Mirko Brandt (2270222)
Tim Fock (2270398)

1. Aufgaben der Software

Bei dieser Software handelt es sich um einen sehr simplen Mathetrainer für die drei Themengebiete: Division, Brüche kürzen und Prozentrechnung. Für jedes Themengebiet gibt es ein anschauliches Tutorial in Textform. Alle Aufgaben sind in zwei Schwierigkeitsgrade aufgeteilt. Die leichteren Aufgaben sollen dazu dienen sich mit den Aufgabentypen vertraut zu machen und den Lösungsweg zu verinnerlichen. Bei den schwereren Aufgaben wird schriftliches Rechnen nötig. Die Zielgruppe sind Schüler der sechsten oder siebten Klasse.

Die Software hat nicht den Anspruch neues Wissen zu vermitteln, sondern ist eher als Unterstützung zum normalen Matheunterricht zu verstehen. Die Tutorials sind zwar einfach gehalten, allerdings auch so begrenzt, dass jemand, der zuvor noch nicht mit diesen Themen zu tun hatte, Probleme haben wird alles zu begreifen.

2. Installationsanleitung

Den Mathetrainer kann man unter https://github.com/disteLLL/Mathetrainer, über die grüne Schaltfläche „Clone or download“, herunterladen. Nach dem Entpacken der ZIP-Datei befindet sich im Hauptordner eine Datei namens „Mathetrainer.jar“. Ein Doppelklick auf diese Datei öffnet den Mathetrainer. Einzige Voraussetzung ist eine aktuelle Java-Version (1.6+) auf dem ausführenden System.

3. Bedienungsanleitung

Beim Start der Software öffnet sich das Hauptfenster und erlaubt den Zugriff auf die Tutorials und den Beginn einer Übung. Drückt man auf den Button „Tutorials“ öffnet sich das Tutorialfenster mit den drei Auswahlmöglichkeiten für die Themengebiete. Die Tutorials sind alle gleich aufgebaut und lassen sich mit Hilfe der Registerkarten „durchklicken“. Hat man ein Tutorial beendet und möchte zum nächsten, kommt man über den Button unten zurück zur Übersicht der Themengebiete.

Um eine Übung zu starten, reicht ein Druck auf den Button „Start“ im Hauptfenster. Vorher kann man sich mit den Häkchen aussuchen, welche Themen vorkommen sollen und welcher Schwierigkeitsgrad gewünscht wird.

Im Übungsfenster werden jetzt Aufgaben, nach den Präferenzen, angezeigt. Um die Bedienung zu vereinfachen, kann man alle Eingaben per Enter-Taste bestätigen und so auf die Maus verzichten. Bei Druck auf den Button „Check“, wird das Ergebnis angezeigt und der Button eingefärbt; grün bei richtiger Antwort, rot bei falscher bzw. keiner Antwort.

Dadurch, dass Tutorials und Übungen in zwei verschiedenen Fenstern sind, kann man parallel zu einer Übung in den Tutorials nachschlagen.

4. Softwarearchitektur

Die Swing-Anwendung ist aufgeteilt in zwei Bereiche: GUI und Logik.

Der erste Bereich beinhaltet die drei GUI-Klassen für die jeweiligen Frames. Beim Start des Programmes werden alle drei Frames erstellt, allerdings ist nur die MainGUI sichtbar. Die anderen beiden Fenster werden durch die entsprechenden Buttons des Hauptfensters sichtbar geschaltet. Ein Druck auf einen Tutorialbutton sorgt dafür, dass die passenden Bilder aus dem Resources-Ordner in ein Tabbed Pane geladen werden.

Wenn im Übungsfenster eine neue Aufgabe generiert wird, kommt der zweite Bereich der Anwendung zum Einsatz. Es folgt ein Call an die erste der beiden statischen Hilfsklassen. Diese wählt, basierend auf der Benutzerauswahl, ein Themengebiet aus und ruft dann die entsprechende Methode der Klasse Math aus und übergibt den gewählten Schwierigkeitsgrad.

In dieser Klasse wird dann von unserem Algorithmus eine Aufgabe erstellt und zur Darstellung an die TaskGUI-Klasse übergeben.

