Dokumentation Projekt Formelrechner

Inhaltsverzeichnis

[**Abstract** 1](#_Toc105010559)

[**Projektidee** 1](#_Toc105010560)

[**Anforderungen** 1](#_Toc105010561)

[**UML-Diagramme:** 2](#_Toc105010562)

[**Einbindung des Vorgehensmodells:** 4](#_Toc105010563)

[**Konzeption und Projektplanung** 5](#_Toc105010564)

[**Stand des Projektes** 7](#_Toc105010565)

[**Funktionsbeschreibung des Programms** 8](#_Toc105010566)

[**Was können wir aus unserem ersten Projekt mitnehmen / Was haben wir daraus gelernt?** 8](#_Toc105010567)

[**Tagebuch:** 10](#_Toc105010568)

[**Abbildungsverzeichnis:** 14](#_Toc105010569)

# **Abstract**

Das Projekt “**Formelrechner**” wurde von der Gruppe **dynamisches Trio** entwickelt und fokussiert sich auf die Entwicklung eines Programms, mit Hilfe dessen sich Schüler eigene **Formellisten** zusammenstellen können. Die enthaltenen Formeln können dann nach einmaliger Abspeicherung beliebig umgestellt werden und die Unbekannte berechnet werden.

Das Anfangsdatum ist der **24.03.2022**

Das Abgabedatum ist der **29.05.2022**

# **Projektidee**

* Ein Programm entwickeln, welches Formeln in gewählten Listen speichert und diese dann anschließend beliebig Umstellen und die Unbekannte ermitteln kann.

# 

# **Anforderungen**

* Liste aus wichtigsten Formeln sollte bereits vorhanden sein
  + Suchfunktion nach Formel oder Name
  + Unterteilung in Bereiche (Fächer)
  + umstellen oder Werte einsetzen
* selbst Formeln eingeben und später der Liste hinzufügen

# **UML-Diagramme:**

*UML-Diagramm vor Änderung:*

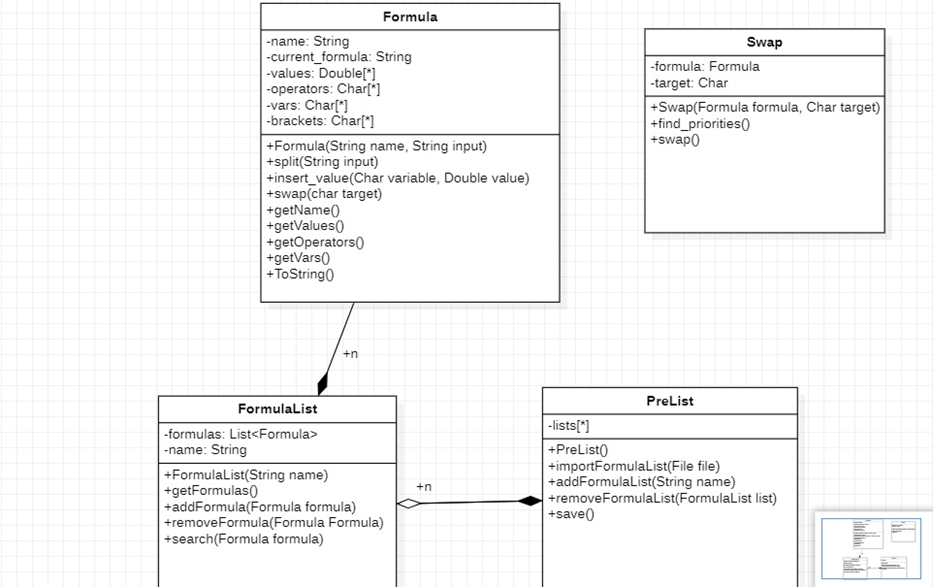


Figure 1: UML-Diagramm vor Änderung

*UML-Diagramm nach Änderung:*

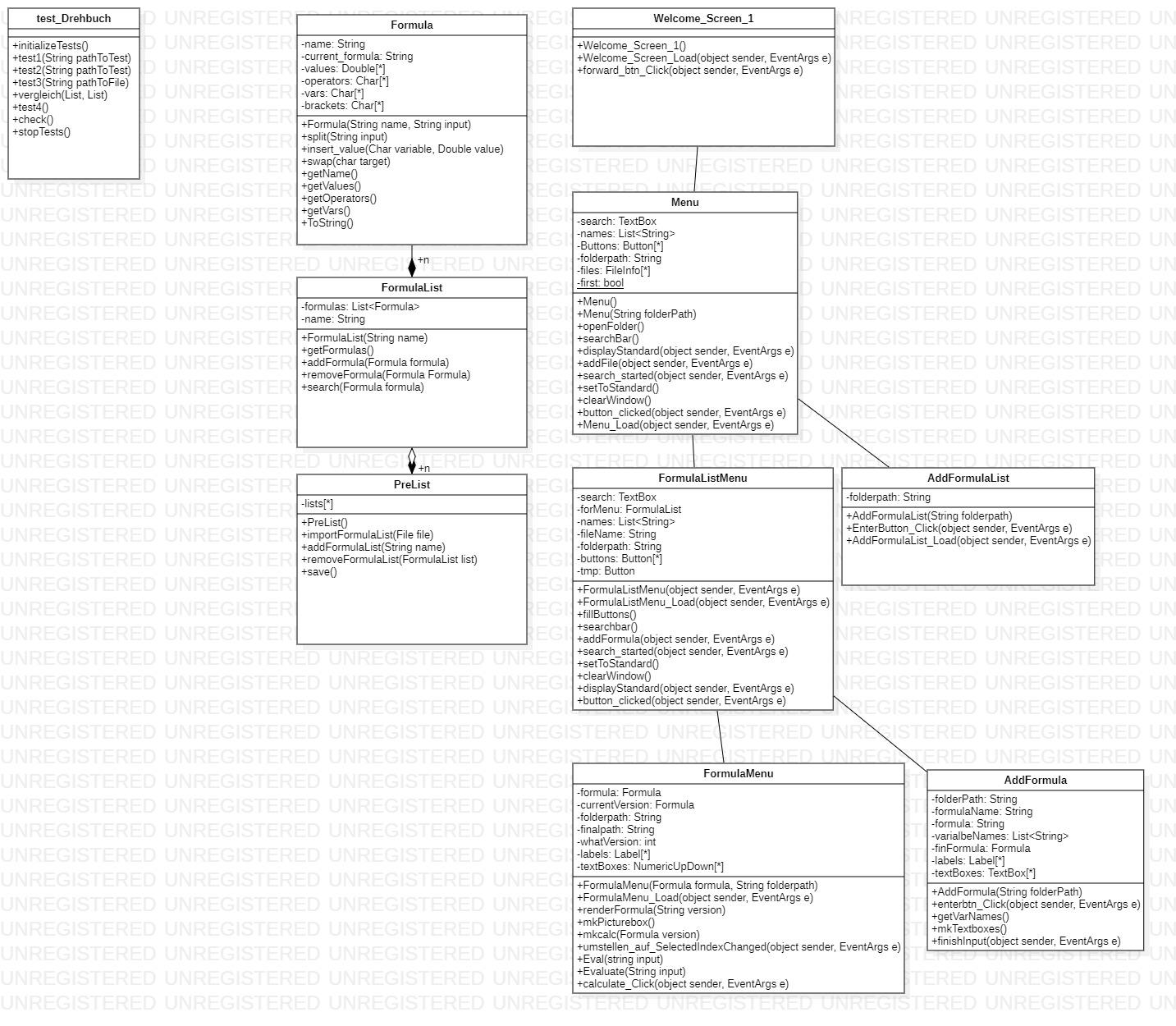
****

Figure 2: UML-Diagramm nach Änderung

# **Einbindung des Vorgehensmodells:**

Dadurch, dass als Vorgehensmodell Kanban gewählt wurde, wurde zunächst ein Pinboard angelegt. Dies kann man auf [kanbanflow.com](https://kanbanflow.com/) gratis online machen. Anschließend wurden die Aufgaben in viele kleine Stücke aufgeteilt. Diese wurden dann den einzelnen Gruppenmitgliedern zugewiesen und anschließend abgearbeitet. Wichtig dabei war, dass immer zunächst ein Schritt fertig abgearbeitet wurde, bevor man mit dem Nächsten begann. Dabei wurde der Status des Schritts stets im Pinboard eingetragen, damit alle Gruppenmitglieder eine genaue Übersicht darüber besaßen.



Figure 3: Kanbanflow Übersicht

Die Einteilung eines Kanbanboards kann man in diesem Bild nochmals sehen (Tasks wurden für diese Ansicht beliebig verschoben, da eigentlich bereits alle erledigt wurden). Würde man nun einen Task anklicken, so erscheinen nochmals spezifische Informationen zum Auftrag:

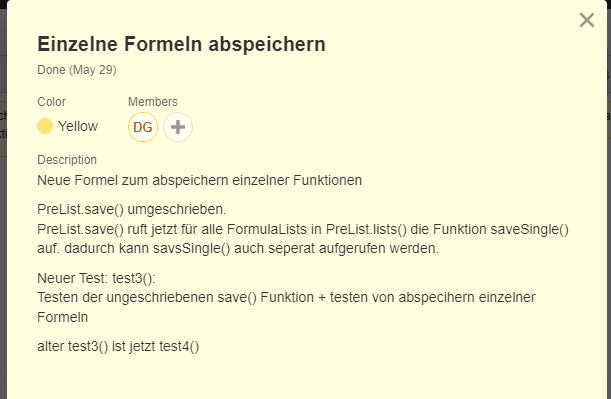


Figure 4: Kanbanboard Eintrag

Zusätzlich zum täglichen Beobachten des Pinboards wurden auch Kadenzen(Name für Meetings bei Kanban) täglich gehalten. Diese müssten zwar nicht täglich gehalten werden, jedoch war es für uns dadurch erheblich leichter, die Einteilung der Aufgaben gut hinzubekommen. Von Vorteil war in unserem Fall zudem besonders, dass es sich bei Kanban um ein agiles Vorgehensmodell handelt. So konnte bei uns spontan umgeplant werden (Umstellen der Formeln manuell statt automatisch), da es zeitlich ansonsten kritisch geworden wäre. Zudem beanspruchte die Änderung, durch die Einteilung des Projekts in kleinere Schritte, nicht viel Zeit und es konnte mit nur leichten Verzögerungen fortgefahren werden.

