Dokumentation zur Projektarbeit:

JAVA Client / Server Applikation

–

Multiplayer-Quiz

FTOOP / FS21

Autor: Abdelrahman Abdelwahed

Dozent: Christian Heitzmann

Inhalt

1. Einführung

Ziel dieser Dokumentation ist die Veranschaulichung der Konzepte von Java Multithreading und Java Networking (Socket Programmierung) die für die Implementierung dieser Client / Server Applikation angewendet wurden. Dabei sollen mehrere Spieler in einem Quiz gegeneinander antreten. Das Spiel soll eine reine Kommandozeilen-Applikation ohne grafische Benutzeroberfläche sein. Die Ein.- und Ausgaben erfolgen über den IDE.

1.1 Quiz-Spiel Anforderungen:

- Es darf den gesamten Unicode-Zeichensatz verwendet werden.
- Sowohl auf den Clients als auch auf dem Server erscheinen saubere Ausgaben auf der Kommandozeile.
- Die Kommunikation zwischen den Clients und dem Server erfolgt direkt über Sockets.
- Der Server liest die Textdatei mit den Quizfragen ein. Eine Quizfrage besteht aus einer Frage mit jeweils vier Antwortmöglichkeiten A bis C (wie bei «Wer wird Millionär?»).
- Die Spieler/Clients bekommen vom Server jeweils gleichzeitig eine Frage mit den 3 Antwortmöglichkeiten, A B C, zugesandt.
- Der Spieler/Client mit der schnellsten richtigen Antwort bekommt einen Punkt.
- Jeder Spieler/Client wird über den aktuellen Spielstand informiert.
- Es darf kein Java Remote Method Invocation (RMI) und keine Drittbibliotheken verwendet werden, mit Ausnahme von JUnit.
- Sicherheitsaspekte und die Behandlung von Verbindungsunterbrüchen können Sie ausser Acht lassen.

2. Design & Konzept

2.1 Game Server als Multithreaded-Server

Die Kommunikation zwischen Client und Server erfolgt über Sockets. Da die Applikation für Multiplayer entwickelt werden muss, wurde hier das Konzept von Multithreading (parallele Programmierung) eingesetzt. Dabei werden Tasks zur selben Zeit verarbeitet. Bezüglich der «Performance», ist es von den Anzahl Prozessorkerne abhängig. Die Anzahl Kerne in einem Prozessor entspricht wie viele Threads (Prozesse) gleichzeitig abgearbeitet werden können. Jeder Prozess besteht aus eine oder mehreren Threads. Jeder Thread kennt 5 Phasen: new -> Runnable -> Running -> Waiting -> Dead.

Als Ansatz wurde von einem «Multi-Threaded Server» ausgegangen, weil der Server mehrere Client Verbindungen (Multiplayer) gleichzeitig bearbeiten muss. Dafür wird der ExecutorService eingesetzt, der sich selbstständig für die parallele Bearbeitung der Tasks (Task Queue), für jeder Thread im Thread Pool, kümmert.

Es gibt verschiedene Typen von Executors. Hier wurde den «newFixedThreadPool(n)» eingesetzt wobei hier die Anzahl «Worker Threads» die man benötigt, angegeben werden muss (n). Sobald alle Tasks vom ExecutorService erledigt sind, wird durch pool.shutDown() den Thread Pool beendet.

In den folgenden Diagrammen werden die Beziehungen von Executor Framework und seine Klassen graphisch erläutert.

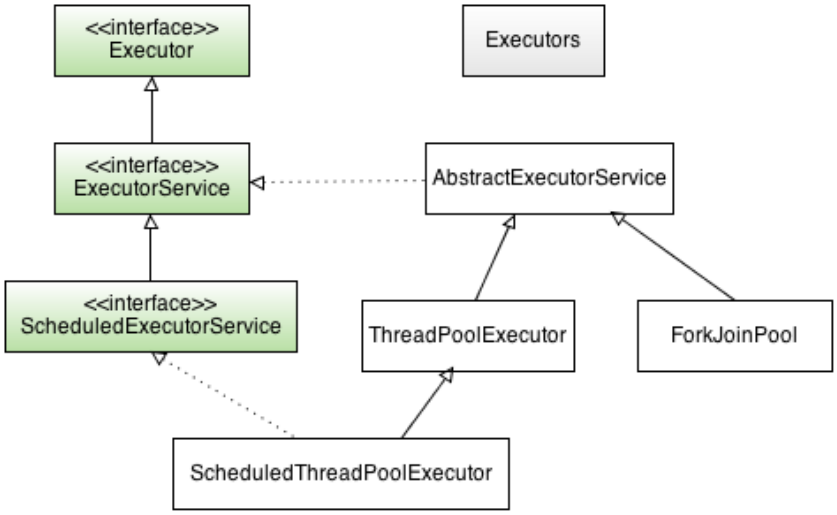


Bild 1: Executor Framework
Quelle : http://geekrai.blogspot.com/2013/07/executor-framework-in-java.html

