Software Entwicklung 2

Wintersemester 2020/2021

MI7/MM7

Math Trainer

Firaz Ilhan, fi007@hdm-stuttgart.de
Maximilian Dolbaum, md127@hdm-stuttgart.de
Jens Schlegel, js414@hdm-stuttgart.de

https://github.com/Firaz-Ilhan/math-trainer.git

1. Kurzbeschreibung

Bei unserem Projekt handelt es sich um einen Mathe Trainer um einfache basis Rechenarten in jeweiligen Schwierigkeitsgraden zu üben.

Man startet ein “New Game” und hat die Option alle oder nur ausgewählte Rechenarten zu benutzen. Ebenfalls kann man dann einen Schwierigkeitsgrad von Beginner/Medium/Hard auswählen.

Aus den ausgewählten Rechenarten wird zufällig eine Aufgabe erstellt, welche der Spieler dann beantworten oder überspringen kann. Oben rechts im Fenster läuft ein Timer um die Gesamtbearbeitungszeit zu veranschaulichen.

Zum beenden der Einheit werden Informationen über die gelösten Prozent je Rechenart angezeigt. Man kann sich dann die Falschen antworten und gesamten Statistiken anzeigen lassen.

2. Startklasse

Die Main-Methode befindet sich in der Klasse **Launcher.**

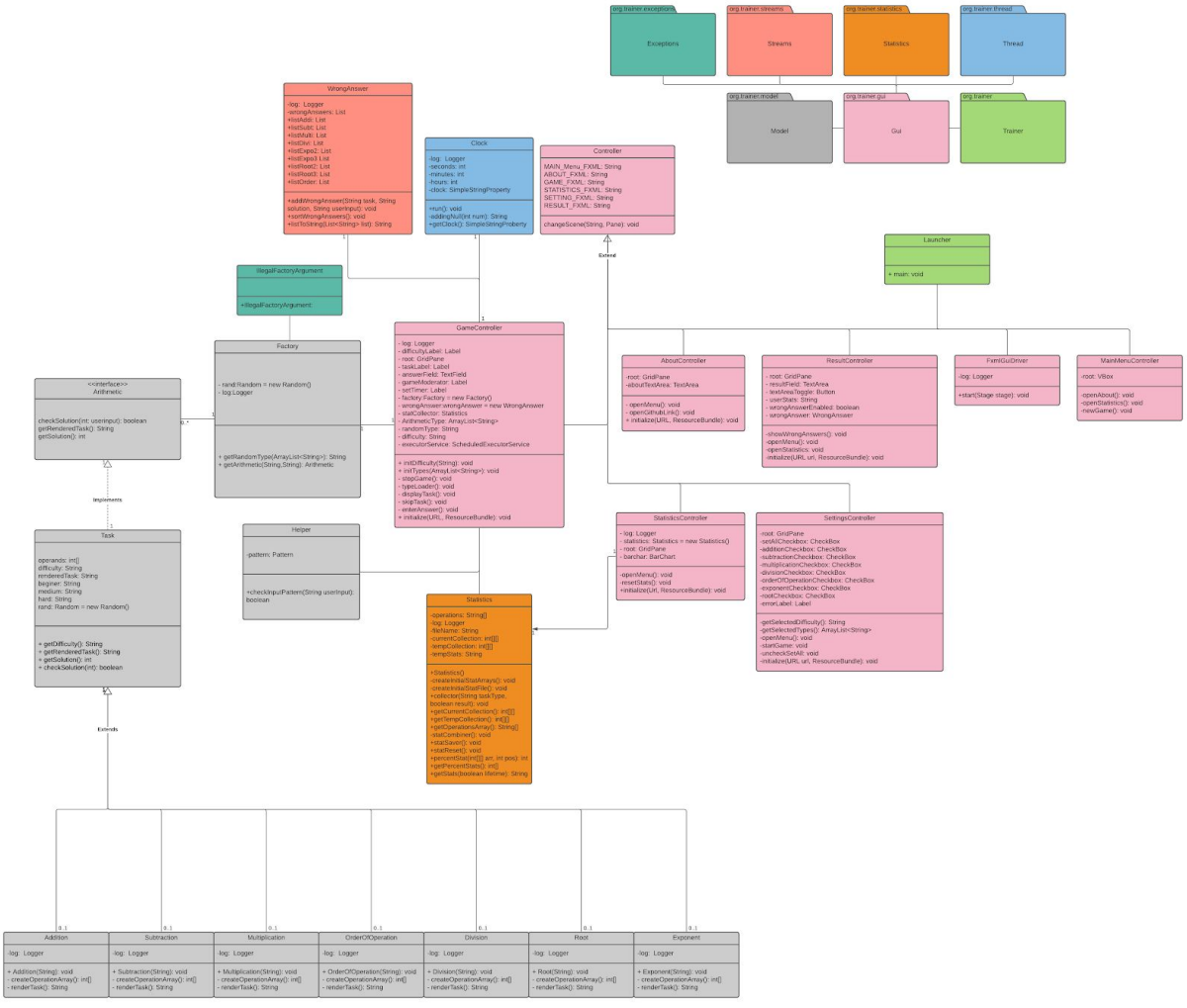
3. Besonderheiten

- Mit *mvn package* kann eine cross-platform jar für Windows, Linux und mac-os erstellt werden.
- Zurzeit gibt es 7 Rechenarten mit je 3 Schwierigkeiten.
- Wir nutzen JDK 11
- Wenn der Ordner schreibgeschützt ist und die stats.txt nicht erstellt werden kann gibt es eine **alert** Fehlermeldung (**GameController**)

4. UML-Klassendiagramm

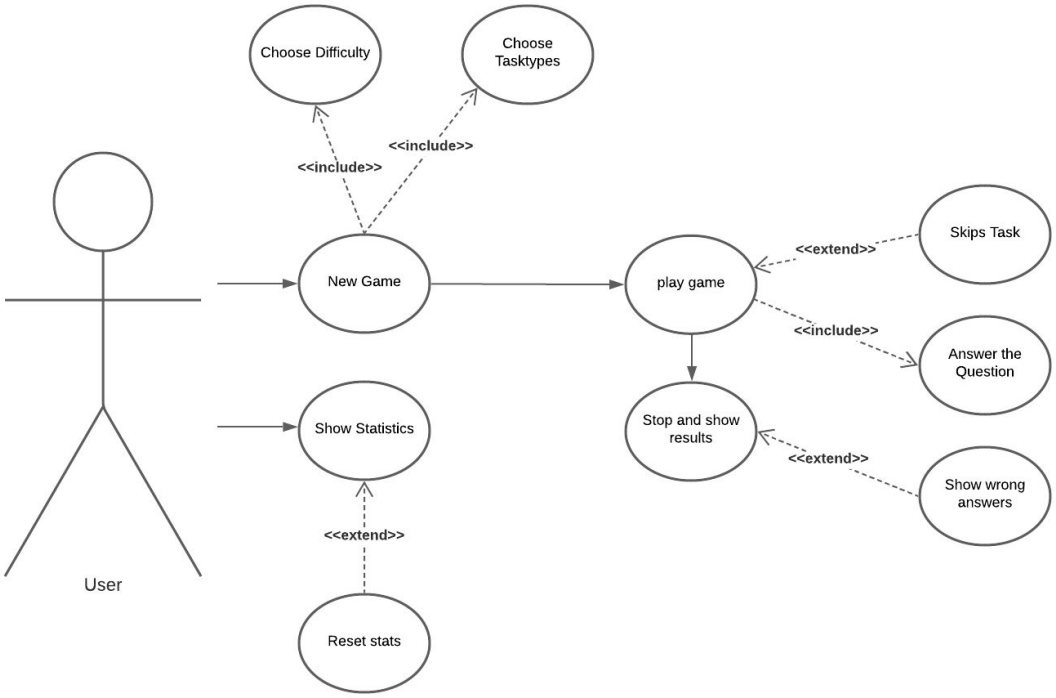
<u>Klassendiagramm:</u>

Als PDF im Repository erhältlich.



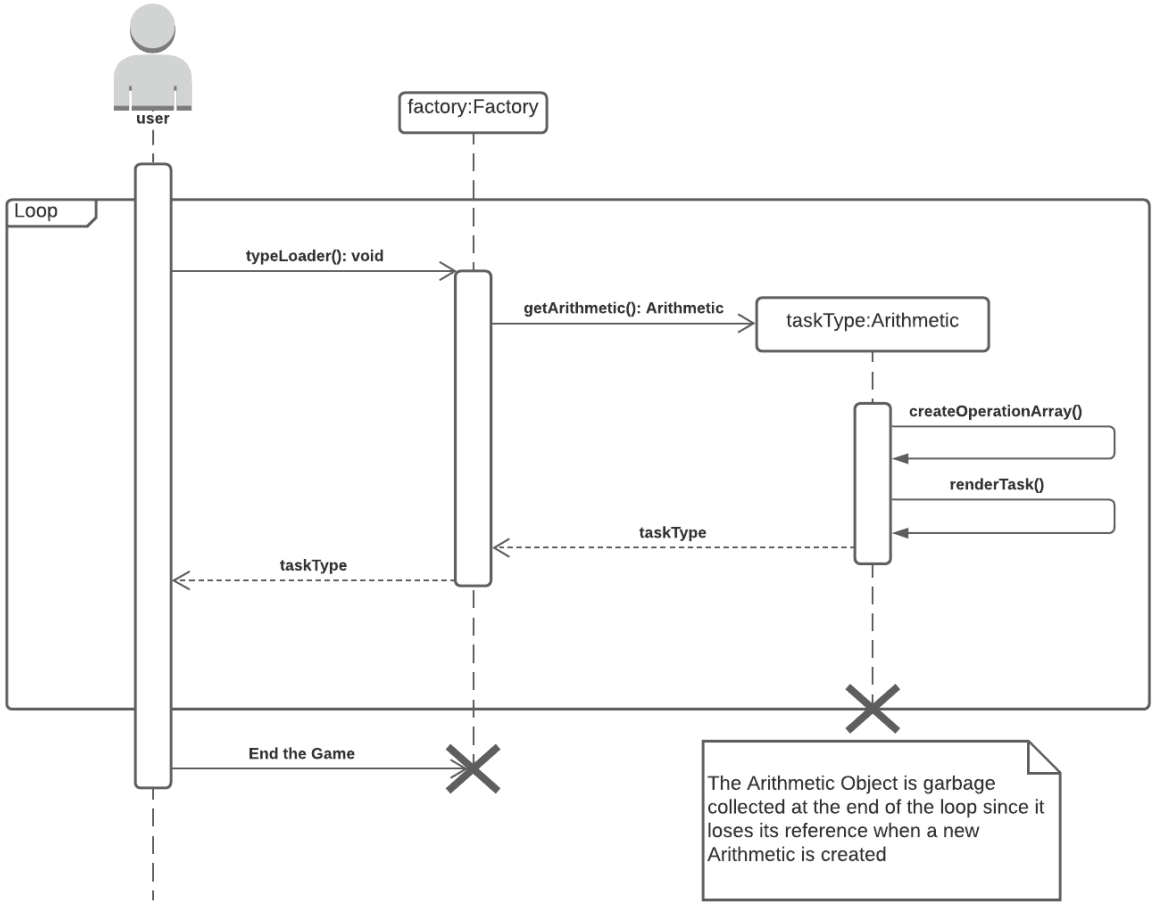
Use Case Diagram:

Als PDF im Repository erhältlich.



