

Dokumentation

Chess



Gruppenname: Not 5D-Chess

25.06.2021

Teammitglieder

An dem im Laufe des 2. Praktikums angefertigten Projekts haben wir zu zweit in einer Gruppe gearbeitet diese Gruppe bestand aus:

Lasse Tristan Feldermann-Welkner
Sven Specht

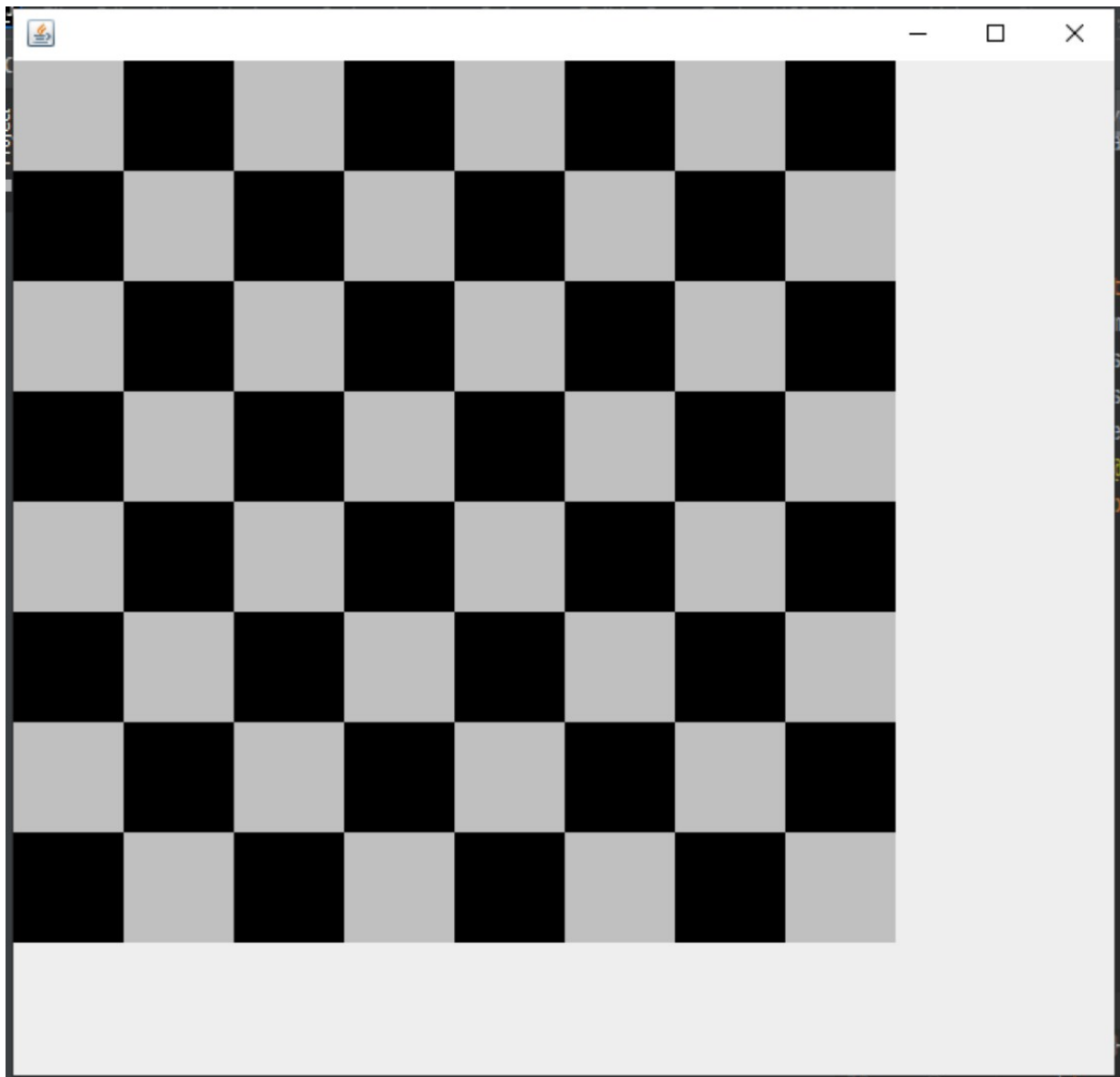
Einleitung

Schach ist ein international bekanntes strategisches Brettspiel, in dem es darum geht den gegnerischen König schachmatt zu setzen. Dies erreicht man indem man den König so angreift, dass er sich nicht mehr verteidigen oder Flüchten kann. Dazu bewegen die beiden Spieler abwechselnd ihre Spielfiguren auf dem 8x8 Spielfeld, dem Schachbrett. Im Spiel gibt es insgesamt 6 Figurentypen für jeden Spieler diese sind: Pawn, Rook, Knight, Bishop, Queen und King. Von den Bauern gibt es jeweils Acht und von den anderen Figuren, bis auf Queen und King, jeweils Zwei. Schach gilt international als Sportart in der offizielle, als auch private Turniere veranstaltet werden.

Wochendokumentation

Woche 1 (15.04-22.04)

