Software Engineering 1
Abgabedokument
Teilaufgabe 1
(Anforderungsanalyse und Planungsphase)

Persönliche Daten, bitte vollständig ausfüllen:

Nachname, Vorname:	Ali, Malaz
Matrikelnummer:	12231169
E-Mail-Adresse:	a12231169@unet.univie.ac.at
Datum:	15.10.2024

Dieses Dokument stellt den empfohlenen Aufbau dar, es muss aber nicht zwingend als Vorlage verwendet werden. Sollten Sie ohne Vorlage, z.B. in Latex, arbeiten behalten Sie bitte trotzdem die Reihenfolge/Struktur bei.

Die hier definierten Aufgabenstellungen stellen nur eine verkürzte unvollständige Zusammenfassung der realen Aufgabenstellungen dar. Diese ersetzen jedoch **nicht** die Übungsangabe bzw. die darin festgelegten Anforderungen.

Die Erstellung des Abgabedokuments kann auch weiterhin mit einem beliebigen Programm (z.B. Latex, LibreOffice etc.) erfolgen.

Aufgabe 1: Anforderungsanalyse

Analyse der Spielidee (Netzwerkprotokolldokumentation kann zusätzlich beim Verständnis der Spielidee helfen) um 8 Anforderungen (bestehend zumindest aus 3 funktionalen, 3 nichtfunktionalen und einer Designbedingung) nach den folgenden Kriterien zu dokumentieren. Achten Sie darauf die in Skriptum und der Vorlesung behandelten **Qualitätsaspekten** (besonders: atomar, Aktor/System, Aktion, wann, standardisierte Schlüsselwörter) durchgehend zu berücksichtigen.

Typ der Anforderung: funktional

Anforderung 1

- **Beschreibung:** Spiel erstellen- Als erstes muss der Mensch das Spiel erstellen
- **Bezugsquelle:** Spielidee: „Initial gilt es am Server ein neues Spiel anzufordern. Dieser erste Schritt wird noch von einem Menschen durchgeführt, alles danach erfolgt immer vollautomatisch durch einen Client/KI-Implementierung.“

Anforderung 2

- **Beschreibung:** Registrierung eines Clients- Der Client übermittelt seinen Vor- und Nachnamen sowie den u-account bei der Registrierung.
- **Bezugsquelle:** Netzwerkprotokoll:“ Als Body der Nachricht werden Informationen zum Entwickler des Clients übertragen. Dies wären der Vorname (studentFirstName), der Nachname (studentLastName) und Ihr u:account username (studentUAccount).“

Anforderung 3

- **Beschreibung:** Schatz verstecken- Nach dem Austausch der Kartenhälften versteckt der Server je einen Schatz auf jeder Kartenhälfte.
- **Bezugsquelle:** Spielidee: „Hierzu wird je ein Schatz vom Server auf jeder der beiden Kartenhälften versteckt.“

Anforderung 4

- **Beschreibung:** Kartenhälfte austauschen - Nach der Erstellung des Spiels, senden die Clients ihre Kartenhälfte an den Server damit dann eine vollständige Karte erstellt werden kann
- **Bezugsquelle:** Spielidee: „Nach Start des Clients registrieren sich die KIs für das Spiel am Server und erstellen/tauschen danach mit dem Server Kartenhälften aus. Die Karte, auf welcher gespielt wird, ist hierbei nicht fest vorgegeben, sondern entsteht durch die Kombination dieser zufällig erzeugten Kartenhälften durch den Server.“

Anforderung 5

Beschreibung: Spielstatus- Der Client muss den Spielstatus vom Server abfragen, um festzustellen, ob er an der Reihe ist oder warten muss.

Bezugsquelle: Spielidee: „Die Spielaktionen selbst werden rundenbasiert durchgeführt. Jede KI kann 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. Eine KI kann hierbei nicht auf das Setzen einer Aktion verzichten, sondern muss immer eine Aktion durchführen. Eine KI darf aber keine Aktion setzen solange die andere KI an der Reihe ist. Der Server unterstützt dies, können Clients doch abfragen ob diese gerade an der Reihe sind. Bestraft (indem

betroffene Clients verlieren) aber auch Clients die diese und andere Spielregeln nicht einhalten.“

Typ der Anforderung: nicht funktional

Anforderung 1

- **Beschreibung:** Kartengenerierung- Bei der Generierung der Karten gibt es bestimmte Regeln, die beachtet werden müssen.
- **Bezugsquelle:** Spielidee: „Jede Kartenhälfte muss mindestens 10% Bergfelder, 48% Wiesenfelder, 14% Wasserfelder und 2% Burg beinhalten. Kartenhälften müssen *zufällig* mit *Algorithmen* generiert und nicht statisch vorgegeben werden. Dürfen **keine nicht erreichbaren** aber potentiell betretbaren Felder enthalten sein. Um den **Wechsel zwischen** beiden **Kartenhälften** sicherzustellen, egal wie diese später vom Server kombiniert werden, müssen ≥ 51% der Felder jedes Randes betretbar sein.“

Anforderung 2

- **Beschreibung:** Bewegung der Spielfigur- Die KI muss innerhalb 5 Sekunden ein Bewegungsbefehl am Server senden.
- **Bezugsquelle:** Spielidee: „Für **jede dieser rundenbasierten Spielaktion hat die KI maximal 5 Sekunden** Bedenkzeit.“

Anforderung 3

- **Beschreibung:** Visualisierung des Spiels – "Die Clients müssen den aktuellen Zustand der Karte und wichtige Spielinformationen über die CLI für den Benutzer verständlich und nachvollziehbar visualisieren.
- **Bezugsquelle:** Spielidee: „Während des Spiels müssen die Karte und deren bekannten Eigenschaften und wichtige Spielzustände von den Clients mittels **command-line interface** (CLI) für Anwender nachvollziehbar visualisiert werden.

Typ der Anforderung: Designbedingung

Anforderung 1

- **Beschreibung:** Datenaustausch – Der Austausch von Daten zwischen Client und Server muss im XML-Format erfolgen.
- **Bezugsquelle:** Netzwerkprotokoll: “Die ausgetauschten Daten bzw. Nachrichten werden im XML Format definiert bzw. erwartet.“

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.

Dokumentation einer Beispielanforderung

- **Name:** Spielstatus
- **Beschreibung und Priorität:** Der Client kann den Spielstatus jederzeit abfragen so kann der Client wichtige Informationen erhalten, wie zum Beispiel ob der Client an der Reihe ist, um eine Bewegung zu machen oder ob der Client warten muss.
  Priorität: Hoch
- **Relevante Anforderungen:**
  - Spiel erstellen – Es muss zunächst ein Spiel erstellt werden, um im späteren Prozess eine Karte generieren zu können, um auf diese sich bewegen zu kömnnen
  - Registrieren eines Clients – Ein Client muss sich am Spiel beteiligen, um eine SpielerID zu erhalten. Mit dieser ID kann der Client eine Statusabfrage machen
  - Kartenhälften Generierung- Der Client generiert gemäß den Regeln eine Kartenhälfte
  - Kartenhälfte austauschen - Nach der Erstellung des Spiels, senden die Clients ihre Kartenhälfte an den Server damit dann eine vollständige Karte erstellt werden kann.
  - Schatz verstecken- Der Server versteckt je einen Schatz auf beiden Kartehälften.
- **Relevante Business Rules**:
  - Nachrichtenaustausch mit XML Format- Es muss eine erfolgreiche Kommunikation zwischen Client und Server stattfinden, damit der Client vom Server erfahren kann ob der Client eine Aktion durchführen kann (Serverantwort: „MustAct“) oder warten soll (Serverantwort: „MustWait“)
  - SpielerID und SpielID- Damit der Sever weiss um welchen Client es sich handelt und um welches Spiel und das entsprechende/passende Spielstatus zu geben, muss der Client bei der Statusabfrage die SpielerID und die SpielID mitgeben.
- **Impuls/Ergebnis – Typisches Szenario:**

  Vorbedingungen:
  - Neues Spiel wird von einem Menschen angefordert, durch Aufrufen einer URL auf einem Browser
  - Der Client bekommt eine GameID
  - Die KIs registrieren sich mit seinen Daten und bekommen anschließend vom Server eine PlayerID
  - Der Client hat eine Karte generiert
  - Der Client hat die Karte dem Server geschickt
  - Der Sever hat die Karte validiert
  - Der Server hat beide Kartenhälften von Client 1 und Client 2 kombiniert und somit eine vollständige Karte generiert
  - Der Server hat je ein Schatz auf beiden Kartenhälften versteckt