Der Executor Service befindet sich in der Package java.util.concurrent. Für die Implementierung eines Thread Pools für das Spiel wurde folgenden Syntax verwendet:

```
private static final ExecutorService pool = Executors.newFixedThreadPool(5);
```

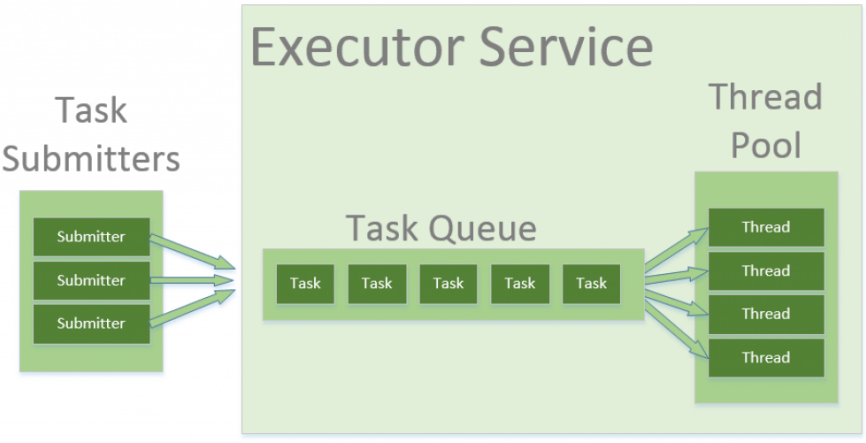
