Software Engineering 1

Abgabedokument

Teilaufgabe 1

(Anforderungsanalyse und Planungsphase)

Persönliche Daten, bitte vollständig ausfüllen:

Nachname, Vorname:	Hajas Flórián
Matrikelnummer:	01207533
E-Mail-Adresse:	a01207533@unet.univie.ac.at
Datum:	30.10.2019

Aufgabe 1: Anforderungsanalyse (2 Punkte)

Analysieren der Spielidee und des Netzwerkprotokolls um 10 Anforderungen (bestehend zumindest aus 3 funktionalen, 3 nichtfunktionalen und einer Designbedingung) nach den folgenden Kriterien zu dokumentieren.

Anforderung 1

- **Beschreibung:** Kriterien zur Kartenhälfte - Bei der Kartengenerierung müssen die KIs jeweils mindestens 3 Berg-, 15 Wiesen- und 4 Wasserfelder erstellen, damit die gesamte Karte eine gewisse Abwechslung hat.
- **Bezugsquelle:** Aus dem Spielidee: *„jede Kartenhälfte muss mindestens 3 Bergfelder, 15 Wiesenfelder und 4 Wasserfelder beinhalten*"
- **Typ der Anforderung:** Nicht funktional

Anforderung 2

- **Beschreibung:** Schatz finden - Jede KI muss erstens seinen eigenen Schatz finden und aufnehmen, damit sie den im Weiteren zur Eroberung der gegnerischen Burg verwenden können.
- **Bezugsquelle:** Aus dem Spielidee: „*Ein Schatz kann nur von der Spielfigur der zugeordneten KI „aufgenommen*" *und „verwendet*" *werden.*
- **Typ der Anforderung:** Funktional

Anforderung 3

- **Beschreibung:** Datenformat zur Kommunikation - Der Client und der Server werden die Daten im XML Format austauschen, um die Kommunikation zwischen einander zu schaffen.
- **Bezugsquelle:** Aus dem Netzwerkprotokoll: „*Die ausgetauschten Daten bzw. Nachrichten werden im XML Format definiert bzw. erwartet.*"
- **Typ der Anforderung:** Designbedingung

Anforderung 4

- **Beschreibung:** Limit des Spiels - Der Server muss ein Spiel auf 200 Runden limitieren, damit ein Spiel nicht all zu lange dauert und es für die Zuschauer spannend bleibt.
- **Bezugsquelle:** Aus dem Spielidee: „…*ein Spiel nicht länger als 200 Spielaktionen dauern darf…*"
- **Typ der Anforderung:** Nicht funktional

Anforderung 5

- **Beschreibung:** Limit der Spielaktionen – Der Client muss jede Spielaktion innerhalb von 3 Sekunden durchführen, damit er nicht die Regeln verletzt und verliert.
- **Bezugsquelle:** Aus dem Spielidee: „…*eine KI für jede Aktion … nicht mehr als 3 Sekunden Bedenkzeit zur Berechnung erhält.*
- **Typ der Anforderung:** Nicht funktional

Anforderung 6

- **Beschreibung:** Kartengröße – Die KIs (Clients) generieren die Karte mit einer Größe von 64 Feldern (8 x 8 oder 4 x 16)
- **Bezugsquelle:** Aus dem Spielidee: *„Die gesamte Karte hat immer eine fixe Ausdehnung von 64 Feldern.“*
- **Typ der Anforderung:** Nicht funktional

Anforderung 7

- **Beschreibung:** Betreten auf Wasser vermeiden - Die KI muss die Bewegung immer so ausrechnen, dass der Spielfigur in keinem Fall auf einen Wasserfeld tritt.
- **Bezugsquelle:** Aus dem Spielidee: *„…Wasser unter keinen Umständen betreten…“*
- **Typ der Anforderung:** Funktional

Anforderung 8

- **Beschreibung:** Burg und Schatz auf unterschiedlichen Wiesenfelder platzieren - Die KIs müssen die Karte so generieren, dass der Burg und der Schatz nur auf Wiesenfeldern platzieren dürfen und in keinem Fall auf demselben Feld gestellt werden dürfen.
- **Bezugsquelle:** Aus dem Spielidee: *„…Burg und der Schatz nur auf Wiesenfeldern und nicht auf demselben Feld platziert werden…“*
- **Typ der Anforderung:** Funktional

