Hochschule der Medien
Mobile Medien
Softwareentwicklung 2
Sommersemester 2021

LOVE CALCULATOR

university editon

Softwareentwicklung 2 Projektaufgabe

LoveCalculator – University Edition

https://gitlab.mi.hdm-stuttgart.de/vv014/lovecalculator

Vorgelegt von:
Simeon Schulz (ss534)
Sascha Schmidt (ss532)
Patrik Kinderknecht (pk090)
Vivien Volpert (vv014)

Abgabedatum: 31.07.2021
Prüfer: Prof. Walter Kriha

KURZBESCHREIBUNG

Was ist denn der LoveCalculator University Edition?
„Du bist verzweifelter Single, unglücklich in einer Beziehung oder einfach auf der Suche nach der großen Liebe? Dann bist du hier genau richtig! Finde hier heraus wer das fehlende Puzzleteil in deinem Leben ist.“

Mit dem LoveCalculator findest du also auf einfache Art und Weise heraus, ob dein Schwarm zu dir passt oder nicht.
Unterteilt wird im Spiel zwischen zwei verschieden Modi: Classic und Advanced. Beim Classic-Mode werden die Namen der Liebenden angegeben und deren Studiengang. So kann in kürzester Zeit die große Liebe gefunden werden. Beim Advanced-Mode hingegen geht es ein bisschen mehr in die Tiefe. Mit der Liebe zum Detail werden innerhalb von 6 vorgegeben Fragen mehr Infos abgefragt.
Um Fragen und alte Sucherergebnisse festzuhalten, kann der User sich ein Profil anlegen. Außerdem gibt es eine Rangliste, mit der sich die User untereinander vergleichen können.

Die Main-Methode befindet sich in der `FxmlGuiDriver` Klasse.

Besonderheiten

Gibt es Dinge, die noch nicht so funktionieren wie wir wollen? Dann hier beschreiben

Damit die Datenbankverbindung richtig funktioniert, muss man erst in seinem IntelliJ die Jar-Datei `mysql-connector-java-8.0.26.jar einbinden.` Mehr Informationen dazu in der README.md Datei.

Leider gab es öfter Probleme mit JavaFX, das ganze System hat immer mal wieder nicht richtig funktioniert (vor allem der SceneBuilder!) wir hatten öfters Abstürze und konnten teils Funktionen nicht nutzen, die wir dann händisch im Code umsetzen mussten.

Manchmal hat unser Programmierwissen noch nicht gereicht, um Methoden richtig auszulagern und dynamisch aufzurufen. Vor allem mit JavaFX kam es hier ein paar Mal zu Problemen, weshalb wir manche Methoden doppelt angelegt haben. Da jedes FXML nur einen Controller haben kann, haben wir es nicht hinbekommen, die Methoden aus anderen Klassen für das JavaFX aufzurufen (Beispiel: `profileScene()` oder `playScene()`)

Erweiterungen:

Unsere Anwendung ermöglicht es, eigene Fragen und Antworten hinzuzufügen. Dies funktioniert direkt in der `question.csv` und `answers.csv`.

Schaut man sich einmal kurz dein Aufbau dieser Dateien an,

Auszug aus der `question.csv`:

```
Wie heißt du?;Wie heißt dein Schwarm?
Was studierst du?;Was studiert deine Flamme?
```

Auszug aus der `answers.csv`:

```
textfield;name
radiobutton;Super!;geht so;naja;frag nicht
```

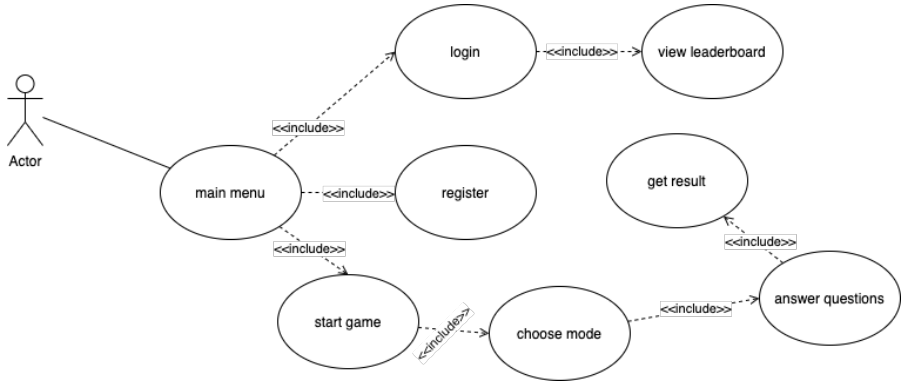
sieht man schnell das diese selbsterklärend aufgebaut sind. Somit haben wir eine leichte Erweiterbarkeit ermöglicht.
Die Fragen trennt man mit einem Semikolon. Bei den Antworten muss man zusätzlich die Art der Ausgabe hinzufügen. Hier kann man zwischen *Texfield, Radiobutton & Checkboxen* wechseln. Momentan haben wir die Texfields und Radiobuttons in Verwendung.

Weitere Erweiterungen, die man in Zukunft umsetzen könnte, wären das Integrieren von neuen Modi, oder beispielsweise das Hinzufügen von Freunden wie auch das Erstellen von eigens kreierten Fragen.

UML

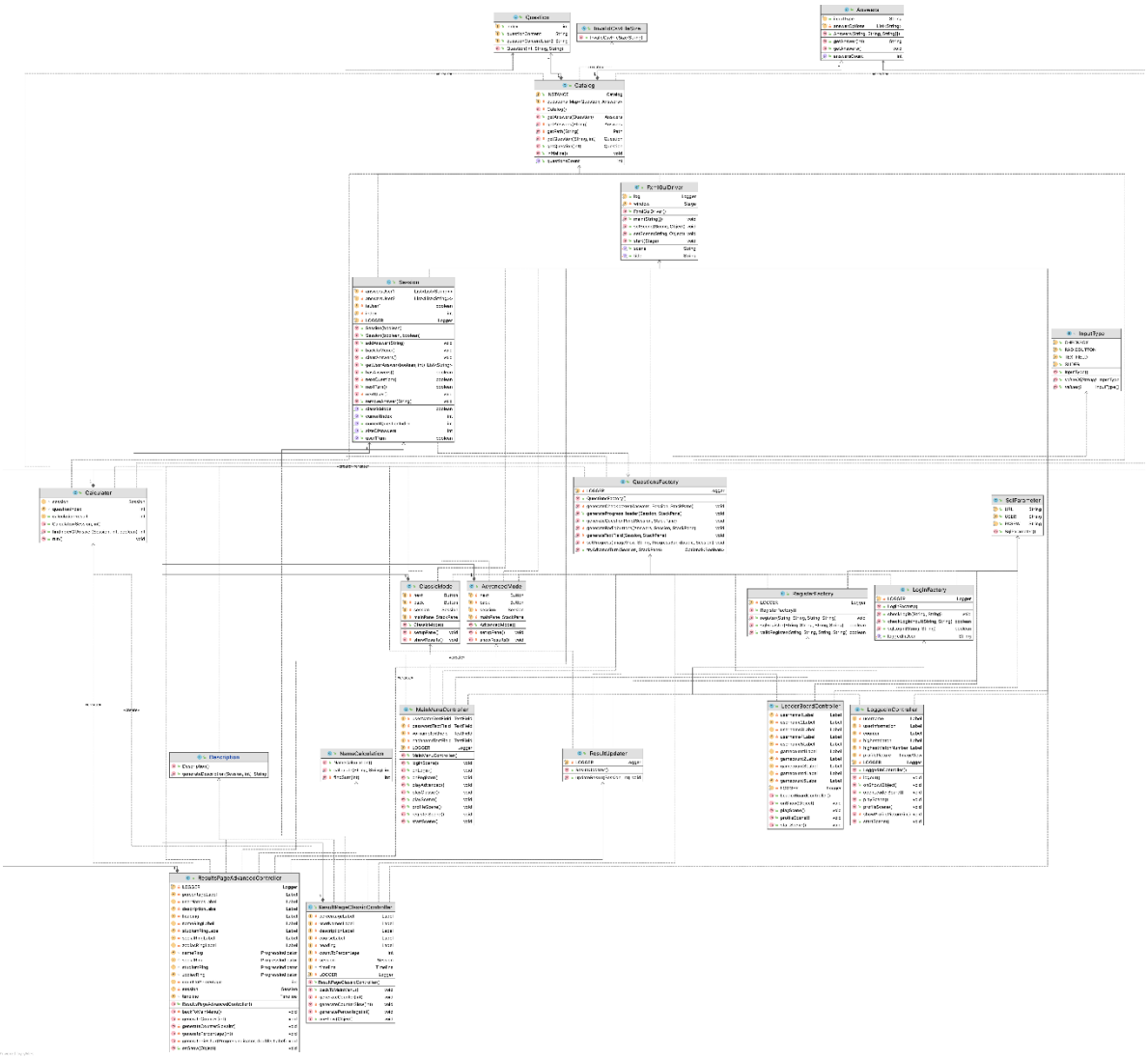
Use-Case-Diagramm

Dateipfad: Dokumentation/UML/UseCaseDiagramm.png



Klassendiagramm

Dateipfad: Dokumentation/UML/Klassendiagramm.png



STELLUNGNAHMEN

Architektur

Enum

Das Enum hält den Inputtype fest, also den Typ der Antwort der jeweiligen Fragen Slider, (Checkbox, Radiobutton und Textfield). Außerdem sorgt er für ein risikofreies Einlesen des Inputtypes aus der CSV-Datei.
Dateipfad: src/main/java/de/hdm_stuttgart/love_calculator/Gui/GuiController/InputType.java

Interface

Hier wird die onShow-Methode implementiert, damit Variablen schon beim Start der Szene geändert werden können. Das ist wichtig für die Profil-Szene, die Leaderboard-Szene und den Result-Szenen. Der Methode wird ein Parameter, Objekt Namens `argument` übergeben. Das Objekt `argument` muss eine Instanz der Session sein.
Dateipfad: src/main/java/de/hdm_stuttgart/love_calculator/Gui/Navigatable.java

Vererbung

Vererbungen haben wir im Classic Modus sowie im Advanced Modus genutzt, die beide von `Scene` erben.
Ein weiteres Vererbungsbeispiel kommt bei dem `Calculator` vor, der von der Superklasse Threads erbt.

Ordnerstrukutur

Im Package `de.hdm_stuttgart.love_calculator` befinden sich fünf weitere Unterordner.
Calculator: Enthält Klassen, in denen die Berechnung und Auswertung der Fragen stattfindet. Außerdem wird hier dem jeweiligen Prozentsatz ein passender Spruch für das Ergebnis zugewiesen.
Dateipfad: src/main/java/de/hdm_stuttgart/love_calculator/Calculator

Exception: Enthält unsere eigene Exception `InvalidCsvFileSize.`
Dateipfad: src/main/java/de/hdm_stuttgart/love_calculator/Exception

Game: Hier werden die Antworten und Fragen in der Map `questions` abgespeichert (`Catalog`-Klasse) und hier werden auch Answer- und Qustions-Objekte erstellt (`Answers` und `Questions`-Klasse). Zudem befinden sich hier die einzelnen Methoden für eine

Session, welche den Spielablauf kontrollieren und die Rahmenbedingungen setzen (`Session`-Klasse)
Dateipfad: src/main/java/de/hdm_stuttgart/love_calculator/game

Gui: In diesem Package befinden sich die nachfolgenden 3 Packages, zusätzlich befindet sich hier die Mainklasse `FXMLGuiDriver`, das Interface in `Navigatable`, ein Enum in `InpuType` und eine Methode, um einen DialogAlert in `AlertDialogue` aufzurufen.
Dateipfad: src/main/java/de/hdm_stuttgart/love_calculator/Gui

GameModes: Enthält die Klassen zum Start des Classic- bzw. Advanced Modus.
Dateipfad: src/main/java/de/hdm_stuttgart/love_calculator/Gui/GameModes

GuiController: Beinhaltet die Methoden für den Szenenwechsel (setScene) und die onShow Methoden für Änderung von Variablen beim Start einer Szene.
Dateipfad: src/main/java/de/hdm_stuttgart/love_calculator/Gui/GuiController

GuiFactory: Hier befinden sich die Methoden, die den Szenen die Logik geben. In der `QuestionFactory` findet der ganze Ablauf der Spielmodi statt, also was passiert, wenn man auf den „Weiter-Button“ klickt, welche Frage erscheinen soll und mit welchem Inputtype. Die Fragen & Antworten werden hier dynamisch in das FX reingeladen.
In den `Login`- & `RegisterFactory` Klassen findet zudem ein Datenbankaufruf statt, um die eingebenen Daten des Users mit der Datenbank abzugleichen. Mehr Informationen zu den Factories gibt es weiter unten in der Doku.
Dateipfad: src/main/java/de/hdm_stuttgart/love_calculator/Gui/GuiFactory

Sql: Ist ein extra Package unabhängig von Gui. Hier sind zum einen die Zugriffsdaten für die Datenbank hinterlegt und zum anderen eine Methode mit Datenbankaufruf, die nach erfolgreichem Spiel den Spielcounter in der Datenbank um 1 erhöht und ggf. bei höchstem Match dieses Match auch in der Datenbank ersetzt.
Dateipfad: src/main/java/de/hdm_stuttgart/love_calculator/Sql

Clean Code

Variablen sind immer, falls es möglich ist, als private final deklariert und Methoden als private. Redundanter Code wurde bestmöglich vermieden. Einen Punkt haben wir uns in der Selbstbewertung abgezogen, da wir teilweise getter/setter und static genutzt haben. Trotzdem haben wir versucht, diese weitestgehend zu vermeiden

Tests

Um unsere Methoden zu testen haben wir im Verzeichnis src/test für die wichtigsten Methoden JUnit Tests angelegt und mit sinnvollen Testdaten gefüllt.

GUI (JavaFX)

Das GUI wurde größtenteils mit dem SceneBuilder erstellt. Die dynamische Generierung der Spieleroberfläche, sprich Classic- und Advanced-Modus, wurde komplett im Code ohne die Hilfe des SceneBuilders umgesetzt. Dies ermöglicht uns eine flexible Generierung der Spieloberfläche, die auch einfach durch neue Fragen und Antworten in den CSV Dateien erweitert werden kann.
Dateipfad: src/main/resources/fxml

Logging/Exceptions

Logging

log.debug: Für generelle Informationen wie beim Aufruf von Methoden.
log.error: Für die Zuweisung des Ergebnis-Spruches bzw. wenn allgemein die Auswahl nicht verfügbar ist.
log.fatal: Für gravierende Fehler, die eine normale Nutzung des Programmes unmöglich machen.
log.info: Für wichtige Funktionen, wie beispielsweise das Laden von Fragen und Antworten.

Exception

`InvalidCsvFileSize` – wie der Name schon verrät, überprüft die Exception die Answers- und Questions-CSV-Dateien auf ihre Größe. Wenn beide Dateien unterschiedlich viele Tupel haben, wird die Exception aufgerufen.

UML

Durch die Empfehlungen von Herrn Prof. Kriha haben wir unser Projekt mit der Erstellung des UseCase-Diagrammes begonnen. So konnten wir schnell aufgefasste Ideen und Anregungen festhalten. Dies war eine wichtige Vorrausetzung, sodass alle Beteiligten des Projekts auf dem gleichen Stand der Dinge waren und blieben.
Nach der Fertigstellung des UseCase-Diagrammes haben wir mit der Erstellung des Klassen-Diagrammes begonnen. Im Laufe der Entwicklung kamen immer wieder neue Klassen hinzu, die regelmäßig und problemlos im Diagramm ergänzt wurden konnten. Auch

mit der Hilfe des Klassen-Diagrammes waren alle Teammitglieder stets auf dem gleichen Stand.

Threads

Wir verwenden Threads zur Berechnung der Prozentsätze aus den Fragen von Studium, Sternzeichen & Party. Die Threads werden in der Klasse `Calculator` erstellt und in den Klassen `resultsPageClassicController` und `resultsPageAdvancedController` aufgerufen.

Streams und Lambda-Funktionen

Zum einen Nutzen wir Streams in der `RegisterFactory` um die gespeicherten Werte der Registrierung mit forEach auszulesen. Zum anderen verwenden wir Streams im `LeaderBoardController` um die Daten der Datenbank in der Scene zu sortieren. Für beide Einsatzgebiete haben wir die Lambda-Expression benutzt.

Factories

Questions-Factory

Die Spieloberfläche für den Classic- und Advanced-Modus wird dynamisch aus den Answer & Question Objekten generiert.
Dateipfad:
src/main/java/de/hdm_stuttgart/love_calculator/gui/GuiController/QuestionsFactory.java

Register-Factory

Die Register-Factory fragt den User-Input der Registrierung ab und registriert neue Nutzer in der Datenbank.
Dateipfad:
src/main/java/de/hdm_stuttgart/love_calculator/gui/GuiController/RegisterFactory.java

Login-Factory

Überprüft den User-Input des Logins und loggt den User ein.
Dateipfad:
src/main/java/de/hdm_stuttgart/love_calculator/gui/GuiController/LoginFactory.java

BEWERTUNGSBOGEN

Dateipfad: Dokumentation/ProjektnotenTemplateLoveCalculator.xlsx

Vorname	Simeon	Sascha	Patrik	Vivien
Nachname	Schulz	Schmidt	Kinderknecht	Volpert
Kürzel	ss534	ss532	pk090	vv014
Martikelnummer	41497	41492	40940	41022
Projekt	LoveCalculator	LoveCalculator	LoveCalculator	LoveCalculator
Architektur	3	3	3	3
Clean Code	2	2	2	2
Doku	3	3	3	3
Tests	2	2	2	2
GUI	3	3	3	3
Logging/Exceptions	3	3	3	3
UML	3	3	3	3
Threads	3	3	3	3
Streams	3	3	3	3
Profiling	3	3	3	3
Summe	28,00	28,00	28,00	28,00
Kommentar	sh. Anmerkungen in Doku	sh. Anmerkungen in Doku	sh. Anmerkungen in Doku	sh. Anmerkungen in Doku
Note	1,30	1,30	1,30	1,30

NACHDENKZETTEL

Die Nachdenkzettel

01_GitGitlab.pdf
02_UML.pdf
03_Collections.pdf
04_BeziehungVererbung.pdf
05_InterfacesUndSoftwareArchitecture.pdf
06_Logging.pdf
07_CleanCode.pdf
08_GUI.pdf
09_StreamsProcessing.pdf
10_Threads.pdf

befinden sich ausgefüllt und beantwortet im Repository.

Dateipfad: Dokumentation/Nachdenkzettel