Software Engineering 1
Abgabedokument
Teilaufgabe 1
(Anforderungsanalyse und Planungsphase)

Persönliche Daten, bitte vollständig ausfüllen:

Nachname, Vorname:	Mazurov Vladislav
Matrikelnummer:	11710356
E-Mail-Adresse:	a11710356@unet.univie.ac.at
Datum:	02.04.2022

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:** Spielkarte – Beide Spieler müssen die Karte auf ihrem Client sehen.
- **Bezugsquelle:** [Spielidee, *„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.“*]

Anforderung 2

- **Beschreibung:** Selbstablaufendes Zwei-Spieler-Spiel – Das Spiel muss selbst Laufen ohne Angaben von Spieler.
- **Bezugsquelle:** [Spielidee, *„Hierbei verfolgen die menschlichen Spieler das eigentliche Spielgeschehen ähnlich eines Livestreams nur passiv, es sollten daher keinerlei Userinteraktionen erforderlich sein.“*]

Anforderung 3

- **Beschreibung:** Turn-Based Spiel – Das Spiel muss rundenbasiert ablaufen.
- **Bezugsquelle:** [Spielidee, *„Die KI, die diese schneller erfüllt, gewinnt das Spiel; welches rundenbasiert abläuft (daher setzt jede KI abwechselnd eine Spielaktion).“*]

Typ der Anforderung: nicht funktional

Anforderung 4

- **Beschreibung:** Spielaktion Zeitbegrenzung – KI soll nicht mehr als 3 Sekunden für die Spielaktion brauchen.
- **Bezugsquelle:** [Spielidee, „*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:** Abfrage des Spielstatus – Der Client kann nur nach der Registrierung Spielstatus abfragen.
- **Bezugsquelle:** [Netzwerkprotokoll, „*Die Abfrage eines Spielstatus durch den Client kann erst nach der Registrierung eines Clients beginnen da hierzu die SpielID aber auch die SpielerID benötigt werden.“*]

Anforderung 6

- **Beschreibung:** Vermeidung von Server Überlastung – Es soll zwischen 2 Abfragen zum Spielstatus von gleichem Client mindestens 0.4 Sekunden vergehen.
- **Bezugsquelle:** [Netzwerkprotokoll, „*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.*“]

Typ der Anforderung: Designbedingung

Anforderung 7

- **Beschreibung:** Nachrichtenaustauschs Protokoll – Client und Server muss HTTP Protokoll sowie die zugehörigen GET und POST Methoden verwenden.
- **Bezugsquelle:** [Netzwerkprotokoll, *„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.“*]

Anforderung 8

- **Beschreibung:** Client und Server Nachrichten – Client und Server soll XML-Nachrichten unterstützen und mit ihnen austauschen.
- **Bezugsquelle:** [Netzwerkprotokoll, „*Der Server antwortet mit einer XML Nachricht, die eine eindeutige SpielID beinhaltet.“, „Der Client sendet einen HTTP Post Request mit einer XML Nachricht im Body an folgenden Endpunkt“*]

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:** Spielaktion soll gemacht werden
- **Beschreibung und Priorität:** Jede Runde KI muss eine Aktion machen. Es gibt keine Möglichkeit, um eine Runde überzuspringen. Andere KI soll auf Aktion gegnerisch KI warten.
  Priorität: hoch.
- **Relevante Anforderungen:**
  - Turn-Based Spiel – Das Spiel muss rundenbasiert ablaufen.
  - Spielaktion Zeitbegrenzung – KI soll nicht mehr als 3 Sekunden für die Spielaktion brauchen.
  - Abfrage des Spielstatus – Der Client kann nur nach der Registrierung Spielstatus abfragen.
  - Nachrichtenaustauschs Protokoll – Client und Server muss HTTP Protokoll sowie die zugehörigen GET und POST Methoden verwenden.
  - Client und Server Nachrichten – Client und Server soll XML-Nachrichten unterstützen und mit ihnen austauschen.
  - Vermeidung von Server Überlastung – Es soll zwischen 2 Abfragen zum Spielstatus von gleichem Client mindestens 0.4 Sekunden vergehen.
- **Relevante Business Rules**:
  - Jede KI macht nur eine Aktion pro Runde.
  - KI kann nicht auf das Setzen einer Aktion verzichten.
  - Andere KI muss auf gegnerische KI Aktion warten.
  - Jede Aktion von KI soll nicht mehr als 3 Sekunden dauern.
  - Das Spiel soll nicht aus mehr als 200 Aktionen zusammen bestehen.
  - Wenn oben genannte Regeln nicht erfüllt werden, dann verliert diese KI das Spiel.
  - Es soll zwischen 2 Abfragen zum Spielstatus von gleichem Client mindestens 0.4 Sekunden vergehen.
  - Client und Server übermitteln das Spielstatus mittels XML Nachrichten und http Protokoll.
- **Impuls/Ergebnis - Typisches Szenario:**
  ***Vorbedingungen*:**
  - Client hat schon registriert und kennt SpielID und SpielerID.
  - Spiel ist noch nicht beendet.

  ***Hauptsächlicher* Ablauf**:
  - Impuls: Client sendet HTTP GET Request und fragt Spielstatus ab.
  - Ergebnis: Server antwortet mittels XML Nachricht und beim Data Bereich im „state“ steht „MustAct“ und „gameStateID“ hat seit letzten Abfrage geändert.
  - Impuls: Client bekommt diese Nachricht und übertragt ihre Bewegung an Server.
  - Ergebnis: Server antwortet mit „responseEnvelope“ und „state“ ist „Okay“.

  Nachbedingungen:
  - Server bekommt Nachrichten von Client über die Bewegung.
  - Client jetzt sendet ständig Abfragen an Server über die Spielstatus.

- **Impuls/Ergebnis - Alternativszenario:**
  Vorbedingungen:
  - o Client hat schon registriert und kennt SpielID und SpielerID.
  - o Spiel ist noch nicht beendet.

  ***Hauptsächlicher* Ablauf**:
  - o Impuls: Client sendet HTTP GET Request und fragt Spielstatus ab.
  - o Ergebnis: Server antwortet mittels XML Nachricht und beim Data Bereich im „state“ steht „MustWait“.
  - o Impuls: Client wartet mindestens 0.4 Sekunden und sendet noch eine Abfrage an Server.
  - o Ergebnis: Server antwortet mittels XML Nachricht und beim Data Bereich im „state“ steht „MustAct“ und „gameStateID“ hat seit letzten Abfrage geändert.
  - o Impuls: Client bekommt diese Nachricht und übertragt ihre Bewegung an Server.
  - o Ergebnis: Server antwortet mit „responseEnvelope“ und „state“ ist „Okay“.

  Nachbedingungen:
  - o Server bekommt Nachrichten von Client über die Bewegung.
  - o Client jetzt sendet ständig Abfragen an Server über die Spielstatus.
- **Impuls/Ergebnis - Fehlerfall:**
  Vorbedingungen:
  - o Client hat schon registriert und kennt SpielID und SpielerID.
  - o Spiel ist noch nicht beendet.

  ***Hauptsächlicher* Ablauf**:
  - o Impuls: Client sendet HTTP GET Request und fragt Spielstatus ab.
  - o Ergebnis: Server antwortet mittels XML Nachricht und beim Data Bereich im „state“ steht „MustAct“ und „gameStateID“ hat seit letzten Abfrage geändert.
  - o Impuls: Client bekommt diese Nachricht und denkt an ihre nächste Bewegung.
  - o Ergebnis: Server wartet auf Antwort von Client.
  - o Impuls: Client denkt an ihre nächste Bewegung mehr als 3 Sekunden.
  - o Ergebnis: Server sendet Client eine Nachricht mit state „Lost“.

  Nachbedingungen:
  - o Client verliert das Spiel.
- **Benutzergeschichten:**
  - o Als Client möchte ich GET Request zu senden, um meine Spielstatus zu wissen.
  - o Als Server möchte ich GET Request von Client bekommen, um eine Antwort mit Spielstatus zu senden.
  - o Als KI möchte ich nächste Schritt in weniger als 3 Sekunden zu berechnen, um Client nicht zu verlieren.
- **Benutzerschnittstelle:**
  User soll immer ihre Status sehen. Deswegen Client muss Spielstatus zeigen, ob Client warten, spielen muss oder hat verloren, gewonnen. Client kann auch zeigen, aus welchen Gründen hat er gewonnen oder verloren (z.B., weil KI von Client mehr als 3 Sekunden braucht, um nächste Schritt zu berechnen.)
- **Externe Schnittstellen:**
  - o Schnittstelle Client: Client und Server verwenden http Protokoll sowie GET und POST Methoden, um Nachrichten auszutauschen. Client fragt nach Spielstatus von Server mittel HTTP GET Request ab. Für diese Abfrage braucht Client beide „SpielID“ und „SpielerID“. Dann bekommt Client XML Nachricht von Server.
  - o Schnittstelle Server: Server bekommt GET Request von Client, dann antwortet mit „responseEnvelope“ um Client über Fehler hinzuweisen. Noch dazu gibt Server im data Element Information über Player wie SpielerID, Vorname, Nachname,

u:account und Spielstatus. Es gibt auch Information über die Karte in diesem Nachricht.

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.

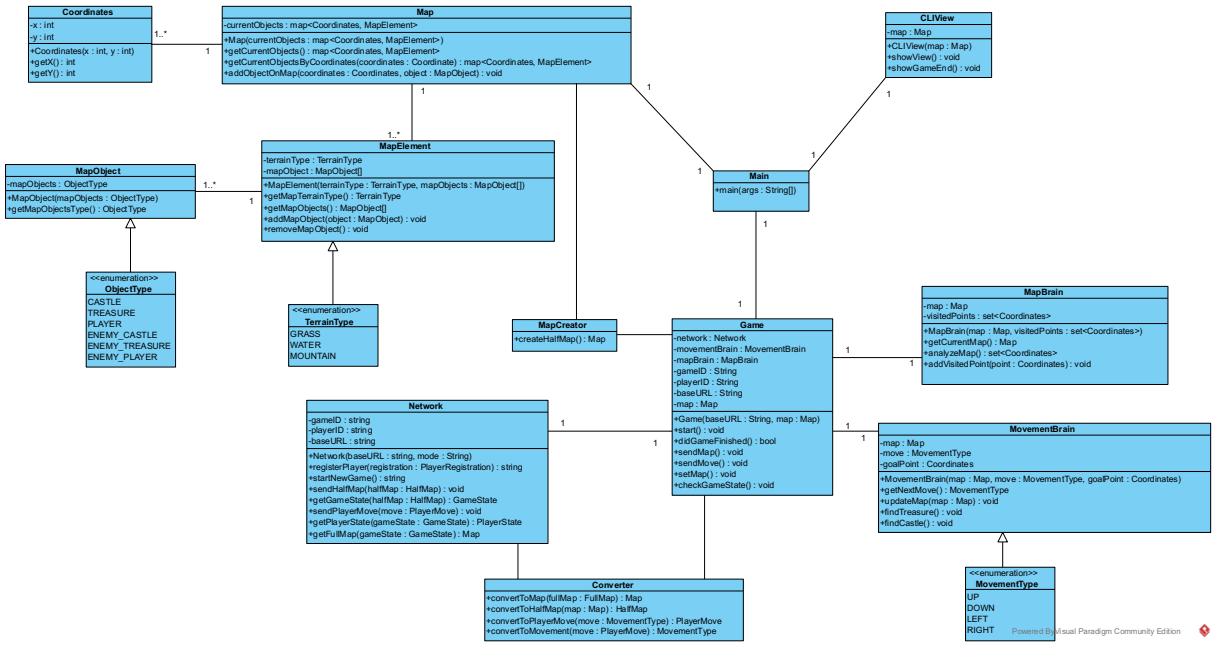

Fig 1. Client - Klassendiagramm

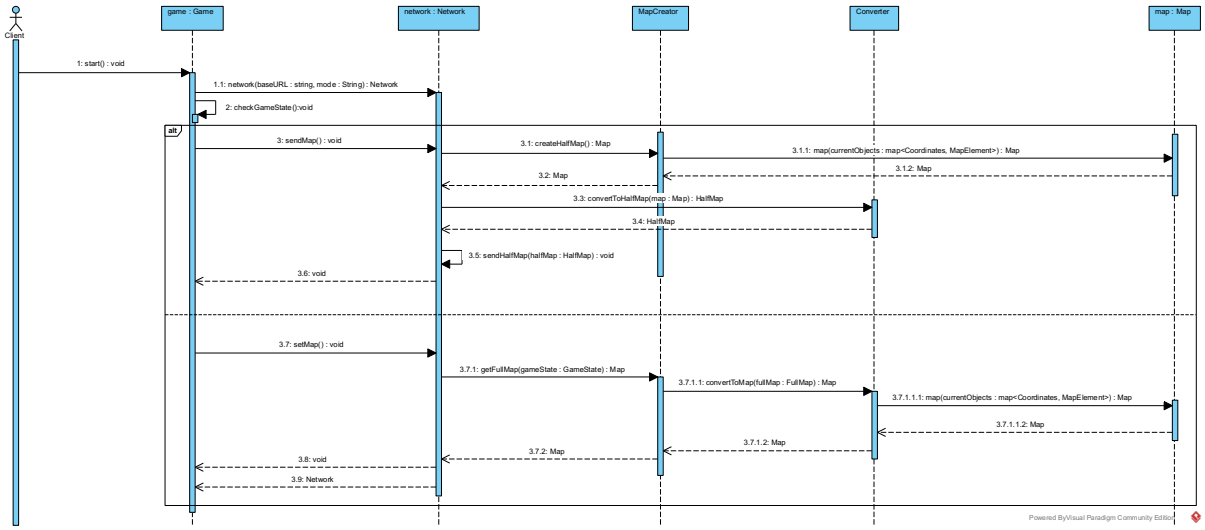

Fig 2. Client - Sequenzdiagramm

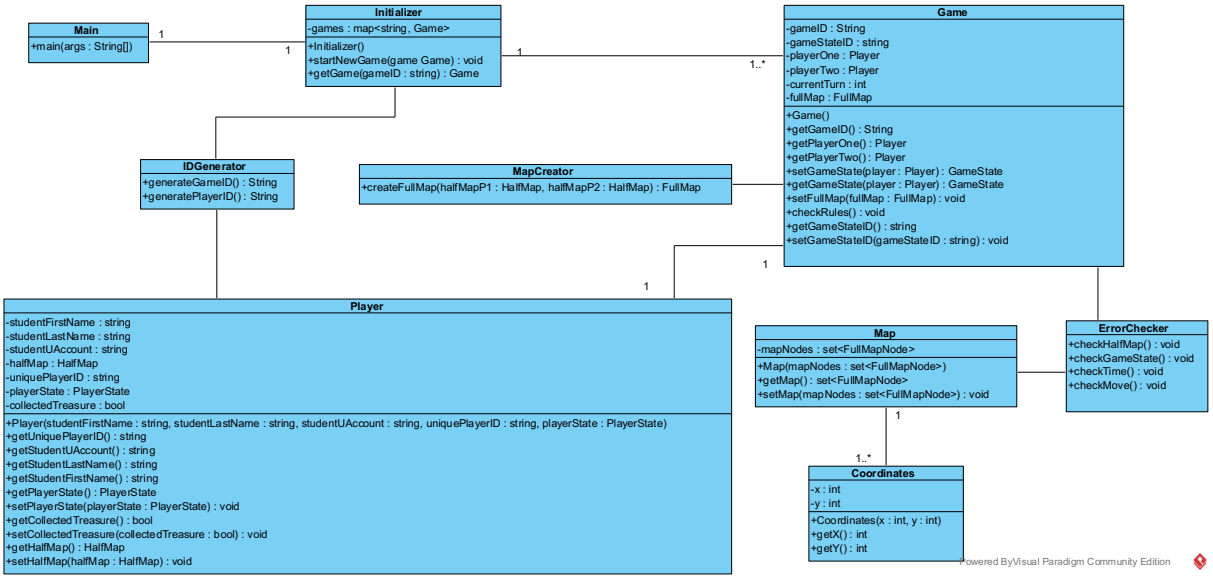

Fig 3. Server - Klassendiagramm

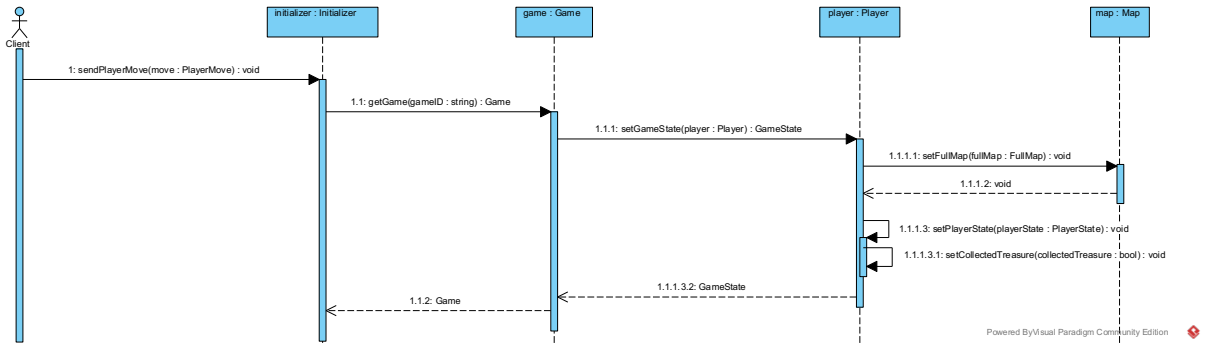

Fig 4. Server - Sequenzdiagramm