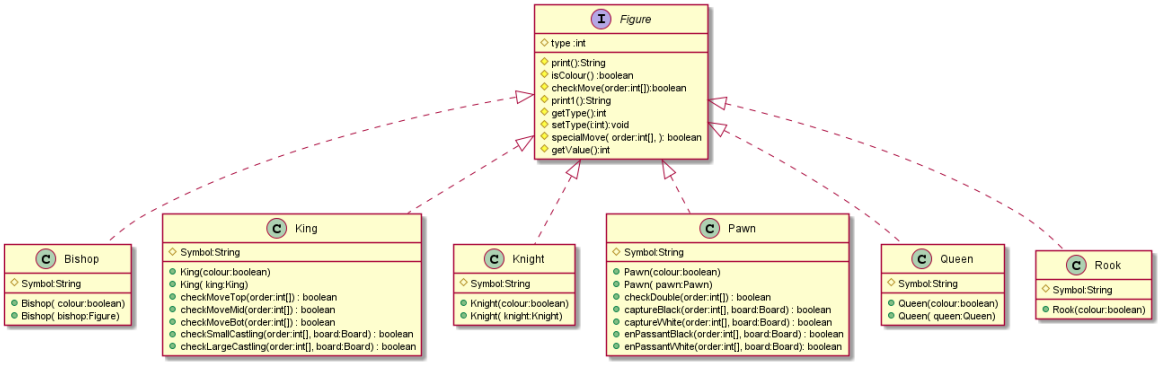
Architekturdokument

Einleitung

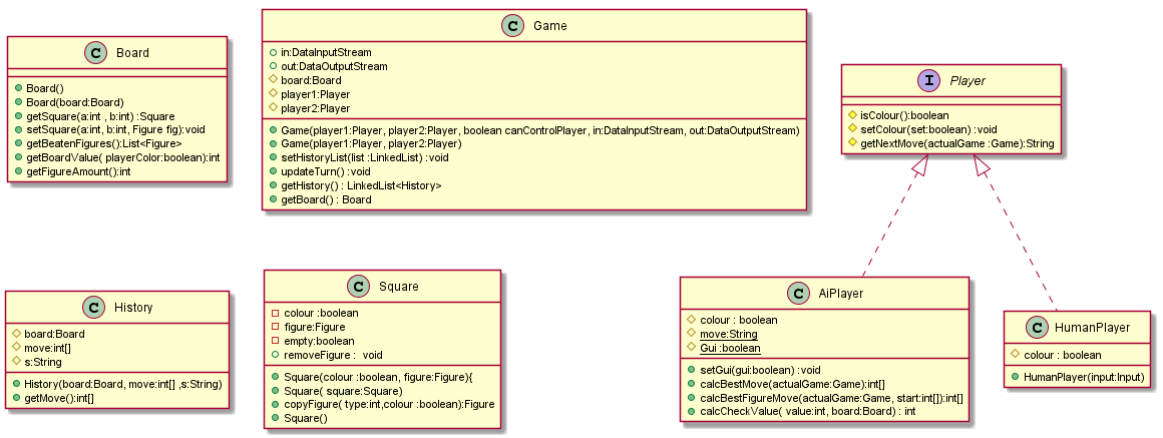
Das Programm ist nach der Model-View-Controller Architektur aufgebaut. Dies ist sinnvoll da wir verschiedene Views nutzen (Konsole/Gui) was bei dieser Architektur gut zu realisieren ist.
Das Modell ist die Grundlegende Datenstruktur, der Controleller Verarbeitet die Eingaben des Benutzers und die View stellt die Datenstruktur buzw veränderungen an ihr dar.

Das Model

Das Package Modell umfasst die Spielfiguren, das Spielbrett, die Felder der Spielbretts, die History sowie die Spieler HumanPlayer und AiPlayer.
Es gibt ein Interface für die Figuren welches grundlegende Methoden verlangt. Jede Spielfigur muss eine Farbe, ein Symbol, einen Figurspezifischen Type und einen Wert haben. Außerdem haben wir uns dazu entschieden das grundlegende Bewegungsmuster der Figuren in den Figuren zu implementieren.

