**Kurzfassung**  
Für jugendliche Sportschützen ist es in einem Wettbewerb schwierig, zu beurteilen, wie viele  
Ringe beim nächsten Schuss zu schießen sind.  
Die Suche nach einem Trainingsprogramm, welches den Schützen im Training dabei hilft, dies  
zu üben und zu lernen, verlief erfolglos.  
Deshalb wurde in dieser Jugend forscht- Arbeit das Programm Chancenrechner entwickelt. Es  
handelt sich dabei um ein Programm, welches Sportschützen im Training unterstützt. Es gibt  
ihnen Empfehlungen für die zu schießende Ringzahl beim nächsten Schuss in Form einer Minimalringzahl und einer empfohlenen Ringzahl. Außerdem enthält das Programm einen Timer,  
der den Schützen dabei hilft, ein besseres Zeitgefühl zu entwickeln.  
Auch für den Trainer stellt das Programm eine Hilfe dar, da es automatisch eine Schussserie  
im Training auswertet und das Ergebnis protokolliert.  
Das Programm wurde in Zusammenarbeit mit einem Informatiklehrer und dem Trainer eines  
lokalen Schützenvereins entwickelt und in diesem Verein getestet.  
1  
**Inhaltsverzeichnis**  
**1 Einleitung 1**  
**2 Dokumentation der Entwicklung des Programms 3**  
2.1 JavaFX als die genutzte Programmiersprache . . . . . . . . . . . . . . . . . . . . 3  
2.2 Dokumentation des Hauptalgorithmus’: Berechnung der Schussempfehlungen . 4  
2.2.1 Grundidee hinter dem Algorithmus . . . . . . . . . . . . . . . . . . . . . . 4  
2.2.2 Teil 1 des Algorithmus: Variableninitialisierungen und nötige Festlegungen 6  
2.2.3 Die grundlegende Arbeitsweise des Algorithmus . . . . . . . . . . . . . . 7  
2.2.4 Die Generierung der Empfehlungsausgabe . . . . . . . . . . . . . . . . . 10  
2.3 Umsetzung in JavaFX . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11  
2.3.1 prinzipielle Umsetzung der grafischen Benutzeroberfläche (GUI) und der  
Ein- und Ausgabe . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11  
2.3.2 Umsetzung wichtiger Algorithmen . . . . . . . . . . . . . . . . . . . . . . 12  
2.4 Beschreibung des Arbeitsprozesses und der Programmentwicklung . . . . . . . 15  
**3 Bewertung des fertigen Programms 16**  
3.1 Auswertung der Tests . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 16  
3.2 Vergleich des fertigen Programms mit seinen Zielen . . . . . . . . . . . . . . . . 17  
**4 Literaturverzeichnis 18**  
**5 Anhang 19**  
5.1 Übersicht über die Variablen im Hauptalgorithmus und ihre Funktionen . . . . . . 19  
5.2 Testprotokolle . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 20  
5.2.1 Testprotokoll vor dem dritten Treffen . . . . . . . . . . . . . . . . . . . . . 20  
5.2.2 Testprotokoll nach dem dritten Treffen . . . . . . . . . . . . . . . . . . . . 21  
**Abbildungsverzeichnis**  
1 Beispielschussfolge eines Schützen . . . . . . . . . . . . . . . . . . . . . . . . . 1  
2 Das Logo von JavaFX . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3  
3 Teil 1 des Hauptalgorithmus’ als Struktogramm . . . . . . . . . . . . . . . . . . . 6  
4 Teil 2 des Hauptalgorithmus’ als Struktogramm . . . . . . . . . . . . . . . . . . . 7  
5 Teil 3 des Hauptalgorithmus’ als Struktogramm . . . . . . . . . . . . . . . . . . . 10  
6 Die GUI des Programms „Chancenrechner“ unter Windows 8 . . . . . . . . . . . 11  
7 Illustration der Herleitung der Berechnung des Abstandes eines Schusses von  
der Mitte . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12  
**Tabellenverzeichnis**  
1 Übersicht über die Variablen im Hauptalgorithmus und ihre Funktionen . . . . . . 19  
**1 Einleitung**  
Ich, Eric Ackermann, wurde von meinem Informatiklehrer, Herrn Thiele, angesprochen, ob ich  
für einen Freund, der einen Schützenverein trainiert, ein Trainingsprogramm finden und ggf.   
installieren und einrichten könne.  
Für einen Sportschützen ist der Wettkampf eine schwierige und anstrengende Situation.  
Einerseits ist der Wettkampf eine hohe sportliche Herausforderung, dessen Ausgang durch  
unzählige Details wie den korrekten Stand oder die richtige Atemtechnik beeinflusst wird. Auch  
das Treffen der Zielscheibe möglichst nah am Zentrum ist eine schwierige Aufgabe.  
Jedoch gibt es auch eine nicht zu unterschätzende geistige Komponente.  
Nehmen wir an, ein Schütze soll in einem Wettkampf bei einer Serie von 10 Schuss 88 Ringe1  
erzielen, d.h. die Summe seiner Treffer soll mindestens 88 sein. Nun schießt er mit seinen  
ersten vier Schuss folgendermaßen:  
Abbildung 1: Beispielschussfolge eines Schützen  
Die Frage, die jeder Schütze sich im Wettkampf bei jedem Schuss stellt, ist folgende: Welche  
Ringzahl muss ich mindestens treffen, um mein Ziel noch erreichen zu können?  
Natürlich könnte ein Schütze die Frage umgehen, indem er bei jedem Schuss versucht, die  
Optimalzahl von 10 Ringen zu schießen. Jedoch würde dies den Schützen unter viel zu hohen  
Druck setzen, da er nicht immer eine 10 schießen muss. Um in diesem Beispiel die 88 Ringe  
zu erreichen, genügt ein Durchschnitt von 8,8 Ringen pro Schuss, er muss also nicht bei jedem  
Schuss die 10 treffen. Dies ist zudem in der Praxis schwer möglich. Der Schütze würde irgendwann bei einem der Schüsse niedriger als 10 schießen und vielleicht frustriert aufgeben, weil  
er denkt, er könnte sein Ziel nicht mehr erreichen. Im Beispiel würde ein Schütze, der so denkt,   
also versuchen, bei den nächsten 6 Schüssen jedes Mal eine 10 zu schießen. Würde ihm dies  
gelingen, erreichte er 89 Punkte statt der anvisierten 88 und fragt sich zu Recht, warum er sich  
1Der Wert des aktuellen Schusses wird in Ringen gemessen. Die Zielscheibe, auf die geschossen wird, besteht  
aus 10 Kreisen, die immer den gleichen Abstand voneinander haben, den sogenannten Ringen. Trifft ein Schuss  
die Scheibe nicht, wird er mit 0 Ringen gewertet, trifft er zwischen dem äußersten und dem zweitäußersten Ring,  
wird er als 1 gewertet usw.  
1  
selbst unter den hohen Druck gestellt und bei jedem Schuss versucht hat, eine 10 zu treffen.  
Diese Situation kann mit einigen Berechnungen vermieden werden.  
Im Beispiel sind mit 6 Schuss noch 59 Ringe zu treffen. Dividiert man nun 59 durch 6, erhält  
man aufgerundet 10. Daraus ergibt sich die Empfehlung für die nächsten 6 Schüsse: möglichst  
jedes Mal 10 Ringe schießen.  
Jedoch kann man mit den fünf übernächsten Schüssen, wenn man jedes Mal die Höchstringzahl von 10 Ringen schießt, maximal 50 Ringe erzielen. Da noch 59 Ringe zu schießen sind,   
ist der minimal beim nächsten Schuss zu erreichende Wert 9.  
Ein Schütze, der dies gerechnet hat, weiß nun: er muss beim nächsten Schuss auf jeden Fall  
eine 9 schießen, versucht aber, die 10 zu treffen. Damit setzt er sich unter geringeren Druck,   
als wenn er versuchen würde, bei jedem folgenden Schuss eine 10 zu schießen, und hat so  
die größere Erfolgsaussicht.  
Die mentale Schwierigkeit für einen Schützen im Wettkampf besteht also darin, diese Überlegungen ohne Taschenrechner unter Zeitdruck bei jedem Schuss neu auszuführen.  
Die Suche nach einem geeigneten Programm, welches Schützen bei diesen Überlegungen  
entlastet, verlief erfolglos. Weder für Windows-PC’s noch für Android-Smartphones ließ sich  
ein adäquates Programm finden.  
Dadurch entstand die Idee, ein solches Programm zu entwickeln, woraus schließlich diese  
Arbeit entstand.  
Da Hilfsmittel im Wettkampf nicht zugelassen sind, soll das Programm den Schützen also im  
Training unterstützen.  
Das Programm soll, während der Schütze unter Wettkampfbedingungen schießt, für ihn nützliche Informationen ausgeben, zum Beispiel die minimal nötige und die empfohlene Ringzahl für  
den nächsten Schuss, außerdem soll er eine Übersicht über die bisherige Leistung in Form einer bisherigen Gesamtringzahl sowie der Information, wo genau die letzten Schüsse getroffen  
haben, erhalten. Außerdem soll der Schütze jederzeit genau wissen, wie viel Zeit ihm für seine  
Schüsse noch bleibt.  
Ziel ist, dass der Schütze sich durch das Training mit dem Programm an die Wettkampfsituation  
gewöhnt, dass er lernt, selbst zu entscheiden, wie viele Ringe er beim nächsten Schuss zu  
treffen versucht, sowie, dass der Schütze ein gutes Zeitgefühl entwickelt, sodass diese Dinge  
im echten Wettkampf ohne das Programm automatisiert ablaufen.  
Dafür muss das Programm natürlich mit den Wettbewerbsformalismen, zum Beispiel den festen  
Zuordnungen des Zeitlimits zur Schusszahl oder der Zuordnung der Waffe zur Zielscheibe,   
kompatibel sein. Dabei muss natürlich jede beliebige Schuss- und Zielringzahl und möglichst  
jede Waffe unterstützt werden.  
Zur Unterstützung des Trainers des Schützen soll zudem ein detailliertes Schussprotokoll generiert werden, welches ihm ermöglicht, Schwachstellen gemeinsam mit dem Schützen zu finden  
und an ihrer Beseitigung zu arbeiten und aktuelle mit älteren Trainingsverläufen vergleichen zu  
können.  
2  
**2 Dokumentation der Entwicklung des Programms**  
**2.1 JavaFX als die genutzte Programmiersprache**  
Abbildung 2: Das Logo von JavaFX  
2  
JavaFX ist eine Spezifikation der Programmiersprache Java. Sie ist in der aktuellen Version  
besonders auf grafische Oberflächen ausgelegt, die mit mehr Designmöglichkeiten kompatibel sind als in früheren Spezifikationen von Java (Swing, AWT). JavaFX wird in der aktuellen  
Version 2.x hauptsächlich für anspruchsvolle grafische Oberflächen wie zum Beispiel bei Videospielen verwendet.  
Java FX wurde aus folgenden Gründen als Programmiersprache für dieses Projekt genutzt:  
*•* JavaFX hat ein moderneres Aussehen als die älteren Spezifikationen AWT und Swing.  
*•* JavaFX ist als modernste Java-Spezifikation zukunftssicher.  
*•* JavaFX ist hervorragend in die benutzte Entwicklungsumgebung NetBeans IDE 8.0.2 integriert, was die Entwicklung erleichtert.  
Zudem hat die Verwendung einer Java-Spezifikation noch einige allgemeine Vorteile:  
*•* Java ist auf möglichst einfache Entwicklung von Programmen ausgelegt.  
*•* Java selbst bringt sehr viele Klassen und ihre Methoden mit, die oft die Entwicklung beschleunigen.  
*•* Java wird in einem Interpreter auf fast jedem Betriebssystem ausgeführt, d.h. ein JavaProgramm läuft auf Windows, Linux, Solaris... überall gleich und muss für alle Systeme  
nur einmal entwickelt werden.  
*•* Java kann auf Milliarden von Geräten bereits ausgeführt werden und ist kostenlos und  
leicht zu installieren.  
2  
http://code.makery.ch/assets/library/javafx-2-tutorial/javafx-logo.png : 08.12.15, JavaFX-Logo  
als PNG-Bild  
3  
**2.2 Dokumentation des Hauptalgorithmus’: Berechnung der Schussempfehlungen**  
Im Folgenden wird der Hauptalgorithmus des Programms in Struktogrammen dargestellt und  
schrittweise erklärt. Verschiedene Arten des auszuführenden Codes wurden farblich hervorgehoben:  
*•* Lachsrot markiert sind für den Programmablauf nötige Festlegungen, zum Beispiel Variableninitialisierungen.  
*•* Gelb markiert sind Ausgaben des Algorithmus’.  
*•* Hellgrün markiert sind die Teile des Hauptalgorithmus’, die ausgeführt werden, wenn weniger Ringe als im Durchschnitt nötig geschossen wurden.  
*•* Türkisblau markiert sind die Teile des Hauptalgorithmus’, die ausgeführt werden, wenn  
mehr Ringe als im Durchschnitt nötig geschossen wurden.  
*•* Pink markiert sind die Teile des Hauptalgorithmus’, die standardmäßig ausgeführt werden  
müssen.  
**2.2.1 Grundidee hinter dem Algorithmus**  
Um n Ringe bei m Schüssen zu erzielen, ist der nötige Durchschnittswert (im Algorithmus  
„optimum“ genannt) immer *m n* 3.  
Also ist die grundlegende Idee, die der Algorithmus verfolgt, den Schützen im Durchschnitt zu  
diesem Wert zu führen.  
Zuerst wird (vor Aufruf des Algorithmus’) dem Schützen also empfohlen, im Durchschnitt diesen  
Wert zu schießen. Die aktuelle Empfehlung ist also optimum. Wird nun ein Schussresultat  
eingegeben, gibt es drei Möglichkeiten:  
1. Der Schuss entspricht dem aktuell anvisierten Durchschnittswert.  
2. Der Schuss entspricht einer höheren...  
3. ... oder einer niedrigeren Ringzahl.  
Im Fall 1 gibt das Programm weiter die aktuelle Empfehlung aus.  
In den Fällen 2 und 3 muss die aktuelle Empfehlung angepasst werden. Wie genau das geschieht, zeigen die Hellgrün und die Türkisblau markierten Teile der Struktogramme, die Grundidee ist folgende:  
Die Summe zweier Aufeinanderfolgender Schussergebnisse muss doppelt so groß sein wie  
optimum, um einen Durchschnitt von optimum zu erreichen. Also wird die Abweichung vom  
aktuellen Schuss zum Optimum ermittelt, die Empfehlung für den nächsten Schuss ist dann  
optimum plus oder minus diese Abweichung.  
3Dieser Quotient muss immer aufgerundet werden, um niemals Empfehlungen zu geben, die zu einer zu geringen Ringzahl führen.  
4  
Im Fall, dass Optimum plus oder minus diese Abweichung aus dem bei einem Schuss möglichen Bereich zwischen 10 und 0 Ringen gerät, muss optimum neu berechnet und als aktuelle  
Empfehlung ausgegeben werden.  
Zusätzlich gibt der Algorithmus bei jedem Schuss noch zur Orientierung den letzten Schuss  
und die minimal zu schießende Ringzahl *verbleibendeRinge−*(*verbleibendeSchuesse −*1)*∗*104  
(wenn diese nicht 0 ist) aus.  
4Herleitung dieser Formel siehe Einleitung.  
5  
**2.2.2 Teil 1 des Algorithmus: Variableninitialisierungen und nötige Festlegungen**  
Anmerkung: eine Übersicht über alle Variablen in der Reihenfolge ihres Auftretens sowie deren  
Funktion befindet sich im Anhang.  
Abbildung 3: Teil 1 des Hauptalgorithmus’ als Struktogramm  
Der Hauptalgorithmus wird bei jedem Schuss mit dessen aktuellen Wert neu aufgerufen.  
Das Array von 11 Integer-Zahlen (Ganzzahlen von 0 bis 10) „zahlen“ speichert alle Empfehlungen, wobei der jeweilige Index die Schussempfehlung und der Wert im Array dessen Anzahl  
ist. Wenn also *zahl*[5] = 2 empfiehlt das Programm, mit den restlichen Schüssen zweimal 5  
Ringe zu treffen.  
Beim ersten Aufruf des Algorithmus’ in der aktuellen Schussserie muss das Array so initialisiert werden, dass es empfiehlt, bei jedem Schuss den Optimalwert zu schießen. Also wird  
*zahl*[*optimum*] = *schuesse* gesetzt.  
Der Wahrheitswert „frischgenullt“ muss auf falsch gesetzt werden, weil das Optimum in diesem  
Durchlauf nicht neu berechnet wurde.  
Der Wahrheitswert „falscheingabe“ ist algorithmisch nicht bedeutend, hier aber erwähnt.  
Bei jedem Schuss muss der Gesamtringzahl „gesamt“ noch der Wert des aktuellen Schusses  
hinzuaddiert werden. „gesamt“ wird vor jeder Schussserie auf 0 gesetzt.   
„letzter“ wird bei jedem Durchlauf auf den Wert des zuletzt eingegebenen Schusses, „aktuell“,   
gesetzt.  
Das Array von Integer-Zahlen „ergebnisse“ speichert die einzelnen Schussergebnisse, um spä-  
ter statistische Erweiterungen wie z.B. eine Trendanalyse hinzufügen zu können. Jeder neu  
eingegebene Schuss wird dazu in der Integer-Zahl mit dem Index seiner Nummer minus 1  
gespeichert (Java-Indizes zählen von 0 bis Anzahl-1).  
Der Algorithmus wird nur ausgeführt, wenn man seine Zielringzahl noch nicht erreicht hat, also  
wird dies vor jedem Durchlauf erst einmal geprüft.  
6  
**2.2.3 Die grundlegende Arbeitsweise des Algorithmus**  
Abbildung 4: Teil 2 des Hauptalgorithmus’ als Struktogramm  
7  
Bei jedem Schuss werden dem Schützen zur besseren Orientierung zu seinem Ziel hin seine  
Gesamtringzahl und sein letzter Schuß ausgegeben.  
Für alle Empfehlungsberechnungen gilt: Diese werden nur ausgeführt, wenn das Ziel nicht  
mehr zu erreichen ist oder bereits erreicht wurde (Wahrheitswert „nichtmehrzuerreichen“), was  
nur Programmabstürze oder unausführbare Empfehlungen (z.B. 12 Ringe zu schießen) zur  
Folge hätte, und es sich nicht um den letzten Schuss handelt, wo Empfehlungen nicht mehr  
sinnvoll wären.  
Wird dies erfüllt, wird der Wahrheitswert „nurabgearbeitet“, welcher beschreibt, dass eine empfohlene Ringzahl geschossen wurde, auf „falsch“ gesetzt, da dies später noch überprüft werden  
muss.  
Danach wird geprüft, ob weniger Ringe geschossen wurden, als „optimum“ vorgibt. Nehmen  
wir an, dies sei gegeben, wird das Kernstück des Hauptalgorithmus ausgeführt:  
1. Die Variable „Guthaben“ speichert die Differenz der Ringe, die man erhält, wenn man den  
vorgeschlagenen Durchschnitt erreicht, und der Zielringzahl. Ist „Guthaben“ größer oder  
gleich der aktuellen Abweichung, wird „Guthaben“ um diese verringert. Damit wird realisiert, dass die Abweichung ausgeglichen und der Schuss wie ein Schuss, der „optimum“  
Ringe getroffen hat, behandelt wird.  
2. Ist Guthaben kleiner als die Abweichung von Optimum, dann wird geprüft, ob die aktuelle  
Ringzahl empfohlen war. Falls ja, wird die Integer-Zahl mit dem Index der aktuellen Ringzahl im Array zahl um 1 erniedrigt und im Wahrheitswert „nurabgearbeitet“ gespeichert,   
dass eine Empfehlung befolgt wurde.  
3. Falls nein, wird geprüft, ob noch empfohlen wird, „optimum“ Ringe zu prüfen. Ist dies gegeben, wird zahl[optimum] um 1 erniedrigt, da „optimum“ in der im Folgenden generierten  
Empfehlung enthalten ist.  
4. Danach wird die Abweichung des aktuellen Wertes von „optimum“ ermittelt und geprüft,   
ob „optimum“ plus diese Empfehlung im Bereich zwischen 0 und 10 Ringen liegt. Falls ja,   
wird empfohlen, „optimum“ plus diese Abweichung Ringe zu schießen, damit die Summe  
des aktuellen und des nächsten Schusses zweimal „optimum“ beträgt.  
5. Fällt die zum Ausgleich nötige Zahl aus diesem Intervall, muss „optimum“ neu berechnet  
werden. Dafür werden alle Empfehlungen auf 0 gesetzt und „optimum“ als Quotient  
der noch zu erreichenden Ringzahl und der verbleibenden Schusszahl neu berechnet.  
Dabei muss „optimum“ aufgerundet werden, um nicht zu zu wenigen Ringen als Endergebnis zu führen.  
6. „Guthaben“ wird als Differenz *ubrigeSch* ¨ *usse* ¨ *∗ optimum* + *gesamt − ziel* neu berechnet.  
7. Im Wahrheitswert „frischgenullt“ wird gespeichert, dass gerade ein neues „optimum“ berechnet wurde.  
8. Zuletzt wird geprüft, ob „optimum“ größer als 10 ist. Falls ja, wird im Wahrheitswert „nichtmehrzuerreichen“ gespeichert, dass man die Zielringzahl nicht mehr erreichen kann,  
sonst wird im Array „zahl“ die Empfehlung gespeichert, bei jedem übrigen Schuss „optimum“ Ringe zu schießen.  
8  
Im Fall, dass mehr Ringe als „optimum“ geschossen wurden, ist das Vorgehen nahezu identisch.  
Zuletzt wird noch geprüft, ob die Anzahl der Empfehlung, „optimum“ zu schießen (zahl[optimum])  
größer 0, „optimum“ nicht neu berechnet und nicht nur die Ringzahl einer Empfehlung geschossen wurde. Ist dies gegeben, wird zahl[optimum] noch um 1 verringert, da entweder „optimum“  
Ringe geschossen wurden oder „optimum“ in der generierten Empfehlung enthalten ist.  
9  
**2.2.4 Die Generierung der Empfehlungsausgabe**  
Abbildung 5: Teil 3 des Hauptalgorithmus’ als Struktogramm  
Wurde gerade der vorletzte Schuss eingegeben, wird als Empfehlung die Differenz zwischen  
„gesamt“ und der Zielringzahl „ziel“ ausgegeben, also die Zahl der noch zu schießenden Ringe.  
Sonst wird geprüft, ob nicht gerade der letzte Schuss eingegeben wurde, wo eine Schussempfehlung keinen Sinn hätte.  
Falls ja, wird die Empfehlung für den nächsten Schuss generiert:  
1. Zuerst erfolgt die Ausgabe der Information, dass jetzt die Schussempfehlung kommt.  
2. Dann wird das Minimum5 berechnet und, wenn es größer als 0 ist, ausgegeben.  
3. Danach wird (um den Schützen emotional zu entlasten) die kleinste Empfehlung gesucht  
und ausgegeben, wenn sie nicht dem Minimum entspricht.  
Zuletzt wird die Zahl der Durchläufe durch den Hauptalgorithmus „i“ erhöht.  
5siehe Abschnitt 2.2.1  
10  
**2.3 Umsetzung in JavaFX**  
**2.3.1 prinzipielle Umsetzung der grafischen Benutzeroberfläche (GUI) und der Ein- und**  
**Ausgabe**  
Abbildung 6: Die GUI des Programms „Chancenrechner“ unter Windows 8  
Die Benutzeroberfläche ist grundlegend in zwei Bereiche aufgeteilt: die Zielscheibe zur Eingabe  
der einzelnen Schüsse und Ausgabe des Protokolls links und das Ein/Ausgabesegment rechts.  
