Software Engineering 1
Abgabedokument
Teilaufgabe 1
(Anforderungsanalyse und Planungsphase)

Persönliche Daten, bitte vollständig ausfüllen:

Nachname, Vorname:	Jäger, Wolfgang
Matrikelnummer:	11817359
E-Mail-Adresse:	a11817359@unet.univie.ac.at
Datum:	26.10.2023

Aufgabe 1: Anforderungsanalyse

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:** Generierung Spielfeldhälfte – Das Spielfeld welches von beiden Clients genutzt wird, soll von der KI's selbstständig generiert werden.
- **Bezugsquelle:** [Spielidee Absatz 2, „Nach Start des Clients registrieren sich die KIs für das Spiel am Server und erstellen/tauschen danach mit dem Server Kartenhälften aus."]

Anforderung 2

- **Beschreibung:** Startpunkt – Die KI wählt ihren Startpunkt selbst und damit auch ihre Burg selbst.
- **Bezugsquelle:** [Spielidee Absatz 3, „Nach dem Kartenhälftenaustausch starten die Spielfiguren jeder KI jeweils auf einer selbst ausgewählten Position (ihrer Burg) auf der von ihr erstellten Kartenhälfte."]

Anforderung 3
- **Beschreibung:** Wasserfelder – Wird ein Wasserfeld von einem Client betreten, verliert die jeweilige KI automatisch.
- **Bezugsquelle:** [Spielidee Absatz 12, „Bewegt sich die Spielfigur einer KI in ein Wasserfeld, verliert die KI automatisch.“]

Anforderung 4
- **Beschreibung:** Bewegung von Wiesenfeld – Wenn eine KI sich auf einem Wiesenfeld befindet kann diese sich durch eine Bewegungsaktion auf ein angrenzendes Feld hinbewegen.
- **Bezugsquelle:** [Spielidee Absatz 13, „Wasser und Wiesen können jeweils mit einer Bewegungsaktion betreten und im Fall von Wiesen auch wieder verlassen werden.“]

Typ der Anforderung: nicht funktional
Anforderung 5
- **Beschreibung:** Bewegung Bergfelder – Es sollen vier Bewegungsbefehle von Nöten sein, um von einem Spieler (Client) ein Bergfeld zu betreten und zu verlassen.
- **Bezugsquelle:** [Spielidee Absatz 13, „Im Gegensatz zu Wiesen benötigen Berge zwei Bewegungsaktionen, um diese (das Bergfeld) zu betreten und zwei zusätzliche, um den Berg wieder zu verlassen.“]

Anforderung 6
- **Beschreibung:** Kartengröße – Die von einem Client erstellte Karte der KI soll 5 x 10 Felder groß sein.
- Bezugsquelle: [Spielidee Absatz 11, „Hierzu erstellt jede der beiden KIs zufällig eine Hälfte der finalen Spielkarte (mit je 5 x 10 Feldern).“]

Anforderung 7
- **Beschreibung:** Spieleaktionen – Ein Spiel zwischen zwei Clients darf nicht länger als 320 Spielaktionen dauern.
- **Bezugsquelle:** [Spielidee Absatz 10, „Um die Spiele für die Zuschauer spannend zu gestalten, wurde festgelegt, dass ein Spiel insgesamt nicht länger als 320 Spielaktionen (und damit 320 Runden) dauern darf.“]

Typ der Anforderung: Designbedingung
Anforderung 8
- **Beschreibung:** Nachrichtenaustasuch – Für den Nachrichtenaustasuch zwischen Client und Server soll das HTTP Protokoll verwendet werden.
- **Bezugsquelle:** [Netzwerkprotokoll (Einleitung zum Netzwerkprotokoll) zweiter Absatz, „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

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.