# **Konzeption und Projektplanung**

* Das Programm speichert eingegebene Formeln in eine Klasse “Formula” ab.
* Formeln werden in “FormulaList”s gespeichert.
* Jede FormulaList repräsentiert ein Fach/eine Formelliste
* FormulaLists werden in der “PreList” gespeichert
* Die PreList dient zum Managen und Abspeichern der Formeln
* Der Name PreList kommt daher, dass in ihr die Standardformeln schon abgespeichert sind...
* Prioritäten
  + Must have:
    - abspeichern der Formeln
    - vorgeschlagene Standardformeln
    - Berechnung der Unbekannten -> Ersetzen der Variablen mit Werten
    - weiteres Hinzufügen von Formellisten und Formeln über Programm
  + Nice to have:
    - Daten aus Digitalem Register (googlen wie (API))
    - Graphische Darstellung
    - Durchschnitt automatisch ausrechnen
    - Kalenderähnliche Darstellung aller Tage
    - Visualisierung der Tests, Prüfungen und Hausaufgaben
    - Selbsteingabe von Tests, Prüfungen und Hausaufgaben (Wenn ein Lehrer etwas nicht einträgt)
    - Erstellen einer eigenen Formelsammlung und Abspeichern dieser in einer PDF in LaTeX- Format
    - Umrechner zwischen verschiedenen Zahlensystemen
* Zeitplan
  + bis Anfang Mai sollte die Planung fertig sein
  + Das Projekt sollte bis Ende Mai fertiggestellt sein
* Test Plan (Unit Tests)
  + Initialisierung der Tests
    - Es wird ein Ordner erzeugt, in dem die Tests mit den Dateien ausgeführt werden
    - Es werden zwei FormulaLists mit einigen Formeln in die PreList gespeichert
  + Test 1 [generelles ein- und auslesen]
    - Die FormulaLists der PreList werden als Liste und als Datei abgespeichert und die PreList geleert
    - Die in der Datei abgespeicherten Listen werden wieder eingelesen und mit den FormulaListen, die in einer List gespeichert wurden verglichen
    - (siehe Vergleiche)
  + Test 2 [single ein- und auslesen]
    - Die erste FormulaList der PreList wird gespeichert
    - Die erste FormulaList der PreList wird noch einmal eingelesen, diesmal mit dem PreList.importSingleFL() Befehl
    - Die beiden Listen werden verglichen
  + Test 3 [einzelne Formeln speichern]
    - Die erste FormulaList der PreList wird kopiert und gespeichert
    - Die erste Formel der FormulaList wird nochmals hinzugefügt
    - Die erste Formel der FormulaList wird gespeichert
    - Die erste Formel der FormulaList wird eingelesen und der Kopie der FormulaList hinzugefügt
    - die beiden Listen werden verglichen
  + Test 4 [Funktionen]
    - 1: Suchen der Formel “Satz des Pythagoras”
    - 2: Überprüfen des Namen der gefundenen Formel
    - 3: löschen der Formel (Überprüfung durch erneutes Suchen)
    - 4: Überprüfen der Standardformel der gefundenen Formel
    - 5: Löschen einer FormulaList
    - 6: Löschen aller Listen der PreList
  + Vergleiche
    - In den Tests 1 2 und 3 werden Listen verglichen. Die Vergleiche laufen wie folgt ab:
      * 1: Die Länge der beiden Listen wird verglichen
      * 2: Die Anzahl der Formeln in beiden Listen wird verglichen
      * 3: Die Namen der beiden Listen werden verglichen
      * 4: Die zum abspeichern benötigten Strings (vergleichbar mit der ToString() Methode) aller Formeln in den beiden Listen werden verglichen
      * 5: Die Namen Aller Formeln in beiden Listen werden verglichen
      * 6: Die Standardformeln aller Formeln in beiden Listen werden verglichen
      * 7: Alle Versionen aller Formeln in beiden Listen werden verglichen
      * 8: Alle Variablen aller Formeln in beiden Listen werden verglichen
    - Der Rückgabewert (Fehler) entspricht der Nummer des Tests oder 0 bei Erfolg

