Programmieren II
JavaQuiz Projekt

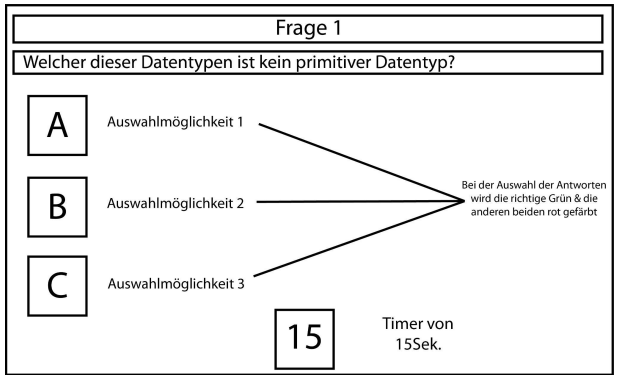
Präsentiert von:
Hakan Koyun, Karuan Osi, Mohamed El Agheb

Agenda

- Einleitung
- Projektplan
- Klassendiagramme
- Benutzeroberfläche
- Klassen und Interfaces
  - QuizModelAndView / IQuizModelAndView
  - ClientRMI
  - Question
  - Storage
- Fazit

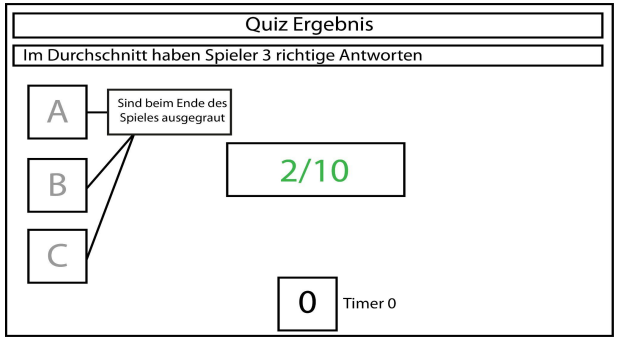
Einleitung

- Quiz Spiel bei dem 10 Fragen rund um die Programmiersprache Java beantwortet werden
- Pro Frage hat man 15 Sekunden Zeit diese zu beantworten
- Wird eine frage nicht innerhalb der 15 Sekunden beantwortet, dann wird Sie als Falsch bewertet und es geht zur nächsten Frage



Ende des Spiels

- Anzahl richtig beantworteter Fragen
- Durchschnitt der richtig beantworteten Fragen im Allgemeinen



Projektplan

Projektplan (Beginn)

Anforderung	Anfang	Ende	Teammitglied
Spielentscheidung durch das gesamte Team	24.04.2021	30.04.2021	A, B
Umgang mit GitHub und Git lernen	24.04.2021	30.04.2021	A, B
Sich Fragen und Antworten für das Quiz überleg	24.04.2021	30.04.2021	B
Klasse QuizView erstellen und daran arbeiten	30.04.2021	07.05.2021	A
Einpflegen der Fragen in die Klasse Question	30.04.2021	07.05.2021	A, B
Auseinandersetzen mit dem MVC Muster	07.05.2021	14.05.2021	A, B
Weiterarbeiten an QuizView	07.05.2021	14.05.2021	A
QuizController erstellen und daran arbeiten	14.05.2021	21.05.2021	C
QuizModel erstellen und daran arbeiten	14.05.2021	21.05.2021	B
QuizStorage erstellen und daran arbeiten	21.05.2021	28.05.2021	A,B,C
Auseinandersetzung mit der verteilten Programmierung	28.05.2021	04.06.2021	A,C
Verteilte Programmierung implemenrtieren	04.06.2021	11.06.2021	A,B,C
Puffer	11.06.2021	18.06.2021	
Optimierung des Designs	18.06.2021	25.06.2021	A,C
Optimierung des Codes	18.06.2021	25.06.2021	A,B,C
Dokumentation	30.04.2021	25.06.2021	A,B,C

- ➔ Lemin: A
- ➔ Karuan: B
- ➔ Hakan: C

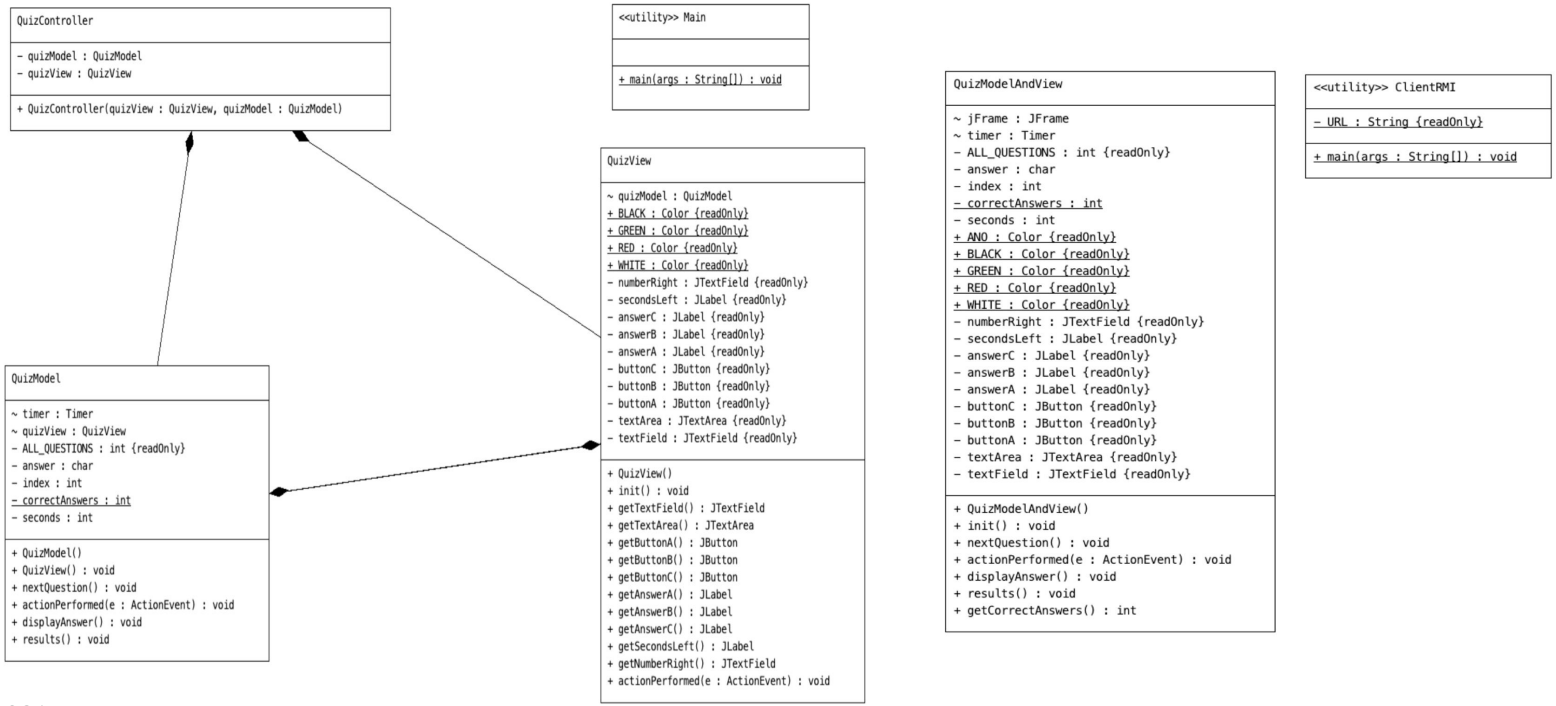
Projektplan (Final)

Anforderung	Anfang	Ende	Teammitglied
Spielentscheidung durch das gesamte Team	24.04.2021	30.04.2021	A, B
Umgang mit GitHub und Git lernen	24.04.2021	30.04.2021	A, B
Fragen und Antworten für das Quiz überlegt	24.04.2021	30.04.2021	B
Klasse QuizView erstellen und daran arbeiten	30.04.2021	07.05.2021	A
Einpflegen der Fragen in die Klasse Question	30.04.2021	07.05.2021	A, B
Auseinandersetzung mit dem MVC Muster	07.05.2021	14.05.2021	A, B
Weiterarbeiten an QuizView	07.05.2021	14.05.2021	A
QuizController erstellt und daran gearbeitet	14.05.2021	21.05.2021	C
QuizModel erstellt und daran gearbeitet	14.05.2021	21.05.2021	B
Suche nach Lösung (MVC funktioniert nicht)	21.05.2021	28.05.2021	A,B,C
Implementierung des Spieles ohne MVC	28.05.2021	04.06.2021	A,C
Erstellen und Arbeiten an den Packages Quiz für die neue Version	04.06.2021	11.06.2021	A,B,C
Auseinandersetzung mit der verteilten Programmierung	11.06.2021	18.06.2021	C
Funktionalität zum Umgang mit Dateien IO	18.06.2021	25.06.2021	A,C
Fertigstellung der verteilten Programmierung	18.06.2021	25.06.2021	A,B,C
Dokumentation	30.04.2021	25.06.2021	A,B,C

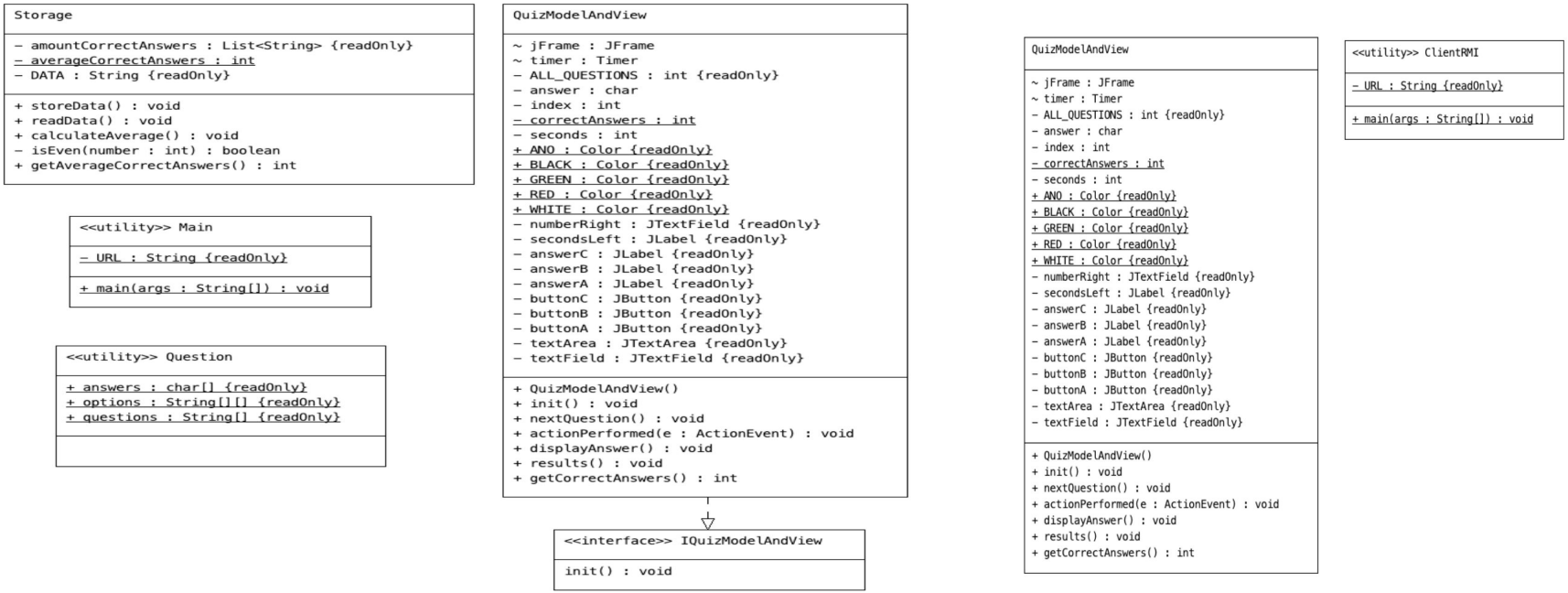
- ➔ Lemin: A
- ➔ Karuan: B
- ➔ Hakan: C

Klassendiagramme

Klassendiagramm (MVC)

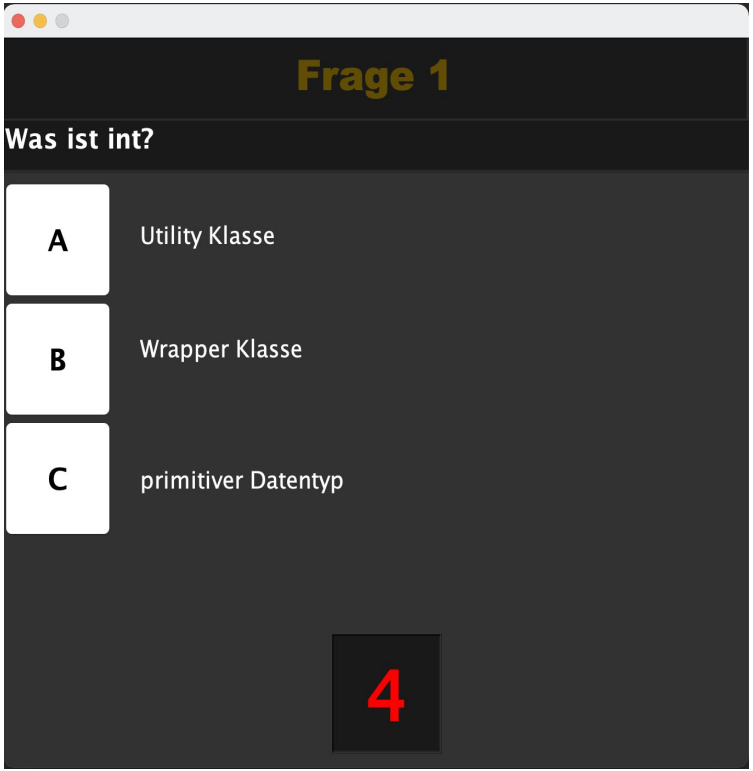


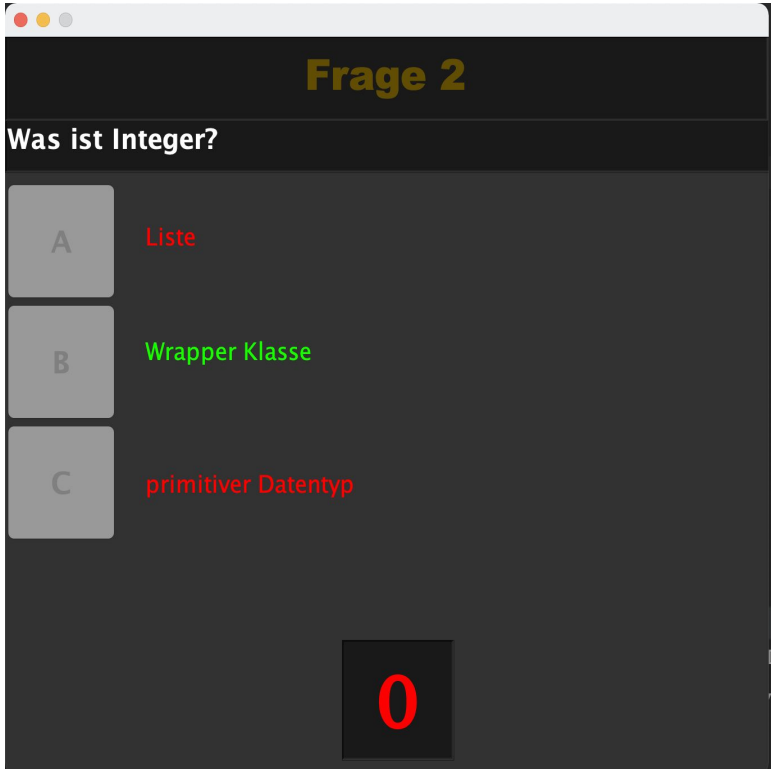
Klassendiagramm



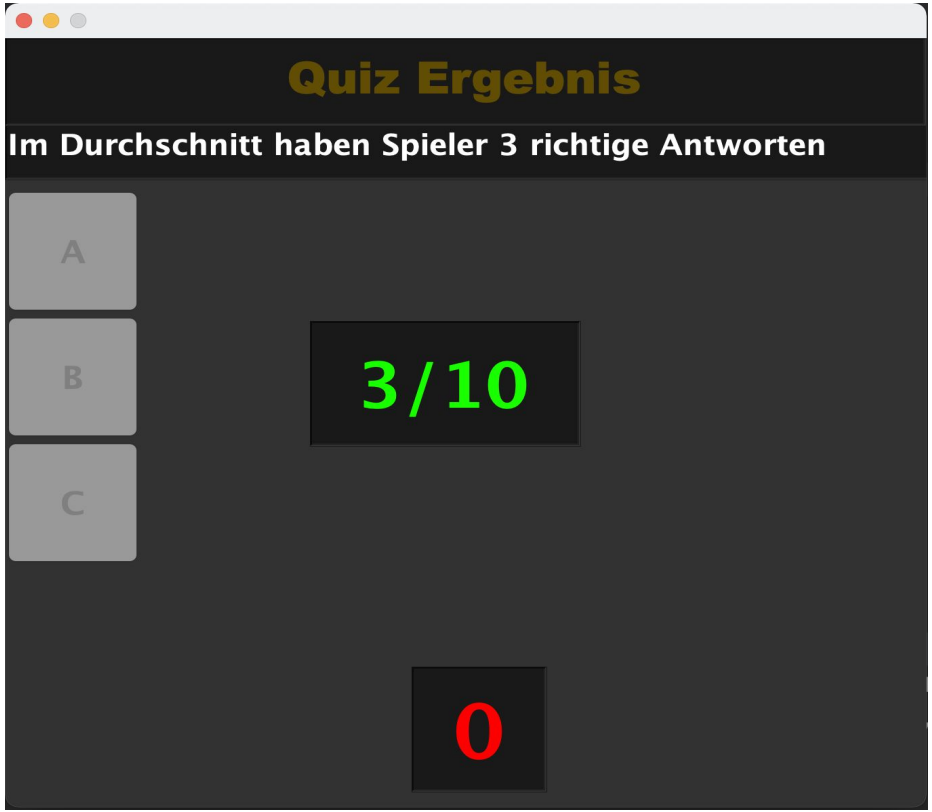
Benutzeroberfläche

Benutzeroberfläche





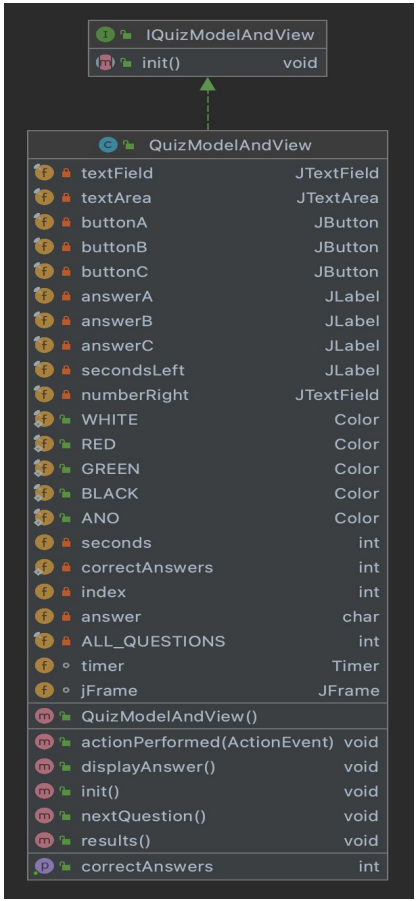
Benutzeroberfläche (Spielende)



Klassen und Interfaces

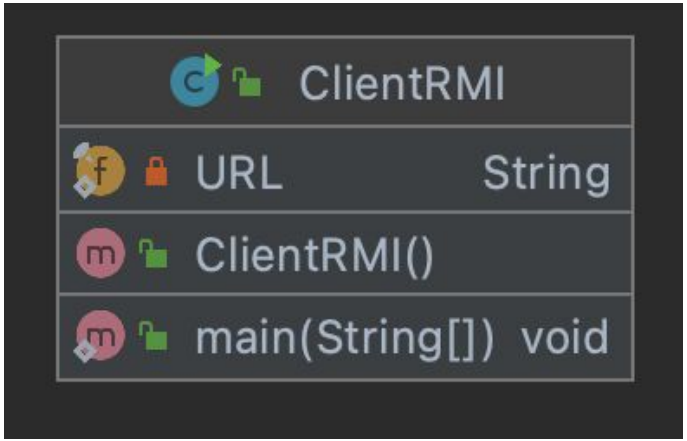
QuizModelAndView und IQuizModelAndView

- ➔ Erbt von JFrame und Implementiert die Interfaces ActionListener, Serializable, IQuizModelAndView
