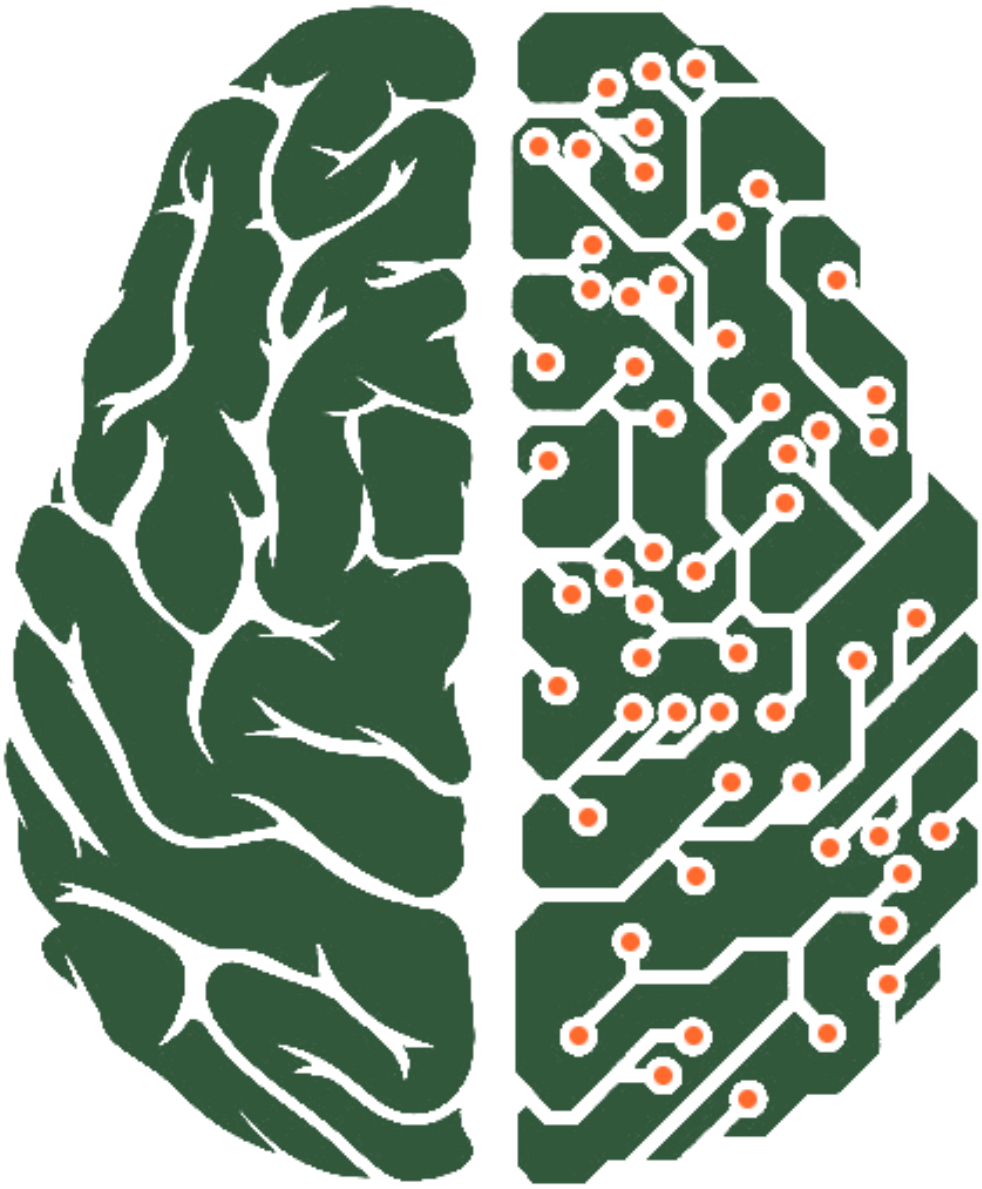
Projekt BrainTrain
Modul 226A



Autor	Version	Versionsdatum
David Zweili	1.0	04.05.2019

Inhaltsverzeichnis

1 Projekt Erläuterung _____ 3
1.1 Vorwort _____ 3
1.2 Zielsetzung _____ 3
2 Pflichtenheft _____ 4
3 Analyse _____ 5
3.1 Benutzeroberfläche _____ 5
3.2 Generieren der Rechnungen _____ 5
3.3 Archivieren der Aufgaben, Antworten und Lösungen. _ 6
3.4 Prüfen der Antworten _____ 6
3.5 Auswertung _____ 6
4 UML _____ 7
4.1 Use Case _____ 7
4.2 Klassen Diagramm _____ 8
4.3 Sequenzdiagramm _____ 10
5 Test Szenarien _____ 11
6 Verweise _____ 12

1 Projekt Erläuterung

1.1 Vorwort

Ein Teil des Moduls 226A besteht aus einer Projektarbeit. In Ihr geht es um die Umsetzung und Programmierung einer eigenen Idee. Die Programmiersprache welche wir benutzen ist Java. Es ist bisher das erste Mal das wir damit Arbeiten.

1.2 Zielsetzung

Meine Idee für das Programm ist wie der Name schon andeutet eine Anwendung auf Benutzerebene, die zum Trainieren des Gehirns gedacht ist. Die Grundidee besteht darin, dass das Programm dem Benutzer zufallsgenerierte mathematische Aufgaben vorlegt, welche er dann lösen darf. Die Aufgaben sollen als Teil eines Prüfungsblockes fungieren, welcher nach Abschluss des Blockes vom Programm benotet wird.

In den Einstellungen soll der Benutzer diverse Anpassungsmöglichkeiten haben, wie zum Beispiel welche Operatoren vorkommen sollen und mit wie hohen Zahlen die Aufgaben gestellt werden sollen.