Anforderung 9

- **Beschreibung:** Inselgenerierung vermeiden – Die KIs müssen die Kartengenerierung so durchführen, dass die Wiesen- und Bergfelder in gar keinem Fall von Wasser oder Kartengrenze umkreist werden.
- **Bezugsquelle:** Aus dem Spielidee: *„…keine Inseln…“*
- **Typ der Anforderung:** Funktional

Anforderung 10

- **Beschreibung:** Maximale Spielanzahl - Der Server darf maximal 999 Spiele parallel laufen lassen, um eine gewisse Performance dauerhaft leisten zu können.

- **Bezugsquelle:** Aus dem Netzwerkprotokoll: *„…999 Spiele parallel ausgeführt werden dürfen…“*
- **Typ der Anforderung:** Nicht funktional

Aufgabe 2: Anforderungsdokumentation (3 Punkte)

Dokumentation von drei in Teilaufgabe 1 identifizierten Anforderungen nach dem vorgegebenen Schema. Wir empfehlen hierzu **funktionale** Anforderungen heranzuziehen.

Dokumentation der Anforderung 1

- **Name:** Schatz finden
- **Beschreibung und Priorität:** Die KI ist in der Lage den Schatz zu finden. Die Schatzfindung läuft vollautomatisch ab. Die KI prüft anhand eines Algorithmus den kürzesten Weg für die Begehung ihrer eigenen Kartenhälfte, um den Schatz zu finden und aufzunehmen.
  Priorität: Hoch
- **Impuls/Ergebnis Listen:**
  - Impuls: Der Server versteckt einen Schatz auf beiden Kartenhälfte für die beiden KIs.
  - Ergebnis: Die KIs starten anhand eines Algorithmus die kürzesten Wege zu kalkulieren, um ihre Kartenhälfte und somit den Schatz zu erkundigen.
  - Impuls: Nach dem kalkulierten nächsten Schritt fragt der Client die nächste Bewegung an und übermittelt sie dem Server.
  - Ergebnis: Der Server antwortet mit Okay wenn der Client eine gültige anfrage stellt oder mit Error, wenn eine ungültige Anfrage zum Server ankommt.
  - Impuls: Die KI tritt auf einem Feld wo es einen Schatz auch gibt.
  - Ergebnis: Die KI nimmt den Schatz automatisch auf und der Schatz wird von der Karte entfernt. Somit kann die KI als nächstes die gegnerische Burg finden und aufdecken.
- **Benutzergeschichten:**
  - Als KI möchte ich meine Bewegung so kalkulieren, dass ich meine Kartenhälfte in möglichst wenigen Schritten begehe und somit mein Schatz finde.
  - Als KI möchte ich den Schatz möglichst schnell finden, um die gegnerische Burg aufdecken zu können.
- **Benutzerschnittstelle:**
  - Hierbei gibt es keine Benutzerschnittstelle, wo der User nach einer Aktivität die Schatzfindung beeinflussen kann, aber bei der Aufgabe 4 kann man nachvollziehen wie die einzelnen Komponenten miteinander verbunden sind, um die CLI für den User anzupassen.
- **Externe Schnittstellen:**
  - *Schnittstelle Netzwerkkommunikation:* Es wird die HTTP Protokoll verwendet, um die XML Daten über die Bewegung von Client zu Server zu schicken und umgekehrt die Antwort vom Server zu bekommen und verarbeiten zu können.

Dokumentation Anforderung 2

- **Name:** Insel vermeiden
- **Beschreibung und Priorität:** Die KI muss verschiedene Kriterien bei der Generierung ihrer Kartenhälfte berücksichtigen. Eine der wichtigsten Kriterien ist, dass die Wiesenfelder in keinem Fall von Wasserfeldern umkreist werden dürfen.
  Priorität: Mittel
- **Impuls/Ergebnis Listen:**
  - Impuls: Zwei Clients registrieren auf ein Spiel.
  - Ergebnis: Der Server antwortet mit dem uniquePlayerID an die Clients und das Speil kann beginnen.
  - Impuls: Die Clients generieren jeweils ihre Kartenhälfte (mit Beachten das Spiel und Kartenregel) und übertragen es an den Server.
  - Ergebnis: Der Server verifiziert die Kartenhälfte nach der Regel und antwortet mit Okay oder Error je nach Ergebnis der Überprüfung.
- **Benutzergeschichten:**
  - Als KI möchte ich meine Kartenhälfte optimal nach den Regeln generieren, um von Server eine Bestätigung zu bekommen.
- **Benutzerschnittstelle:**
  - Hierbei gibt es keine Benutzerschnittstelle, wo der User nach einer Aktivität die Kartengenerierung beeinflussen kann, aber bei der Aufgabe 4 kann man nachvollziehen wie die einzelnen Komponenten miteinander verbunden sind, um die CLI für den User anzupassen.

[Geben Sie hier, falls möglich und sinnvoll, ein Beispiel für die zugehörige Benutzerschnittstelle an, um darzustellen welchen Einfluss diese Anforderung auf die CLI Benutzeroberfläche hat. Dies soll Ihnen ermöglichen schon vor dem Entwurf festzustellen was und wie Informationen in einer Anwendung dargestellt werden.]

- **Externe Schnittstellen:**
  - *Schnittstelle Netzwerkkommunikation:* Es wird die HTTP Protokoll verwendet, um die XML Daten über die Kartengenerierung von Client zu Server zu schicken und umgekehrt die Antwort vom Server zu bekommen und zu verarbeiten.

  [Beschreiben Sie hier kurz die von dieser Applikation zur Realisierung verwendeten Schnittstellen zu externen Systemen. 1-2 Sätze je Schnittstelle mit Informationen zu den ausgetauschten Informationen, Netzwerkverbindungen, Protokollen, etc. die EntwicklerInnen erkennbar machen sollen wie mit den Schnittstellen interagiert werden muss.]

Dokumentation der Anforderung 3

- **Name:** Betreten auf Wasser vermeiden
- **Beschreibung und Priorität:** Die KI muss die nächste Bewegung nach einem Algorithmus kalkulieren. Der Algorithmus muss nicht nur den kürzesten Weg finden, sondern die Regeln auch berücksichtigen. Eine dieser Kriterien ist, dass der Spielfigur nicht auf Wasserfeld treten darf. Die Bewegung läuft automatisch ab, wenn die Anfrage bei Server ankommt und bestätigt wird. Wenn die KI auf eine Wasserfeld tritt verliert das Spiel.
  Priorität: Hoch
- **Impuls/Ergebnis Listen:**
  - Impuls: Die KIs generieren die Kartenhälften und übermitteln es dem Server.

- Ergebnis: Der Server überprüft die Regeln der Kartengenerierung und bestätigt die Anfrage.
- Impuls: Das Spiel beginnt und die KI, die auf der Reihe ist, muss die nächste Bewegung kalkulieren.
- Ergebnis: Die Bewegung wird kalkuliert und die Anfrage dem Server geschickt.
- Impuls: Der Server bekommt eine Anfrage für Bewegung vom Client und überprüft sie.
- Ergebnis: Nach der Überprüfung bestätigt der Server die Anfrage mit einer Antwort Okay oder Error.
- Impuls: Die andere KI kommt auf die Reihe und führt denselben Prozess durch.
- Ergebnis: Der Prozess läuft so lange, bis der eigene Schatz und die gegnerische Burg einer KI nicht gefunden wird oder das Spiel aus anderen Gründen nicht beendet wird.

- **Benutzergeschichten:**
  - Als KI möchte ich meine Bewegung so kalkulieren, dass ich nicht ins Wasser trete und somit nicht an den Regeln stoße (nicht verliere).
- **Benutzerschnittstelle:**
  - Hierbei gibt es keine Benutzerschnittstelle, wo der User die Bewegung beeinflussen kann, aber bei der Aufgabe 4 kann man nachvollziehen, wie die einzelnen Komponenten miteinander verbunden sind, um die CLI für den User anzupassen.
- **Externe Schnittstellen:**
  - *Schnittstelle Netzwerkkommunikation:* Es wird die HTTP Protokoll verwendet, um die XML Daten über die Bewegung von Client zu Server zu schicken und umgekehrt die Antwort vom Server zu bekommen und verarbeiten zu können.

