Software Engineering 1
Abgabedokument
Teilaufgabe 1
(Anforderungsanalyse und Planungsphase)

Persönliche Daten, bitte vollständig ausfüllen:

Nachname, Vorname:	Žák, Michal Robert
Matrikelnummer:	11922222
E-Mail-Adresse:	a11922222@unet.univie.ac.at
Datum:	27.03.2021

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:** Schatz verstecken - Der Server versteckt einen Schatz auf jeder Hälfte der Karte. Dieser wird dann von Clients gesucht.
- **Bezugsquelle:** Spielidee: „*Hierzu wird vom Server jeweils ein Schatz auf jeder Hälfte der Karte versteckt.*“

Anforderung 2

- **Beschreibung:** Rundenbasierter Ablauf – Jeder Client kann immer nur eine Aktion setzen und muss danach warten bis der gegnerische Client eine Aktion setzt, damit das Spiel rundenbasiert abläuft.
- **Bezugsquelle:** Spielidee: „*Die Spielaktionen selbst werden rundenbasiert durchgeführt. Daher kann jede KI immer nur eine Aktion setzen (z.B. einen Bewegungsbefehl oder die Übertragung einer Kartenhälfte) und muss danach warten, bis die andere KI ihre Aktion gesetzt hat.*“

Anforderung 3

- **Beschreibung:** Client; Erstellung eines Spieles – Die Clients können ein Spiel am Server erstellen, damit sie gegeneinander antreten können.
- **Bezugsquelle:** Netzwerkprotokoll: „*Bevor zwei Clients gegeneinander antreten können muss einer von diesen ein neues Spiel am Server erstellt haben*.“

Typ der Anforderung: nicht funktional

Anforderung 4

- **Beschreibung:** Kartenhälfte Feld Bedingungen – Auf jeder Kartenhälfte muss es mindestens 3 Bergfelder, 15 Wiesenfelder und 4 Wasserfelder geben.
- **Bezugsquelle:** „ [...] jede Kartenhälfte muss mindestens 3 Bergfelder, 15 Wiesenfelder und 4 Wasserfelder beinhalten.“

Anforderung 5

- **Beschreibung:** Spielaktionen Limit – Ein Spiel (das vom Client gespielt und vom Server verwaltet wird) besteht maximal aus 200 Spielaktionen, damit es spannend bleibt.
- **Bezugsquelle:** Spielidee: „*Um die Spiele für die Zuschauer spannend zu gestalten, wurde festgelegt, dass ein Spiel insgesamt nicht länger als 200 Spielaktionen dauern darf*“

Anforderung 6

- **Beschreibung:** Spielaktion Dauer Limit – Der Client hat für jede Spielaktion nicht mehr als 3 Sekunden Zeit, damit das Spiel spannend bleibt.
- **Bezugsquelle:** Spielidee: „*Um die Spiele für die Zuschauer spannend zu gestalten,* […] und *eine KI für jede dieser rundenbasierten Spielaktion nicht mehr als 3 Sekunden*“

Anforderung 7

- **Beschreibung:** Server; Erstellung eines Spiel – Wenn der Client ein neues Spiel auf dem Server erstellen will, antwortet der Server mit einer XML Nachricht, welche eine eindeutige `SpielID` beinhalten.
- **Bezugsquelle:** Netzwekprotokoll: „*Der Server antwortet mit einer XML Nachricht, die eine eindeutige `SpielID` beinhaltet.*“

Typ der Anforderung: Designbedingung

Anforderung 8

- **Beschreibung:** Nachrichten Format – Die Nachrichten die der Client und der Server austauschen werden im XML Format codiert sein da der Nachrichtenaustausch mittels einer Restschnittstelle abläuft.
- **Bezugsquelle:** Netzwerkprotokoll: „*Die ausgetauschten Daten bzw. Nachrichten werden im XML Format definiert bzw. erwartet.*“

Aufgabe 2: Anforderungsdokumentation (2 Punkte)

Dokumentation von *einer* in Aufgabe 1 identifizierten Anforderungen 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:** Rundenbasierter Ablauf
- **Beschreibung und Priorität:** Jeder Client kann immer nur eine Aktion setzen und muss danach warten bis der gegnerische Client eine Aktion setzt, damit das Spiel rundenbasiert abläuft. Der Server muss dies erzwingen und die Clients müssen sich dementsprechend verhalten.
  Priorität: Hoch
- **Relevante Anforderungen:**
  - *Anforderung 5 – Spielaktionen Limit:* Es gibt eine obere Schranke wie viele Runden es geben kann.
  - *Anforderung 6 – Spielaktion Dauer Limit:* Es gibt eine obere Schranke wie lange eine Runde dauern kann.
  - *Anforderung 8 – Nachrichtenformat:* Um eine Aktion durchzuführen muss der Client den Server eine Nachricht schicken.
- **Relevante Business Rules**:
  - Spielaktionen müssen rundenbasiert ablaufen
  - Pro Spielzug gibt es 3 Sekunden
  - Der Client darf den Status vom Server jede 0.4 Sekunden abfragen
  - Das Spiel darf insgesamt nicht länger als 200 Spielaktionen dauern
  - Der Austausch von Nachrichten erfolgt im XML Format.
  - Der Nachrichtenaustausch erfolgt über eine Restschnittstelle
- **Impuls/Ergebnis - Typisches Szenario:**
  ***Vorbedingungen*:**
  - Ein Client hat ein neues Spiel erstellt
  - Zwei Clients sind diesem Spiel beigetreten
  - Beide Clients haben ihre generierten Karten den Server geschickt

  ***Hauptsächlicher* Ablauf**:
  - Der Client, wessen Zug gerade ist muss eine Aktion an den Server stellen.
  - Der Client, wessen Zug gerade nicht ist muss warten und keine Aktion an den Server schicken. Er kann aber Abfragen ob es Jetzt sein Zug ist.

  Nachbedingungen:
  - Es wurde eine Runde gespielt.
  - Der Client wessen Zug war muss jetzt warten.
  - Der Client welcher gewartet hat ist jetzt am Zug
- **Impuls/Ergebnis - Alternativszenario:**
  Vorbedingungen:
  - Ein Client hat ein neues Spiel erstellt
  - Zwei Clients sind diesem Spiel beigetreten
  - Beide Clients haben ihre generierten Karten den Server geschickt
  - Ein Client hat den Schatz gefunden
  - Der Client, welcher den Schatz gefunden hat, ist am Zug.

  - Der Client, welcher den Schatz gefunden hat, ist ein Zug entfernt von der gegnerischen Burg

***Hauptsächlicher* Ablauf**:

  - Der Client welcher den Schatz gefunden hat, schickt dem Server eine Aktion die bewirkt das er sich in die gegnerische Burg bewegt
  - Der Client, welcher den Schatz nicht gefunden hat, wartet auf seine Runde

Nachbedingungen:

  - Der Client, welcher den Schatz gefunden hat, gewinnt.
  - Der Client, welcher den Schatz nicht gefunden hat, verliert.

- **Impuls/Ergebnis – Fehlerfall:**

Vorbedingungen:

  - Ein Client hat ein neues Spiel erstellt
  - Zwei Clients sind diesem Spiel beigetreten
  - Beide Clients haben ihre generierten Karten den Server geschickt

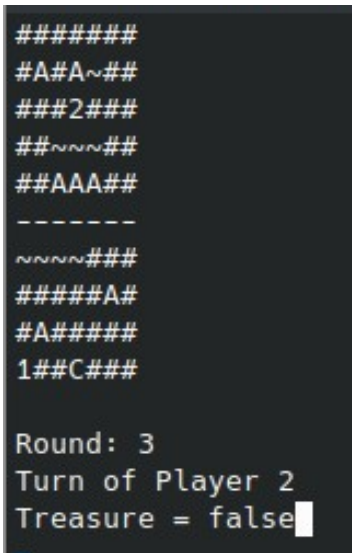
***Hauptsächlicher* Ablauf**:

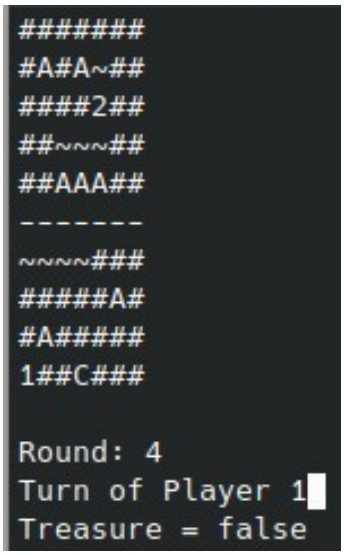
  - Der Client, welcher am Zug ist, sendet keine Spielaktion für länger als 3 Sekunden
  - Der Client, welcher nicht am Zug ist, wartet.

