Vorgehensplan

23.04.2021:

- Anforderungsanalyse
  - Anwendungsfalldiagramm ✔
  - Story-Cards
- Vorgehensplan ✔
- Einrichtung der Infrastruktur ✔

1. Iteration (12.04. - 16.05.), Präsentation & Theorie-Abfrage am 19.05.
Spiel auf Konsole; Mensch-gegen-Mensch; ungültige Züge nicht zugelassen

- Aktuelle Aufgaben:
  - Positionen aus Figuren entfernen und durch Boardpositionen ersetzen, CoreGame gibt stattdessen Positionen mit (Achtung: Figur führt den Zug noch nicht aus!!!) [Lydia]
  - In Methoden auslagern: Feld frei, Gegnerischer Spieler [Vincent]
- Methoden für
  - Rochade
  - en Passant (Methode gibt zurück, ob Bauer geschlagen en Passant geschlagen werden kann)
  - Umwandlung eines Bauern (Eingabe: z.B. e7-e8Q; kann jedem Zug hinzugefügt werden; Standard: Dame) --> *parse() erweitern*
  - Zug nur gültig, wenn eigener König dadurch nicht bedroht wird
  - Schach, Schach-Matt, Unentschieden
- Konsolenausgabe:
  - syntaktisch fehlerhaft: "!Invalid move"
  - regelwidrig: "!Move not allowed"
  - gültig: "![Zug wiederholt]"
  - Eingabe „beaten“ --> Auflistung geschlagene Figuren
  - Schach/Schach matt/Unentschieden --> Meldung
    nach Schach matt/Unentschieden: Spiel beenden
- Modus "--simple"
  - direkt Mensch-gegen-Mensch-Spiel beginnen
  - Ausrufezeichen nur bei „!Invalid move“ und „!Move not allowed“ und “![Zug wiederholt]“
  - Meldungen auf Standardausgabe (stdout) ausgeben, durch Zeilenumbruch beendet
  - andere Ausgaben ignorieren
  - Eingaben: nur die spezifizierten Eingaben in Form von Zügen
- Tests (--> Tutorial)
  - Abdeckung von 90 % des Codes (ohne GUI Klassen) durch JUnit-Tests (JaCoCo)
  - PMD-Regeln für Code und Testfälle einhalten
- Dokumentation
  - Anforderungsanalyse
    - Anwendungsfalldiagramm
    - Story Cards
  - Bedienungsanleitung.pdf
    - alle Funktionen des Systems kurz vorstellen
    - erklären, wie diese angesteuert werden können
  - Architektur.pdf
    - grundsätzliche Ideen hinter der gewählten Architektur
    - In welche Komponenten wurde das System eingeteilt?

        - Welche Klassen dienen der Umsetzung der einzelnen Komponenten?
        - Welche Kommunikation findet zwischen den einzelnen Klassen statt?
        - Inwiefern ist die Architektur erweiterbar / einfach wartbar?
        - ggf. UML-Diagramme
    - Javadoc-Kommentare: Alle öffentlichen Klassen, Methoden (abgesehen von Gettern und Settern) und Attribute

2. Iteration (17.05. - 13.06.), Präsentation am 16.06.
2D-GUI; Spiel gegen den Computer (einfache KI)

- Zu Programmstart kann gewählt werden, ob gegen den Computer oder gegen einen Menschen gespielt wird. Bei einem Spiel gegen den Computer kann außerdem die Farbe gewählt werden, in der der Mensch spielt.
- Die GUI zeigt ein Schachbrett in der Draufsicht. Die Figuren sind durch die gängigen Symbole dargestellt.
- Wahlweise (jederzeit umstellbar) kann das Schachbrett nach dem Zug gedreht werden, sodass sich die Seite des aktiven Spielers unten befindet.
- Durch Anklicken einer eigenen Figur gefolgt von einem Feld, auf das gezogen werden kann, wird ein Zug ausgeführt.
- Ungültige Zugversuche werden ignoriert.
- Neben dem Schachbrett befindet sich ein Feld, in dem die geschlagenen Figuren angezeigt werden.
- Das Spiel verfügt über eine Historie, wo die erfolgten Züge in der gleichen Notation angegeben werden, wie sie im konsolenbasierten Frontend verwendet wird.
- Klickt der Spieler, der gerade am Zug ist, auf eine seiner Figuren, werden aller Felder, auf die er ziehen kann, grafisch hervorgehoben. Dieses Hervorheben kann jedoch im Einstellungsdialog jederzeit aus- und wieder angeschaltet werden.
- Ebenso kann jederzeit eingestellt werden, ob nach dem Anklicken einer Figur nochmals eine andere Figur ausgewählt werden darf.
- Ebenso einstellbar ist die Anzeige, dass sich einer der Spieler im Schach befindet.
- Bei Schachmatt oder Unentschieden teilt das Spiel dies mittels einer Meldung mit. Daraufhin bleibt der aktuelle Stand der Figuren erhalten, es gibt jedoch keine Möglichkeit mehr die Figuren zu bewegen.
- Es kann jederzeit zum Auswahlbildschirm des Spiels zurückgekehrt werden und von dort aus ein neues Spiel begonnen werden.
- Für das einfache Spiel gegen den Computer soll der Computer aus allen Zügen, die er aktuell ausführen kann, den wählen, der aktuell am sinnvollsten erscheint. Eine Verbesserung kann als Zusatzfeature umgesetzt werden. Die Bewertung der Züge soll durch Implementierung einer Stellungsbewertung erfolgen.

3. Iteration (14.06. - 14.07.), Endpräsentation am 14.07.

- Wahlpflicht-Features (10 Einheiten)
    - Verbesserte KI mithilfe Min-/Max-Suche mit α-β-Pruning (5)
    - 3D-GUI (5)
    - Eindimensionales Schach (5)
    - Rückgängig-Machen von Zügen (3)
    - Speichern/Laden von Spielen (3)
    - Schachuhren (2)
    - Zweisprachigkeit (2)
    - Resizeable GUI (1)
- Netzwerkspiel (mit dem Spiel der anderen Gruppe kompatibel)