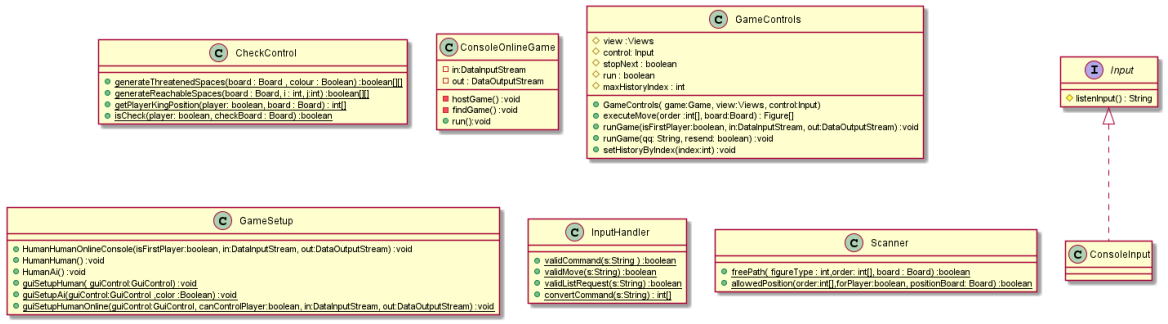


Die Klasse Game stellt grundlegende Informationen über ein erzeugtes Spiel bereit. Sie besitzt ein Board, zwei Spieler und speichert welcher Spieler gerade am Zug ist. Außerdem ist eine Liste mit History Objekten auf ihr gespeichert.
Jedes History Objekt beinhaltet ein Board, der letzte Zug der zu diesem Board geführt hat als String sowie als int Array und die Farbe des Spielers der in diesem Spielzustand dran war.
Das Board beinhaltet ein 2d Array mit Square Elementen, sowie eine Liste mit bereits geschlagenen Figuren. Ein Square Objekt beinhaltet entweder eine Figur oder ist leer. Eine besondererheit ist der Copy Constructor unseres Boardes mit welchem wird deep copys erstellen können. Dies war insbesondere dafür hilfreich einen Zug auf einer Kopie eines Boardes auszuführen und danach zu prüfen ob sich eine Schachsituation ergeben hat.
Außerdem können wir so jeden Spielstand in der History List Zwischenspeichern.
Die verschiedenen Spieler besitzen mindestens eine Farbe und eine Methode die den nächsten Move zurückgibt.



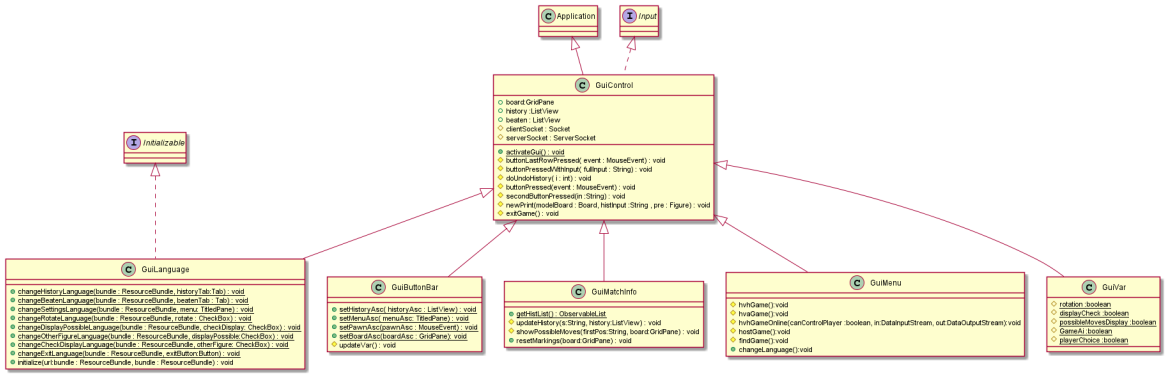
Die Control

In der Klase GameControls wird der Spielfluss Kontrolliert. Hier werden die Eingaben über die Konsole verarbeitet sowie Bewegungen überprüft und ausgeführt. Für bewegungen ist die Klasse Scanner notwendig. Mit der Methode Freepath kann überprüft werden ob auf einem gegebenen Board der weg für einen gegebenen Figurtypen auf einem angegebenen Weg möglich ist. Die Methode AllowedPosition überprüft ob die Start und Endposition eiiner -Bewegung erlaubt sind.
CheckControl besitzt die Methode GenerateThreatenedSpaces welche für einen Spieler und ein Board angibt welche Felder dieser Spieler unmittelbar bedroht. Die Methode GenerateReachableSpaces gibt an welche Felder ein Spieler auf einem Board erreichen kann. Außerdem lässt sich die Position des Königs für einen Spieler mit getPlayKingPosition ermitteln. Die Methode isCheck gibt an ob ein Spieler sich auf einem Board im Schach befindet.
Die Klasse GameSetup richtet die verschiedenen Arten von Spielen ein. Die Klasse ConsoleOnlineGame besitzt Kontrollmechanismen für das OnlineSpiel über die Konsole.
Die Klasse InputHandler kontrolliert die Eingaben über die Konsole. Hier wird eine Eingabe auf Syntax überprüft. Außerdem besitzt sie Methoden um von der Eingabe in die interne Repräsentation von Zügen (2d Arrays) zu wechseln.
Die Klasse Input liefert eine Methode um eine Eingabe über die Konsole zu tätigen.

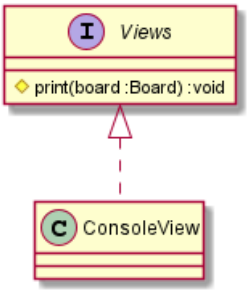


Die Kontroll für die Gui

Die Control für die Gui liefert zum einen die Anzeige der Verschiedenen Gebiete in der Gui, zum anderen sorgt sie dafür dass die Eingaben über die Gui weiter an die bereits genannten Controller Klassen geleitet werden.



Die View



Die View ist für die Darstellung des Modells in einer für den Nutzer verständlichen Form zuständig. Im Falle der Konsole übernimmt dies die Klasse ConsoleView welche mit ihrer Print Methode das Schachbrett in textbasierter Form darstellt. Nach jedem Spielzug bei dem das Model verändert wurde gibt diese das aktualisierte Model aus. Bei der Gui agiert die Fxml Datei als View. Diese hat den "Bauplan" für die 2D Nutzeroberfläche welche in der Control Datei durch den load() Befehl geladen wird. Auf dieser Benutzeroberfläche wird dann die aktuelle Stellung der Figuren auf dem Schachbrett angezeigt. Mit diesen Informationen kann der Spieler dann seine nächste Entscheidung treffen.

Verarbeitung eines Spielzuges

Zuerst wird ein eingegebener Zug auf die Syntax überprüft. Ist diese Korrekt wird der Zug in die interne Form gebracht (von String zu int array) und weiter verarbeitet:
Der eingegebene Spielzug wird erst mit der Methode AllowedPosition, dann mit der Methode der betreffenden Figur und zuletzt mit der FreePath Methode auf dem aktuellen Board überprüft.
Eine Besonderheit sind hier die Methode SpecialMove der Figuren. Hier weichen wir leicht von der strikten Trennung zwischen MVC ab und lassen zu dass der aufruf von SpecialMove das Modell Board manipuliert. Wir haben uns dazu entschieden um leichter besondere Moves wie rochade oder en passant implementieren zu können. Da es Im Schach nicht viele solcher Sonderzüge gibt ist es unserer Ansicht nach in Ordnung. Wenn jedoch viele Figuren mit Sonderzügen hinzugefügt werden Sollen würde sich eine eigene Klasse dafür anbieten.
Des weiteren wird mit der isCheck Methode überprüft ob man sich selbst Schach gesetzt hat.

Sind alle Überprüfungen erfolgreich passiert wird der Zug ausgeführt.

Darstellung Programmablauf

Hier eine leicht vereinfachte Darstellung des Programmablaufs in der Konsolenschnittstelle.