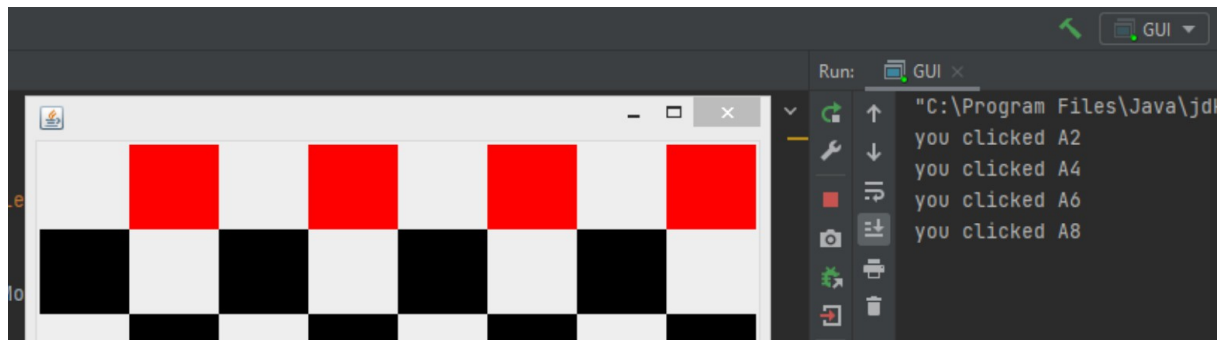
In der ersten Projektwoche, haben wir unsere Accounts auf Github erstellt und uns nach kurzer Besprechung darauf geeinigt, als Program für unser Praktikumsprojekt, ein Schachspiel zu programmieren. Darauf hin haben wir ein erstes Konzept des Spiels erstellt und eine erste Version des Gameboards erarbeitet. Im Laufe der Woche haben wir das Gameboard weiter überarbeitet und z.B. den einzelnen Feldern Namen gegeben.



Erste Version des Gameboards

Woche 2 (23.04-30.04)

In der zweiten Woche des Projekts, haben wir die ursprüngliche Idee des Gameboards abgeändert und durch eine Version mit einem JPanel Array in einem GridLayout ersetzt. Zu dem haben wir zum Test ein MouseEvent eingebaut, mit dem wir im Späteren Verlauf des Projekts, auf die Figuren zugreifen wollten um diese bewegen zu können. Das MouseEvent konnte zunächst nur die Farbe der angeklickten Felder ändern sowie den Namen des Feldes ausgeben.



MouseEvent: Färben der geklickten Felder und Ausgabe des Feld Namens

Woche 3 (01.05-7.05)