Nachbedingungen:

  - Der Client, welcher am Zug ist, verliert
  - Der Client, welcher nicht am Zug ist, gewinnt

- **Benutzergeschichten:** [Zugehörige User Stories, welche die Anforderung genauer beschreiben mit folgendem Aufbau: Als <Rolle> möchte ich <Ziel/Wunsch> um s folgendes <Nutzen> zu erreichen. Geeignete Rollen wären: (Menschlicher) Anwender, Client, KI oder Server.]
  - Als Client möchte ich in einem fairen Spiel meine Implementierung gegen einen Anderen Client messen.
  - Als Server möchte ich garantieren das die Clients die gegeneinander Spielen den Regeln folgen.
  - Als Server möchte ich Aktionen vom Client entgegen nehmen und entsprechend auf sie reagieren.
  - Als Zuschauer möchte ich ein Spannendes Duell von zwei Clients anschauen können. Dazu dürfen die Aktionen aber nicht zu schnell nacheinander sein, aber auch nicht zu lange von einander entfernt.
- **Benutzerschnittstelle:** Im CLI wird der jetzige stand der Runde dargestellt. Es wird unter anderem ausgegeben wo sich die Spieler befinden, ob der Schatz schon gefunden wurde und wessen Runde gerade ist.





Mögliche Repräsentation des CLI Interface das jede runde ausgegeben wird.
# = Grass
A = Berg
~ = Wasser
1,2 = Spieler
C = Burg
- = Trennung zwischen Karten Hälften

Hier wichtig ist der Round zaehler, welcher anzeigt in welcher Runde man sich gerade befindet.
Die '2' (also der Client 2) hat sich zwischen Runde 3 und 4 um eins nach links bewegt.

- **Externe Schnittstellen:**
  - *Schnittstelle Server:* Der Client muss Informationen an den Server schicken, und Informationen vom Server empfangen können. Dazu wird eine Restschnittstelle benutzt, welche es ermöglicht Daten im XML Format auszutauschen.

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

Client:
[Relevante Dateien: client_class.svg, client_seq.svg]
Das Klassen Diagramm des Clients habe ich in 6 Haupt Packages aufgeteilt.
Das Execution Package ist für den tatsächlichen Ablauf zuständlich und beinhaltet die main Methode.
Das UI Package beinhaltet alle views (UI's) die im Projekt definiert werden.
Das Networking Package übernimmt die Kommunikation mit dem Server und stellt auch eine Translator Klasse zur Verfügung, die die Network Datenstrukturen auf die vom Client übersetzt.
Das HalfMapGeneration Package enthält die Klassen zuständlich für die Generation von HalfMaps. Es bietet ein Interface an das implementiert werden kann um verschiedene HalfMap-generation Algorithmen zu unterstützten.
Das MoveGeneration Package ist für die Lieferung von Zügen zuständlich. Es besteht aus einigen abstrakten Klassen die durch verschiedene Implementierungen ausgenutzt werden können.
Das Map Package beinhaltet alle Daten über die Karte die der Client braucht. Es bietet eine Factory Class, da das erstellen von MapData kompliziert werden kann. Genau so bietet es eine Updater und Accesser Class, weil diese Funktionalität nicht von jedem gebraucht wird.

Server:
[Relevante Dateien: server_class.svg, server_seq.svg]
Das Server Klassen Diagramm besteht aus 2 Haupt Packeges.
Das ServerNetworking Package dient zur Kommunikation mit der Arsenwelt. Die Methoden der EndPoint Class werden aufgerufen wenn eine Nachricht von einen Client ankommt.
Das Game Package enthält die Spiel Daten und Spiel Logik. Der GamesManager speichert alle laufenden spiele. Die Game Class beinhaltet alle relevante Details rund um ein Spiel. Sollte eine Spielregel gebrochen werden wird eine Exception geworfen die dann den Callstack runter kaskadiert.