Architektur

## Erklärung / Begründung der Architektur

Als grobes Designpattern haben wir das Model-View-Controller (MVC) Design Pattern gewählt und grundsätzlich versuchen wir möglichst einen objektorientierten Ansatz umzusetzen.

## Controller

Hier befinden sich Klassen, die den Ablauf des Spiels durch Inputs und Regeln bestimmen

* Class Input: liest den Input des Users und steuert den Ablauf des Spiels durch eine Ein-, und Ausgaberoutine
* Class Check: beinhaltet die Regeln zu Schach- und Schachmatt-Situationen und regelt dadurch den Spielverlauf

Package AI:

* Class BoardValueNode: dient dafür, eine Art Baumstruktur darzustellen bei dem jedes Piece-Objekt als einzelnes Blatt gilt. Es werden hier dann die verschiedenen Werte für die jeweiligen Pieces vergeben und verglichen. Pawn hat zum Beispiel nur einen Wert von +-10, die Queen dagegen jedoch schon einen Wert von +-90. Außerdem befinden sich die verschiedenen Methoden zum MinMax-Algorithmus und zum Alpha-Beta-Pruning-Algorithmus hier.
* Class MinMaxAi: Hier wird der MinMax-Algorithmus aus BoardValueNode aufgerufen wodurch ein neuer Move in Form eines Strings generiert wird.
* Class SimpleAi: Dies ist eine AI die quasi nur einen zufälligen möglichen Move heraussucht und diesen dann ausführt. Die Klasse BoardValueNode wird hierbei in keiner weise berücksichtigt.

### Model

Hier finden sich die Klassen, die nach einem objektorientierten Vorgehen die Objekte des Schachs und ihre Funktionen repräsentieren

* Class Board: Repräsentiert das Schachbrett an sich und beinhaltet Methoden, die den Spielverlauf auf dem Brett darstellen und steuern
* Class Square: Stellt die einzelnen Felder auf dem Schachbrett dar, diese können verschiedene Werte aufweisen, wie zum Beispiel ob eine Spielfigur auf dem Feld steht
* Interface Piece: Bildet ein Interface, welches die einzelnen unterschiedlichen Figuren zusammenfasst und gemeinsame Funktionen und Verhaltensweisen bündelt
* Class King, Queen, Bishop, Knight, Rook, Pawn: Sind die konkreten einzelnen unterschiedlichen Figuren

## View

Hier liegen die verschiedenen Klassen für die grafische Benutzeroberfläche. Es gibt verschiedene Klassen um die jeweiligen Elemente auf dem GameScreen (FXML) richtig steuern zu können.

* Class GuiMain:
* Class StartMenu:
* Class GameScreen:
* Interface ChessBoardController:
* Class ChessboardHuman:
* Class ChessboardComputer:
* Class LastMoveController:
* Class ClockController:
* Class CemeteryController:

## Resources

Hier liegen die FXML auf die mit JavaFX zugegriffen werden kann und auf dem Bildschirm als eigenes Fenster geöffnet werden kann.

* gameScreen:
* startMenu: