Software Engineering 1
Abgabedokument
Teilaufgabe 1
(Anforderungsanalyse und Planungsphase)

Persönliche Daten, bitte vollständig ausfüllen:

Nachname, Vorname:	Srdanovic Nemanja
Matrikelnummer:	01576891
E-Mail-Adresse:	a01576891@unet.univie.ac.at
Datum:	01.11.2020

Aufgabe 1: Anforderungsanalyse (2 Punkte)

Typ der Anforderung: funktional

Nr.	Anforderung	Dokumentteil (Zitat)	Dokument
1.	Beim Start des Spiels muss der Client eine Hälfte der Karte (4x8 Felder) erstellen.	„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).“	Spielidee
2.	Beim generieren der Karte muss der Client die Karte zufällig generieren und nicht statisch vorgeben.	„..Die erstellten Karten sollten dabei zufällig generiert und nicht statisch vorgegeben werden“	Spielidee
3.	Beim generieren der Karte muss der Client sie in Felder aufteilen.	„Die Karten sind in Felder aufgeteilt.. „	Spielidee
4.	Beim generieren der Karte muss der Client jedes Feld als eins von drei Terrainarten (Wasser, Wiese, Berg) repräsentieren.	„Jedes Feld repräsentiert genau eine von drei möglichen Terrainarten, nämlich Wasser, Wiese und Berg.“	Spielidee
5.	Beim generieren der Karte muss der Client beachten, dass eine minimale Anzahl an Terraintyp Feldern vorkommen (3xBergfelder ; 15xWasserfelder; 4xWasserfelder).	„Außerdem muss jeder Terraintyp vorkommen, das heißt jede Kartenhälfte muss mindestens 3 Bergfelder, 15 Wiesenfelder und 4 Wasserfelder beinhalten.“	Spielidee
6.	Beim generieren der Karte muss der Client beachten, dass keine Inseln generiert werden dürfen bzw. Felder nicht vollständig von Wasserfeldern umschlossen werden dürfen.	Weiters dürfen keine Inseln generiert werden, daher dürfen Berge oder Wiesen niemals vollständig von Wasser oder Kartengrenzen (oder einer Kombination aus beidem) umschlossen werden“	Spielidee
7.	Beim generieren der Karte muss der Client beachten, dass maximal 3 Wasserfelder an den langen und 1 Wasserfeld an den kurzen Seiten generiert werden können.	„Abschließend gilt, dass jede Kartengrenze einer zu generierenden Kartenhälfte nur zu weniger als der Hälfte aus Wasserfeldern“.	Spielidee
8.	Beim generieren der Karte muss der Client beachten, dass zu jedem Feld einer Kartenhälfte (das nicht aus Wasser besteht) von jedem anderen Feld der Kartenhälfte (welches nicht aus Wasser besteht) ein Weg aus einer Reihe von validen Bewegungsbefehlen generiert werden können.	Es gilt außerdem, dass zu jedem Feld einer Kartenhälfte (das nicht aus Wasser besteht) von jedem anderen Feld der Kartenhälfte (welches nicht aus Wasser besteht) ein Weg aus einer Reihe von validen Bewegungsbefehlen generiert werden können muss, um dieses Feld zu betreten.“	Spielidee
9.	Beim erstellen der Kartenhälfte muss der Client, auf den von ihm erstellten Teil der Karte seine Burg platzieren.	„Die Spielfigur der KI startet anschließend jeweils an einer von ihr selbst definierten Position (ihrer Burg) auf dem von ihr erstellten Teil der Karte.“	Spielidee
10.	Wenn er die Burg platzieren möchte, muss der Client das auf einem Wiesenfeld machen.	„Allerdings können die Burg und der Schatz nur auf Wiesenfeldern und nicht auf demselben Feld platziert werden“	Spielidee