Sequenz-Diagram:

Als PDF im Repository erhältlich.



5. Stellungnahme

Architektur:

Packages: Die Packages erlauben einfachere Navigation zwischen zusammenhängenden Klassen. Die Packages sind:
org.trainer. ...
...**exceptions** (Eigene Exception)
...**gui** (Controller der Anwendung)
...**model** (Rechenarten und Helper)
...**statistics** (Statistics)
...**streams** (WrongAnswer)
...**thread** (Clock)

Die **Launcher** Klasse um das Programm zu starten ist direkt im org.trainer package.

Interfaces: **Arithmetic** wird von **Task** implementiert. Dies implementiert **Arithmetic** dann für alle Kindklassen von **Task**.

Vererbung:

Elternklasse	Kindklasse
Task	Addition, Subtraction, Multiplication, Division, Exponent, Root, OrderOfOperation
Controller	AboutController, GameController, MainMenuController, ResultController,

	SettingController, StatisticsController

Clean Code:

Static methode ist in der **model.Helper** Klasse angebracht da diese Funktion universell aufgerufen wird.
Public methoden und Variablen sind nur da wo es nötig ist, eingesetzt.
Alle nicht primitiven Datentypen außer Strings werden wenn, nur mit **.clone()** übergeben.

Tests:

Die methoden der Klasse Statistics, Model, Factory haben *JUnit* Tests im Ordner **test**. Negative Tests sind in **FactoryTest** in der **test_negative** methode verwendet.

GUI (JavaFX):

Die Views haben wir mit dem **Scene Builder** erstellt, welche sich in **resources/fxml** befinden. Insgesamt haben wir 6 Views mit responsive Design.

Logging/Exceptions:

- Exceptions werden mit **.error** geloggt
- Szenenwechsel, Aufgaben, Eingaben vom Benutzer und die Statistiken werden mit **.info** geloggt

UML:
Mithilfe von lucid.app/lucidchart haben wir vor beginn der Programmierung sowohl Klassen als auch Use-Case Diagram erstellt.
Am Ende des Projekts haben wir diese nochmal überholt da Ideen verworfen wurden und neue entstanden.

Threads:
Clock-Timer läuft während eines Games oben rechts im Fenster. Es gibt die Zeit an, wie lange der User bereits im Game ist. **Clock.start()** wird jede Sekunde mithilfe von **ScheduledExecutorService** im **GameController** aufgerufen. Bei jedem Aufruf erhöht sich die Variable **seconds** um 1. Wenn 60 Sekunden erreicht sind, dann erhöht sich die Variable minute um 1. Das Gleiche passiert auch bei den Stunden.

Streams und Lambda-Funktionen:
Bei jeder falsche Antwort vom Benutzer wird die Aufgabe, die Antwort und die Lösung zu der Liste **wrongAnswers** in der Klasse **WrongAnswer** hinzugefügt. Diese Liste wird mithilfe von Streams nach den verschiedenen Rechenarten sortiert und zu der jeweiligen Liste hinzugefügt. Damit der Benutzer zu einer Rechenart seine falschen Antworten anschauen kann.
Fast jeder Gui Controller hat lambda-funktionen für Tastaturkürzel.

Factories:
Die Factory erstellt ein Arithmetic mit **getArithmetic()**. Die Methode **getRandomType()** gibt ein zufälligen typ zurück.

Dokumentation:
Wir haben JavaDoc erstellt und die Funktionalität der methoden beschrieben. Manchmal haben wir noch inline Kommentare eingefügt um Funktionen besser verständlich zu machen.
Selbstbewertung:

Thema	**Punkte**
Architektur	3
Clean Code	3
Dokumentation	3
Tests	3
UML	3
GUI	3
Logging/Exceptions	3
Threads	3(thread.start() is enough for the online semester)
Streams	3
Nachdenkzettel	3
Points:	30/30
Actual Points:	