Dieses enthält zwei Textfelder, in die Schusszahl und Zielringzahl eingegeben werden. Darunter  
kann mit zwei RadioButtons die anzuzeigende Zielscheibe ausgewählt werden, aktuell werden  
„Luftpistole 10m“ und „Luftgewehr 10m“ unterstützt.  
Zudem gibt es weiter unten eine Textausgabefläche, in welche die Schussempfehlungen ausgegeben werden. Ganz unten ist der Timer integriert. Es handelt sich hierbei um ein Textfeld  
mit großer Schriftgröße, um immer gut lesbar zu sein. Solange noch Zeit übrig ist, wird diese in  
grüner Schriftfarbe dargestellt, ist sie abgelaufen, steht im Timer in roter Schrift „Zeit abgelaufen!“.  
Zuletzt gibt es etwa auf mittlerer Höhe im rechten Segment noch einen Mehrfunktionsbutton.  
Zu Programmstart und immer nach Abschluss einer Serie zeigt er „Initialisieren“. Klickt man  
ihn an, wird man aufgefordert, seinen Namen für das Protokoll einzugeben, die Parameter für  
Schuss- und Zielringzahl werden auf Eingabe Fehler geprüft und ggf. ein Dialog zur Korrektur  
dieser angezeigt. Des Weiteren werden alle Markierungen auf der Zielscheibe gelöscht, die  
Empfehlung ausgegeben, möglichst bei jedem Schuss den insgesamt nötigen Durchschnitt zu  
11  
schießen und der Timer auf die insgesamt für diese Schussserie vorgegebene oder berechnete  
Zeit eingestellt. Zum Schluss ändert der Button seinen Titel in „Start“.  
Zeigt der Button „Start“, startet er beim Klick auf ihn den Timer, ermöglicht es, auf der Zielscheibe Schüsse einzugeben und verhindert das Wechseln der Zielscheibe. Sein Titel wird auf  
„Stopp“ gesetzt.  
Zeigt der Button „Stopp“, bricht er bei einem Klick die aktuelle Schussserie ab, stoppt den Timer,  
ermöglicht das Wechseln der Zielscheibe und ändert seinen Titel in „Initialisieren“.  
Läuft das Programm, kann durch Klick auf die Zielscheibe an der Stelle des Treffers dessen  
Schusswert eingegeben werden; das Programm bietet die Möglichkeit, bei Fehlern diesen manuell zu korrigieren. Ein Klick startet jeweils die Empfehlungsberechnung für den nächsten  
Schuss. Nach der erfolgreichen Eingabe bleibt eine Markierung mit Ort des Schusses und  
dessen Nummer zurück.  
Nach Abschluss der Serie wird auf die Zielscheibe Name und Uhrzeit sowie die Auswertung  
mit Schusszahl, Ziel, erreichten Ringen, Erreichen der Vorgabe und Zeitbedarf geschrieben.  
Die Zielscheibe mit Schüssen und Bewertung wird am Ende als PNG-Bild auf dem Desktop  
gespeichert, dazu ein schriftliches Protokoll mit Empfehlungen des Programms. Zudem wird  
alle 10 Schuss die Scheibe gespeichert und geleert, um sie nicht zu sehr zu füllen (Übersichtlichkeit). Eine mögliche Erweiterung wäre es, Bild- und Textprotokoll zusammen als PDF zu  
speichern.  
Die technische Umsetzung der GUI erfolgt mittels einer auf FXML basierenden JavaFX-GUI,  
auf einem Pane6 liegt ein ImageView7, darauf wird ein Canvas8 angeheftet (als Zeichenfläche  
für die Kreise um die eingegebenen Schüsse), an das Pane ist ein MouseListener9 angeheftet.  
**2.3.2 Umsetzung wichtiger Algorithmen**  
**Umsetzung der Eingabe eines Schusswertes** Das Bild der Zielscheibe liegt als PNG vor  
und ist wie unter 2.3.1 beschrieben in die GUI integriert. Der Algorithmus zur Ermittlung des  
Schusswertes wird bei einem Klick auf dieses Bild gestartet, wenn eine Serie läuft. Dabei liefert der angeheftete MouseListener die Koordinaten des Klicks. Die Koordinaten der Mitte der  
Zielscheibe sind bekannt. Somit kann der Abstand a eines Schusses von der Zielscheibenmitte  
ermittelt werden durch: ∆*x* = *|x*1 *− x|* und ∆*y* = *|y*1 *− y|* durch *a* = r(*x*1 *− x*)2 + (*y*1 *− y*)2  
nach Satz des Pythagoras.  
Abbildung 7: Illustration der Herleitung der Berechnung des Abstandes eines Schusses von der  
Mitte  
6Ein Oberflächenelement, auf dem andere Elemente angeheftet werden können.  
7Ein Oberflächenelement, welches Bilder anzeigen kann.  
8„Canvas“ (engl.) : „Leinwand“  
9Ein Objekt, das bei einem Klick auf das Oberflächenelement, an das es angeheftet ist, eine Aktion ausführt.  
12  
Durch den Vergleich von a mit den Abständen der jeweiligen Ringe von der Mitte kann nun dem  
Schuss eine Ringzahl zugeordnet werden.  
Diese Methode hat den Nachteil, dass die Eingabe besonders an den Rändern der Ringe  
nicht 100% genau ist und die Zielscheibe immer in einer Auflösung von 800 \* 800 Bildpunkten  
dargestellt werden muss, da die gespeicherten Abstände Absolutwerte sind. Für den Fall, dass  
die Eingabe einmal nicht funktionieren sollte, schafft die Möglichkeit, sie in einem JOptionPane  
zu korrigieren, Abhilfe.  
13  
**Umsetzung der Protokollerstellung** Für diese Methode wird zunächst eine neue Instanz  
der Klasse Calendar erstellt. Diese Instanz enthält das aktuelle Datum und die aktuelle Uhrzeit.  
Der Dateiname des Protokolls besteht aus dieser Uhrzeit, die in einem String namens Protokoll  
hinter dem Wort „Protokoll“ gespeichert wird.  
Die statistische Auswertung der aktuellen Schussserie wird nun mittels einer Instanz der Klasse  
Graphics auf die Zielscheibe geschrieben. Sie besteht aus Name und Uhrzeit sowie Schusszahl, Zielringzahl, erreichte Ringe, Erreichen der Vorgabe und Zeitbedarf.  
Das Pane „flaeche“, auf dem das ImageView, welches die Zielscheibe enthält, sowie die Zeichenfläche mit allen Markierungen und der Statistik, liegen, stellt die Methode „snapshot“ bereit. Diese liefert ein Bildschirmfoto der Zielscheibe mit den Markierungen und der Statistik auf  
ihr. Dieses Foto wird in einem WritableImage zwischengespeichert und mittels ImageIO auf  
den Desktop geschrieben.  
Für das schriftliche Protokoll wurden alle Programmein- und Ausgaben mit Ausnahme der Zeit  
im String „Protokoll“ gespeichert. Dieser wird nun nur noch in eine Textdatei mit dem selben  
Dateinamen ebenfalls auf dem Desktop gespeichert.  
**Umsetzen der Zeitfunktion** Immer beim Initialisieren des Programms wird die Gesamtzeit  
ermittelt. Es gibt hierbei feste Zuordnungen (z.B. 10 Schuss zu 15 Minuten), standardmäßig  
rechnet das Programm mit einer Minute für einen Schuss. Diese Gesamtzeit wird einmal als  
„gesamtzeit“ und einmal als „zeit“ in einer Variable mit diesen Namen gespeichert.  
In einer Instanz namens TimerTask der Klasse Runnable ist die Aktion gespeichert, die jede  
Sekunde ausgeführt werden soll:  
1. Zuerst wird geprüft, ob „zeit“ noch größer ist als 0.  
2. Falls ja, wird „zeit“ um 1 Sekunde gesenkt und in grüner Farbe im Textfeld „Zeitanzeiger“  
in Minuten und Sekunden getrennt ausgegeben.  
3. Sonst wird im Textfeld in roter Farbe „Zeit abgelaufen!“ angezeigt.  
Die Instanz der Klasse ScheduledExecutorService „exec“ führt TimerTask jede Sekunde aus.  
Diese Ausführung ist ScheduledFuture<?> „result“ zugewiesen. Die Ausführung von TimerTask  
würde endlos laufen, auch wenn die aktuelle Serie schon vorbei ist, aber „result“ kann die  
Ausführung beenden.  
14  
**2.4 Beschreibung des Arbeitsprozesses und der Programmentwicklung**  
Dieses Projekt wurde in regelmäßigen Treffen mit einem Trainer eines Vereins jugendlicher  
Sportschützen und einem Informatiklehrer erarbeitet.  
Vor den Sommerferien 2015 wurde beim ersten Treffen eine grundlegende Konzeption des  
Programms ausgearbeitet.  
Während der Sommerferien wurde ein erster Prototyp des Algorithmus entwickelt. Es handelte  
sich anfangs noch um ein in der Programmiersprache C++ geschriebenes Programm, welches  
ohne GUI in einem DOS-Fenster lief. Diese ursprüngliche Version ermittelte Empfehlungen  
nach der jeweiligen Abweichung vom Optimum (x Ringe zu wenig getroffen bedeutet Optimum  
+ x Ringe schießen usw.), wenn die zum Ausgleich nötige Zahl kleiner 0 oder größer  
10war, dann wurde diese Zahl reduziert und die Anzahl der Schüsse auf diese Zahl erhöht,  
was zur Folge hatte, dass das Programm viel zu viel vom Schützen erwartete. Zudem wurde  
das gesamte Arrays mit den Empfehlungen ausgegeben statt einer einzelnen Empfehlung und  
die Ausgabe des Minimalwertes fehlte.  
In den ersten Schulwochen des Schuljahres 2015/2016 wurden viele kleine Fehler am Hauptalgorithmus behoben und die Neuberechnung des Optimalwertes eingebaut.  
Beim zweiten Treffen wurde der Prototyp dem Trainer zum Testen übergeben sowie einige  
Details verbessert.  
Beim dritten Treffen wurden Verbesserungen an der Ausgabe besprochen10 und der Algorithmus so erweitert, dass er bis zum Ende der Schussserie durchläuft, auch wenn das Ringziel  
nicht mehr erreichbar ist, statt dann abzubrechen.  
Danach wurde die Ausgabe des Minimalwertes hinzugefügt und die Empfehlungsausgabe so  
geändert, dass immer nur eine Empfehlung ausgegeben wird. Außerdem wurde noch ein Prototyp eines JavaFX-Programms mit GUI entwickelt, der jedoch nur Ein- und Ausgabe beherrschte.  
Beim vierten Treffen wurden beide Prototypen übergeben sowie an beiden einige Details verbessert, besonders die Hinzufügung der Zeitbetrachtung.  
Danach wurde der Hauptalgorithmus in den Prototypen mit GUI implementiert und die GUI so  
fertiggestellt, dass sie alle Funktionen hatte, die sie auch jetzt noch hat.  
Beim fünften Treffen erfolgten Detailverbesserungen an der GUI und dem Algorithmus, besonders, dass alle 10 Schuss die Scheibe geleert wird. Danach wurde die Funktion ergänzt, dass  
alle 10 Schuss die Scheibe zusätzlich zum Leeren noch als Bild gespeichert wird.  
Beim sechsten Treffen erfolgten noch Detailverbesserungen an der Zeitfunktion und der Oberfläche allgemein.  
Während der gesamten Arbeitszeit wurden permanent Fehler im Programm gesucht und ausgebessert, wobei auch die Testprotokolle herangezogen wurden.  
10Siehe die angehängten Protokolle vor und nach diesem Treffen  
15  
**3 Bewertung des fertigen Programms**  
**3.1 Auswertung der Tests**  
Das Programm läuft nach bisherigem Erkenntnisstand stabil, keine Abstürze aufgrund von Fehlern sind bekannt.  