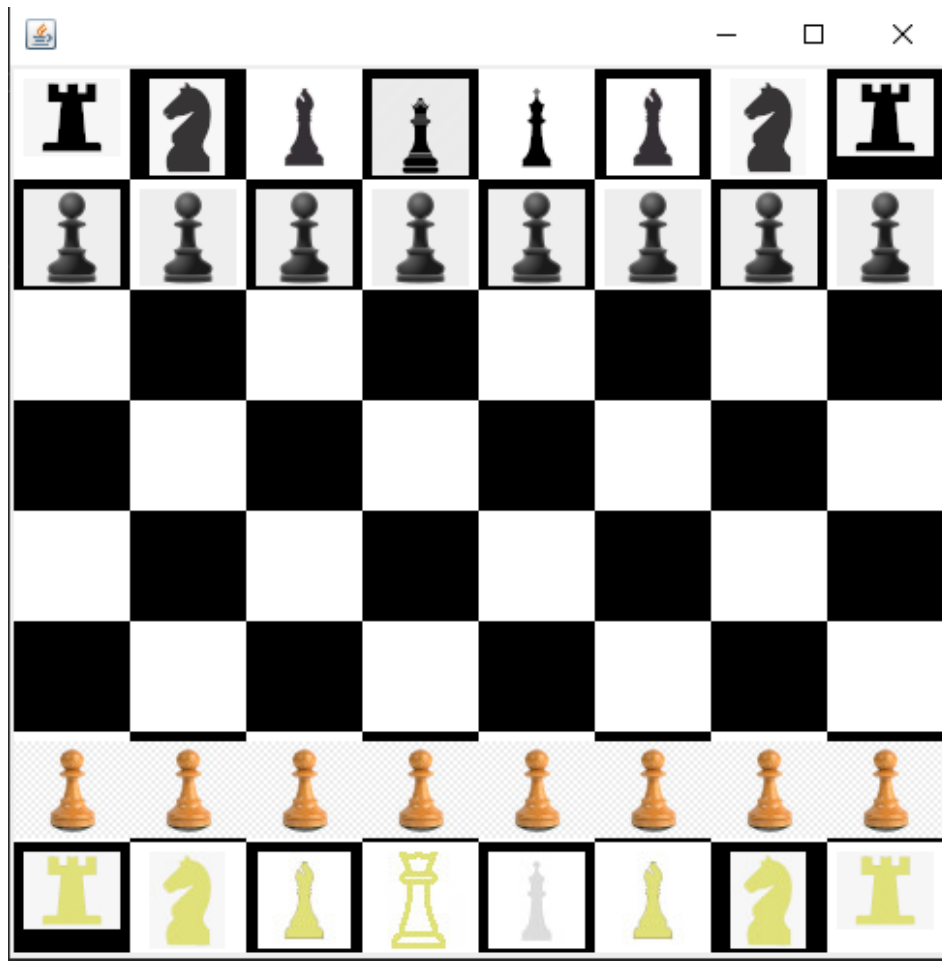
In der dritten Woche gab es keine Änderungen am Projekt.

Woche 4 (08.05-14.05)

Hier haben wir zunächst das "Piece" Interface angelegt und auf Basis dieser dann die "Pawn", sowie weitere Figure- Klassen implementiert.

Woche 5 (15.05-21.05)

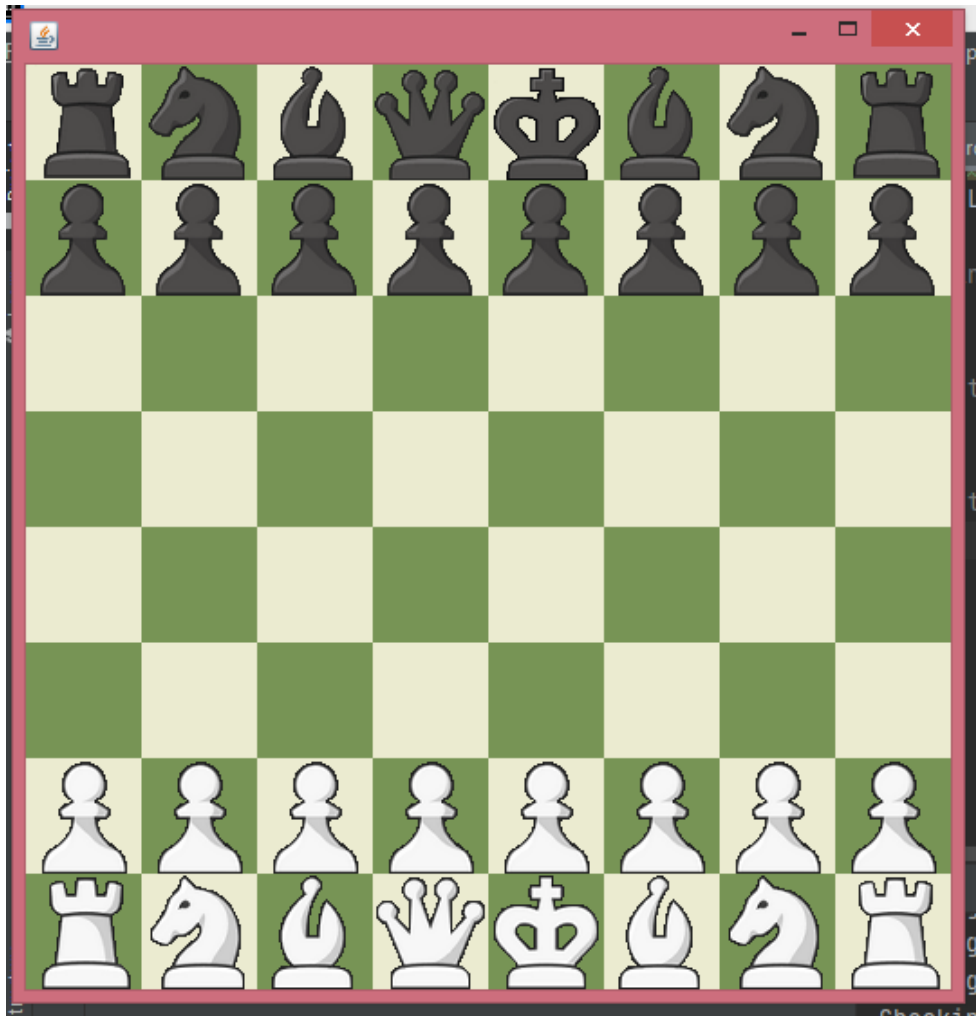
In dieser Woche hatten wir ein Problem mit dem Repository und mussten daher alles neu hochladen. Des Weiteren wurden in dieser Woche zunächst erst nur die “Pawns” und später die restlichen Figuren eingefügt, hierzu bedienten wir uns des “ImageIO Filereaders” um die jeweiligen Bilder der Figuren auf Labels zu setzen, welche dann auf das ihnen vorher zugewiesene Feld gesetzt werden. Außerdem machten wir einen ersten Ansatz an das Bewegen der Figuren. Dazu erstellten wir die “movePiece()”-Methode, welche hier aber noch weit von der finalen Version entfernt war.



Gameboard mit den Spielfiguren

Woche 6 (22.05-28.05)

Die sechste Woche verbrachten wir zum einen damit, die "movePiece()" -Methode fertig zu stellen und somit allen Figuren auf dem Feld das Bewegen zu ermöglichen, zu diesem Zeitpunkt allerdings noch ohne Regeln d.h. Alles war möglich. Außerdem haben wir das Design der Figuren und des Gameboards angepasst um ein besseres Nutzererlebnis zu erzeugen.



Neues Design des Gameboards und der Figuren

Woche 7 (29.05-04.06)

Um mit unserer grob angelegten movePiece()-Methode sinnvoll weiterarbeiten zu können, mussten wir nun eine Methode implementieren die diese mit den nötigen Daten füttert. Dies ist die getPos()-Methode geworden, welche lediglich als "Hilfsfunktion" für die movePiece()-Methode agiert. Sie stellt dieser bei 2 aufeinanderfolgenden Klicks auf das Gameboard alle nötigen Informationen zur Verfügung, zu diesem Zeitpunkt allerdings nur welches Feld zuerst geklickt wurde und wohin die sich dort befindliche Spielfigur, sollte sie existieren, bewegen soll.

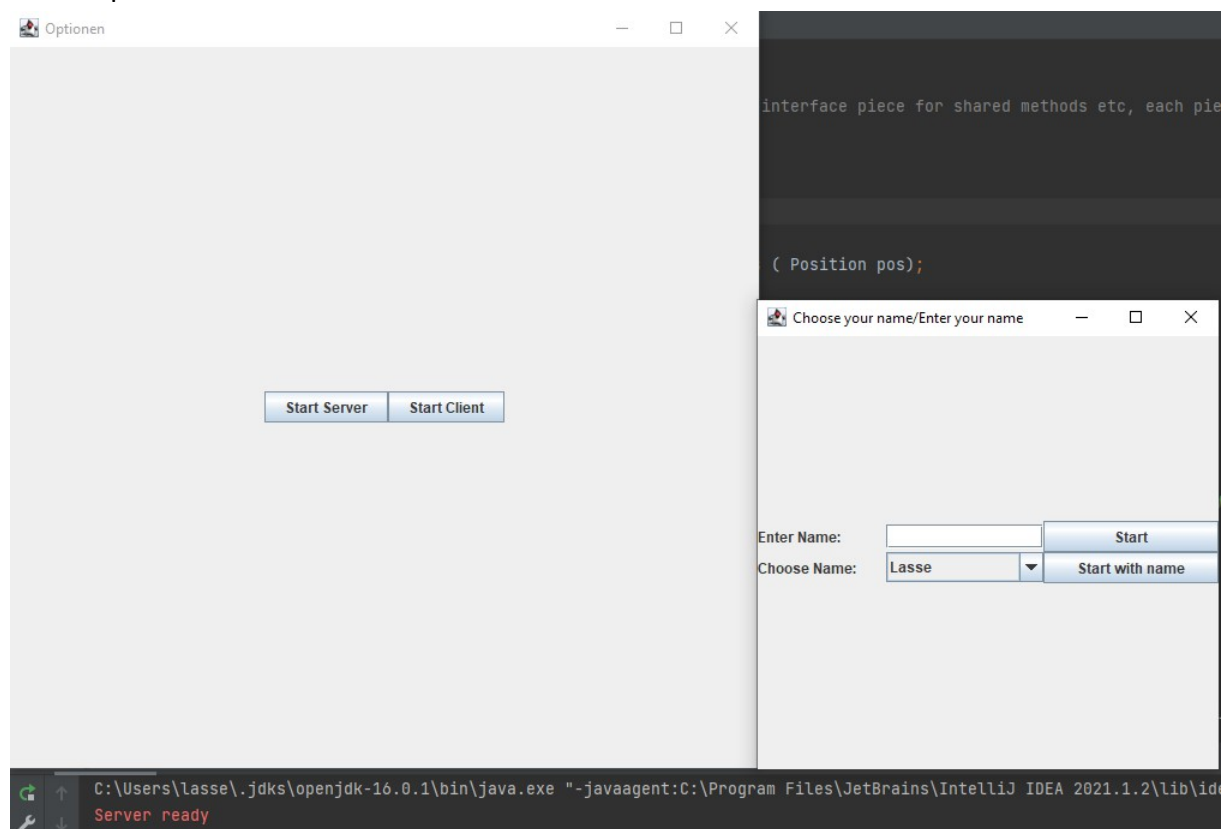
Außerdem wurden hier einige kleine Überarbeitungen am Code vorgenommen um einen "saubereren" Stil zu erreichen.

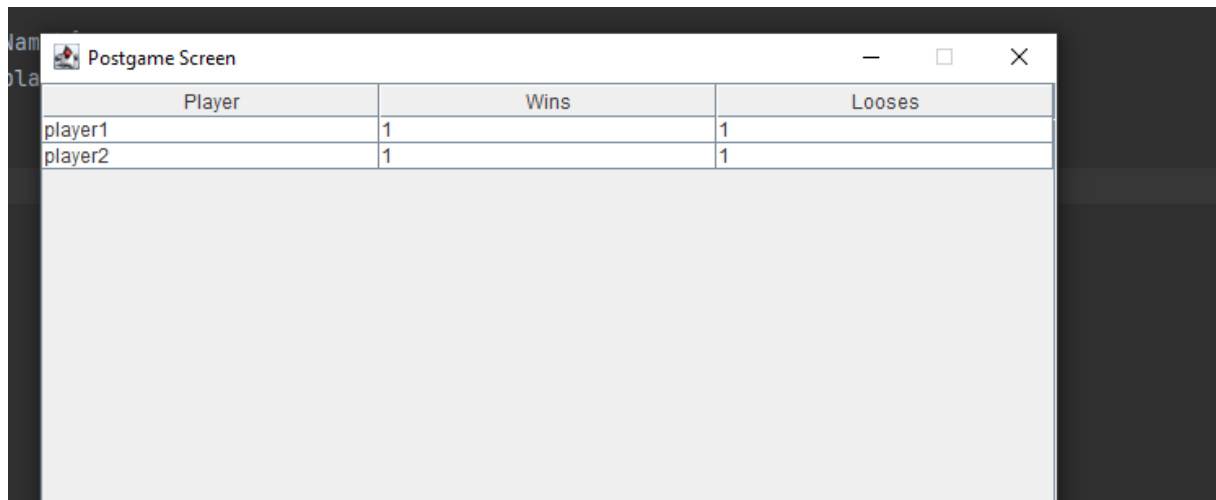
Woche 8 (05.06-11.06)

Nachdem unsere movePiece()-Methode nun die ersten relevanten Daten zur Verfügung gestellt bekommt war es an der Zeit diese weiter auszubauen, sodass es dem Spieler möglich ist Figuren zu schlagen. Zunächst war dies noch unabhängig von den Farben möglich.

Woche 9 (12.06-18.06)

In dieser Woche wurden einige RMI Funktionen, die EntryUI Class sowie die PostgameUI Class implementiert.





Player	Wins	Looses
player1	1	1
player2	1	1

Leider sind weder die Synchronisation der unterschiedlichen Clienten noch der Postgame Screen vollständig implementiert.

Außerdem wurde die Logic Class implementiert, welche den Postgame Screen via der `whiteWins()`- oder `blackWins()`-Methode aufruft.

Hier entstand auch die Idee Serialisierung zu implementieren um so Player Objekte persistent zu Speichern und diese zu nutzen um den Postgame Screen mit den Win/Looses aller Spieler zu füllen. Diese Idee wurde allerdings aufgrund von Schwierigkeiten bei der Implementierung verworfen. Stattdessen setzten wir uns das Ziel ein Menü zur Auswahl bereits eingegebener Spielernamen zu implementieren.

Woche 10 (19.06-25.06)

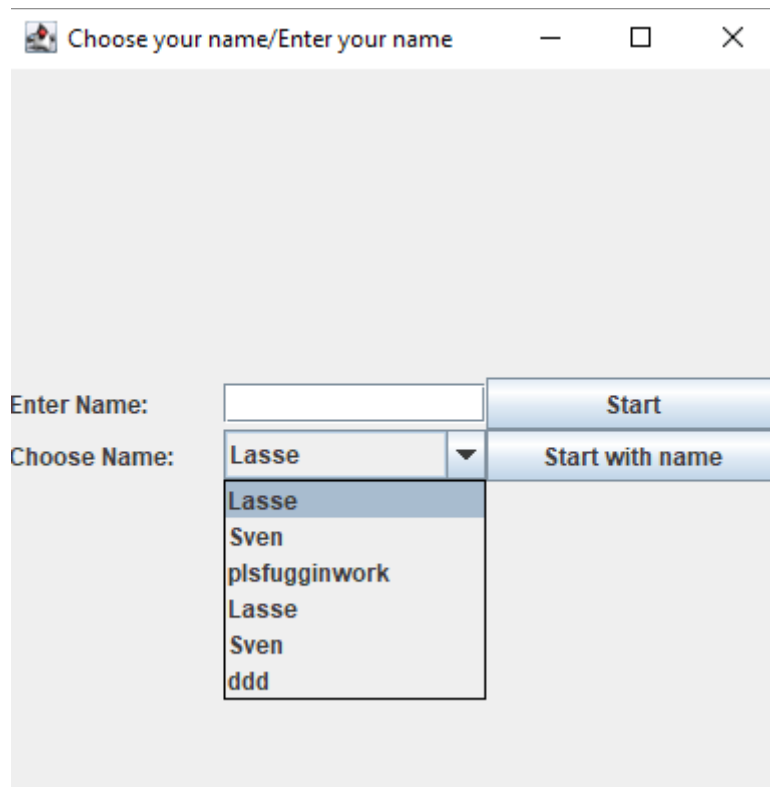
Neben dem geplanten Menü zur Auswahl bereits eingegebener Spielernamen wurden in dieser Woche hauptsächlich Feinheiten verbessert.

Die `movePiece()`-Methode wurde erneut überarbeitet um nur noch "legale" moves zuzulassen und nur die Farbe, welche aktuell am Zug ist, darf bewegt werden.

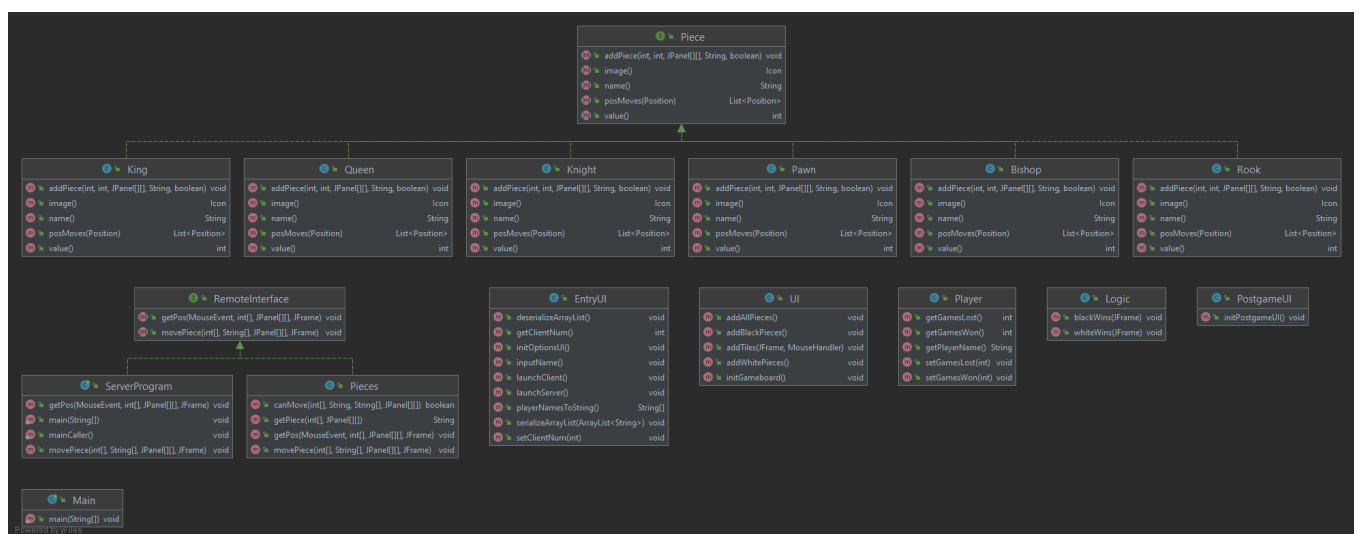
Außerdem wurde unnötiger Code zu großen Teilen entfernt und einige Methoden effizienter gestaltet.

Das Menü zur Auswahl bereits eingegebener Namen wurde durch eine `JcomboBox` realisiert, welcher als Parameter eine `playerNamesToArray()`-Methode übergeben wird. Diese Wandelt eine `ArrayList` der Länge `n` in ein `Array` um mit welchem die `JcomboBox` initialisiert wird.

Die `ArrayList` wird beim Aufruf des Programms deserialisiert und, sollte ein neuer Name statt eines bereits existierenden Namens eingegeben werden, kurz vor dem Start des Clienten neu serialisiert.



Klassendiagramm:



Kopie zur besseren Lesbarkeit ist im GitHub Repository hinterlegt.

Aufgabenverteilung (Programmieren)

Lasse Tristan Feldermann-Welkner entwickeln der Klassen:

- Logic Class
- EntryUI Class
- PostgameUI Class
- ServerProgramm Class
- Diverse Korrekturen/Überarbeitungen

Sven Specht entwickeln der Klassen:

- UI Class
- Pieces Package
- RemoteInterface Interface
- Piece Interface
- Diverse Korrekturen/Überarbeitungen

Aufgabenverteilung (Dokumentation)

Lasse Tristan Feldermann-Welkner dokumentieren der:

- Wochen 6-7

Sven Specht dokumentieren der:

- Wochen 1-6