**Python - Programmentwurf**

**Team ID:**

IT2

**Team Mitglieder (Name + e-mail):**

Alessa Bayer bayer.alessa-it19@it.dhbw-ravensburg.de

Manuel Emmert emmert.manuel-it19@it.dhbw-ravensburg.de

Tim Bader bader.tim\_alex-it19@it.dhbw-ravensburg.de

Xena Letters letters.xena\_c-it19@it.dhbw-ravensburg.de

**Git verwendet? (fließt nicht in die Note ein):**

Ja

**Buildserver verwendet? (fließt nicht in die Note ein):**

Nein

**Verwendete Python Version:**

3.8.2

**Verwendete Pylint Version:**

2.4.4

**Beschreibung des Dateiformats:**

XXXXXXXXXXXXXXXXXXXXXXXXXXXX

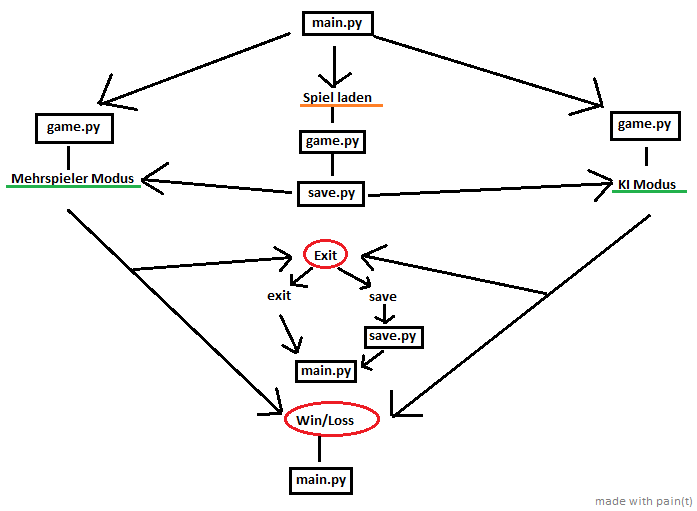
aAHHHHHHHHAHHAHHAHHAHHAHHHH

**Beschreibung des Benutzerinterfaces:**

Die Steuerung erfolgt über Eingaben in der Konsole. Um den Spielmodi auszuwählen, können bspw. die Steuerbefehle AI (Computergegner), M (Mehrspieler) eingegeben werden.

Im Spiel können Figuren z.B. mit *A2* ausgewählt und durch eine Auswahl an Befehlen bewegt werden.

**Beschreibung der Architektur:**

Der Spielablauf startet in der Main, welche für das Laden des Hauptmenüs und das Verarbeiten der zugehörigen Eingaben zuständig ist. Dazu gehören unteranderem das Starten eines neuen Spiels und die Auswahl des Spielmodus oder das Laden einer bestehenden Datei.

Beendet man vorzeitig ein Spiel, kann man dieses entweder speichern oder verwerfen und wird zur Main weitergeleitet. Gewinnt oder verliert man, wird man ebenfalls weitergeleitet.

Neben den in der Grafik dargestellten Klassen spielen figure.py, pawn.py und const.py ebenfalls eine Rolle. Während const.py ausschließlich Konstanten speichert, sorgen figure.py und pawn.py für eine objektorientierte Verwaltung der Spielfiguren. Dabei ist die Superklasse figure.py.

Der objektorientierte Ansatz bietet Möglichkeiten für Erweiterungen wie z.B. mehr Figuren.

**Veränderung an der Vorlage:**

Move\_to() wurde in check\_move\_result() umbenannt, da die vorgedachte Funktion vom move\_handler() & make\_move() in der Game-Klasse übernommen wurde und die Methode nur noch überprüft, ob der gemachte Zug gültig ist und ob dieser zum Sieg führt. Außerdem wurde sie in Pawn-Klasse verschoben, da die Funktion spezifisch für jeden Figurtyp ist.

Die Exception-Klasse wurde nicht verwendet, da dies durch vollständige Fallunterscheidung redundant ist.

Weiterhin wurden zusätzliche Dateien hinzugefügt (Game.py, Save.py, Tests, main.py & ein Savefileordner.

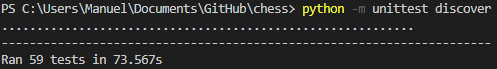
**Beschreibung des Computergegners (4er Gruppen):**

Der Grundgedanke des Computergegners besteht aus der Zuweisung von Vertrauenswerten für jeden Zug. Bringt ein Spielzug die KI näher an den Sieg, so erhält dieser einen höheren Vertrauenswert. Falls diese gleich hoch sind, wird randomisiert ausgewählt.

**Log von einem Spiel:**

* Der Log eines Spieldurchlaufs kann in *game\_log.txt* eingesehen werden.

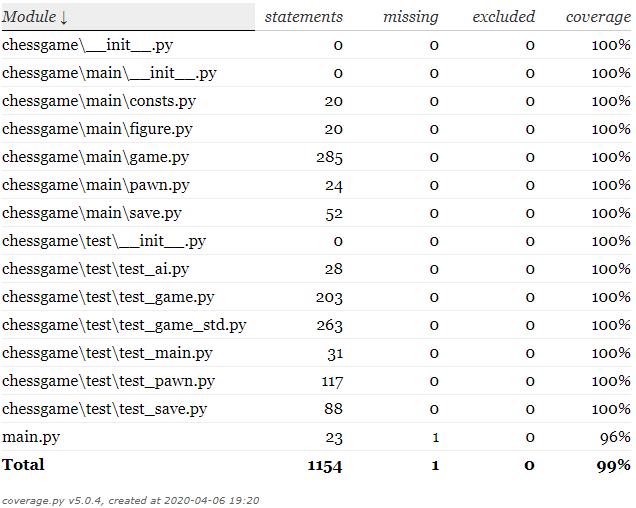
**Log von den Tests:**



**Bewertung der Testergebnisse:**

Alle Tests sind erfolgreich.

**Code-Coverage Ausgabe:**



**Bewertung der Coverage und Sinnvollheit der Tests:**

Die Coverage liegt bei 99%, da alle Methoden getestet wurden. Alle ersichtlichen Fälle wurden innerhalb der Tests simuliert. Falls Fehler aufgetreten sind, wurden diese behoben.

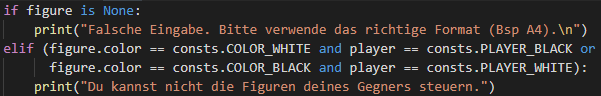
**Bewertung der Fehlersicherheit:**

Bei sämtlichen Eingaben werden diese vor ihrer Verarbeitung auf deren Richtigkeit geprüft.

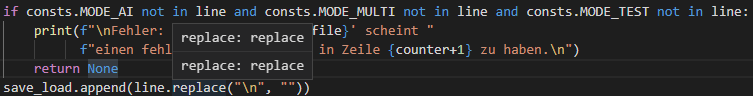
Bei falschen Eingaben wird der Benutzer darauf hingewiesen.

Hier wird die Auswahl der Figur auf ihre Gültigkeit überprüft.

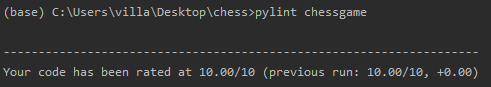
Sollte es die Figur eines Gegners sein oder an genannter Stelle keine Figur stehen, so wird eine Fehlermeldung ausgegeben und die Eingabe kann wiederholt werden.



In diesem Abschnitt wird überprüft, ob sich in der ersten Zeile eines gespeicherten Spiels ein gültiger Spielmodus befindet. Ist dies nicht der Fall, wird der Benutzer darauf hingewiesen.



**Pylint Ausgabe:**



**Bewertung der Code-Qualität/Lesbarkeit:**

Es gibt keine Pylint-Warnungen. Variablen-, Klassen- und Funktionsnamen wurden nach ihrer Funktion vergeben. Jede Methode jede Klasse und jedes Modul besitzt einen einheitlich formulierten Docstring. Außerdem ist der Code ist vollständig testbar und eine Erweiterung ist durch eine objektorientierte Realisierung leicht möglich.

**Bewertungstabelle:**

* Ausgefüllte Bewertungstabelle

**Externe Quellen:**

<https://gist.github.com/metatoaster/64139971b53ad728dba636e34b8a5558>

Für die Tests wurden aus dieser Quelle eine Klasse und mehrere Methoden verwendet.