Das Gute an dieser Idee ist, dass wenn die Zeit reicht, kann ich das Programm noch beinahe grenzenlos erweitern. Beispielsweise könnten noch verschiedene Modi eingebaut werden wie z.B. mit einem Timer möglichst viele Aufgaben in einer bestimmten Zeit zu lösen, eine Bestenliste einbauen, Die Ergebnisse ausdrucken oder ganz andere zusätzliche Aufgaben die in die Sparte Gehirn Training gehören.

2 Pflichtenheft

Muss Kriterien

- Ein eigenständiges Programm mit einer Benutzeroberfläche.
- Automatisch generierte Rechenaufgaben.
- Zu generierende Aufgaben können über die Einstellungen angepasst werden, wie zum Beispiel welche Operatoren vorkommen sollen oder die Höhe der Generierten Zahlen.
- Der Benutzer soll zu jedem Operator die Stellenanzahl der dazugehörigen Operanden bestimmen können.
- Der Benutzer kann die Anzahl der Fragen in einem Fragenblock bestimmen.
- Nach Abschluss des Fragenblockes werden alle gelösten Aufgaben überprüft und dem Benutzer in einer Auswertungsansicht präsentiert.
- Falsche gelöste Aufgaben werden vermerkt und dem Benutzer in der Auswertung mit der richtigen Lösung angezeigt.

Kann Kriterien

- Einbindung eines zweiten Modi, der sich nicht auf eine Anzahl abzuarbeitenden Fragen bezieht, sondern auf einen Timer, welcher bei Ablauf den Frageblock beendet.
- Die Dauer des Timers soll vom Benutzer definiert werden.
- Der Status des Timers soll dem Benutzer jederzeit angezeigt werden.
- Hinzufügen einer Info Seite mit einer kurzen Zusammenfassung über das Programm.

3 Analyse

3.1 Benutzeroberfläche

Das Programm besteht aus mehreren Ansichten, durch die sich der Benutzer mit Buttonklicks Navigieren kann. Um dies zu bewerkstelligen, benutze ich das Cardlayout aus der Java.awt Bibliothek.

Mit dem Cardlayout kann ich alle Ansichten dem Layout hinzufügen. Jede Ansicht erhält einen String als «Indikator».

Mit einem Befehl und dem dazugehörigen «Indikator» kann nun zwischen den verschiedenen Ansichten gesprungen werden.

3.2 Generieren der Rechnungen

Die Rechnungen die generiert werden, sollen den Einstellungen des Benutzers entsprechen. Beim Starten eines Aufgabenblockes werden die Einstellungen dem Aufgabengenerator übergeben. Dieser Speichert die Werte in einem Array ab.

Als erstes wird der Rechen Operator generiert (+, -, x, :). Dann wird überprüft ob der generierte Operator laut den Benutzereinstellungen akzeptiert ist. Wenn nicht wird ein neuer Operator generiert, bis ein passender gefunden wurde.

Nun, da der Operator gesetzt ist, werden die beiden Operanden generiert. Durch den Wert des Operators greift der Generator an die Stelle im Array, in dem die Stellenanzahl der dazugehörigen Operanden gespeichert ist. Mit diesen Stellenzahlen werden mit einer Random Funktion passende Zahlen generiert.

Die generierten Rechnungen sollen zusätzliche Kriterien erfüllen:

- Das Resultat einer Subtraktion muss im positiven Bereich liegen.
- Bei der Subtraktion darf das Ergebnis nicht 0 sein.
- Multiplikationen mit 0 oder 1 sollen nicht generiert werden.
- Division durch 0 vermeiden
- Das Resultat einer Division muss im positiven Bereich liegen.
- Das Resultat einer Division darf nicht 1 sein.
- Das Resultat einer Subtraktion muss eine Ganzzahl sein.

3.3 Archivieren der Aufgaben, Antworten und Lösungen.

All die gelösten Rechnungen müssen irgendwo gespeichert werden, damit sie später ausgewertet werden können. Dies wird durch ein zweidimensionales Array gelöst, welches die drei Faktoren (Aufgabe, Antwort, Lösung) als «Packet» archiviert. Folgende Tabelle soll die Idee verdeutlichen:

	Index	0	1	2	3	4	…
Aufgabe	0	15 + 20	45 - 8	3 x 14	32 : 4	7 - 3	…
Antwort	1	35	38	42	8	4	…
Lösung	2	35	37	42	8	4	…

3.4 Prüfen der Antworten

Wenn der Aufgabenblock abgearbeitet ist, wird das Array mit den Aufgaben dem Aufgaben Prüfer übergeben. Dieser erstellt eine Liste als String, mit allen Aufgaben.

Bei jeder Aufgabe prüft er, ob die Antwort mit der Lösung übereinstimmt. Wenn eine Rechnung falsch gelöst wurde, wird die Jeweilige Aufgabe Optisch hervorgehoben und mit der korrekten Lösung ergänzt.

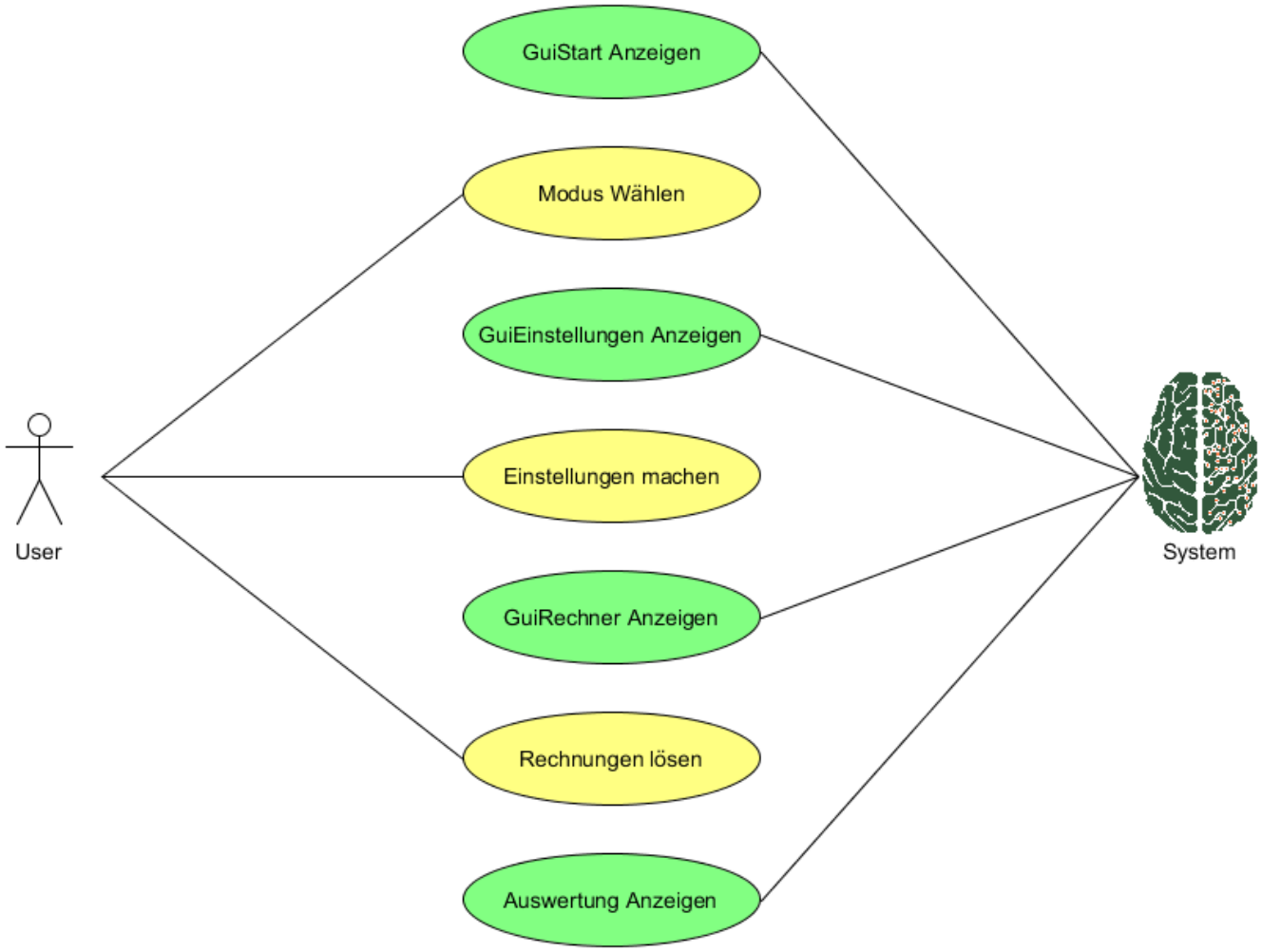
3.5 Auswertung

In der Auswertung sollen dem Benutzer folgende Daten präsentiert werden:

- Liste mit allen Fragen, Antworten und Korrekturen.
- Prozentuale Angabe der korrekt gelösten Fragen.
- Anzahl der Fragen im Fragenblock
- Übersicht der getätigten Einstellungen für diesen Fragenblock.

4 UML

4.1 Use Case



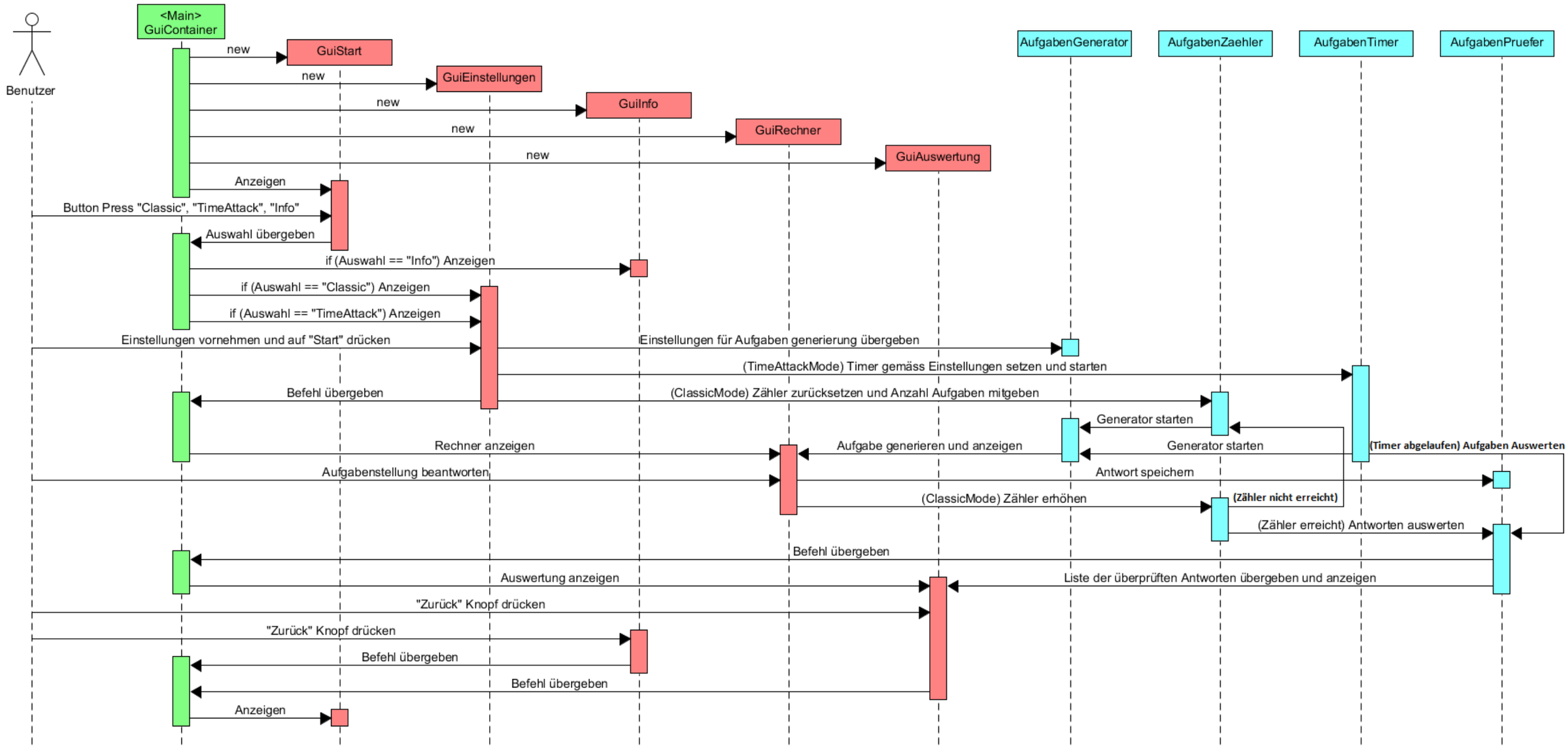
4.2 Klassen Diagramm

Berechnungs-Klassen

Gui-Klassen


GuiContainer
frame = JFrame
guiStart = GuiStart
guiInfo = GuiInfo
guiEinstellungen = GuiEinstellungen
guiRechner = GuiRechner
guiAuswertung = GuiAuswertung
cl = CardLayout
panelContainer = JPanel
+ changePanel(String panel)
GuiStart
implements ActionListener
layersStart = JLayeredPane
buttonClassic = JButton
buttonTimeAttack = JButton
buttonInfo = JButton
overlayStartJLabel = JLabel
actionPerformed(ActionEvent event)
GuiRechner
implements ActionListener
layersRechner = JLayeredPane
buttonBack = JButton
labelStatus = JLabel
labelAufgabe = JLabel
overlayRechnerJLabel = JLabel
textAreaLoesung = JTextField
consoleFrontGreen = Color
consoleBackGreen = Color
actionPerformed(ActionEvent event)
+ setStatus(String statusString)
+ setAufgabe(String aufgabe)
GuiInfo
implements ActionListener
layersInfo = JLayeredPane
buttonBack = JButton
buttonSpenden = JButton
spendeSlider = JSlider
overlayInfo = JLabel
cashEmoji = JLabel
sliderBackGrey = Color
actionPerformed(ActionEvent event)
GuiEinstellungen
implements ActionListener
buttonGoClassic = JButton
buttonGoTimeAttack = JButton
buttonBack = JButton
overlayClassicJLabel = JLabel
overlayTimeAttackJLabel = JLabel
textFieldAnzahlFragen = JTextField
textFieldDauer = JTextField
comboBoxListe = String[]
comboBoxListeSpez = String[]
checkBoxPlus = JCheckBox
checkBoxMinus = JCheckBox
checkBoxMal = JCheckBox
checkBoxGeteilt = JCheckBox
comboBoxPlusErsteStelle = JComboBox<?>
comboBoxPlusZweiteStelle = JComboBox<?>
comboBoxMinusErsteStelle = JComboBox<?>
comboBoxMinusZweiteStelle = JComboBox<?>
comboBoxMalErsteStelle = JComboBox<?>
comboBoxMalZweiteStelle = JComboBox<?>
comboBoxGeteiltErsteStelle = JComboBox<?>
comboBoxGeteiltZweiteStelle = JComboBox<?>
classicMode = boolean
anzahlStellenArray = String[][]
consoleFrontGreen = Color
consoleBackGreen = Color
checkBoxFrontGrey = Color
checkBoxBackGrey = Color
actionPerformed(ActionEvent event)
+ getAuswahlCheckboxenArray()
+ getAnzahlStellenArray()
+ setClassic()
+ setTimeAttack()
+ getEinstellungen()
GuiAuswertung
implements ActionListener
layersAuswertung = JLayeredPane
auswertungProzentJLabel = JLabel
einstellungenJLabel = JLabel
auswertungListeLabel = JLabel
overlayAuswertungJLabel = JLabel
scrollListe = JScrollPane
buttonBack = JButton
actionPerformed(ActionEvent event)
+ setAuswertung(String liste, String prozent)

4.3 Sequenzdiagramm



5 Test Szenarien

Der unterstrichene Text zeigt an, in welcher Ansicht das der Prozess getestet wurde. Text in «Anführungszeichen» weisen auf den jeweiligen Buttonpress hin.

Prozess	Erwartung	Ausgabe	Fehler	Korrektur	Iteration
Programm starten	Programm startet (GuiStart)	Programm startet (GuiStart)	-	-	1.0
GuiStart «Info»	Anzeige (GuiInfo)	Anzeige (GuiInfo)	-	-	1.0
GuiStart «Classic»	Anzeige (GuiEinstellungen im Classic Mode)	Anzeige (GuiEinstellungen im Classic Mode)	-	-	1.0
GuiEinstellungen KeinOperator gewählt «Start»	Fehlermeldung: Kein Operator gewählt.	Fehlermeldung: Kein Operator gewählt.	-	-	1.0
GuiEinstellungen Geteilt, erste Zahl weniger Stellen als zweite Zahl «Start»	Fehlermeldung: Zweite Zahl muss kleiner sein.	Fehlermeldung: Zweite Zahl muss kleiner sein.	-	-	1.0
GuiEinstellungen (TimeAttack Mode) Eingabefeld «Minuten» mit mehr als zwei Kommastellen «Start»	Kommastellen werden auf zwei gekürzt. Siehe Anzeige (GuiAuswertung)	Kommastellen werden auf zwei gekürzt. Siehe Anzeige (GuiAuswertung)	-	-	1.0
GuiRechner (TimeAttack Mode) Timer läuft ab ohne eine Aufgabe gelöst.	Anzeige (GuiAuswertung) Auswertungsfeld ist leer.	Anzeige (GuiAuswertung) Auswertungsfeld ist leer.	-	-	1.0
Tester: David Zweili	Testdatum: 04.05.2019		Umgebung/System: Windows 10		

6 Verweise

Aus den folgenden Quellen habe ich meine Grundlagenkenntnisse und Informationen für mein Projekt erhalten:

Java

https://www.java-tutorial.org/

https://stackoverflow.com/

https://coderanch.com/c/java

html

https://www.w3schools.com/

https://www.toptal.com/designers/htmlarrows/math/