Ein Problem, welches bei einem leistungsmäßig eher schwachen Laptop auftrat: Bei der Korrektur eines Eingabefehlers kam es zu einem Absturz des Programms, wenn man das JOptionPane zu lange angezeigt ließ. Dieses Problem konnte bisher nicht gelöst werden, tritt aber  
bei mittelmäßig leistungsstarken PC’s nicht auf, weshalb man es ignorieren kann.  
Alle bekannten Programmfehler wurden behoben.  
In einer frühen Version des C++- Programms zum Beispiel trat der Fehler auf, dass durch einen  
Umwandlungsfehler von Ganz- in Gleitkommazahlen die Berechnung des Minimums versagte.  
Dieser Fehler konnte behoben werden. Auch der Hauptalgorithmus selbst neigte am Anfang zu  
Fehlern, die jedoch behoben wurden.  
Ein großer Vorteil von Java kommt hier zu tragen: im Gegensatz zu anderen Sprachen wie  
C++, deren Programme bei Programmfehlern abstürzen, wirft Java eine Exception und setzt  
die Ausführung des Programms fort.  
Teilweise kommt es zu Verzögerungen beim Protokollieren, da die relativ großen Datenmengen der Bildprotokolle erst geschrieben werden müssen; eine denkbare Lösung wäre nur Multithreading (hier kaum sinnvoll) oder eine schnellere Festplatte; diese Verzögerungen liegen aber  
lediglich im Bereich der 1-2 Sekunden, sind also gerade so spürbar und kein echtes Problem.  
Die Sportschützen, die dieses Programm getestet haben, und deren Trainer mögen es und  
bezeichnen es als echte Hilfe beim Training.  
16  
**3.2 Vergleich des fertigen Programms mit seinen Zielen**  
Das Programm gibt einem Schützen (lt. Hr. Kunze) sinnvolle und hilfreiche Informationen und  
hilft ihm, über die aktuelle Serie den Überblick zu bewahren und die aktuelle Leistung einzuschätzen (letzter Schuss, der allgemeine Schusstrend mit den Markierungen der Schüsse auf  
der Scheibe. . . ).  
Besonders die Timerfunktion des Programms wird als echte Hilfe gesehen.  
Das Programm ist aktuell nutzbar für das Training mit Luftgewehr und Pistole auf 10 m Entfernung durch die eingebauten sportordnungsgerechten Scheiben, andere Scheiben können  
jedoch leicht eingebaut werden.  
Jede mögliche Kombination von Schüssen und Zielringzahl kann simuliert werden.  
Die Zeitvorgaben im Programm und die angezeigten Zielscheiben basieren auf der deutschen  
Sportordnung.  
Die Protokolle sind für den Trainer interessant, besonders der Durchschnitt.  
Die Protokolle können archiviert und später erneut zur Auswertung des Trainingserfolges der  
Schützen herangezogen werden.  
Die Ein-und Ausgabe ist relativ genau, die Bedienung einfach und eingabefehlerresistent gelungen.  
Dieses Projekt kann noch erweitert werden. Denkbar ist zum Beispiel eine Vertiefung der Statistikfunktionen. Da heute fast jeder ein Smartphone hat, ist auch die Ergänzung des Programms  
durch eine App für den Schützen im Gespräch, die ihm noch einmal die Empfehlungen anzeigt und sich mit dem PC-Programm des Trainers synchronisiert. Zuletzt ist noch geplant,  
den reinen Hauptalgorithmus in ein Programm zu importieren, welches an eine elektronische  
Schießscheibe angeschlossen ist und von dieser die aktuellen Schusswerte bekommt.  
Alles in Allem kann man sagen, das Programm hat seine Ziele voll erfüllt und ist eine echte  
Hilfe für angehende Sportschützen geworden.  
17  
**4 Literaturverzeichnis**  
**Bildquellen**  
*•* https://pixabay.com/static/uploads/photo/2012/04/24/16/43/targets-40383\_640.  
png : 01.01.2016 14:10, Bild der Zielscheibe Luftgewehr 10 m im Programm  
*•* http://www.schuetzenbedarf-baur.com/images/Krueger%20Pistole%2025-50%20m%20C50%  
203101\_100.jpg : 01.01.2016 14:11, Bild der Zielscheibe Luftpistole 10 m im Programm  
*•* http://code.makery.ch/assets/library/javafx-2-tutorial/javafx-logo.png : 08.12.15,  
JavaFX-Logo als PNG-Bild  
*•* andere Bilder in der Projektarbeit: selbst erstellt  
**Hilfe bei der Programmentwicklung**  
*•* http://code.makery.ch/blog/javafx-2-snapshot-as-png-image/ : 01.01.2016 14:14,  
Marco Jacob, JavaFX 2 Snapshot as PNG Image  
*•* http://stackoverflow.com/questions/16128423/how-to-update-the-label-box-every  
-2-seconds-in-java-fx : 01.01.2016 14:18, Stackoverflow.com, How to update the label  
box every 2 seconds in java fx?  
*•* http://tomasmikula.github.io/blog/2014/06/04/timers-in-javafx-and-reactfx.htm  
l : 01.01.2016 14:19, Tomas Mikula, Timers in JavaFX and ReactFX  
*•* http://stackoverflow.com/questions/16764549/timers-and-javafx : 01.01.2016 14:20,  
Stackoverflow.com, Timers and javafx  
*•* https://de.wikibooks.org/wiki/Java\_Standard:\_Datum : 01.01.2016 14:21, de.wikibooks.org,  
Java Standard: Datum  
**persönliche Unterstützung**  
*•* Thiele, Otto, Informatiklehrer, Carl-Zeiss-Gymnasium, Jena, Art der Unterstützung: Hilfe  
bei der Themenwahl und Konzeption des Programms, Unterstützung bei der Umsetzung  
*•* Kunze, Bert, Trainer, Schützenverein am Bürgel eV, Jena, Art der Unterstützung: Hilfe bei  
der Konzeption des Programms, Test der Programmprototypen im Schützenverein  
**5 Anhang**  
**5.1 Übersicht über die Variablen im Hauptalgorithmus und ihre Funktionen**  
Variable Typ Funktion  
erstesmal boolean speichert, ob gerade der erste Schuss der  
aktuellen Serie eingegeben wurde  
zahl Array von 11 intZahlen  
speichert die Empfehlungen, die das  
Programm geben wird; dabei sind die Indizes  
die Schusswerte von 0 bis 10 und deren Werte  
jeweils die Anzahl, wie oft der Wert zu  
schießen ist  
optimum int speichert den nötigen Durchschnittswert, um  
die Zielringzahl zu erreichen; immer  
aufgerundet bei der Berechnung, um nie zu  
wenig Ringe zu empfehlen  
frischgenullt boolean speichert, ob gerade optimum neu berechnet  
wurde  
gesamt int speichert die insgesamt geschossene  
Ringzahl  
aktuell int speichert den aktuell eingegebenen Schuss  
ergebnisse Array von int-Zahlen;  
für jeden Schuss der  
Serie eine  
speichert die einzelnen Ringzahlen der  
Schüsse  
letzter int speichert die Ringzahl des letzten Schusses  
echteschuesse int Nummer des aktuellen Schusses  
ziel int speichert die Zielringzahl  
nichtmehrzuerreichen boolean speichert, ob die Zielringzahl nicht mehr zu  
erreichen ist oder bereits erreicht wurde  
i int speichert die Nummer des aktuellen  
Durchlaufs  
nurabgearbeitet int speichert, ob eine Empfehlung eingehalten  
wurde  
Guthaben int speichert die Differenz zwischen der  
Zielringzahl und dem Wert, den man erhält,  
wenn man die Empfehlungen befolgt  
zwischensumme int Mehrzweck-Variable: speichert die Zahl aller  
verbleibender Schüsse  
j int speichert die aktuelle Empfehlung  
Tabelle 1: Übersicht über die Variablen im Hauptalgorithmus und ihre Funktionen  
19  
**5.2 Testprotokolle**  
**5.2.1 Testprotokoll vor dem dritten Treffen**  
H  
Programm Chancenrechner Version 0.4 ALPHA  
Entwickler: Eric Ackermann- Carl-Zeiss-Gymnasium Jena  
Experimentelle Programmversion, Nutzung auf eigene Gefahr.  
Schuetze: 10  
Schuesse: 5  
Zielpunkte: 32  
Empfehlungen werden gegeben zum Erreichen von 35 Punkten.  
Ihr Ziel: durschschnittlich 7 Punkte.  
Schuss 1: 7  
Ergebnisse der Schuesse: 7;  
Bisherige Gesamtpunktzahl: 7  
Sie muessen noch 4 mal 7 Treffer erzielen.  
Schuss 2: 7  
Ergebnisse der Schuesse: 7;7;  
Bisherige Gesamtpunktzahl: 14  
Sie muessen noch 3 mal 7 Treffer erzielen.  
Schuss 3: 7  
Ergebnisse der Schuesse: 7;7;7;  
Bisherige Gesamtpunktzahl: 21  
Sie muessen noch 2 mal 7 Treffer erzielen.  
Schuss 4: 7  
Ergebnisse der Schuesse: 7;7;7;7;  
Bisherige Gesamtpunktzahl: 28  
Sie muessen noch 4 Treffer erzielen. Dies ist Ihr letzter Schuss.  
Schuss 5: 7  
Ergebnisse der Schuesse: 7;7;7;7;7;  
Mit 35 Punkten haben Sie Ihr Ziel von 32 Punkten erreicht. Ihnen stehen noch 0 Schuesse  
zur Verfuegung.  
Sie haben 35 Punkte erzielt.  
Sie haben durchschnittlich 7 Treffer erzielt.  
Herzlichen Glueckwunsch, Sie haben Ihre Zielpunktzahl erreicht.  
20  
**5.2.2 Testprotokoll nach dem dritten Treffen**  
H  
Programm Chancenrechner Version 0.5 ALPHA  
Entwickler: Eric Ackermann- Carl-Zeiss-Gymnasium Jena  
Experimentelle Programmversion, Nutzung auf eigene Gefahr.  
Schuetze: Alex  
Serie: 10  
Ringe zu erreichen: 78  
Das Ziel: durschschnittlich 7.8 Ringe.  
Schuss 1: 9  
Gesamtringzahl: 9;  
Naechster Schuss:  
Empfehlung: 8 oder mehr Ringe.  
Schuss 2: 10  
Gesamtringzahl: 19;  
Naechster Schuss:  
Empfehlung: 8 oder mehr Ringe.  
Schuss 3: 7  
Gesamtringzahl: 26;  
Naechster Schuss:  
Empfehlung: 8 oder mehr Ringe.  
Schuss 4: 8  
Gesamtringzahl: 34;  
Naechster Schuss:  
Empfehlung: 8 oder mehr Ringe.  
Schuss 5: 7  
Gesamtringzahl: 41;  
Naechster Schuss:  
Empfehlung: 8 oder mehr Ringe.  
Schuss 6: 7  
Gesamtringzahl: 48;  
Naechster Schuss:  
Empfehlung: 8 oder mehr Ringe.  
Schuss 7: 8  
Gesamtringzahl: 56;  
Naechster Schuss: mindestens 2 Ringe;  
Empfehlung: 8 oder mehr Ringe.  
Schuss 8: 8  
Gesamtringzahl: 64;  
Naechster Schuss: mindestens 4 Ringe;  
Empfehlung: 8 oder mehr Ringe.  
Schuss 9: 7  
Gesamtringzahl: 71;  
21  
Sie muessen noch 7 Ringe erzielen. Dies ist Ihr letzter Schuss.  
Schuss 10: 8  
Gesamtringzahl: 79;  
Sie haben 79 Ringe erzielt und dafuer 1 Minuten 25 Sekunden gebraucht.  
Sie haben durchschnittlich 7.9 Ringe erzielt.  
Herzlichen Glueckwunsch, Sie haben Ihre Zielringzahl erreicht.  
22