Software Engineering 1
Abgabedokument
Teilaufgabe 1
(Anforderungsanalyse und Planungsphase)

Persönliche Daten, bitte vollständig ausfüllen:

Nachname, Vorname:	Lee, Jin-Jin
Matrikelnummer:	11913405
E-Mail-Adresse:	a11913405@unet.univie.ac.at
Datum:	31.10.2021

Aufgabe 1: Anforderungsanalyse (2 Punkte)

Analysieren der Spielidee und des Netzwerkprotokolls um 8 Anforderungen (bestehend zumindest aus 3 funktionalen, 3 nichtfunktionalen und einer Designbedingung) nach den folgenden Kriterien zu dokumentieren. Achten Sie darauf den im Skriptum und der Vorlesung behandelten Qualitätsaspekten Genüge zu tun.

Typ der Anforderung: funktional

Anforderung 1

- **Beschreibung:** Kartengenerierung – Die KI muss vor Spielbeginn eine zufallsgenerierte 4 x 8 Kartenhälfte erstellen, damit das Spiel beginnen kann.
- **Bezugsquelle:** „Die Karte, auf welcher gespielt wird, ist hierbei nicht fest vorgegeben, sondern wird von beiden KIs selbstständig und automatisch beim Start des Spiels erstellt (jeweils eine Hälfte der Karte).“

Anforderung 2

- **Beschreibung:** Bewegung der Spielfiguren – Die KI kann sich entweder waagrecht oder senkrecht, aber nicht diagonal auf der Karte bewegen.
- **Bezugsquelle:** „Die Karten sind in Felder aufgeteilt, zwischen denen sich die Spielfiguren bewegen können (waagrecht und senkrecht).“

Anforderung 3

- **Beschreibung:** Spielstatus anzeigen – Der Client kann den Spielverlauf graphisch darstellen, damit die menschlichen Spieler die Spiele mitverfolgen können.
- **Bezugsquelle:** „Während des Spiels müssen die Karte und deren bekannte Eigenschaften auf den Rechnern (den Clients) der beiden menschlichen Spieler visualisiert werden, z.B., gegnerische Spielfiguren (beide Spielfiguren sind immer für beide KIs sichtbar), Burgen, Felder (und deren jeweiliges Terrain) etc.“

Typ der Anforderung: nicht funktional

Anforderung 4

- **Beschreibung:** Begrenzte Spieldauer - Der Server beendet Spiele, die länger als 200 Spielaktionen dauern, damit es für die Zuschauer spannend bleibt.
- **Bezugsquelle:** „Um die Spiele für die Zuschauer spannend zu gestalten, wurde festgelegt, dass ein Spiel insgesamt nicht länger als 200 Spielaktionen dauern darf und eine KI für jede dieser rundenbasierten Spielaktion nicht mehr als 3 Sekunden Bedenkzeit zur Berechnung erhält (relevant sind hierbei Spielerbewegung und Kartengenerierung).“

Anforderung 5

- **Beschreibung:** Spielaktion Berechnung – Die KI muss innerhalb von drei Sekunden eine neue Spielaktion berechnen können, damit das Spiel spannend bleibt.
- **Bezugsquelle:** „Um die Spiele für die Zuschauer spannend zu gestalten, wurde festgelegt, dass ein Spiel insgesamt nicht länger als 200 Spielaktionen dauern darf und eine KI für jede dieser rundenbasierten Spielaktion nicht mehr als 3 Sekunden Bedenkzeit zur Berechnung erhält (relevant sind hierbei Spielerbewegung und Kartengenerierung).“

Anforderung 6

- **Beschreibung:** Abfrage eines Spielstatus - Der Client muss mindestens 0,4 Sekunden warten bis er erneut nach dem Status fragen kann, damit der Server nicht überlastet wird.
- **Bezugsquelle:** „Um zu verhindern, dass der Server überlastet wird sollte zwischen zwei vom gleichen Client durchgeführten Abfragen zum Spielstatus mindestens eine Zeitspanne von 0,4 Sekunden vergehen.

Anforderung 7

- **Beschreibung:** Ressourcen schonen – Der Server entfernt ältere Spiele, wenn mehr als 999 Spiele parallel auf den Server laufen um Ressourcen zu schonen.
- **Bezugsquelle:** „Um die Ressourcen des Servers zu schonen gilt die Einschränkung, dass maximal 999 Spiele parallel ausgeführt werden dürfen.“

Typ der Anforderung: Designbedingung

Anforderung 8

- **Beschreibung:** Nachrichtenaustausch – Der Nachrichtenaustausch erfolgt über HTTP Protokoll.
- **Bezugsquelle:** „Die technologische Basis des Nachrichtenaustauschs stellt eine Restschnittstelle dar, daher es wird das HTTP Protokoll verwendet sowie die zugehörigen Operationen GET und POST.“

Aufgabe 2: Anforderungsdokumentation (2 Punkte)

Dokumentation einer zum relevanten Bereich passenden Anforderung nach dem vorgegebenen Schema. Ziehen Sie eine Anforderung heran, für die alle Bestandteile der Vorlage mit relevantem Inhalt befüllt werden können. Wir empfehlen hierzu eine **funktionale** Anforderung auszuwählen.

Dokumentation Anforderung

- **Name:** Spielstatus anzeigen
- **Beschreibung und Priorität:** – Der Client ist in der Lage das Spiel graphisch darzustellen und fragt nach jeder Spielaktion nach dem Spielstatus um alle Informationen korrekt in der CLI abzubilden.
  Priorität: Hoch
- **Relevante Anforderungen:**
  - Kartengenerierung: Wir brauchen eine Karte, damit das Spiel beginnen kann. Die Karte wird in der CLI angezeigt, wenn das Spiel beginnt.
  - Bewegung der Spielfiguren: Der Spielstatus muss nach jedem Spielzug geupdatet und neu visualisiert werden.
  - Begrenzte Spieldauer: Überschreitet ein Spiel 200 Spielaktionen so wird das Spiel beendet und eine Benachrichtigung soll angezeigt werden.
  - Spielaktion Berechnung: Braucht ein Client länger als 3 Sekunden um eine Spielaktion zu berechnen so gewinnt der andere Spieler und eine Benachrichtigung soll angezeigt werden.
  - Abfrage eines Spielstatus: Der Client muss mindestens 0,4 Sekunden warten bis er erneut nach dem Status fragen kann, damit der Server nicht überlastet wird.
  - Nachrichtenaustausch: Der Client kann mit einem HTTP GET Request nach dem Spielstatus fragen.
- **Relevante Business Rules**:
  - Die Spielkarte ist 4x16 oder 8x8 groß
  - Das Spiel darf nicht länger als 200 Spielaktionen dauern
  - Die Berechnung einer Spielaktion darf nicht länger als 3 Sekunden dauern
  - Zwischen zwei Statusabfragen müssen mindestens 0,4 Sekunden liegen
  - Das Spiel ist rundenbasiert
  -

- **Impuls/Ergebnis - Typisches Szenario:** [Welche Schritte werden durchlaufen, um das der Anforderung zugehörige Verhalten auszulösen bzw. zu nützen und was ist das unmittelbare Ergebnis jedes Schrittes.]
  ***Vorbedingungen*:**
  - Spiel wurde erzeugt
  - Spieler hat eine eindeutige SpielID
  - Gegner für die Spielrunde wurde gefunden
  - Kartenhälften wurden generiert und zusammengefügt

  ***Hauptsächlicher* Ablauf**:
  - Impuls: Client 1 sendet Spielaktion
  - Ergebnis: Spielstatus hat sich verändert
  - Impuls: Client 2 fragt Spielstatus ab
  - Ergebnis: Server sendet Spielstatus
  - Impuls: Client 2 liest Spielstatus aus

- Ergebnis: Client 2 zeigt neue Karte an
- Impuls: Client 2 sendet Spielaktion
- Ergebnis: Spielstatus hat sich verändert
- Impuls: Client 1 fragt Spielstatus ab
- Ergebnis: Server sendet Spielstatus
- Impuls: Client 1 liest Spielstatus aus
- Ergebnis: Client 1 zeigt neue Karte an
- ***Nachbedingungen***:
  - Spieler befindet sich auf einem anderen Feld
  - Spieler findet den Schatz/Burg
  - Der gegnerische Spieler ist am Zug

- **Impuls/Ergebnis - Alternativszenario:** [Ein Alternativszenario, das vergleichbare Vorbedingungen aufweist, jedoch sich im Ablauf und Nachbedingungen unterscheidet].