- **Name:** Bewegung von Wiesenfeld
- **Beschreibung und Priorität:** Befindet Sich ein Client (KI) auf einem Wiesenfeld so kann er sich waagrecht und senkrecht auf ein neues Feld bewegen. Bewegungen auf ein Wasserfeld resultieren in einer Niederlage für den jeweiligen Client.
  Priorität: Hoch
- **Relevante Anforderungen:**
  - Generierung Spielhälfte – Die Generierung einer korrekten Spielhälfte ist essenziel für Bewegungen über das Spielfeld.
  - Wasserfelder – Betreten von Wasserfeldern resultiert in einer Niederlage für den jeweiligen Client.

- **Relevante Business Rules**:
  - Da die Applikation in Java laufen wird, wird vorausgesetzt, dass auf ihrem Gerät eine passende Java-Version vorab installiert ist, um die Software überhaupt ausführen zu können.
  - Eine Verbindung zum Internet muss aufgebaut sein, um die Kommunikation zwischen Client und Server zu führen.
  - Jeweils ein Client bzw. eine KI hat 5 Sekunden Zeit um einen Bewegungsbefehl an den Server zu senden.
  - Jeweils eine Kartenhälfte muss aus 10% Bergfeldern, 48% Wiesenfeldern und 14% Wasserfeldern bestehen.
  - Es sollen immer 400ms vergehen bevor man wieder den Spielstatus beim Server abfragt.

- **Impuls/Ergebnis - Typisches Szenario:**
  ***Vorbedingungen*:**
  - Die Anwendung wurde vorab auf dem Gerät installiert und kann auch gestartet werden.
  - Zwei Clients verbinden sich durch Starten der Anwendung mit dem Server.
  - Die Kartenhälften werden erzeugt und ausgetauscht.
  - Ein Spieler ist am Zug und es wird eine Aktion vom Server erwartet.

Hauptsächlicher Ablauf:

- Impuls: Der Client, welcher am Zug ist, sendet dem Server seine Bewegung zu.
- Ergebnis: Der Server verarbeitet und überprüft die Bewegung und „bewegt“ den Spieler in die gewünschte Richtung.

- Impuls: Der andere Client ist am Zug und sendet ebenfalls eine Bewegung an den Server.
- Ergebnis: Die Spielfigur bewegt sich nach Überprüfung in die vom zweiten Client gewünschte Richtung.

***Nachbedingungen*:**

- Eine Runde an Bewegungen beiderseits wurde erfolgreich durchgeführt
- Die Clients sind beide noch im Spiel und können gewinnen.

- **Impuls/Ergebnis - Alternatives Szenario:**

***Vorbedingungen*:**

- Die Anwendung wurde vorab auf dem Gerät installiert und kann auch gestartet werden.

- Zwei Clients verbinden sich durch Starten der Anwendung mit dem Server.

- Die Kartenhälften werden erzeugt und ausgetauscht.

- Ein Spieler ist am Zug und es wird eine Aktion vom Server erwartet.

Hauptsächlicher Ablauf:

- Impuls: Der Client, welcher am Zug ist, sendet dem Server seine Bewegung zu.
- Ergebnis: Der Server verarbeitet und überprüft die Bewegung und „bewegt“ den Spieler in die gewünschte Richtung.

- Impuls: Der andere Client ist am Zug und sendet ebenfalls eine Bewegung, welche jedoch ein Wasserfeld betreten würde an den Server.
- Ergebnis: Die Spielfigur bewegt sich nach Überprüfung nicht mehr und der erste Client hat somit das Spiel gewonnen.

- Impuls: Der Server sendet an beide Clients den Spielstatus.
- Ergebnis: Der erste Client hat die Partie gewonnen und der Zweite verloren.

***Nachbedingungen*:**

- Eine Runde an Bewegungen beiderseits wurde nicht erfolgreich durchgeführt.
- Der erste Client hat gewonnen.
- Der zweite Client hat verloren.
- Die Anwendung wird von beiden Clients angehalten.

- **Impuls/Ergebnis - Fehlerfall:**
  ***Vorbedingungen*:**
    - Die Anwendung wurde vorab auf dem Gerät installiert und kann auch gestartet werden.
    - Zwei Clients verbinden sich durch Starten der Anwendung mit dem Server.
    - Die Kartenhälften werden erzeugt und ausgetauscht.
    - Ein Spieler ist am Zug und es wird eine Aktion vom Server erwartet.