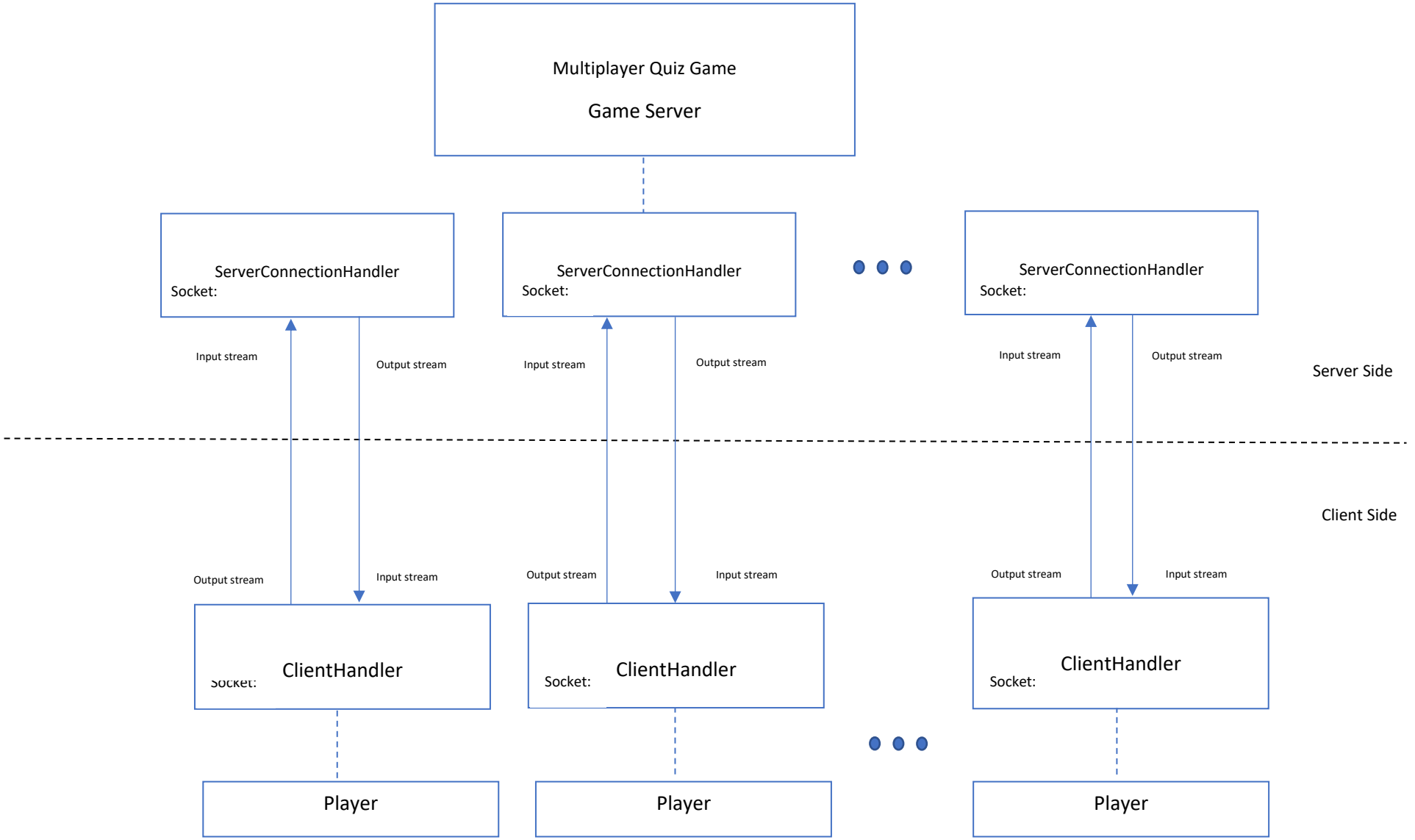
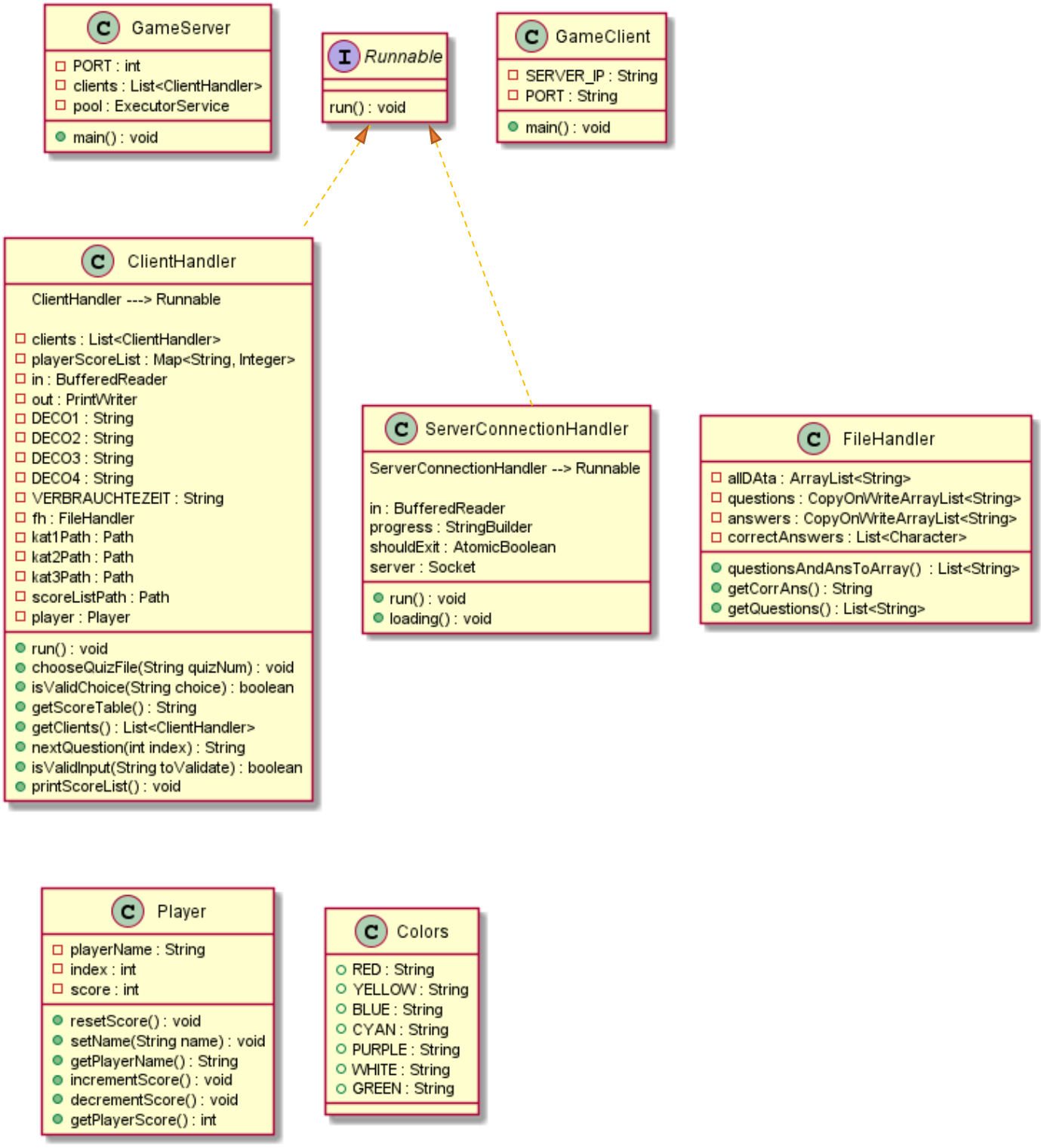
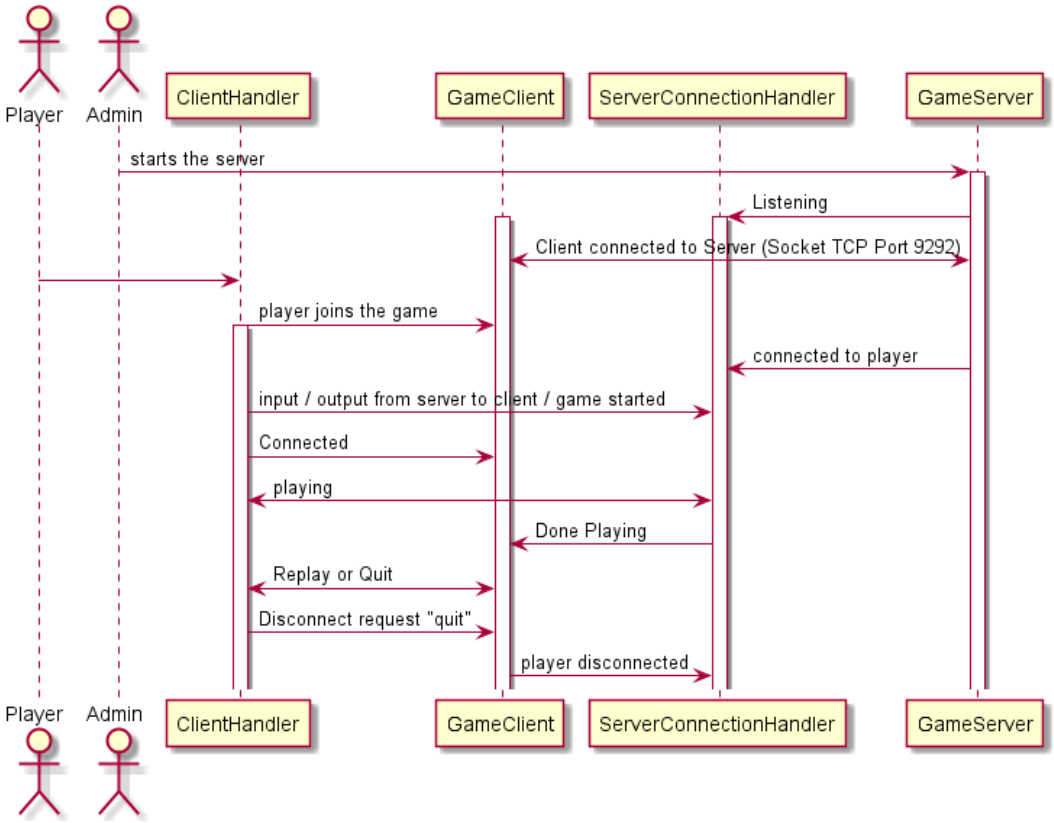


Bild 2 : Executor Service
Quelle : https://www.baeldung.com/wp-content/uploads/2016/08/2016-08-10_10-16-52-1024x572.png


Multiplayer Quiz Game
Game Server
ServerConnectionHandler
Socket:
ServerConnectionHandler
Socket:
ServerConnectionHandler
Socket:
Input stream
Output stream
Input stream
Output stream
Input stream
Output stream
Server Side
Client Side
Output stream
Input stream
Output stream
Input stream
Output stream
Input stream
ClientHandler
Socket:
ClientHandler
Socket:
ClientHandler
Socket:
Player
Player
Player

3. UML / Sequence Diagram





4. Funktionalität

Sobald ein Client (Player) sich erfolgreich mit dem Server verbunden hat, fängt das Spiel an. Der Spieler wird zuerst aufgefordert ein Quiz auszuwählen. Nach Eingabe der Quiz Nummer, wird der Spieler aufgefordert seinen Namen einzugeben. Nach diesem Schritt fängt das Quiz Spiel an und die Erste Frage wird an den Spieler gesendet. Der Spieler hat 30 Sekunden Zeit, um jede Frage zu beantworten. Nach Eingabe der Antwort, wird das Resultat und der Spielstand für alle Spieler angezeigt. Jedes Quiz besteht aus 13 Fragen. Das Format der Fragen darf nicht geändert werden, und sieht folgendermassen aus:

```
# Frage 1

Welchem Umstand verdankt das erstmals im Mittelalter gebraute Bockbier seinen Namen?

A Wer es trank, verspürte danach Lust auf fettige Speisen.
B In Regensburg gab man der Maische Stroh aus dem Ziegenstall bei.
C* Aus der ehemaligen Hansestadt Einbeck wurde in Bayern «Oan Bock!».
```

Die Klasse «FileHandler» ist dafür verantwortlich, dass die Zeilen aus dem File eingelesen werden und die Fragen und Antworten jeweils in einem Array gespeichert werden. Wenn eine Zeile das Wort «Frage» enthält, soll die Zeile danach in den «questions» Array gespeichert werden. Wenn die Zeile einen «*» enthält, soll dies mit einem " " ersetzt werden und der dazugehörenden char[0] in den «answers» Array gespeichert werden. Nachdem die letzte Frage beantwortet wurde, ist das Spiel fertig, und der Spieler hat die Möglichkeit, das Quiz erneut zu spielen oder mit «quit» zu beenden.

5. Klassen

Die Applikation besteht aus 7 Klassen. Jede Klasse wird nun kurz beschrieben und die wesentlichen Funktionalitäten erklärt.

5.1 GameClient

Diese Klasse repräsentiert den Client/Spieler. Es verfügt über einen Socket, um eine Verbindung zum Server auf demselben TCP-PORT 9292 herzustellen. Das ServerConnectionHandler-Objekt, serverConn, stellt über denselben Socket eine Verbindung zum Client her. Die Daten werden mit BufferedReader «in» und Printwriter «out» übermittelt. Sobald alle Objekte initialisiert sind, übergibt der Client die Serververbindung als Thread und wird dieser Thread wird gestartet. Sobald "quit" eingegeben wird, werden die Streams geschlossen und somit findet keine Verbindung mehr zwischen diesem Client und dem Server.

5.2 GameServer

Das ist der Multithreaded Server, der über den ServerSocket listener mit dem Client kommuniziert. Der Server «lauscht» ständig auf Clientverbindungen und die Kommunikation zwischen Client und Server erfolgt über den gleichen TCP PORT 9292. Sobald der Client erfolgreich verbunden ist, werden die Daten/Informationen an den Client, über den input und output Streams gesendet. Durch Starten der Main Methode in dieser Klasse, führt der Executor Service den Thread-Pool aus, indem er jede Client-Verbindung in einen Thread absetzt. Jeder verbundene Client wird in die Client ArrayList hinzugefügt. Jeder Client-Thread wird als ClientHandler-Objekt, mit einem Socket und einer Liste von Clients als Argumente, instanziiert. Wenn die Verbindung erfolgreich ist, wird die Meldung "Player 1 Connected" auf der Konsole angezeigt. Wenn der Thread-Pool des Executor Service die Ausführung aller Tasks abgeschlossen hat, wird der Thread-Pool heruntergefahren.

5.3 ClientHandler

Das ist der Client, der die Fragen vom Server an alle Clients übermittelt. Diese Klasse beschreibt die Logik des Spiels. Über diese Klasse werden die Fragen, abhängig davon welchen Katalog der Benutzer zu Beginn des Spiels ausgewählt hat, angezeigt. Der Spieler kann zu Beginn des Spiels (Ausführung der run() Methode) eines aus 3 verschiedenen Quiz auswählen. Es wird dann nun jene Frage mit dem vorhin erwähnten Format angezeigt und die Antwort des Spielers (Konsoleneingabe) wird mit der jeweiligen Antwort im Array «correctAnswers» der Klasse FileHandler verglichen. Weiter unter Abschnitt 6, Spiel Demo, wird es anhand von Bildern noch demonstriert.
Sobald die Runde fertig ist, hat der Spieler die Wahl entweder das Spiel zu beenden oder ein neues Quiz zu spielen. Die Klasse beinhaltet insgesamt 8 Methoden. Die Methode printScoreList () erstellt eine Textdatei mit den Spielerangaben, Name und Score, aller Spieler. Die Liste heisst ScoreList.txt.

5.4 ServerConnectionHandler

Diese Klasse verarbeitet die multiplen Verbindungen zum Server. Es druckt alle Serverantworten aus dem Server an alle Clients auf der Konsole gleichzeitig aus. Die Fragen werden vom ClientHandler zum Server übermittelt und die Klasse ServerConnectionHandler leitet die Responses an die Clients weiter. Das geschieht Clientseitig über «InputStream» und «OutputStream» der Klasse GameClient. Die Methode loading() druckt den «Progress Bar» aus.

5.5 FileHandler

Hier werden die Quiz Files bearbeitet und die Arrays für die Fragen und Antworten erstellt. Diese Klasse enthält 3 Methoden:

- getCorrAns (): holt die Richtige Antwort aus dem Array, um sie mit dem vom Spieler eingegebener Antwort zu vergleichen.
- getQuestions (): Gibt alle Fragen aus dem Array Fragen zurück
- getQuestionsAndAnsToArray (): diese Methode erstellt die Frage mit den 3 Antwortmöglichkeiten gemäss den Anforderungen.

5.6 Player

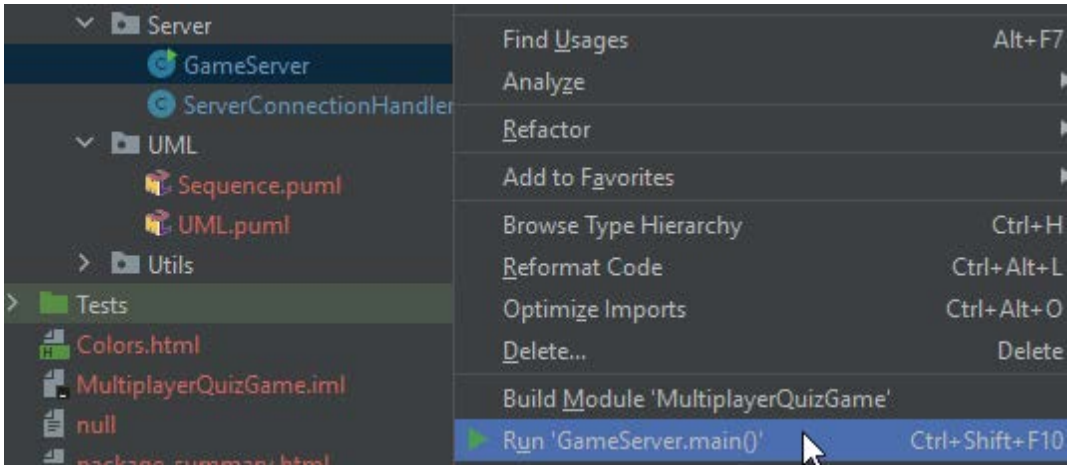
Die Player Klasse enthält die Attribute Name, Score und Index für den Spieler. Die Klasse besitzt 5 Methoden: resetScore(), getPlayerName(), setName(), getPlayerScore(), sowie incrementScore().

5.7 Colors

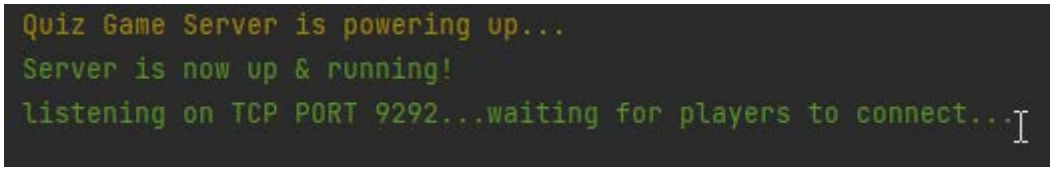
Diese Klasse enthält die Farbkodierungen für die Konsolen-Schriftfarbe als String Variablen.