5. Technischer Teilaspekt

5.1. Logik

Bei dem Projekt Mathetrainer habe ich mich auf die Logik hinter dem Aufbau der GUI beschäftigt. Dabei ging es größtenteils um das Erstellen zufälliger Mathematik-Aufgaben, die einen gewissen Rahmen je nach Schwierigkeit haben sollten. Aber auch das Zusammenführen von GUI und Logik stellte sich als eine ernstzunehmende Aufgabe dar.
Im weiteren Verlauf des Textes werde ich mich mit dem Aufbau und der Denkweise hinter der Logik, den Schwierigkeitsgraden sowie den zahlreichen Schwierigkeiten, die dabei aufgekommen sind, beschäftigen.

5.2. Aufbau und Denkweise

Da Tim und ich die Aufgaben des Projektes in GUI und Logik unterteilten, fingen wir beide an zu programmieren, ohne wirklich vorher zu wissen, wie das Ganze am Ende wieder zusammen passen sollte. Dementsprechend startete ich damit, mir eine eigene GUI zu bauen, die aber nur das Allernötigste beinhaltete, um die Zufallsaufgaben selbst testen zu können. Daraus entstanden drei Klassen mit der Logik für das Erstellen der Aufgabe und dem Errechnen des Ergebnisses.
Uns war von Anfang an wichtig, dass man nur mit ganzen Zahlen rechnen sollte, so mussten in jede Klasse Algorithmen eingebaut werden, die dafür sorgten, dass es nur ganzzahlige Ergebnisse gab.

Division

Bei der Division fing ich damit an, eine zufällige Zahl zu erstellen, die der Dividend werden würde. In einer Schleife wurden alle ganzzahligen Teiler in einer ArrayList gespeichert. Hierbei verwendete ich Modulo, um alle Zahlen, deren Rest beim Teilen nicht null war, auszuschließen.
Um zu vermeiden, dass Aufgaben mit dem gleichen Dividend und Divisor entstehen (typischerweise bei Primzahlen, da sie nur durch eins und sich selber teilbar sind), schloss ich

durch die Kombination einer do-while-Schleife und einer if-Schleife alle Dividenden aus, die weniger als fünf Teiler hatten. So ging ich gleichzeitig sicher, dass es eine größere Auswahl bei den Divisoren gab.
Um nun einen der fünf oder mehr Divisoren auszuwählen, die in der ArrayList abgespeichert waren, erstellte ich eine Zufallszahl zwischen eins und drei (bei fünf Divisoren). So wurden weder der kleinste noch der größte Divisor ausgewählt, welche immer eins und der Dividend selber wären.

Brüche kürzen

Das Erstellen einer Aufgabe bei den Brüchen war wesentlich leichter, als bei der Division. Man musste lediglich sicherstellen, dass sich der Bruch am Ende durch eine ganze Zahl kürzen ließ. Um das zu erreichen, erstellte ich drei zufällige Zahlen. Zwei dieser Zahlen ergaben einen Bruch und sowohl Teiler als auch Nenner wurden mit der dritten Zahl multipliziert. Die Schwierigkeit hier war vielmehr das Ergebnis festzustellen. Grob betrachtet könnte man annehmen, dass man einfach mit der dritten Zahl kürzen muss und fest steht das Ergebnis. Doch natürlich können bei zwei zufälligen Zahlen Brüche entstehen, die selber nochmals kürzbar sind, wie z.B. 3/6tel.
Deswegen musste ich eine Methode finden, wie das Programm selber den Bruch soweit kürzen würde, wie es geht. Dazu schaute ich mir einen Teil der Methode des euklidischen Algorithmus ab. Das Ergebnis musste immer der komplett gekürzte anfänglich erstellte Bruch sein. In zwei ArrayLists wurden alle Teiler der beiden Zahlen gespeichert und dann in einer doppelten for-Schleife miteinander verglichen. Das größte gleiche Ergebnis ist der größte gemeinsame Teiler der beiden Zahlen.

Prozent

Für Prozentrechnung gibt es grundsätzlich drei verschiedene Aufgabentypen:

- X% von Y = Z (20% von 200 = 40)
- Wie viel sind Z von Y? Ergebnis: X% (40 von 200 = 20%)
- Wenn Z X% sind, wie viel sind dann 100%? Ergebnis: Y (40 = 20% , 100% = 200)

Wie man hierbei schon sehen kann, reicht es eine einzige Aufgabe zu erstellen, aus der dann alle anderen Aufgabentypen erstellt werden können durch simples Vertauschen der Zahlen. Ich habe die oberen Beispiele X, Y und Z bereits so angeordnet, dass man dies sehen kann.