Nr.	Anforderung	Dokumentteil (Zitat)	Dokument
11.	Wenn die Kartenhälften beider Client ankommen, kann der Server sie entweder an der kurzen (4x16) oder langen Seite (8x8) kombinieren.	„Zu beachten ist hierbei, dass die Kartenhälften entweder an den kurzen (Kartenmaße 4 x 16) oder den langen Seiten (Kartenmaße 8 x 8) kombiniert werden – dies kann der Client nicht beeinflussen, sondern wird vom Server durchgeführt.“	Spielidee
12.	Beim zusammenfügen der beiden Kartenhälften muss der Server überprüfen, dass die gesamte Karte eine fixe Ausdehnung von 64 Felder hat.	„Die gesamte Karte hat immer eine fixe Ausdehnung von 64 Feldern.“	Spielidee
13.	Wenn er eine Kartenhälfte erhält, muss der Server den Schätz auf dieser Kartenhälfte verstecken.	„Hierzu wird vom Server jeweils ein Schatz auf jeder Hälfte der Karte versteckt.“	Spielidee
14.	Wenn er den Schatz platziert, muss der Server ihm jeweils dem Client zuordnen, der auf dieser Kartenhälfte startet.	„Ein Schatz ist hierbei jeweils der KI zugeordnet, die auf dieser Kartenhälfte startet.“	Spielidee
15.	Wenn er den Schatz platzieren möchte, muss der Server ihm auf einem Wiesenfeld platzieren, auf welchem sich nicht die Burg des Spielers (wem der Schatz zugeordnet wird) befindet.	„Allerdings können die Burg und der Schatz nur auf Wiesenfeldern und nicht auf demselben Feld platziert werden“	Spielidee
16.	Falls eine der Kartenbedingungen nicht erfüllt ist, muss der Server den Client welcher gegen die Bedingung verstoßen hat, über die Niederlage und den Gegner über den Sieg informieren.	„Sollte eine der beiden von den KIs erstellten Kartenhälften gegen diese Bedingungen verstoßen, verliert die KI, welche die verstoßende Kartenhälfte generiert hat. Die andere KI gewinnt hierdurch automatisch.“	Spielidee
17.	Nach der Positionierung der Burg muss der Client seine Spielfigur auch von dieser Position beim Spielbeginn starten.	„Die Spielfigur der KI startet anschließend jeweils an einer von ihr selbst definierten Position (ihrer Burg) auf dem von ihr erstellten Teil der Karte.“	Spielidee
18.	Nachdem die Spielfigur positioniert ist, muss der Client (KI) die Karte mit der Spielfigur erkunden, um den versteckten Schatz zu finden.	„Anschließend muss sie die Karte erkunden, um einen versteckten Schatz zu finden.“	Spielidee
19.	Wenn er auf der Suche nach dem Schatz ist, muss der Client (KI) immer auf der von ihm erstellten Kartenhälfte suchen.	„Der Schatz einer KI wird immer auf der Kartenhälfte versteckt, die von der jeweiligen KI erstellt wurde.“	Spielidee
20.	Wenn er die Spielfigur über die Karte bewegt, kann der Client (KI) dadurch mit seiner Spielfigur den Schatz oder die Burg entdecken.	„Um den Schatz zu finden, bewegen beide KIs ihre Spielfigur über die Karte und decken jeweils mit ihrer Spielfigur den versteckten eigenen Schatz und die gegnerische Burg auf.“	Spielidee
21.	Wenn der persönliche Schatz aufgedeckt wurde, muss der Client (KI) seine Spielfigur zu diesem bewegen.	„Sobald eine KI "ihren" Schatz gefunden bzw. aufgedeckt hat, muss sie ihre Spielfigur zu diesem bewegen, um ihn aufnehmen zu können.“	Spielidee