# **Stand des Projektes**

* Schon nach kurzer Zeit wurde klar, dass aus Zeitgründen nicht alle Funktionen realisiert werden können. So wurde schon früh die Einbindung des digitalen Registers und eine Visualisierung der Tests in einem Kalender als zu aufwändig erklärt und weggelassen. Auch das Erstellen einer PDF, welche eine personalisierte Formelsammlung in LaTeX- Format enthält, ein Umrechner für verschiedene Zahlensysteme und ein Rechner für den Notendurchschnitt standen zur Debatte, wurden aber ebenfalls wegen zeitlichen Gründen weggelassen.
* Beim Programmieren wurde auch klar, dass die anfangs geplante Funktion, Formeln automatisch umzustellen technisch viel zu aufwändig und für uns zeitlich nicht realisierbar ist. Es wurde der Kompromiss festgelegt, dass der Benutzer die Formeln und deren Variationen händisch eingibt.
* **Noch zu beseitigende Fehler**: Bei den Formeln können noch keine negativen Werte eingesetzt werden. Dies liegt daran, dass wir zuerst DataTable.Compute() zum Berechnen des Strings verwenden wollten. Jedoch bietet diese Methode und viele andere Lösungen, welche wir im Internet finden konnten, kein Operation für Potenzen an. Nach langer Suche konnten wir zwar eine Lösung für Potenzen finden, jedoch leider ohne die Fähigkeit, mit negativen Werten umzugehen. Wir waren gezwungen, diesen Kompromiss einzugehen. Auch logarithmische Operationen werden noch nicht angeboten. Zudem wird noch in seltenen Fällen eine System.IO-Exception nach dem Erstellen einer neuen Formelliste geworfen, da 2 Prozesse gleichzeitig versuchen, auf die Datei zuzugreifen. Da wir das File mit File.Create() erstellen fanden wir keinen Weg, diese Exception zu unterbinden, weil dort auch keine Verbindung geschlossen werden kann. Zudem wird pi mit LaTeX noch als Fragezeichen dargestellt, das Rechnen funktioniert jedoch. Auch das Beseitigen der PNGs, welche bei der Erzeugung der LaTeX-Ansicht im Ordner anfallen funktioniert noch nicht, da wir hier nochmals das bereits genannte Problem haben, dass bereits ein anderer Prozess darauf zugreift und wir diese somit nicht automatisiert löschen können.

* Im Laufe der Entwicklung stellte sich heraus, dass der Arbeitsaufwand größer war, als anfangs erwartet. Dies führte zu einem erhöhten Stress im Team gegen Ende des Projekts.
* Arbeitsaufteilung
  + Anfangs suchte sich jeder unter Absprache mit den anderen selbst eine oder mehrere Backend- Klassen, an denen er arbeiten wollte.
  + Als diese Klassen großteils fertig waren, konzentrierten sich Damian und Jonas auf das FrontEnd und die UI, während Diego sich auf die Erweiterung und Fertigstellung des Backends fokussierte.

# **Funktionsbeschreibung des Programms**

Beim Start des Programms wird zunächst ein Welcome-Screen angezeigt. Dieser fordert zum Öffnen eines Ordners aus. Nun sollte der Ordner mit den Formeln geöffnet werden, der sich standardmäßig im Ordner des Projekts befindet. Es kann natürlich auch ein neuer Ordner angelegt werden, nur würde dieser dann nicht von Beginn Formeln enthalten. Nachdem der Ordner geöffnet wurde, erscheint das Hauptmenü. Hierbei sind alle Formellisten anhand von Buttons aufgelistet. Es wird auch eine Suchleiste angeboten. Diese zeigt nach Bestätigung der Suche durch den vorliegenden Button alle Formellisten, welche das eingegebene Wort enthalten. Durch den Return-Button kann vom Suchmenü wieder ins Standardmenü gewechselt werden. Mit dem + Button kann eine neue Formelliste erstellt werden. Wird nun eine Formelliste ausgewählt, erscheint das gleiche Menü mit den enthaltenen Formeln und den genannten Optionen. Wird nun hier eine Formel angeklickt, dauert es einige Sekunden, bis sich das Fenster der Formel öffnet, da die PNG-Datei der LaTeX-Formatierung erst gerendert werden muss. Schließlich wird das Fenster angezeigt. Hier kann nun mit einem Dropdown-Menü die Formel ausgewählt werden, auf die umgestellt werden soll. Zudem werden darunter die Namen der Variablen, falls möglich, angezeigt. Darunter öffnen sich gemäß der Umstellung Textboxen für die einzelnen Variablen. In diesen kann nun der Wert der Variablen eingegeben werden. Anschließend wird der Wert der Unbekannten berechnet.

# **Was können wir aus unserem ersten Projekt mitnehmen / Was haben wir daraus gelernt?**

**Man sollte wirklich Git verwenden**

Anfangs hat uns Git in seiner kompliziertheit abgeschreckt und wir dachten es wäre viel zu zeitaufwendig und umständlicher, wenn wir es für unser erstes Projekt verwenden würden. Da Live-Share sehr praktisch aussah, haben wir dieses Tool verwendet. Es stand aber ziemlich schnell fest: Es war nicht sehr praktisch. Ganz im Gegenteil: Jener, der das Projekt anfangs speichert, muss immer online sein und eine Sitzung eröffnen, damit die anderen auch arbeiten können. Ein anderes großes Problem war das kompilieren des Codes. Wenn man den Code ausführen will, muss jemand anderes kurz mit dem programmieren aufhören, damit keine Syntaxfehler auftreten. Wir versuchten dann das zu umgehen, indem jeder eine Kopie des Projekts anfertigte und lokal programmierte. Das ist natürlich auch nicht der beste Ausweg: dieses herumschieben und kopieren macht alles nur noch komplizierter. usw.

**Die Planung sollte viel ausführlicher bzw. effizienter geregelt werden**

Wir haben uns viel zu große Ziele gesetzt und haben dadurch sehr viel Zeit verloren. Bei unserer Planung fanden wir während der Programmierung viele Lücken und unbeantwortete Fragen.

# 

# **Tagebuch:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Datum** | **Damian** | **Jonas** | **Diego** |
| **09.05.22** | Zuerst gab es eine Absprache mit den anderen Gruppenmitgliedern.  Dann wurden die Klasse der Formelliste und der Formel mit einzelnen Methoden erstellt. Anschließend wurde mit der Programmierung dieser begonnen. | Beginn der Programmierung der Klassen Formule und FormulaList | Beginn der Programmierung der Klasse PreList |
| **10.05.22** | Besprechung zum Zwischenstand, anschließend wurde an der Klasse der Formelliste weiterprogrammiert und diese auch fertiggestellt. | Frühe Korrektur einiger auffallenden Programmierfehler und Erweiterung der Klasse Formula | Beginn der genaueren Planung der Programmierung der Funktionen zum Speichern/laden |
| **11.05.22** | Besprechung zum Zwischenstand (ob bereits mit der grafischen Oberfläche begonnen werden soll), anschließend Weiterprogrammierung an der Klasse der Formel und somit auch Fertigstellung der Klasse | Erweiterung der Klasse Swap. System für die Speicherung eines Strings als eine Formel die man umstellen kann muss festgelegt werden. Idee: 4 Arrays (operators, vars, values, brackets); länge aller arrays = Länge der Formel; Formel wird durchgegangen und der char wird je nach symbol in eines der 4 arrays abgespeichert. Alle anderen bleiben an diesem Index null. | Festlegen des Formats:  Datei "[Fach].csv"  [Formelname1]\n  [Formel];[FormelVariante];...|[VariablenName1];[VariablenName2];...\n  [Formelname2].... |
| **12.05.22** | Besprechungen ergaben, dass die Klasse Swap vor dem Beginn der grafischen Oberfläche fertig sein musste, anschließend Programmierung an der find\_priorities- Methode der Klasse Swap | Erstes Mal die Klasse Prelist mit der Klasse Formula Testen  ->eine Formel und alle dazugehörigen informationen in eine .csv Datei schreiben und auslesen. |  |
| **16.05.22** | Tägliche Besprechung ergab, dass das Umsetzen des automatischen Umstellen der Formel zu zeitaufwändig werden würde, anschließend wurden alle damit verbundenen Attribute gelöscht und die neue Umsetzung besprochen | Tägliche Besprechung ergab, dass das Umsetzen des automatischen Umstellen der Formel zu zeitaufwändig werden würde, anschließend wurden alle damit verbundene Attribute gelöscht und die neue Umsetzung besprochen | tägliche Besprechung ergab, dass das Umsetzen des automatischen Umstellen der Formel zu zeitaufwändig werden würde, anschließend wurden alle damit verbundenen Attribute gelöscht und die neue Umsetzung besprochen.  Debugging einiger Fehler beim ein-auslesen |
| **17.05.22** | Einteilung in die neuen Aufgaben, anschließende Programmierung der Hauptmenü-Form: zugreifen auf Ordner durch FolderBrowserDialog und Absicherung | Beginn am Design mit der ersten Form (Welcome Screen). .ico File generieren lassen um es als Icon zu verwenden (in der Taskleiste und im Formtitel) | Erste funktionierende Zwischenversion der Klasse PreList |
| **18.05.22** | tägliche Besprechung zum Zwischenstand, anschließend Weiterprogrammierung am Hauptmenü: Erstellen der Buttons für jede gefundene CSV-Datei(Formelliste) im Ordner und Skalierung dieser, Beschriftung der Buttons durch Filenames (davon mussten die Dateitypen abgetrennt werden) | Erste Form fertiggestellt und Verknüpfung zum Folderbrowserdialog hergestellt (nach diesem öffnet sich das Hauptmenü an dem Damian arbeitet. | Schreiben eines Tests für das ein-auslesen |
| **19.05.22** | tägliche Besprechung zum Zwischenstand, anschließend Weiterprogrammierung am Hauptmenü:  Hinzufügen einer Suchleiste, um Formellisten leichter zu finden | Start der Programmierung des FormulaListMenu’s.  Eigentlich nicht viel Unterschied zwischen dem was Damian in seinem Hauptmenü macht. | Schreiben eines Tests für die Funktionen |
| **20.05.22** | tägliche Besprechung zum Zwischenstand, anschließend Weiterprogrammierung am Hauptmenü:  Weiterprogrammieren an der Suchleiste, aktualisieren des Fensters, wenn eine Suche angewendet wurde und Hinzufügen eines Return Buttons, um Suchfilter wieder zu entfernen | Fertigstellen der Klasse FormulaListMenu mit Suchleiste… (wie bei Damian) | Tests würden in eine eigene Klasse verschoben |
| **23.05.22** | tägliche Besprechung zum Zwischenstand, anschließend Weiterprogrammierung am Hauptmenü:  Hinzufügen eines Buttons, welcher das Hinzufügen einer neuen Formelliste erlaubt, Erstellen einer neuen Form fürs hinzufügen | Beginn der Programmierung der Form AddFormula, die dazu dient eine Formel hinzuzufügen (Prelist-Funktionen nötig). Speichern der Formel sobald alles passt und rückkehr zum FormulaListmenu (Auch direkte Aktualisierung dieser Form!) | Neue Funktionen für das ein-auslesen einer einzelnen FormulaList + Test |
| **24.05.22** | tägliche Besprechung zum Zwischenstand, anschließend Weiterprogrammierung am Hauptmenü:  Überprüfung des Namens beim Erstellen der neuen Formelliste, anschließend Aktualisierung des Hauptmenüs (Hinzufügen des Buttons für die neuerstellte Formelliste) | Fertigstellung der Form AddFormual  -> Anzahl der Versionen die hinzugefügt werden können = Anzahl der Variablen in der eingegebenen Formel  -> Überprüfung ob alles richtig abgespeichert wird | Neuschreiben der Tests, sodass man diese separat ausführen kann |
| **25.05.22** | tägliche Besprechung zum Zwischenstand, auskommentieren des eigenen Codes mit „JavaDoc“, anschließend Bearbeiten der Formentwürfe:  skalieren der Größe, Position im Screen… | Beginn der Programmierung der letzten Form: FomulaMenu bei der eine jeweilige Formel auf eine Png-Datei gerendert wird (mit LaTeX-api). | Neue Funktion zum speichern einer einzelnen Funktion +Test |

# **Abbildungsverzeichnis:**

[Figure 1: UML-Diagramm vor Änderung 2](#_Toc105010480)

[Figure 2: UML-Diagramm nach Änderung 3](#_Toc105010481)

[Figure 3: Kanbanflow Übersicht 4](#_Toc105010482)

[Figure 4: Kanbanboard Eintrag 4](#_Toc105010483)