Aufgabe 3: Definition von Meilensteinen und Aufwandsabschätzung (Projektplanung) (3 Punkte)

Zumindest 3 Meilensteine festlegen und für diese Meilensteine jeweils 3 Arbeitspakete definieren. Danach für alle Arbeitspakete den Aufwand beispielsweise mittels „Planning Poker“ (siehe Einheiten der LV) abschätzen. Achten Sie darauf, dass es sich um sinnvoll benannte Meilensteine/Arbeitspakete handelt und die Vorgaben der Spielidee bzw. des Netzwerkprotokolls abgedeckt werden, das heißt die 3 Meilensteine sind wahrscheinlich nicht ausreichend. Werden Aspekte bei dieser Zeitabschätzung vergessen führt das oft zu scheiternden Projekten, Verlusten sowie unzufriedenen Kunden (Verspätungen) und Entwicklern (Crunchtime).

- **Meilenstein 1: Client: Netzwerkkommunikation**
  - AP 1: Spiel anfragen und registrieren
    - Aufwand: 4 Stunden
  - AP 2: Kartenhälfte mitteilen
    - Aufwand: 4 Stunden
  - AP 3: Bewegung mitteilen
    - Aufwand: 3 Stunden
  - AP 4: Status abfragen
    - Aufwand: 4 Stunden

- **Meilenstein 2: Client: KI**
  - AP 1: Zufällige Kartengenerierung (Datenstruktur: Map)
    - Aufwand: 4 Stunden
  - AP 3: Bewegung planen (Wegfindung mittels Dijkstra Algorithmus)
    - Aufwand 5 Stunden
  - AP 3: Spielaktion (Schatz finden, Bewegung, Burg erobern usw.)
    - Aufwand 5 Stunden
  - AP 4:
    - Aufwand 6 Stunden

- **Meilenstein 3: Client: Benutzeroberfläche**
  - AP1: Karte anzeigen
    - Aufwand: 3 Stunden
  - AP 2: MVC implementieren (mit empfohlenen Property Changed Handling)
    - Aufwand: 3 Stunden
  - AP 3: Spielstatus anzeigen
    - Aufwand: 4 Stunden
  - AP 4: Spielablauf anzeigen
    - Aufwand: 4 Stunden

- **Meilenstein 4: Client: Qualitätssicherung während der Implementierung**
  - AP1: Logging
    - Aufwand: 3 Stunden
  - AP 2: Fehlerbehandlung/Exceptions

    - Aufwand: 3 Stunden
  - AP 3: Unit Tests
    - Aufwand: 4 Stunden
  - AP 4: Dokumentation
    - Aufwand: 4 Stunden

- **Meilenstein 5: Server: Netzwerkkommunikation**
  - Arbeitspaket 1: Spielerstellung mitteilen
    - Aufwand: 3 Stunden
  - Arbeitspaket 2: Spielern zum Spiel - beantworten
    - Aufwand: 3 Stunden
  - Arbeitspaket 3: Spielstatus mitteilen
    - Aufwand: 4 Stunden
  - Arbeitspaket 4: Kartengenerierung und Bewegung beantworten
    - Aufwand: 4 Stunden

- **Meilenstein 6: Server: Logik**
  - Arbeitspaket 1: Spiel erstellen
    - Aufwand: 4 Stunden
  - Arbeitspaket 2: Spielern zu Spiel hinzufügen
    - Aufwand: 4 Stunden
  - Arbeitspaket 3: Spielstatus mitteilen
    - Aufwand: 4 Stunden
  - Arbeitspaket 4: Karten und Bewegung überprüfen
    - Aufwand: 4 Stunden

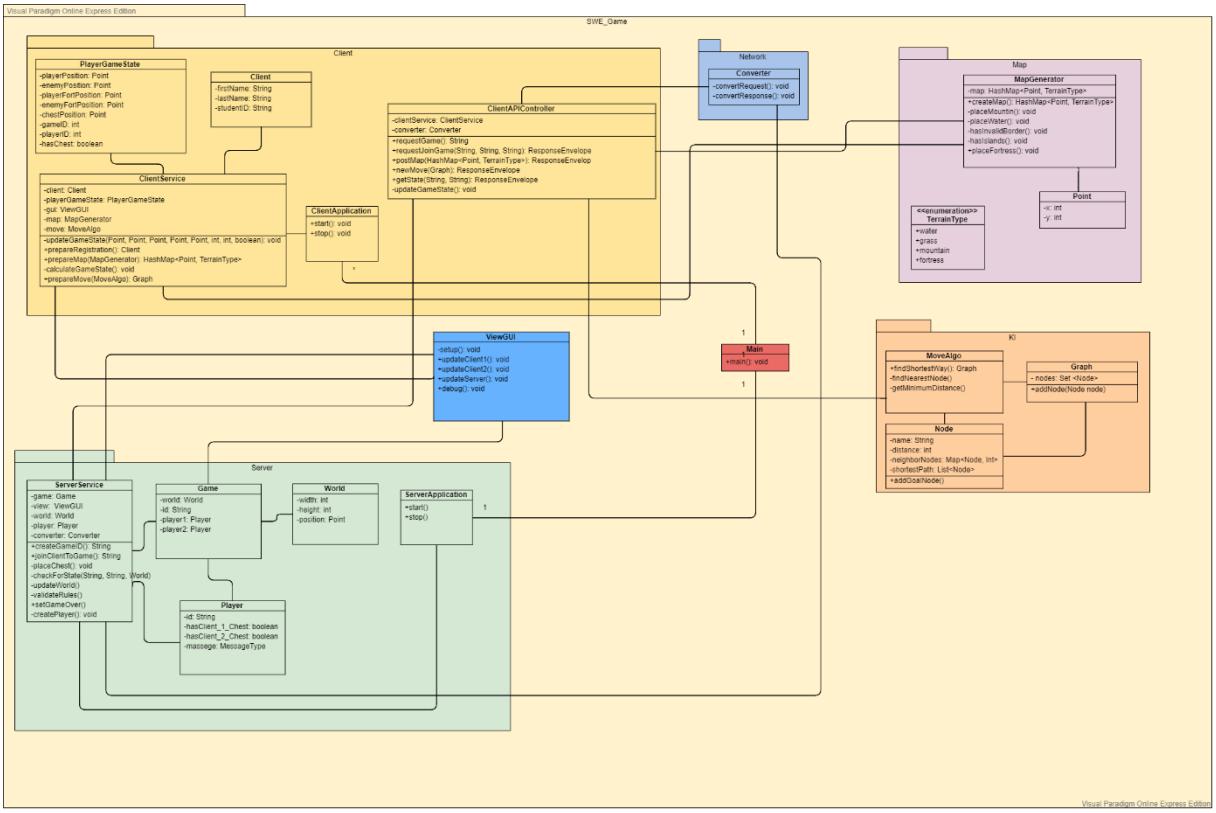
Aufgabe 4: Architektur entwerfen, modellieren und validieren (7 Punkte)

Modellieren Sie händisch alle notwendigen Packages, Klassen und deren Methoden (samt Beziehungen) als UML Klassendiagramm. 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 verdeutlich welche Erwartungen an das Klassendiagramm beziehungsweise die dazugehörigen Sequenzdiagramme gestellt werden (Darstellung, Inhalte, Detailgrad, etc.) und wie diese zusammenhängen.

[Alle drei Diagramme mit einer kurzen Beschreibung hier einfügen. Sollten die Diagramme zu viel Platz benötigen können Sie diese auch als separate SVG Dateien im Dokumentationsordner (siehe GitLab Repo) einfügen.]



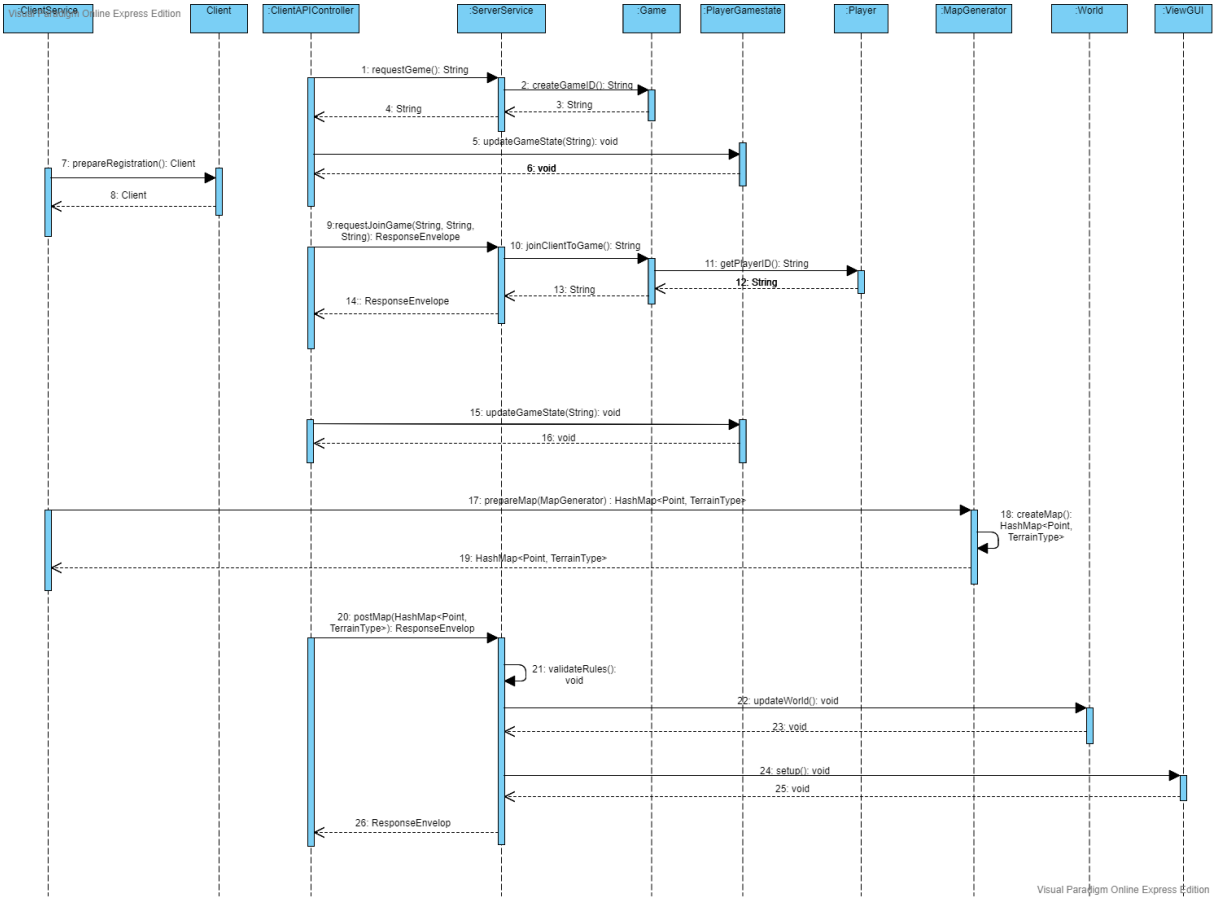
Das Spiel startet bei dem Main Klasse (main() Methode) wo ein ServerApplication bzw. ClientApplication gestartet wird. Der Server fängt an auf Client zu warten und der Client sollte dann ein Spiel requesten.

Die größeren Einheiten als packages sind Client, Server, Network, Map und KI. Main und ViewGUI würde ich im root Folder lassen.

