Quiz Projekt

Gruppenmitglieder:

Liliya Ivanova li006
Zsanet Süli zs005
Feride Usabaev fu004
Sanja Perovic sp079

Pfad zum Repository:
https://gitlab.mi.hdm-stuttgart.de/zs005/QuizProjekt

1. *Kurzbeschreibung:*

   Unser Projekt ist ein Quiz, welches folgende Spielarten anbietet: Einzel oder Multiplay. Wenn man sich für eine Art des Spieles entschieden hat, muss der User einen Namen eingeben. Beim Multiplayer können maximal 4 Personen mitspielen. Anschließend steht es dem Spieler frei zur Auswahl, sich zwischen dem klassischen Spiel mit 4 Antwortmöglichkeiten oder eigenem Input von Antworten zu entscheiden. Egal wie sich der Spieler entschiedet, er muss im nächsten Schritt eine von vier Kategorien auswählen: Politik/Geschichte, IT, Unterhaltung, Geografie. Mit dem Klick auf eine der Kategorien, startet das Spiel und der Spieler muss 10 Fragen beantworten. Dabei gibt jede richtige Antwort 1 Punkt. Ist das Spiel zu Ende, bekommt der Spieler seine erreichten Punkte angezeigt.

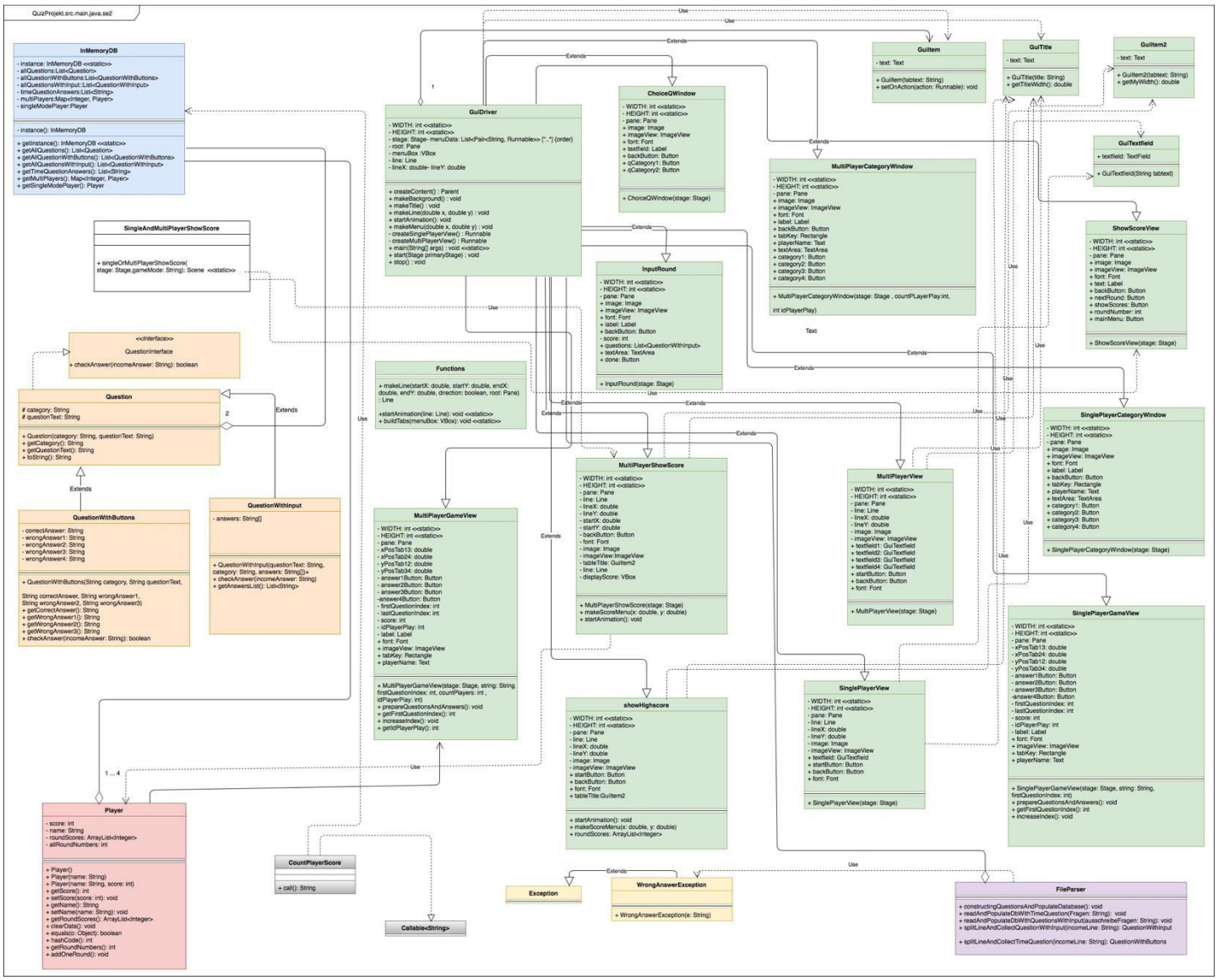
2. *Startklasse:*

   Die Klasse “GuiDriver” enthält die main – Methode unseres Projekts.

3. *Besonderheiten:*

   Folgendes funktioniert nicht richtig:
   Wenn beim Spiel die Buttons sehr schnell hintereinander angeklickt werden, kann es zu einem „Bug“ kommen. Wenn dies auftritt, kann man nicht mehr weiterspielen.

Unser UML – Diagramm:



Stellungnahmen:

Interfaces/Vererbung:
Das Interface wurde im package „questions" in dem Interface „IquestionInterface" deklariert und in folgenden Klassen implementiert: QuestionswithInput, QuestionswithButtons.

Package-Struktur:
Wir haben uns dafür entschieden, möglichst alle Klassen logisch zu gruppieren und in eigene Packages zu setzen.
Deshalb haben wir viele Packages bzw. für manche Klassen ein eigenes Package, damit falls das Spiel später erweitert werden würde, eine logische Struktur bestehen bleibt.
Wir verwenden folgende Packages: db, exception, factory, model, thread und parser, welche jeweils nur eine Klasse/File beinhalten. Das Package gui enthält alle für die GUI notwendigen Klassen sowie die in „Main"-Logik. Das Package questions enthält unser Interface und die entsprechenden Implementierungen.