Y ist die größte Zahl und immer 100%, Z eine Zahl, die immer kleiner als Y ist und X der Prozentwert von Z an Y. Hat man dies verstanden ist es recht einfach Prozentaufgaben innerhalb dieser drei Typen zu erstellen.
Wie bereits am Anfang geschrieben, wollten wir aber auch darauf achten, dass die Ergebnisse immer ganze Zahlen sind. Deswegen fängt das Programm damit an, innerhalb einer do-while-Schleife eine zufällige große Zahl (Y) und eine Zahl zwischen zwei und 99 (X) zu erstellen. Die while-Komponente kontrolliert nach jedem Erstellen, ob das Ergebnis (Z) der Prozentrechnung eine ganze Zahl ist oder nicht.
Je nachdem welcher Aufgabentyp von dem Programm zufällig ausgewählt wird, werden die X, Y und Z unterschiedlich angeordnet.

5.3. Schwierigkeitsgrade

Für jeden der drei Aufgabentypen wollten wir zwei Schwierigkeitsgrade erstellen. Diese sollten jedoch nicht nur einfach die Zahlenbereiche vergrößern, sodass höhere Zahlen in den Aufgaben vorkamen. Es sollte einen merkbaren Unterschied zwischen den Levels geben. So entschieden wir uns viel damit zu arbeiten, die zufällig erstellten Zahlen zu beschränken. Für den einfachen Schwierigkeitsgrad beschränkten wir bei z.B. der Division den Dividenden auf eine Zahl, die durch zwei oder durch fünf teilbar sein musste. Bei dem schwierigen hingegen wurde genau das Gegenteil gemacht und alle Dividenden, die durch zwei oder fünf teilbar waren, ausgeschlossen.
Die Schwierigkeitsgrade der Prozentrechnung arbeiten nach dem selben Prinzip.
Nur die Aufgaben des Brüche Kürzens bedurften einer besonderen Behandlung. Hier arbeitete ich vor allem mit dem Verstellen der Grenzwerte der Zahlenbereiche. Bei dem einfachen Schwierigkeitsgrad wurden niedrigere Zahlen erstellt und darauf geachtet, dass wenig schwierig zu teilende Zahlen aufkamen. Bei Level zwei wieder genau das Gegenteil.

5.4. Schwierigkeiten

Wie ich bereits am Anfang des Textes erwähnte, planten Tim und ich in unserem ersten Treffen, wie die einzelnen Module grob aussehen sollten. Dabei besprachen wir jedoch nicht, wie das ganze am Ende zusammenpassen sollte.

Um alles zusammenzuführen trafen wir uns ein weiteres Mal und es wurde klar, dass ich einen Großteil meines Codes neu schreiben musste. Natürlich konnte die Logik dahinter einfach übernommen werden.
Weitere Schwierigkeiten barg das Balancieren der Schwierigkeitsgrade. Hier musste ich herausfinden, wie ich einfache und schwierige Aufgaben innerhalb der einzelnen Schwierigkeitsgrade kreieren konnte. Dies nahm viel Zeit in Anspruch, da immer wieder getestet werden musste, was funktionierte und was nicht.

6. Auswertung

Zu Beginn dieses Projektes haben wir uns zu einem Brainstorm getroffen und ein Konzept ausgearbeitet. An dieses haben wir uns gehalten und brauchten so nur ein weiteres Treffen, bei dem wir beide Programmteile verbunden und kleinere Anpassungen vorgenommen haben. Insgesamt sind wir mit dem Ergebnis sehr zufrieden und glauben, dass das Programm seinen Zweck gut erfüllt.

7. Quellen für die Tutorials

Division:

- http://www.frustfrei-lernen.de/mathematik/division-dividieren-zahlen-quotient.html
- http://www.frustfrei-lernen.de/mathematik/schriftliche-division-dividieren-zahlen-mathematik.html

Brüche kürzen:

- http://www.mathepower.com/infobruchkur.html
- https://www.formelsammlung-mathe.de/grundrechenarten/groester-gemeinsamer-teiler.html

Prozentrechnung:

- https://de.serlo.org/mathe/zahlen-groessen/prozent-zinsrechnung/prozent
- https://de.serlo.org/mathe/zahlen-groessen/prozent-zinsrechnung/prozentrechnung-mittels-formeln