- ➔ Methode init(), mit der die View erzeugt wird
- ➔ Methode *nextQuestion()*, welche die View mit der nächsten Frage füllt
- ➔ Methode *displayAnswer()*, mit der die falschen Antworten rot markiert werden und die Richtigen grün.
- ➔ Methode *results()*, die das Ergebnis des Spiels anzeigt



ClientRMI

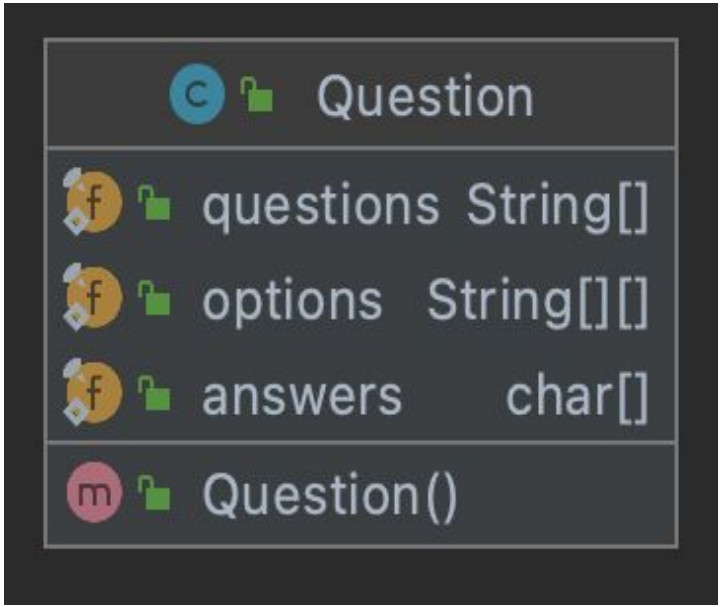
- ➔ Erzeugt ein Objekt aus dem Interface *IQuizModelAndView*
- ➔ Mit der erzeugten Instanz wird die Methode *init()* aufgerufen.



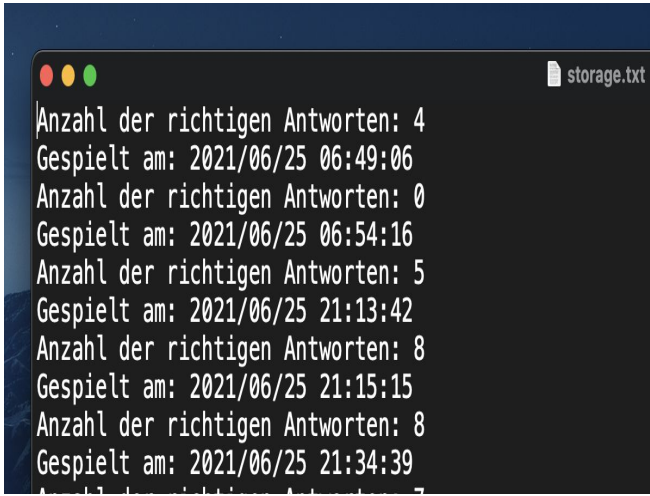

```java
public class ClientRMI {

    private static final String URL = "rmi://localhost:1099/quiz";

    public static void main(String[] args) {
        try {
            IQuizModelAndView stub;
            stub = (IQuizModelAndView) Naming.lookup(URL);
            stub.init();
        } catch (NotBoundException | IOException remoteException ){
            remoteException.printStackTrace();
        }
    }
```

Question

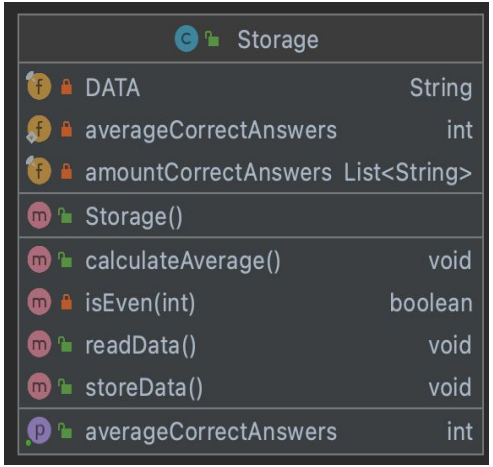
- ➔ Enthält das eindimensionale Array *questions*, in welchem die Fragen untergebracht sind.
- ➔ Enthält das zweidimensionale Array *options*, in welchem die Antwortmöglichkeiten untergebracht sind.
- ➔ Enthält das Char-Array *answers*, in dem die richtige Antwort auf die jeweilige Frage untergebracht ist.



Storage



- ➔ Enthält Methode *storeData()*, mit der die Anzahl der richtigen Antworten und das Datum in eine Text-Datei gespeichert wird.
- ➔ Enthält die Methode readData(), mit der die Anzahl der richtigen Antworten gelesen und in eine ArrayList gespeichert wird.
- ➔ Der Durchschnitt der im ArrayList gespeicherten Daten wird mit der Methode *calculateAverage()* berechnet.



Fazit

Was lief gut?

- Angenehme Gruppendynamik
- Mehrere Meetings
- Hilfestellung von Teammitgliedern bei Problemen
- Schnelle Aufgabenverteilung für die nächsten Wochen.

Was lief schlecht?

- Probleme bei der Umsetzung des Model View Controllers
- Viele kleine Bugs, die auf den ersten Blicken nicht zu erkennen waren.
- Zeitplan musste neu strukturiert werden.
- Schwierigkeit bei der Implementierung verteilter Programmierung

Vielen Dank für eure Aufmerksamkeit

Noch Fragen ?