Nr.	Anforderung	Dokumentteil (Zitat)	Dokument
22.	Wenn er einen Schatz findet, kann der Client (KI) ihn nur aufnehmen und verwenden, falls es der Schatz ist, der ihm zugeordnet ist.	„Ein Schatz kann nur von der Spielfigur der zugeordneten KI "aufgenommen" und "verwendet" werden.“	Spielidee
23.	Wenn er den Schatz aufgenommen hat, muss der Client (KI) seine Spielfigur auf die Kartenhälfte der gegnerischen KI bewegen und die gegnerische Burg suchen.	„Sobald der Schatz aufgenommen wurde, muss die KI ihre Spielfigur auf den Teil der Karte bewegen, der von der gegnerischen KI erstellt wurde, um dort die gegnerische Burg zu finden bzw. aufzudecken.“	Spielidee
24.	Wenn er die gegnerische Burg gefunden hat, muss der Client (KI) seine Spielfigur auf das Feld mit der Burg bewegen.	„ Sobald dies gelungen ist, muss sie sich zur gegnerischen Burg bewegen, die Bewacher der Burg mit dem gerade gefundenen Schatz bestechen und so die Burg "übernehmen"	Spielidee
25.	Wenn der Client (KI) den Schatz aufgenommen hat und sich auf dem Feld der gegnerischen Burg befindet, muss der Server den Kauf der Burg automatisch durchführen.	„Das Aufnehmen eines Schatzes und der Kauf der Burg erfolgen automatisch, sobald eine Spielfigur sich auf dem passenden Spielfeld der Karte befindet.“	Spielidee
26.	Wenn das Spiel beginnt, kann der Client (KI) die ganze Karte bzw. ihr Terrain sehen.	„Bis auf die Schätze, die gegnerische Burg sowie die gegnerische KI in den ersten 10 Spielzügen eines Spiels ist bei Spielbeginn für beide KIs bereits die ganze Karte sichtbar.“	Spielidee
27.	Wenn sich das Spiel in den ersten 10 Spielzügen befindet, muss der Server eine zufällige Position des Gegners (KI) melden.	„Die ersten 10 Spielzüge eine Spiels hat die vom Server gemeldete Position des Gegners zufällig zu sein“	Spielidee
28.	Wenn der Client (KI) eine Spielaktion durchgeführt hat, muss er warten, bis der Gegner seine durchführt, um eine neue Spielaktion durchzuführen.	„Daher kann jede KI immer nur eine Aktion setzen und muss danach warten, bis die andere KI ihre Aktion gesetzt hat.“	Spielidee
29.	Wenn der Client (KI) an der Reihe ist, eine Spielaktion zu setzten, muss er diese durchführen.	„Eine KI kann hierbei nicht auf das Setzen einer Aktion verzichten, sondern muss z.B. immer eine Bewegungsaktion durchführen. „	Spielidee
30.	Wenn er die Spielfigur zwischen Feldern bewegen möchte, kann der Client (KI) das nur waagerecht und senkrecht machen.	„ Die Karten sind in Felder aufgeteilt, zwischen denen sich die Spielfiguren bewegen können (waagrecht und senkrecht).“	Spielidee
31.	Wenn er mit der Spielfigur ein Feld betreten möchte, sollte der Client (KI) das nur machen, falls es sich um kein Wasserfeld handelt.	„Außerdem darf Wasser unter keinen Umständen betreten werden.“	Spielidee
32.	Falls er mit der Spielfigur ein Wiesenfeld betreten oder verlassen möchte, muss der Client (KI) dafür jeweils eine Bewegungsaktion durchführen.	„Wasser und Wiesen können jeweils mit einer Bewegungsaktion betreten und im Fall von Wiesen auch wieder verlassen werden.“	Spielidee

Nr.	Anforderung	Dokumentteil (Zitat)	Dokument
33.	Wenn sich seine Spielfigur auf einem Wiesenfeld befindet, wird der Client(KI) benachrichtigt, ob sich dort eine Burg oder Schatz befinden bzw. ob der Schatz aufgenommen wurde.	„Sobald eine Wiese betreten wurde, wird aufgedeckt, ob sich auf dieser Wiese eine Burg oder ein Schatz befindet bzw. wird dieser direkt aufgenommen. „	Spielidee
34.	Falls er mit der Spielfigur ein Bergfeld betreten oder verlassen möchte, muss der Client (KI) dafür zwei Bewegungsaktionen durchführen.	„Im Gegensatz zu Wiesen benötigen Berge zwei Bewegungsaktionen, um diese (das Bergfeld) zu betreten und zwei zusätzliche, um den Berg wieder zu verlassen.“	Spielidee
35.	Wenn sich seine Spielfigur auf einem Bergfeld befindet, wird der Client(KI) benachrichtigt, ob sich auf irgendeinem Feld mit einer Entfernung von eins (auch diagonal) rund um das betretene Bergfeld, der Schatz oder die Burg befinden.	„..zusätzlich zum Bergfeld selbst werden alle Felder mit einer Entfernung von eins (auch diagonal) aufgedeckt die sich rund um die Spielfigur bzw. den gerade betretenen Berg befinden..“	Spielidee
36.	Falls er die Spielfigur auf ein anderes Feld bewegen möchte, kann der Client (KI) das nur auf eins der benachbarten Felder machen (überspringen von Feldern ist nicht möglich).	„Eine Spielfigur kann sich nur horizontal und vertikal zu direkt benachbarten Feldern bewegen, das Überspringen von Feldern ist daher nicht möglich.“	Spielidee
37.	Wenn er sich innerhalb einer Bewegungsaktion für ein anderes Feld entscheidet, kann der Client (KI) eine neue Bewegungsrichtung angeben.	„Während den benötigten Bewegungen kann jederzeit eine neue alternative Richtung eingeschlagen werden…“	Spielidee
38.	Während des Spiels muss der Client die Karte und deren bekannte Eigenschaften auf den Rechnern des menschlichen Spielers visualisieren.	„Während des Spiels müssen die Karte und deren bekannte Eigenschaften auf den Rechnern (den Clients) der beiden menschlichen Spieler visualisiert werden..“	Spielidee
39.	Wenn der Client eine inkorrekte Anfrage sendet, muss der Server eine Fehlermeldung retournieren.	„Darüber hinaus wird im Falle einer inkorrekten Anfrage typischerweise eine Fehlermeldung zusätzlich retourniert.“	Netzwerk-protokoll
40.	Wenn eine entsprechende Anfrage an eine entsprechende Adresse (../games) ankommt, muss der Server ein neues Spiel erstellen und eine Antwort mit entsprechender SpielID retournieren.	„.dies wird nicht von den KIs sondern von einem Menschendurchgeführt. Hierzu ist es notwendig auf den unten beschriebenen Endpoint zuzugreifen, indem eine HTTP GET Anfrage an diesen gesandt wird…“	Netzwerk-protokoll
41.	Wenn eine entsprechende Anfrage mit einer korrekten SpielID an eine entsprechende Adresse (../players) ankommt, muss der Server den Client registrieren und eine einsprechende SpielerID retournieren.	„Der Server antwortet auf die Registrierungsnachricht (siehe oben) mit einer Bestätigung über die Durchführung der Registrierung sowie einer entsprechenden eindeutigen SpielerID .“	Netzwerk-protokoll

Nr.	Anforderung	Dokumentteil (Zitat)	Dokument
42.	Wenn eine entsprechende Anfrage mit einer korrekten SpielID und SpielerID an eine entsprechende Adresse (../halfmaps) ankommt, muss der Server die bearbeiten und eine Antwort mit entsprechendem Zustand und ggf. Fehlermeldung retournieren.	" Der Server antwortet generisch mit einem responseEnvelope. Daher im Fehlerfall sind exceptionName und exceptionMessage mit Details zum Fehler befüllt sowie das state Element mitdem Text Error definiert. Sollte kein Fehler vorliegen ist state mit dem Wert Okay definiert."	Netzwerk-protokoll
43.	Wenn eine entsprechende Anfrage mit einer korrekten SpielID und SpielerID an eine entsprechende Adresse (../states/<SpielerID>) ankommt, muss der Server die bearbeiten und eine Antwort mit entsprechendem Zustand und ggf. Fehlermeldung, oder Daten retournieren.	„..entsprechende Fehlerbeschreibung beinhaltet und mit dem state Element verdeutlicht ob ein Fehler aufgetreten ist oder nicht. Das data Element beinhaltet die eigentlichen Nutzdaten…"	Netzwerk-protokoll
44.	Erst wenn im Spielstatus angezeigt wird, dass der Client an der Reihe ist, kann der Client eine Spielbewegung durchführen.	„Sobald ein Client an der Reihe ist (daher im Spielstatus angezeigt wird, dass die nächste Aktion gesendet werden muss) kann der Client eine Spielbewegung übertragen."	Netzwerk-protokoll
45.	Wenn eine entsprechende Anfrage mit einer korrekten SpielID und SpielerID an eine entsprechende Adresse (../moves) ankommt, muss der Server die bearbeiten und eine Antwort mit entsprechendem Zustand und ggf. Fehlermeldung retournieren.	„Der Server antwortet generisch mit einem responseEnvelope. Daher im Fehlerfall sind exceptionName und exceptionMessage mit Details zum Fehler befüllt sowie das state Element mitdem Text Error definiert. Sollte kein Fehler vorliegen ist state mit dem Wert Okay definiert."	Netzwerk-protokoll

Typ der Anforderung: nicht funktional

Nr.	Anforderung	Dokumentteil (Zitat)	Dokument
1.	Wenn der Client an der Reihe ist, eine Spielaktion durchzuführen, muss er diese innerhalb von 3 Sekunden machen.	„..KI für jede dieser rundenbasierten Spielaktion nicht mehr als 3 Sekunden Bedenkzeit zur Berechnung erhält.“	Spielidee
2.	Wenn das Spiel beginnt, kann der Client in den ersten 10 Spielzügen den Schatz, gegnerische Burg sowie die gegnerische KI nicht sehen.	„Bis auf die Schätze, die gegnerische Burg sowie die gegnerische KI in den ersten 10 Spielzügen eines Spiels ist bei Spielbeginn für beide KIs bereits die ganze Karte sichtbar.“	Spielidee
3.	Wenn das Spiel länger als 200 Spielaktionen dauern sollte, muss der Server das Spiel beenden.	„Um die Spiele für die Zuschauer spannend zu gestalten, wurde festgelegt, dass ein Spiel insgesamt nicht länger als 200 Spielaktionen dauern darf“	Spielidee
4.	Wenn die Anzahl gleichzeitig laufender Spiele 999 Spiele überschreitet, muss der Server alte Spiele automatisch entfernen und mit neuen, danach erstellten Spielen, ersetzten.	„Um die Ressourcen des Servers zu schonen gilt die Einschränkung, dass maximal 999 Spiele parallel ausgeführt werden dürfen. Sobald die Zahl der gleichzeitig laufenden Spieledie 999 Spiele übersteigt werden ältere Spiele automatisch entfernt und mit den neueren danach erstellen Spielen ersetzt.“	Netzwerk-protokoll
5.	Wenn das Spiel vor mehr als 10 Minuten erzeugt wurde, muss der Server das Spiel automatisch beenden.	„Weiters wird jedes Spiel 10 Minuten nach der Spielerzeugung automatisch entfernt/beendet, sodass keine weiteren Nachrichten an dieses gesandt werden können.“	Netzwerk-protokoll
6.	Wenn er eine Abfrage des Spielstatus durchgeführt hat, muss der Client(KI) mindestens 0.4 Sekunden warten, um die nächste Abfrage durchzuführen.	„Um zu verhindern, dass derServer überlastet wird sollte zwischen zwei vom gleichen Client durchgeführten Abfragen zum Spielstatus mindestens eine Zeitspanne von 0,4 Sekunden vergehen.“	Netzwerk-protokoll

Typ der Anforderung: Designbedingung

Nr.	Anforderung	Dokumentteil (Zitat)	Dokument
1.	Die Abgabe der Teilaufgabe 1 – Analyse & Entwurf erfolgt über GitLab	„**Durchführung der Abgabe:** Die Abgabe erfolgt über GitLab“	Moodle
2.	Austausch der Nachrichten zwischen Client und Server erfolgt im XML Format:	„Die ausgetauschten Daten bzw. Nachrichten werden im XML Format definiert bzw. erwartet. (Netzwerkprotokol)“	Netzwerk-protokoll

Aufgabe 2: Anforderungsdokumentation (2 Punkte)

Dokumentation Anforderung

- **Name:** Client erstellt Kartenhälfte
- **Beschreibung und Priorität:** Der Client ist verpflichtet, beim Start des Spiels selbstständig und automatisch eine von zwei Kartenhälften zu erstellen und sie dem Server zu übergeben. Dabei muss der Client bestimmte Regeln befolgen, sonst verliert er das Spiel.

Priorität: Hoch

- **Relevante Anforderungen:**
    - Beim Start des Spiels muss der Client eine Hälfte der Karte (4x8 Felder) erstellen.
    - Beim generieren der Karte muss der Client die Karte zufällig generieren und nicht statisch vorgeben.
    - Beim generieren der Karte muss der Client sie in Felder aufteilen.
    - Beim generieren der Karte kann der Client jedes Feld als eins von drei Terrainarten (Wasser, Wiese, Berg) repräsentieren.
    - Beim Generieren der Karte muss der Client beachten, dass eine minimale Anzahl an Terraintyp Feldern vorkommen (3xBergfelder; 15xWasserfelder; 4xWasserfelder).
    - Beim generieren der Karte muss der Client(KI) beachten, dass keine Inseln generiert werden dürfen bzw. Felder nicht vollständig von Wasserfeldern umschlossen werden dürfen.
    - Beim generieren der Karte muss der Client beachten, dass maximal 3 Wasserfelder an den langen und 1 Wasserfeld an den kurzen Seiten generiert werden können.
    - Beim generieren der Karte muss der Client beachten, dass zu jedem Feld einer Kartenhälfte (das nicht aus Wasser besteht) von jedem anderen Feld der Kartenhälfte (welches nicht aus Wasser besteht) ein Weg aus einer Reihe von validen Bewegungsbefehlen generiert werden können.
    - Beim erstellen der Kartenhälfte muss der Client, auf den von ihm erstellten Teil der Karte seine Burg platzieren.
    - Wenn er die Burg platzieren möchte, muss der Client das auf einem Wiesenfeld machen.
    - Falls eine der Kartenbedingungen nicht erfüllt ist, muss der Server den Client, welcher gegen die Bedingung verstoßen hat, über die Niederlage und den Gegner über den Sieg informieren.

- **Relevante Business Rules**:

    - 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).
    - Die Karte ist in Felder aufgeteilt, wobei jedes Feld genau eine von drei möglichen Terrainarten, nämlich Wasser, Wiese und Berg repräsentiert.
    - Die Spielfigur der KI startet jeweils an einer von ihr selbst definierten Position (ihrer Burg) auf dem von ihr erstellten Teil der Karte.
    - Wenn er die Burg platzieren möchte muss der Client beachten, dass die Burg und der Schatz nur auf Wiesenfeldern und nicht auf demselben Feld platziert werden können.

- **Impuls/Ergebnis - Typisches Szenario:**

***Vorbedingungen*:**
- o Ein neues Spiel am Server muss durch einen Menschen (User) erstellt werden. Hierzu ist es notwendig, dass er auf ein bekanntes Endpoint mittels einer HTTP GET Anfrage zugreift.
- o Der Server muss dem User auf seine Anfrage mit einer eindeutigen SpielID antworten.
- o Der Client muss mit der SpielID als Startparameter gestartet werden.

***Hauptsächlicher* Ablauf**:
- o Impuls: Der Client registriert sich beim Server und enthält seine SpielerID vom Server als Antwort.
- o Ergebnis: Der Client übernimmt die SpielerID uns sendet eine Kombination aus SpielerID und SpielID an den Server um anzufragen ob er an der Reihe ist einen Spielzug zu tätigen.
- o Impuls: Der Server antwortet dem Client mit einer responseEnvelope in welcher verzeichnet ist, dass der Client an der Reihe ist.
- o Ergebnis: Im Anschluss an die Statusabfrage startet der Client einen Algorithmus und generiert die Kartenhälfte.
- o Impuls: Das Prüfsystem des Clients bestätigt, dass die Kartenhälfte unter Einhaltung der Regeln erstellt wurde.
- o Ergebnis: Der Clients sendet die Kartenhälfte an den Server.

Nachbedingungen:
- o Der Server Antwortet auf die Kartenhälfte generisch mit einem responseEnvelope wo der state Element mit dem Text Okay definiert ist.
- o Der Client kann jetzt weitere Abfragen zum Spielstatus machen.

- **Impuls/Ergebnis - Alternativszenario:**

Vorbedingungen:
- o Ein neues Spiel am Server muss durch einen Menschen (User) erstellt werden. Hierzu ist es notwendig, dass er auf ein bekanntes Endpoint mittels einer HTTP GET Anfrage zugreift.
- o Der Server muss dem User auf seine Anfrage mit einer eindeutigen SpielID antworten.
- o Der Client muss mit der SpielID als Startparameter gestartet werden.

***Hauptsächlicher* Ablauf**:
- o Impuls: Der Client registriert sich beim Server und enthält seine SpielerID vom Server als Antwort.
- o Ergebnis: Der Client übernimmt die SpielerID uns sendet eine Kombination aus SpielerID und SpielID an den Server um anzufragen ob er an der Reihe ist einen Spielzug zu tätigen.
- o Impuls: Der Server antwortet dem Client mit einer responseEnvelope in welcher verzeichnet ist, dass der Client an der Reihe ist.
- o Ergebnis: Im Anschluss an die Statusabfrage startet der Client einen Algorithmus und generiert die Kartenhälfte.
- o Impuls: Das Prüfsystem des Clients entdeckt, dass eine der Regeln bezüglich Kartengenerierung nicht eingehalten wurde.
- o Ergebnis: Das Prüfsystem des Clients bestreitet, dass die Kartenhälfte unter Einhaltung der Regeln erstellt wurde.

Nachbedingungen:
- o Der Client muss erneut den Algorithmus zur Generierung der Kartenhälfte starten und die Einhaltung der Regeln prüfen.
- o Dieser Vorgang wird solange wiederholt bis alle Regeln zur Generierung der Kartenhälfte eingehalten sind.

- **Impuls/Ergebnis - Fehlerfall:**

Vorbedingungen:

- Ein neues Spiel am Server muss durch einen Menschen (User) erstellt werden. Hierzu ist es notwendig, dass er auf ein bekanntes Endpoint mittels einer HTTP GET Anfrage zugreift.
- Der Server muss dem User auf seine Anfrage mit einer eindeutigen SpielID antworten.
- Der Client muss mit der SpielID als Startparameter gestartet werden.

***Hauptsächlicher* Ablauf**:

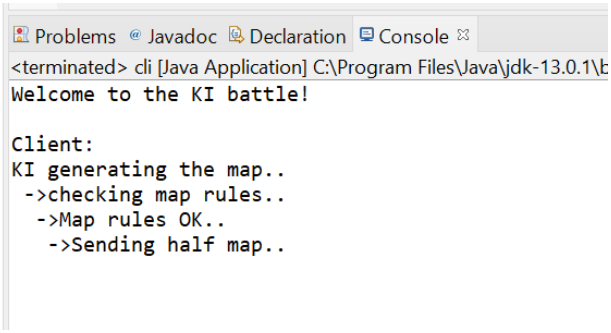
- Impuls: Der Client registriert sich beim Server und enthält seine SpielerID vom Server als Antwort.
- Ergebnis: Der Client übernimmt die SpielerID uns sendet eine Kombination aus SpielerID und SpielID an den Server um anzufragen ob er an der Reihe ist einen Spielzug zu tätigen.
- Impuls: Der Server antwortet dem Client mit einer responseEnvelope in welcher verzeichnet ist, dass der Client an der Reihe ist.
- Ergebnis: Im Anschluss an die Statusabfrage startet der Client einen Algorithmus und generiert die Kartenhälfte.
- Impuls: Das Prüfsystem des Clients bestätigt, dass die Kartenhälfte unter Einhaltung der Regeln erstellt wurde.
- Ergebnis: Der Clients sendet die Kartenhälfte an den Server.
- Impuls: Der Server Antwortet auf die Kartenhälfte generisch mit einem responseEnvelope wo der state Element mit dem Text Error definiert ist.
- Ergebnis: Der User wird durch die CLI über seine Niederlagen benachrichtigt.

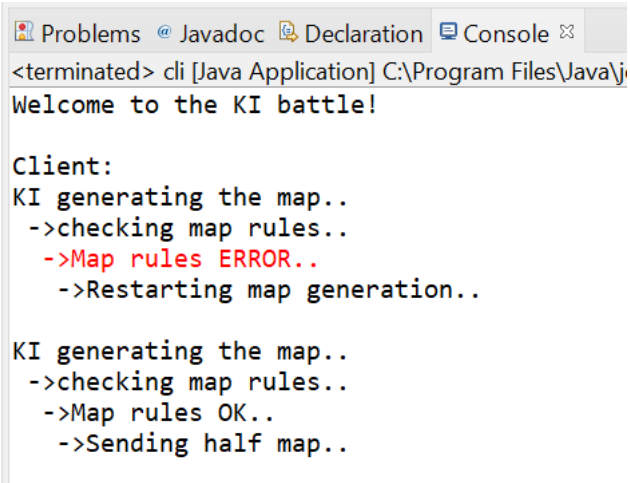
Nachbedingungen:

- Die Antwort des Servers auf die Kartenhälfte enthält im responseEnvelope neben dem state Element auch ein excepitonName und exceptionMessage.
- Der User ist durch die exceptionMessage im Klaren, wo der Misserfolg entstanden ist und kann das System entsprechend anpassen.

- **Benutzergeschichten:** Als (menschlicher) Anwender möchte ich mich nicht mit der komplexen Generierung der Karte unter Einhaltung der Regel befassen, sondern möchte diese Aufgabe dem Client überlassen.

- **Benutzerschnittstelle:** Während des Spiels wird über die CLI die Karte und deren bekannte Eigenschaften auf den Rechnern des menschlichen Spielers visualisieren. Dargestellt sind Beispiele für eine mögliche Darstellung des typischen Szenarios und eines alternativen Szenarios.





- **Externe Schnittstellen:**

  - Schnittstelle Server: Um das Spiel mit der gegnerischen KI realisieren zu können, muss der Client Daten mit dem Server (der unter anderem den Richter darstellt) austauschen.

  - Ausgetauschten Informationen:

    - Identifikationsnummern der Spieler sowie des Spiels
    - Daten zur Kartenhälfte bzw. die ganze Kartenhälfte
    - Informationen zu Spielstatus
    - Information zur Position bzw. zu einer beabsichtigten Bewegung auf eine neue Position

  - Protokol: Der Nachrichtenaustausch wird über eine Restschnittstelle durchgeführt, also wird das HTTP Protokoll verwendet sowie die zugehörigen Operationen GET und POST. Die ausgetauschten Daten werden im XML Format definiert.

  - Es ist außerdem notwendig eine stabile Netzwerkverbindung zu gewährleisten, um die Daten zwischen einander austauschen zu können

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

- **Klassendiagramme:**

  - Client: Das Klassendiagramm ist anhand der im Netzwerkprotokoll und Spielidee angegebenen Daten erstellt wurden und beinhaltet die Modelle: Map, Game, KI, Enumerations und Main sowie die in den Modellen enthaltenen Klassen inkl. Methoden. Dazu beinhaltet es auch ein nach dem Netzwerkprotokoll nachmodelliertes Network Protocol Modell inkl. Klassen und Methoden.

    Da das Klassendiagramm zu viel Platz benötigen können Sie dieses als separate SVG Datei (Client.svg) im Dokumentationsordner (siehe GitLab Repo) finden.

  - Server: Das Klassendiagramm ist anhand der im Netzwerkprotokoll und Spielidee angegebenen Daten erstellt wurden und beinhaltet die Modelle: Server, Data, IDGenerator, Controller, Converter, Verification, und Enumerations sowie die in den Modellen enthaltenen Klassen inkl. Methoden. Dazu beinhaltet es auch ein nach dem Netzwerkprotokoll nachmodelliertes Network Protocol Modell inkl. Klassen und Methoden.

    Da das Klassendiagramm zu viel Platz benötigen können Sie dieses als separate SVG Datei (Server.svg) im Dokumentationsordner (siehe GitLab Repo) finden.

- **Sequencediagramme**:

  - Client: Die Registrierung des Clients/Spielers am Server für eine bei Programmstart übergebene SpielID inklusive der Kartengenerierung und Übertragung wurde mittels Sequencediagramm unter Beachtung der im Client Klassendiagramm modellierten Klassen dargestellt.

    Da das Sequencediagramm zu viel Platz benötigen können Sie dieses als separate SVG Datei (Client-SequenceDiagram.svg) im Dokumentationsordner (siehe GitLab Repo) finden.

  - Server: Das empfangen und verarbeiten einer Bewegung. Begonnen mit der initial notwendige Spielstatusabfrage bis hin zum ausliefern eines aktualisierten Spielstatus nach Bewegungsabschluss wurde mittels Sequencediagramm unter Beachtung der im Server Klassendiagramm modellierten Klassen dargestellt.

    Da das Sequencediagramm zu viel Platz benötigen können Sie dieses als separate SVG Datei (Server-SequenceDiagram.svg) im Dokumentationsordner (siehe GitLab Repo) finden.