SOFTWARE ARCHITEKTUR
für
MELT Chess
Version 3.0
12. Juli 2021

Inhaltsverzeichnis

1. Einführung

Die Applikation MELT-Chess unterteilt sich in die Module

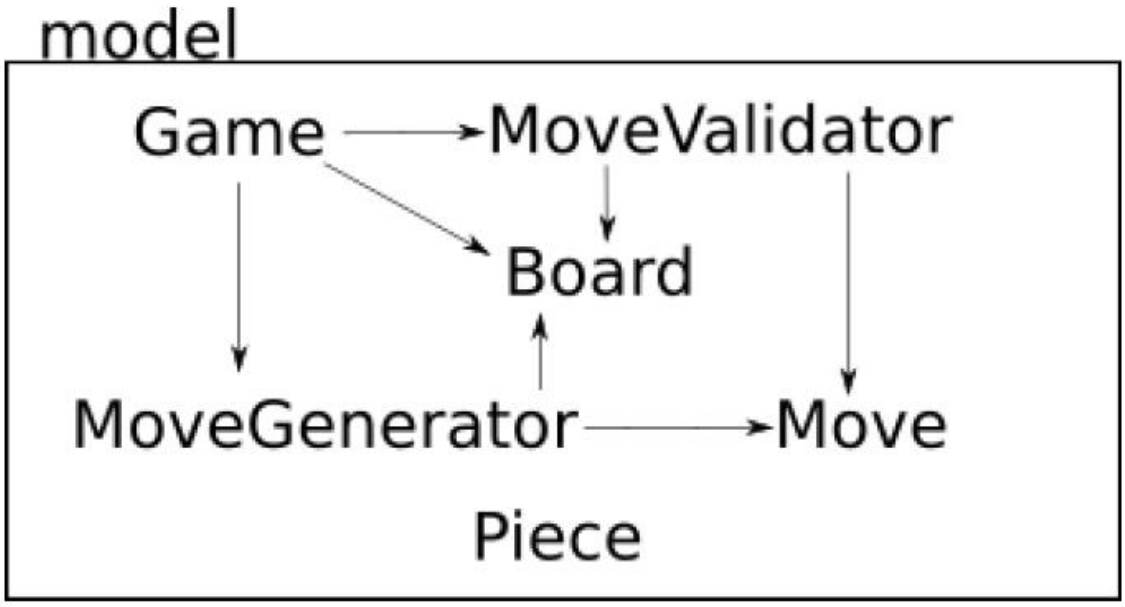
- model: für die Implementierung der Datenstrukturen und Schachregeln
- cli: für die Implementierung der rudimentären Konsolenschnittstelle mit eingeschränktem Umfang der Features[1].
- gui: für die Implementierung der grafischen 2d-Schnittstelle.
- engine: für die Implementierung der simplen Schach Engine

Im Folgenden wird nun der Aufbau dieser vier Module erläutert.

[1]Siehe Anforderungen.pdf Dokument

2. Model

Das model Paket teilt sich im Wesentlichen in zwei Arten von Klassen: Zum einen die Datenstrukturen Piece, Move, Board und die Implementierung der Schach-Regeln[2] in den Klassen MoveGenerator und MoveValidator. Die Klasse Game verpackt obige Klassen noch einmal für den einfachen Zugriff durch die Clients.



2.1 Piece

Eine Figur wird vollständig durch einen Integer Wert beschrieben, der sich aus Typ + Farbe zusammensetzt. Dazu sind in der Klasse Piece folgende konstante Felder definiert:

Typ	Wert
None	0
King	1
Pawn	2
Knight	3
Bishop	4
Rook	5
Queen	6
White	8
Black	16

Weiter bietet die Klasse ein paar statische Helfermethoden, eine komplette Auflistung befindet sich in der Java Documentation.
Das Erweitern um mögliche neue Spielfiguren ist „etwas umständlich", da die Flags mit Hinblick auf eine Implementierung durch byte anstelle von int geplant wurden. Soll mehr als eine neue Figur eingeführt werden, müssen daher die Flags für die Farben sowie die entsprechenden Bitmasken angepasst werden.

[2]gemäß den FIDE-Regeln von 2018

2.2 Move

Die Klasse Move ist eine simple Kapselung für die drei Felder:
Move(int startSquare, int targetSquare, int flag), wobei der flag-Parameter optional ist. Für einen Zug sind folgende mögliche Flags
definiert:

Typ	Wert
None	0
EnPassantCapture	1
Castling	2
PromoteToQueen	3
PromoteToKnight	4
PromoteToRook	5
PromoteToBishop	6
PawnTwoForward	7

2.3 Board

Die Klasse Board liefert eine vollständige Beschreibung einer Position auf dem Schachbrett. Dies ist notwendig damit Instanzen von Board dem Spielverlauf hinzugefügt, und somit die Funktionalität „Rückgängig machen" implementiert werden kann.
Board merkt sich die Spielfiguren in einem int[64] Array, sowie in einer List<Integer> für die geschlagenen Figuren.
Eine wichtige Methode ist Board.makeMove(Move move), welche eine neue Instanz von Board zurückgibt in der move ausgeführt wurde. Die Methode beachtet dabei keine der Schachregeln, setzt allerdings beim Bewegen von Figuren die an der Rochade-Regel teilnehmen (also die vier Türme sowie beide Könige) ein entsprechendes Flag das die Rochade verhindert[3].

2.4 Coordinate

Die statische Klasse Coordinate liefert einfache Wege um die verschiedenen Repräsentationen von Brettpositionen (zB „b7" $ 9 $ {1, 1}) zu konvertieren.

2.5 MoveGenerator

Die MoveGenerator Klasse mit den statischen Auslagerungen implementieren die Regeln zum Bewegen und nur zum Bewegen einer Figur[4]. Von außen betrachtet sind zwei Methoden interessant:
List<Move> generateMoves()
gibt alle möglichen Züge zu gegebener Spielposition zurück.
List<Move> generateMovesStartingAt(int position)
gibt alle möglichen Züge zu gegebener Spielposition zurück, die vom Feld position ausgehen. Für den Fall, dass sich die Anforderungen an die Regeln zum Bewegen von Schach-Figuren ändern, muss lediglich von einer dieser Klasse geerbt und gewünschte Regeln überschrieben werden.

[3]Damit wird jedoch keine Regel umgesetzt, die Flags dienen lediglich zum frühzeitigen Abbrechen bei der Generierung von Zügen
[4]Das bedeutet der MoveGenerator liefert auch regelwidrige Züge!

2.6 MoveGeneratorDirectional

Diese Klasse implementiert die Regeln für die Bewegung der Figuren in jede Richtung. Mit der Einschränkung sich nicht vom Brett zu bewegen.

2.7 MoveGeneratorKing

Diese Klasse implementiert die Regeln für die Bewegung des Königs. Zum einen beinhaltet sie die Methode generateKingMoves, die die allgemeinen Bewegungen des Königs regelt. Zum anderen werden hier über mehrere Methoden die Regeln der Rochade implementiert.

2.8 MoveGeneratorKnight

Diese Klasse implementiert die Regeln für den Spielstein Springer.

2.9 MoveGeneratorPawn

Implementiert Regeln für die Bewegung der Bauernfigur.

2.10 MoveValidator

Der MoveValidator ist eine rein statische Klasse, die zu gegebener Spielposition und einem Zug überprüft, ob der Zug Regelkonform ist. Da die Bewegungsregeln bereits in MoveGenerator implementiert sind, wird zunächst überprüft ob der Zug von ihm gefunden wurde. Danach werden noch die folgenden Regeln überprüft:

- Der König steht nach ausführen des Zugs nicht im Schach
- Falls der Zug eine Rochade ist, überprüfe das keines der betroffenen Felder angegriffen wird.
- Ist der König im Schach und es existiert kein valider Zug nachdem der König nicht mehr im Schach steht, ist das Spiel verloren: SCHACHMATT!
- Falls kein valider Zug möglich ist, gibt es ein remis: PATT![5]

2.11 Game

Die Klasse Game bietet ein einfaches, übersichtliches Interface zum Ausführen eines Zugs und dem erhalten von Informationen der aktuellen Partie für die Clients.

[5]Laut FIDE-Regeln gibt es noch weitere Möglichkeiten ein remis zu generieren, diese sind aber nicht Teil der Anforderungen.

3. Cli

3.1 Cli

Der Vordergrund im Cli ist das Initialisieren der Klasse Game. Damit kann ein Spiel im Client gestartet werden. Von der Methode „runGame" aus werden die Spielschleifen für PVP, PVPC und Netzwerk gestartet:
gameLoopPVP()
gameLoopPVPC()
gameLoopNetwork()
Eine weitere Methode der Cli Klasse ist die PerformAction, die einen String entgegennimmt und dem Spieler ermöglicht das Spiel neu zu starten, die geschlagenen Figuren auszugeben und ins Menü zu gelangen. Durch die vorige Sting Eingabe des Spielers.

Die Methode PerformeMove führt die Bewegung des Spielers und der KI aus.

Allgemein verarbeitet diese Klasse die User-Eingaben und führt die entsprechenden Befehle aus.

3.2 CliMenus

In dieser Klasse wird das Verhalten der Konsolenmenüs verwaltet und gesteuert.
Durch die Methode runMainMenu() wird das Standardmenü aufgerufen.

Durch runGameModeMenu() wird das Spielmodimenü implementiert. Dieses Menü startet durch die Eingaben des Users die Spielmodi Methoden.

Die Methode startNetworkGame() gibt das Netzwerkmenü aus. Diese Methode nimmt die Eingabe der IP-Adresse und des Ports entgegen und startet die entsprechende Methode.

Die Methode runLanguageMenu() ruft das Sprachmenü auf. Von hier aus werden die Textmanager Methoden aufgerufen, die die Sprachen regeln.

Die Methode runLoadMenu() regelt das Menü für das Laden des Spielstandes.
Die Methode runSaveMenu() regelt das Menü für das Speichern des Spielstandes.

3.3 ConsoleColors

Diese Klasse beinhaltet die Funktionen für die Stil- und Farbänderung der Konsolenausgabe.

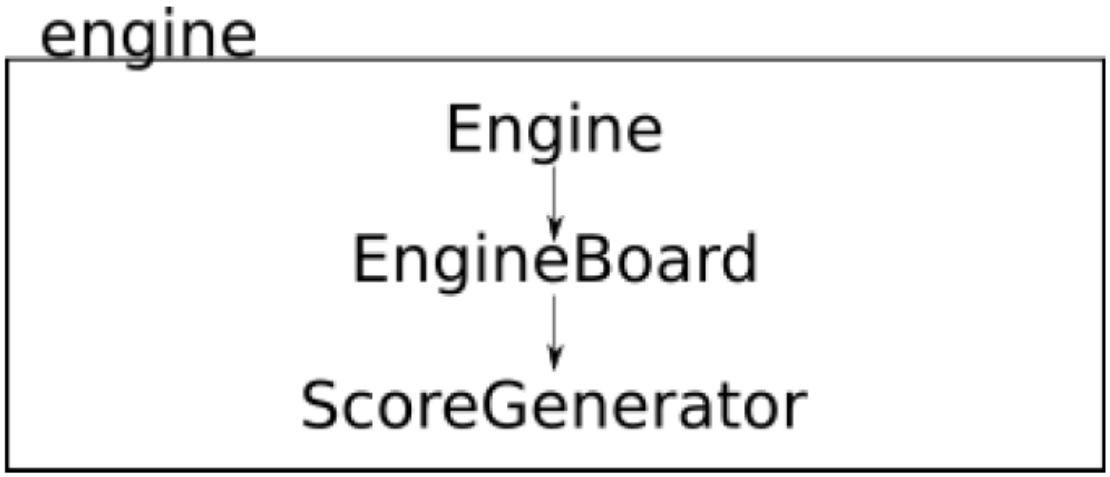
3.4 Help

Diese Klasse beinhaltet die Funktionen zur Steuerung der Anzeige und Ausgabe von Hilfsbefehlen.

3.5 Menu

Die Funktionen dieser Klasse steuert die Auswahl der verfügbaren Optionen. Außerdem wird hier das Menü repräsentiert.

4. Engine



4.1 Engine

Die Engine-Klasse bietet ein einfaches Interface durch das Bereitstellen der Funktion Move generateBestMove(Board board). Außerdem beinhaltet diese Klasse das Alpha-Beta-Pruning.

4.2 EngineBoard

Diese Klasse erweitert die Boardklasse um eine Funktion zur Bewegung einer Position.
Die im Vordergrund liegende Methode ist Comparable. Die Funktion ist die Sortierung einer Liste von Positionen nach der Bewegung.

4.3 ScoreGenerator

Diese Klasse weist einer gegebenen Brettposition einen Punktwert („Score“) zu, basierend auf dem Wert der geschlagenen Figuren und dem Wert der Position der Spielsteine.
Die Position einer Figur, wird durch eine hartcodierte Positionstabelle bewertet für jede Figur und Phase des Spiels. Der Score ergibt sich durch Aufsummieren aller Werte. Ein
positiver Score ist gut für weiß, negativ für schwarz.

5. GIU

Die GUI zeigt, wie ein Klick des Users auf dem Spielfeld verarbeitet wird. Dabei ist zu sehen, dass es nicht für jeden Spielmodus einen eigenen loop gibt. Stattdessen teilen sich die Spielmodi weitgehend in Funktionen.

Die GUI besteht aus fünf Paketen und der GUI Klasse. Die Pakete enthalten die graphischen Benutzeroberflächen der Spielmenüs:

- Hauptmenü: menu
- Spielmenü: game
- Netzwerkmenü: network
- Einstellungsmenü: Settings

In den Packages befinden sich die folgenden Klassen:

Menu Packaging:

- MenuController

Game Packaging:

- BoardController
- GameController
- GameModel
- GamePopup
- PerformeEngineMoveService
- PerformeEngineMoveTask.

Network Packaging:

- NetworkController
- NetworkModel

Settings Packaging:

- SettingsController
- SettingsModel

Util Packeging:

- GraphicsManager
- ResizeManager

Die Controller-Klassen steuern das Verhalten der entsprechenden Menüklassen. Die Modelklassen beinhalten die Verknüpfungen der Funktionen mit den graphischen Oberflächen der jeweiligen Menüs.

6. Util

Hier befinden sich zum einen die Funktionen für das Speichern, in den Klassen Saving und SavingManager. Zum anderen wird hier das Netzwerkspiel geregelt.