  Vorbedingungen:
  - Spiel wurde erzeugt
  - Spieler hat eine eindeutige SpielID
  - Gegner für die Spielrunde wurde gefunden
  - Kartenhälften wurden generiert und zusammengefügt

  ***Hauptsächlicher* Ablauf**:
  - Impuls: Client 1 sendet Spielaktion
  - Ergebnis: Spielstatus hat sich verändert
  - Impuls: Client 2 fragt Spielstatus ab
  - Ergebnis: Server sendet Spielstatus
  - Impuls: Client 2 liest Spielstatus aus
  - Ergebnis: Client 2 zeigt neue Karte an
  - Impuls: Client 2 braucht länger als 3 Sekunden um eine Spielaktion zu berechnen
  - Ergebnis: Client 2 verliert das Spiel
  - Impuls: Client 1 fragt Spielstatus ab
  - Ergebnis: Server sendet Spielstatus
  - Impuls: Client 1 liest Spielstatus aus
  - Ergebnis: Client 1 zeigt Gewinnbenachrichtigung an

  Nachbedingungen:
  - Spieler befindet sich auf einem anderen Feld
  - Der gegnerische Spieler ist am Zug
  - Client 1 hat das Spiel gewonnen
  - Spiel wurde beendet

  - **Impuls/Ergebnis - Fehlerfall:** Ein Client will nochmal den Spielstatus abfragen, obwohl noch keine 0,4 Sekunden vergangen sind.

- ***Vorbedingungen***:
  - Spiel wurde erzeugt
  - Spieler hat eine eindeutige SpielID
  - Gegner für die Spielrunde wurde gefunden
  - Kartenhälften wurden generiert und zusammengefügt
  - Client 1 hat Spielstatus abgefragt

***Hauptsächlicher* Ablauf**:
- o Impuls: Client 1 fragt zuerst ob er dran ist
- o Ergebnis: Client 1 ist nicht dran
- o Impuls: Client 1 fragt nochmal ob er dran ist
- o Ergebnis: Client 1 ist dran
- o Impuls: Client 1 fragt Spielstatus ab
- o Ergebnis: Server sendet Spielstatus
- o Impuls: Client 1 liest Spielstatus aus
- oErgebnis: Client 1 zeigt neue Karte an

Nachbedingungen:
- o Spieler befindet sich auf einem anderen Feld
- o Der gegnerische Spieler ist am Zug

- **Benutzergeschichten:**
  - o Als menschlicher Spieler möchte ich eine graphische Visualisierung des Spieles sehen, damit ich nachvollziehen kann wieso der ich/der Gegner gewonnen/verloren hat.
  - o Als Client möchte ich nach jeder Spielaktion den neuen Spielstatus erhalten, damit ich Spielkarte aktualisieren kann.
  - o Als KI möchte ich nach jeder Spielaktion den neuen Spielstatus erhalten, damit ich eine neue Spielaktion berechnen kann.

- **Benutzerschnittstelle:** Der Spielstatus soll in der CLI angezeigt werden. Wenn das Spiel noch läuft, soll die Spielkarte mit Spielfiguren und Burg angezeigt werden. Wenn einer beiden Spieler gewinnt/verliert soll anstatt der Karte eine Benachrichtigung angezeigt werden.

- **Externe Schnittstellen**: Der Client sendet ein HTTP GET Request an folgenden Endpunkt: http(s)://<domain>:<port>/games/<SpielID>/states/<SpielerID> um den Spielstatus zu erhalten. Der Server sender dann GameState-Object zurück aus dem der SpielStatus ausgelesen werden kann.

Aufgabe 3: Architektur entwerfen, modellieren und validieren (10 Punkte)

Modellieren Sie händisch alle notwendigen Packages, Klassen und deren Methoden (samt Beziehungen) als zwei UML Klassendiagramme. Achten Sie darauf, dass die Modelle sinnvoll benannte Packages, Klassen, Methoden und Felder beinhalten und die Vorgaben der Spielidee bzw. des Netzwerkprotokolls vollständig in sinnvoller Granularität abgedeckt werden.

Basierend auf dem Klassendiagramm: Erstellen Sie zwei Sequenzdiagramme zu den beiden in der Übungsangabe vorgegebenen Aspekten. Alle erstellten Diagramme sollten semantisch und syntaktisch korrekt sowie untereinander konsistent sein.

TIPP: Beachten Sie zur Ausarbeitung das auf Moodle zu Verfügung gestellte auszugsweise abgebildete 4+1 Diagrammbeispiel. Dieses sollte als Vorlage dienen, und verdeutlicht welche Erwartungen an das Klassendiagramm beziehungsweise die dazugehörigen Sequenzdiagramme gestellt werden (Darstellung, Inhalte, Detailgrad, etc.) und wie diese zusammenhängen.

[Alle vier Diagramme mit einer kurzen Beschreibung hier einfügen. Sollten die Diagramme zu viel Platz benötigen können Sie diese auch als separate SVG Dateien (empfohlenen) oder PNG Dateien im Dokumentationsordner (siehe GitLab Repo) einfügen. Achten Sie bei der Verwendung von SVG Dateien darauf, dass sich diese fehlerfrei darstellen lassen.]