Exceptions:
Wir verwenden eine Exception, falls im Konstruktor von QuestionWithButtons die Antworten ein leerer String sind.
Dies ist besonders wichtig, weil wir die Fragen aus einem File parsen. Hierzu verwenden wir eine eigene Exception WrongAnswerException. Zudem verwenden wir Exceptions im Konstruktor von QuestionWithInput.

Graphische Oberfläche (JavaFX):
Die GUI wurde komplett in der GuiDriver Klasse verwaltet, dort wurden z.B. event handler und stage erstellt. Außerdem wurden hier auch alle Buttons und Textfelder erstellt, die benötigt werden. Es wurden mehrere weitere Fenster erstellt, die der Spieler zu sehen bekommt, z.B. wenn er seinen Namen eingibt oder die Kategorie auswählen darf und dann später die Fragen erscheinen.

Logging:
Unser Logging Konzept verwendet Log4J. Dabei setzen wir auf verschiedenen Log-Stufen, wir verwenden Info für allgemeine Ereignisse, wie wichtige Methodenaufrufe etc. Error verwenden wir, falls Exceptions auftreten, so war es uns möglich schnell Fehler zu finden und zu beheben.

UML:
Unser UML zeigt alle Klassen, Methoden und deren Attribute, sowie sämtliche Beziehungen untereinander und Vererbung etc. Erstellt haben wir das Klassendiagramm mit draw.io

Threads:
Der Thread/ Callable hat sein eigenes Package und wird für die Sortierung der Spieler anhand der Anzahl der Punkte verwendet. Der synchronisierte Zugriff auf gesharte Daten findet im package „db" in der Klasse „InMemoryDB" statt. Die Call-Methode sperrt die ganze Klasse MemoryDb da sie sowohl set als auch get verwendet.

Streams und Lambda-Funktionen:

Sowohl (parallel) Streams als auch Lambda werden unter anderem in CountPlayerScore Callable verwendet. Weitere Lambda Funktionen sind in der GUI zu finden.

Dokumentation und Test-Fälle:
Tests wurde zum FileParser und zu den Fragen geschrieben. Bei den Fragen wird der Konstruktor getestet.
Hierbei werden Negativtests verwendet und auf eine Exception geprüft.

Factories:
Factories werden für die GUI verwendet. Hierbei wird eine Scene zurückgegeben, welche entweder ShowScoreView enthält oder MultiPlayerShowScore.