  ***Hauptsächlicher* Ablauf**:
    - Impuls: Der Client, welcher am Zug ist, sendet dem Server seine Bewegung zu.
    - Ergebnis: Der Server verarbeitet und überprüft die Bewegung und „bewegt“ den Spieler in die gewünschte Richtung.
    - Impuls: Der andere Client ist am Zug und sendet ebenfalls eine Bewegung, welche jedoch das Spielfeld verlassen würde an den Server.
    - Ergebnis: Die Spielfigur bewegt sich nach Überprüfung nicht mehr und der erste Client hat somit das Spiel gewonnen.
    - Impuls: Der Server sendet an beide Clients den Spielstatus.
    - Ergebnis: Der erste Client hat die Partie gewonnen und der Zweite verloren.

  ***Nachbedingungen*:**
    - Eine Runde an Bewegungen beiderseits wurde nicht erfolgreich durchgeführt.
    - Der erste Client hat gewonnen.
    - Der zweite Client hat verloren.
    - Die Anwendung wird von beiden Clients angehalten.

- **Benutzergeschichten:**
    - Als Anwender möchte ich, dass sich die Position meiner Spielfigur auf ein möglichst effizientes Feld weiterbewegt um so schnell wie möglich den Schatz zu erreichen bzw. die gegnerische Burg zu erreichen.
    - 2. Als Client möchte ich, dass sich die Position meiner Spielfigur auf ein valides Feld weiterbewegt um das Spiel am laufen zu halten und keine Fehlermeldungen auftreten.
    - Als KI möchte ich, dass die zu absolvierenden Bewegungen mich am schnellsten zu den jeweiligen Zielen führen.
    - 4. Als Server möchte ich, dass sich ankommende Bewegungsbefehle als valide herausstellen, um das Spiel korrekt zum Ende führen zu können.

- **Benutzerschnittstelle:**

A … Bergfelder
… Wiesenfelder
_ … Wasserfelder
1 … eigene Position der KI
2 … Position vom Spielpartner
X … Schatz
M … Burg

```
# A # 1 A # A A A #
# A X # A M A A A #
_ A A A # # _ # _ A
_ _ _ _ # A A # _ #
# # _ # # # # # # _
# # A # A A # _ A #
_ A A A # # _ # _ A
_ _ _ _ # A A # _ #
# M _ # # # # # 2 _
# # A # A A # _ A #
```

- **Externe Schnittstellen:**
  - Server: Der Server stellt die Verbindung zwischen zwei Clients dar, welche über ihn kommunizieren. Es wird eine Verbindung zu der REST-Schnittstelle des Servers aufgebaut bei der die Kommunikation mittels HTTP durchgeführt wird mit den Operationen POST und GET.

  - Erstellung eines neuen Spiels: Um ein neues Spiel zu erstellen, so sendet man eine HTTP GET Operation (alternativ auch mit HTTP POST möglich) an folgenden Endpoint http(s)://<domain>:<port>/games

  - Registrierung eines Clients: Um einen Client zu registrieren, so sendet man einen HTTP POST Request mit einer XML Nachricht im Body an folgenden Endpunkt: http(s)://<domain>:<port>/games/<SpielID>/players

  - Übertragung einer von zwei Kartenhälften: Um eine Kartenhälfte an den Server zu schicken, so sendet man einen HTTP POST Request mit einer XML Nachricht im Body an folgenden Endpunkt: http(s)://<domain>:<port>/games/<SpielID>/halfmaps

  - Abfrage des Spielstatus: Um den aktuellen Spielstatus abzufragen, sendet man ein HTTP GET Request an folgenden Endpunkt: http(s)://<domain>:<port>/games/<SpielID>/states/<SpielerID>

  - Übertragung einer Bewegung: Eine Bewegung der KI wird an den Server gesendet indem man einen HTTP POST Request an folgenden Endpunkt sendet: http(s)://<domain>:<port>/games/<SpielID>/moves