  Hauptsächlicher Ablauf:
  - Impuls: Der Client sendet ein HTTP GET Request an folgenden Endpunkt: **http(s)://<domain>:<port>/games/<SpielID>/states/<SpielerID>**

- Ergebnis: Der Server antwortet mit „MustAct“
- **Nachbedingungen**:
  - Der Client schickt dem Server eine Bewegung

- **Impuls/Ergebnis - Alternativszenario:**

  Vorbedingungen:
  - Neues Spiel wird von einem Menschen angefordert, durch Aufrufen einer URL auf einem Browser
  - Der Client bekommt eine GameID
  - Die KIs registrieren sich mit seinen Daten und bekommen anschließend vom Server eine PlayerID
  - Der Client hat eine Karte generiert
  - Der Client hat die Karte dem Server geschickt
  - Der Sever hat die Karte validiert
  - Der Server hat beide Kartenhälften von Client 1 und Client 2 kombiniert und somit eine vollständige Karte generiert
  - Der Server hat je ein Schatz auf beiden Kartenhälften versteckt

  Hauptsächlicher Ablauf:
  - Impuls: Der Client sendet ein HTTP GET Request an folgenden Endpunkt: **http(s)://<domain>:<port>/games/<SpielID>/states/<SpielerID>**
  - Ergebnis: Der Server antwortet mit „MustWait“

  Nachbedingungen:
  - Der Server macht immer wieder eine Statusabfrage, bis er die Antwort vom Server „MustAct“ bekommt

- **Impuls/Ergebnis – Fehlerfall:**

  Vorbedingungen:
  - Neues Spiel wird von einem Menschen angefordert, durch Aufrufen einer URL auf einem Browser
  - Der Client bekommt eine GameID
  - Die KIs registrieren sich mit seinen Daten und bekommen anschließend vom Server eine PlayerID
  - Der Client hat eine Karte generiert
  - Der Client hat die Karte dem Server geschickt
  - Der Sever hat die Karte validiert
  - Der Server hat beide Kartenhälften von Client 1 und Client 2 kombiniert und somit eine vollständige Karte generiert
  - Der Server hat je ein Schatz auf beiden Kartenhälften versteckt

  Hauptsächlicher Ablauf:
  - Impuls: Der Client sendet ein HTTP GET Request an folgenden Endpunkt: **http(s)://<domain>:<port>/games/<SpielID>/states/<SpielerID>**
  - Ergebnis: Das Internet funktioniert nicht und somit ist weder der Status noch das Spiel abrufbar

  Nachbedingungen:
  - Der Nutzer stellt eine Internetverbindung wieder her

- **Benutzergeschichten:**
  - Als Anwender möchte ich eine erfolgreiche Kommunikation zu dem Server haben, um zu erfahren, ob ich eine Aktion durchführen darf oder nicht.
  - Als Zuschauer möchte ich gern klar, deutlich und nachvollziehbar erkennen, welche Informationen im Spielstatus stehen.
- **Benutzerschnittstelle:** Im CLI wird ein Beispiel für ein Spielstatus und die dazu angegebenen Informationen dargestellt. Die dargestellten Informationen sind die

Nutzdaten der Spieler, die Spielstatus des Spielers und die Karte. Wobei „B“ auf der Karte für Burg steht, „S“ für Schatz und „EP“ für EnemyPosition.

```
uniquePlayerID:          33
firstName:               Malaz
lastName:                Ali
uaccount:                malaza97
state:                   MustWait
map:
```

W	G	M	G	G	G	G	W	W	G
W	G	W	G	G	M	W	G	G	G
M	G	G	W	M	G	EP	W	W	G
S	W	M	G	G	M	W	G	G	W
W	G	W	G	W	G	G	M	G	**B**

- **Externe Schnittstellen:**
  - Es wird folgende HTTP GET Request gesendet: ***http(s)://<domain>:<port>/games/<SpielID>/states/<SpielerID>***
  - Um das Spiel und den Spielsatus visualisieren zu können wird eine command- line- interface benötigt

Aufgabe 3: Architektur entwerfen, modellieren und validieren

Überlegen Sie sich und modellieren Sie händisch (daher z.B. nicht aus Code generiert oder abgemalt) alle notwendigen Packages, Klassen und deren Methoden/Konstruktoren (samt Daten und Beziehungen) als UML Klassendiagramme.

Es empfiehlt sich hierbei zuerst frei Hand - auf Papier oder digital - erste Ideen und Überlegungen festzuhalten, Listen potentieller Objekte zu erzeugen, usw. Danach, erst wenn ein grober Plan steht, diese dann digital in UML zu übertragen. Eine Anleitung/Empfehlung zum Vorgehen finden Sie in der Angabe und auch (samt Beispielanwendung) im dort verlinkten Skriptum.

Achten Sie darauf, dass die Modelle sinnvoll benannte Packages, **Klassen & deren Beziehungen**, **Konstruktoren** (inkl. Parameter), **Methoden** (inkl. Parameter und Rückgaben) und **Felder** beinhalten und die Vorgaben der Spielidee bzw. des Netzwerkprotokolls vollständig in sinnvoller Granularität abgedeckt werden. Parameter und Rückgabetypen sind wichtig um den Datenfluss abzubilden und stellen auch eine wesentliche Basis für die Sequenzdiagramme dar.

Basierend auf dem Klassendiagramm: Erstellen Sie Sequenzdiagramme zu den in der Übungsangabe vorgegebenen Szenarien. 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. Sie können diese Ausgangsbasis gerne aufgreifen. Erweitern und ergänzen Sie diese dann mit eigenen Überlegungen und fehlenden Details. Zum Nachlesen ist in der Angabe ein passendes Skriptum mit Beispielentwurf und Empfehlungen zum Entwurf (Best Practices) 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 dient als Vorlage und verdeutlicht welche Erwartungen an das Klassendiagramm beziehungsweise die dazugehörigen Sequenzdiagramme gestellt werden (Darstellung, Inhalte, Detailgrad etc.) und wie diese beiden Diagrammarten zusammenhängen. Geforderte Sequenzdiagramme *müssen* die Interaktion Ihrer Klassen aus *Ihrem* Klassendiagramm für die angegebenen Szenarien zeigen.
- 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 genutzt 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 reinen Datenaustausch 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 Sequenz- und Klassendiagramme zu übersehen.

Alle Diagramme mit einer kurzen Beschreibung hier einfügen. Als Dateiformat SVG nützen, dieses verlustlose Vektorformat (*keine* Rastergrafiken nützen) ist für UML sehr gut passend. Sollten die Diagramme zu viel Platz benötigen und deshalb schlecht lesbar sein können Sie diese auch als separate SVG Dateien im Dokumentationsordner der Teilaufgabe 1 (siehe, GitLab Repository) einfügen. Achten Sie bei der Verwendung von SVG Dateien darauf, dass sich diese fehlerfrei darstellen lassen. Als Referenz für die Anzeige ziehen wir den Renderer von Chrome/Chromium heran.

Aufgabe 4: Quellen dokumentieren

Dokumentieren Sie Ihre Quellen. Dies ist für Sie wichtig um die Einstufung einer Arbeit als Plagiat zu vermeiden. Inhalte die direkt aus dem Moodle Kurs dieses Semesters der LV Software Engineering 1 stammen können zur Vereinfachung weggelassen werden. Alle anderen Inhalte sind zu zitieren. Die Vorgabe des Studienpräses der Universität Wien lautet: *"Alle fremden Gedanken, die in die eigene Arbeit einfließen, müssen durch Quellenangaben belegt werden."*

Aufgabe 1: Anforderungsanalyse:

- Kurzbeschreibung der Übernommenen Teile: *Was & Wo im Projekt, In welchem Umfang (Idee, Konzept, Texte, Grafik etc.) mit und ohne Anpassungen, etc.*
- Quellen der Übernommenen Teile: *Folien, Bücher, Namen der Quell-Studierenden, URLs zu Webseiten, KI Prompts, etc.*

Aufgabe 2: Anforderungsdokumentation:

- Kurzbeschreibung der übernommenen Teile:
- Quellen der übernommenen Teile:

Aufgabe 3: Architektur entwerfen, modellieren und validieren:

- Kurzbeschreibung der übernommenen Teile:
- Quellen der übernommenen Teile: