschreibung und Analyse der Lernmaterialien . . . 42
4.5.1 Spielbrett 42
4.5.2 Arbeitsmappe 42
4.5.3 Dokumentation 46

5 Evaluation **47**
5.1 Methodik 47
5.2 Auswertung 48
5.2.1 Fragebogen 48
5.2.2 Befragung der Studenten 55
5.2.3 Bewertung der Arbeitsergebnisse 56
5.2.4 Eindrücke des Autors 58
5.3 Schlussfolgerungen 59

6 Fazit und Ausblick **62**

A Lerneinheit **69**
A.1 Ankündigungsplakat 69
A.2 Materialliste 70
A.3 Zeitplan 71

B Arbeitsmaterialien **72**

B.1 Arbeitsblätter 72

B.1.1 Spielbrett 73

B.1.2 Spielregeln 74

B.1.3 Entwicklungsumgebung „Eclipse“ 75

B.1.4 Go-Moku Programmgerüst 77

B.1.5 Klasse GomokuBoard für das Spielbrett . . 78

B.1.6 Klasse FieldArray für die Spielstrategie . . . 80

B.1.7 Spielstrategie 82

B.1.8 Der Wettkampf 85

B.2 Ergänzungen 86

B.2.1 Ergänzung zu Arbeitsauftrag 8 86

B.2.2 Ergänzung zu Arbeitsauftrag 9 87

B.3 Musterlösungen 88

C Quelltext der Android App **93**

D Spielauswahl **94**

E Evaluierung **112**

E.1 Fragebogen 112

E.1.1 Vor-Test 113

E.1.2 Nach-Test 121

E.2 Auswertung 126

E.2.1 Stammdaten 126

E.2.2 Zufriedenheit mit dem Studium 127

E.2.3 Lernen im Studium 129

E.2.4 Zuversicht gegenüber Herausforderungen . . 130
E.2.5 Schulkenntnisse in der Informatik 132
E.2.6 Fachliche Kenntnisse 134
E.2.7 Bewertung des Workshops 135
E.2.8 Leistung der Studenten im Workshop 138

F Erklärung **140**

Abbildungsverzeichnis

3.1 Diagramm: Schwundquote RWTH 12

4.1 Ablaufplan des Workshops 27
4.2 Bildschirmfotos der App 36

5.1 Diagramm: Auswertung der Stammdaten 50
5.2 Diagramm: Kenntnis der Programmiersprachen . 54
5.3 Diagramm: Punkte der Arbeitsgruppen 57

A.1 Ankündigungsplakat für den Workshop 69

B.1 Arbeitsblatt: Spielbrett 73
B.2 Arbeitsblatt: Spielregeln 74
B.3 Arbeitsblatt: Entwicklungsumgebung „Eclipse“ I . 75
B.4 Arbeitsblatt: Entwicklungsumgebung „Eclipse“ II 76
B.5 Arbeitsblatt: Go-Moku Programmgerüst 77
B.6 Arbeitsblatt: Programmierung der Klasse Gomku-Board für das Spielbrett 78
B.7 Arbeitsblatt: Programmierung der Klasse Gomku-Board für das Spielbrett 79

B.8 Arbeitsblatt: Programmierung der Klasse FieldArray für die Spielstrategie I 80
B.9 Arbeitsblatt: Programmierung der Klasse FieldArray für die Spielstrategie II 81
B.10 Arbeitsblatt: Spielstrategie I 82
B.11 Arbeitsblatt: Spielstrategie II 83
B.12 Arbeitsblatt: Spielstrategie III 84
B.13 Arbeitsblatt: Der Wettkampf 85
B.14 Arbeitsblatt: Ergänzung zu Arbeitsauftrag 8 . . . 86
B.15 Arbeitsblatt: Ergänzung zu Arbeitsauftrag 9 . . . 87
B.16 Arbeitsblatt: Musterlösung I 88
B.17 Arbeitsblatt: Musterlösung II 89
B.18 Arbeitsblatt: Musterlösung III 90
B.19 Arbeitsblatt: Musterlösung IV 91
B.20 Arbeitsblatt: Musterlösung V 92

E.1 Pre-Frage: Persönlicher Hintergrund 113
E.2 Pre-Frage: Nationalität 113
E.3 Pre-Frage: Muttersprache 113
E.4 Pre-Frage: Hochschulabschluss der Eltern 114
E.5 Pre-Frage: Studiengang 114
E.6 Pre-Frage: Studiensemester 114
E.7 Pre-Frage: Studienzufriedenheit 115
E.8 Pre-Frage: Lernen im Studium 115
E.9 Pre-Frage: Umgang mit Problemen 116
E.10 Pre-Frage: Schulinformatik 116

E.11 Pre-Frage: Leistungskurs 117
E.12 Pre-Frage: Umgang mit Programmiersprachen . . 118
E.13 Pre-Frage: Java-Wissen I 118
E.14 Pre-Frage: Java-Wissen II 119
E.15 Pre-Frage: Java-Wissen III 119
E.16 Pre-Frage: Java-Wissen IV 120
E.17 Pre-Frage: Verständnis Algorithmus 120
E.18 Post-Frage: Workshop-Inhalte 121
E.19 Post-Frage: Studienzufriedenheit 121
E.20 Post-Frage: Lernen im Studium 122
E.21 Post-Frage: Umgang mit Problemen 122
E.22 Post-Frage: Schwierigkeitsgrad 123
E.23 Post-Frage: Ermutigung Studienwahl 123
E.24 Post-Frage: Benotung Workshop 123
E.25 Post-Frage: Verbesserungsvorschläge 123
E.26 Post-Frage: Java-Wissen I 124
E.27 Post-Frage: Java-Wissen II 124
E.28 Post-Frage: Java-Wissen III 124
E.29 Post-Frage: Java-Wissen IV 125
E.30 Post-Frage: Verständnis Algorithmus 125
E.31 Diagramm: Nationalität 126
E.32 Diagramm: Hochschulabschluss der Eltern 126
E.33 Diagramm: Studiengang 126
E.34 Diagramm: Zufriedenheit mit dem Studium 127
E.36 Diagramm: Lernen im Studium 129
E.37 Diagramm: Umgang mit Problemen 130

E.39 Diagramm: Schulinformatik 132
E.40 Diagramm: Erster Leistungskurs 133
E.41 Diagramm: Zweiter Leistungskurs 133
E.42 Diagramm: Abiturnote 133
E.43 Diagramm: Fachliche Kenntnisse 134
E.44 Diagramm: Workshop-Inhalte 135
E.45 Diagramm: Schwierigkeitsgrad 136
E.46 Diagramm: Studienwahl 136
E.47 Diagramm: Benotung des Workshops 137
E.48 Diagramm: Punkte Teilaufgaben 138
E.49 Diagramm: Punkte Arbeitsaufträge 139

Tabellenverzeichnis

4.1 Unterscheidung typischer Spielsteinkonstellationen 39

4.2 Übersicht über die Bewertung für Spielsteinreihen 41

A.1 Zeitplan für den Workshop 71

1. Einleitung

Die Förderung des Frauenanteils im Informatikstudium und die Reduzierung der Abbrecherquote in der Studieneingangsphase soll an der RWTH Aachen University unter anderem mit Workshops im ersten und zweiten Fachsemester bewerkstelligt werden. Das meist theoretische Wissen aus den Vorlesungen soll in den Workshops praktisch angewendet werden.

In den Workshops können Studenten[1] des Fachbereichs Informatik an einem halben Tag unter Begleitung eine kleine alltagstaugliche App programmieren. (vgl. Holz (2012a)) In diesen Anwendungs-Workshops werden Kenntnisse aus dem regulären Vorlesungsbetrieb aufgegriffen und in den kleinen Programmier-Projekten von den Studenten vertieft. (vgl. Glaap (1993), S. 143)

Als Inhalt der App wurde für diesen Workshop das ursprünglich chinesische Brettspiel Go-Moku ausgewählt. Im Rahmen des Workshops für das zweite Fachsemester werden Teile des Go-Moku Spiels für Android-Geräte programmiert. Der Nutzer der App soll sowohl

[1] Zum Zwecke des besseren Leseflusses wird darauf verzichtet, geschlechtsspezifische Unterscheidungen vorzunehmen. So sind zum Beispiel in der Bezeichnung „Studenten“ alle männlichen und weiblichen Personen inbegriffen. Sollten geschlechtsspezifische Unterscheidungen aus wissenschaftlichen Gründen notwendig sein, werden diese explizit erwähnt.

gegen eine Person, als auch gegen den Computer antreten können.[2] Im zweiten Fachsemester ist durch das Modulhandbuch (vgl. RWTH Aachen University (2010), S. 103) für den fachwissenschaftlichen Bachelor-Studiengang Informatik mit der Vorlesung „Datenstrukturen und Algorithmen“ ein Schwerpunkt auf selbige Themen gelegt. Der Schwerpunkt des Workshops liegt aus diesem Grund auf den Datenstrukturen. Des Weiteren werden grundlegende Programmierkenntnisse aufgegriffen, kleinere Algorithmen implementiert und ein Einblick in die Programmierung einer künstlichen Intelligenz gegeben.

In dieser Ausarbeitung wird im zweiten Kapitel die Motivation für einen Workshop in der Studieneingangsphase aus unterschiedlichen Perspektiven betrachtet. Im dritten Kapitel wird die Ausgangslage analysiert, indem die Kompetenzen der Studenten dargestellt werden. Aus den curricularen Vorgaben der Studienordnung, sowie den Kenntnissen der Studenten aus der schulischen Bildung, ergeben sich Voraussetzungen für die Studieneingangsphase. Im vierten Kapitel wird die Konzeption des Workshops dargestellt, analysiert und die Auswahl begründet. Dabei wird sowohl die Auswahl der Inhalte betrachtet als auch die möglichen Auswirkungen des Workshops auf den Lernerfolg. Im fünften Kapitel dieser Ausarbeitung wird die erste Durchführung analysiert. Mithilfe eines Fragebogens wurden quantitative Erhebungen zu Beginn und am Ende des Workshops getätigt. Des Weiteren wurden kurze informelle Inter-

[2] In der Ausarbeitung wird die Strategie, die von der künstlichen Intelligenz angewandt wird, als Computerstrategie bezeichnet. Der zugehörige Spieler wird Computerspieler genannt.

views und persönliche Eindrücke hinzugefügt. Den Abschluss der Ausarbeitung bildet ein Ausblick mit weiteren Anwendungs- und Optimierungsmöglichkeiten des Workshops und Folgen, die aus einem Workshop in der Studieneingangsphase entstehen. Im Anhang sind die Materialien des Workshops an die Arbeit angefügt worden.

2. Motivation

Das Ziel des Workshops besteht darin, dass die Studenten verschiedene Inhalte aus den Vorlesungen wiederholen und lernen, mit den Konstrukten umzugehen. Das Lernen ist ein vielschichtiger Prozess, der grundsätzlich aus drei psychologischen Prozessen besteht: die Motivation, die Emotion und die Kognition, die im Zusammenspiel einen Lernprozess ermöglichen. (vgl. Niegemann (2004), S. 205)

2.1. Komponenten der Motivationsförderung

Im wissenschaftlichen Kontext wird unter Motivation, die „aktivierende Ausrichtung des momentanen Lebensvollzugs auf einen positiv bewerteten Zielzustand“ (Rheinberg und Salisch (2008), S. 16) verstanden. Die Motivation wird durch die Umgebung und die Bedingungen beeinflusst. Die Motivation kann abnehmen, wenn die aktivierende Ausrichtung abnimmt. Die Inversion bei zunehmender Motivation gilt ebenfalls. Beides kann durch die Lernumgebung gefördert bzw. verhindert werden. Im Workshop wird die Motivation

durch zwei Komponenten grundsätzlich gefördert. Die Implementierung eines Computerspiels und der Einsatz von mobilen Endgeräten sind zwei motivationsfördernde Komponenten.

2.1.1. Computerspiel

Seit der Verbreitung der ersten Computerspiele werden Jugendliche in der Freizeit von diesen begeistert. Seit Anfang der neunziger Jahre erfasst das Statistische Bundesamt im Zehnjahresrhythmus in der Studie zu Freizeitaktivitäten von Jugendlichen eine eigene Kategorie Computerspiele. Bei der letzten Erhebung im Jahr 2001 wurde festgestellt, dass Jungen durchschnittlich 56 Minuten von ihrem täglichen Freizeitbudget von 430 Minuten in das Spielen von Computerspielen investieren. Bei Mädchen sind es lediglich 16 Minuten von 395 Minuten ihres täglichen Freizeitbudgets. (vgl. Cornelißen und Blanke (2004), S. 169) Eine neuere Untersuchung in der JIM-Studie[3] des Jahres 2012 zeigt eine erweiterte Steigerung auf durchschnittlich 62 Minuten Computerspiel Nutzung am Tag. Bei Jungen sind es durchschnittlich 87 Minuten und 35 Minuten bei den Mädchen. (vgl. Behrens und Rathgeb (2012), S. 48) Die Verbreitung von Smartphones und Tablet-Geräten und zugehörigen Spielen wurde von beiden Studien nicht erfasst. Die dafür aufgewendete Zeit ist hinzuzunehmen.

In der Öffentlichkeit werden des Öfteren die negativen Folgen von Computerspielen dargestellt. Unter dem Schlagwort „Serious Games“

[3] Akronym der Studie: Jugend, Information, (Multi-) Media des Medienpädagogischen Forschungsverbund Südwest.

(vgl. Helm (2011), S. 18) sind didaktisch aufbereitete Computerspiele zum Einsatz in der Bildung in den letzten Jahren entstanden. Des Weiteren ist seit Mitte der neunziger Jahre bekannt, dass der Einsatz von Computerspielen zur Motivationsförderung und für die Arbeit am Computer eine geeignete Trainingsmöglichkeit ist. (vgl. Jürgen (1995), S. 13) Die kognitive Leistungsfähigkeit und insbesondere die Problemlösefähigkeit kann durch ein Computerspiel gefördert werden. Wenn durch ein Spiel Lösungsstrategien eingeübt werden, können diese auf andere Situationen übertragen werden. Im Workshop sollen die Problemlösestrategien zur Implementierung und Optimierung des Spiels erarbeitet und angewendet werden. Die Zielperspektive eines selbst programmierten Computerspiels und die Möglichkeit, bei der Implementierung des Spiels zu lernen, ist eine doppelte Möglichkeit der Motivation.

2.1.2. Mobiles Endgerät

Die zweite motivationsfördernde Komponente bildet der Einsatz von mobilen Endgeräten. In der Literatur wird der Einsatz von Neuen Medien des Öfteren als Mittel für die Motivationsförderung beschrieben, unabhängig von den Inhalten bzw. des Einsatzszenarios. Viele Forschungsergebnisse zeigen, dass dieser Effekt nur kurzfristig eintritt, wenn die Lerngruppe die Geräte zum ersten Mal einsetzt. (vgl. Aschersleben (1977), S. 133; vgl. Kammler (2007), S. 121; vgl. Ludwig (2013), S. 25 und vgl. Schulmeister (2009), S. 3) Im Workshop kann dieser Neuigkeitseffekt ausgenutzt werden, weil die

Teilnehmer zuvor nicht mit dem Einsatz von mobilen Endgeräten rechnen. Die Inhalte und das Einsatzszenario der Tablet-Geräte sollten aufeinander abgestimmt werden.

2.2. Kompetenzvermittlung in der Studieneingangsphase

Jedes Bachelor-Studium bietet den Studenten nach erfolgreichem Abschluss die Möglichkeit, ein vertiefendes Masterstudium anzuschließen, oder eine berufliche Tätigkeit im passenden Segment aufzunehmen. Die Kultusministerkonferenz hat im Jahr 2003 einen Schwerpunkt auf die berufliche Qualifizierung der Studenten nach einem erfolgreichen Abschluss gelegt, indem jedem Bachelor-Studium ein „eigenständiges berufsqualifizierendes Profil“ (Kultusministerkonferenz (2003), S. 2) als Vorgabe für die Akkreditierung vorgegeben wurde. Die beiden Zielvorgaben stellen besondere Herausforderungen an die Inhalte und Methoden, die von den Hochschuldozenten angewendet werden. Zum Teil stehen die Vorgaben zueinander in einem Konflikt, weil praktisches Wissen für eine berufliche Laufbahn nach dem Bachelor-Studium und theoretisches Wissen für eine weitere akademische Laufbahn benötigt wird.
Zur genaueren Analyse der benötigten Kenntnisse für den erfolgreichen Abschluss eines Bachelor-Studiengangs im Bereich der Informatik sind das Berufsfeld und das Wissenschaftsfeld der Informatik separat zu betrachten.

An der RWTH Aachen University wurde in der Studienordnung für den grundständigen Bachelor-Studiengang Informatik das Ziel für die Vorbereitung auf eine „berufliche Tätigkeit bzw. eine Vertiefung in einem Master-Studiengang“ (RWTH Aachen University (2010), S. 3) festgelegt.

Im Bereich der Wissenschaft ist bei der Informatik zwischen der Kerninformatik und der angewandten Informatik zu unterscheiden. (vgl. Schubert und Schwill (2011), S. 2). Eine zentrale Komponente der Kerninformatik ist die praktische Informatik, deren Aufgabe in der Lösung von beliebigen Problemen besteht. Eine benötigte Kernkompetenz besteht in der Programmierung und dem zur Programmierung benötigten Wissen. Bei der Implementierung werden Kernkompetenzen unabhängig von der spezifischen Anwendung vermittelt. Der Vorteil der längeren Halbwertszeit des Wissens ist ein Hinderungsgrund für den praktischen Einsatz des Wissens. Ein Student sieht in den ersten Semestern die Anwendungsmöglichkeiten des Wissens zum Teil nicht und wird dadurch demotiviert.

Die Bachelor-Studienordnung der RWTH Aachen University sieht in der aktuellen Fassung eine „breit angelegte Ausbildung in den Grundlagen der Informatik“ (RWTH Aachen University (2010), S. 3) vor. Ein Nachteil einer breit angelegten Ausbildung kann in der inhaltlichen Überfrachtung und der Entstehung von Inselwissen bestehen. Der Bezug zwischen den unterschiedlichen Theorien ist nicht automatisch bei jedem Studenten zu Beginn des Studiums gegeben. Wenn ein Student das Wissen nicht miteinander verknüpfen kann, nimmt die Motivation bei dem Studenten schnell ab.

Im Bereich der beruflichen Tätigkeit eines Informatikers ist eine projektorientierte Arbeitsweise eine der führenden Arbeitsstile. Diese Arbeitsform kann durch anwendungsbezogene Lernumgebungen simuliert bzw. in diesen Lernumgebungen erlernt werden.
Die beschriebenen Ziele der Prüfungsordnung, die aus dem Entwurf und der Implementierung von Programmen bestehen, werden zum Großteil in den Basisvorlesungen „Einführung in die Programmierung“ und „Datenstrukturen und Algorithmen“ der ersten beiden Semester vermittelt und sind entscheidend unter anderem für die Motivation für das weitere Studium und die Qualifikation für den Arbeitsmarkt.

2.3. Wissensrepräsentation

In der Studieneingangsphase eines fachwissenschaftlichen Bachelor-Studiengangs im Fach Informatik werden in den unterschiedlichen Vorlesungen verschiedenste Theorien aus den unterschiedlichen Teildisziplinen der Informatik dargestellt. Die Studenten können wenig Bezüge zwischen den Inhalten aus den verschiedenen Vorlesungen herstellen und es entsteht Inselwissen bei den Studenten. (vgl. Edelmann (2000), S. 171) Das erworbene Wissen kann mit einem Kontextbezug längerfristig nutzbar gemacht werden und für weitere Tätigkeiten während des Studiums und des Berufslebens aktiv genutzt werden. Im Gegensatz dazu kann Wissen ohne ausreichende kognitive Verknüpfungen nur träge abgerufen werden und wird somit in kurzer Zeit zu trägem Wissen. (vgl. Leopold (2009),

S. 144-145) Des Weiteren ist das Lernen mit mobilen Endgeräten, in diesem Workshop mit Android-Tablet-Computern, eine zusätzliche Motivation für die Studenten. Die Studenten erhalten eine direkte Rückmeldung nach dem Abschluss eines Programmstücks, weil dieser Teil des Programms auf dem mobilen Endgerät ausgeführt werden kann. Die direkte Rückmeldung wirkt positiv auf die Motivation. (vgl. Holz u. a. (2011), S. 93)

Aus den beschriebenen Gründen sind im Studium und auch in der Studieneingangsphase verschiedene anwendungsbezogene Lernumgebungen von großer Relevanz für das Erreichen der Ziele.

3. Theoretischer Hintergrund

3.1. Ausgangslage und Ziele

Bei den Studenten in den ersten Semestern eines Informatikstudiums wurde von der Fachgruppe Informatik der RWTH Aachen University zwei zentrale Probleme festgestellt. Zum einen ist der Frauenanteil unter den Studierenden mit 14 Prozent im Jahr 2012 deutlich unter den gewünschten Zielen. (vgl. Hötte u. a. (2013), S. 47) Zum anderen ist die Studienabbruchquote in den MINT-Fächern[4] sehr hoch. An der RWTH Aachen University liegt im Bachelor Studiengang Informatik in den letzten Jahren eine Schwundquote[5] zwischen 40 Prozent und 44 Prozent vor. (vgl. Abbildung 3.1 auf Seite 12) Deutschlandweit liegt die Studienabbruchquote[6] im Studiengang Informatik bei 47 Prozent (vgl. Heublein u. a. (2012), S. 18) für den Absolventenjahrgang 2010. Diese Werte sind im Vergleich zu den Zielen des

[4] Zusammenfassende Bezeichnung für die Schulfächer: Mathematik, Informatik, Naturwissenschaften und Technik.

[5] Die Schwundquote ist eine Zusammenfassung der Abbrecher- und Wechslerquote. Die Quote bezieht sich auf die Studienanfänger eines Jahrgangs und betrachtet einen Zeitraum von 6 Semestern.

[6] Die Studienabbruchquote betrachtet die Studenten, die das Studium abbrechen und kein neues Hochschulstudium aufnehmen.

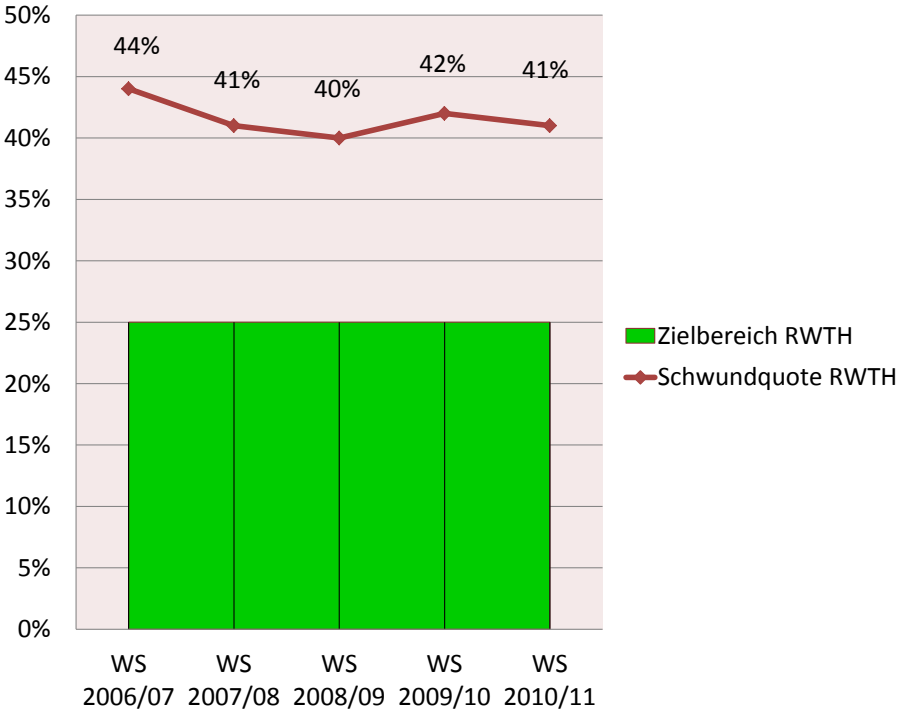


Abbildung 3.1.: Schwundquote der RWTH Aachen University. Die Daten der zentralen Hochschulverwaltung der RWTH Aachen University basieren auf der jeweiligen Anfängerkohorte bei einem Betrachtungszeitraum von jeweils sechs Semestern.

Rektorats der RWTH Aachen University zu hoch. Das Rektorat wünscht sich eine hochschulweite Bestehensquote[7] von 75 Prozent der Studenten, die ein Studium an der RWTH Aachen University aufnehmen. (vgl. Hartmann (2013) und vgl. Krieg (2013)) Die beiden Ziele werden für den fachbezogenen Bachelor-Informatikstudiengang angestrebt.

Diese globalen Ziele sollen in der Informatik im Rahmen des Forschungsprojekts IGaDtools4MINT[8] (vgl. Holz (2012b)), untersucht

[7] Die Bestehensquote ist der Gegensatz zur Schwundquote.

[8] Akronym des Projekts: Integration von Gender and Diversity in MINT-Studiengänge an Hochschulen.

werden und eine erste Verbesserung mit Workshops in der Studieneingangsphase erzielt werden.
In diesem Workshop soll vor allem die Motivation für das Studienfach Informatik durch den Einsatz von mobilen Endgeräten gefördert werden. Des Weiteren steht die Aufarbeitung von inhaltlichen Verständnisproblemen aus dem ersten Studienjahr im Mittelpunkt des Workshops. Der Fokus liegt hierbei auf dem Thema der Datenstrukturen. Im Rahmen des Workshops können ausgewählte Inhalte aus den Vorlesungen aufgegriffen werden und im Rahmen eines kleinen Projektes von den Studenten selbständig zur Lösung eines Problems eingesetzt werden.

3.2. Vorwissen

Das Vorwissen der Studenten ist differenziert, nach den formalen Kenntnissen und den zusätzlichen praktischen Fähigkeiten, zu betrachten. Bei den formalen Fähigkeiten sind die Vorgaben aus der Prüfungsordnung zu betrachten und bei den praktischen Fähigkeiten zum Beispiel Softskills hervorzuheben.

3.2.1. Formale Kenntnisse

Vorlesung: Einführung in die Programmierung

Anhand des standardisierten Inhalts der Vorlesung „Einführung in die Programmierung“ ist festzuhalten, dass bereits verschiedene

imperative und objektorientierte Programmierkonzepte vermittelt wurden. Als erste und einfachste Datenstruktur wurden Arrays in verschiedensten Formen in der Vorlesung behandelt. Die Datenstrukturen Liste und Bäume werden nur indirekt im Modulhandbuch genannt, indem einfache Algorithmen auf diesen Strukturen erlernt werden. Dazu zählt unter anderem auch die Rekursion, die beim Aufbau eines Suchbaumes, ein weitverbreitetes Konzept darstellt. Ein Suchbaum beinhaltet zwangsläufig eine Baum-Struktur.
Bei der Auswertung der Ergebnisse der Klausur vom ersten Prüfungstermin ist festzustellen, dass die vierte Klausuraufgabe, in der die Programmierung in Java überprüft und dabei auf die Datenstruktur des Baumes von den Studenten angewendet wurde, die Aufgabe ist, die insgesamt die schlechtesten Ergebnisse im Java-Teil darstellt. (vgl. Otto (2013)) Im Durchschnitt wurden nur 63 Prozent der Punkte von den Klausurteilnehmern erzielt. Des Weiteren haben Studenten Probleme die Klausur zu bestehen, wenn in der vierten Aufgabe, die den Umgang mit Datenstruktur überprüft, nicht ausreichend Punkte erzielt wurden. Von 35 Studenten, die die Klausur trotz Erscheinens nicht bestanden haben, hatten 28 Studenten, dies entspricht 80 Prozent, weniger als die Hälfte der Punkte in dieser Klausuraufgabe. Die Standardoperationen Einfügen und Entfernen von Knoten in einem Baum sind für das Verständnis von sehr vielen Informatiksystemen von zentraler Bedeutung. (vgl. Otto (2013))

Vorlesung: Datenstrukturen und Algorithmen

Die Teilnehmer des Workshops besuchen die Vorlesung „Datenstrukturen und Algorithmen“ und haben zu diesem Zeitpunkt circa die Hälfte der Vorlesungstermine bereits absolviert. Auf die Kenntnisse aus dem ersten Teil der Vorlesung und auf die Voraussetzung für die Vorlesung kann zurückgegriffen werden. Im Wesentlichen ist dies die Vertiefung der Kenntnisse aus der Vorlesung „Einführung in die Programmierung“ aus dem vorherigen Studiensemester. Schwerpunktmäßig werden die Kenntnisse in der Programmiersprache Java für den Workshop benötigt und vertieft. Es ist zu beachten, dass nicht alle Studenten zuvor die Vorlesung „Einführung in die Programmierung“ absolviert haben müssen bzw. die Klausur erfolgreich bestanden haben. Vor allem Studienanfänger im Sommersemester können die Vorlesung „Datenstrukturen und Algorithmen“ ohne eine Grundlagenvorlesung in der Programmierung besuchen. Einige dieser Studenten werden zur Zielgruppe gehören, weil ohne grundständig erworbene Programmierkenntnisse des Öfteren Probleme im zweiten Semester auftreten. Der größere Teil der Studenten wird den regulären Studienablauf durchlaufen und die Kenntnisse aus der Programmierungsvorlesung erworben haben. (vgl. RWTH Aachen University (2010), S. 103)

In der Vorlesung „Datenstrukturen und Algorithmen“ werden verschiedene Datenstrukturen vermittelt. Es werden aufbauend auf den Kenntnissen aus dem ersten Studiensemester weitere Strukturen und Lösungsstrategien vermittelt. Dabei sind Listen, Bäume und

Graphen im Besonderen hervorzuheben. Die unterschiedlichen Professoren, die diese Vorlesung abwechselnd halten, setzen unabhängig vom Modulhandbuch verschiedene Schwerpunkte.

Die Voraussetzungen zur Vorlesung „Datenstrukturen und Algorithmen“ im Modulhandbuch unterteilen sich in mehrere Bereiche. Besonders hervorzuheben, wegen der Relevanz für den Workshop, sind die „Beherrschung der wesentlichen Konzepte [...] sowie elementarer Programmiertechniken“ (vgl. RWTH Aachen University (2010), S. 24), „Kenntnisse linearer Datenstrukturen wie Arrays, Listen, Stacks, Queues“ (Ebenda, S. 24) und die „Fähigkeiten zur selbständigen Entwicklung kleinerer Programme“ (Ebenda, S. 24). Die beschriebenen Voraussetzungen werden im Workshop aufgegriffen und vertieft. Des Weiteren sind diese Voraussetzungen laut Modulhandbuch auch im Rahmen der Lernziele der Vorlesung „Datenstrukturen und Algorithmen“ zu vertiefen, um zum Beispiel mit dem Erwerb von „grundlegende[n] Entwurfsmethoden für Algorithmen“ (Ebenda, S. 23) die Ziele der Vorlesung zu erfüllen. Der Schwerpunkt des Workshops liegt deswegen auf den Datenstrukturen und Algorithmen, die diese Datenstrukturen nutzen. Zugehörige Lernziele für die Vorlesung sind im Modulhandbuch unter anderem: „Kenntnis effizienter Algorithmen und Datenstrukturen für Standardprobleme“ (Ebenda, S. 23), die „[...]Anpassung von vorhandenen Algorithmen und Datenstrukturen an die gegebene Problemstellung“ (Ebenda, S. 23) und die „Fähigkeit zur Implementierung der erlernten algorithmischen Methoden unter Berücksichtigung programmiertechnischer Konzepte“ (Ebenda, S. 23). Diese Lernziele sind in der abschließen-

den Klausur bei der Vorlesung „Datenstrukturen und Algorithmen“ durch die Studenten nachzuweisen.

Kenntnisse in der Spieltheorie

Die Spieltheorie ist für die Studenten ein neues Thema. Erst im folgenden Semester wird in der Vorlesung „Mathematische Logik“ ein wenig Spieltheorie mit den Themen „Auswertungsspiele“ und „Modellvergleichsspiele“ behandelt. Ein erster praktischer Einblick in diese Themenfelder kann den Studenten an dieser Stelle zugutekommen, weil in der Vorlesung „Mathematische Logik“ die Spieltheorie eher theorielastig und stark formalisiert dargestellt wird. Die Verknüpfung der Praxis und der Theorie kann zu einem erweiterten Verständnis bei den Studenten führen.
Der Teil des Workshops zur Spieltheorie ist auf die Grundlagen zu reduzieren, weil diese Inhalte im Studium noch nicht vorgekommen sind und in der jetzigen Situation für die Zielgruppe wenig Relevanz für den Studienfortschritt haben.

Erworbene formale Kenntnisse

Insgesamt ist festzustellen, dass bei vielen Studenten in der Studieneingangsphase ein Problem bei dem Umgang mit Datenstrukturen und einfachen Operationen auf diesen Strukturen besteht. Besondere Aufmerksamkeit ist aus diesem Grund den mehrdimensionalen Arrays, Listen und Bäumen als Spezialform von Graphen, sowie den Operationen zur Bearbeitung der Strukturen zu schenken.

Diese drei Grundstrukturen sollten unabhängig von der Programmiersprache von jedem Informatiker beherrscht werden, um effiziente und effektive Programme zu gestalten.

3.2.2. Praktische Fähigkeiten

In den ersten beiden Studiensemestern sind viele Vorlesungen und keine Seminare und Praktika auf dem Studienverlaufsplan für die Studenten vorgesehen. Im dritten Semester stehen die „Einführung in das wissenschaftliche Arbeiten“ und das „Praktikum E-Technische Grundlagen und Hardwarenahe Programmierung“ an. (vgl. RWTH Aachen University (2010), S. 103) In Proseminar und Praktikum werden entsprechende Softskills wie Teamfähigkeit, Präsentationstechniken und die gemeinsame Planung eines Projektes von den Studenten gefordert.
Im Workshop arbeiten die Studenten in Partnerarbeit zusammen und müssen gemeinsam zu einer Lösung kommen. Des Weiteren wird während des Workshops ein kleines Projekt bearbeitet und die Studenten müssen zum Beispiel den Ablauf der Programmierung einer Spielstrategie selbständig organisieren.

3.3. Thesen

Aus der beschriebenen Ausgangslage und den Kompetenzen der Studenten sind die folgenden vier Thesen abzuleiten:

1. Der Einsatz von mobilen Endgeräten und Computern zur Implementierung einer Android-App verstärkt die Motivation, das Projekt zu bearbeiten.
2. Die Studenten erkennen während des Workshops die praktische Arbeit eines Informatikers an einem Projekt.
3. Die Grundlagen in der Programmierung und in den Datenstrukturen werden in einem Workshop vertieft.
4. Ein Workshop, ohne Leistungsdruck, kann einen Beitrag zur Motivierung für das weitere Informatikstudium leisten.

Zur Überprüfung der Thesen wird zum einen eine quantitative Erhebung mittels eines Fragebogens durchgeführt. Vor und nach dem Workshop wird jeweils ein Fragebogen durch die Studenten ausgefüllt. Zum anderen werden die Studenten informell befragt und die Eindrücke während und zum Abschluss des Workshops gesammelt. Im fünften Kapitel dieser Arbeit ist die Evaluierung detailliert beschrieben und die Auswertung dargestellt. Zunächst wird im vierten Kapitel die Konzeption des Workshops näher dargestellt.

4. Konzeption und Analyse

4.1. Voraussetzungen

Als Voraussetzung für den Workshop werden Grundlagen in der Programmierung aus der Vorlesung „Einführung in die Programmierung“ von den Studenten verlangt. Die Kenntnisse aus der Vorlesung „Datenstrukturen und Algorithmen“ sind für die Bearbeitung hilfreich, werden allerdings nicht vorausgesetzt. Diese Inhalte werden im Lernmaterial zum Workshop aufgegriffen, erklärt und vertieft.
Zu anderen Themen der Studieneingangsphase werden ebenfalls Workshops angeboten. Im letzten Semester sind zum Beispiel die Workshops „Quiz App“ und „Rekursive Bilder“ angeboten wurden. In diesen Workshops wurden Themen aus der Vorlesung „Einführung in die Programmierung“ aufgegriffen. In diesem Semester wurde der Workshop „2D-Transformation von Grafiken“ durchgeführt. In dem Workshop wurden Konzepte aus der Linearen Algebra und Datenstrukturen miteinander verknüpft, um Grafiken zu transformieren. (vgl. Holz (2012a))

4.2. Hintergrund zur Auswahl des Inhalts der App

Die Auswahl eines geeigneten Inhalts für die Android-App ist für das Gelingen des Workshops von großer Bedeutung. Wenn die Studenten während der beschränkten Zeit im Workshop ein Produkt erzeugen, dass eine fertige und nutzbare App darstellt, ist dieses für die Motivation förderlich.

Eine abgeschlossene App, deren Inhalt zum Großteil in vier Stunden von Studenten programmiert werden kann, ist zum Beispiel ein Spiel. Die meisten Spiele bieten den Vorteil, dass keine Kommunikation über das Internet mit Datenbanken und Ähnlichem notwendig ist. Die Programmierkonzepte oder die Datenstrukturen anhand eines Spiels kennenzulernen, wird in vielen Programmierkursen genutzt. (vgl. Post (2012), S. 15-39) Dabei sollte stets die Devise gelten: „Das Programmieren des Computers ist das Spiel; so lernt der Leser spielend das Programmieren.“ (Baumann (1983), S. XII). Im Workshop entsteht ein doppelter Effekt, weil zum einen das Programmieren der App Spaß macht und zum anderen das Ergebnis genutzt werden kann und ebenfalls Spaß bereitet.

Die Auswahl eines geeigneten Spiels ist eine komplexere Frage. Zur begründeten Auswahl des Spiels sind zunächst geeignete Kriterien notwendig. Die Grundlagen des Spiels sollten geeignete Datenstrukturen, passend zum Schwerpunkt des Workshops, sein. Grundlagen für die Datenstrukturen sollen ein Array und eine Liste sein. Als algorithmische Grundlage bietet sich eine Baumsuche an. Meistens

ist die praktische und korrekte Anwendung der Konstrukte eine Hürde für die Studenten.

Die Studenten haben im zweiten Semester noch keine Kenntnisse in der mathematischen Logik erworben. Die Kenntnisse sind für die Erschaffung einer schwachen künstlichen Intelligenz bei den meisten Spielen notwendig. (vgl. Breuer (2012), S. 5) Bei der Auswahl der Spiele ist auf eine einfache und verständliche Beschreibung der künstlichen Intelligenz zu achten. Beispielsweise sind klassische Logikspiele wie Schach (vgl. Levy (1988), S. 116-153) und Backgammon (vgl. Berliner (1988), S. 3-28) zu komplex für einen Workshop mit den zeitlichen Rahmenbedingungen und der angestrebten Zielgruppe. Auf der anderen Seite bewegt ein Spiel wie Tic-Tac-Toe (vgl. Luger (2002), S. 148) keinen Studenten dazu, an einem Workshop teilzunehmen.

Zwei weitere Kriterien sind die Möglichkeit zur Steigerung der Komplexität und die Neuartigkeit der App, weil im Android-App-Store keine oder wenige Apps zu dieser Spielidee existieren. Die Komplexität sollte während des Workshops erhöht werden können, damit im Workshop das Niveau gesteigert werden kann. Allerdings sollen schon zu einem frühen Zeitpunkt die ersten Arbeitsergebnisse ausprobiert werden können. Wenn eine App mit ähnlichem Inhalt im App Store existiert und die Qualität der eigenen App deutlich schlechter ist, besteht bei der Entwicklung wenig Motivation bei den Studenten.

Insgesamt wurden im Rahmen der Staatsexamensarbeit 18 mögliche Spiele detaillierter betrachtet. Im Anhang dieser Arbeit sind im

Kapitel D des Anhangs (ab Seite 94) Beschreibungen der Spielideen für die verschiedenen Spiele aufgeführt. Des Weiteren wurde die Erfüllung der Kriterien

- Existenz einer Android-App,
- Motivation für die Studenten,
- Programmierbarkeit der künstlichen Intelligenz,
- enthaltene Datenstrukturen,
- Erweiterbarkeit bezüglich der Größe des Spielfelds und
- der Wahl des Schwierigkeitsniveaus

bei den einzelnen Spielen überprüft. Einige Spiele fielen bei der Überprüfung weg, weil ein Kriterium überhaupt nicht erfüllt wurde. Zum Beispiel sind beim Mühle-Spiel keine geeigneten Datenstrukturen vorhanden. (vgl. Schürmann u. a. (1980), S. 8) Das Spielbrett kann nur mit größerem Aufwand im Computer repräsentiert werden. Einen Gegensatz dazu bietet das Sudoku Spiel. Bei diesem Spiel ist die Programmierung kein Problem (vgl. Wolmeringer (2008), S. 185-190), aber eine vernünftige Eingabe der Werte auf einem Tablet-Gerät schwierig.

In die engere Auswahl wurden die Spiele Go-Moku[9] und Reversi genommen, weil diese Spiele die verschiedenen Kriterien am besten erfüllten. Die letztendliche Wahl ist auf das Go-Moku Spiel gefallen, weil dieses Spiel unbekannter ist und dadurch etwas Neues für die

[9] Original japanische Schreibweise für das Spiel ist „Gomoku“. Für den Workshop und die App wurde die in Deutschland nicht geschützte Schreibweise „Go-Moku“ verwendet.

Studenten darstellt. Dadurch ist die Gefahr, dass ein Student das Spiel schon zu einem früheren Zeitpunkt programmiert hat, etwas kleiner.

4.3. Aufbau des Workshops

Das Ziel des Workshops besteht in der Vermittlung der Grundlagen der Datenstrukturen, ergänzt um die Vertiefung der Programmierkenntnisse und ersten Erfahrungen mit einer künstlichen Intelligenz. Es wurde bereits beschrieben, dass dazu Tablet-Geräte mit einem Android-Betriebssystem eingesetzt werden. Der zeitliche Umfang des Workshops ist auf circa vier Stunden beschränkt, weil mehr Zeit bei den Bachelor-Studenten während der Vorlesungszeit nicht zur Verfügung steht. Für eine ausreichende Anzahl an Computer sind einer oder gegebenenfalls mehrere Computerräume vom Rechnerbetrieb-Informatik zu nutzen. In den Computerräumen wurden die Tische verankert in Reihen aufgebaut und handelsübliche Computer aufgestellt. Die Anordnung kann für den Workshop nicht verändert werden. Aus diesen Vorgaben ergeben sich für den Aufbau des Workshops einige didaktische Vorgaben.

Die Lerngruppe wird sich aus Studenten mit unterschiedlichsten Kenntnisständen zusammensetzen. Zum einen werden Studenten am Workshop teilnehmen, die mit der eigenständigen Programmierung von kleinen Programmen Probleme haben und zum anderen werden Studenten am Workshop teilnehmen, die Interesse an der Android-Programmierung und wenig bis gar keine Probleme mit der

Implementierung haben. Die Studenten, die im vorherigen Semester an einem Workshop teilgenommen haben, werden auch am Workshop teilnehmen wollen. Trotz dieser Heterogenität in der Lerngruppe muss die Förderung der schwächeren Studenten im Fokus bei der Konzeption des Workshops stehen. Eine gute Differenzierung und die angemessene Förderung eines jeden Workshop Teilnehmers ist zu gewährleisten. (vgl. Fischer (1973), S. 9) Die Studenten mit sehr guten Vorkenntnissen können zum Beispiel mit Optimierungsaufgaben weitere und komplexere Herausforderungen erhalten. Wegen der Heterogenität sind regelmäßige Plenumsdiskussionen zu vermeiden, weil die schnellen Studenten warten müssten und die langsamen Studenten gedrängt würden. Die Lerneinheit kann im ersten Teil gemeinsame Anteile besitzen, diese sind im weiteren Verlauf des Workshops zu vermeiden. Aus diesem Grund wird es beim Workshop eine gemeinsame Einführung und Abschlussphase geben. Dazwischen können die Studenten in einem angemessenen und selbstgewählten Tempo die unterschiedlichen Phasen durchlaufen, um im eigenen Lerntempo die Inhalte des Workshops zu erarbeiten.

Damit die Studenten während des Workshops über die Inhalte diskutieren können, ist für den gesamten Workshop Partnerarbeit vorgesehen. Diese Sozialform bietet den Vorteil, dass die Studenten sich ergänzen können und gemeinsam über die Lösungen diskutieren können. Bei der Softwareentwicklung treten des Öfteren kleinere Fehler, meist durch Tippfehler, auf. Die Korrektur kann viel Zeit in Anspruch nehmen. Wenn die Studenten das Programm gemeinsam entwickeln, werden diese Fehler zügig gefunden und können beseitigt

werden. Des Weiteren werden auch Verständnisprobleme zügig bemerkt und entweder mit dem Partner oder mit einem Tutor geklärt. Für die Strukturierung des Workshops wurde auf das Vier-Phasen-Modell von Fässler zurückgegriffen. Die Phasen werden mit „SEE“, „TRY“, „DO“ und „EXPLAIN“ bezeichnet. In jeder Phase wird das Wissensnetz zum Thema in unterschiedlichen kognitiven Dimensionen angesprochen. (vgl. Fässler (2007), S. 121) Ein detaillierter Zeitplan kann der Tabelle A.1 auf Seite 71 entnommen werden. (vgl. Abbildung 4.1 auf Seite 27)

Für den Workshop wurde eine Arbeitsmappe mit Arbeitsblättern, das Go-Moku Spiel in einer Brettspielvariante und eine Dokumentation des Grundgerüstes des Programmcodes, den Studenten zur Verfügung gestellt.

4.3.1. SEE-Phase

Nach einem Überblick über den Aufbau des Workshops steht in der SEE-Phase das Entdecken der Konzepte und Probleme im Vordergrund. Auf der kognitiven Dimension bedeutet dies die Betrachtung der Inhalte auf der deklarativen Ebene. Für die Problematik wird ein entsprechendes Wissensnetz im Gehirn erzeugt, um im weiteren Verlauf auf dieses Wissensnetz zurückzugreifen. (vgl. Fässler (2007), S. 121)

Für den Workshop bedeutet dies, dass im ersten Abschnitt das Erlernen der Spielregeln des Go-Moku Spiels im Vordergrund steht. Für die weiteren Abschnitte des Workshops sind Kenntnisse der

Abbildung 4.1.: Ablaufplan des Workshops

Spielregeln notwendig. Die Studenten erhalten dazu die Spielregeln und eine nicht elektronische und fassbare Variante des Spielbretts mit Spielsteinen. In den jeweiligen Zweiergruppen können die Studenten einige Male das Spiel durchführen. Währenddessen werden drei Leitfragen vom ersten Arbeitsblatt (vgl. Abbildung B.2 auf Seite 74) beantwortet, damit eine Ergebnissicherung stattfindet und für die Entwicklung einer eigenen Strategie erste Ideen gesammelt werden.

4.3.2. TRY-Phase

In der TRY-Phase, die einen Umfang von circa einer Stunde haben soll, erhalten die Studenten die Möglichkeit, die ersten eigenen Erfahrungen mit der Programmierumgebung und den Aufgaben zu sammeln. Dazu sind stark angeleitete Aufgaben erstellt worden, damit die Studenten zunächst in kleinen Schritten an die Programmierumgebung herangeführt werden. Wenn die Studenten zu Beginn eine gewisse Grundsicherheit erhalten, sind komplexere Aufgaben im weiteren Verlauf möglich. In dieser Phase soll das deklarative Wissen aus der SEE-Phase mit prozeduralem Wissen verbunden werden. (vgl. Fässler (2007), S. 121)
Konkret lernen die Studenten die Arbeitsumgebung für den Workshop kennen. Die Entwicklungsumgebung Eclipse wird in den ersten Semestern von den Studenten wenig genutzt. Zur Android-Entwicklung sind diese Entwicklungsumgebung eine Standardanwendung und der kompetente Umgang mit den Grundfunktionen

unerlässlich. (vgl. Abbildung B.3 auf Seite 75)

In das vorgegebene Programmgerüst müssen sich die Studenten erst einarbeiten, weil das gesamte Projekt umfangreicher ist, als ein durchschnittliches Projekt aus den ersten Studiensemestern. Das Programmgerüst besteht aus drei Quellcodepaketen. Die Aufgaben der Pakete teilen sich auf in die Strukturierung der graphischen Oberfläche, die Reaktionen der künstlichen Intelligenz und die Gestaltung des Spielablaufs inklusive der Gewinnerkennung.

Die Programmieraufgaben wurden für die Studenten in kleinere Teilaufgaben gegliedert und im Sinne der bloomschen Taxonomie angeordnet. (vgl. Bloom und Engelhart (1972), S. 31) Bei der ersten Programmieraufgabe (vgl. Aufgabe 3 der Abbildung B.6 auf Seite 78) wird die Struktur des Quellcodes im Aufgabentext klar vorgegeben und lediglich das Wissen um Standardkonstrukte wie Initialisierung eines Arrays wird benötigt.

In diesen ersten Schritten bei der Programmierung wird die Verwaltung für das Spielbrett erstellt. In einem zweidimensionalen Array müssen die Zustände der Spielbrettzellen gespeichert werden. Des Weiteren sind verschiedene Methoden für den Zugriff auf die Datenstruktur notwendig, damit eine sinnvolle Datenkapselung des Spielbretts gewährleistet werden kann. Das Verstehen, im Sinne der zweiten Klasse der boomschen Taxonomie, einer sinnvollen Struktur für die Datenkapselung, muss von den Studenten angewendet werden. Zum Beispiel die Entscheidung über den richtigen Sichtbarkeitsmodifikator (public, protected und private) obliegt den Studenten und wird nicht vorgegeben. (vgl. Ullenboom (2012), S. 487)

Für die Implementierung der verschiedenen Strategien in der nächsten Phase wird eine Datenstruktur zur Speicherung der möglichen Positionen für einen Stein benötigt. Zum einen ist ein normales Array für diese Aufgabe nicht ausreichend, weil das Einfügen eines Elementes am Ende und das Entfernen eines Elementes am Anfang des Arrays nicht problemlos möglich sind. Zum anderen ist eine komplette Listenstruktur mit entsprechenden Zeigern für die Aufgabe überdimensioniert und in der beschränkten Zeit im Workshop nur mit großer Mühe realisierbar. Der Kompromiss in dieser Situation ist eine erweiterte Arraystruktur mit Lese- und Schreibkopf zur Verwaltung der Positionen.[10] Die einzelnen Elemente des Arrays werden in einer Klasse BoardElement implementiert und mit Getter- und Setter-Methoden kann der Zeilen- und Spaltenindex ausgewertet werden. (vgl. Abbildung B.8 auf Seite 80)
Das Ziel der TRY-Phase besteht darin, dass zum einen zwei Menschen auf dem Tablet-Gerät gegeneinander spielen können und die App einen möglichen Sieg erkennt. Zum anderen werden wichtige Grundlagen für die künstliche Intelligenz gelegt.

4.3.3. DO-Phase

Die DO-Phase soll im Sinne des Konzepts von Fässler die längste Arbeitsphase in der Lerneinheit mit circa zwei Stunden bilden. In der DO-Phase sollen die Studenten die Konzepte selbständig anwenden. Dazu sind Aufgaben mit wenig konkreten Vorgaben notwendig. In

[10] Die Datenstruktur ist angelehnt an die Container-Klasse vector in der Programmiersprache C++. (vgl. Wolf (2010), S. 533)

dieser Phase des Workshops eignen sich die Studenten vor allem konditionales Wissen an. Konditionales Wissen ist die Ergänzung des deklarativen und des prozeduralen Wissens mit der Dimension zur richtigen Anwendung des Wissens. (vgl. Fässler (2007), S. 121) Die Studenten können selbständig den Einsatz der Methoden und Konzepte einschätzen. (vgl. Hoy und Schönpflug (2008), S. 344)
Aus diesen Gründen sind die Aufgaben in der DO-Phase deutlich freier formuliert als in den beiden vorherigen Phasen. Die Entwicklung der Spielstrategie erfolgt in drei Schritten und anschließend können weitere Optimierungen angebracht werden. Im ersten Schritt soll per Zufall ein Stein auf eine freie Spielfeldzelle gesetzt werden.[11] (vgl. Abbildung B.10 auf Seite 82) Die **erste Strategie** dient zum einen zur Überprüfung des bisherigen Quellcodes und zum anderen erlangen die Studenten ein Gefühl für das Vorgehen bei der Entwicklung einer Strategie. Die **zweite Strategie** geht auf die aktuelle Spielsituation ein, indem längere Spielsteinketten des Gegners verhindert werden und wenn nötig Steine zur Blockade gelegt werden. Wenn keine Gefahr besteht, wird weiterhin eine Spielsteinzelle per Zufall ausgewählt und der Stein dorthin gelegt. (vgl. Abbildung B.11

[11] **Spielregeln für ein Spielbrett mit acht mal acht Spielfeldzellen**
Die Spieler haben das Ziel als erster Spieler eine Reihe mit fünf Steinen der eigenen Farbe zu bilden. Die Steine müssen in einer Linie liegen, die sich horizontal, vertikal oder diagonal auf dem Spielbrett befindet.
Zu Beginn des Spiels erhält jeder Spieler 32 Spielsteine in einer der beiden verfügbaren Farben. Ein Spieler setzt am Anfang einen Spielstein und anschließend wird abwechselnd jeweils ein Spielstein in der gewählten Spielsteinfarbe auf ein freies Feld des Spielfelds gelegt.
Jeder Spieler versucht fünf Steine nebeneinanderzulegen und den Gegner am Erreichen des Ziels zu hindern, indem der Bau einer Reihe durch blockierende Spielsteine verhindert wird. Längere Spielsteinreihen des Gegners stellen eine Gefahr dar, weil das Risiko auf eine fertige Reihe steigt und die Gefahr einer Niederlage wächst.

auf Seite 83) Die **dritte Strategie** basiert auf einem vereinfachten Min-Max-Algorithmus. Der Algorithmus ermittelt eine optimale Spielstrategie für ein Nullsummenspiel mit perfekter Information. Für die nächsten Spielschritte wird ein Suchbaum mit den möglichen Spielsteinkonstellationen aufgebaut. Die eigenen Züge werden positiv bewertet und die Züge des Gegners werden negativ bewertet. Die Werte der eigenen Züge werden maximiert und die Züge des Gegners minimiert. Die Bewertung der Spielsituation erfolgt mit einer spielspezifischen Bewertungsfunktion. Wenn keine Gefahr droht, wählt der Algorithmus eine Position auf dem Spielbrett aus und legt einen Stein auf diese Position. (vgl. Abbildung B.12 auf Seite 84) Bei der Bewertung werden eigene längere Spielsteinketten ausgebaut und versucht einen Sieg zu erzielen. Bei der Optimierung können die Studenten eigene Ideen einbauen und zum Beispiel den Min-Max-Algorithmus erweitern.

Der Aufbau der Strategien wurde ebenfalls am bloomschen Taxonomie Konzept gegliedert. (vgl. Bloom und Engelhart (1972), S. 31) Die erarbeiteten Konstrukte aus der TRY-Phase müssen bei der ersten Strategie angewendet werden. Zum Beispiel ist die Array-Struktur notwendig um die freien Positionen auf dem Spielbrett zu zwischenspeichern. Bei der Programmierung der zweiten Strategie sind die Analyse des Verhaltens des Programms und die Modifizierung des Quellcodes notwendig. Bei einem Problem muss das Studentenpaar in der Lage sein, das entsprechende Problem zu erklären und im Quellcode entsprechende Modifizierungen anzubringen. Bei der dritten Strategie müssen die Methoden der vorherigen

Strategien genutzt werden. Die berechneten Werte aus den Methoden müssen für die Entscheidung zusammengesetzt werden. Der Min-Max-Algorithmus bietet eine Hilfestellung, um eine sinnvolle Struktur zu entwickeln. Bei der Optimierung der Strategie muss das Studentenpaar Entscheidungen auf Grundlage von eigenen Bewertungen treffen. Nachdem eine Idee für die Optimierung entstanden ist, sollte diese zunächst theoretisch überprüft werden, ob mit diesem Schritt eine Verbesserung erzielt werden kann.
Am Ende der DO-Phase sollte jedes Studentenpaar eine lauffähige Strategie für das Tablet implementiert haben. Je mehr sinnvolle Optimierungsschritte erledigt werden, umso besser ist die Leistung des programmierten Spielers.

4.3.4. EXPLAIN-Phase

In der abschließenden EXPLAIN-Phase müssen die Lösungen von den Studenten beschrieben werden. Das Erklären der eigenen Lösung erwirkt eine Metakognition des Wissens. Die Studenten benötigen einen Überblick über das Wissen des gesamten Workshops. Die Formulierung zur Erklärung der Inhalte verknüpft die verschiedenen erlernten Themen aus dem Workshop miteinander. (vgl. Fässler (2007), S. 121)
Diese abschließende Phase wird für den Workshop mit einem kleinen Wettkampf der verschiedenen Implementierungen eingeleitet. (vgl. Abbildung B.13 auf Seite 85) Dieser Wettkampf fördert bei der Optimierung der eigenen Strategie die Motivation für weitere Schritte.

Die jeweils beste lauffähige Computerstrategie von zwei Studentengruppen kann auf ein Tablet-Gerät übertragen werden. Anschließend können die beiden Strategien automatisch gegeneinander spielen. Der Wettkampf fördert den Dialog zwischen den verschiedenen Gruppen und die Implementierung und die Optimierungsschritte werden, nach konkreter Aufforderung eines Tutors, untereinander erklärt. Der Wettkampf ist angesichts der beschränkten Zeit eine gute Alternative zu einer Präsentation der verschiedenen Implementierungen.

4.4. Implementierung der Android-App

4.4.1. Aufbau des Programms

Bei der Implementierung der Go-Moku App stand die Benutzerfreundlichkeit des gesamten Quellcodes im Vordergrund. Die Teilnehmer am Workshop müssen in der Lage sein den Quellcode in relativ kurzer Zeit zu verstehen und Ergänzungen anbringen zu können. Für die Übersichtlichkeit wurden die drei Pakete

- activity,
- intelligence und
- logic

angelegt. Die Aufgaben, die in den Paketen erledigt werden, sind die Verwaltung der graphischen Oberfläche (activity), die künstliche Intelligenz zu simulieren (intelligence) und den Spielablauf inklusive

der Gewinnerkennung durchzuführen (logic). In den Paketen gibt es verschiedene Klassen mit möglichst kurzen und verständlichen Methoden.

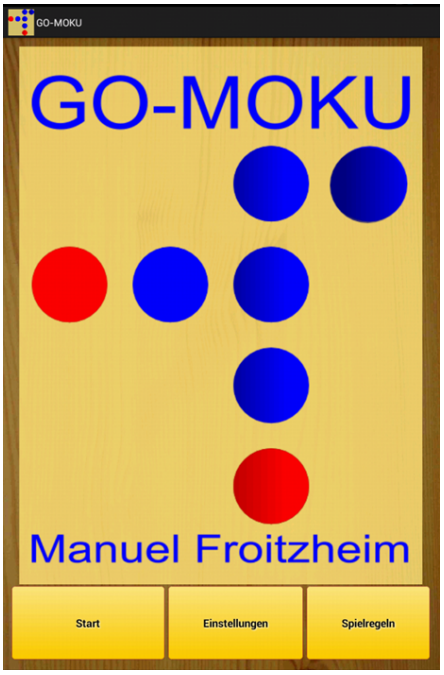
Die App besteht aus den vier Ansichten (vgl. Abbildung 4.2):

- Startansicht,
- Spielregeln,
- Einstellungen und
- Spiel

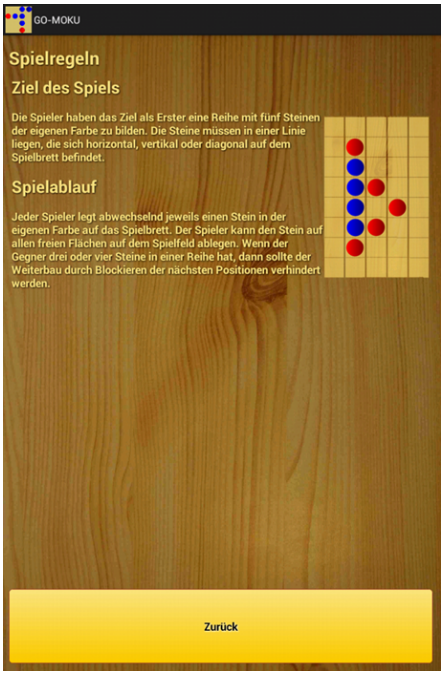
für die unterschiedlichen Teilbereiche des Programms. Nach dem Start der App wird dem Benutzer die Startansicht gezeigt. (vgl. Abbildung 4.2a) Von dieser Stelle kann er zu den drei unterschiedlichen Ansichten der App gelangen.

Auf der **Spielregel-Ansicht** (vgl. Abbildung 4.2b) bekommt der Nutzer der App die Spielregeln erklärt. Dies ist im Workshop nicht notwendig, weil die Teilnehmer zu den Spielregeln ein Arbeitsblatt bekommen haben, allerdings ist diese Funktion notwendig, wenn die App an andere Personen, die das Spiel eventuell nicht kennen, weitergegeben wird.

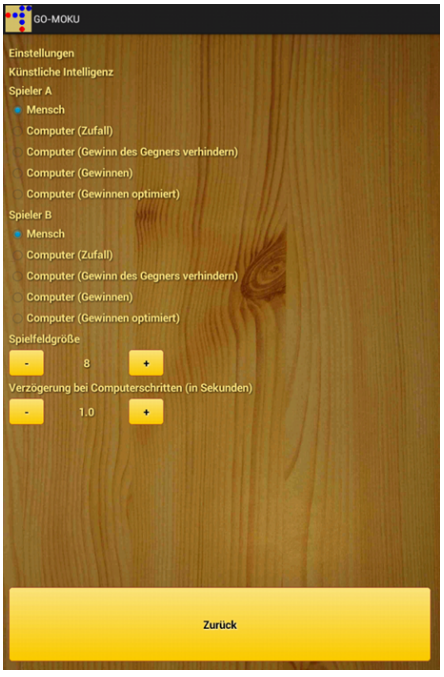
Die **Einstellungs-Ansicht** (vgl. Abbildung 4.2c) bietet die Möglichkeit den Spielmodus, die Größe des Spielbretts und die Zeitverzögerung bei zwei Computerspielern festzulegen. Beim Spielmodus kann der Benutzer für Spieler A und Spieler B festlegen, ob es sich um einen Menschen oder einen Computerspieler handelt. Bei einem Computerspieler können verschiedene Strategien ausgewählt



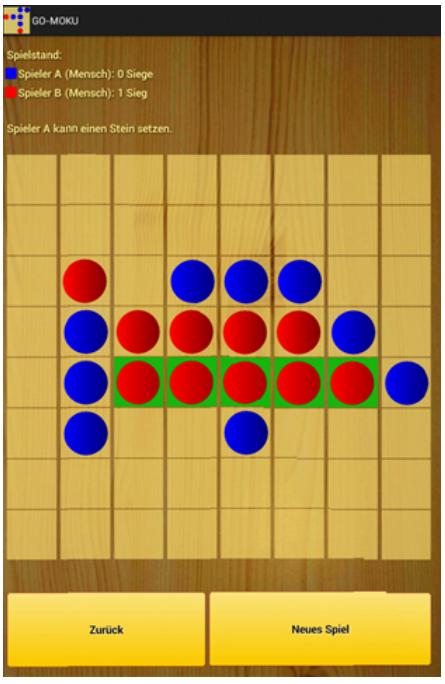
(a) Startansicht



(b) Spielregeln



(c) Einstellungen



(d) Spiel

Abbildung 4.2.: Bildschirmfotos der App

werden. Dies dient zum einen dazu, dass die Zwischenstände der Spielstrategien während des Workshops gespeichert werden können und auch ausgeführt werden. Zum anderen dient es dazu, dass man von unterschiedlichen Strategien die Stärke testen kann, indem man die Strategien gegeneinander antreten lässt. Die Spielfeldgröße kann zwischen acht mal acht Spielfeldzellen und neunzehn mal neunzehn Spielfeldzellen flexibel eingestellt werden. Die Spielfeldgröße von acht mal acht Spielfeldzellen ist in Europa eine weitverbreitete Spielfeldgröße. Diese Größe des Spielfelds, angelehnt an das klassische Dame-Spielbrett, haben die Studenten auch als Brettspiel vorliegen. Die Spielfeldgröße neunzehn mal neunzehn Spielfeldzellen beruht auf dem chinesischen Originalspiel. Die Zeitverzögerung ist für ein Spiel zwischen zwei Computerspieler von Interesse. Die Verzögerung kann mit null Sekunden ausgeschaltet werden, wenn man zum Beispiel nur überprüfen möchte, welche Strategie besser ist. Eine Zeitverzögerung von einigen Sekunden ist für die Kontrolle der Abfolge der Spielzüge sehr hilfreich. Bei zwei Computerspielern kann der Benutzter, unabhängig von der eingestellten Zeit, eine Ausführung in einzelnen Schritten wählen.

Die **Spielansicht** (vgl. Abbildung 4.2d) ist die interessanteste Ansicht, weil diese Ansicht für das eigentliche Spiel genutzt wird. Im oberen Teil des Bildschirms werden der aktuelle Spielstand und der Spieler, der den nächsten Zug machen soll, angezeigt. Darunter wird das Spielbrett in der zuvor eingestellten Größe angezeigt. Im unteren Bereich der Ansicht kann ein neues Spiel gestartet werden. Wenn zwei Computerspieler gegeneinander antreten, kann der

nächste Zug ausgelöst werden oder die Simulation, bis zum Sieg eines Computerspielers gestartet werden. Um als Mensch einen Stein auf dem Spielbrett zu positionieren, ist die Berührung einer freien Spielfeldzelle auf dem Spielbrett ausreichend.

4.4.2. Spiellogik

Für das Go-Moku Spiel gibt es zahlreiche Lösungsstrategien und daraus resultieren unterschiedliche Möglichkeiten für die Implementierung. Weil die Spiellogik auf einem Tablet-Computer ausgeführt werden soll, ohne auf eine große Datenbank zurückzugreifen (vgl. Allis (1994), S. 150), muss bei der Auswahl auf entsprechende Ressourcenschonung geachtet werden.

Im Jahr 1994 wurde nachgewiesen, dass der Spieler, der den ersten Stein legt, bei einer optimalen Strategie immer gewinnt. (vgl. Allis u. a. (1993), S. 2) Bei dieser Strategie wird das Gefahrenpotential an jeder Stelle des Spielbretts bestimmt und mit einer komplexen Baumsuche mit einer Tiefe von 35 der optimale Zug bestimmt. Trotz aller Verbesserungen wurde eine Baumgröße von 138.790 Knoten erzeugt und laut Angaben der Autoren dauerte es 1,1 Millionen Sekunden (circa 12 Tage), bevor das Ergebnis zu einem Zug vorlag. Für den praktischen Einsatz ist dieser Algorithmus nicht geeignet. (vgl. Allis u. a. (1993), S. 9)

Eine praktisch nutzbare Strategie, mit verkraftbar schlechteren Ergebnissen als rechenintensiverere Strategien wurde von Zahle vorgestellt. (vgl. Zahle (1988), S. 325-338) Der Algorithmus basiert

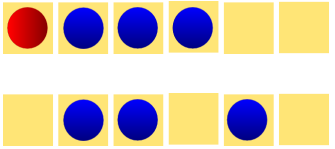
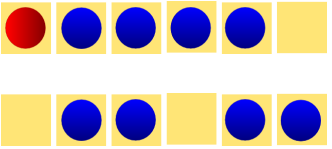
	Dreier-Spielsteinreihe	**Vierer-Spielsteinreihe**
Offen:		
Geschlossen:		

Tabelle 4.1.: Unterscheidung typischer Spielsteinkonstellationen

auf einem zwei Schritte-Konzept. Im ersten Schritt wird überprüft, ob es eine Siegersequenz gibt und im zweiten Schritt wird ein strategischer Zug gemacht.

Für die Überprüfung auf eine Siegersequenz werden Reihen mit drei oder vier Steinen auf dem Spielbrett gesucht. Des Weiteren wird zwischen offenen und geschlossenen Reihen unterschieden. Eine geschlossene Spielsteinreihe liegt vor, wenn genau eine Position existiert, um die Reihe fortzuführen. Zum Beispiel, wenn eine Lücke in der Reihe ist, die gefüllt werden muss, oder eine Reihe bereits an einer Seite blockiert ist, liegt eine geschlossene Reihe vor, weil die Position bzw. die Positionen an denen der Stein bzw. die Steine positioniert werden können, eindeutig bestimmt sind. Im Gegensatz dazu gibt es bei einer offenen Reihe zwei Möglichkeiten, um einen Sieg herbeizuführen, weil an beiden Seiten ein Stein angelegt werden kann. Bei einer offenen und einer geschlossenen Viererreihe, sowie einer offenen Dreierreihe hat der Gegner im nächsten oder übernächsten Zug gewonnen, wenn keine Blockade angebracht wird. (vgl. Tabelle 4.1) Deswegen soll dies mit einer

Blockade beim Gegner verhindert werden und umgekehrt für die eigene Strategie angestrebt werden.

Wenn kein Sieg des Gegners oder ein eigener Sieg in Aussicht ist, kann ein strategischer Zug gemacht werden. Aus strategischen Gründen sollten eigene Reihen erweitert werden. Es ist notwendig zu überprüfen, an welcher Stelle auf dem Spielbrett Spielsteinreihen vorhanden sind. Für jede freie Position auf dem Spielbrett ist testweise ein Stein zu legen und anschließend die Anzahl an Zweier-, Dreier-, Vierer- und Fünferreihen zu bestimmen. Mögliche Bewertungen sind der Tabelle 4.2 zu entnehmen. Diese Werte wurden aus Testläufen, bei denen ein Computerspieler gegen einen Computerspieler angetreten ist, berechnet. Auf jede freie Spielfeldzelle wird ein temporärer Spielstein platziert und anschließend das gesamte Spielfeld bewertet. Bei der Bewertung des Spielfelds werden die einzelnen Spielsteinkonstellationen für das gesamte Spielbrett summiert. Ein höherer Wert beschreibt eine bessere Siegchance.

Eine Optimierungsmöglichkeit besteht in der Distanzmessung. Wenn zu Beginn des Spiels noch kein Spielstein auf dem Spielbrett liegt oder die Wahl zwischen zwei gleichwertigen Feldern besteht, legen erfahrene Go-Moku Spieler die ersten Spielsteine in den mittleren Teil des Spielbretts. Mithilfe einer Distanzmessung, zum Beispiel mit dem Satz von Pythagoras, kann die Entfernung zum Mittelpunkt des Spielbretts festgestellt werden. Diese Distanz kann zu der bisherigen Bewertung eines jeden Spielfelds mit einbezogen werden. Dadurch orientiert sich der Algorithmus bei der Auswahl der nächsten Spielsteinpositionierung an der Mitte des Spielfelds.

Wenn die Bewertung, ohne Optimierung ausschließlich auf die Länge der Spielsteinreihen basierend, für jedes freie Feld bestimmt wurde, kann mit einem vereinfachten Min-Max-Algorithmus der beste Wert ausgewählt werden. Wenn ein Stein zur Blockade des Gegners auf dem Spielbrett positioniert werden soll, ist dieser Wert zu minimieren. Wenn hingegen ein eigener Stein gelegt wird, ist dieser Wert zu maximieren. Durch die beschränkte Rechenleistung auf einem Tablet-Computer und der kurzen Zeit für die Implementierung wird auf die Rekursion verzichtet, also nur eine Ebene im Baum betrachtet. In der optionalen Optimierungsphase können die Studenten diese Erweiterung bei genügend verbleibender Zeit dem Projekt hinzufügen.

Nebeneinander-liegende Steine	Bewertung geschlossene Spielsteinreihe	Bewertung offene Spielsteinreihe
5	200	500
4	100	200
3	40	110
2	10	50

Tabelle 4.2.: Übersicht über die Bewertung für Spielsteinreihen

4.5. Beschreibung und Analyse der Lernmaterialien

4.5.1. Spielbrett

Für den Einstieg in den Workshop wurde für jede Gruppe ein Go-Moku Spielbrett (vgl. Abbildung B.1 auf Seite 73) und zugehörige Spielsteine zur Verfügung gestellt. Das Spielbrett kann zum einen für den Einstieg zum klassischen Spielen genutzt werden. Das klassische Spielen dient dem Verständnis der Spielregeln und damit sich die Studenten kennenlernen. Zum anderen können mithilfe des Spielbretts während des Workshops Spielsituationen nachgebildet werden. Dies kann zum Beispiel bei Diskussionen über Modifizierungen genutzt werden.

4.5.2. Arbeitsmappe

Die Arbeitsmappe als Sammlung der verschiedenen Arbeitsblätter ist für den Workshop besser geeignet als eine Lerneinheit in einer Online-Plattform. Bei einer Online-Plattform muss auf dem Bildschirm zwischen dem Quellcode und den Arbeitsaufträgen gewechselt werden. Vor allem im ersten Teil mit stark geleiteten Arbeitsaufträgen wäre dies kompliziert.
Das erste Arbeitsblatt (vgl. Abbildung B.2 auf Seite 74) beinhaltet eine kurze Erklärung der Spielregeln und einen Arbeitsauftrag zur Sicherung der Ergebnisse, die beim Spielen entstehen. Die Studen-

ten sollen sich beim Spielen Gedanken zur angewendeten Strategie machen. Diese Ergebnisse können später bei der Optimierung der eigenen Strategie eingesetzt werden. Um eine ausreichende Strukturierung der Notizen zu erzielen, wurden einige Leitfragen für den Arbeitsauftrag formuliert.

Auf dem zweiten Arbeitsblatt (vgl. Abbildung B.3 und B.4 auf den Seiten 75 und 76) ist eine Einführung in die Entwicklungsumgebung Eclipse abgedruckt worden. Der Einsatz von Eclipse wird in den ersten Semestern von vielen Hochschuldozenten nicht empfohlen. Die Entwicklungsumgebung nimmt dem Benutzer sehr viel Schreibarbeit bei den Grundkonzepten der Programmierung ab. Zum Beispiel können Getter- und Setter-Methoden automatisch generiert werden. Diese Möglichkeiten sind bei den Klausuren nicht vorhanden und deswegen ist Eclipse für die Grundlagenvorlesungen nicht vorgesehen. Für den Workshop wurde trotzdem Eclipse ausgewählt, weil es zum einen die Standardentwicklungsumgebung für die Android-Programmierung darstellt und als solche auch in vielen Lehrbüchern zur Android-Programmierung genutzt wird. (Künneth (2012), S. 29 und Post (2012), S. 69) Zum anderen verwendet eine große Zahl der Studenten diese Entwicklungsumgebung. Die Erklärungen auf dem Arbeitsblatt sind auf die wesentlichen Merkmale, die für den Workshop notwendig sind, beschränkt. Eine Einführung, in die weiteren Möglichkeiten der Entwicklungsumgebung, würde nur Zeit kosten und die Aufmerksamkeit für das Wesentliche im Workshop nehmen. Wenn die Teilnehmer am Workshop mit der Entwicklungsumgebung vertraut sind, kann dieses Arbeitsblatt übersprungen werden.

Das dritte Arbeitsblatt (vgl. Abbildung B.5 auf Seite 77) beinhaltet eine Einführung in die Struktur des Programmgerüsts. Das Programmgerüst ist etwas komplexer als die Beispiele, mit denen die Studenten in den ersten Semestern konfrontiert werden. Durch die Aufteilung in verschiedene Quellcodepakete und Klassen ist die Übersichtlichkeit des Quellcodes nach einer kurzen Einführung deutlich größer, als wenn sehr lange Klassen gegeben wären. Zur zusätzlichen Unterstützung wurde den Studenten eine Dokumentation online zur Verfügung gestellt. Mit einem Symbol wird bei den weiteren Aufgaben, bei denen die Dokumentation hilfreich sein kann, darauf hingewiesen.

Auf den folgenden Arbeitsblättern wird die Programmierung der App schrittweise beschrieben. Zunächst sind stark angeleitete Aufgaben ausgewählt worden, damit sich die Studenten im Quellcode und in der Entwicklungsumgebung zurechtfinden. Im Verlauf der Arbeitsblätter werden die Arbeitsaufträge offener und die Studenten können immer mehr eigene Ideen in die Programmierung einfließen lassen. Bei der Programmierung des Spielbretts (vgl. Abbildung B.6 und B.7 auf den Seiten 78 und 79) ist dies nicht möglich, weil das Spielbrett durch die Spielregeln vorgegeben ist und die Struktur zum vorgegebenen Programmgerüst passen muss. Beim Arbeitsblatt für das Spielbrett müssen die Studenten ein mehrdimensionales Array und einige Methoden, um auf dieses Array zuzugreifen, erzeugen.

Das nächste Arbeitsblatt (vgl. Abbildung B.8 und B.9 auf den Seiten 80 und 81) dient zur Programmierung einer erweiterten Array-Struktur. Bei den später zu implementierenden Spielstrategien wird

eine einfache Möglichkeit zur Speicherung der Spielpositionen innerhalb einer Strategie benötigt.

Auf den folgenden Arbeitsblättern (vgl. Abbildung B.10, B.11 und B.12 auf den Seiten 82 bis 84) werden die Studenten zur Programmierung der Strategie angeleitet. Die Implementierung der Strategie erfolgt in drei Teilschritten, wobei in jedem Teilschritt die vorherige Strategie modifiziert wird und somit eine neue Strategie entsteht. Bei den Studenten entstehen regelmäßig Erfolgserlebnisse. Wenn eine Gruppe nur einen Teil der Arbeitsaufträge schafft, haben diese Studenten trotzdem ein lauffähiges Programm. Die Teilschritte sind aufeinander aufbauend und mit steigendem Schwierigkeitsgrad. Der erste Schritt beinhaltet lediglich eine Zufallsstrategie. Diese Strategie ermöglicht den Studenten den ersten Kontakt mit dem Konzept der künstlichen Intelligenz in der App. Der zweite Teilschritt beinhaltet die Verhinderung des Siegs des Gegners und im dritten Schritt wird versucht, einen eigenen Sieg zu erzielen. Diese Schritte sind auf den Arbeitsblättern relativ offen beschrieben. Gute und schnelle Studenten können ihre eigenen Ideen einfließen lassen. Für Studenten, die Probleme bei den Teilschritten haben, gibt es ergänzende Arbeitsblätter. Auf diesen Arbeitsblättern werden mehr Hinweise zu den Aufgaben gegeben. (vgl. Abbildung B.14 und B.15 auf den Seiten 86 und 87)

Das letzte Arbeitsblatt (vgl. Abbildung B.13 auf Seite 85) bildet die Anleitung zu einem Wettkampf. Damit die Daten von einem Computer auf einen anderen Computer übertragen werden können müssen bestimmte Schritte in der Entwicklungsumgebung ausge-

führt werden. Ein Wettkampf ist ein zusätzlicher Motivationsfaktor für die Optimierung der eigenen Strategie.

4.5.3. Dokumentation

Für das Grundgerüst und die Methoden des Programms, die von den Studenten implementiert werden, steht eine Javadoc Dokumentation über eine Webseite zur Verfügung. (vgl. beiliegende CD) In dieser Dokumentation kann zu allen Methoden und Klassen des Programms eine Erklärung abgerufen werden. Die Übergabeparameter und Rückgabewerte werden erklärt, damit die Studenten die Methoden, ohne Betrachtung der expliziten Programmierung, nutzen können. Die Anweisungen für die Erstellung der Dokumentation stehen ebenfalls als Kommentare im Quellcode zur Verfügung und können bei Bedarf auch direkt im Quellcode nachgeschlagen werden.

5. Evaluation

5.1. Methodik

Die Evaluierung des Workshops in Bezug zu den Thesen aus Kapitel 3.3 wurde mit einem Fragebogen vor und nach dem Workshop als quantitative Erhebung durchgeführt. Des Weiteren wurde gegen Ende des Workshops ein Teil der Studenten informell nach ihren Eindrücken vom Workshop gefragt. Diese qualitative Erhebung dient als Ergänzung zu dem Fragebogen, weil bei der relativ kleinen Stichprobe von 20 Teilnehmern nur wenig signifikante Zusammenhänge ermittelt werden können. Die Kombination aus den Methoden der quantitativen und qualitativen Erhebung lassen einige Rückschlüsse auf die Thesen zu.

Durch die zwei Erhebungszeitpunkte wird ein Vergleich der Motivation und der Kenntnisse der Studierenden zu den verschiedenen Zeitpunkten ermöglicht. Im Vorfeld des Workshops wurden zusätzlich die Stammdaten und nach dem Workshop der Eindruck der Studenten erhoben.

Die beschriebenen Instrumente der Evaluation beruhen alle auf der

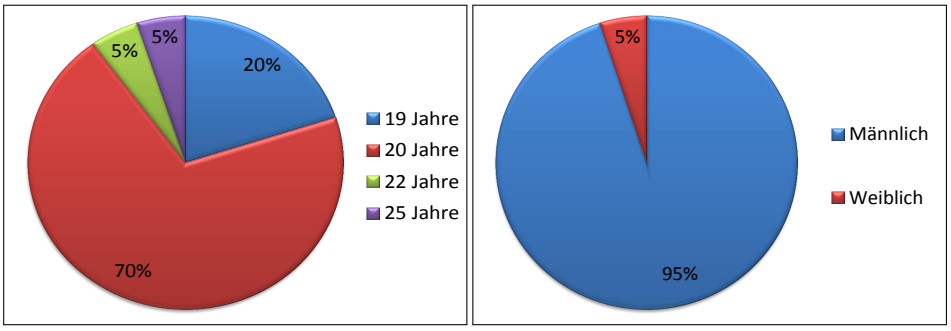
Selbsteinschätzung der Studenten. Die Programmierergebnisse der Studenten werden ebenfalls bewertet und in die Evaluierung einbezogen. Durch dieses Verfahren können objektiv bewertete Ergebnisse betrachtet und herangezogen werden.
Durch die dreiteilige Analyse aus Fragebogen, informellem Interview und bewerteten Arbeitsergebnissen wird trotz der kleinen Stichprobe ein relativ gutes Bild der Ergebnisse aus dem Workshop ermöglicht. (vgl. Diekmann (2011), S. 356)

5.2. Auswertung

5.2.1. Fragebogen

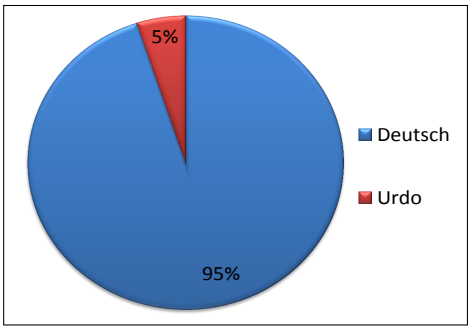
Am Workshop haben 20 Studenten teilgenommen. Beim ersten Erhebungszeitpunkt sind 20 Fragebögen ausgefüllt worden und zum zweiten Erhebungszeitpunkt sind 15 Fragebögen zurückgegeben worden. (vgl. Vor-Test ab Seite 113 und Nach-Test ab Seite 121) Vier Teilnehmer haben den Workshop aus verkehrstechnischen Gründen vorzeitig verlassen. Wegen der geringen Anzahl an Teilnehmern ist bei der gesamten Auswertung zu beachten, dass die Werte nur in geringem Maße verallgemeinerbar sind. Für eine markante Veränderung des Gesamtergebnisses sind nur vereinzelte Veränderungen notwendig.
Bei der Erhebung der **Stammdaten** wurde Geschlecht, Alter, Nationalität, Muttersprache, Studiengang und Fachsemester erhoben. (vgl. Abbildung E.1 auf Seite 113) Die ermittelten Werte

sind den Diagrammen in Abbildung 5.1 auf Seite 50 zu entnehmen. Die Mehrheit der Studenten war im Alter von 19 bis 20 Jahre alt. Bei der Nationalität und der Muttersprache waren fast alle Teilnehmer gebürtige Deutsche mit einer Sozialisation im deutschen Sprachraum. Das Geschlechtsmerkmal war mit 95 Prozent männliche Teilnehmer und entsprechend nur 5 Prozent, genau eine Studentin, weibliche Teilnehmer, nicht ausgewogen verteilt. Der formale Bildungsabschluss im tertiären Bildungsbereich (Fach- oder Hochschulabschluss) der Eltern liegt bei 50 Prozent. (vgl. Abbildung E.33 auf Seite 126) Dieser Wert ist deutlich über dem Durchschnitt in der deutschen Bevölkerung mit circa 30 Prozent. (vgl. Geissler (2010), S. 278) Die angeworbenen Studenten sind fast alle im zweiten Fachsemester. Nur ein Student ist im vierten Fachsemester.

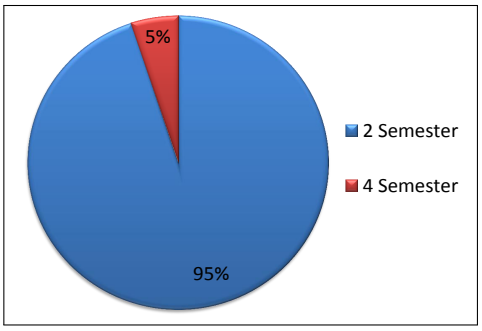


(a) Altersstruktur

(b) Geschlecht



(c) Muttersprache



(d) Studiensemester

Abbildung 5.1.: Auswertung der Stammdaten

Im Bereich der **allgemeinen Einstellung zum Studium** sind die Antworten bei fast allen Fragen stark gestreut über die verschiedenen Merkmalsausprägungen. (vgl. Abbildung E.1 auf Seite 113) Die Fragen in diesem Bereich wurden zu beiden Erhebungszeitpunkten gestellt. Für alle Fragen in dieser Kategorie wurde eine Skala von sieben Merkmalsausprägungen mit der Randbeschriftung „trifft gar nicht zu“ und „trifft vollkommen zu“ gewählt. (vgl. Abbildung E.7 auf Seite 115) Bei der Frage „Ich bin mir sehr unsicher, ob das Studium das Richtige für mich ist“, sind nur acht Studenten der Meinung, dass diese Aussage gar nicht zutrifft. Zwei weitere Studenten wählten die nebenliegende Kategorie aus. (vgl. Abbildung E.34a auf Seite 127) Die Auswahl ist bei diesen zehn Studenten

bei beiden Erhebungszeitpunkten identisch. Bei den weiteren Merkmalsausprägungen ist eine Tendenz zu mehr Unsicherheit nach dem Workshop festzustellen. Die Studenten finden das Lernen in der Universität spannend und wollen im Studium kompetenter werden. Diese Werte sind während des Workshops konstant geblieben. (vgl. Abbildung E.34b auf Seite 127) Vor dem Workshop spielte der Druck von außen, damit im Studium mehr geleistet wird, nur eine kleine Rolle. Während des Workshops veränderten sich die Einschätzungen der Studenten erheblich. Nach dem Workshop gaben deutlich mehr Studenten an, dass sie auf diesen Druck angewiesen sind.

Für die Kategorie des **Lernens während des Studiums** wurde die gleiche Skala wie bei der Einstellung zum Studium gewählt. (vgl. Abbildung E.8 auf Seite 115) Die Studenten sollen eine Selbsteinschätzung abgeben zum persönlichen Lernverhalten im Studium. Es wird die Erreichbarkeit der eigenen Ziele im Studium eingeschätzt. Des Weiteren wird die Gefühlslage zu anstehenden Herausforderungen getestet. Die Selbsteinschätzung, dass man in der Lage ist, den Stoff des Studiums zu lernen und das Vertrauen in die eigenen Fähigkeiten den Stoff zu lernen. Die Antworten zu den vier Aussagen in diesem Bereich veränderten sich während des Workshops in Richtung „trifft vollkommen zu“. Nach dem Workshop fühlt sich ein Teil der Studenten besser in der Lage die Anforderungen im Studium zu erfüllen und ist den Herausforderungen eher gewachsen. (vgl. Abbildung E.36 auf Seite 129)

In der folgenden Kategorie erhielten die Studenten Aussagen zu ihrem **Umgang mit Problemen** im Studium. (vgl. Abbildung E.9

auf Seite 116) In dem Fragenblock wird der Umgang mit Problemen im Studium überprüft. Eine Frage ist zum Beispiel: „Wenn eine neue Sache auf mich zukommt, weiß ich, wie ich damit umgehen kann.“ Als Skala für die Fragen in dieser Kategorie wurden die vier Begriffe „stimmt nicht“, „stimmt kaum“, „stimmt eher“ und „stimmt genau“ gewählt. Die Stimmungslage ist zu beiden Erhebungszeitpunkten sehr positiv. Die Studenten gehen davon aus, dass die anstehenden Herausforderungen im Studium gut gemeistert werden können. Die Stimmungslage hat sich während des Workshops bei allen Fragen kaum verändert. Die geringen Veränderungen, die aufgetreten sind, wirkten sich positiv auf die Stimmung aus. (vgl. Abbildung E.37 auf Seite 130)

Die meisten Teilnehmer am Workshop hatten in der **Schule Informatikunterricht**. Die Hälfte der Teilnehmer haben durchgängig von der Unterstufe, über die Mittelstufe bis zur Oberstufe ein Informatikangebot an der Schule und haben an diesem auch teilgenommen. Lediglich drei Teilnehmer haben in ihrer Schulzeit keinen Informatikunterricht genossen. Ein sehr großer Anteil hatte an der Schule das Angebot eines Leistungskurses in der Oberstufe. Die Studenten wählten zu 95 Prozent Mathematik als Leistungskurs in der Oberstufe. Physik, Erdkunde und Sozialwissenschaften sind von mehr Studenten gewählt wurden, als ein Leistungskurs in Informatik. Trotz des Angebots und des späteren Studiums der Informatik waren die Studenten in ihrer Schulzeit an einem Informatik-Leistungskurs nicht interessiert. Der Abiturdurchschnitt, der 13 Workshop Teilnehmer die bei dieser Frage geantwortet haben,

liegt bei 2,1. Dies kommt zustande, durch einen ungefähr gleich großen Anteil an Studenten mit einer guten und einer befriedigenden Abiturnote. Die 19 Prozent der Teilnehmer mit einer sehr guten Note heben den Durchschnitt deutlich an. (vgl. Abbildung E.39 auf Seite 132)

Die Studenten wurden zu ihren **Kenntnissen in verschiedenen Programmiersprachen** befragt. Für jede Programmiersprache sollten die Studenten auswählen aus den Kategorien „sehr sicher“, „eher sicher“, „eher unsicher“, „sehr unsicher“ und „nie eingesetzt“. Alle Studenten haben die Programmiersprache Java schon mal eingesetzt. Dies ist durch das Vorlesungsangebot an der der Universität auch nicht anders möglich. Allerdings hat keiner „sehr sicher“ und nur einer „eher sicher“ gewählt. Bei der Programmiersprache Java haben die wenigsten Studenten angeben, dass diese Programmiersprache „eher sicher“ beherrscht wird. Somit erzielen alle anderen Programmiersprachen bessere Werte. (vgl. Abbildung 5.2 auf Seite 54)

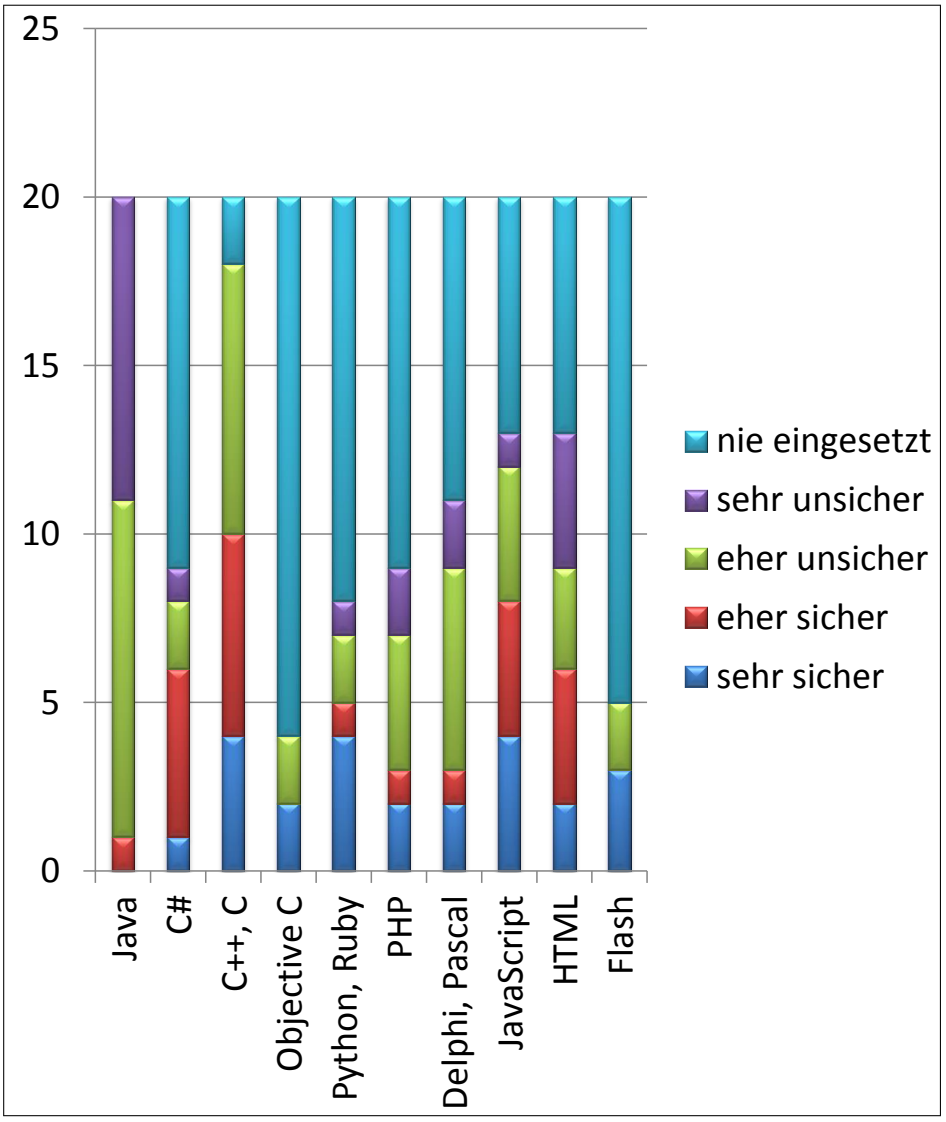


Abbildung 5.2.: Wie sicher fühlen Sie sich im Umgang mit folgenden Programmiersprachen?

Bei den kleinen Aufgaben zur Java-Programmierung sind die Ergebnisse vor dem Workshop gut. Es gibt nur vereinzelte Ausreißer, die keine nennenswerte Abweichung darstellen.

Im Fragebogen für den zweiten Erhebungszeitpunkt waren einige Fragen enthalten, die sich direkt auf den Workshop beziehen. (vgl. Abbildung E.22 auf Seite 123) Fast die Hälfte der Studenten erachten

den **Schwierigkeitsgrad des Workshops** als angemessen und die anderen Studenten als eher einfach. (vgl. Abbildung E.45 auf Seite 136) Bei der Frage nach der Ermutigung zur Studienwahl, durch den Workshop, hat die Hälfte der Studenten eine Angabe gemacht. Diese Frage soll überprüfen, ob die Studenten durch den Workshop in der Studienwahl bekräftigt wurden. Diese Studenten waren bis auf eine Ausnahme vom Workshop in ihrer Studienwahl bestärkt worden. Der Workshop ist von den Studenten im Schulnotensystem im Durchschnitt mit einer 1,9 benotet worden. (vgl. Abbildung E.47 auf Seite 137) Bei der Begründung der Noten und den Verbesserungsvorschlägen wurde des Öfteren die fehlende Vermittlung der Grundlagen der Android-Programmierung genannt. (vgl. Seite 137)

5.2.2. Befragung der Studenten

Nach dem Workshop wurde ein Teil der Studenten informell befragt. Ein Großteil der Studenten hatte durch die Ankündigung für den Workshop die Hoffnung eine **detaillierte Einführung in die Android-Programmierung** zu erhalten. Die Grundlagenvermittlung von Inhalten aus der Vorlesung Datenstrukturen und Algorithmen ist laut Aussage der Studenten auf der Ankündigung zum Workshop untergegangen. (vgl. Abbildung A.1 auf Seite 69) Einige Studenten fanden die vier Zeitstunden relativ knapp für den Workshop und das umfangreiche Projekt. Die Studenten hätten sich mehr Zeit zur Optimierung der eigenen Strategie gewünscht. Insgesamt waren die Studenten vom Endprodukt begeistert und erstaunt,

dass in den vier Stunden ein komplettes Programm, natürlich mit den Hilfestellungen von den Arbeitsblättern und dem vorgegebenen Programmgerüst, entstehen konnte. Die klaren Formulierungen bei den Arbeitsaufträgen wurden von den Studenten gelobt.
Von einigen Paaren wurde die **Sozialform**, also die Partnerarbeit, als sehr gut bezeichnet. Die Studenten gaben an, dass durch die Partnerarbeit gegenseitige Hilfe möglich war. Durch die gegenseitige Hilfe wurde weniger Zeit bei Verständnisproblemen benötigt, weil die Unterstützung innerhalb der Arbeitsgruppe gegeben war.
Die Studenten lobten die Möglichkeit, die eigene App direkt auf einem Tablet-Gerät zu testen. Auf dem Tablet-Computer wirkt die App besser, als auf einem Android-Emulator.

5.2.3. Bewertung der Arbeitsergebnisse

Die Arbeitsergebnisse wurden nach dem Workshop gesichert und bewertet. Insgesamt wurden 50 Punkte für eine perfekte Lösung vergeben. Bei der Bewertung wurde auf die korrekte Programmierung inklusive der Beachtung von Ausnahmefällen geachtet. Beispielsweise sollte eine Beschränkung der Spielfeldgröße implementiert werden. Wenn diese nicht vorhanden ist, gibt es einen Punkt Abzug. Trotzdem ist eine Lösung ohne diese Beschränkung lauffähig.
Das beste Studentenpaar hat 47,5 Punkte und das schlechteste Paar hat, trotz einer vorzeitigen Abgabe, 22,5 Punkte erzielt. (vgl. Abbildung 5.3 auf Seite 57) Bei den Grundlagen der Hierarchisierung des Quellcodes gibt es zum Teil Mängel. Die Unterscheidung zwischen

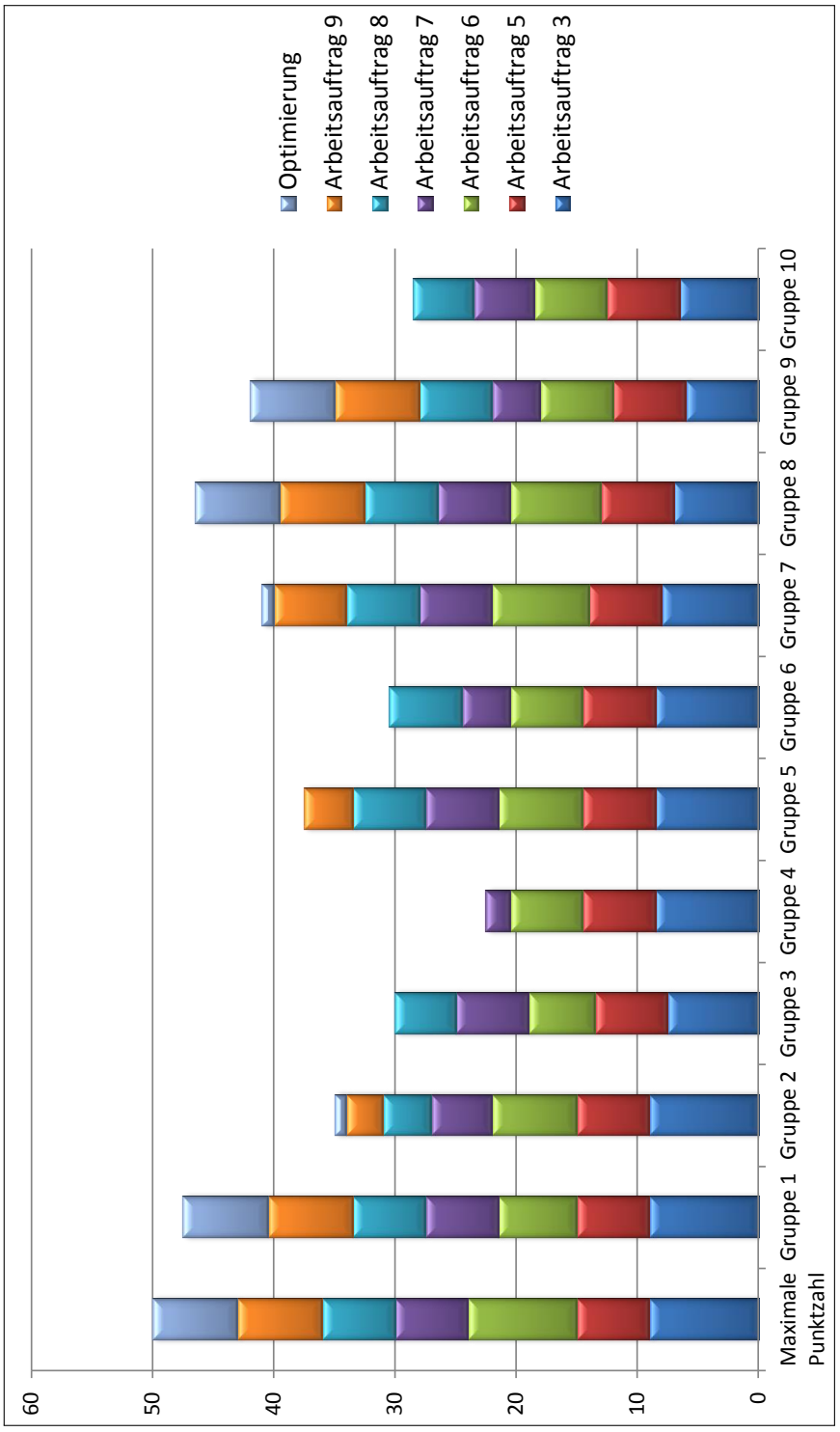


Abbildung 5.3.: Vergleich der Punkte der verschiedenen Gruppen

der Sichtbarkeit einer Variablen für eine Klasse oder ein Paket wurde zum Großteil falsch gesetzt.
In den stark strukturierten Aufgaben im ersten Teil sind die Ergebnisse bei den meisten Gruppen sehr ähnlich. Bei den Aufgaben zur Spielstrategie unterscheiden sich die Lösungen deutlich stärker. Bei der Optimierungsaufgabe sind bei drei Gruppen sehr gute Ergebnisse entstanden. Die verschiedenen Gruppen haben unterschiedliche Ansätze verfolgt. Eine Gruppe hat den Min-Max-Algorithmus um die Rekursion erweitert. Wie in Kapitel 4.4.2 beschrieben sind die Tablet-Geräte zu schwach, um eine Rekursion in einer angemessenen Zeit für ein großes Spielbrett durchzuführen. Die beiden anderen Gruppen haben die Evaluierung der Spielsituation optimiert, indem die Werte für die Stellung der Steine verändert und weitere Konstellationen hinzugefügt wurden. Beispielsweise sind L-Förmige Steingruppen in die Evaluierung des Spielbretts einbezogen worden.

5.2.4. Eindrücke des Autors

Während des Workshops sind bei den Teilnehmern vor allem Probleme durch Tippfehler entstanden. Eine falsch gesetzte Variable oder das Vertauschen eines Index gehörten zu den meist vorkommenden Fehlern. Bei drei Gruppen kam es zu Verständnisproblemen beim Umgang und der Initialisierung des Konstruktors für das Spielfeld. Die Arbeitsatmosphäre war während des gesamten Workshops als sehr angenehm zu bezeichnen. Die einzelnen Studentenpaare halfen sich bei Problemen und erklärten sich bei Verständnisproblemen

den Quellcode gegenseitig.

Die Zeitplanung (vgl. Kapitel A.1 auf Seite 71) wurde von den meisten Gruppen eingehalten. Zwei Gruppen waren bei den ersten Aufgaben deutlich schneller fertig. Diese Gruppen haben bei der Optimierung mehr Zeit gehabt und konnten mehr eigene Ideen in ihre Strategie einfließen lassen. Alle Paare haben eine lauffähige Zufallsstrategie implementiert.

Bei den Studenten bestand großes Interesse an der Android-Programmierung und nach tiefergehenden Veranstaltungen in diesem Bereich. Das Projekt wurde von vielen Teilnehmern auf Speichergeräten mitgenommen, um bei Gelegenheit weiter bearbeitet zu werden.

5.3. Schlussfolgerungen

Die Teilnehmer am Workshop haben sich freiwillig zum Workshop angemeldet und sind nicht unbedingt die zuerst angestrebte Zielgruppe. Es sollten die Studenten mit Problemen in Datenstrukturen bzw. der Programmierung am Workshop teilnehmen. Aus den Evaluationsergebnissen bilden die Teilnehmer am Workshop das Mittelfeld der Studenten. Zum einen haben die Studenten bereits in der Schule Informatik gehabt und zum anderen sind Kenntnisse in den verschiedensten Programmiersprachen vorhanden. Die Aufgaben im Fragebogen zu den Programmierkenntnissen wurden sehr gut beantwortet und kaum Fehler gemacht. Dies erklärt die Einschätzung der Studenten, dass der Workshop von einem Teil als einfach eingeschätzt

wurde. Bei schwächeren Studenten wäre dieses Ergebnis anders ausgefallen. Bei vielen Studenten bestand die Motivation am Workshop teilzunehmen, um einen Einblick in die Android-Programmierung zu erhalten. Dies konnte in dem Workshop nur zum Teil geleistet werden, weil deutlich schlechtere Programmierkenntnisse erwartet wurden.

Die Motivation war während des Workshops sehr hoch. Es gab für den Workshop keinerlei Übungspunkte oder andere Vergütungen. Am frühen Freitagabend ist den Studenten ein inhaltlich angemessenes Angebot zu machen, damit die Motivation erhalten bleibt. In den individuellen Befragungen wurde die Idee für ein Projekt, indem ein Spiel programmiert wird, unter Einsatz von Tablet-Geräten, positiv bewertet. Eine fertige Android-App mit einer graphischen Oberfläche ist für einen Studenten in den ersten Semestern interessanter als eine Konsolenausgabe auf dem Bildschirm. Die **erste These**, dass der Einsatz von mobilen Endgeräten und Computern zur Implementierung einer Android-App, die Motivation stärkt, konnte beim Testlauf nicht widerlegt werden. Dies wird auch in den Evaluationsergebnissen zur Motivation für das Studium in sehr geringem Ausmaß deutlich. Eine größere Veränderung ist durch einen punktuellen Workshop im Studium auch nicht zu erwarten.

Die **zweite These**, die zu Beginn der Arbeit aufgestellt wurde, besteht in der Erfahrung der Studenten, an einem Projekt zu arbeiten. In einem Informatikprojekt gehören Teamfähigkeit und Kommunikation zu den wichtigsten Indikatoren für ein erfolgreiches Projekt. Die Partnerarbeit ist bei allen Paaren sehr gut gelungen. Die Studenten

haben sich bei den Fähigkeiten ergänzt und gemeinsam das Projekt nach vorne gebracht.

Die **dritte These** zur Vertiefung der Programmierkenntnisse und der Datenstrukturen wurde im Workshop ebenfalls erfüllt. Bei den Fragen im Workshop konnte festgestellt werden, dass bei einigen Sprachkonstrukten Probleme bestehen. Diese Probleme konnten entweder vom Partner oder von einem Betreuer beseitigt werden. Die Anwendung der Konstrukte aus den Vorlesungen in einem Projekt und die praktische Relevanz, zum Beispiel von Getter- und Setter-Methoden, verdeutlichte die Relevanz der Inhalte. Die Studenten gaben an, dass sie sich zum Teil die Anwendung im Vorfeld als unnötig oder umständlich vorgestellt hatten.

Bei der **vierten These** zur Motivation für das Studium konnten kleine Erfolge verzeichnet werden. In der Kategorie zur Motivation für das Studium zu lernen konnte in der quantitativen Erhebung eine Verbesserung ermittelt werden. Diese Erfolge sind relativ klein und direkt im Anschluss an den Workshop erhoben. Es ist gut möglich, dass nach einigen Wochen oder Monaten diese zusätzliche Motivation verflogen ist. Erst einmal ist eine zusätzliche Motivation dadurch entstanden, dass die Studenten die Anwendung ihrer Fähigkeiten in einem kleinen Softwareprojekt gesehen haben.

6. Fazit und Ausblick

Der beschriebene Workshop umfasst die Anwendung der Grundlagen aus der Vorlesung „Einführung in die Programmierung“ und dem ersten Teil der Vorlesung „Datenstrukturen und Algorithmen“ in einer Android App. Bei den Teilnehmern entstand mehr Motivation für das weitere Studium und Verständnisprobleme konnten geklärt werden.

Der erste Durchgang des Workshops wurde von den Studenten positiv aufgenommen und evaluiert. Die 20 freiwilligen Teilnehmer beim ersten Durchgang konnten für die Inhalte begeistert werden und waren in der Lage die Aufgaben in angemessener Zeit zu lösen.

Der Workshop kann durch den begrenzten Umfang nur einen kleinen Beitrag zum Ziel der Fachgruppe Informatik der RWTH Aachen University beisteuern. Den Evaluationsergebnissen zufolge konnte bei den 20 Teilnehmern ein Erfolg verzeichnet werden. Die vier Thesen, die in Kapitel 3.3 aufgestellt wurden, konnten nicht widerlegt werden, soweit es das Untersuchungsdesign zulässt. Zum Teil konnten nur kleine Auswirkungen diagnostiziert werden, allerdings ist dies eine Folge der sehr kleinen Stichprobe.

In Zukunft ist es mit den erstellten Materialien ohne größeren Aufwand möglich, diesen Workshop zu wiederholen. Mit geringem personellen Aufwand kann ein längerfristiger Erfolg erzielt werden und den nächsten Studentengenerationen bei der Motivation für die Inhalte der Informatik geholfen werden. In Kombination mit weiteren ähnlichen Workshops zu anderen Themengebieten der Informatik kann auch ein größerer und längerfristiger Erfolg erzielt werden.
Ein Einsatz der Materialien bei begabten Schülern ist in der Zukunft eine weitere Option. Wenn die Bearbeitungszeit für das Arbeitsmaterial des Workshops in angemessener Form erweitert wird, können auch Schüler aus einem Leistungskurs oder Schüler aus einem Grundkurs mit guten bis sehr guten Noten den Workshop durchlaufen und gute Ergebnisse erzielen. Dies ist zum Beispiel als Modul im Schülerlabor Informatik InfoSphere der RWTH Aachen University möglich. Ein Einsatz der Materialien in einer Projektphase in der gymnasialen Oberstufe ist ebenfalls eine Option. Wenn mehr Zeit zur Verfügung steht, können die Android-Grundlagen gemeinsam mit den Schülern erarbeitet werden und zum Beispiel beginnend mit der Entwicklung der graphischen Oberfläche kann in das Projekt eingestiegen werden.
Insgesamt gibt es zahlreiche Möglichkeiten die für den Workshop entwickelten Materialien einzusetzen. Mit kleinen Modifizierungen bzw. didaktischen Reduktionen ist der Kreis der Interessenten deutlich erweiterbar. In der vorliegenden Form sind Studenten in der Studieneingangsphase und Oberstufenschüler mit guten Leistungen in der Informatik eine geeignete Zielgruppe.

Literaturverzeichnis

[Allis 1994] Allis, Louis V.: *Searching for solutions in games and artificial intelligence*. Wageningen : Ponsen & Looijen, 1994

[Allis u. a. 1993] Allis, Louis V. ; Herik, Hendrik J. van den ; Huntjens, Matty. P.: *Go-Moku and Threat-Space Search*. University of Limburg, Department of Computer Science, 1993

[Aschersleben 1977] Aschersleben, Karl: *Motivationsprobleme in der Schule*. Bd. 237. 1. Stuttgart, Berlin, Köln und Mainz : Kolhhammer, 1977

[Baumann 1983] Baumann, Rüdeger: *Computerspiele und Knobeleien programmiert in BASIC*. 2. Würzburg : Vogel, 1983

[Behrens und Rathgeb 2012] Behrens, Peter ; Rathgeb, Thomas: *JIM 2012 - Jugend, Information, (Multi-)Media*. Stuttgart : Medienpädagogischer Forschungsverbund Südwest, 2012

[Berliner 1988] Berliner, Hans J.: BKG - A Program That Plays Backgammon. In: Levy, David N. L. (Hrsg.): *Computer games I*. New York : Springer-Verlag, 1988, S. 3–28

[Bloom und Engelhart 1972] Bloom, Benjamin S. ; Engelhart, Max D.: *Taxonomie von Lernzielen im kognitiven Bereich*. Bd. 35. Weinheim [u.a.] : Beltz, 1972

[Breuer 2012] Breuer, Klaus: *Computerspiele programmieren: Künstliche Intelligenz für künstliche Gehirne*. München : Oldenbourg, 2012

[Cornelißen und Blanke 2004] Cornelissen, Waltraud ; Blanke,

Karen: Zeitverwendung von Mädchen und Jungen. In: Schmidt, Renate (Hrsg.): *Alltag in Deutschland*. Wiesbaden : Statistisches Bundesamt, 2004, S. 160–174

[Diekmann 2011] Diekmann, Andreas: *Empirische Sozialforschung*. 5. Hamburg : Rowohlt Taschenbuch Verlag, 2011

[Edelmann 2000] Edelmann, Walter: *Lernpsychologie*. 6. Weinheim : Beltz PVU, 2000

[Fässler 2007] Fässler, Lukas: *Das 4-Schritte-Modell: Grundlage für ein kompetenzorientiertes E-Learning von Lukas Emanuel Fässler*. Zürich : Eidgenössische Technische Hochschule Zürich, 2007

[Fischer 1973] Fischer, Margret: *Differenzierung im Schulunterricht*. Bd. 43. Weinheim und Basel : Beltz, 1973

[Geissler 2010] Geissler, Rainer: *Die Sozialstruktur Deutschlands: Zur gesellschaftlichen Entwicklung mit einer Bilanz zur Vereinigung*. Wiesbaden : Vs Verlag für Sozialwissenschaften, 2010

[Glaap 1993] Glaap, Dieter: Kreativ Arbeiten mit Computern. In: Ernst, Tilman (Hrsg.): *Computerspiele*. Bonn : Bundeszentrale für Politsche Bildung, 1993, S. 137–145

[Hartmann 2013] Hartmann, Guido M.: *Die RWTH will an die Weltspitze*. 2013. – URL `http://www.welt.de/117618505`. – Zugriffsdatum: 16.07.2013

[Helm 2011] Helm, Maren: *Digitale Lernwelt - Serious games*. Bielefeld : Bertelsmann, 2011

[Heublein u. a. 2012] Heublein, Ulrich ; Richter, Johanna ; Schmelzer, Robert ; Sommer, Dieter: *Die Entwicklung der Schwund und Studienabbruchquoten an den deutschen Hochschulen: Statistische Berechnungen auf der Basis des Absolventenjahrgangs 2010*. Hannover : Hochschul-Informations-System GmbH, 2012

[Holz 2012a] Holz, Jan: *IGaDtools4MINT: Integration von Gender and Diversity in MINT-Studiengängen an Hochschulen*. 2012. – URL `http://www.igadtools4mint.de/`. – Zugriffsdatum: 16.07.2013

[Holz 2012b] Holz, Jan: *Progra Workshops: Eigene Apps im ersten Semester*. 2012. – URL `http://www.igadtools4mint.de/PrograWorkshops`. – Zugriffsdatum: 24.06.2013

[Holz u. a. 2011] Holz, Jan ; Leonhardt, Thiemo ; Schroeder, Ulrik: Using smartphones to motivate secondary school students for informatics. In: Korhonen, Ari (Hrsg.): *Proceedings of the 11th Koli Calling International Conference on Computing Education Research*. New York : ACM, 2011, S. 84–94

[Hötte u. a. 2013] Hötte, Hans-Dieter ; Junge, Mario ; Fritz, Hubertine: *RWTH Aachen University Zahlenspiegel 2012*. Aachen : VD Vereinte Druckwerke GmbH, 2013

[Hoy und Schönpflug 2008] Hoy, Anita W. ; Schönpflug, Ute: *Pädagogische Psychologie*. 10. München und Boston [u.a.] : Pearson Studium, 2008

[Jürgen 1995] Jürgen, Fritz: *Warum Computerspiele faszinieren: Empirische Annäherungen an Nutzung und Wirkung von Bildschirmspielen*. Weinheim : Juventa Verlag, 1995

[Kammler 2007] Kammler, Arnd: *Der Einsatz eines Computer-Programms für geistig behinderte Schüler - eine empirische Studie*. München : GRIN Verlag GmbH, 2007

[Krieg 2013] Krieg, Aloys: *Wettbewerb exzellente Lehre: „Was wurde erreicht – wie geht es weiter?“*. 2013. – URL `http://www.rwth-aachen.de/cms/root/Die_RWTH/Profil/Lehre/Exzellente_Lehre/Talk_Lehre/~dssk/Talk_Lehre_2013/`. – Zugriffsdatum: 16.07.2013

[Kultusministerkonferenz 2003] Kultusministerkonferenz: *10 Thesen zur Bachelor- und Masterstruktur*

in Deutschland. 2003. – URL `http://www.kmk.org/fileadmin/veroeffentlichungen_beschluesse/2003/2003_06_12-10-Thesen-Bachelor-Master-in-D.pdf`. – Zugriffsdatum: 08.07.2013

[Künneth 2012] Künneth, Thomas: *Android 4: Apps entwickeln mit dem Android SDK*. 2. Bonn : Galileo Press, 2012

[Leopold 2009] Leopold, Claudia: *Lernstrategien und Textverstehen: Spontaner Einsatz und Förderung von Lernstrategien*. Bd. 76. Münster und München [u.a.] : Waxmann, 2009

[Levy 1988] Levy, David: Before the Jet Age. In: Levy, David N. L. (Hrsg.): *Computer games I*. New York : Springer-Verlag, 1988, S. 116–153

[Ludwig 2013] Ludwig, Luise: „Schlagt eure Tablets auf ...“ – Vom Hype zum Zukunftsmodell? In: Aufenanger, Stefan (Hrsg.) ; Schlieszeit, Jürgen (Hrsg.): *Computer + Unterricht*. Seelze : Friedrich Verlag, 2013, S. 24–26

[Luger 2002] Luger, George F.: *Künstliche Intelligenz*. 4. München : Pearson Studium, 2002

[Niegemann 2004] Niegemann, Helmut M.: *Kompendium E-Learning*. Berlin und New York : Springer, 2004

[Otto 2013] Otto, Carsten: *Vorlesung Programmierung WS 2012/13: Klausurergebnisse*. 2013. – URL `http://verify.rwth-aachen.de/programmierungWS12/`. – Zugriffsdatum: 08.07.2013

[Post 2012] Post, Uwe: *Android-Apps entwickeln: Für Programmiereinsteiger geeignet*. 2. Bonn : Galileo Press, 2012

[Rheinberg und Salisch 2008] Rheinberg, Falko ; Salisch, Maria v.: *Motivation*. Bd. 555. 7. Stuttgart : Kohlhammer, 2008

[RWTH Aachen University 2010] RWTH Aachen University: *Prüfungsordnung für den Bachelor-Studiengang Informatik der*

Rheinisch-Westfälischen Technischen Hochschule Aachen: BPO Informatik. 2010

[Schubert und Schwill 2011] Schubert, Sigrid ; Schwill, Andreas: *Didaktik der Informatik*. 2. Heidelberg und Berlin : Spektrum Akademischer Verlag, 2011

[Schulmeister 2009] Schulmeister, Rolf: *Grundlagen hypermedialer Lernsysteme: Theorie - Didaktik - Design*. 4. München und Wien : Oldenbourg, 2009

[Schürmann u. a. 1980] Schürmann, Hans ; Diem, Heike ; Nüscheler, Manfred: *So gewinnt man Mühle*. Bd. 39. Ravensburg : A. Maier, 1980

[Ullenboom 2012] Ullenboom, Christian: *Java ist auch eine Insel*. 10. Bonn : Galileo Press, 2012

[Wolf 2010] Wolf, Jürgen: *C++ von A bis Z: Das umfassende Handbuch*. 2., aktual. Aufl., 1., korr. Nachdr. Bonn : Galileo Press, 2010

[Wolmeringer 2008] Wolmeringer, Gottfried: *Coding for Fun: IT-Geschichte zum Nachprogrammieren - für Interessierte, Einsteiger und Profis*. 1. Bonn : Galileo, 2008

[Zahle 1988] Zahle, Torben: A Program of Master-Strength to Play Five-in-a-Row. In: Levy, David N. L. (Hrsg.): *Computer games II*. New York : Springer-Verlag, 1988, S. 325–338

A. Lerneinheit

A.1. Ankündigungsplakat

Abbildung A.1.: Ankündigungsplakat für den Workshop

A.2. Materialliste

Für die Durchführung des Workshops werden jeweils für zwei Studenten die folgenden Materialien benötigt:

- Go-Moku Spiel bestehend aus
  - Spielbrett mit acht mal acht Spielfeldzellen
  - 32 rote Spielsteine[12]
  - 32 blaue Spielsteine[12]
- Arbeitsmappe bestehend aus den Arbeitsblättern
  - Spielregeln
  - Entwicklungsumgebung „Eclipse“
  - Go-Moku Programmgerüst
  - Programmierung der Klasse GomokuBoard für das Spielbrett
  - Programmierung der Klasse FieldArray für die Spielstrategie
  - Spielstrategie
  - Der Wettkampf
- Ergänzung zu den Arbeitsblättern
  - Ergänzung zu Arbeitsauftrag 8

[12] Bei einem Spielbrett mit mehr Spielfeldzellen wird eine entsprechend größere Anzahl an Spielsteinen benötigt.

  - Ergänzung zu Arbeitsauftrag 9

- Computer bzw. Laptop mit folgender Software
  - Eclipse-Entwicklungsumgebung
  - Android-SDK
- Android-Tablet-Computer mit folgenden Voraussetzungen
  - Android in der Version 2.1 oder höher
  - Bildschirmdiagonale von mindestens 8 Zoll
- Musterlösung für die Tutoren

A.3. Zeitplan

Arbeitsauftrag	Optimaler Zeitplan		Zeitplan ohne Optimierung	
	Dauer in Min.	Anfang	Dauer in Min.	Anfang
Pre-Test	10	16:00	10	16:00
Arbeitsauftrag 1	15	16:10	15	16:10
Arbeitsauftrag 2	10	16:25	15	16:25
Arbeitsauftrag 3	15	16:35	20	16:40
Arbeitsauftrag 4	5	16:50	5	17:00
Arbeitsauftrag 5	10	16:55	15	17:05
Arbeitsauftrag 6	20	17:05	30	17:20
Arbeitsauftrag 7	30	17:25	35	17:50
Arbeitsauftrag 8	30	17:55	35	18:25
Arbeitsauftrag 9	35	18:25	40	19:00
Optionaler Arbeitsauftrag	40	19:00		
Arbeitsauftrag 10	10	19:40	10	19:40
Post-Test	10	19:50	10	19:50
Ende		20:00		20:00

Tabelle A.1.: Zeitplan für den Workshop

B. Arbeitsmaterialien

B.1. Arbeitsblätter

Im Folgenden sind die Arbeitsblätter des Workshops dargestellt. Auf der beiliegenden CD sind die Arbeitsblätter in einer Druckversion vorhanden.

B.1.1. Spielbrett

Abbildung B.1.: Arbeitsblatt: Spielbrett

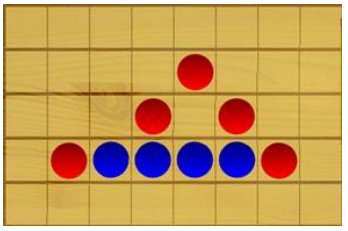
B.1.2. Spielregeln

Schülerlabor Informatik

Android-Workshop
GO-MOKU

InfoSphere
World of Informatics

Spielregeln



Spielmaterial

- Dame-Spielbrett
- 32 rote Spielsteine
- 32 blaue Spielsteine

Ziel des Spiels

Die Spieler haben das Ziel als erster Spieler eine Reihe mit fünf Steinen der eigenen Farbe zu bilden. Die Steine müssen in einer Linie liegen, die sich horizontal, vertikal oder diagonal auf dem Spielbrett befindet.

Spielregeln

Zu Beginn des Spiels erhält jeder Spieler 32 Spielsteine in einer der beiden verfügbaren Farben. Ein Spieler beginnt einen Spielstein zu setzen und anschließend wird abwechselnd jeweils ein Spielstein in der eigenen Farbe auf ein freies Feld des Spielfelds gelegt.

Jeder Spieler versucht fünf Steine nebeneinanderzulegen und den Gegner am Erreichen des Ziels zu hindern, indem der Bau einer Reihe durch blockierende Spielsteine verhindert wird. Längere Spielsteinreihen des Gegners stellen eine Gefahr dar, weil das Risiko auf fünf Steine steigt und die Gefahr einer Niederlage wächst.

Arbeitsauftrag 1: Der Weg zur eigenen Strategie

Spielen Sie zu zweit das Go-Moku Spiel und beantworten Sie währenddessen folgende Fragen:

- Welche Spielsituationen führen zu einem unvermeidbaren Sieg des Gegners? Zeichnen Sie mögliche Spielsituationen ein.

- Beschreiben Sie in Stichworten, welche Tricks Sie beim Spiel erkennen können?

- Welcher Bereich des Spielfelds ist bei freier Auswahl für das Setzen eines Steines auszuwählen?

1

Abbildung B.2.: Arbeitsblatt: Spielregeln

B.1.3. Entwicklungsumgebung „Eclipse“

Schülerlabor Informatik

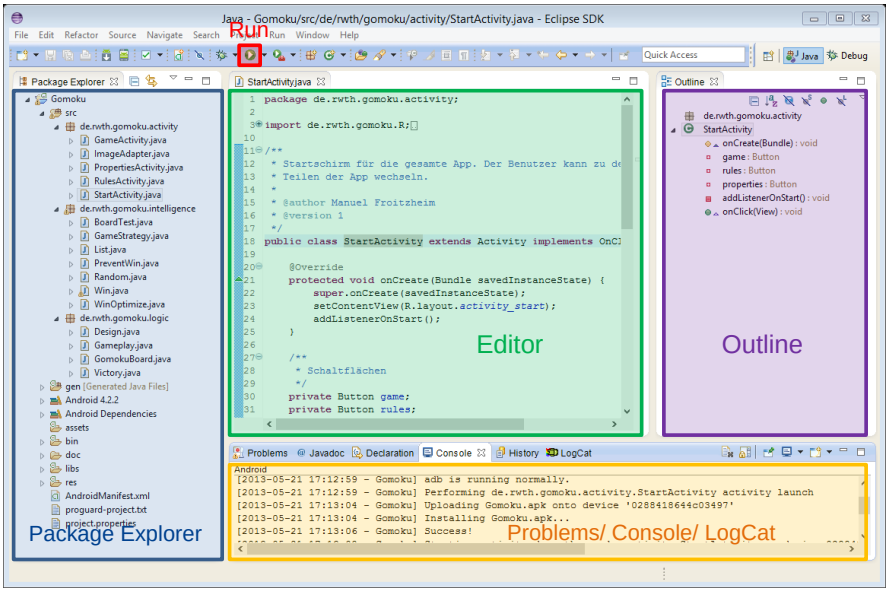
Android-Workshop
GO-MOKU

InfoSphere
World of Informatics

Entwicklungsumgebung „Eclipse“

Arbeitsauftrag 2: Entdecken Sie die Entwicklungsumgebung Eclipse

Die Entwicklungsumgebung „Eclipse“ ist eine Open-Source-Software, die für alle gebräuchlichen Betriebssysteme verfügbar ist. Die benötigten Komponenten zur JAVA- bzw. Android-Programmierung sind durch den modularen Aufbau leicht zu integrieren. Nach dem Öffnen der Anwendung ist für den Workshop der Workspace „gomoku“ zu wählen. Die Entwicklungsumgebung vereint den Editor, den Pakete-Explorer, die Konsole, Probleme, Protokoll Ausgaben und die Möglichkeit die App auf das Tablet zu übertragen.



2

Abbildung B.3.: Arbeitsblatt: Entwicklungsumgebung „Eclipse“ I

Schülerlabor Informatik

Android-Workshop
GO-MOKU

InfoSphere
World of Informatics

Package Explorer

Alle Ressourcen des Projektes werden im Pakete-Explorer in Baumstruktur dargestellt. Im Ordner „src“ (Source/ Quellcode) werden die Java-Pakete mit den enthaltenen Klassen dargestellt. Im Ordner „res“ (Ressourcen) werden die Dateien für das Layout der Android-App erstellt.

Editor

Der Editor ermöglicht die Bearbeitung des Quellcodes. Die Schlüsselwörter der Programmiersprache werden automatisch farblich hervorgehoben. Mit der Tastenkombination Strg + Leertaste schlägt Eclipse mögliche Vervollständigungen des Quellcodes vor.

Im Hintergrund wird der Quellcode in regelmäßigen Schritten kompiliert und fehlerhafter Programmcode rot unterstrichen und am Rand ein Fehlersymbol oder angezeigt. Details zu den Fehlern können im gelb umrandeten Bereich unter „Problems“ überprüft werden.

Problems/ Console/ LogCat

Im Bereich „Problems“ werden Syntax-Fehler des Programmcodes angezeigt und erläutert. Im Tabulator „Console“ werden die Ausgaben der klassischen Konsole angezeigt, wenn das Programm in der Konsole ausgeführt würde. Zum Beispiel Verbindungsfehler zum Android-Gerät oder „NullPointerException“. Im Bereich LogCat wird der Datenaustausch zwischen Eclipse und dem Android-Tablet dargestellt. Zur Überprüfung des Quellcodes kann das Paket „`android.util.Log`“ eingebunden werden und über die Quellcodezeile:

```
Log.v("DEBUG", "Wert für i ist " + i);
```

zum Beispiel der Wert für „i“ an der Stelle des Programmcodes zur Laufzeit ausgegeben werden.

Run

Ein Klick auf die Schaltfläche „Run“ führt das Programm auf einem per USB-Kabel angeschlossenen Android-Gerät oder dem Android-Emulator aus. Rückmeldungen des Programms werden direkt in der Eclipse Oberfläche im Fenster „LogCat“ angezeigt.

Outline

Im „Outline“ Bereich werden die Methoden der jeweiligen Klasse angezeigt.

3

Abbildung B.4.: Arbeitsblatt: Entwicklungsumgebung „Eclipse“ II

B.1.4. Go-Moku Programmgerüst

Schülerlabor Informatik

Android-Workshop
GO-MOKU

InfoSphere
World of Informatics

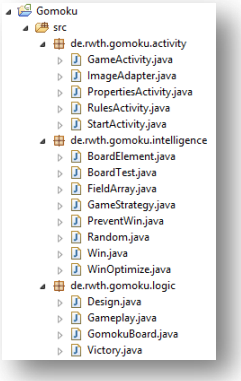
Go-Moku Programmgerüst

Im gesamten Projekt wurden die Methoden und Variablen, die während des Workshops implementiert werden sollen, entfernt. Damit das Projekt lauffähig ist, wurden leere Rückgabewerte (vgl. Abbildung) erzeugt. Diese sind bei der Implementierung der verschiedenen Methoden zu entfernen und durch den eigenen Quellcode zu ersetzen.

```
/*
 * Löschen und ersetzen durch den
 * eigenen Quellcode
 */
 return 0;

/*
 * Löschen und ersetzen durch den
 * eigenen Quellcode
 */
 return (int[][]) null;
```

Der Quellcode für das Go-Moku Spiel ist in drei Pakete aufgeteilt. Die Pakete sind für die Spiellogik, die Verwaltung/Darstellung des Spiels und die Spielstrategie verantwortlich.



activity

Das Paket „activity“ ist für die Kommunikation zwischen dem Logikteil des Quellcodes und der Benutzeroberfläche verantwortlich. Jede Benutzeroberfläche ist in einer Quellcodedatei repräsentiert und braucht im Workshop nicht verändert werden.

intelligence

In diesem Paket werden im Verlauf des Workshops verschiedene Spielstrategien programmiert. Die Schnittstelle (Interface) zu anderen Ressourcen der Software sind in der Klasse „GameStrategy“ definiert. Eine Strategie muss die enthaltenen Methoden

```
public String getName()
public BoardElement setToken(int[][] board)
```

implementieren. Die Methode `getName()` gibt den Namen der Strategie zur Identifizierung bei der Ausführung zurück. Die Methode `setToken(int[][] board)` bekommt das aktuelle Spielbrett übergeben und gibt eine Position für den folgenden Spielstein zurück. Die Berechnungen innerhalb der Methode können beliebig komplex sein.

Die Klasse „BoardTest“ ist eine Hilfestellung für die Überprüfung des Spielbrettes auf bestimmte Spielsteinkonstellation. Die Klasse „FIeldArray“ stellt eine klassische Liste dar und wird bei verschiedenen Strategien zur Verarbeitung der Spielsituationen benötigt.

logic

In der Klasse „GomokuBoard“ wird das Spielbrett statisch mit der Methode `newBoard()` definiert. Die Spielbrettgröße kann während der Laufzeit in den Einstellungen vom Benutzer verändert werden. Mit der Methode `setBoardCountRows(int countRowsNew)` wird ein neuer Wert übergeben. Es sind verschiedene Methoden zur Rückgabe der Werte in dieser Klasse zu implementieren.

Die Klassen „Design“, „Gameplay“ und „Victory“ sind für den Workshop nicht von Bedeutung.

Weitere Informationen zum Aufbau des Programmcodes finden Sie in der JavaDoc-Dokumentation unter:

http://subprogra.informatik.rwth-aachen.de/~mfroitzheim

4

Abbildung B.5.: Arbeitsblatt: Go-Moku Programmgerüst

B.1.5. Programmieren der Klasse GomokuBoard

Schülerlabor Informatik

Android-Workshop
GO-MOKU

InfoSphere
World of Informatics

Programmierung der Klasse `GomokuBoard` für das Spielbrett

In dieser Klasse des Pakets „logic“ wird das Spielbrett definiert und verwaltet. Für das Spielbrett wird ein zweidimensionales Array mit dem primitiven Datentyp „int“ als Datenstruktur gewählt. Der Inhalt des Feldes wird wie folgt repräsentiert:

- Leeres Feld: 0
- Feld mit einem Stein von Spieler A: 1
- Feld mit einem Stein von Spieler B: 2

Eine Variable für die Anzahl der Zeilen bzw. Spalten des Spielbrettes wird benötigt, weil dieser Wert vor der Erzeugung des Spielbrettes festgelegt wird. Die Initialisierung soll standardmäßig auf acht Einheiten eingestellt werden. Das Spielbrett ist quadratisch, deswegen ist die Anzahl Spalten immer gleich der Anzahl Zeilen. Im Quellcode ist aus diesem Grund keine Unterscheidung zwischen diesen Werten notwendig.

Aus Gründen der Datenkapselung ist die Sichtbarkeit der Variablen auf das Paket (protected) zu beschränken. Ein Zugriff aus anderen Paketen sollte nur über entsprechende Methoden geschehen dürfen.

Arbeitsauftrag 3:

Implementieren Sie die beschriebenen Methoden in der Klasse `GomokuBoard`. Legen Sie zunächst eine statische Variable für die Zeilenanzahl an und initialisieren Sie diese Variable auf den Wert 8.

a) `public static void setBoardCountRows(int countRowsNew)`

b) `public static int getBoardsCountRows()`

Die beiden Methoden stellen klassische „Getter-“ bzw. „Setter-“ Methoden für die Variable zur Speicherung der Anzahl an Zeilen des Spielfelds dar. Die minimale Größe für das Spielbrett sind genau acht Zeilen und die maximale Anzahl ist 19 Zeilen.

c) `public static void newBoard()`

Die Methode generiert ein neues Spielbrett bzw. leert das vorhandene Spielbrett für weitere Spielrunden. Die Größe des Spielbrettes wird anhand der Variablen für die Anzahl Zeilen/Spalten festgelegt.

Hinweis: Arrays mit einem primitiven Datentyp werden standardmäßig mit 0 initialisiert.

d) `public static int[][] getBoard()`

Die Methode gibt bei einem Aufruf das aktuelle Spielbrett zurück.

5

Abbildung B.6.: Arbeitsblatt: Programmierung der Klasse GomkuBoard für das Spielbrett

Schülerlabor Informatik

Android-Workshop
GO-MOKU

InfoSphere
World of Informatics

e) `public static int getFieldContent(int row, int coulmn)`

Wenn im Programmcode der Inhalt eines bestimmten Feldes benötigt wird, kann diese Methode verwendet werden. Es wird der Inhalt der entsprechenden Stelle zurückgegeben.

f) `public static boolean freeField()`

Diese Methode überprüft das gesamte Spielbrett auf freie Spielfelder. Es wird der Wert wahr (true) zurückgegeben, wenn ein freies Feld gefunden wurde. Ansonsten wird der Wert falsch (false) zurückgegeben.

Arbeitsauftrag 4: Das Tablet-Spiel kann so viel wie die analoge Variante

Überprüfen Sie Ihre Implementierung, indem Sie das Spiel im „Mensch“ gegen „Mensch“ Modus ausprobieren. In den Einstellungen zur App können Sie den Modus umstellen.

6

Abbildung B.7.: Arbeitsblatt: Programmierung der Klasse Gomku-Board für das Spielbrett

B.1.6. Programmieren der Klasse FieldArray für die Spielstrategie

Schülerlabor Informatik

Android-Workshop
GO-MOKU

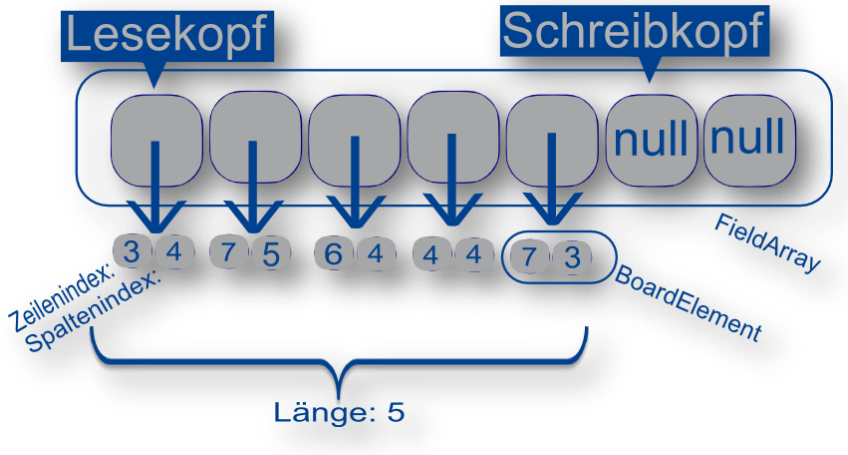
InfoSphere
World of Informatics

Programmierung der Klasse `FieldArray` für die Spielstrategie

Bei der Entwicklung der Spielstrategie wird eine erweiterte Arraystruktur zur Speicherung von Spielpositionen des Spielbrettes benötigt. Diese Positionen werden durch einen Zeilen- und einen Spaltenindex repräsentiert.

In diesem Array werden maximal so viele Elemente gespeichert, wie Felder auf dem Spielbrett existieren. Zur internen Repräsentation wird ein Board Element bestehend aus zwei „int“-Variablen für den Zeilen- und den Spaltenindex gewählt.

Für den Zugriff auf das Array werden ein Schreibkopf und ein Lesekopf benötigt. Diese können mit einer einfachen Indexvariablen repräsentiert werden.



Arbeitsauftrag 5: Auf dem Weg zur Computerstrategie fehlt noch eine Spielsteinverwaltung

Implementieren Sie die beschriebenen Methoden in der Klasse `BoardElement` des Paketes „intelligence“.

a)
```
private int row;
private int column;
public int getRow()
public void setRow(int row)
public int getColumn()
public void setColumn(int column)
```

DOKU

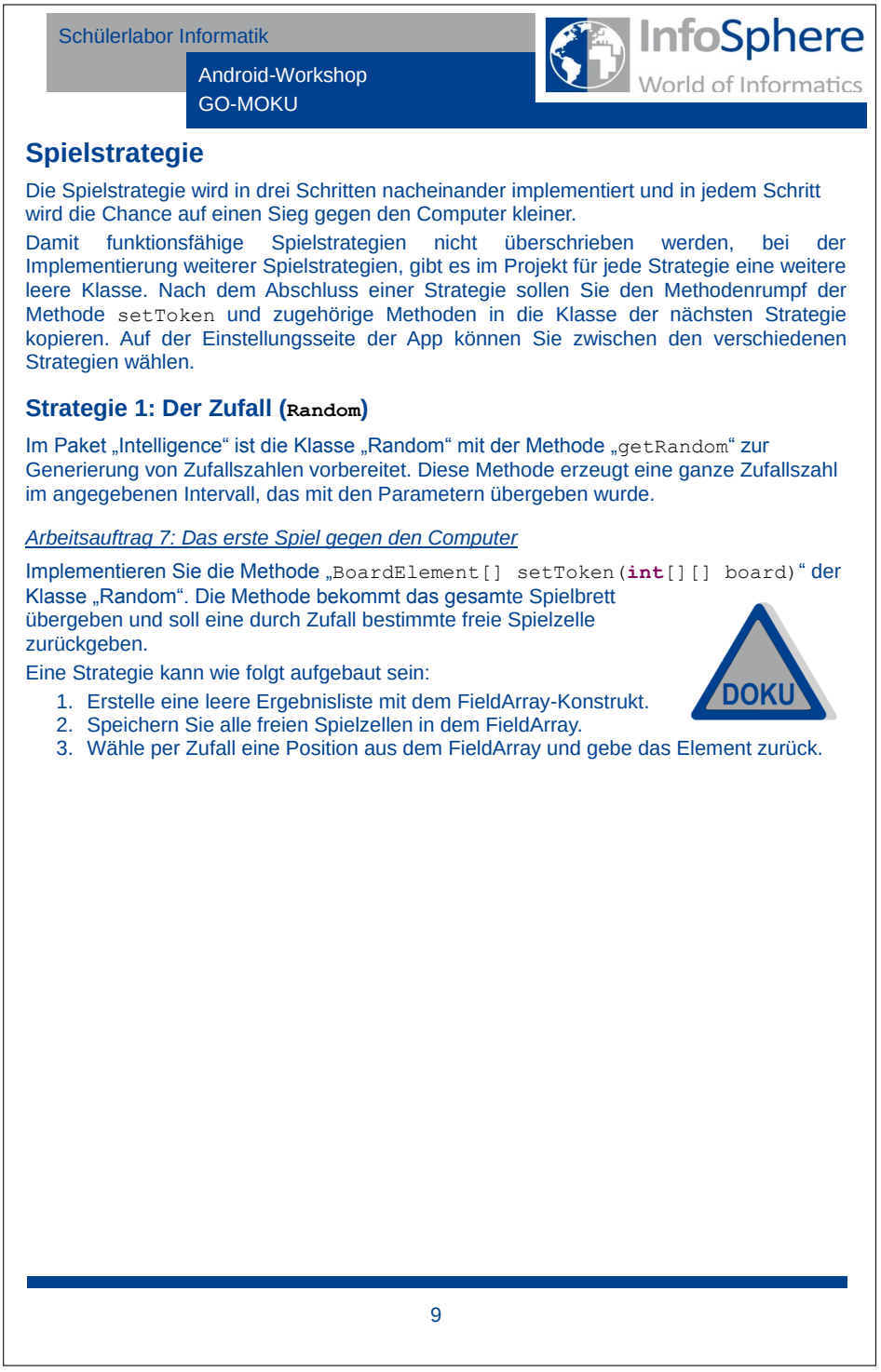
Die Variablen repräsentieren in der Klasse `BoardElement` den Zeilen- und den Spaltenindex. Der Zugriff auf die beiden Variablen soll über Getter- und Setter-Methoden erfolgen.

7

Abbildung B.8.: Arbeitsblatt: Programmierung der Klasse FieldArray für die Spielstrategie I

Abbildung B.9.: Arbeitsblatt: Programmierung der Klasse FieldArray für die Spielstrategie II

B.1.7. Spielstrategie



Schülerlabor Informatik

Android-Workshop
GO-MOKU

InfoSphere
World of Informatics

Spielstrategie

Die Spielstrategie wird in drei Schritten nacheinander implementiert und in jedem Schritt wird die Chance auf einen Sieg gegen den Computer kleiner.

Damit funktionsfähige Spielstrategien nicht überschrieben werden, bei der Implementierung weiterer Spielstrategien, gibt es im Projekt für jede Strategie eine weitere leere Klasse. Nach dem Abschluss einer Strategie sollen Sie den Methodenrumpf der Methode `setToken` und zugehörige Methoden in die Klasse der nächsten Strategie kopieren. Auf der Einstellungsseite der App können Sie zwischen den verschiedenen Strategien wählen.

Strategie 1: Der Zufall (`Random`)

Im Paket „Intelligence“ ist die Klasse „Random“ mit der Methode „`getRandom`“ zur Generierung von Zufallszahlen vorbereitet. Diese Methode erzeugt eine ganze Zufallszahl im angegebenen Intervall, das mit den Parametern übergeben wurde.

Arbeitsauftrag 7: Das erste Spiel gegen den Computer

Implementieren Sie die Methode „`BoardElement[] setToken(int[][] board)`“ der Klasse „Random“. Die Methode bekommt das gesamte Spielbrett übergeben und soll eine durch Zufall bestimmte freie Spielzelle zurückgeben.

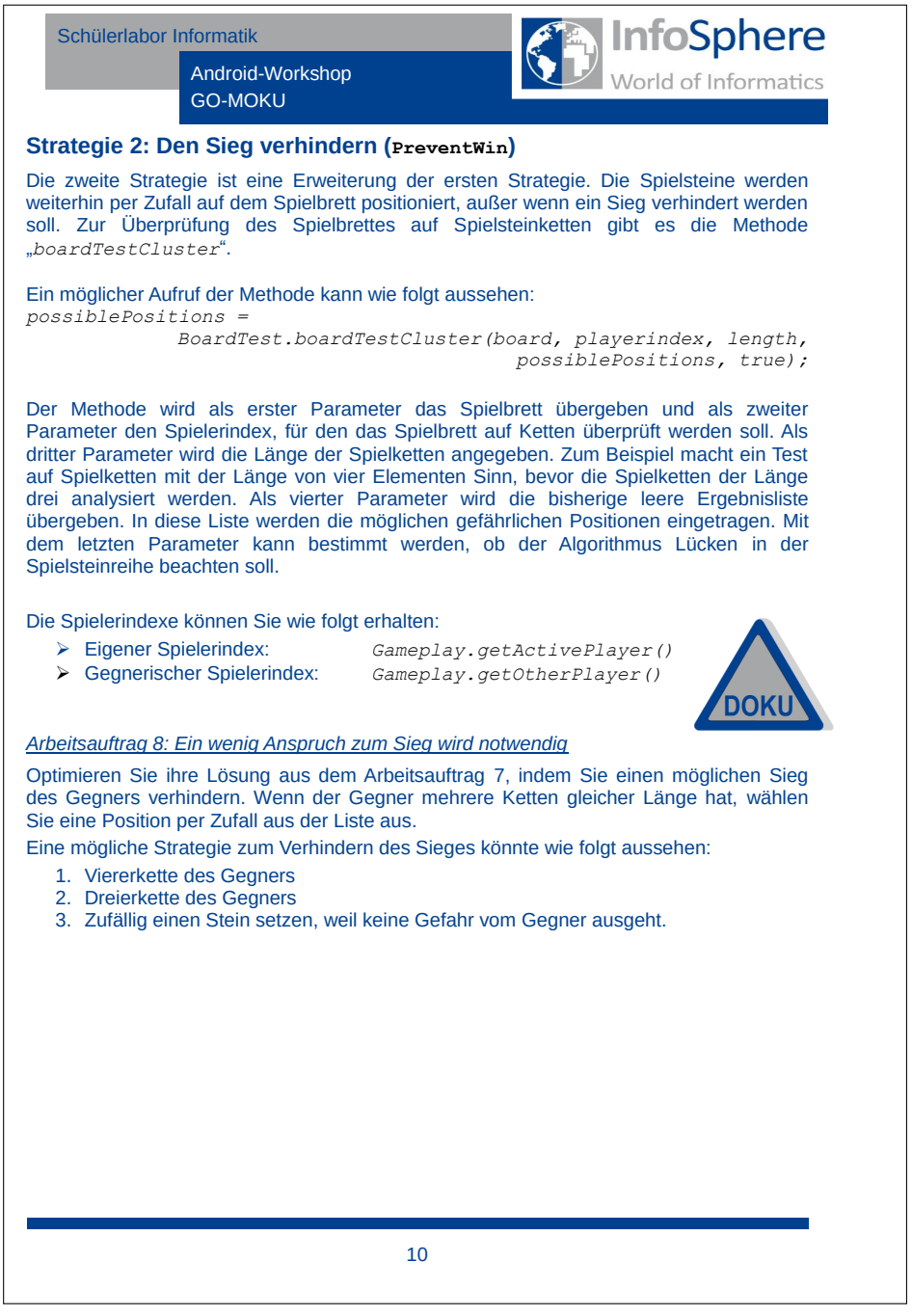
DOKU

Eine Strategie kann wie folgt aufgebaut sein:

1. Erstelle eine leere Ergebnisliste mit dem FieldArray-Konstrukt.
2. Speichern Sie alle freien Spielzellen in dem FieldArray.
3. Wähle per Zufall eine Position aus dem FieldArray und gebe das Element zurück.

9

Abbildung B.10.: Arbeitsblatt: Spielstrategie I



Strategie 2: Den Sieg verhindern (`PreventWin`)

Die zweite Strategie ist eine Erweiterung der ersten Strategie. Die Spielsteine werden weiterhin per Zufall auf dem Spielbrett positioniert, außer wenn ein Sieg verhindert werden soll. Zur Überprüfung des Spielbrettes auf Spielsteinketten gibt es die Methode „*boardTestCluster*“.

Ein möglicher Aufruf der Methode kann wie folgt aussehen:

```
possiblePositions =
            BoardTest.boardTestCluster(board, playerindex, length,
                                                possiblePositions, true);
```

Der Methode wird als erster Parameter das Spielbrett übergeben und als zweiter Parameter den Spielerindex, für den das Spielbrett auf Ketten überprüft werden soll. Als dritter Parameter wird die Länge der Spielketten angegeben. Zum Beispiel macht ein Test auf Spielketten mit der Länge von vier Elementen Sinn, bevor die Spielketten der Länge drei analysiert werden. Als vierter Parameter wird die bisherige leere Ergebnisliste übergeben. In diese Liste werden die möglichen gefährlichen Positionen eingetragen. Mit dem letzten Parameter kann bestimmt werden, ob der Algorithmus Lücken in der Spielsteinreihe beachten soll.

Die Spielerindexe können Sie wie folgt erhalten:

- Eigener Spielerindex: `Gameplay.getActivePlayer()`
- Gegnerischer Spielerindex: `Gameplay.getOtherPlayer()`

DOKU

Arbeitsauftrag 8: Ein wenig Anspruch zum Sieg wird notwendig

Optimieren Sie ihre Lösung aus dem Arbeitsauftrag 7, indem Sie einen möglichen Sieg des Gegners verhindern. Wenn der Gegner mehrere Ketten gleicher Länge hat, wählen Sie eine Position per Zufall aus der Liste aus.

Eine mögliche Strategie zum Verhindern des Sieges könnte wie folgt aussehen:

1. Viererkette des Gegners
2. Dreierkette des Gegners
3. Zufällig einen Stein setzen, weil keine Gefahr vom Gegner ausgeht.

Abbildung B.11.: Arbeitsblatt: Spielstrategie II

Strategie 3: Siegen (`Win`)

Das erste Ziel bleibt die Verhinderung eines Sieges des Gegners. Wenn kein Sieg zu verhindern ist, sollten Sie möglichst viele zusammenhängende Ketten erzeugen. Mithilfe des Min-Max-Algorithmus können die einzelnen Positionen auf dem Spielbrett analysiert werden. Eine vereinfachte Form des Min-Max-Algorithmus ohne Rekursion kann auch auf einem leistungsschwachen Tablet in akzeptabler Zeit ausgeführt werden. Wenn mehrere mögliche Positionen vorhanden sind, können die Werte mit dem Min-Max-Algorithmus bewertet werden.

Vereinfachter Min-Max-Algorithmus

Die Methode `minMax` erhält das Spielbrett, eine Liste der möglichen Positionen, den Spielerindex und einen boolean als Anweisung, ob ein Sieg verhindert oder die eigene Lage verbessert werden soll. Bei der Verhinderung des Sieges des Gegners muss der Wert minimiert werden und im anderen Fall maximiert.

Die möglichen Positionen müssen nacheinander durchlaufen werden, indem jeweils zu Bewertungszwecken ein Stein des Spielers auf das Feld gelegt wird. Anschließend kann das gesamte Spielbrett mit der Methode `evaluatePosition(board, player)` bewertet werden. Wenn die Bewertung besser als die vorherigen Bewertungen ist, muss die Position zwischengespeichert werden. Der zwischenzeitlich abgelegte Spielstein muss wieder vom Spielbrett entfernt werden. Wenn alle möglichen Positionen überprüft wurden, ist die Position mit der besten Bewertung zurückzugeben. Es ist zu beachten ob der Optimierungsalgorithmus zur Minimierung oder Maximierung aufgerufen wurde.

Arbeitsauftrag 9: Der Computer möchte auch mal gewinnen

Implementieren Sie den vereinfachten rekursionsfreien Min-Max-Algorithmus zur Verbesserung ihrer Spielstrategie.

Optionaler Arbeitsauftrag:

Optimieren Sie ihre Strategie, indem Sie eigenen Ideen einarbeiten in den Programmcode.

Abbildung B.12.: Arbeitsblatt: Spielstrategie III

B.1.8. Der Wettkampf

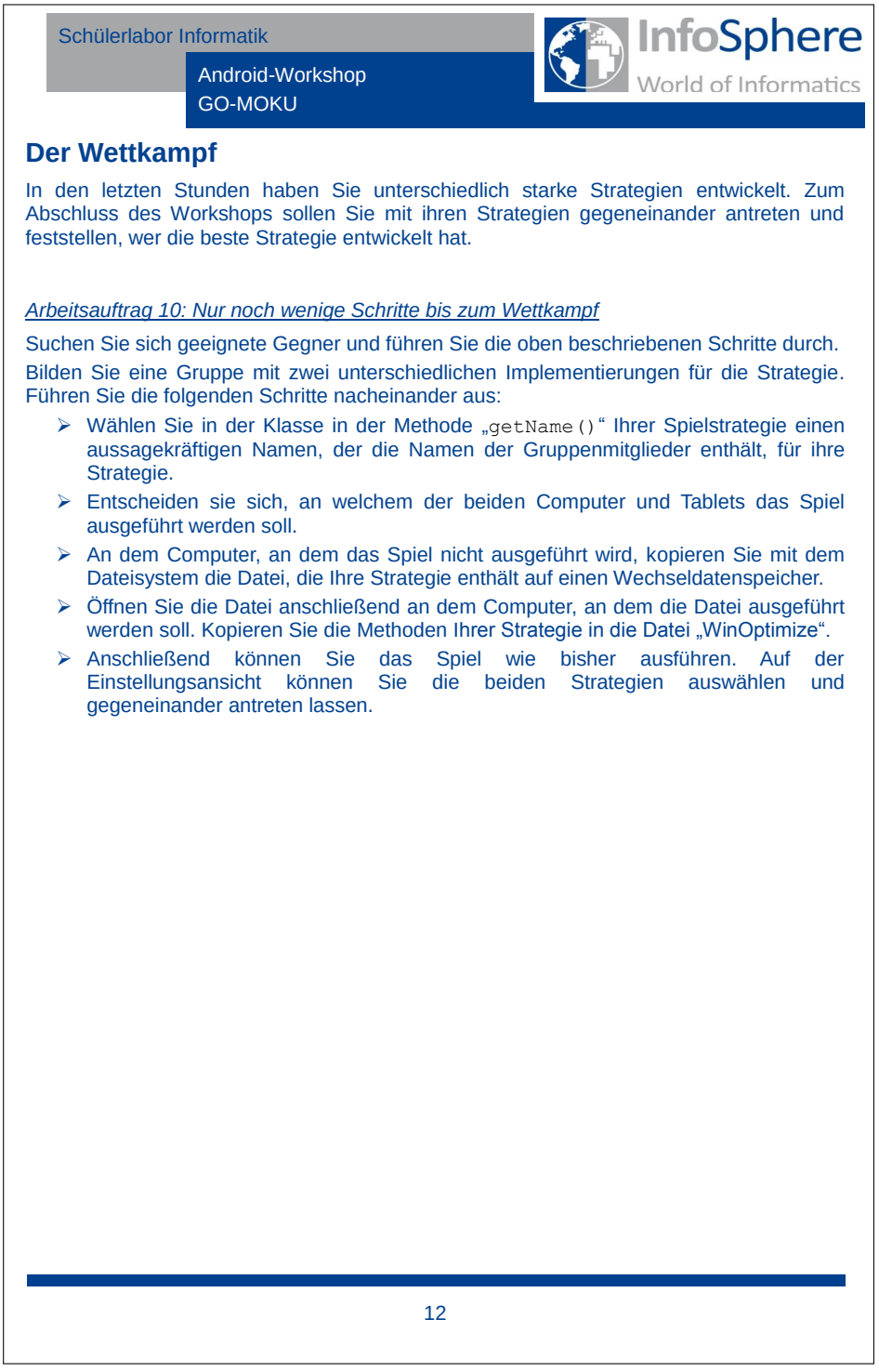


Abbildung B.13.: Arbeitsblatt: Der Wettkampf

B.2. Ergänzungen

B.2.1. Ergänzung zu Arbeitsauftrag 8

Schülerlabor Informatik

Android-Workshop
GO-MOKU

InfoSphere
World of Informatics

Ergänzung zu Arbeitsauftrag 8

Die Strategie besteht im Wesentlichen aus drei Schritten. Die drei Schritte können nacheinander ausgeführt werden und nach jedem Schritt wird gegebenenfalls eine gefundene Spielbrettposition ausgegeben. Die anderen Schritte brauchen anschließend nicht ausgeführt werden.

1. Viererkette des Gegners

Die Überprüfung auf eine Viererkette wird durch die Methode *boardTestCluster* des Pakets *BoardTest* übernommen. Der folgende Aufruf fügt alle möglichen Positionen in die bisherige Ergebnisliste ein:

```
possiblePositions =
BoardTest.boardTestCluster(board,     Gameplay.getActivePlayer(),
                                    4, possiblePositions,true);
```

Nach dem Aufruf der Methode kann die Länge der Liste überprüft werden. Wenn ein oder mehrere Elemente in der Liste enthalten sind, dann kann ein beliebiges Element ausgegeben werden. Weitere Schritte sind in diesem Fall nicht notwendig.

2. Dreierkette des Gegners

Die Überprüfung auf eine Dreierkette erfolgt simultan wie die Überprüfung auf eine Viererkette. Lediglich die „4“ wird durch eine „3“ ersetzt.

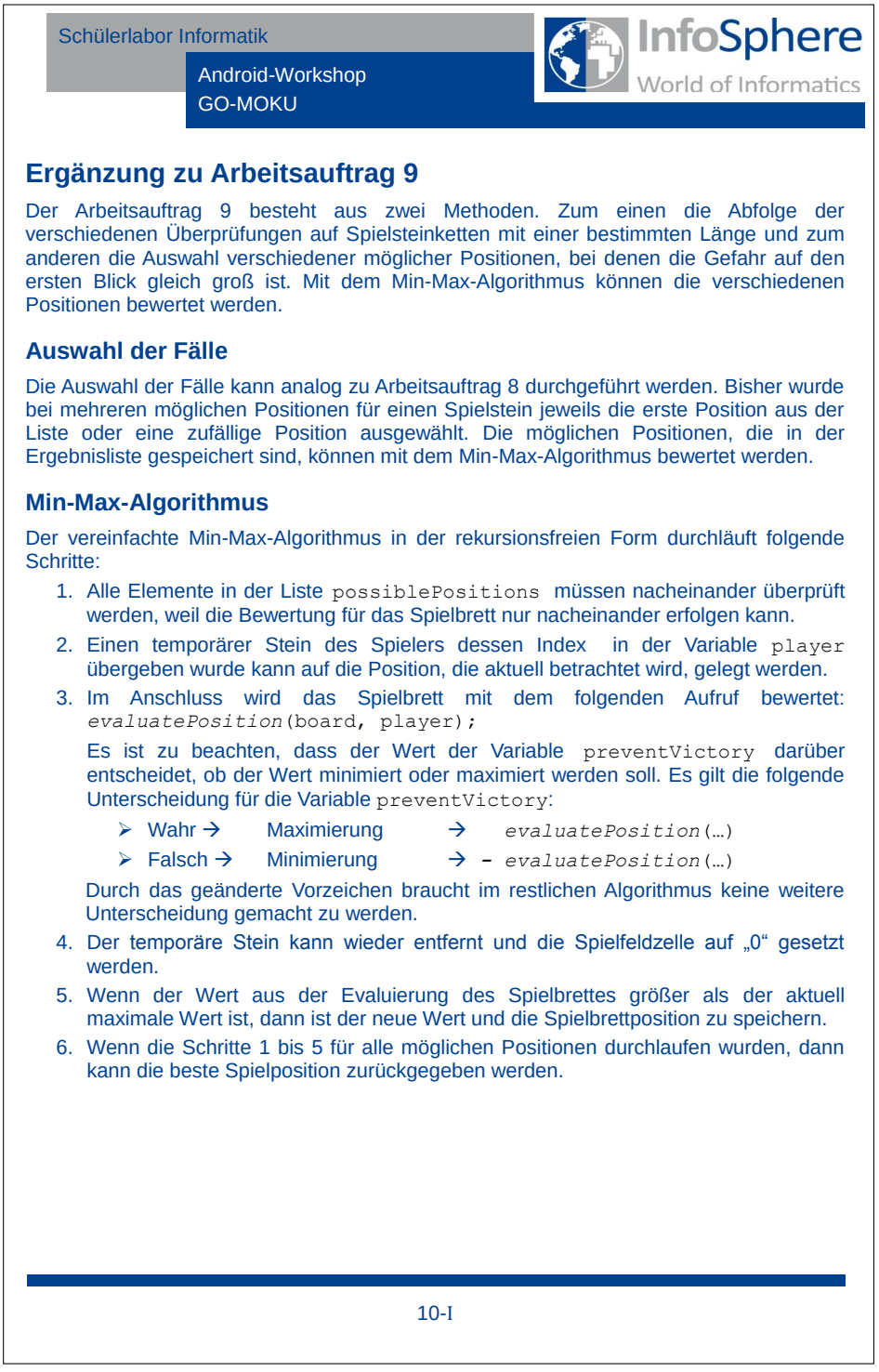
3. Zufällig einen Stein setzen, weil keine Gefahr vom Gegner ausgeht

Die an dieser Stelle notwendige Arbeit wurde bereits in Arbeitsauftrag 7 erledigt. Den Quellcode können Sie ohne weitere Modifizierungen übernehmen.

9-I

Abbildung B.14.: Arbeitsblatt: Ergänzung zu Arbeitsauftrag 8

B.2.2. Ergänzung zu Arbeitsauftrag 9



Schülerlabor Informatik

Android-Workshop
GO-MOKU

InfoSphere
World of Informatics

Ergänzung zu Arbeitsauftrag 9

Der Arbeitsauftrag 9 besteht aus zwei Methoden. Zum einen die Abfolge der verschiedenen Überprüfungen auf Spielsteinketten mit einer bestimmten Länge und zum anderen die Auswahl verschiedener möglicher Positionen, bei denen die Gefahr auf den ersten Blick gleich groß ist. Mit dem Min-Max-Algorithmus können die verschiedenen Positionen bewertet werden.

Auswahl der Fälle

Die Auswahl der Fälle kann analog zu Arbeitsauftrag 8 durchgeführt werden. Bisher wurde bei mehreren möglichen Positionen für einen Spielstein jeweils die erste Position aus der Liste oder eine zufällige Position ausgewählt. Die möglichen Positionen, die in der Ergebnisliste gespeichert sind, können mit dem Min-Max-Algorithmus bewertet werden.

Min-Max-Algorithmus

Der vereinfachte Min-Max-Algorithmus in der rekursionsfreien Form durchläuft folgende Schritte:

1. Alle Elemente in der Liste `possiblePositions` müssen nacheinander überprüft werden, weil die Bewertung für das Spielbrett nur nacheinander erfolgen kann.
2. Einen temporärer Stein des Spielers dessen Index in der Variable `player` übergeben wurde kann auf die Position, die aktuell betrachtet wird, gelegt werden.
3. Im Anschluss wird das Spielbrett mit dem folgenden Aufruf bewertet: `evaluatePosition(board, player);`

   Es ist zu beachten, dass der Wert der Variable `preventVictory` darüber entscheidet, ob der Wert minimiert oder maximiert werden soll. Es gilt die folgende Unterscheidung für die Variable `preventVictory`:

   - Wahr → Maximierung → `evaluatePosition(…)`
   - Falsch → Minimierung → `- evaluatePosition(…)`

   Durch das geänderte Vorzeichen braucht im restlichen Algorithmus keine weitere Unterscheidung gemacht zu werden.
4. Der temporäre Stein kann wieder entfernt und die Spielfeldzelle auf „0“ gesetzt werden.
5. Wenn der Wert aus der Evaluierung des Spielbrettes größer als der aktuell maximale Wert ist, dann ist der neue Wert und die Spielbrettposition zu speichern.
6. Wenn die Schritte 1 bis 5 für alle möglichen Positionen durchlaufen wurden, dann kann die beste Spielposition zurückgegeben werden.

10-I

Abbildung B.15.: Arbeitsblatt: Ergänzung zu Arbeitsauftrag 9

B.3. Musterlösungen

Schülerlabor Informatik

Android-Workshop
GO-MOKU - Musterlösung

InfoSphere
World of Informatics

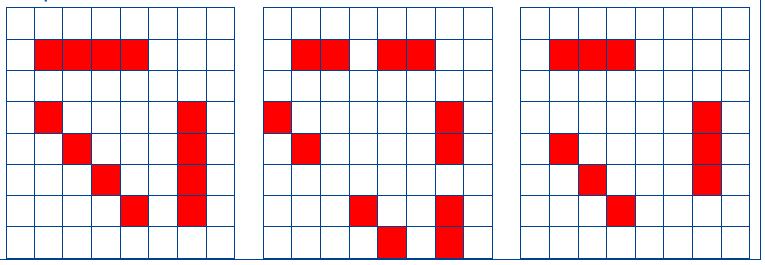
Musterlösung

Die Musterlösung enthält mögliche Lösungen, denn eine einzig wahre Lösung gibt es bei den Aufgaben nicht.

Arbeitsauftrag 1:

Spielen Sie zu zweit das Go-Moku Spiel und beantworten Sie währenddessen folgende Fragen:

- Welche Spielsituation führt zu einem unvermeidbaren Sieg des Gegners?
  - Vier Steine mit einer benachbarten Lücke führen zu einem Sieg.
  - Drei Steine mit Lücken auf den beiden Seiten führen ebenfalls zu einem Sieg.
  - Beispiel:



- Welche Tricks können Sie beim Spiel erkennen?
  - Steingruppen mit mehreren Möglichkeiten zur Bildung von größeren Spielsteingruppen sind von Vorteil, weil der Gegner mit der Verhinderung des Ausbaus an der eigenen Planung gehindert werden kann.
- Welcher Bereich des Spielfelds ist bei freier Auswahl für das Setzen eines Steins auszuwählen?
  - Von der Mitte des Spielfelds ausgehend, bestehen mehr Möglichkeiten weitere Steine anzulegen. Deswegen sollten Positionen in der Mitte des Spielbrettes bevorzugt werden.

Arbeitsauftrag 2:

Entdecken Sie die Entwicklungsumgebung Eclipse

Individuelle Lösung.

1

Abbildung B.16.: Arbeitsblatt: Musterlösung I

Schülerlabor Informatik

Android-Workshop
GO-MOKU - Musterlösung

InfoSphere
World of Informatics

Arbeitsauftrag 3:

Implementieren Sie die beschriebenen Methoden in der Klasse `GomokuBoard`.

```
protected static int[][] board;
protected static int countRows = 8;
```

a)
```
public static void setBoardCountRows(int countRowsNew)
if (countRowsNew > 7 && countRowsNew < 20) {
    countRows = countRowsNew;
}
```

b)
```
public static int getBoardsCountRows()
return countRows;
```

c)
```
public static void newBoard()
board = new int[countRows][countRows];
```

d)
```
public static int[][] getBoard()
return board;
```

e)
```
public static int getFieldContent(int row, int column)
    return board[row][column];
```

f)
```
public static boolean freeField()
for (int i = 0; i < board.length; i++) {
      for (int j = 0; j < board.length; j++) {
          if (board[i][j] == 0) {
               return true;
          }
      }
}
return false;
```

Der Zugriff auf die Variable „*countRows*“ zur Bestimmung der Größe des Spielfelds ist ebenfalls möglich.

Arbeitsauftrag 4:

Überprüfen Sie Ihre Implementierung, indem Sie das Spiel im „Mensch“ gegen „Mensch“ Modus ausprobieren. In den Einstellungen zur App können Sie den Modus umstellen.

Individuelle Lösung.

2

Abbildung B.17.: Arbeitsblatt: Musterlösung II

Schülerlabor Informatik

Android-Workshop
GO-MOKU - Musterlösung

InfoSphere
World of Informatics

Arbeitsauftrag 5:

Implementieren Sie die beschriebenen Methoden in der Klasse `BoardElement` des Paketes „intelligence“.

a)

```
private int row;
private int column;


public int getRow() {
	return row;
}

public void setRow(int row) {
	this.row = row;
}

public int getColumn() {
	return column;
}

public void setColumn(int column) {
	this.column = column;
}
```

b) `BoardElement(int newRow, int newColumn){`

```
	row = newRow;
	column = newColumn;
```

Arbeitsauftrag 6:

Implementieren Sie die beschriebenen Methoden in der Klasse `FieldArray`.

```
BoardElement[] fieldArray;
int write;
int read;
```

a) `public FieldArray(int maxNumberElements)`

```
fieldArray = new BoardElement[maxNumberElements];
```

b) `public int count()`

```
return write - read;
```

3

Abbildung B.18.: Arbeitsblatt: Musterlösung III

Schülerlabor Informatik

Android-Workshop
GO-MOKU - Musterlösung

InfoSphere
World of Informatics

c) `public void addElement(int row, int column)`

```
fieldArray[write] = new BoardElement(row, column);
write++;
```

d) `public BoardElement getElement()`

```
if (read <= write) {
      return fieldArray[read];
} else {
      return null;
}
```

e) `public BoardElement getElement(int index)`

```
if (read + index <= write) {
      return fieldArray[read + index];
} else {
      return null;
}
```

f) `public void removeElement()`

```
read++;
```

Arbeitsauftrag 7:

Implementieren Sie die Methode „`BoardElement setToken(int[][] board)`“ der Klasse „Random“. Die Methode bekommt das gesamte Spielbrett übergeben und soll ein durch Zufall bestimmtes freies Spielfeld zurückgeben.

```
public BoardElement setToken(int[][] board) {
      return getRandomToken(board);
}

private BoardElement getRandomToken(int[][] board) {
// Erzeuge eine leere Liste mit Platz für alle Elemente des Spielfelds
      FieldArray possiblePositions = new FieldArray(board.length * board.length);

      // Spielbrett Zeilen- und Spaltenweise durchlaufen
      for (int j = 0; j < board.length; j++) {
            for (int i = 0; i < board.length; i++) {
                  if (board[j][i] == 0) { // Feld leer?
                      possiblePositions.addElement(j, i);
                  }
            }
      }
      // Wähle per Zufall ein Element aus der Liste und gebe es zurück.
      return possiblePositions.getElement(getRandom(0,
                                          possiblePositions.count()));
}
```

4

Abbildung B.19.: Arbeitsblatt: Musterlösung IV

Schülerlabor Informatik

Android-Workshop
GO-MOKU - Musterlösung

InfoSphere
World of Informatics

Arbeitsauftrag 8:

Optimieren Sie ihre Lösung aus dem Arbeitsauftrag 4, indem Sie einen möglichen Sieg des Gegners verhindern. Wenn der Gegner mehrere Ketten gleicher Länge hat, wählen Sie eine Position per Zufall aus der Liste aus.

```
public BoardElement setToken(int[][] board) {
      FieldArray possiblePositions = new FieldArray(board.length *
                                                          board.length);

      // Prüfe auf vier und drei Steine in einer Reihe
      int j = 4;
      while (possiblePositions.count() == 0 && j > 2) {
            possiblePositions = BoardTest.boardTestCluster(board,
            Gameplay.getOtherPlayer(), j, possiblePositions, true);
            j--;
      }

      // Setze eine Zufallszahl, weil kein Sieg zu verhindern ist
      switch (possiblePositions.count()) {
      case 0:
            // Position per Zufall auswählen
            return getRandomToken(board);
      case 1:
      // Liste hat nur ein Element, deswegen wird die Position gewählt.
            return possiblePositions.getElement();
      default:
      // Liste hat mehrere Elemente. Es wird per Zufall ein Element
                                                      // ausgewählt.
            return possiblePositions.getElement(getRandom(0,
                                        possiblePositions.count()));
      }
}
```

Arbeitsauftrag 9:

Implementieren Sie den vereinfachten rekursionsfreien Min-Max-Algorithmus zur Verbesserung ihrer Spielstrategie.

```
public BoardElement setToken(int[][] board) {
      FieldArray possiblePositions = new FieldArray(board.length *
                                                        board.length);
      // Eigene Möglichkeit zum direkten Gewinn überprüfen
      boolean preventVictory;
      possiblePositions = BoardTest.boardTestCluster(board,
            Gameplay.getActivePlayer(), 3, possiblePositions, true);
      preventVictory = false;
      // Prüfe auf vier und drei Steine in einer Reihe
      int j = 4;
      while (possiblePositions.count() == 0 && j > 2) {
            possiblePositions = BoardTest.boardTestCluster(board,
            Gameplay.getOtherPlayer(), j, possiblePositions, true);
            j--;
            preventVictory = true;
      }
```

5

Abbildung B.20.: Arbeitsblatt: Musterlösung V

C. Quelltext der Android App

Der Quellcode der gesamten App kann auf der beiliegenden CD eingesehen werden.

D. Spielauswahl

7x7

Existiert eine Android App? Ja

Motivierend: Das Spiel existiert erst seit kurzem auf dem Markt. Die App wurde in letzter Zeit oft heruntergeladen und die Spielidee von den Nutzern positiv bewertet.

Spielidee: Das Spielprinzip von 7x7 ist eine Mischung aus Tetris und vier Gewinnt. Die Bildung von Reihen mit Steinen in der gleichen Farbe ist das Ziel des Spiels. Der Spieler kann in jeder Runde Spielsteine verschieben und sieht von welcher Farbe automatisch in der nächsten Spielrunde Steine auf dem Spielbrett an einer zufälligen Stelle hinzukommen.

Künstliche Intelligenz: In dem Spiel gibt es keine künstliche Intelligenz, weil der Computer die neuen Spielsteine willkürlich anordnet.

Datenstrukturen: Ein mehrdimensionales Array ist für das Spielbrett enthalten.

Erweiterbares Spielbrett: Das Spielbrett kann beliebig vergrößert und verkleinert werden.

Direktes Ausschlusskriterium: Der Zufall spielt eine sehr große Rolle und es wird als Datenstruktur nur ein Array verwandt.

Backgammon

Existiert eine Android App? Nein

Motivierend: Das Backgammon Spiel ist ein interessantes Strategie und Logik Spiel.

Spielidee: Das Backgammon Spiel ist eine Kombination aus einem Würfel- und Logik Spiel. Das Ziel des Spiels besteht darin, dass man zunächst alle Spielsteine in das untere rechte Spielbrettviertel bringt und anschließend hinauswürfelt. Des Weiteren gibt es verschiedene Sonderregeln.

Künstliche Intelligenz: Die künstliche Intelligenz muss sehr viele Spezialfälle beachten. Im Prinzip kann mit einem Min-Max-Algorithmus das Problem gelöst werden.

Datenstrukturen: Das Spielfeld ist sehr komplex in einer Datenstruktur zu erfassen. Im Prinzip ist dies nur mit verketteten Listen möglich.

Erweiterbares Spielbrett: Das Spielbrett hat eine feste Größe.

Direktes Ausschlusskriterium: Komplexität des Spiels ist zu aufwendig für die kurze Zeit im Workshop.

Damenverteilungsspiel

Existiert eine Android App? Nein

Motivierend: Das Damenverteilungsproblem wird in den Grundvorlesungen angesprochen und als angehender Informatiker möchte man die Hintergründe erfahren.

Spielidee: Auf einem Schachbrett mit n mal n Feldern sollen n Damen so verteilt werden, dass keine Dame eine andere Schlagen kann.

Künstliche Intelligenz: Keine künstliche Intelligenz notwendig, weil es kein Spiel ist.

Datenstrukturen: Das Schachbrett ist ein zweidimensionales Array und die Verteilung der Damen erfolgt über einen Suchbaum.

Erweiterbares Spielbrett: Die Spielfeldgröße kann nach belieben verkleinert und vergrößert werden. Die Laufzeit wird bei einer Vergrößerung länger.

Direktes Ausschlusskriterium: Es ist kein klassisches Spiel und wenn man sich die Lösung einmal angeschaut hat, ist der Reiz weg.

Go-Moku/ Gomoku

Existiert eine Android App? Ja

Motivierend: Ist als Variante des Go-Spiels eines der meist gespielten Spiele auf der Welt.

Spielidee: Zwei Spieler legen abwechselnd auf einem Spielbrett der Größe 19 mal 19 oder auf einem Damenbrett Steine in ihrer Farbe und versuchen fünf Steine der eigenen Farbe nebeneinander zu legen.

Künstliche Intelligenz: Mit einem Min-Max-Algorithmus kann das Verhalten für den Computer bestimmt werden.

Datenstrukturen: Das Spielbrett ist ein Array, für die Suche wird eine Liste benötigt und für die Suche kann ein Suchbaum eingesetzt werden.

Erweiterbares Spielbrett: Die Größe des Spielbretts ist beliebig. Bei zu kleinen Spielbrettern sind allerdings zu wenig Möglichkeiten zum Sieg gegeben.

Direktes Ausschlusskriterium: Nein

Go-Bang

Existiert eine Android App? Ja

Motivierend: Das Spiel ist interessanter als „Vier-gewinnt“ und „Go-Moku“ und von der Programmierung nicht viel komplizierter.

Spielidee: Das Go-Bang Spiel ist eine Abwandlung des bekannten chinesischen Go Spiels. Das Spiel wird ebenfalls auf einem Spielbrett mit 19 mal 19 Spielfeldzellen gespielt. Beide Spieler legen abwechseln jeweils einen Stein in der gewählten Farbe auf das Spielbrett. Das Ziel des Spiels ist eine ununterbrochene Spielsteinreihe von fünf Steinen auf das Spielbrett zu legen. Wenn ein gegnerischer Stein mit zwei Steinen eingeschlossen wird, ist der Stein geschlagen und der Stein wird gegen einen Stein in der anderen Farbe ausgetauscht.

Künstliche Intelligenz: Mit dem Min-Max-Algorithmus kann die künstliche Intelligenz zum Leben erweckt werden.

Datenstrukturen: Das Spielbrett ist ein zweidimensionales Array mit 19x19 Kästchen. Die Suche erfolgt mit einem Suchbaum.

Erweiterbares Spielbrett: Das Spielbrett kann zunächst verkleinert werden, um die Struktur einzuüben.

Direktes Ausschlusskriterium: Durch die Option des Schagens ist die Spiellogik relativ komplex.

Käse-Kästchen

Existiert eine Android App? Ja, unter der Bezeichnung „Dots and Boxes“.

Motivierend: Das Spiel gehört zu den Klassikern zum Vertreiben von Langeweile.

Spielidee: Auf einer umrandeten karierten Fläche markieren die Spieler abwechselnd nacheinander die Ränder der Kästchen. Das Ziel des Spiels besteht darin, bei möglichst vielen Kästchen den letzten Strich zu machen und damit das Kästchen für sich zu gewinnen.

Künstliche Intelligenz: Das Spiel Käse-Kästchen kann mit dem Min-Max-Algorithmus gelöst werden.

Datenstrukturen: Die Kästchen können mit einem Array dargestellt werden. Dabei ist zu beachten, dass mit komplexen Datentypen die vier Striche pro Kästchen repräsentiert werden.

Erweiterbares Spielbrett: Die Größe des Spielbretts ist flexibel.

Direktes Ausschlusskriterium: Die Datenstruktur ist relativ komplex, weil zu jedem Kästchen vier Striche gespeichert werden müssen. Zum Beispiel ist die untere Linie in einem anderen Kästchen auch als obere Linie gespeichert. Dies gibt sehr viele Sonderfälle und ist nicht intuitiv.

Master Mind

Existiert eine Android App? Ja, unter der Bezeichnung „Mind Master“.

Motivierend: Das Spiel ist relativ unbekannt und kann eventuell auch mit ein wenig Kodierungstheorie verbunden werden.

Spielidee: Die Spieler kodieren gegenseitig eine Nachricht mit Farbsymbolen. Anschließend versucht der jeweils andere Spieler diese Nachrichten in möglichst wenigen Schritten zu erraten. Das Raten wird durch Lösungsversuche und der Anzahl an falsch gesetzten Farben durchgeführt.

Künstliche Intelligenz: Ein Computer kann den Code mit den Tipps dekodieren.

Datenstrukturen: Das zu erratende Farbgebilde kann mit einem Array repräsentiert werden.

Erweiterbares Spielbrett: Das Farbgebilde und die Anzahl an Farben kann flexibel erweitert werden.

Direktes Ausschlusskriterium: Das Spiel ist nach wenigen Runden relativ langweilig und es gibt recht wenig Datenstrukturen.

Maus-Käse-Spiel

Existiert eine Android App? Ja

Motivierend: Es ist kein klassisches Spiel. Deswegen kann es ziemlich schnell langweilig werden.

Spielidee: Im einem Labyrinth befindet sich eine Maus und ein Käsestück. Die Maus soll auf einem möglichst kurzen weg zum Käse gebracht werden.

Künstliche Intelligenz: Das Labyrinth muss in einen Baum transformiert werden und eine Wegsuche zwischen dem Käse und der Maus durchgeführt werden.

Datenstrukturen: Das Labyrinth ist zu Beginn ein Array und wird in einen Graphen transformiert.

Erweiterbares Spielbrett: Das Labyrinth kann in einer beliebigen Größe erstellt werden.

Direktes Ausschlusskriterium: Auf einem kleinen Handydisplay wird das Labyrinth nur wenige Gänge haben und die Lösung offensichtlich sein.

Mühle

Existiert eine Android App? Ja

Motivierend: Das Mühle Spiel ist als Logikspiel eine große Herausforderung.

Spielidee: Der Spielablauf des Mühle Spiels ist in drei Phasen aufgeteilt:

- **Setzphase:** Auf das zu Beginn leere Spielfeld legen die beiden Spieler abwechselnd einen Spielstein auf einen Eckpunkt oder einer Kreuzung der Linien auf dem Spielbrett. Jedem Spieler stehen insgesamt neun Spielsteine zur Verfügung.
- **Zugphase:** In der Zug Phase werden die Steine auf dem Spielbrett verschoben, um eine Mühle zu bilden. Wenn eine Mühle gebildet wurde, darf der Spieler einen gegnerischen Stein, der sich nicht in einer Mühle befindet, entfernen.
- **Endphase:** Wenn ein Spieler nur noch drei Steine hat, darf er mit jeweils einem Stein pro Runde auch springen. Sobald er weniger als drei Steine hat, hat er verloren.

Künstliche Intelligenz: Das Spiel kann mit dem Alpha Beta bzw. dem Min-Max-Algorithmus vom Computer gespielt werden.

Datenstrukturen: Das Spielfeld ist sehr komplex darzustellen und die Überprüfung der Mühle ist mit sehr vielen Ausnahmen verbunden.

Erweiterbares Spielbrett: Das Spielbrett ist vorgegeben und eine andere Form macht keinen Sinn.

Direktes Ausschlusskriterium: Die drei Phasen sind mit unterschiedlichen Spielregeln verbunden und des Weiteren ist das Spielbrett recht kompliziert darzustellen.

Onplon

Existiert eine Android App? Ja

Motivierend: Das Spiel ist relativ unbekannt und deswegen wäre es für sehr viele Studenten etwas komplett neues.

Spielidee: Das Onplon-Spiel ist ein Zahl Platzierungs Puzzle. Zu Beginn des Spiels ist das Spielbrett mit Zahlen gefüllt. Der Spieler darf eine Zahl wegstreichen, wenn die benachbarte Zahl entweder die gleiche Zahl ist oder das Zehner-Komplement der Zahl ist. Wenn also eine zwei auf dem Spielbrett steht, kann diese mit einer benachbarten zwei oder acht entfernt werden.

Künstliche Intelligenz: Die Generierung des Spielfelds muss so erfolgen, dass es möglich ist alle Zahlen zu entfernen.

Datenstrukturen: Das Spielfeld ist ein zweidimensionales Array. Bei der Erzeugung des Spielfelds wird ein Graph-Algorithmus benötigt.

Erweiterbares Spielbrett: Das Spielfeld ist beliebig erweiterbar.

Direktes Ausschlusskriterium: Nein

Reversi

Existiert eine Android App? Ja

Motivierend: Das Spiel erfreut sich großer Beliebtheit und ist trotzdem nicht jedem bekannt.

Spielidee: Auf einem Schachbrett werden in der Mitte vier Steine der beiden Spielfarben positioniert. Anschließend legen die beiden Spieler weitere Steine an und drehen die eingeschlossenen Spielsteine um. Das Ziel des Spiels besteht darin, dass zum Schluss möglichst viele Steine der eigenen Farbe auf dem Spielbrett liegen.

Künstliche Intelligenz: Das Spiel kann mit dem Alpha Beta bzw. dem Min-Max-Algorithmus vom Computer gespielt werden.

Datenstrukturen: Das Spielfeld ist ein zweidimensionales Array und die Schritte des Computers werden mit einem Suchbaum bestimmt.

Erweiterbares Spielbrett: Das Spielbrett kann zu Beginn verkleinert werden.

Direktes Ausschlusskriterium: Nein

Rubido (vgl. Solitär)

Existiert eine Android App? Ja, unter der Bezeichnung „Solitär Mania“.

Motivierend: Die Herausforderung alle Stäbe zu entfernen löst eine gewisse Begeisterung aus. Wenn der Weg verstanden wurde bzw. kein Erfolg in Sicht ist, wird es schnell demotivierend.

Spielidee: Zu Beginn des Spiels ist das Spielbrett mit stehenden Holzsteinen gefüllt. Der Spieler wählt einen Stein aus und kann einen Nachbarstein überspringen. Dadurch wird der Nachbarstein „gelöscht“. Das Ziel des Spiels besteht darin, dass man bis auf einen Spielstein die Spielsteine durch überspringen entfernt. Der letzte Stein ist vorgegeben und steht meistens in der Mitte des Spielbretts.

Künstliche Intelligenz: Die künstliche Intelligenz soll die minimale Anzahl an verbleibenden Steinen bestimmen.

Datenstrukturen: Das Spielfeld ist ein mehrdimensionales Array.

Erweiterbares Spielbrett: Die Größe des Spielbretts ist festgelegt, damit eine Möglichkeit existiert alle Steine zu entfernen.

Direktes Ausschlusskriterium: Nein

Schach

Existiert eine Android App? Ja

Motivierend: Das Schachspiel erfreut sich seit langer Zeit immer wieder erneuter Beliebtheit.

Spielidee: Komplexe Spielregeln, durch sechs unterschiedliche Spielfiguren mit jeweils anderem Verhalten. Insgesamt existieren 32 Spielfiguren, die fast alle in jeder Runde bewegt werden können.

Künstliche Intelligenz: Die Bearbeitung des aktuellen Spielstandes kann zu Problemen führen.

Datenstrukturen: Das Spielfeld könnte als Array dargestellt werden und die einzelnen Spielfiguren in einer eigenen Klasse als Implementierung eines Interfaces der Spielfiguren.

Erweiterbares Spielbrett: Die Größe des Spielbretts ist fest und eine Verkleinerung ist nicht möglich.

Direktes Ausschlusskriterium: Die Spielregeln und die Anzahl der Figuren sind zu komplex für die Programmierung in einem beschränkten Zeitraum.

Schiffe Versenken

Existiert eine Android App? Ja

Motivierend: Das Spiel ist kurzfristig interessant. Allerdings wird es durch den großen Zufallsfaktor leicht langweilig.

Spielidee: Zwei Gegner einigen sich auf eine Anzahl von Schiffen und die Größe der Schiffe, diese entspricht der Anzahl notwendiger Treffer bis zum Untergang des Schiffs. Die Schiffe werden auf einem rechteckigen Spielfeld positioniert. Anschließend versuchen die Spieler gegenseitig, die Schiffe des Gegners zu treffen, indem Koordinaten genannt werden und ein dort befindliches Schiff als getroffen gilt. Es hat derjenige gewonnen, der zuerst alle Schiffe des anderen Spielers versenkt hat.

Künstliche Intelligenz: Bei der künstliche Intelligenz ist zum Großteil der Zufall entscheidend. Erst im späteren Spielverlauf kann anhand der Daten eine Analyse stattfinden.

Datenstrukturen: Das Wasser ist in Quadrate aufgeteilt, deswegen liegt ein Array dem Spielfeld zu Grunde.

Erweiterbares Spielbrett: Die Anzahl Schiffe und die Größe des Spielfelds können variieren.

Direktes Ausschlusskriterium: In dem Spiel hat der Zufall eine zu große und die Theorie eine zu kleine Rolle.

Solitär (vgl. Rubido)

Existiert eine Android App? Ja, unter der Bezeichnung „Solitär Mania“.

Motivierend: Die Herausforderung alle Stäbe zu entfernen löst eine gewisse Begeisterung aus. Wenn der Weg verstanden wurde bzw. kein Erfolg in Sicht ist, wird es schnell demotivierend.

Spielidee: Zu Beginn des Spiels ist das Spielbrett mit stehenden Holzsteinen gefüllt. Der Spieler wählt einen Stein aus und kann einen Nachbarn überspringen. Dadurch wird der Nachbar „gelöscht“. Das Ziel des Spiels besteht darin, dass man bis auf einen Spielstein die Spielsteine durch überspringen entfernt. Der letzte Stein ist vorgegeben und steht meistens in der Mitte des Spielbretts.

Künstliche Intelligenz: Die verbleibende Anzahl an Steinen soll mit der künstlichen Intelligenz bestimmt werden.

Datenstrukturen: Das Spielfeld ist ein mehrdimensionales Array.

Erweiterbares Spielbrett: Die Größe des Spielbretts ist festgelegt, damit eine Möglichkeit existiert alle Stäbe zu entfernen.

Direktes Ausschlusskriterium: Nein

Tackle

Existiert eine Android App? Nein

Motivierend: Tackle ist ein neueres Logik Spiel der Süddeutschen Zeitung.

Spielidee: Bei Tackle sollen vorgegebene Figuren auf einem Spielfeld gelegt werden. Es wird abwechselnd jeweils ein Spielstein gelegt. Dabei kann der Gegner auch am weiteren Aufbau gehindert werden. Es gibt sehr viele Einschränkungen und Besonderheiten.

Künstliche Intelligenz: Das Spiel kann mit dem Alpha Beta bzw. dem Min-Max-Algorithmus durch den Computer gespielt werden.

Datenstrukturen: Das Spielfeld ist ein Array und die Schritte des Computers werden mit einem Suchbaum bestimmt.

Erweiterbares Spielbrett: Die Größe des Spielfelds ist vorgegeben.

Direktes Ausschlusskriterium: Sehr viele Ausnahmen in den Spielregeln und dadurch höchstwahrscheinlich zu komplex.

Triominos

Existiert eine Android App? Ja

Motivierend: Das Legespiel ist eine tolle Erweiterung des bekannten Domino Spiels.

Spielidee: Das Spiel Triominos ist eine Erweiterung des Domino Spiels. Der große Unterschied besteht in der Form der Spielsteine. Bei Triominos sind die Spielsteine dreieckig und an allen drei Kanten stehen Zahlen. Das Ziel des Spiels besteht im erreichen einer möglichst hohen Punktzahl.

Künstliche Intelligenz: Die Optionen des Anlegen und der Spieloptionen, die dadurch kreiert werden, können mit einer künstlichen Intelligenz analysiert werden.

Datenstrukturen: Die Dreiecke können mit einem Graphen repräsentiert werden. Allerdings gestaltet sich der Aufbau des Spielfelds dadurch wesentlich komplexer.

Erweiterbares Spielbrett: Die Anzahl der Steine ist flexibel zu gestalten.

Direktes Ausschlusskriterium: Die Datenstruktur ist für den Kenntnisstand der Studenten sehr komplex.

Vier Gewinnt

Existiert eine Android App? Ja, unter der Bezeichnung „4 in a Line“.

Motivierend: Das Spiel gehört zu den Klassikern, die jedes Kind spielt. Das Spiel wird sehr oft für Programmierkurse verwendet. Daher ist es keine innovative Idee.

Spielidee: In diesem Spiel treten zwei Gegner gegeneinander an und werfen in ein aufrechtstehendes Spielbrett mit sieben Spalten und fünf Zeilen nacheinander Steine ihrer Farbe ein. Die Steine fallen in der jeweiligen Spalte bis zur untersten Position durch. Ziel ist es vier Steine der eigenen Farbe nebeneinander zu haben.

Künstliche Intelligenz: Die künstliche Intelligenz für den Computer kann mit dem Min-Max-Algorithmus angestellt werden.

Datenstrukturen: Die Programmierung des Spielbretts kann mit einem mehrdimensionalen Array erfolgen.

Erweiterbares Spielbrett: Die Möglichkeiten für die Veränderung der Größe des Spielfelds sind sehr stark beschränkt. Ansonsten ist es zu leicht zu gewinnen, wenn die Anzahl der benötigten Spielsteine verringert wird bzw. überhaupt keine Möglichkeit besteht zu gewinnen, wenn das Spielfeld verkleinert wird.

Direktes Ausschlusskriterium: Das Spiel wurde sehr oft für Programmierkurse verwendet und ist dadurch keine innovative Idee.

E. Evaluierung

E.1. Fragebogen

Für die Befragung der Studenten wurden zwei Erhebungszeitpunkte ausgewählt. Der Vor-Test wurde vor dem Workshop und der Nach-Test wurde zum Abschluss des Workshops durchgeführt. (vgl. Diekmann (2011), S. 356)

Die Fragebögen stammen aus dem Forschungsprojekt „IGaDtools4MINT“ und wurden nicht im Rahmen dieser Arbeit entwickelt, sondern nur für die Evaluierung herangezogen. (vgl. Holz (2012a))

E.1.1. Vor-Test

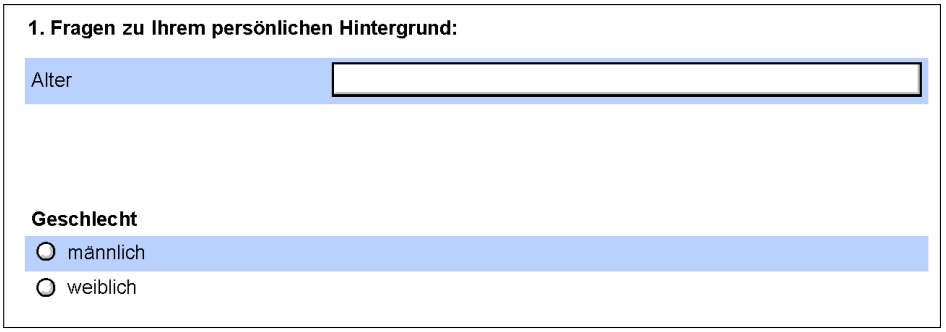


Abbildung E.1.: Persönlicher Hintergrund

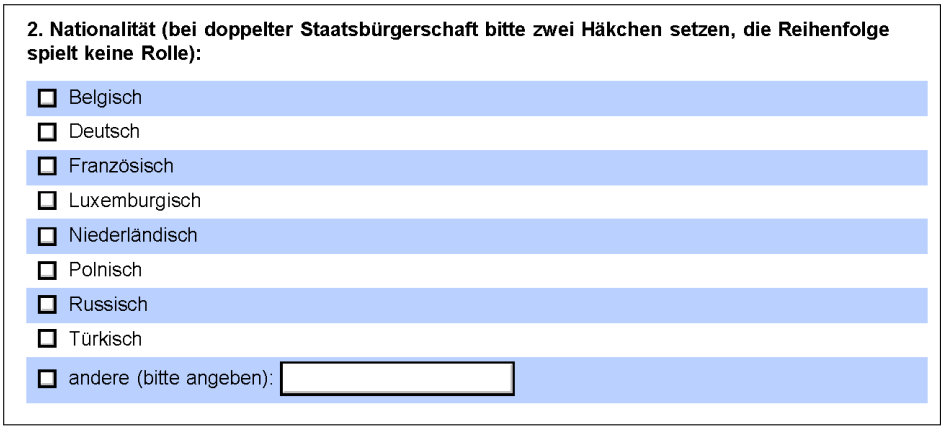


Abbildung E.2.: Nationalität

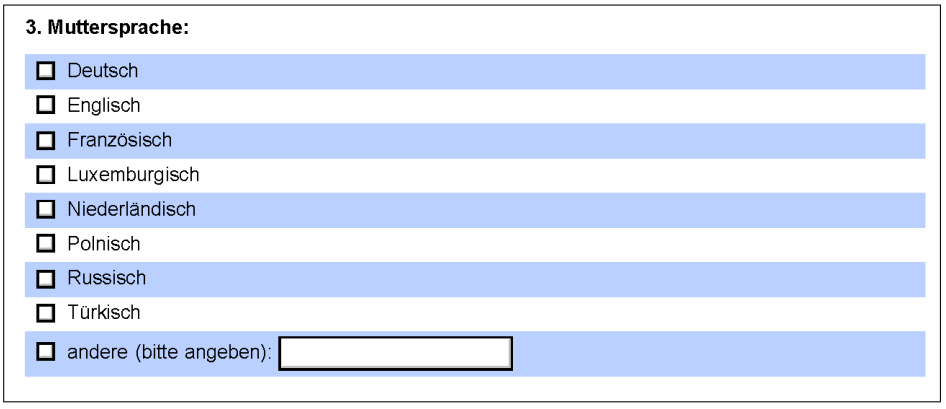


Abbildung E.3.: Muttersprache

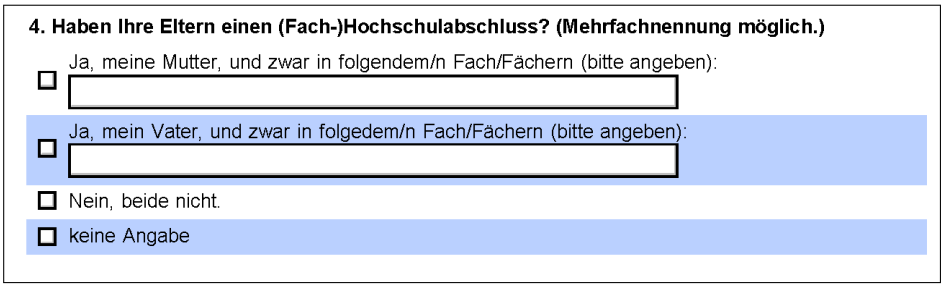


Abbildung E.4.: Haben Ihre Eltern einen (Fach-) Hochschulabschluss?

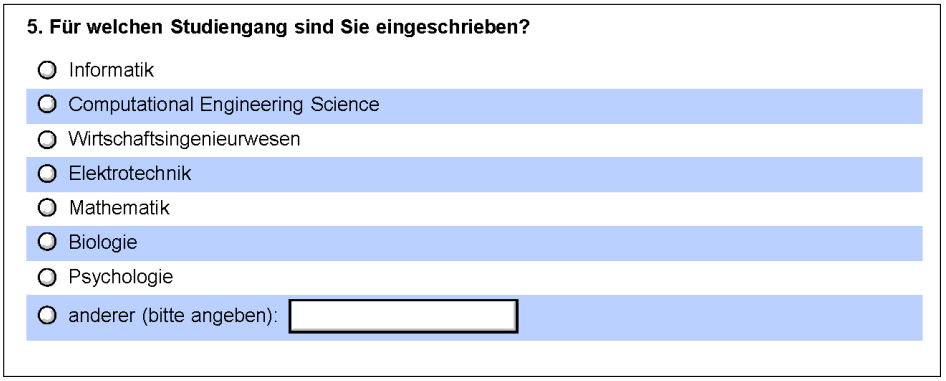


Abbildung E.5.: Für welchen Studiengang sind Sie eingeschrieben?

Abbildung E.6.: Im wievielten Semester befinden Sie sich im Sommersemester 2013?

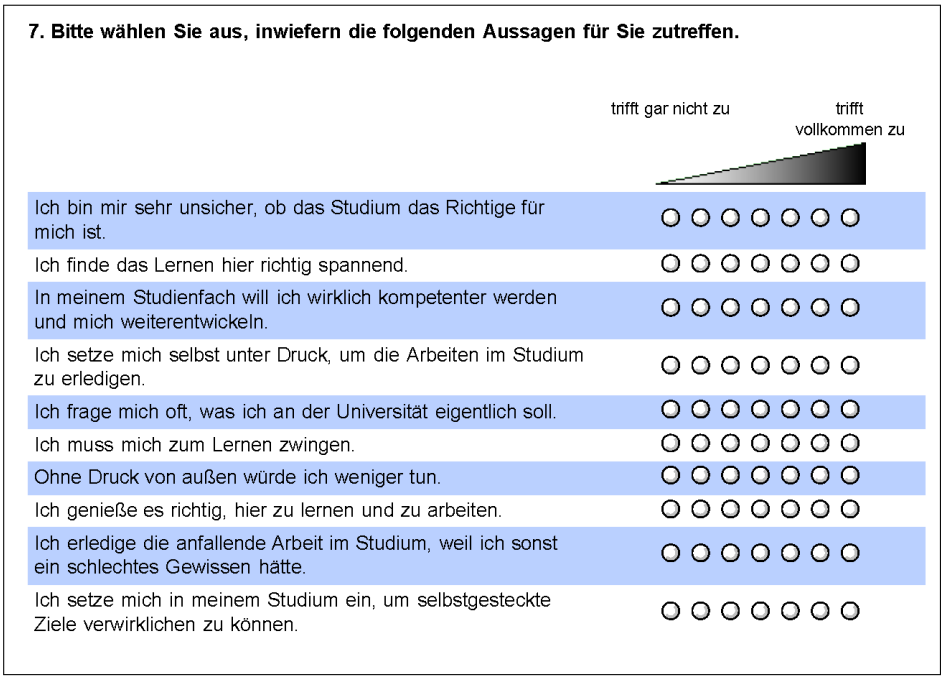


Abbildung E.7.: Bitte wählen Sie aus, inwiefern die folgenden Aussagen für Sie zutreffen.

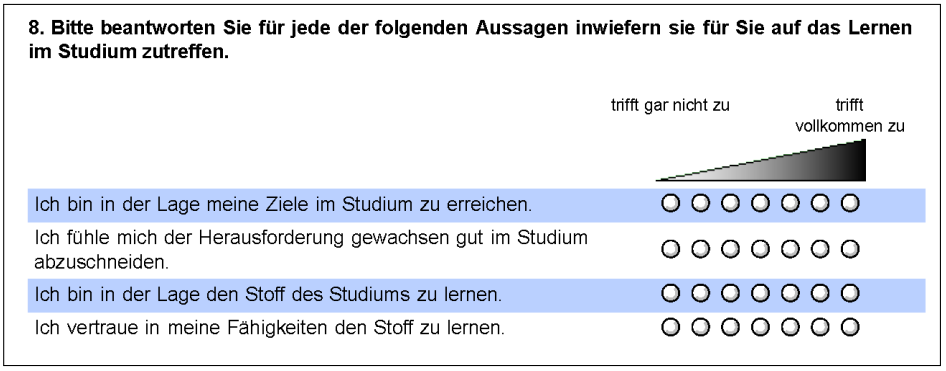
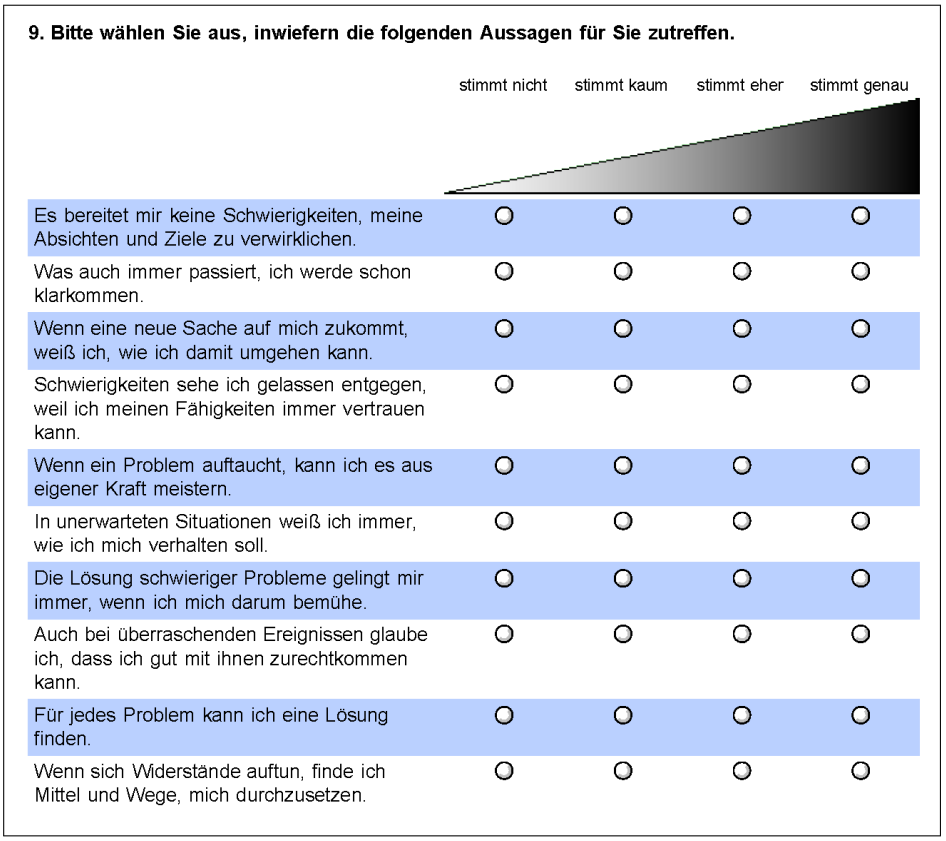


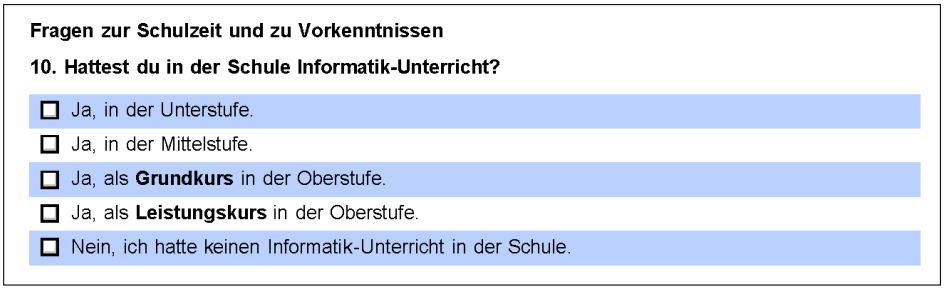
Abbildung E.8.: Bitte beantworten Sie für jede der folgenden Aussagen inwiefern sie für Sie auf das Lernen im Studium zutreffen.



9. Bitte wählen Sie aus, inwiefern die folgenden Aussagen für Sie zutreffen.

	stimmt nicht	stimmt kaum	stimmt eher	stimmt genau
Es bereitet mir keine Schwierigkeiten, meine Absichten und Ziele zu verwirklichen.	○	○	○	○
Was auch immer passiert, ich werde schon klarkommen.	○	○	○	○
Wenn eine neue Sache auf mich zukommt, weiß ich, wie ich damit umgehen kann.	○	○	○	○
Schwierigkeiten sehe ich gelassen entgegen, weil ich meinen Fähigkeiten immer vertrauen kann.	○	○	○	○
Wenn ein Problem auftaucht, kann ich es aus eigener Kraft meistern.	○	○	○	○
In unerwarteten Situationen weiß ich immer, wie ich mich verhalten soll.	○	○	○	○
Die Lösung schwieriger Probleme gelingt mir immer, wenn ich mich darum bemühe.	○	○	○	○
Auch bei überraschenden Ereignissen glaube ich, dass ich gut mit ihnen zurechtkommen kann.	○	○	○	○
Für jedes Problem kann ich eine Lösung finden.	○	○	○	○
Wenn sich Widerstände auftun, finde ich Mittel und Wege, mich durchzusetzen.	○	○	○	○

Abbildung E.9.: Bitte wählen Sie aus, inwiefern die folgenden Aussagen für Sie zutreffen.



Fragen zur Schulzeit und zu Vorkenntnissen

10. Hattest du in der Schule Informatik-Unterricht?

- ☐ Ja, in der Unterstufe.
- ☐ Ja, in der Mittelstufe.
- ☐ Ja, als **Grundkurs** in der Oberstufe.
- ☐ Ja, als **Leistungskurs** in der Oberstufe.
- ☐ Nein, ich hatte keinen Informatik-Unterricht in der Schule.

Abbildung E.10.: Hattest du in der Schule Informatik-Unterricht?

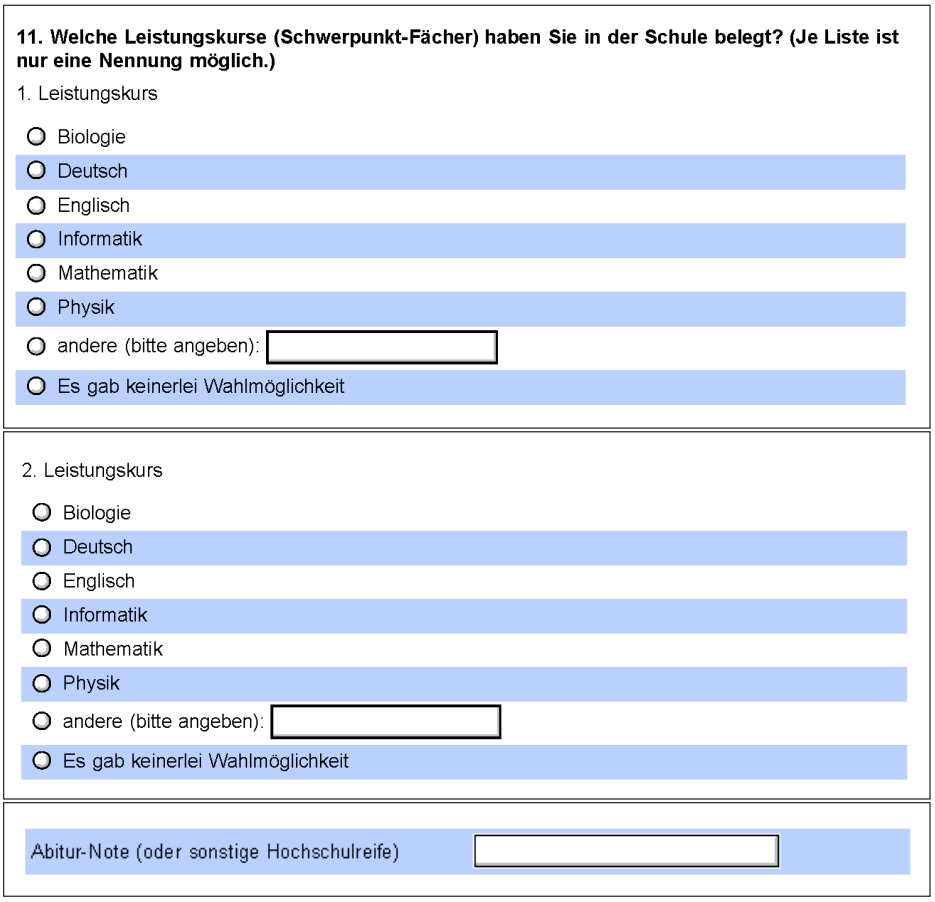
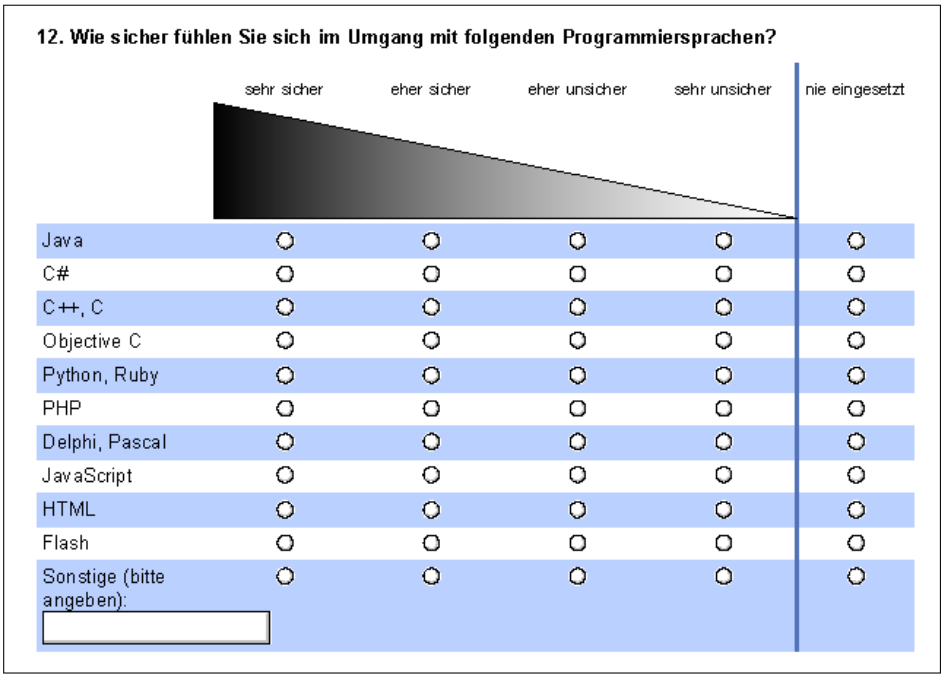


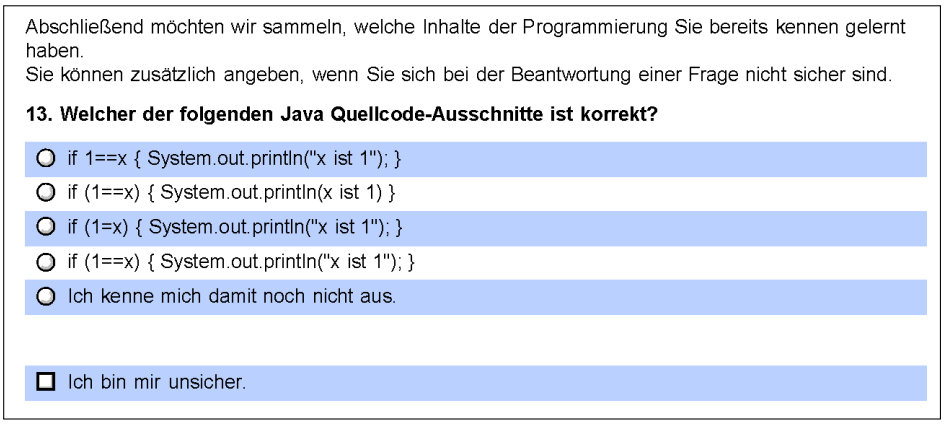
Abbildung E.11.: Welche Leistungskurse (Schwerpunkt-Fächer) haben Sie in der Schule belegt?



12. Wie sicher fühlen Sie sich im Umgang mit folgenden Programmiersprachen?

	sehr sicher	eher sicher	eher unsicher	sehr unsicher	nie eingesetzt
Java	○	○	○	○	○
C#	○	○	○	○	○
C++, C	○	○	○	○	○
Objective C	○	○	○	○	○
Python, Ruby	○	○	○	○	○
PHP	○	○	○	○	○
Delphi, Pascal	○	○	○	○	○
JavaScript	○	○	○	○	○
HTML	○	○	○	○	○
Flash	○	○	○	○	○
Sonstige (bitte angeben):	○	○	○	○	○

Abbildung E.12.: Wie sicher fühlen Sie sich im Umgang mit folgenden Programmiersprachen?



Abschließend möchten wir sammeln, welche Inhalte der Programmierung Sie bereits kennen gelernt haben.
Sie können zusätzlich angeben, wenn Sie sich bei der Beantwortung einer Frage nicht sicher sind.

13. Welcher der folgenden Java Quellcode-Ausschnitte ist korrekt?

- ○ if 1==x { System.out.println("x ist 1"); }
- ○ if (1==x) { System.out.println(x ist 1) }
- ○ if (1=x) { System.out.println("x ist 1"); }
- ○ if (1==x) { System.out.println("x ist 1"); }
- ○ Ich kenne mich damit noch nicht aus.

- ☐ Ich bin mir unsicher.

Abbildung E.13.: Welcher der folgenden Java Quellcode-Ausschnitte ist korrekt?

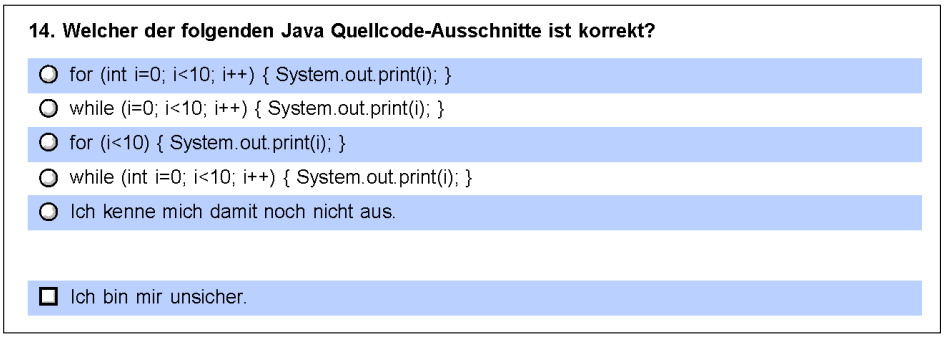


Abbildung E.14.: Welcher der folgenden Java Quellcode-Ausschnitte ist korrekt?

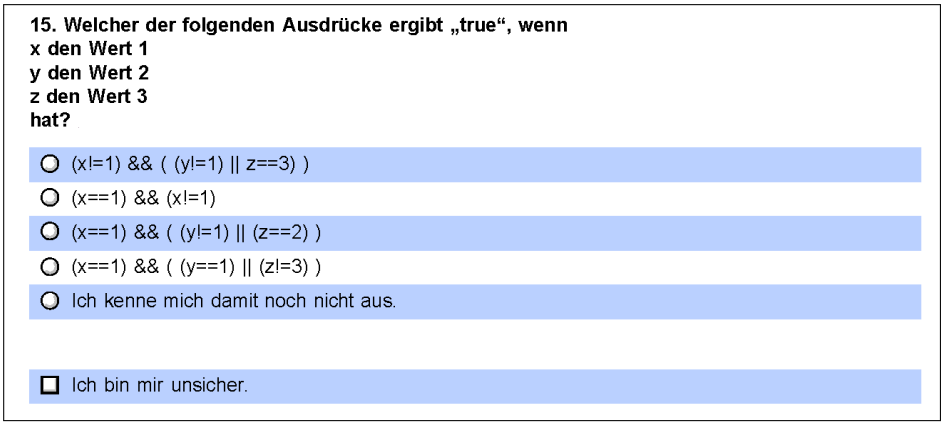


Abbildung E.15.: Welcher der folgenden Ausdrücke ergibt „true“?

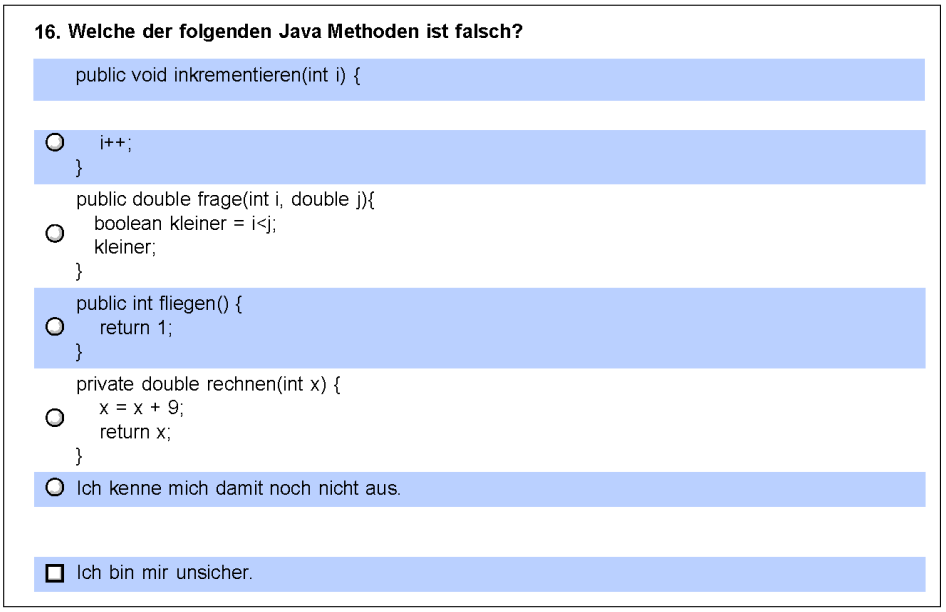


Abbildung E.16.: Welche der folgenden Java Methoden ist falsch?

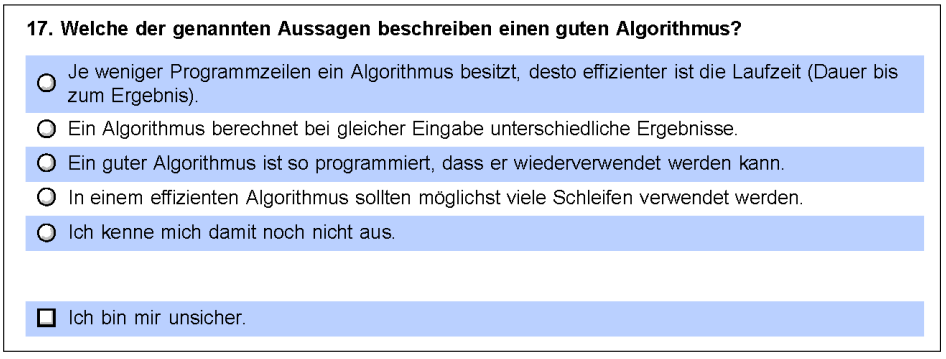


Abbildung E.17.: Welche der genannten Aussagen beschreiben einen guten Algorithmus?

E.1.2. Nach-Test

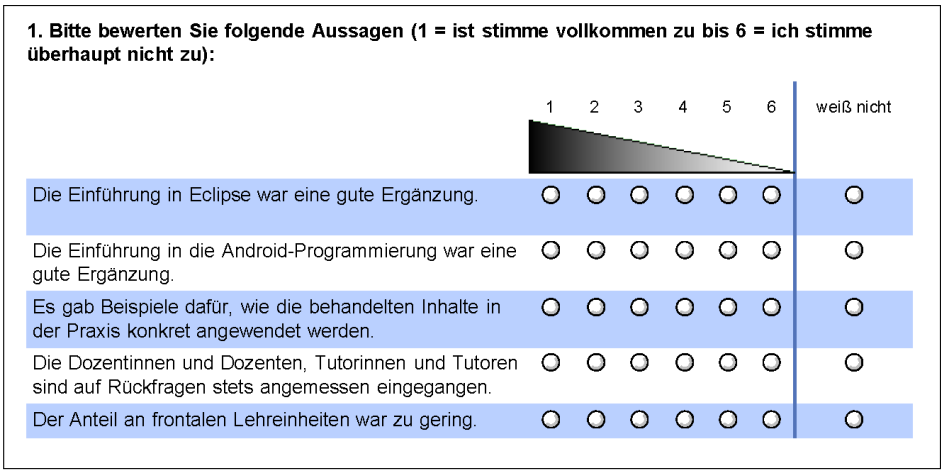


Abbildung E.18.: Bitte bewerten Sie folgende Aussagen?

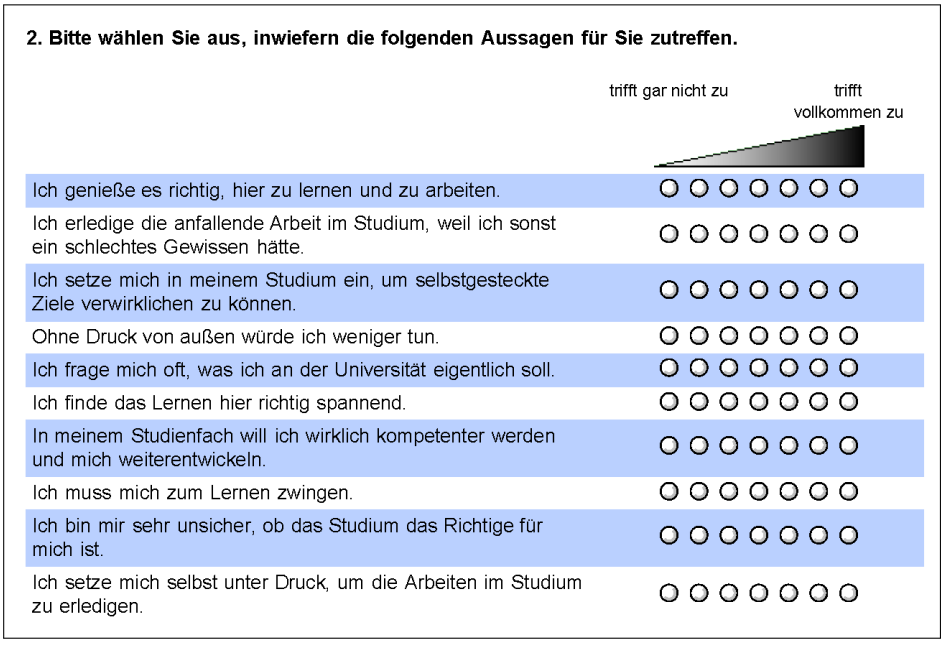
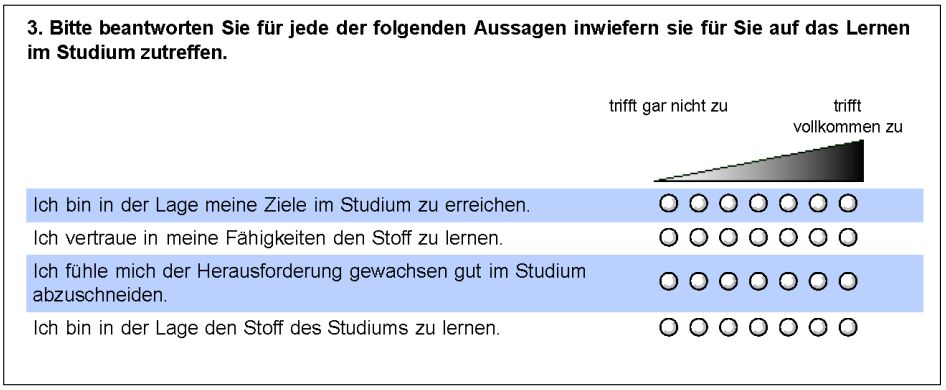


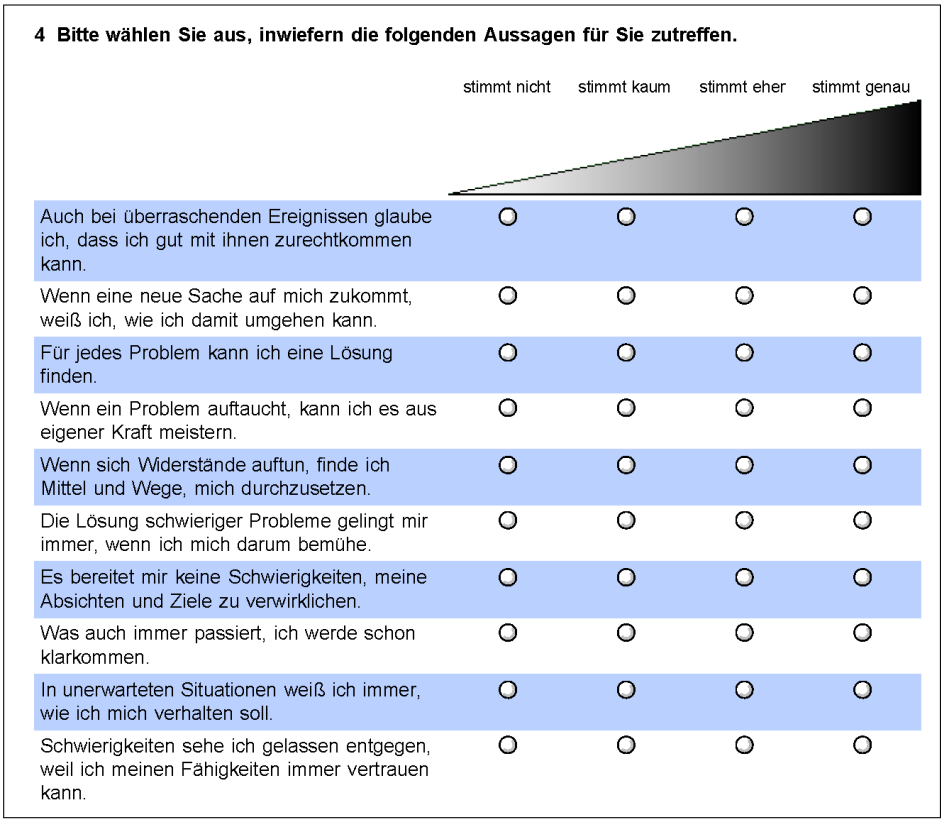
Abbildung E.19.: Bitte wählen Sie aus, inwiefern die folgenden Aussagen für Sie zutreffen.



3. Bitte beantworten Sie für jede der folgenden Aussagen inwiefern sie für Sie auf das Lernen im Studium zutreffen.

	trifft gar nicht zu						trifft vollkommen zu
Ich bin in der Lage meine Ziele im Studium zu erreichen.	○	○	○	○	○	○	○
Ich vertraue in meine Fähigkeiten den Stoff zu lernen.	○	○	○	○	○	○	○
Ich fühle mich der Herausforderung gewachsen gut im Studium abzuschneiden.	○	○	○	○	○	○	○
Ich bin in der Lage den Stoff des Studiums zu lernen.	○	○	○	○	○	○	○

Abbildung E.20.: Bitte beantworten Sie für jede der folgenden Aussagen inwiefern sie für Sie auf das Lernen im Studium zutreffen.



4 Bitte wählen Sie aus, inwiefern die folgenden Aussagen für Sie zutreffen.

	stimmt nicht	stimmt kaum	stimmt eher	stimmt genau
Auch bei überraschenden Ereignissen glaube ich, dass ich gut mit ihnen zurechtkommen kann.	○	○	○	○
Wenn eine neue Sache auf mich zukommt, weiß ich, wie ich damit umgehen kann.	○	○	○	○
Für jedes Problem kann ich eine Lösung finden.	○	○	○	○
Wenn ein Problem auftaucht, kann ich es aus eigener Kraft meistern.	○	○	○	○
Wenn sich Widerstände auftun, finde ich Mittel und Wege, mich durchzusetzen.	○	○	○	○
Die Lösung schwieriger Probleme gelingt mir immer, wenn ich mich darum bemühe.	○	○	○	○
Es bereitet mir keine Schwierigkeiten, meine Absichten und Ziele zu verwirklichen.	○	○	○	○
Was auch immer passiert, ich werde schon klarkommen.	○	○	○	○
In unerwarteten Situationen weiß ich immer, wie ich mich verhalten soll.	○	○	○	○
Schwierigkeiten sehe ich gelassen entgegen, weil ich meinen Fähigkeiten immer vertrauen kann.	○	○	○	○

Abbildung E.21.: Bitte wählen Sie aus, inwiefern die folgenden Aussagen für Sie zutreffen.

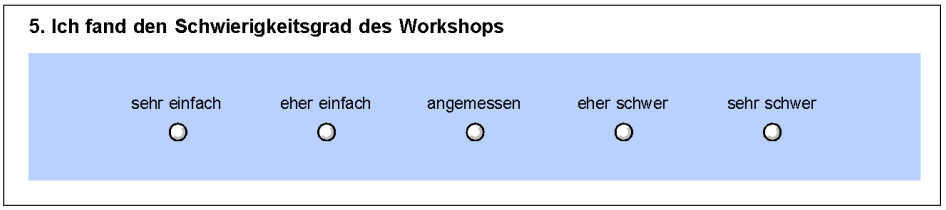


Abbildung E.22.: Ich fand den Schwierigkeitsgrad des Workshops

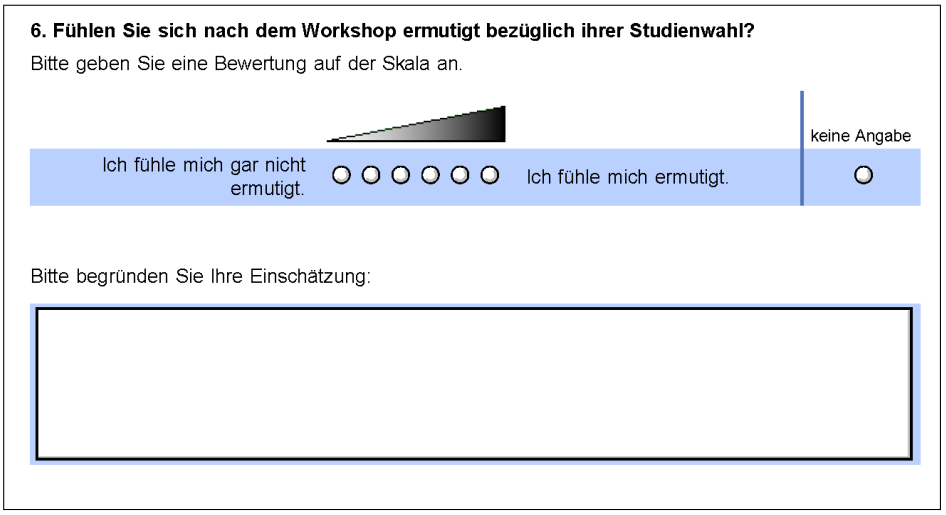


Abbildung E.23.: Fühlen Sie sich nach dem Workshop ermutigt bezüglich ihrer Studienwahl?

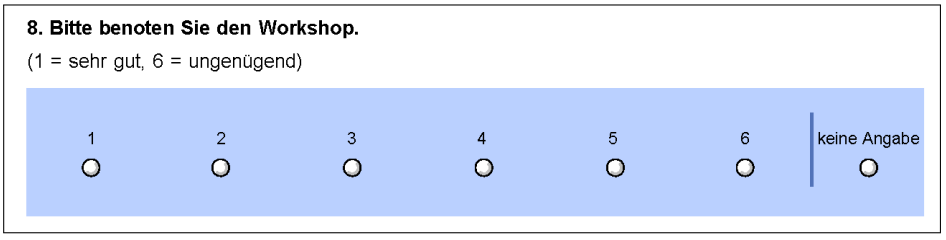


Abbildung E.24.: Bitte benoten Sie den Workshop.

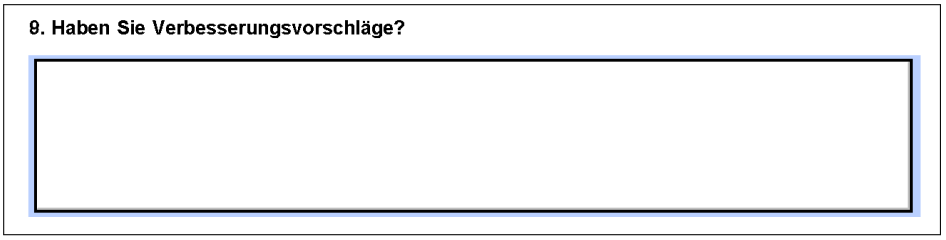
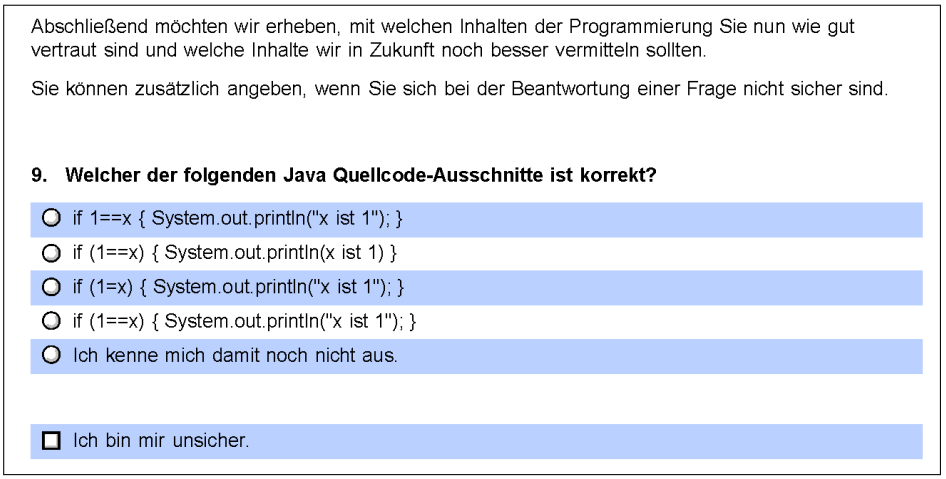


Abbildung E.25.: Haben Sie Verbesserungsvorschläge?



Abschließend möchten wir erheben, mit welchen Inhalten der Programmierung Sie nun wie gut vertraut sind und welche Inhalte wir in Zukunft noch besser vermitteln sollten.

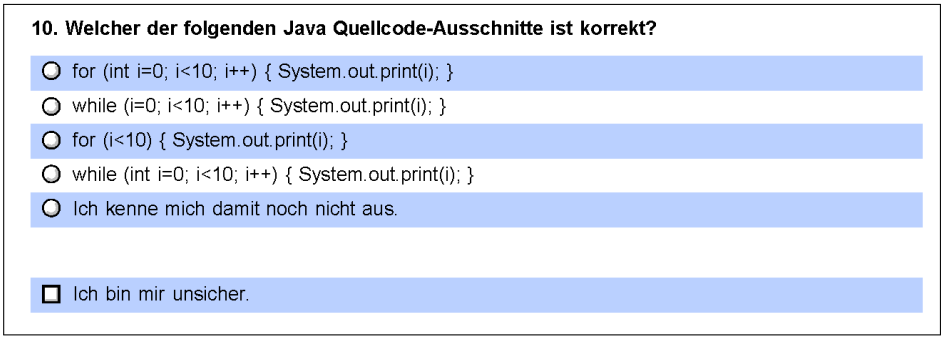
Sie können zusätzlich angeben, wenn Sie sich bei der Beantwortung einer Frage nicht sicher sind.

9. Welcher der folgenden Java Quellcode-Ausschnitte ist korrekt?

- ○ if 1==x { System.out.println("x ist 1"); }
- ○ if (1==x) { System.out.println(x ist 1) }
- ○ if (1=x) { System.out.println("x ist 1"); }
- ○ if (1==x) { System.out.println("x ist 1"); }
- ○ Ich kenne mich damit noch nicht aus.

- ☐ Ich bin mir unsicher.

Abbildung E.26.: Welcher der folgenden Java Quellcode-Ausschnitte ist korrekt?

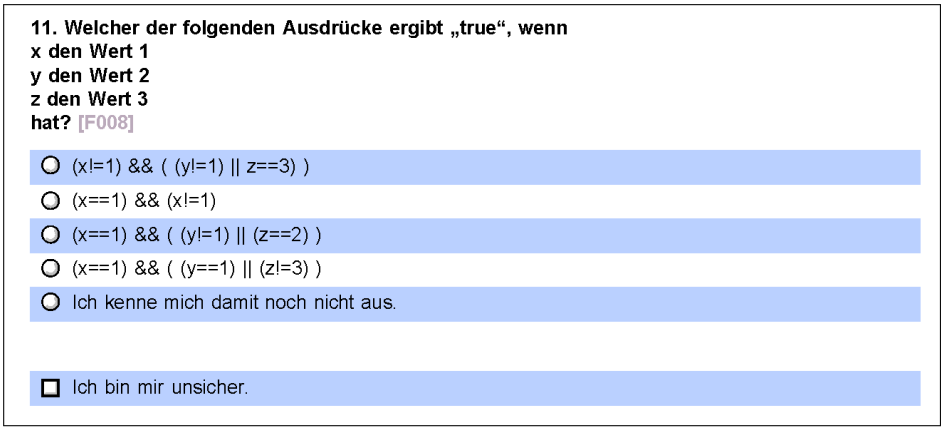


10. Welcher der folgenden Java Quellcode-Ausschnitte ist korrekt?

- ○ for (int i=0; i<10; i++) { System.out.print(i); }
- ○ while (i=0; i<10; i++) { System.out.print(i); }
- ○ for (i<10) { System.out.print(i); }
- ○ while (int i=0; i<10; i++) { System.out.print(i); }
- ○ Ich kenne mich damit noch nicht aus.

- ☐ Ich bin mir unsicher.

Abbildung E.27.: Welcher der folgenden Java Quellcode-Ausschnitte ist korrekt?



11. Welcher der folgenden Ausdrücke ergibt „true“, wenn
x den Wert 1
y den Wert 2
z den Wert 3
hat? [F008]

- ○ (x!=1) && ((y!=1) || z==3))
- ○ (x==1) && (x!=1)
- ○ (x==1) && ((y!=1) || (z==2))
- ○ (x==1) && ((y==1) || (z!=3))
- ○ Ich kenne mich damit noch nicht aus.

- ☐ Ich bin mir unsicher.

Abbildung E.28.: Welcher der folgenden Ausdrücke ergibt „true“?

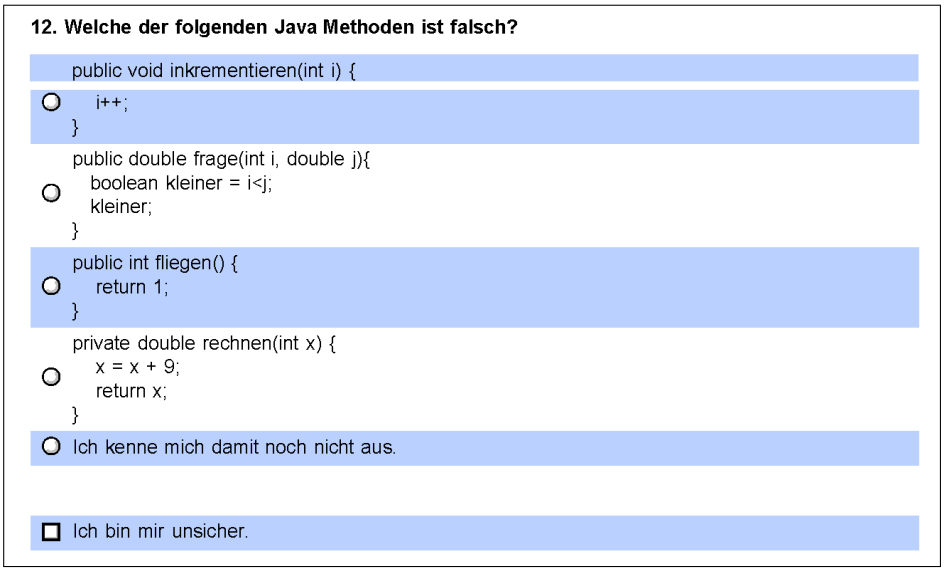


Abbildung E.29.: Welche der folgenden Java Methoden ist falsch?

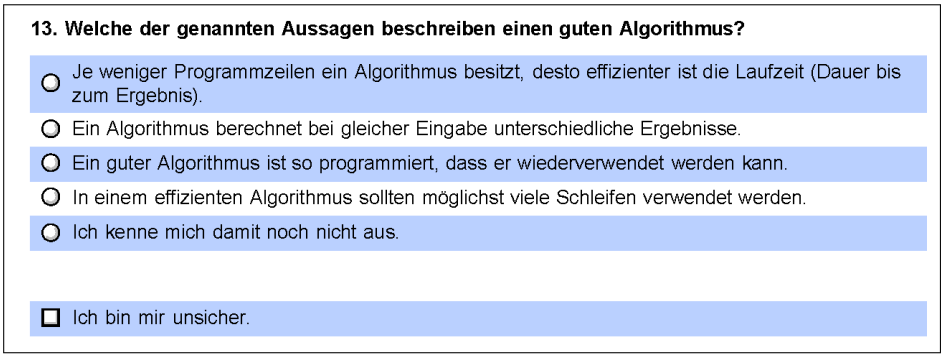


Abbildung E.30.: Welche der genannten Aussagen beschreiben einen guten Algorithmus?

E.2. Auswertung

E.2.1. Stammdaten

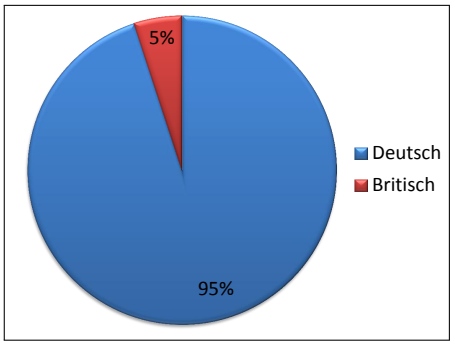


Abbildung E.31.: Nationalität (bei doppelter Staatsbürgerschaft bitte zwei Häkchen setzen, die Reihenfolge spielt keine Rolle)

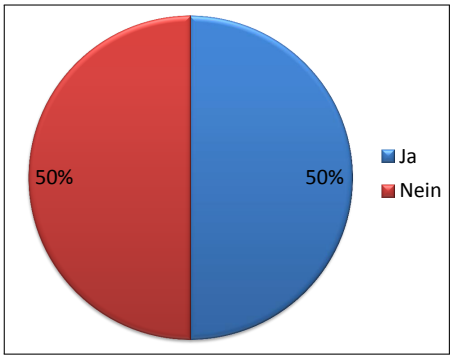


Abbildung E.32.: Haben Ihre Eltern einen (Fach-) Hochschulabschluss? (Mehrfachnennung möglich.)

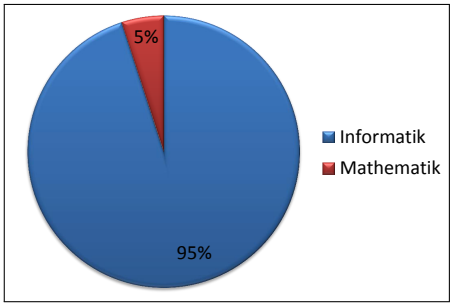
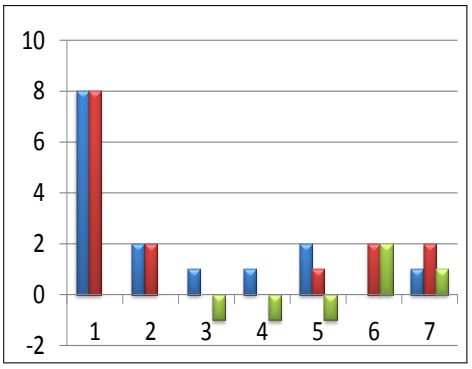
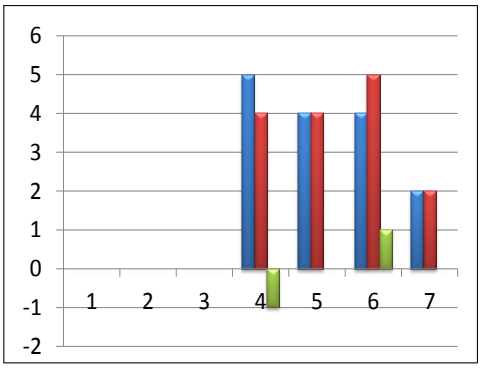


Abbildung E.33.: Für welchen Studiengang sind Sie eingeschrieben?

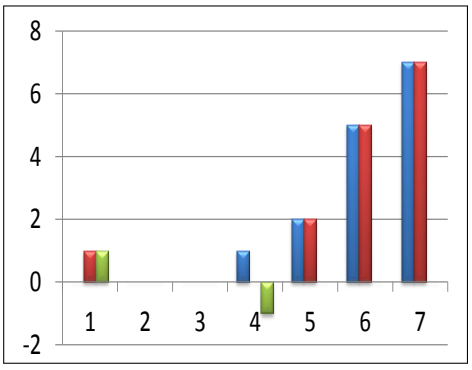
E.2.2. Zufriedenheit mit dem Studium



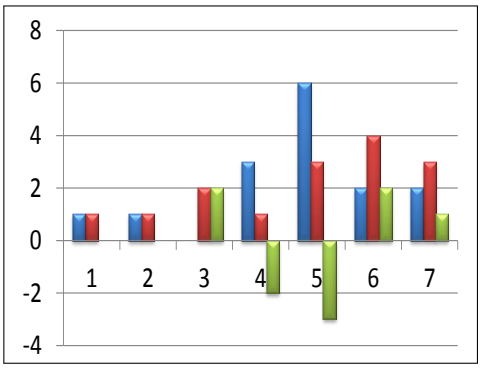
(a) Ich bin mir sehr unsicher, ob das Studium das Richtige für mich ist.



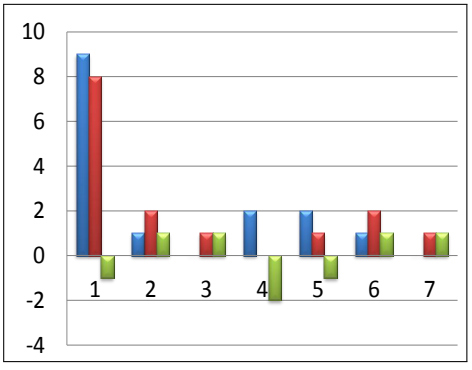
(b) Ich finde das Lernen hier richtig spannend.



(c) In meinem Studienfach will ich wirklich kompetenter werden und mich weiterentwickeln.



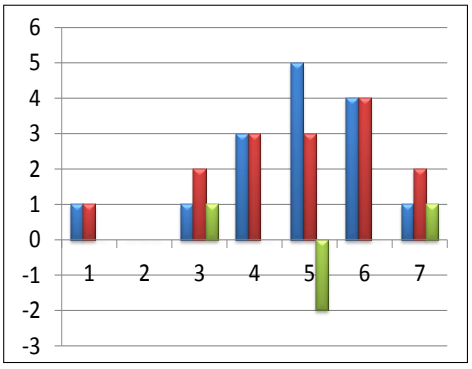
(d) Ich setze mich selbst unter Druck, um die Arbeiten im Studium zu erledigen.



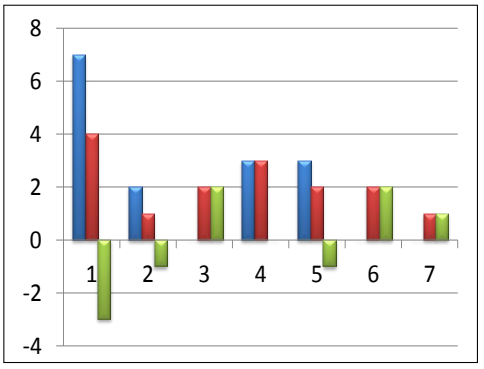
(e) Ich frage mich oft, was ich an der Universität eigentlich soll.

(f) Legende

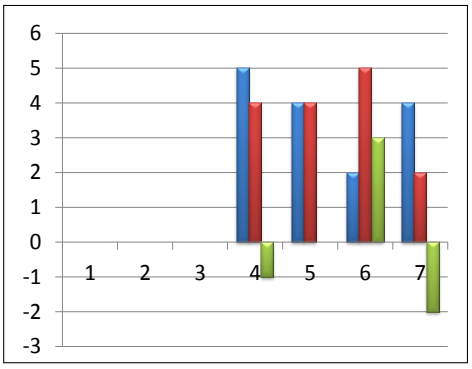
Abbildung E.34.: Zufriedenheit mit dem Studium



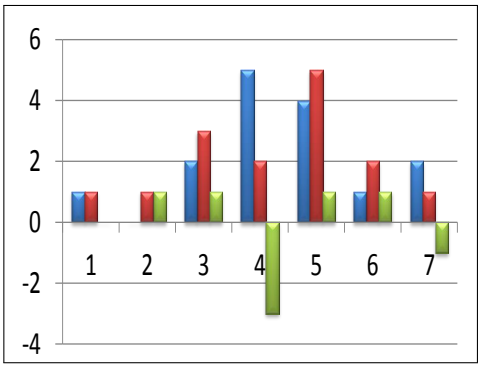
(a) Ich muss mich zum Lernen zwingen.



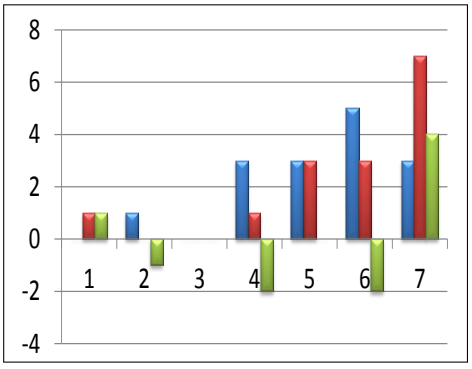
(b) Ohne Druck von außen würde ich weniger tun.



(c) Ich genieße es richtig, hier zu lernen und zu arbeiten.



(d) Ich erledige die anfallende Arbeit im Studium, weil ich sonst ein schlechtes Gewissen hätte.

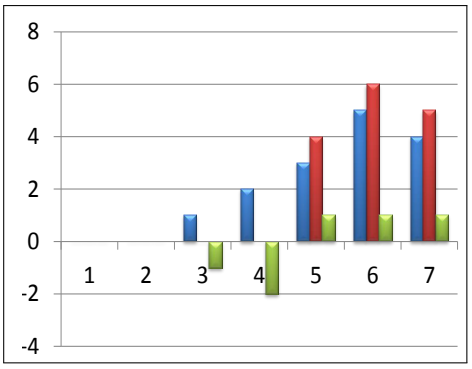


(e) Ich setze mich in meinem Studium ein, um selbstgesteckte Ziele verwirklichen zu können.

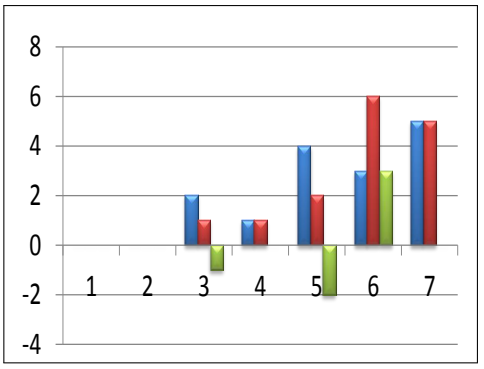
(f) Legende

Abbildung E.35.: Zufriedenheit mit dem Studium (Fortsetzung)

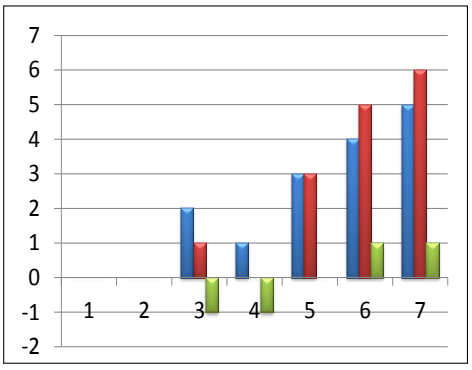
E.2.3. Lernen im Studium



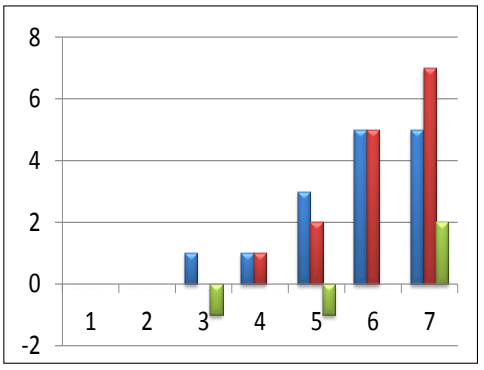
(a) Ich bin in der Lage meine Ziele im Studium zu erreichen.



(b) Ich fühle mich der Herausforderung gewachsen gut im Studium abzuschneiden.



(c) Ich bin in der Lage den Stoff des Studiums zu lernen.

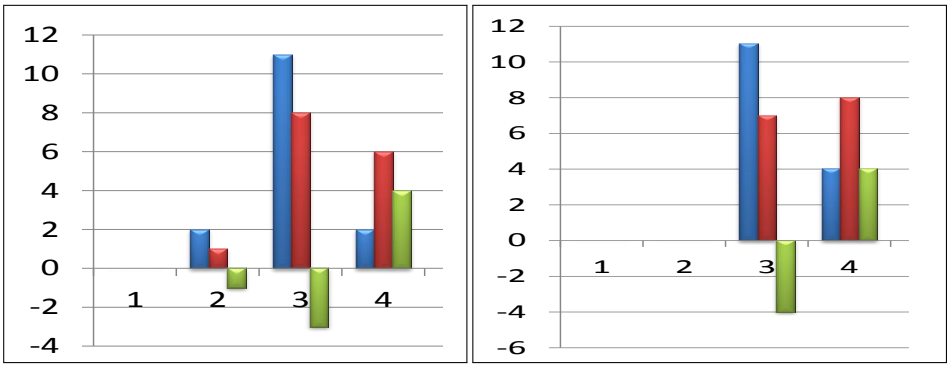


(d) Ich vertraue in meine Fähigkeiten den Stoff zu lernen.

(e) Legende

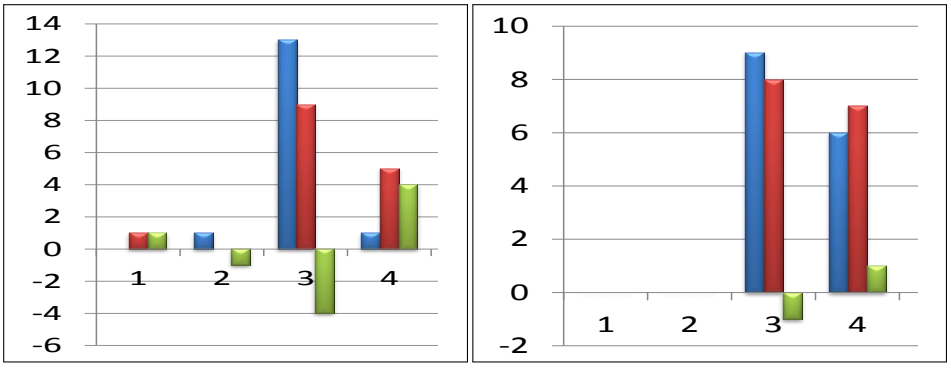
Abbildung E.36.: Lernen im Studium

E.2.4. Zuversicht gegenüber Herausforderungen im Studium



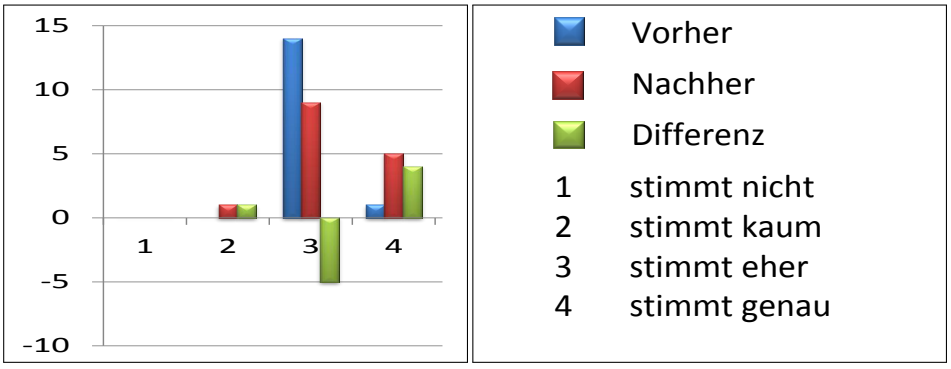
(a) Es bereitet mir keine Schwierigkeiten, meine Absichten und Ziele zu verwirklichen.

(b) Was auch immer passiert, ich werde schon klarkommen.



(c) Wenn eine neue Sache auf mich zukommt, weiß ich, wie ich damit umgehen kann.

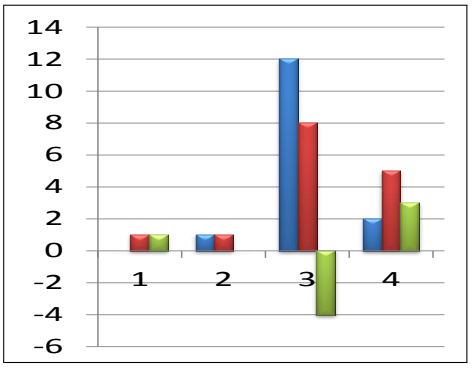
(d) Schwierigkeiten sehe ich gelassen entgegen, weil ich meinen Fähigkeiten immer vertrauen kann.



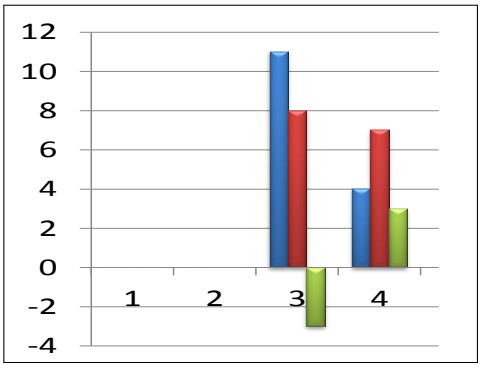
(e) Wenn ein Problem auftaucht, kann ich es aus eigener Kraft meistern.

(f) Legende

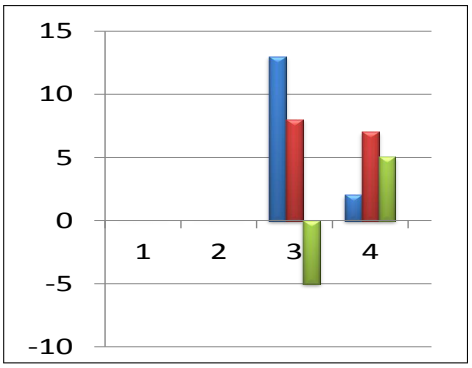
Abbildung E.37.: Zuversicht gegenüber Herausforderungen im Studium



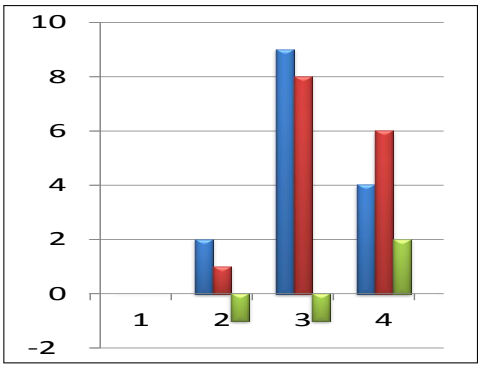
(a) In unerwarteten Situationen weiß ich immer, wie ich mich verhalten soll.



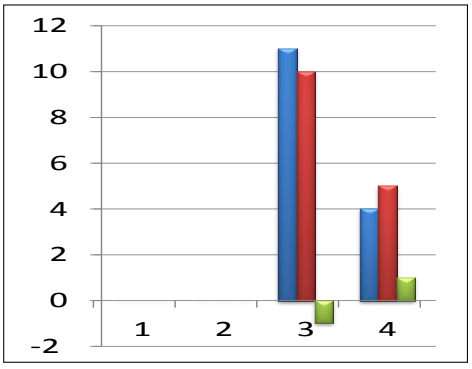
(b) Die Lösung schwieriger Probleme gelingt mir immer, wenn ich mich darum bemühe.



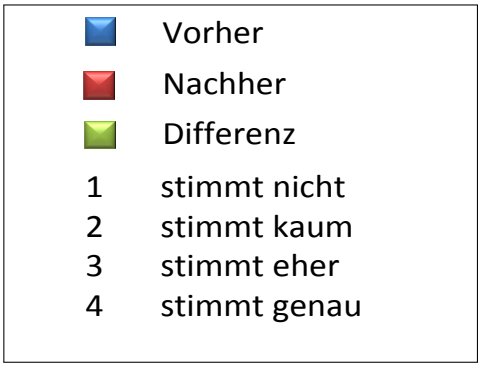
(c) Auch bei überraschenden Ereignissen glaube ich, dass ich gut mit ihnen zurechtkommen kann.



(d) Für jedes Problem kann ich eine Lösung finden.



(e) Wenn sich Widerstände auftun, finde ich Mittel und Wege, mich durchzusetzen.



(f) Legende

Abbildung E.38.: Zuversicht gegenüber Herausforderungen im Studium (Fortsetzung)

E.2.5. Schulkenntnisse in der Informatik

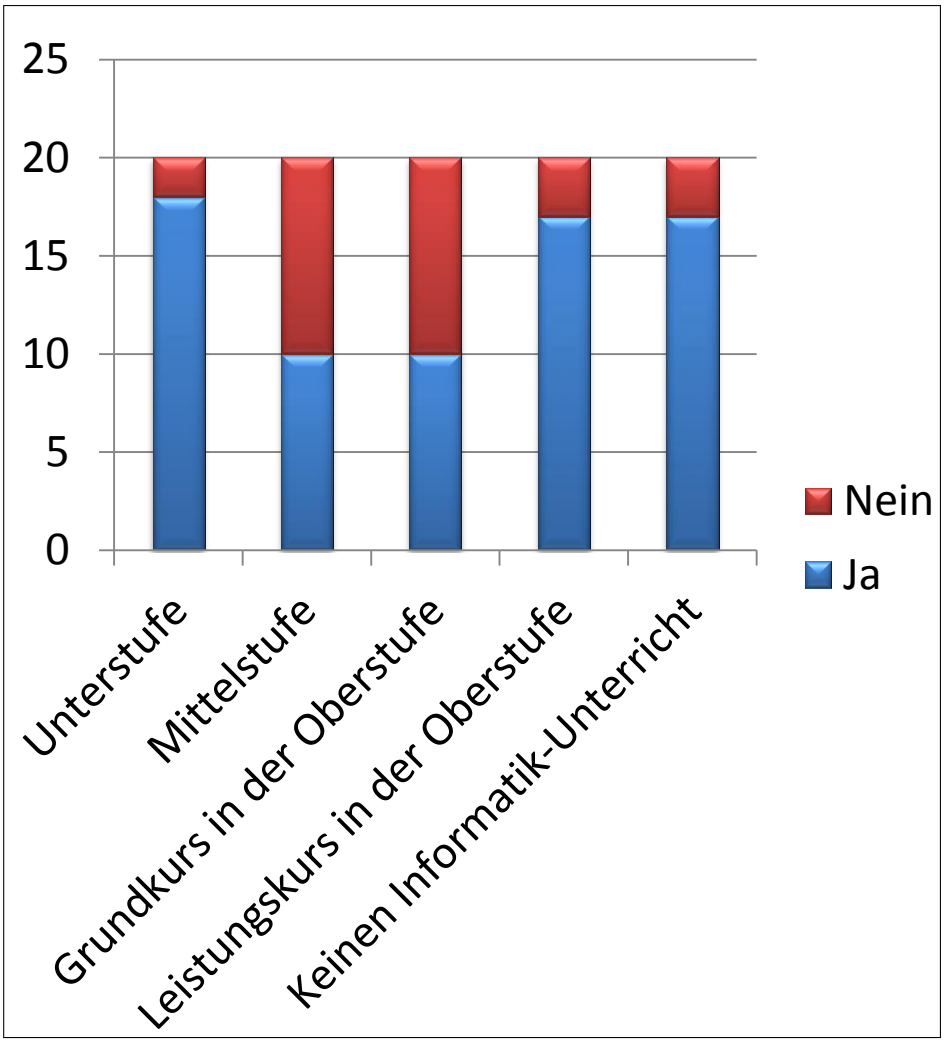


Abbildung E.39.: Hattest du in der Schule Informatik-Unterricht?

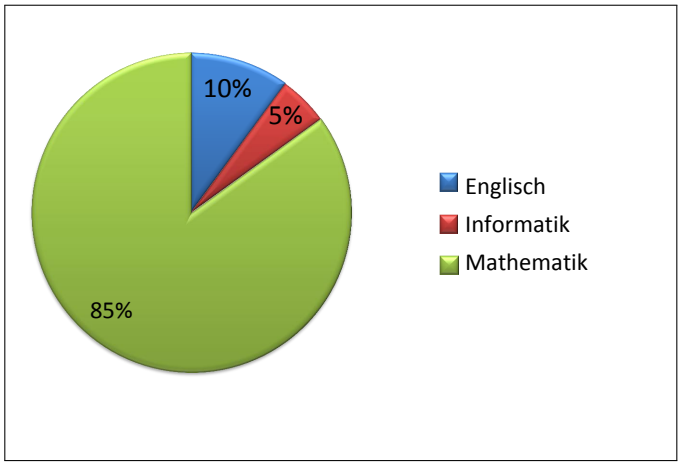


Abbildung E.40.: Erster Leistungskurs

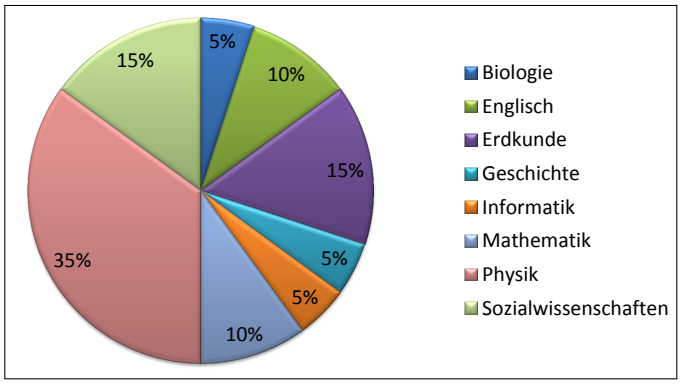


Abbildung E.41.: Zweiter Leistungskurs

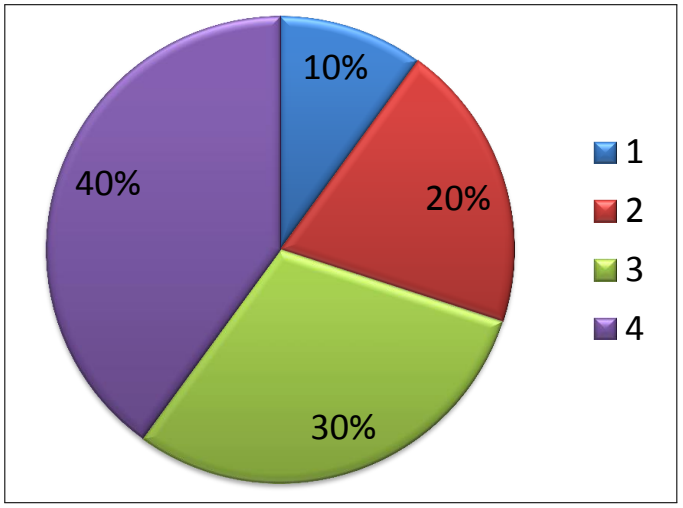
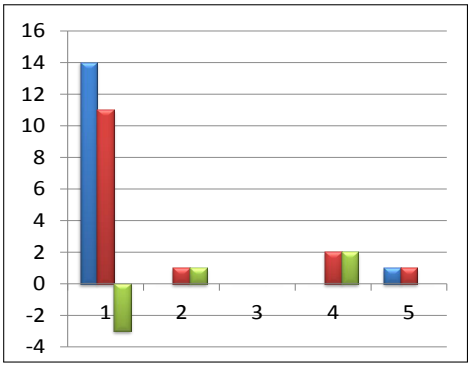
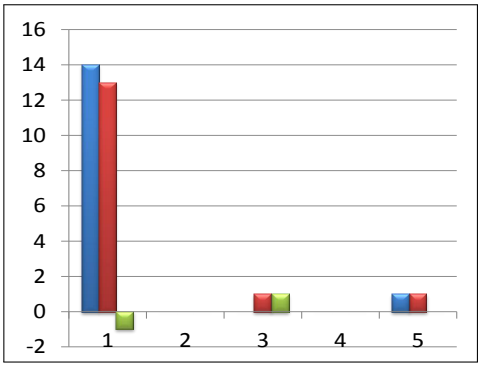


Abbildung E.42.: Abiturnote (Gerundet auf ganze Noten)

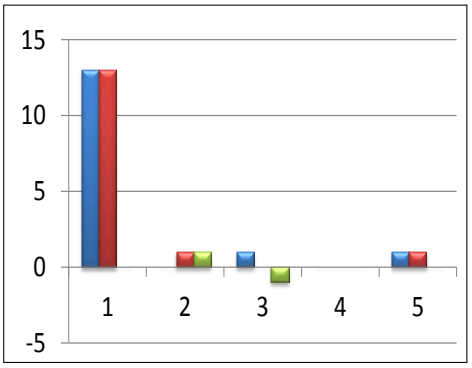
E.2.6. Fachliche Kenntnisse



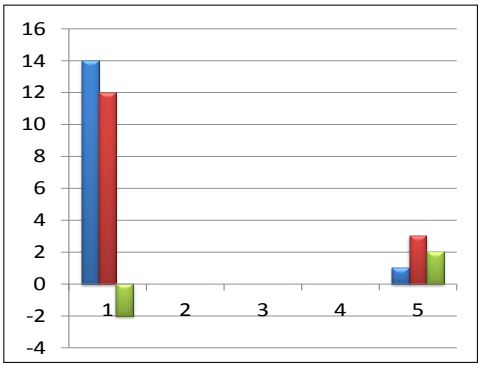
(a) Welcher der folgenden Java Quellcode Ausschnitte ist korrekt?



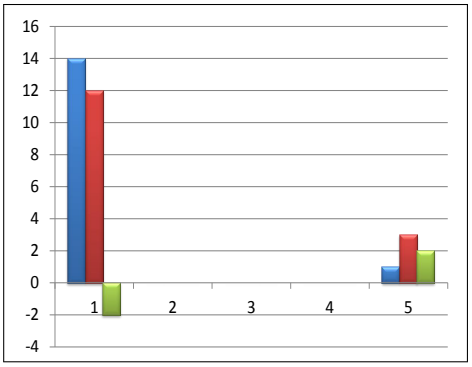
(b) Welcher der folgenden Java Quellcode-Ausschnitte ist korrekt?



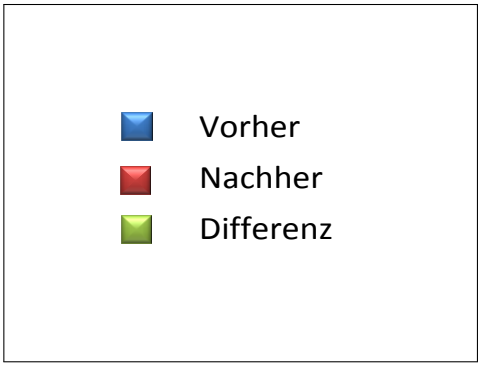
(c) Welcher der folgenden Ausdrücke ergibt „true“, wenn x den Wert 1 y den Wert 2 z den Wert 3 hat?



(d) Welche der folgenden Java Methoden ist falsch?



(e) Welche der genannten Aussagen beschreiben einen guten Algorithmus?



(f) Legende

Abbildung E.43.: Fachliche Kenntnisse

E.2.7. Bewertung des Workshops

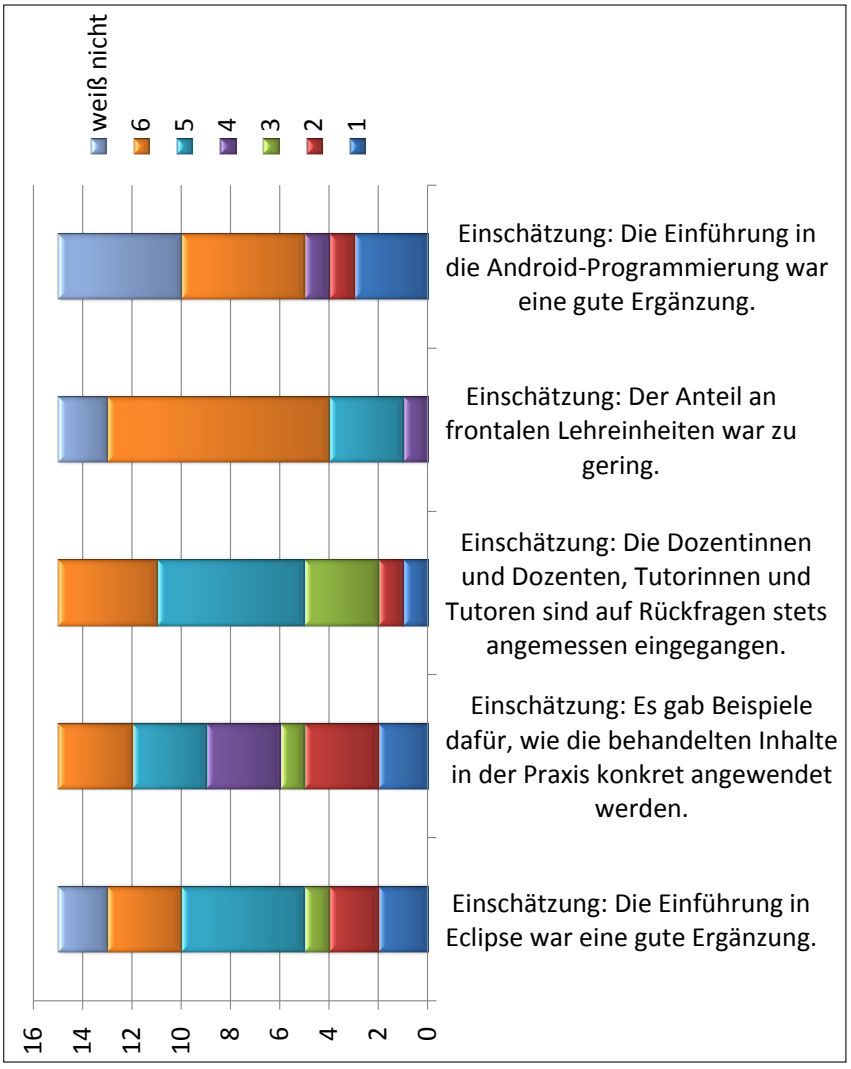


Abbildung E.44.: Auswertung einzelner Abschnitte des Workshops

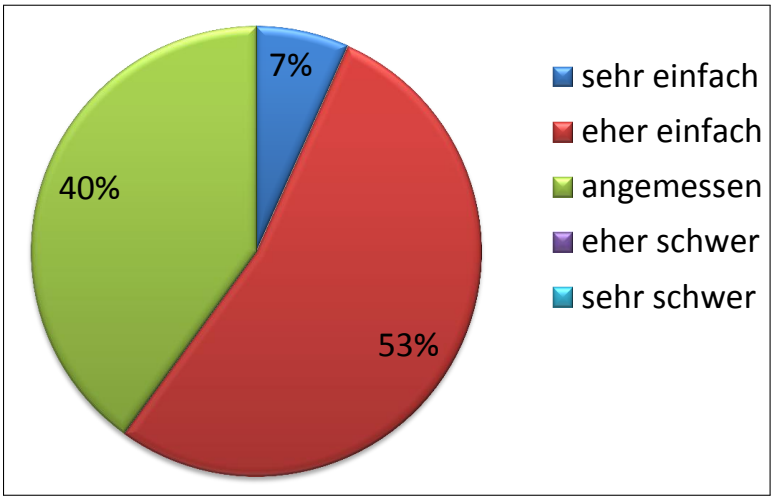


Abbildung E.45.: Ich fand den Schwierigkeitsgrad des Workshops

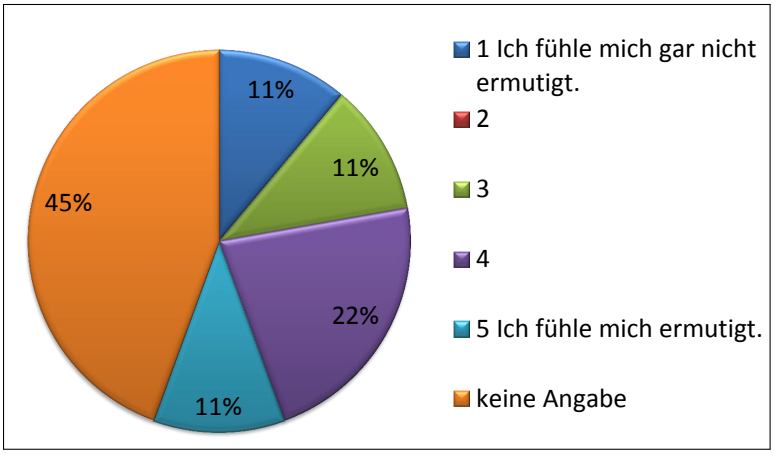


Abbildung E.46.: Fühlen Sie sich nach dem Workshop ermutigt bezüglich ihrer Studienwahl?

Begründung der Einschätzung[13]:

- Gutes Endprodukt, wenig eigenes Denken.
- Ja, da es mir Spaß gemacht hat, Algorithmen für die Probleme zu verfassen und auch in Aktion sehen zu können.
- Keinen wesentlichen Einfluss auf Einstellung zur Studienwahl.

[13]Es wurden vereinzelte orthographische Korrekturen an den Begründungen vorgenommen.

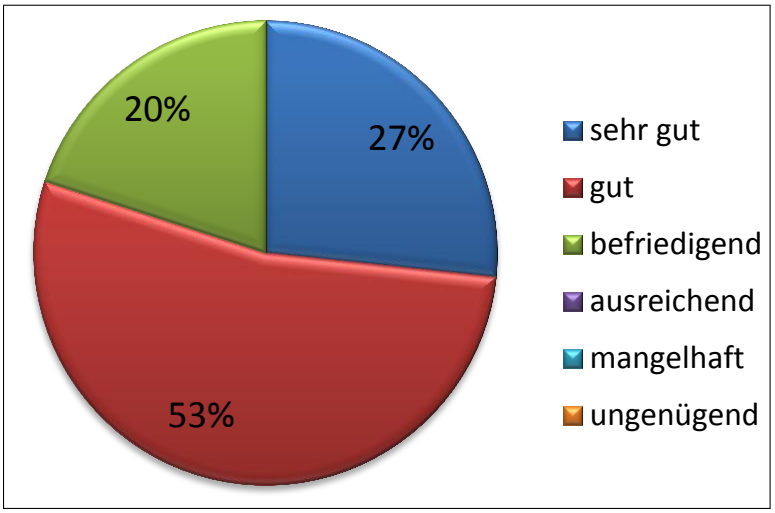


Abbildung E.47.: Benotung des Workshops

Bitte begründen Sie Ihre Einschätzung[14]:

- Ausführlichere Formulierung der Texte.
- Der Workshop hat meistens nur Dinge aus der Progra- und DSAL-Vorlesung aufgearbeitet, als Neues beigebracht.
- Entweder stärkeren Schwerpunkt auf Android Programmierung legen oder dies nicht vorankündigen.
- Etwas mehr „Android“, weniger Java an sich.
- Grundlagen zu API's usw. für Android würden mir Spaß machen.
- Mehrtägiger Kurs mit den Grundlagen bezüglich Android oder DSAL.

[14]Es wurden vereinzelte orthographische Korrekturen an den Begründungen vorgenommen.

E.2.8. Leistung der Studenten im Workshop

Arbeitsauftrag	Maximale Punktzahl	Gruppe 1	Gruppe 2	Gruppe 3	Gruppe 4	Gruppe 5	Gruppe 6	Gruppe 7	Gruppe 8	Gruppe 9	Gruppe 10
3 (Initialisierung)	1	1	1	1	1	1	1	1	1		1
3a	1	1	1	1	1	0,5	0,5	1	0,5	1	1
3b	1	1	1	1	1	1	1	0,5	0,5	1	1
3c	1	1	1	0,5	0,5	1	1	1	0,5	1	0,5
3d	1	1	1	1	1	1	1	1	0,5	1	1
3e	1	1	1	1	1	1	1	1	1	1	1
3f	3	3	3	2	3	3	3	2,5	3	1	1
5a	5	5	5	5	5	5	5	5	5	5	5
5b	1	1	1	1	1	1	1	1	1	1	1
6 (Initialisierung)	1	1	1	1	1	1	1	1	1	1	1
6a	1	1	1	0,5	1	1	1	1	1	1	1
6b	1	1	1	0	0	1	0,5	1	1	0	0
6c	1	0,5	1	1	1	1	1	1	0,5	1	1
6d	2	1	1	1	1	1	1	2	1	1	1
6e	2	1	1	1	1	1	0,5	1	2	1	1
6f	1	1	1	1	1	1	1	1	1	1	1
7	6	6	5	6	2	6	4	6	6	4	5
8	6	6	4	5	0	6	6	6	6	6	5
9	7	7	3	0	0	4	0	6	7	7	0
Optimierung	7	7	1	0	0	0	0	1	7	7	0
Summe	50	47,5	35	30	22,5	37,5	30,5	41	46,5	42	28,5

Summe

Arbeitsauftrag	Maximale Punktzahl	Gruppe 1	Gruppe 2	Gruppe 3	Gruppe 4	Gruppe 5	Gruppe 6	Gruppe 7	Gruppe 8	Gruppe 9	Gruppe 10
Arbeitsauftrag 3	9	9	9	7,5	8,5	8,5	8,5	8	7	6	6,5
Arbeitsauftrag 5	6	6	6	6	6	6	6	6	6	6	6
Arbeitsauftrag 6	9	6,5	7	5,5	6	7	6	8	7,5	6	6
Arbeitsauftrag 7	6	6	5	6	2	6	4	6	6	4	5
Arbeitsauftrag 8	6	6	4	5	0	6	6	6	6	6	5
Arbeitsauftrag 9	7	7	3	0	0	4	0	6	7	7	0
Optimierung	7	7	1	0	0	0	0	1	7	7	0

Abbildung E.48.: Punkte der Studenten in den Teilaufgaben

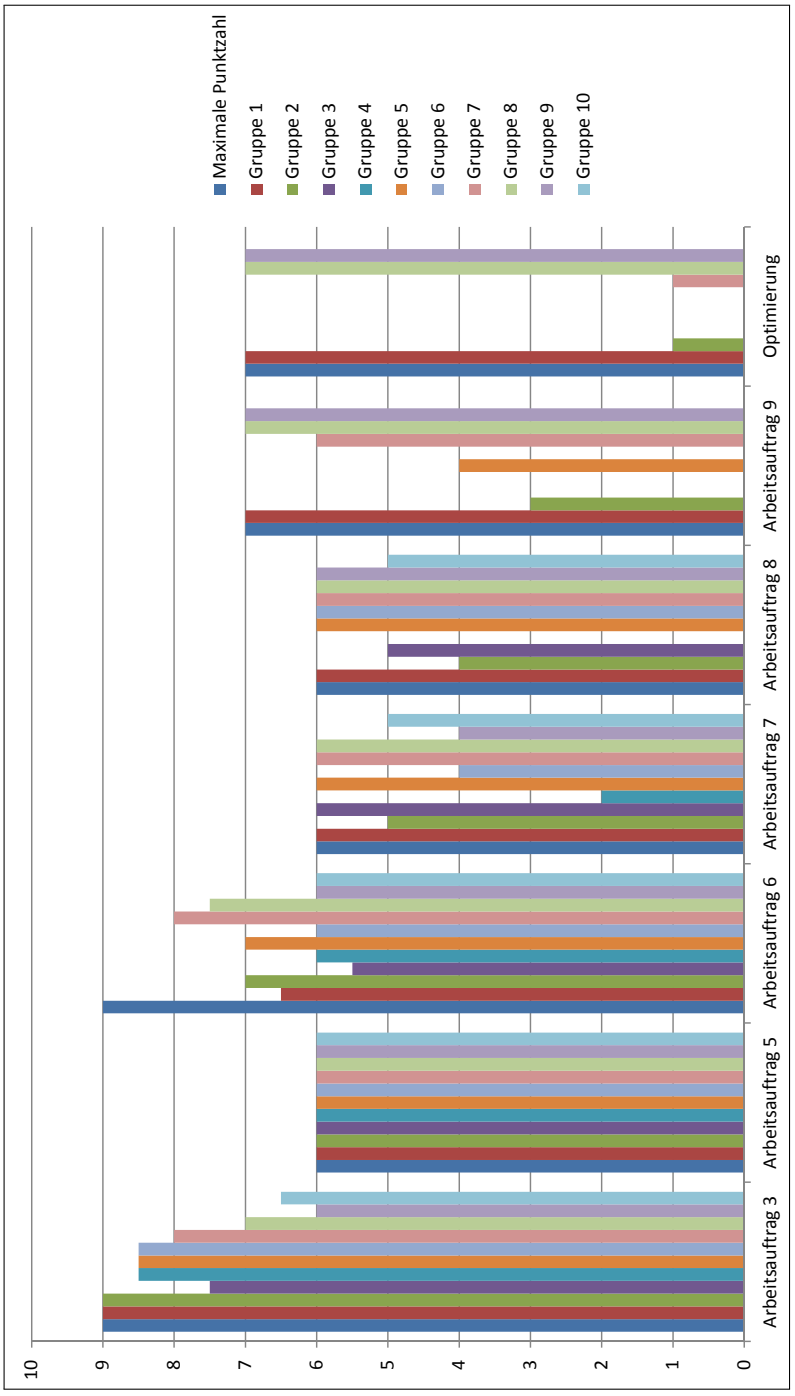


Abbildung E.49.: Punkte nach Arbeitsaufträgen ausgewertet

F. Erklärung

Ich versichere, dass ich die schriftliche Hausarbeit - einschließlich beigefügter Zeichnungen, Kartenskizzen und Darstellungen - selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Alle Stellen der Arbeit, die dem Wortlaut oder dem Sinn nach anderen Werken entnommen sind, habe ich in jedem Fall unter Angabe der Quelle deutlich als Entlehnung kenntlich gemacht.

Datum: 14. August 2013

Unterschrift:

Manuel Froitzheim