6 Spiel Demo

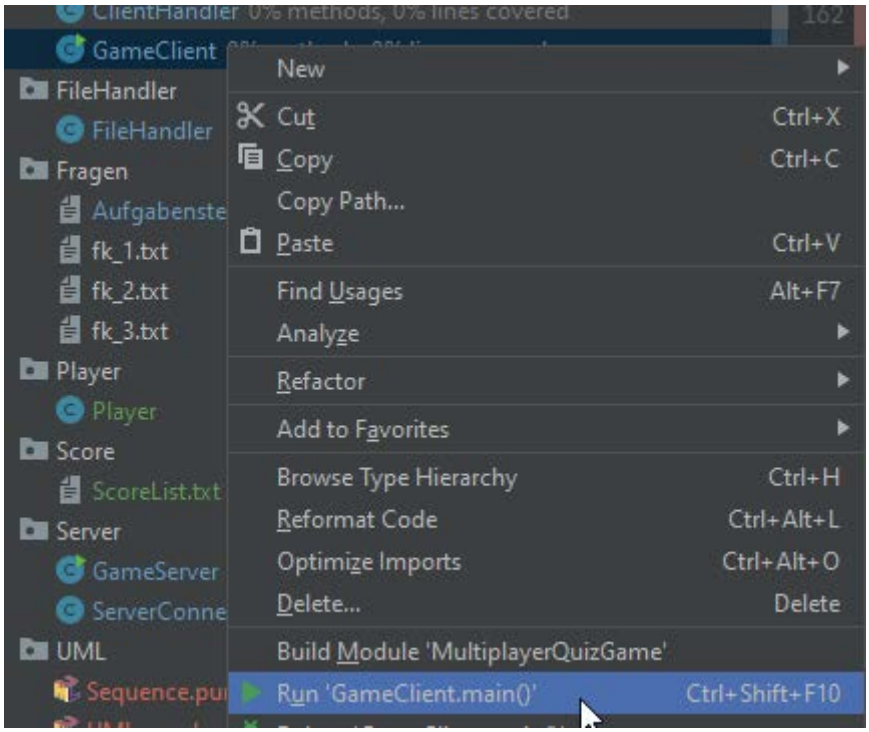
1. Server starten: GameServer.java starten



2. Folgendes wird auf der Konsole angezeigt, mit je einem kleinen Zeitabstand von 1.2 Sekunden zwischen den Ausgaben.



3. Client starten: GameClient.java starten



4. Die Progress Bar wird auf dem Client angezeigt und auf dem rechten Bild, was Serverseitig angezeigt wird:

```
Quiz Fragen werden geladen...

 65% ################################ /
```

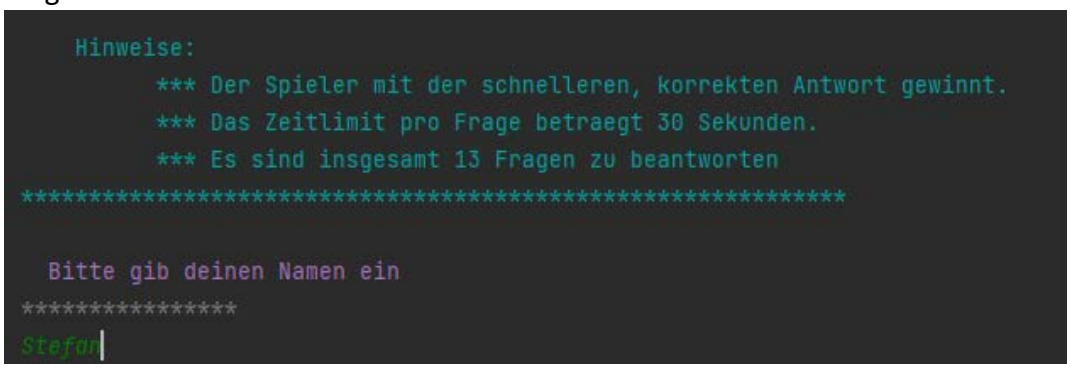
```
Quiz Game Server is powering up...
running!
listening...waiting for players to connect...
[SERVER] Player 1 127.0.0.1 connected
[SERVER] Player 2 127.0.0.1 connected
[SERVER] Player 3 127.0.0.1 connected
```

5. Willkommensmaske auf dem Client. Das Spiel hat nun begonnen.

6. Quiz auswählen

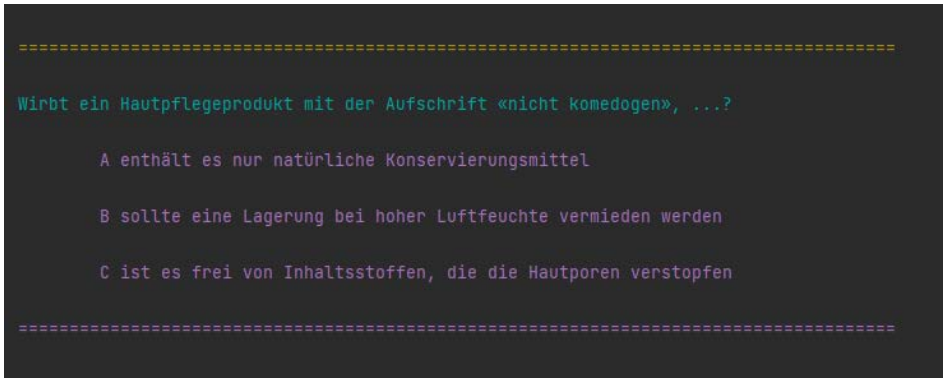
```
        WILLKOMMEN ZUM SPIEL
        WER WEISS DENN SOWASS?
************************************************************

Bitte waehle eines der 3 Quiz aus. Auswahlmoeglichkeiten sind: 1, 2 oder 3
    ---------------------------------------------
      1 - Quiz vom 2. Januar 2020
      2 - Quiz vom 3. Januar 2020
      3 - Quiz vom 7. Januar 2020
```

7. Die Spielhinweise werden angezeigt, danach folgt die Spielernamen-Eingabe:



8. Der Spieler wird begrüsst und der Count Down zum Quiz wird gestartet:

9. Die erste Frage wird dann angezeigt:



10. Wenn die Antwort richtig ist:

```
Antwort ist richtig!

Verbrauchte Zeit: 1 sek
-.-.-.-.-.-.-.-.-.-
Deine Punktenanzahl: 2
-.-.-.-.-.-.-.-.-.-

Spielstand
------------------------------------------------------------
Stefan : 1 / 13     Sandra : 2 / 13     Thomas : 1 / 13
------------------------------------------------------------
```

11. Wenn die Antwort falsch ist:

```
Antwort ist falsch!
die richtige Antwort ist B
****************

Verbrauchte Zeit: 1 sek

-.-.-.-.-.-.-.-.-.-
Deine Punktenanzahl: 1
-.-.-.-.-.-.-.-.-.-

Spielstand
------------------------------------------------------------
Stefan : 1 / 13     Sandra : 1 / 13     Thomas : 1 / 13
------------------------------------------------------------
```

12. Wenn der Spieler zu spät antwortet:

```
Sorry, du warst zu spaet!

die Antwort ist A
Verbrauchte Zeit: 52 sek

-.-.-.-.-.-.-.-.-.-
Deine Punktenanzahl: 1
-.-.-.-.-.-.-.-.-.-

Spielstand
-----------------------------------------------------------
Stefan : 5 / 13     Sandra : 1 / 13     Thomas : 3 / 13
-----------------------------------------------------------
```

13. Ende des Spiels:

```
Spielstand
-----------------------------------------------------------
Stefan : 6 / 13     Sandra : 2 / 13     Thomas : 7 / 13
-----------------------------------------------------------


>>>>>>>>>>>>>>>>>>>>>>>SPIELENDE<<<<<<<<<<<<<<<<<<<<<<<


Danke fuers Spielen!

Gib 'Quit' ein um das Spiel zu beenden.   Oder drucke 1, 2 oder 3 um fortzufahren!
```

14. Wenn «quit» eingegeben wird:

- Seitens Clients:

```
quit
Stream closed
user quit the game

Process finished with exit code 0
```

- Seitens Server wenn ein/alle Spieler während dem Spiel «quit» eingeben:

```
Quiz Game Server is powering up...
Server is now up & running!
listening on TCP PORT 9292...waiting for players to connect...
[SERVER] Player 1 127.0.0.1 connected
[SERVER] Player 2 127.0.0.1 connected
[SERVER] Player 3 127.0.0.1 connected
Spieler Stefan hat das Spiel verlassen
Spieler Sandra hat das Spiel verlassen
Spieler Thomas hat das Spiel verlassen
```

15. Das Verhalten bei falschen Eingaben von Quiz-Nummer und Antwortmöglichkeiten: (Exception Handling)

```
        WILLKOMMEN ZUM SPIEL
        WER WEISS DENN SOWASS?
************************************************************

    Bitte waehle eines der 3 Quiz aus. Auswahlmoeglichkeiten sind: 1, 2 oder 3
    ------------------------------------------------
      1 - Quiz vom 2. Januar 2020
      2 - Quiz vom 3. Januar 2020
      3 - Quiz vom 7. Januar 2020
5
ungueltige Eingabe. Per default wird Quiz 1 gewaehlt.
```

```
================================================================================

Forscher haben herausgefunden, dass zehn Minuten Baden im Meer ...?

        A die Hautflora grundlegend verändert

        B die Nasenschleimhäute anschwellen lässt

        C die Kopfhaare schneller grau werden lässt

================================================================================

X

Eingabe nicht gueltig. Gueltige Eingaben sind: A, B oder C
Verbrauchte Zeit: 2 sek
-.-.-.-.-.-.-.-.-.-.-
Deine Punktenanzahl: 0
-.-.-.-.-.-.-.-.-.-.-
```

16. Um den GameServer endgültig auszuschalten (Admin) :