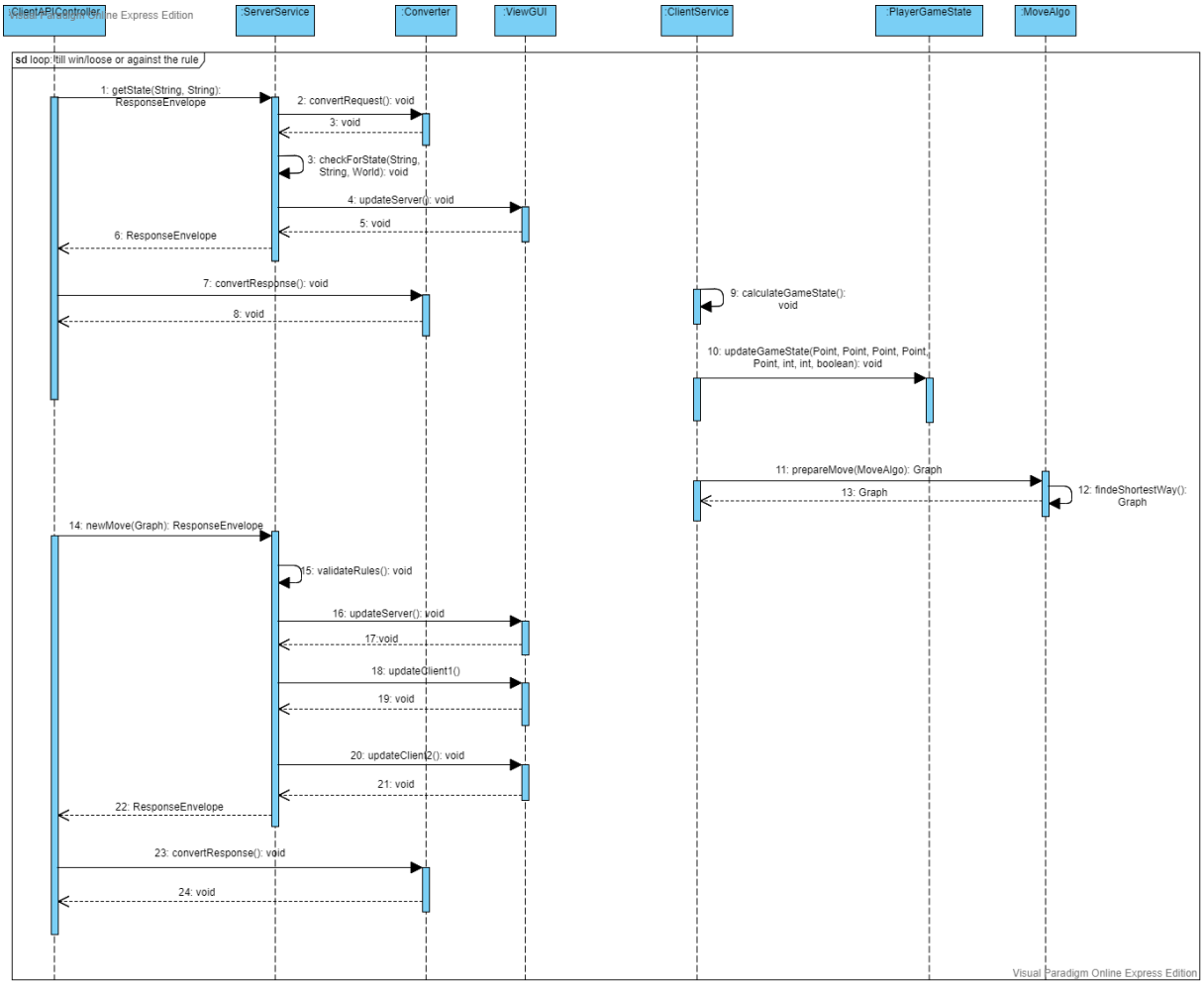
MVC pattern

Nummer	Rolle	Klasse	Beschreibung
1	Model	PlayerGameState	PlayerGameState ist ein Model, der grundlegende Informationen über den Spielstatus beider Spieler hat.
		Game	Game hat die Informationen über das aktuelle Spiel. Game beinhaltet World.
		World	World hat die Informationen über die ganze Karte
2	View	ViewGUI	Visualisiert die Modelle und die Änderungen aus der Client- und Serverseite
3	Controller	ClientService bzw. ClientAPIController	ClientService und ClientAPIController haben die Logik aus der Clientseite und beide sind für alle Änderungen aus der Clientseite verantwortlich.
		ServerService	ServerService hat die Logik aus der

			Serverseite und ist für die Veränderungen von Game und World verantwortlich.



Die Kommunikation läuft hauptsächlich zwischen den ClientAPIController und ServerService ab, da die für die eigentliche Logik verantwortlich sind. Sie sorgen auch dafür, dass die Models upgedatet werden.



Hier sind die restlichen möglichen Schritte zu sehen. Der Client kann noch Status abfragen und verarbeiten bzw. eine Bewegung kalkulieren und an Server schicken. Der Server kann den Status an die Client mitteilen und die von der Client bekommene Bewegung überprüfen und den View anpassen und an Client alles als Antwort zurückschicken.