Aufgabe 3: Architektur entwerfen, modellieren und validieren

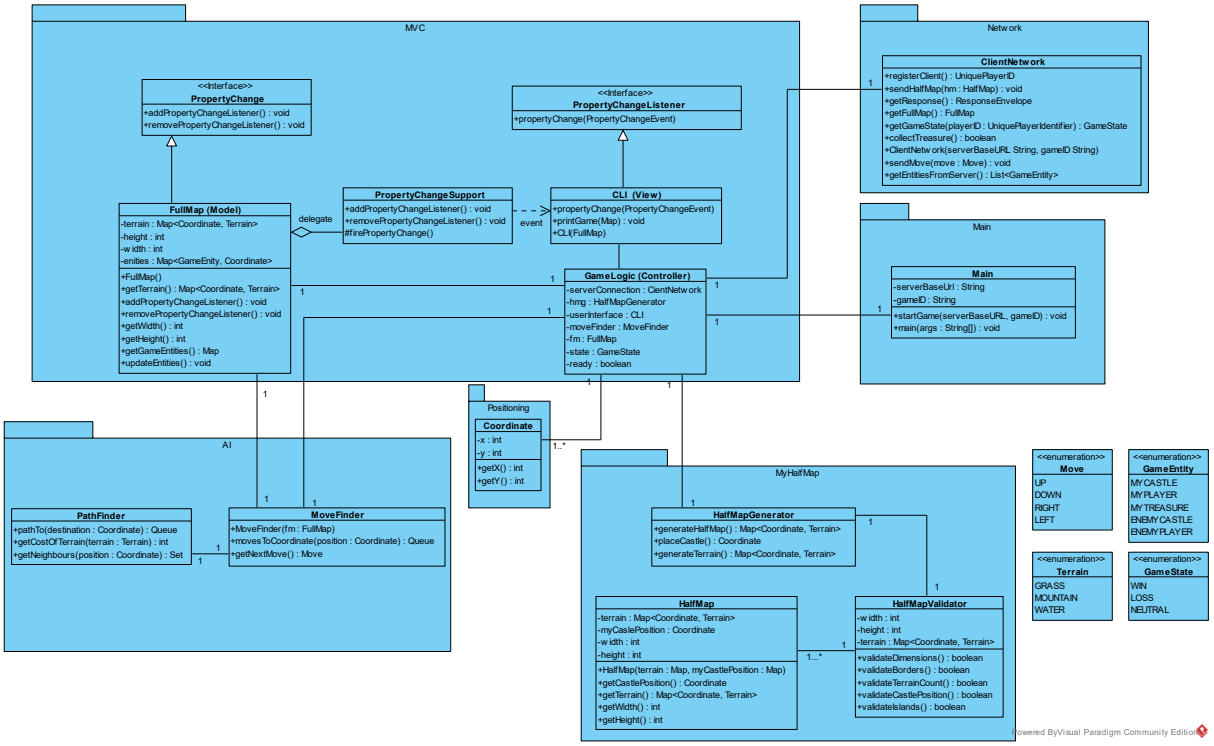
Modellieren Sie händisch alle notwendigen Packages, Klassen und deren Methoden (samt Beziehungen) als zwei UML Klassendiagramme. Achten Sie darauf, dass die Modelle sinnvoll benannte Packages, **Klassen**, **Methoden** (inkl. Parameter und Rückgaben) und **Felder** beinhalten und die Vorgaben der Spielidee 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.

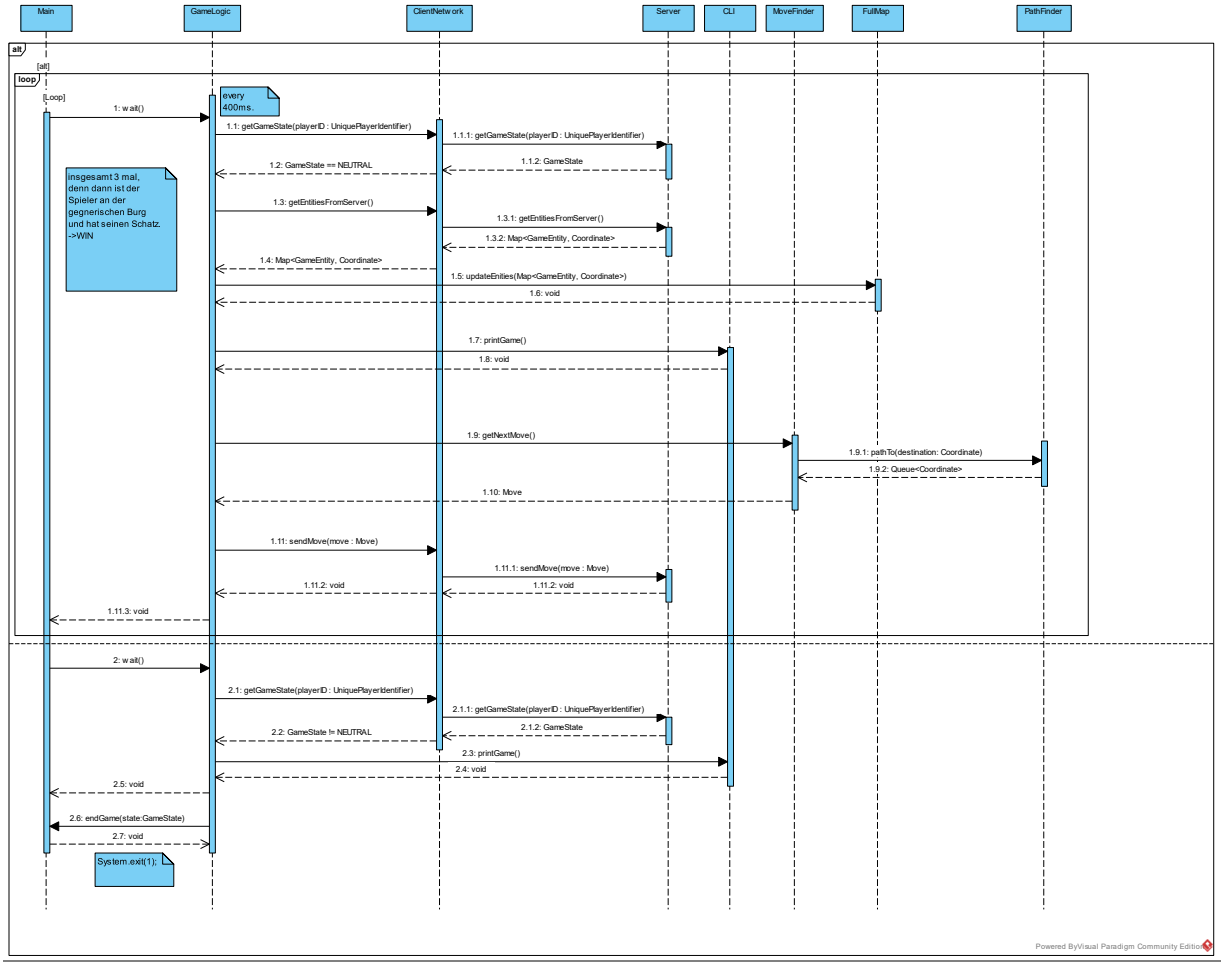
Wir bieten hierzu Unterstützung:

- Wie man die Modellierung und den Entwurf einer Architektur angeht wird im zugehörigen *Tutorial* besprochen samt einer *Ausgangsbasis* für den Client. Zum Nachlesen ist in der Angabe ein passendes Skriptum mit Beispielentwurf hinterlegt. Wenn Sie Ihr UML-Modellierungswissen ausbauen möchten/müssen wird im Skriptum zusätzlich auch dazu passende Literatur referenziert.
- Beachten Sie zur Ausarbeitung das auf Moodle zur Verfügung gestellte auszugsweise abgebildete 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.
- Für die Modellierung von Model-View-Controller (nur für Client verpflichtend) gibt es ein fertiges Beispieldiagramm in den Unterlagen. Sie können dieses als Grundlage heranziehen und für Ihre Architektur anpassen/integrieren.
- Für die Modellierung der für das Netzwerk relevanten Klassen gibt es im Netzwerkprotokoll auf Moodle eine Anleitung. Es ist nicht notwendig Netzwerknachrichten als Datenklassen in die Diagramme aufzunehmen. Wenn diese an einer Stelle genützt werden (z.B. als Parameter) kann der Namen der Klassen als Type angeführt werden. *Empfehlung*: Eigene auf konkrete Use Cases spezialisierte (und deshalb besser geeignete) Datenklassen zu erstellen statt nur die bereitgestellten Netzwerkklassen zu „kopieren“ da diese nur auf den Datenaustauch fokussiert sind.
  Im Netzwerkprotokoll wird auch der Ablauf (Wer, Wann, Was) der Interaktion zwischen Clients/Server dargestellt. Nützen Sie das um diesen nachvollziehen und vor allem um diesen Ablauf auch in den Diagrammen zu berücksichtigen und auch nicht wesentliches für die Sequenzdiagrammabläufe zu übersehen.

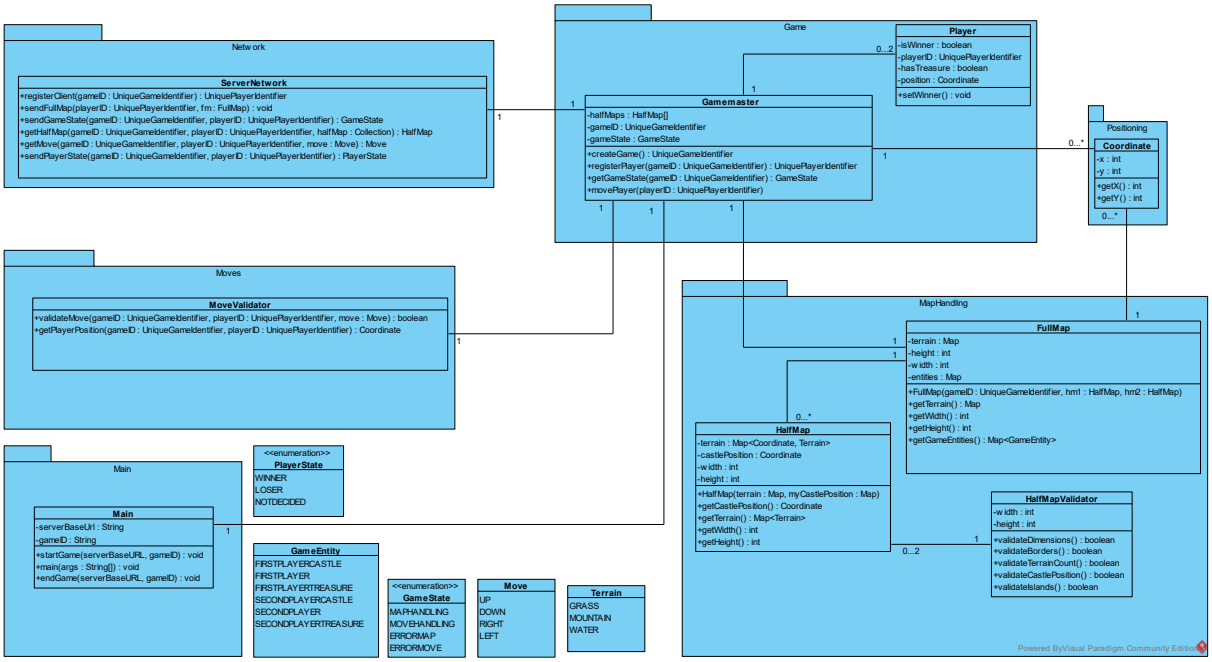
Klassendiagramm Client:



Sequenzdiagramm Client:



Klassendiagramm Server:



Sequenzdiagramm Server: