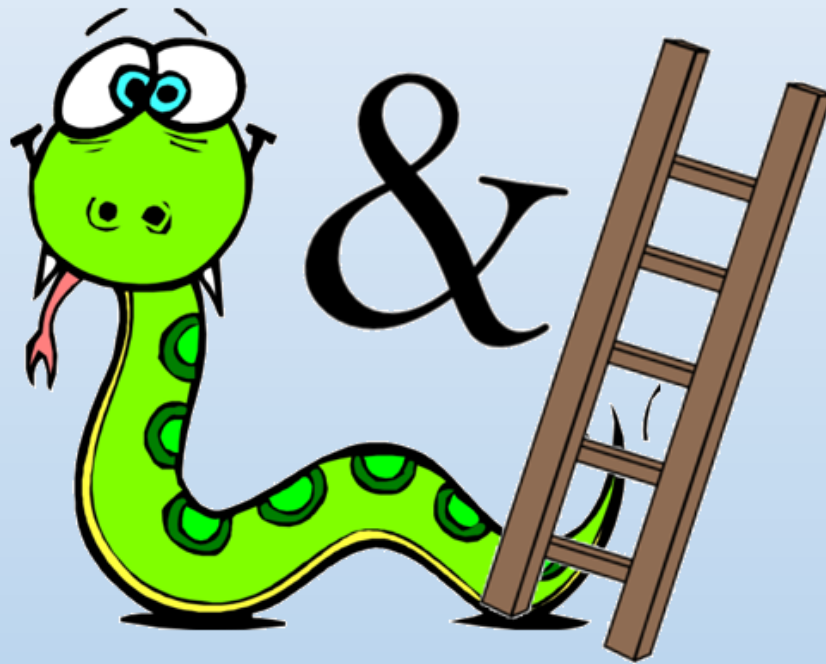


# PSIT3 – Snakes and Ladders

---

## Schlussbericht



Namen:	Dominik Steiner Jan Sonderegger Milan Natkeeran Philipp Wetzel Severin Roost
--------	------------------------------------------------------------------------------------------

Klasse:	IT16a_WIN
---------	-----------

Gruppe:	3
---------	---

Schule:	ZHAW
---------	------

Abgabetermin:	12.12.2017
---------------	------------

## Inhaltsverzeichnis

Versionierung .....	4
1. Projektidee .....	5
1.1. Ausgangslage .....	5
1.2. Idee .....	5
1.3. Kundennutzen .....	5
1.4. Stand der Technik/Konkurrenzanalyse .....	6
1.5. Hauptablauf .....	6
1.6. Wirtschaftlichkeit .....	7
2. Analyse .....	8
2.1. Domänenmodell .....	8
2.2. Anwendungsfälle .....	9
2.2.1. Anwendungsfalldiagramm .....	9
2.2.2. Use Case 1: Spiel Umgebung starten .....	9
2.2.3. Use Case 2: Spiel beitreten .....	10
2.2.4. Use Case 3: Frageliste einlesen .....	11
2.2.5. Use Case 4: Spiel spielen .....	12
3. Design .....	14
3.1. Architektur .....	14
3.1.1. Umsetzung auf dem Server .....	14
3.1.2. Umsetzung auf den Clients .....	14
3.2. Design-Klassendiagramm .....	15
3.2.1. Package Game: .....	15
3.2.2. Package Network: .....	16
3.2.3. Package Questionlist: .....	17
3.2.4. Package Util: .....	17
3.3. Interaktionsdiagramm .....	18
3.3.1. UC1 Spiel Umgebung starten .....	19
3.3.2. UC3 Frageliste einlesen .....	20
3.3.3. UC4 Spiel spielen .....	21
3.4. Designentscheide .....	23
4. Implementation .....	25
4.1. Testbericht .....	25
4.1.1. Spiel erstellen (Standard Spiel-modus) .....	25
4.1.2. Spiel erstellen (Erweiterter Spiel-Modus) .....	25
4.1.3. Frageliste einlesen .....	25
4.1.4. Spiel beitreten .....	25
4.1.5. Würfeln .....	26
4.1.6. Fragefeld-Aktion .....	26
4.1.7. Schlangenfeld-Aktion .....	26

4.1.8.	Leiterfeld-Aktion .....	26
4.1.9.	Spiel gewinnen.....	26
4.2.	Installationsanleitung.....	27
4.3.	Code Dokumentation.....	27
5.	Resultat.....	28
5.1.	Erreichte Ziele .....	28
5.2.	Mögliche Weiterentwicklungen .....	28
6.	Anhang.....	29
6.1.	Bedienungsanleitung.....	29
6.1.1.	Menü .....	29
6.1.2.	Spiel Umgebung starten.....	30
6.1.3.	Spiel beitreten .....	32
6.1.4.	Spielablauf .....	33
6.1.5.	Spielende.....	34
6.2.	Projektmanagement .....	35
6.2.1.	Grobplanung .....	35
6.2.2.	Risikoliste.....	36
6.2.3.	Vergangene Iterationen.....	37
6.2.3.1.	Iterationsplan #6 .....	37
6.2.3.2.	Iterationsplan #7 .....	38
6.2.4.	Feedback als Projektteam.....	39
6.3.	Anmerkung zum Design der Server-Architektur.....	40
6.4.	Literaturverzeichnis .....	41
6.5.	Glossar.....	42

## Versionierung

Version	Datum	Bearbeitet von	Änderung / Grund
1	24.11.17	Severin	Erstellung des Dokuments
1.1	25.11.17	Dominik	Domänenmodell und die Anwendungsfälle hinzugefügt
1.2	28.11.17	Severin	Projektidee aus Projektskizze übernommen und ergänzt
1.3	04.12.17	Severin	UC 4 und UC 4.1 überarbeitet und zu einem einzelnen zusammengefügt. Diverse bereits vorhandene Elemente eingefügt (Diagramme, Beschreibungen etc.)
1.4	10.12.17	Philipp	Kapitel Projektmanagement hinzugefügt
1.5	11.12.17	Philipp Jan Dominik Milan Severin Alle	Klassendiagramme durch aktuelle ersetzt Kapitel Designentscheide Kapitel Testbericht Kapitel Installationsanleitung Kapitel Resultat Diverse Ergänzungen und Korrekturen
1.6	11.12.17	Philipp	Rechtschreibkorrekturen durchgeführt
1.7	12.12.17	Philipp	PDF erstellt

Tabelle 1: Versionierung

# 1. Projektidee

## 1.1. Ausgangslage

Das «Leiterlenspiel» oder auch «Schlangen und Leitern» (Englisch: Snakes and Ladders) galt bisher als Kinder-Brettspiel und bis heute gibt es in der digitalen Welt noch keine Möglichkeit, dieses Spiel online mit Freunden zu spielen. Ausserdem sind die Regeln sehr begrenzt, da in diesem Spiel nur gewürfelt wird und es darum geht, wer als erster das Zielfeld erreicht. Zusätzlich gibt es Fallen, nämlich Felder mit einer Schlange, denn wenn ein Spieler auf ein solches Feld gelangt, muss er der Schlange entlang runterrutschen. Jedoch gibt es noch Glücksfelder mit einer Leiter, bei dem der Spieler hochklettert und dem Ziel näherkommt. Das Spiel ist in seiner Grundform ein reines Glücksspiel, weshalb es vor allem von Kindern gespielt wird.

## 1.2. Idee

Es soll eine Applikation entwickelt werden, welche das Leiterlenspiel mit einem Quiz erweitert und die Regeln damit anpassen lässt. Die Idee ist, dass mit einer bestehenden Liste von Fragen und Antworten das Spiel gestartet werden kann. Wenn eine Frage falsch beantwortet wird, muss der Spieler auf sein ursprüngliches Feld zurück. Das Leiterlenspiel wird somit ein dynamisches Spiel und kann zum Beispiel als Lernspiel verwendet werden. Mit der Umsetzung dieser Idee, wird dieses Spiel für alle Altersklassen interessant.

## 1.3. Kundennutzen

Wir sehen die folgenden Punkte als den wichtigsten Kundennutzen an:

Am Beispiel Lehrperson:

- Lehrpersonen haben die Möglichkeit, Fragen und Antworten selbst zu erfassen, dadurch kann das Spiel unabhängig vom Fach (Mathematik, Englisch etc.) genutzt werden.
- Lehrpersonen haben die Möglichkeit, den Lernstoff in einer spielerischen Art zu vermitteln, um den Unterricht abwechslungsreicher zu gestalten.
- Die Lehrperson kann Kontrollen und Lernstände besser erfassen, vergleichen und mit diesen Informationen dem Schüler Unterstützung bieten.

Am Beispiel Schüler/in:

- Die Schüler haben eine Möglichkeit ihren Schulstoff spielerisch zu erlernen.
- Die Eltern, wie auch die Schüler selbst, haben die Möglichkeit die Lernfortschritte selbst zu verfolgen.
- Das Spiel kann in der Schule, wie auch zuhause, für Lernzwecke eingesetzt werden.

Am Beispiel Schule/Bildung:

- Der Umgang mit Spielen und elektronischen Medien kann vermittelt werden.
- Durch das Spielen in Gruppen wird der Reiz, besser zu werden, gefördert werden, um dadurch eine bessere schulische Leistung zu erzielen.

Am Beispiel einer privaten Person:

- Der Benutzer / Die Benutzerin hat die Möglichkeit, die Fragen im Spiel selbst zu erfassen und sie seinen Bedürfnissen entsprechend anzupassen.
- Es besteht die Möglichkeit, in Gruppen das Spiel zu spielen.
- Die Applikation kann unterwegs, zum Beispiel im Zug, eingesetzt werden und ist Betriebssystem unabhängig.
- Die Applikation kann ebenfalls zum Lernen von Schulstoff eingesetzt werden.

## 1.4. Stand der Technik/Konkurrenzanalyse

Es gibt «Snakes and Ladders» bereits als Browser Game.

Unsere Version grenzt sich jedoch durch ein erweitertes, selbstkonfigurierbares Regelwerk ab, welches für ein viel flexibleres Spielerlebnis sorgt.

Zudem kann das Spiel auch als Lernmittel verwendet werden, wodurch es ein anderes bzw. grösseres Zielpublikum anspricht als andere Applikationen.

Folgende Beispiele wurden online gefunden und mit der Idee von Snakes and Ladders verglichen:

Beispiele Konkurrenzprodukte [Online]

URL: <http://www.playonlinedicegames.com/snakesandladders> [Stand: 26.09.2017].

URL: <http://www.counon.org/games/virtualmathfest/snakesladders.html> [Stand: 26.09.2017].

URL: <http://www.agame.com/game/snakes-and-ladders> [Stand: 26.09.2017].

Originale Spielregeln [Online]

URL: <https://de.wikihow.com/Snakes-and-Ladders-spielen> [Stand: 26.09.2017].

Keine dieser aufgeführten Spiele bieten die Möglichkeit freidefinierbaren Lerninhalt in das Spielgeschehen einzubringen.

## 1.5. Hauptablauf

Der Hauptanwendungsfall ist eine Gruppe Spieler, die eine Partie «Snakes & Ladders» spielen wollen:

- Die Spieler starten ein neues Spiel und geben die Anzahl Teilnehmer an
- Die Spieler wählen den Modus «klassisch» (ohne zusätzliche Regeln) oder «erweitert» (zusätzliche Regeln wie z.B. Quiz-Felder). Falls Letzteres gewählt wird, muss zudem noch das Themengebiet für die Quiz-Fragen angegeben werden.
- Ist ein Spieler am Zug, kann er die Aktion wählen, einen (virtuellen) Würfel zu werfen. Anschliessend bewegt sich seine Spielfigur entsprechend.
- Bleibt die Figur auf einem Quiz-Feld stehen, wird eine Frage aus der hinterlegten Datenbank gewählt, auf welche der Spieler (per Tastatureingabe) eine Antwort eingeben kann.
- Bei einer falschen Antwort wird die Figur auf ihr ursprüngliches Feld zurückbewegt.
- Das Spielfeld, die Figuren und wer an der Reihe ist, können die Spieler jederzeit der visuellen Darstellung entnehmen.
- Der Spielvorgang wird wiederholt und es wird angezeigt, welcher Spieler jeweils am Zug ist.
- Eine Partie endet, wenn ein Spieler das Ziel erreicht hat.

## 1.6. Wirtschaftlichkeit

Um das Spiel einem breiten Publikum zugänglich zu machen, müssen Spiel-Server bereitgestellt werden.

Dies bedeutet Anschaffungs- sowie auch Maintenance-Kosten für die Serverumgebung. Die monatlichen Ausgaben belaufen sich auf ca. 50 CHF für die Miete einer solchen Umgebung, pro Jahr ergibt dies einen Aufwand von 600 CHF.

Für die Programmierung der Softwarekomponenten des Projekts sind 100h pro Teammitglied geplant. Bei 5 Teammitgliedern ergibt dies bei einem Stundenansatz von 100 CHF/h einen Aufwand von ungefähr 50'000 CHF.

Als Einnahmequellen sind zwei Szenarien denkbar:

Zum einen kann durch den Einsatz im Bildungssektor eine Lizenzgebühr pro Schule verlangt werden. Die potentiellen Abnehmer sind dort beschränkt, in der Schweiz gibt es etwas über 9500 Schulen auf Primar- und Sekundarstufe I.<sup>1</sup> Bei einer Lizenzgebühr von 100-200 CHF wird davon ausgegangen, dass bei einem guten Produkt nach 3 Jahren ca 10% der Institutionen als Kunden gewonnen werden können. Dies entspricht im Minimum Einnahmen von 100'000 CHF nach 3 Jahren. Der Break-Even-Point wird also nach etwa 1.5 Jahren erreicht sein.

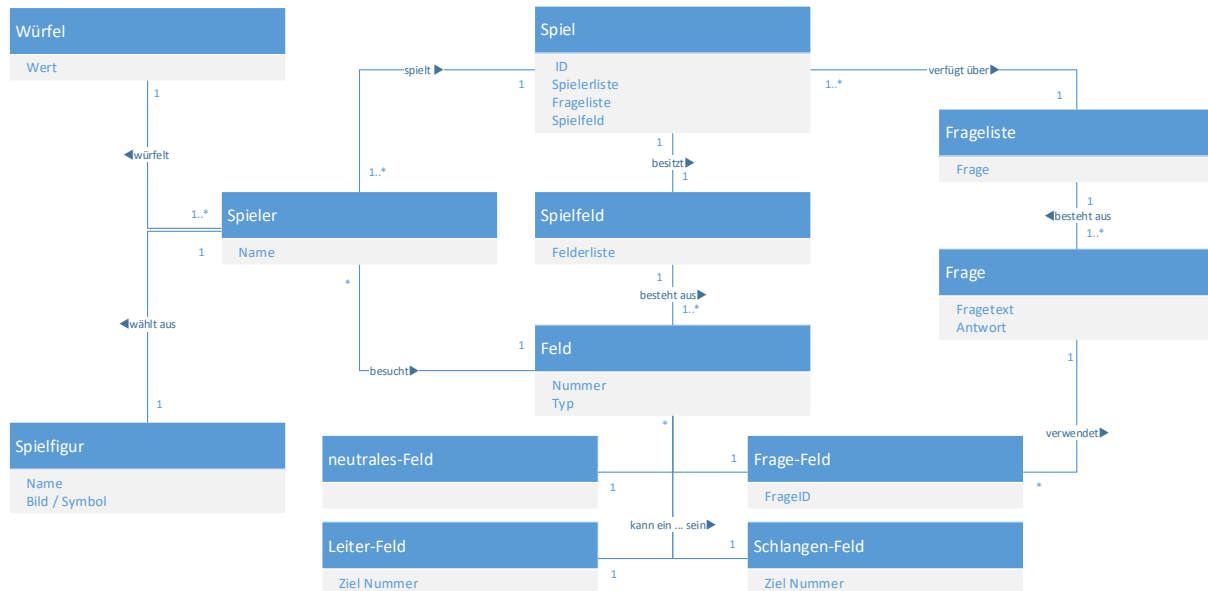
In einem zweiten Szenario könnte versucht werden, das Spiel gratis über das Internet einer breiten Masse bekannt zu machen und so die Attraktivität für Werbung zu erhöhen, wie das bei Spielen heutzutage üblich ist. Eine Vorhersage über die Einnahmen ist hierbei jedoch schwierig zu machen. Es wird nicht erwartet, dass bei diesem Modell der Break-Even-Point schneller erreicht wird.

---

<sup>1</sup> Quelle: Bundesamt für Statistik: <https://www.bfs.admin.ch/bfs/de/home/statistiken/bildungswissenschaft.html> , Stand: Dezember 2017

## 2. Analyse

### 2.1. Domänenmodell



Ein Spieler kann eine Spielfigur auswählen und einen Würfel würfeln, der von allen Spielern geteilt wird, d.h. alle Spieler haben zusammen für das Spiel ein Würfel.

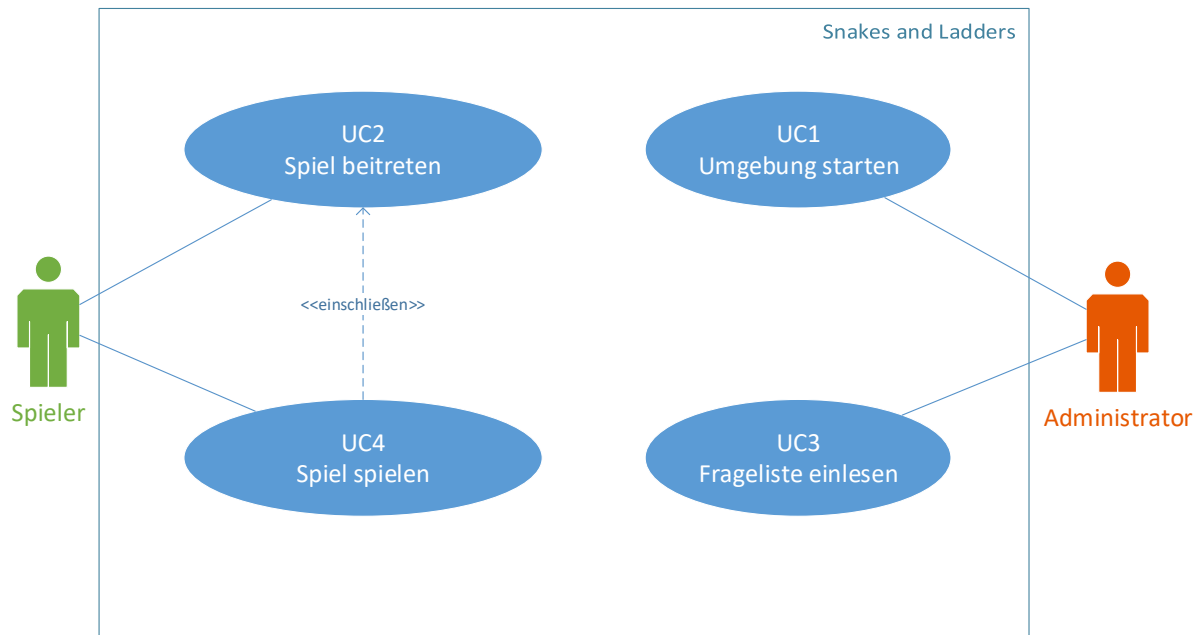
Im Spiel gibt es ein Spielfeld, das eine bestimmte Anzahl Felder hat. Diese Felder können vier verschiedene Typen haben. Entweder ist das Feld ein neutrales-Feld, Frage-Feld, Leiter-Feld oder ein Schlangen-Feld. Bei den zwei Felder Leiter und Schlangen-Feld gibt es ein Attribut mit Ziel Nummer, da die Spielfigur von diesem Feld zu einem anderen Feld (Ziel Feld) positioniert wird.

Das Frage-Feld verwendet die Fragen von der Frageliste, welche das Spiel zur Verfügung hat.



## 2.2. Anwendungsfälle

### 2.2.1. Anwendungsfalldiagramm



### 2.2.2. Use Case 1: Spiel Umgebung starten

Umfang:	Snakes and Ladders	
Ebene:	Anwenderziel	
Primärakteur:	Administrator	
Stakeholder und Interessen:	Spieler: Möchten das Spiel so unkompliziert wie möglich starten.  Administrator: Das Spiel den Voraussetzungen entsprechend einrichten und das Spiel bereitstellen zum Verbinden.	
Vorbedingungen:	Entscheidung zwischen klassischen oder erweiterten Modus.  Beim Erweiterten Modus können eigene Fragen für das Quiz verwendet werden. Die Fragenliste muss vorher vorbereitet werden. Eine Vorlage für die Fragenliste kann von der Applikation entnommen werden, um diese dann zu bearbeiten und auch hochzuladen.	
Nachbedingungen:	Applikation wurde nach dem Spielen erfolgreich beendet (Verbindungen wurden sicher getrennt).	
Standardablauf:	<u>Klassisches Spiel-Modus:</u>  1. Administrator gibt für das Spiel einen Namen ein. 2. Administrator gibt für das Spiel ein Passwort ein (Passwort ist optional). 3. Administrator tippt die Spieleranzahl ein. 4. Administrator wählt den klassischen Spiel-Modus aus (Standard Spiel-Modus).	<u>Erweitertes Spiel-Modus:</u>  1. Administrator gibt für das Spiel einen Namen ein. 2. Administrator gibt für das Spiel ein Passwort ein (Passwort ist optional). 3. Administrator tippt die Spieleranzahl ein. 4. Administrator wählt den erweiterten Spiel-Modus aus. 5. Administrator liest die vorbereiteten Fragen ein.

	5. Administrator stellt den Server für die Verbindung bereit bzw. erstellt die Spiel Umgebung.	6. Administrator stellt den Server für die Verbindung bereit bzw. erstellt die Spiel Umgebung.
Erweiterungen:	5a) Statt im erweiterten Spiel-Modus die Frageliste selber zu vorbereiten können bereits zur Verfügung gestellte Fragelisten ausgewählt werden.	
Spezielle Anforderungen:	Die Fragenliste muss der Vorlage entsprechen, damit das Spiel erfolgreich gestartet werden kann.  Netzwerkverbindung muss gewährleistet sein.	
Liste der Technik- und Datenvariationen:	Computer wird für die Server-Applikation benötigt.	
Häufigkeit des Auftretens:	Einmalig (bevor das Spiel gestartet wird)	
Verschiedenes (z.B. Offene Fragen):	Keine	

### 2.2.3. Use Case 2: Spiel beitreten

Umfang:	Snakes & Ladders Software
Ebene:	Benutzer-Funktion
Primärakteur:	Spieler
Stakeholder und Interessen:	Ein Spieler möchte an einem Spiel teilnehmen und sich auf dem Server registrieren. Der Administrator hat den Server aufgesetzt und wartet auf Spieler, welche dem Server beitreten möchten.
Vorbedingungen:	Spiel-Server wurde konfiguriert und kann erreicht werden.
Nachbedingungen:	Spieler kann an einem Spiel teilnehmen, Spielfeld und Gegenspieler werden angezeigt. Das Spiel kann gestartet werden.
Standardablauf:	<ol style="list-style-type: none"> <li>1. Spieler öffnet GUI und gelangt zum Menü</li> <li>2. Spieler wählt Knopf «Server beitreten»</li> <li>3. Spieler kann aus einer Liste den gewünschten Server auswählen</li> <li>4. Spieler schickt eine Anfrage an den Server-Administrator</li> <li>5. Server-Administrator nimmt den Spieler an</li> <li>6. Spieler ist auf dem Server registriert und kann am Spiel nun teilnehmen</li> </ol>
Erweiterungen:	<p>4a. Der Server ist nicht erreichbar Es erscheint eine Fehlermeldung Der Spieler kann weitere Anfragen starten</p> <p>5a. Der Admin nimmt den Spieler nicht auf dem Spiel-Server an Dem Spieler wird eine entsprechende Meldung angezeigt Der Spieler kann weitere Anfragen starten</p>

Spezielle Anforderungen:	Ein Spiel-Server ist gestartet Es muss eine Internet-Verbindung bestehen
Liste der Technik- und Datenvariationen:	-
Häufigkeit des Auftretens:	Pro Spieler einmal, sofern die Verbindung korrekt hergestellt wird. Im Falle eines Fehlers pro Spieler mehrfach.
Verschiedenes (z.B. Offene Fragen):	Muss ein Passwort eingegeben werden um einem Server beizutreten? Wie wird die Liste der verfügbaren Server erstellt bzw. wie werden die Server identifiziert?

#### 2.2.4. Use Case 3: Frageliste einlesen

Umfang:	Server Applikation
Ebene:	Anwenderziel
Primärakteur:	Administrator
Stakeholder und Interessen:	Administrator: Will schnell und ohne Probleme seine Frageliste in das System laden Spieler: Möchte das Spiel spielen
Vorbedingungen:	Das Spiel ist nicht gestartet worden, denn der Server ist noch in der Konfigurationsphase.
Nachbedingungen:	Die eingelesenen Fragen und Antworten sind im Server für das nächste Spiel zwischengespeichert.
Standardablauf:	<ol style="list-style-type: none"> <li>1. Der Administrator begibt sich im Server zur Ansicht, bei dem er die Frageliste einlesen kann.</li> <li>2. Administrator wählt die Option um eine neue Frageliste hochzuladen.</li> <li>3. Administrator wählt im eigenem Dateisystem die Datei mit der Frageliste aus und lädt sie hoch</li> <li>4. Das System speichert die Frageliste für das nächste Spiel</li> <li>5. Administrator bestätigt die neuen Änderungen</li> </ol>
Erweiterungen:	<p>*2a. Administrator kann auswählen, welche Fragen nicht im Spiel sein sollten.</p> <p>*2b. Administrator kann das Format für die Frageliste-Datei selber definieren.</p> <p>*2c. In den Fragen kann man auch Bilder anzeigen lassen.</p> <p>3a. Ungültiges Format bei der Frageliste</p> <ol style="list-style-type: none"> <li>1. System signalisiert den Fehler und ignoriert die Frageliste</li> <li>2. Der Benutzer hat die Möglichkeit, nochmals eine Frageliste hochzuladen, falls er das möchte.</li> </ol> <p>3b. Frageliste wird doppelt hochgeladen</p> <ol style="list-style-type: none"> <li>1. System signalisiert, dass die hochgeladene Frageliste schon im System ist.</li> <li>2. Der Benutzer hat die Möglichkeit eine andere Frageliste hochzuladen, falls er das möchte.</li> </ol>
Spezielle Anforderungen:	<ul style="list-style-type: none"> <li>• auch grosse Fragelisten sollten mühelos benutzbar sein.</li> <li>• Die Frageliste soll auch spezielle Symbole (z.B. mathematische) unterstützen</li> </ul>

Liste der Technik- und Datenvariationen:	2a. Eingabe der Fragen und Antworten per Tastatur direkt im System.
Häufigkeit des Auftretens:	oft – maximal ein Mal pro Spiel
Verschiedenes (z.B. Offene Fragen):	Was passiert wenn das System des Administrators während der Konfiguration abstürzt?

### 2.2.5. Use Case 4: Spiel spielen

Umfang:	Snakes and Ladders
Ebene:	Anwenderziel
Primärakteur:	Spieler
Stakeholder und Interessen:	Spieler: Will das Spiel spielen und als erstes das letzte Feld erreichen.
Vorbedingungen:	Spieler konnte einem Spiel beitreten. Das Spiel wurde gestartet.
Nachbedingungen:	Neue Position des Spielers wurde gespeichert und wird ihm visuell korrekt auf dem Spielfeld dargestellt. Beantwortete Fragen wurden im Spielstand des Spielers gespeichert.
Standardablauf:	<ol style="list-style-type: none"> <li>1. Spieler würfelt (virtuell), Spieler sieht gewürfelte Zahl</li> <li>2. Applikation fährt mit der Spielerfigur die gewürfelte Anzahl Felder weiter</li> <li>3. Bei einem normalen Feld <ol style="list-style-type: none"> <li>3.1 Spieler sieht neue Position seiner Figur</li> </ol> </li> <li>4. Bei einer Leiter <ol style="list-style-type: none"> <li>4.1 Spielfigur bewegt sich ans Ende der Leiter</li> <li>4.2 Spieler sieht neue Position seiner Figur</li> </ol> </li> <li>5. Bei einer Schlange <ol style="list-style-type: none"> <li>5.1 Spielfigur bewegt sich ans Ende der Schlange</li> <li>5.2 Spieler sieht neue Position seiner Figur</li> </ol> </li> <li>6. Bei einem Fragefeld <ol style="list-style-type: none"> <li>6.1 Die Applikation wählt eine Frage aus und stellt diese dem Spieler</li> <li>6.2 Spieler beantwortet Frage</li> <li>6.3 Die Applikation wertet die Antwort aus und zeigt dem Spieler das Resultat</li> <li>6.4 Bei richtiger Antwort bleibt die Spielfigur auf dem Feld</li> <li>6.5 Bei falscher Antwort wird die Spielfigur zurückgesetzt</li> </ol> </li> <li>7. Beim letzten Feld <ol style="list-style-type: none"> <li>7.1 Die Applikation wählt eine Frage aus und stellt diese dem Spieler</li> <li>7.2 Spieler beantwortet Frage</li> <li>7.3 Die Applikation wertet die Antwort aus und zeigt dem Spieler das Resultat</li> <li>7.4 Bei richtiger Antwort ist das Spiel beendet</li> <li>7.5 Bei falscher Antwort wird die Spielfigur zurückgesetzt</li> </ol> </li> </ol>
Erweiterungen:	<p>1a. Spieler gibt Würfelzahl per Tastatur ein.</p> <p>*a. Jederzeit, wenn das System des Spielers ausfällt:</p> <ol style="list-style-type: none"> <li>1. Spieler verbindet sich neu</li> <li>2. Standardablauf</li> </ol>

Spezielle Anforderungen:	Synchronisation der Spielstände an die Spieler muss in 0.5s erfolgen.
Liste der Technik- und Datenvariationen:	1a. Die Eingabe des Wertes des Würfels erfolgt über die eingblendete oder angeschlossene Tastatur.
Häufigkeit des Auftretens:	Beinahe laufend.
Verschiedenes (z.B. Offene Fragen):	Wie ist der Ablauf für den Spieler, wenn das System auf den Administratoren Seite ausfällt? Was passiert, wenn beim letzten Feld eine höhere Zahl gewürfelt wird?

## 3. Design

### 3.1. Architektur

Die Applikation ist eine klassische Client-Server-Anwendung. Auf einem Server ist die gesamte Game-Logik implementiert und Clients können sich auf den Server verbinden und so an einem Spiel teilnehmen. Die Clients erhalten dabei für gelieferte Inputs (Würfeln, Frage beantworten) von Server eine entsprechende Antwort und aktualisieren aufgrund dieser Antworten ihr GUI.

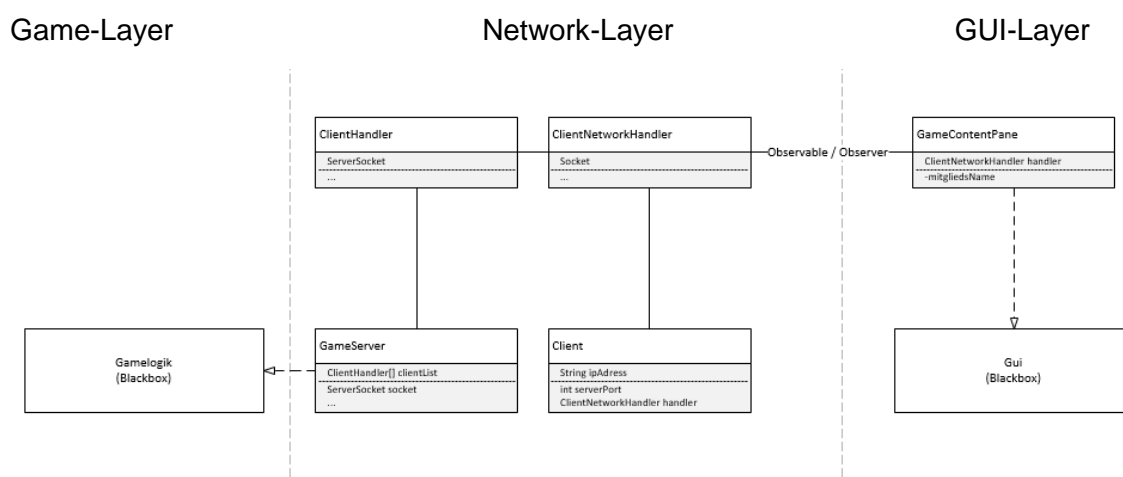
#### 3.1.1. Umsetzung auf dem Server

Die Verbindung von Client und Server wird über die Networking-API von Java realisiert. Ein Server-Objekt bindet dabei einen Port einer IP-Adresse an einen ServerSocket. Ist dieser Socket erfolgreich erstellt worden, hört der Server auf einem eigenen Thread auf Verbindungsanfragen von Clients. Für jede erstellte Verbindung mit einem Client wird ein ClientHandler mit eigenem Thread erstellt, welcher Nachrichten von und zu seinem Client behandelt. Die dazu benötigten Streams werden über entsprechende getter-Methoden des Socket geholt (siehe dazu JAVA-Dokumentation).

#### 3.1.2. Umsetzung auf den Clients

Ein Client-Objekt versucht über einen Socket eine Verbindung zum Server aufzubauen. Dabei wird ein ClientNetworkHandler erstellt, welcher bei erfolgreich erstellter Verbindung in einem eigenen Thread auf Nachrichten wartet oder bei Bedarf an den Server versendet. Auch hier werden die getter-Methoden des Socket verwendet, um die Streams zu erstellen und benutzen (siehe dazu JAVA-Dokumentation).

Damit das GUI des Clients sich stets ankommende Nachrichten aktualisieren kann, wird das Observer-Interface von Java verwendet. Dabei implementiert der ClientNetworkHandler das Observable-Interface und die Klasse GameContentPane das Observer-Interface. Empfängt der ClientNetworkHandler eine neue Nachricht, wird die notify-Methode aufgerufen um das GameContentPane zu informieren und die Aktualisierung des GUIs einzuleiten.

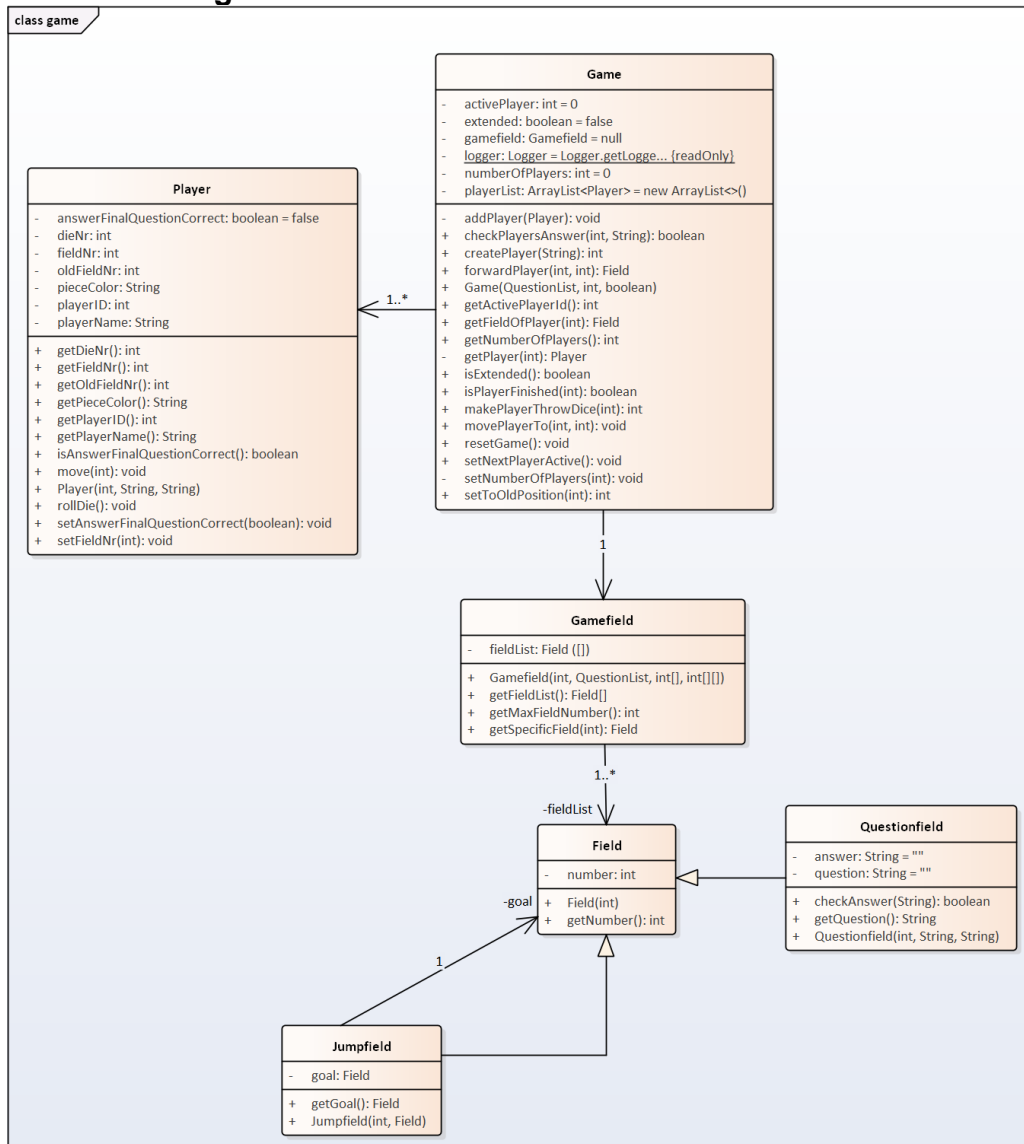


Die Grafik soll nicht die Klassen in ihrer endgültigen Form beschreiben, sondern lediglich die Abgrenzung zwischen Netzwerk-, Game-, und Gui-Logik verdeutlichen. Dabei laufen jeweils die Klassen `GameServer`, `ClientHandler` und `ClientNetworkHandler` in einem eigenen Thread.

## 3.2. Design-Klassendiagramm

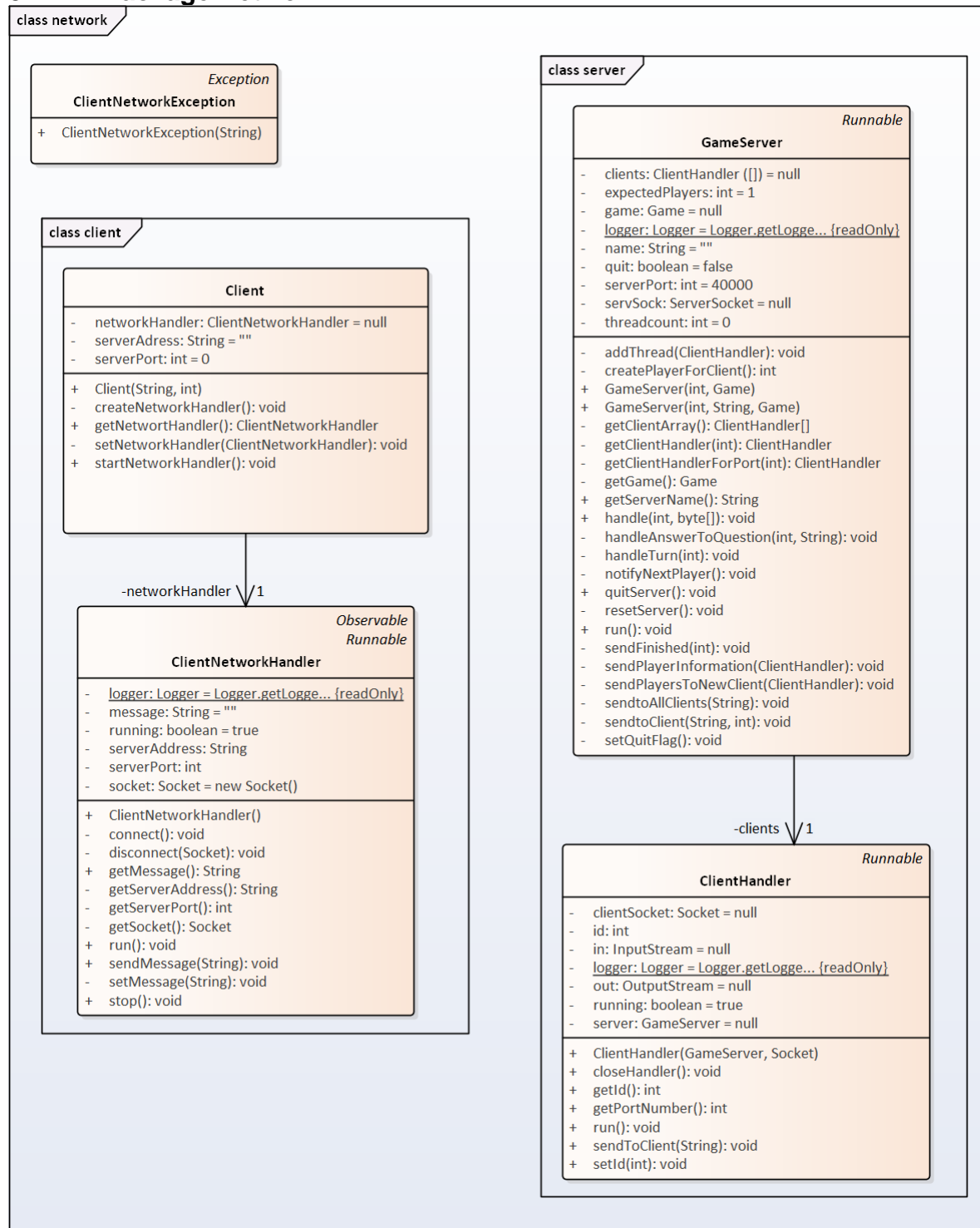
Für Snakes and Ladders wurden die vier Haupt-Packages Game, Network, Questionlist und GUI für die jeweiligen Klassen erstellt, zusätzlich haben wir noch das Package Util für die allgemeinen Einstellungen.

### 3.2.1. Package Game:



Im Package Game wurden die Klassen Player, Game, Gamefield, Field, Jumpfield und Questionfield erstellt. Die Klasse Game hat Zugriff auf die Klassen Player und Gamefield, damit das Spiel diese zwei Haupt-Objekte darstellen kann. Die Klasse Gamefield besteht aus der Klasse Field, welches entweder ein neutrales Field, Jumpfield oder ein Questionfield ist.

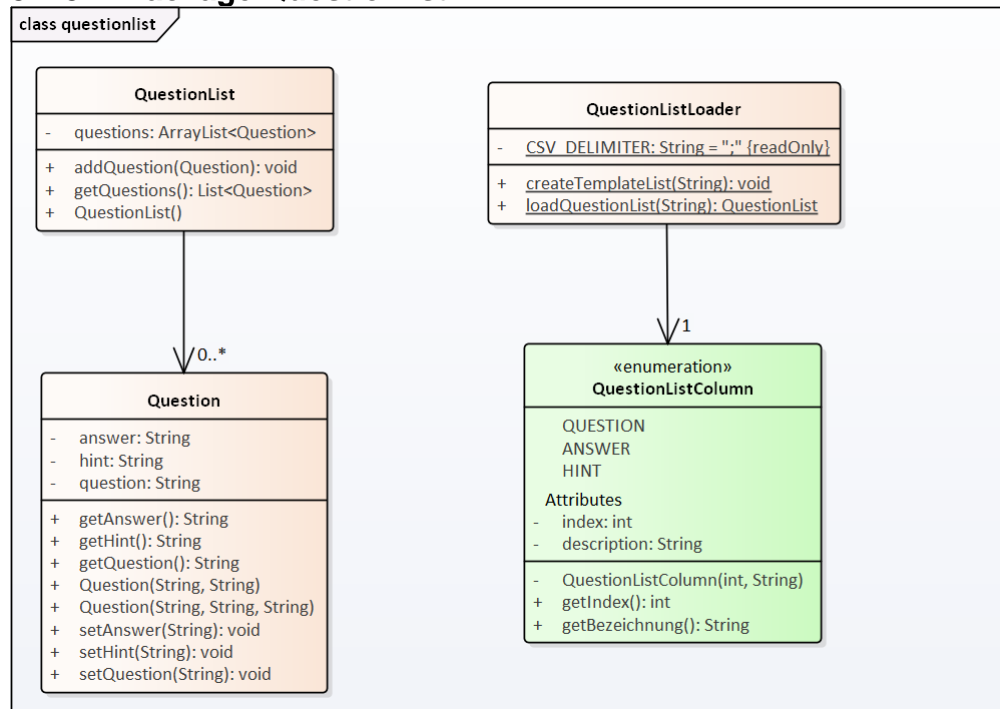
### 3.2.2. Package Network:



Im Package Network wurden die zwei Unterpackages Client und Server erstellt. Diese zwei Unterpackages beinhalten die Klassen Client, ClientNetworkHandler, GameServer und ClientHandler. Diese sind für die Kommunikation für das Spiel Snakes and Ladders verantwortlich.

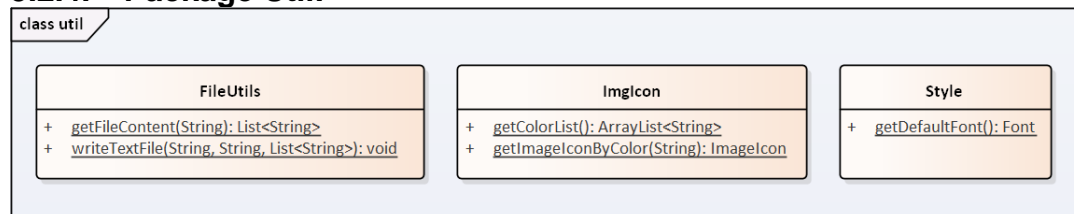


### 3.2.3. Package Questionlist:



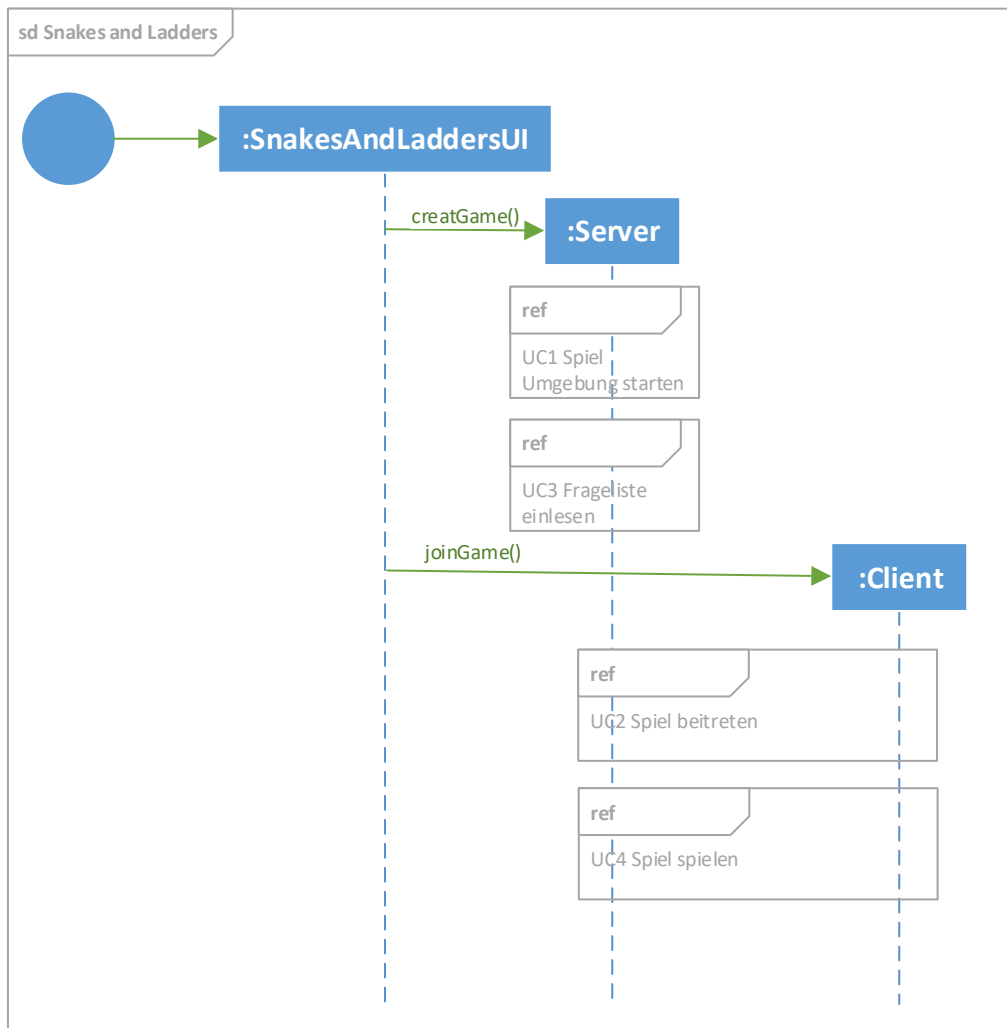
Im Package Questionlist wurden die vier Klassen QuestionList, Question, QuestionListLoader und QuestionListColumn erstellt. Diese Klassen sind dafür verantwortlich die Fragen von der Liste entsprechend zu laden und sie anschliessend dem Spiel zur Verfügung zu stellen.

### 3.2.4. Package Util:



Im Package Util werden in den drei Klassen FileUtils, ImgIcon und Style allgemeine Einstellungen für das Spiel festgelegt.

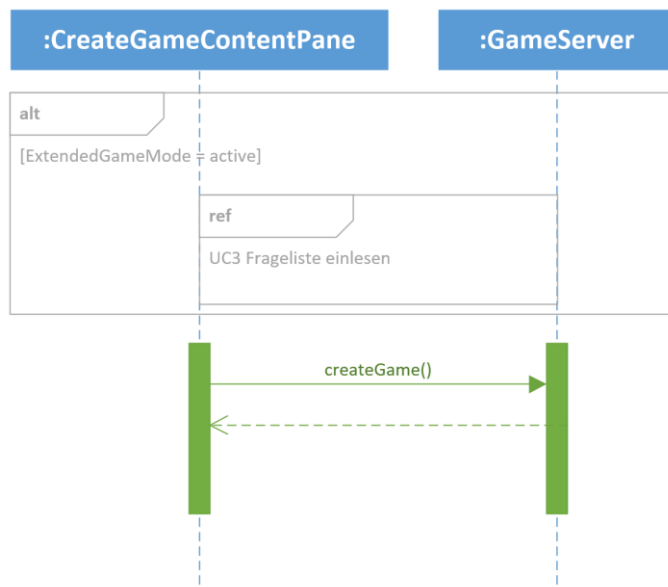
### 3.3. Interaktionsdiagramm



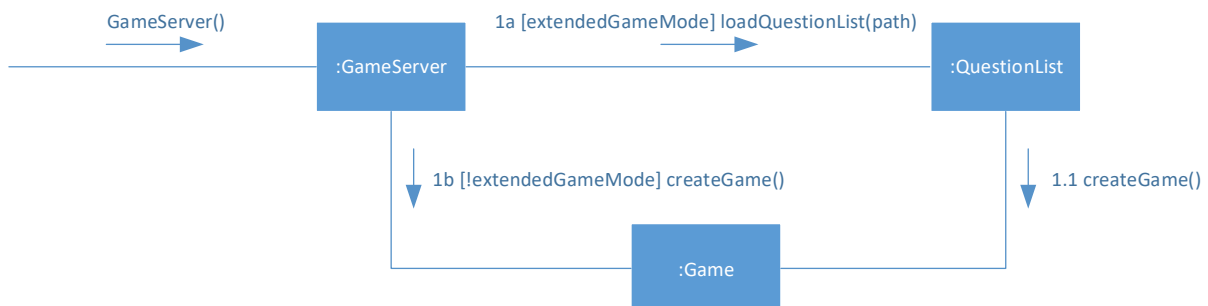
Dieses Diagramm dient zur Übersicht die nachfolgenden Sequenzdiagramme und ist sehr einfach gehalten.

Der Start folgt im SnakesAndLadders UI welche zwei Optionen zu Verfügung stellt. Man kann einen Server erstellen oder einen Client starten. Details zum Server und dem Client sind in den Nachfolgenden Diagrammen beschrieben.

### 3.3.1. UC1 Spiel Umgebung starten

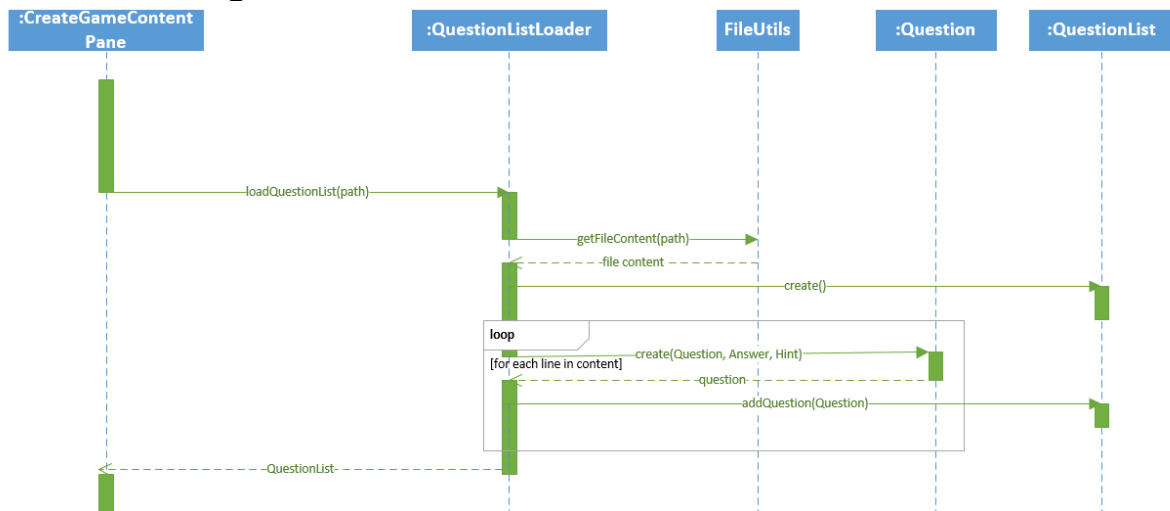


Nachdem die erforderlichen Felder im CreateGameContentPane ausgefüllt wurden, kann noch entschieden werden, ob der erweiterte Spielmodus gestartet werden soll und wenn dies zutrifft, muss die Frageliste eingelesen werden (siehe Kapitel 5.3) und das Spiel kann erstellt werden, ansonsten kann das Spiel direkt gestartet werden.



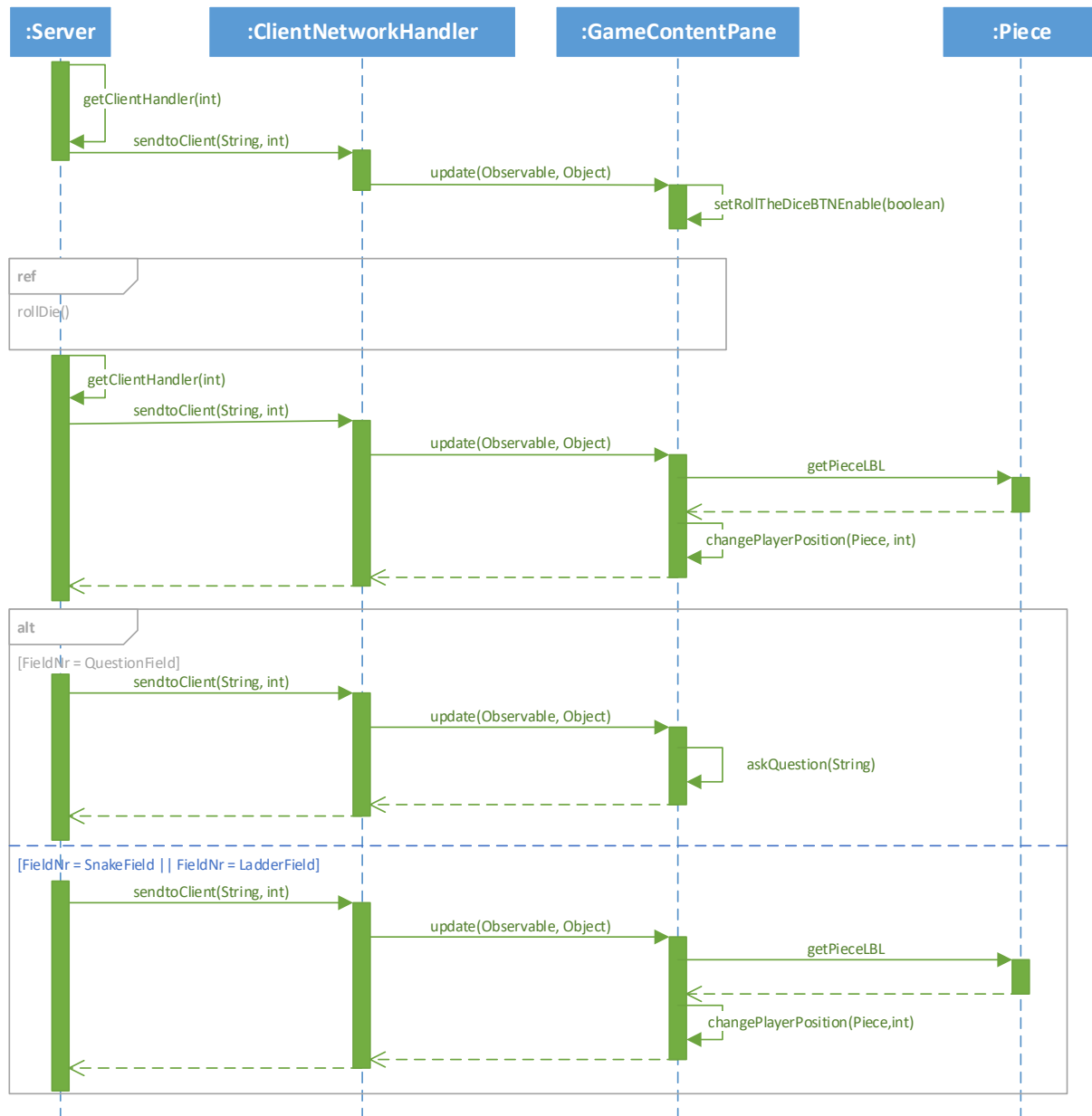
Sobald der Administrator das Spiel erstellt, wird die Kommunikation analog dem Sequenzdiagramm ausgelöst.

### 3.3.2. UC3 Frageliste einlesen

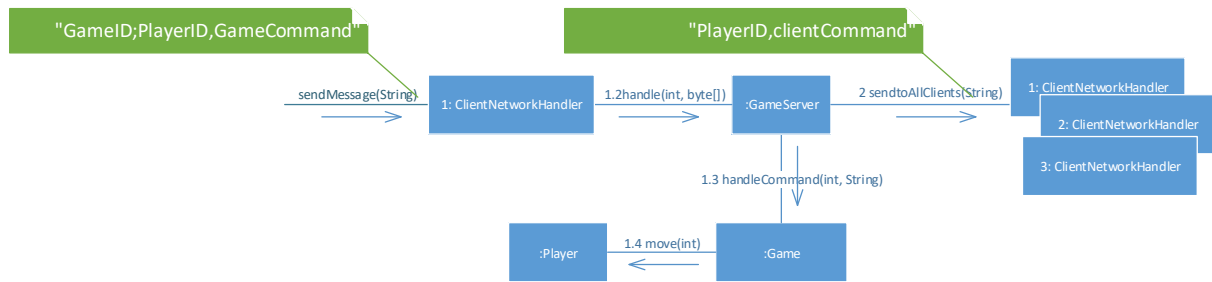


Nachdem der Benutzer die Frageliste-Datei durch den CreateContentPane angegeben hat, wird der Dateipfad an den QuestionListLoader übergeben. Dieser holt mit Hilfe von FileUtils den Inhalt der Datei. Anschliessend erstellt der QuestionListLoader zeilenweise (vom Inhalt der Datei) ein neues Question-Objekt und fügt es dem QuestionList-Objekt hinzu. Nach beenden der Schleife wird das QuestionList-Objekt wieder an den CreateContentPane zurückgegeben.

### 3.3.3. UC4 Spiel spielen



Um das Spiel zu spielen muss der Spieler den Würfel werfen, dies wurde hier in einem eigenen Sequenzdiagramm behandelt. Anschliessend wird das Resultat und somit auch die neue Position der Spielfigur (Piece) dem GUI (GameContentPane) mitgeteilt und gezeichnet. Weiter wird geprüft, ob das neue Feld spezielle Eigenschaften besitzt. Handelt es sich um eine Frage, wird eine Frage versendet, ist es ein Schlangen- oder ein Leiter-Feld wird erneut eine Veränderung der Spielfigur an das GUI (GameContentPane) übermittelt.



Die Kommunikation wird durch den Benutzer ausgelöst, wenn er den Würfel werfen möchte. Anschliessend erfolgt die Interaktionen zwischen dem ClientHandler (der Spieler der würfelt), dem Spieleserver (GameServer). Zum Schluss werden alle Spieler über den neuen Stand informiert.

### 3.4. Designentscheide

Eine grundlegende Entscheidung für dieses Projekt wurde bereits zu Beginn des Arbeitsprozesses gefällt: Die Game-Logik soll sich ausschliesslich auf dem Server befinden und die Clients sollen keinerlei Wissen über das Spiel haben. Es soll lediglich die Anzeige auf den Clients realisiert sein. Sämtliche Informationen über das Spiel sollen vom Server bereitgestellt werden.

Als Konsequenz dieser Entscheidung musste ein Protokoll zur Kommunikation zwischen Client und Server geschaffen werden. Die Entscheidung fiel dabei auf ein simples Protokoll-Muster: Die Befehle sollten aus einem Befehlswort bestehen, gefolgt von Parametern, wenn dies nötig sein sollte. Befehlswort und Parameter werden mit Kommas abgetrennt und ein Strichpunkt deutet das Ende eines Befehles an. Die einzelnen Befehle und ihre Syntax sind in folgender Tabelle dokumentiert:

Befehl	Aktion	Sender -> Empfänger
MovePlayer, playerId, NrOfFields;	Bewegt Spieler mit der ID die Anzahl Felder vorwärts.	Server -> Clients
ShowNumber, playerId, number;	Zeigt gewürfelte Augenzahl des Spielers mit der playerId an.	Server -> Clients
PositiveAnswer, playerId;	Dem Spieler mit der playerId wird angezeigt, dass er die Frage richtig beantwortet hat.	Server -> Clients
NegativeAnswer, playerId;	Dem Spieler mit der playerId wird angezeigt, dass er die Frage falsch beantwortet hat.	Server -> Clients
YourTurn, playerId;	Befehl an den Client, dass er den Würfelbutton freigeben soll.	Server -> Clients
Question, question;	Frage, welche vom Server an den entsprechenden Client gesendet wird.	Server -> Clients
ExtendedMode;	Hintergrund auswechseln, sodass Fragezeichen angezeigt werden.	Server -> Clients
Finished, playerId;	Nachricht an alle Clients, dass Spiel beendet ist.	Server -> Clients
YouAre, playerId	Nachricht an einen Client, dass entschieden werden kann, welcher Spieler zu diesem Client gehört.	Server -> Clients
GameReady	Befehl an alle Clients, dass Spiel bereit ist und der Wartedialog geschlossen werden kann.	Server -> Clients
NewTurn;	Nachricht des Clients an den Server, dass er seinen Spielzug beginnen möchte.	Client -> Server
AnswerToQuestion, answer;	Nachricht des Clients an den Server mit der Antwort auf eine zuvor gestellte Frage.	Client -> Server

Weiter ist festgelegt worden, dass die Benutzereingaben und somit letztlich die gesendeten und empfangenen Befehle zwischen Client und Server Aktionen in der Game-Logik oder auf den GUI auslösen sollen. Dies bedeutet, dass beim Empfang eines Befehls die entsprechende Aktion ausgelöst werden soll und das Resultat davon allenfalls wieder versendet werden soll.

Zwischen GUI und Client ist dies durch ein Observer-Pattern erreicht worden. Das Game-Panel observiert seinen ClientNetworkHandler, welcher in einem eigenen Thread auf eingehende Befehle wartet. Das Panel wird informiert, wenn ein Befehl eingegangen ist und kann darauf entsprechend reagieren.

Im Gegensatz dazu, besteht zwischen Server und Game-Logik eine Attributabhängigkeit. Dies hat zur Folge, dass der Server aufgrund von eingehenden Befehlen aktiv Methoden der Game-Klasse aufruft und das Resultat dann weiter an die Clients übermittelt. Damit die Kopplung zwischen den beiden Komponenten nicht so gross wird, haben wir entschieden, dass die Game-Klasse den Server nicht kennen soll und die Abhängigkeit nur in eine Richtung bestehen soll. Die Game-Klasse kann den Server also nicht selber anweisen, einen Befehl zu versenden und muss diese Entscheidung der Server-Klasse überlassen. Rückblickend würden wir mit dem heutigen Wissenstand eine andere Entscheidung treffen. Die Konsequenzen und eine alternative Lösung werden im Anhang kurz geschildert.



## 4. Implementation

### 4.1. Testbericht

- ⊗ = Test schlug fehl.  
 ✓ = Test war erfolgreich.  
 ⚠ = Es gibt Abweichung zur ursprünglichen Definition

#### 4.1.1. Spiel erstellen (Standard Spiel-modus)

✓	<b>Spiel erstellen (Standard Spiel-Modus)</b>
<b>Ausgangssituation</b>	Es wurde noch kein Spiel erstellt, die Applikation wurde gestartet.
<b>Ereignis</b>	Der Administrator versucht, ein Spiel im Standard Spiel-Modus zu erstellen.
<b>Erwartetes Ergebnis</b>	Das Spiel steht zum Verbinden bereit.
<b>Schlussendliches Ergebnis</b>	Das Spiel konnte erfolgreich erstellt werden und stand zur Verbindung bereit.

#### 4.1.2. Spiel erstellen (Erweiterter Spiel-Modus)

✓	<b>Spiel erstellen (Erweiterter Spiel-Modus)</b>
<b>Ausgangssituation</b>	Es wurde noch kein Spiel erstellt, die Applikation wurde gestartet.
<b>Ereignis</b>	Der Administrator versucht, ein Spiel im erweiterten Spiel-Modus zu erstellen.
<b>Erwartetes Ergebnis</b>	Das Spiel steht zum Verbinden bereit.
<b>Schlussendliches Ergebnis</b>	Das Spiel konnte erfolgreich erstellt werden und stand zur Verbindung bereit.

#### 4.1.3. Frageliste einlesen

✓	<b>Frageliste einlesen</b>
<b>Ausgangssituation</b>	Applikation wurde gestartet und ist in der Spiel-Konfiguration.
<b>Ereignis</b>	Der Benutzer hat die Möglichkeit, eine Datei aus seinem System auszuwählen, um sie als Frageliste für das Spiel zu verwenden.
<b>Erwartetes Ergebnis</b>	Frageliste wurde in den Server geladen und wird für das nächste Spiel verwendet.
<b>Schlussendliches Ergebnis</b>	Frageliste wurde erfolgreich in den Server geladen und wurde für das Spiel angewendet.

#### 4.1.4. Spiel beitreten

✓	<b>Spiel beitreten</b>
<b>Ausgangssituation</b>	Es wurde ein Spiel vom Administrator gestartet, der Spiel-Modus ist dabei irrelevant. Der Benutzer hat die Applikation gestartet und es hat noch freie Plätze im Spiel.
<b>Ereignis</b>	Benutzer verbindet sich mittels IP-Adresse zum Server.
<b>Erwartetes Ergebnis</b>	Benutzer ist erfolgreich dem Spiel beigetreten.
<b>Schlussendliches Ergebnis</b>	Der Benutzer konnte erfolgreich einem Spiel vom Administrator beitreten.

#### 4.1.5. Würfeln

✓	<b>Würfeln</b>
<b>Ausgangssituation</b>	Das Spiel, mit mindestens einem Benutzer, ist gestartet.
<b>Ereignis</b>	Benutzer betätigt den Würfel.
<b>Erwartetes Ergebnis</b>	Die Spielfigur des Benutzers wird mit der Anzahl gewürfelter Punkte nach vorne bewegt.
<b>Schlussendliches Ergebnis</b>	Die Figur wurde erfolgreich mit der Anzahl gewürfelter Punkte nach vorne bewegt.

#### 4.1.6. Fragefeld-Aktion

✓	<b>Fragefeld-Aktion</b>
<b>Ausgangssituation</b>	Das Spiel ist im Gange und ein Benutzer muss würfeln.
<b>Ereignis</b>	Benutzer landet auf dem Fragefeld.
<b>Erwartetes Ergebnis</b>	Dem Benutzer wird eine zufällige Frage aus der Frageliste gestellt und der Benutzer muss die Frage beantworten.
<b>Schlussendliches Ergebnis</b>	Die Frage wurde dem Benutzer gestellt und die Antwort wird richtig vom Server validiert.

#### 4.1.7. Schlangenfeld-Aktion

✓	<b>Schlangenfeld-Aktion</b>
<b>Ausgangssituation</b>	Das Spiel ist im Gange und ein Benutzer muss würfeln.
<b>Ereignis</b>	Benutzer landet auf einem Feld mit einem Schlangenkopf.
<b>Erwartetes Ergebnis</b>	Der Benutzer fährt mit seiner Figur zum Feld, bei dem sich das Ende der Schlange befindet.
<b>Schlussendliches Ergebnis</b>	Beim Erreichen eines Felder auf dem sich der Kopf der Schlange befindet wurde die Figur des Benutzers erfolgreich zum entsprechenden Ende dieser Schlange bewegt.

#### 4.1.8. Leiterfeld-Aktion

✓	<b>Leiterfeld-Aktion</b>
<b>Ausgangssituation</b>	Das Spiel ist im Gange und ein Benutzer muss würfeln.
<b>Ereignis</b>	Benutzer landet auf einem Feld mit einem Leiteranfang.
<b>Erwartetes Ergebnis</b>	Der Benutzer fährt mit seiner Figur zum Feld, bei dem sich das Ende der Leiter befindet.
<b>Schlussendliches Ergebnis</b>	Die Figur des Benutzers wurde beim Erreichen eines Feldes mit dem Beginn einer Leiter erfolgreich die Leiter hoch bewegt.

#### 4.1.9. Spiel gewinnen

✓	<b>Spiel gewinnen</b>
<b>Ausgangssituation</b>	Das Spiel ist im Gange und ein Benutzer ist auf dem letzten Spielfeld angekommen.
<b>Ereignis</b>	Der Benutzer muss eine Frage aus der Frageliste beantworten.
<b>Erwartetes Ergebnis</b>	Falls der Benutzer die Frage richtig beantwortet, hat er das Spiel gewonnen. Ansonsten geht er wieder zu seinem vorherigen Feld zurück.
<b>Schlussendliches Ergebnis</b>	Der Benutzer, der auf dem letzten Feld landete, hatte nach dem Beantworten der Frage das Spiel gewonnen.

## 4.2. Installationsanleitung

Um Snakes and Ladders spielen zu können, kann die **SnakesAndLadders.jar** als Java-Appikation gestartet werden.

Es ist auch möglich unter Windows die **SnakesAndLadders.exe** zu starten.

Beide Varianten setzen aber voraus, dass eine Java-Version von 8 oder höher auf dem Zielsystem installiert ist. Die Version 8 von Java kann hier heruntergeladen werden:

<https://www.java.com/de/download/win10.jsp>

## 4.3. Code Dokumentation

Die Code Dokumentation befindet sich im Projektverzeichnis *snakesandladder/doc* und kann mit der Datei *index.html* im Browser geöffnet werden.

## 5. Resultat

### 5.1. Erreichte Ziele

Die Applikation erfüllt alle zu Beginn definierten Funktionen. Anwender können ein neues Spiel starten oder treten einem bestehenden Spiel bei. Beim Erstellen eines Spiels kann zwischen zwei Varianten gewählt werden, der Standard Spiel-Modus, welchen wir von früher alle kennen, oder den erweiterter Spiel-Modus mit benutzerspezifischen Fragen.

Für die Frageliste kann eine Vorlage heruntergeladen werden, welche nach Bearbeitung durch den Spielersteller in das Spiel eingelesen werden kann.

Anschliessend steht nichts mehr im Wege das Spiel zu spielen, die Fragen zu beantworten und die Leitern bzw. die Schlangen hoch und runter zu gehen.

### 5.2. Mögliche Weiterentwicklungen

Folgende Möglichkeiten für die Weiterentwicklung wären denkbar.

- Eine Statistikfunktion, welche die Siege der Spieler und die Häufigkeit der richtig beantworteten Fragen aufzeigt. Dies natürlich in Abhängigkeit der Frageliste die verwendet wurde. Dadurch können auch Lernfortschritte nachverfolgt werden.
- Eine Erweiterung der Frageliste und deren Darstellung gegenüber dem Spieler sollen die Möglichkeit geben, zum Beispiel Bilder oder Musikstücke als Fragen einzubauen. So könnte zum Beispiel auf Französisch ein Text vorgespielt werden, welcher einen Suchbegriff umschreibt.
- Hinweise zu Fragen, welche dem Spieler als Hilfe dienen sollten, wären eine weitere Option.
- Detaillierte Konfigurationen sollen dem Ersteller des Spiels die Möglichkeit geben, zu bestimmen, was genau bei einer Leiter oder nach einer Frage geschehen soll. Soll der Spieler einfach zurückgesetzt werden, soll er an den Startpunkt gehen oder soll ein anderer Mitspieler die Möglichkeit erhalten, die Frage richtig zu beantworten. Dies wären ein paar Ideen, wie das Spielverhalten verändert werden könnte.

## 6. Anhang

### 6.1. Bedienungsanleitung

#### 6.1.1. Menü

1 Bevor das Spiel gespielt werden kann, muss es durch den «Administrator» über das Menü «Spiel erstellen» (*siehe Kapitel 3 Spiel Umgebung starten*) eingerichtet werden.

2 Sobald das Spiel eingerichtet wurde und bereit ist, können Spieler über das Menü «Spiel beitreten» (*siehe Kapitel 4 Spiel beitreten*) dem Spiel beitreten. Wenn alle Spieler dem Spiel beigetreten sind, wird das Spiel automatisch gestartet.

Das Menü lässt sich über «Snakes and Ladders.exe» öffnen.



## 6.1.2. Spiel Umgebung starten

### Standard-Modus (ohne Fragefelder):

Sobald über das Menü auf «Spiel erstellen» geklickt wird, erscheint ein neues Fenster (rechts abgebildet).

- 1 Für die Identifikation des Spiels muss der Name eingetippt werden. Der Name kann frei gewählt werden.
- 2 Anschliessend kann die Spieleranzahl ausgewählt werden. Maximal sind 8 Spieler möglich.

### Erweiterter Spiel-Modus (mit Fragefelder):

- 1 Für den erweiterten Spiel-Modus muss zusätzlich der Haken bei «Erweiterter Spiel-Modus» gesetzt werden.
- 2 Um eine Frageliste zu erstellen, kann über den Knopf «Vorlage herunterladen» eine bereits vorbereitete Liste heruntergeladen und mit Fragen und Antworten ergänzt werden (siehe Beispiel rechts).
- 3 Danach muss die Frageliste über den Knopf «Eigene Frageliste importieren» importiert werden. Es öffnet sich ein Fenster per Klick, worin die Frageliste ausgewählt werden kann.
- 4 Der Pfad der importierten Frageliste wird im Textfeld angezeigt.

	A	B	C
1	Frage	Antwort	Hinweis
2	Wie heisst die Hauptstadt der Schweiz	Bern	
3	Wann ist der Nationalfeiertag der Schweiz?	01.08	
4	Wie viele Amtssprachen hat die Schweiz ?	4	
5	Wie viele Bundesräte gibt es?	7	

Beispiel einer Liste von Fragen.

1

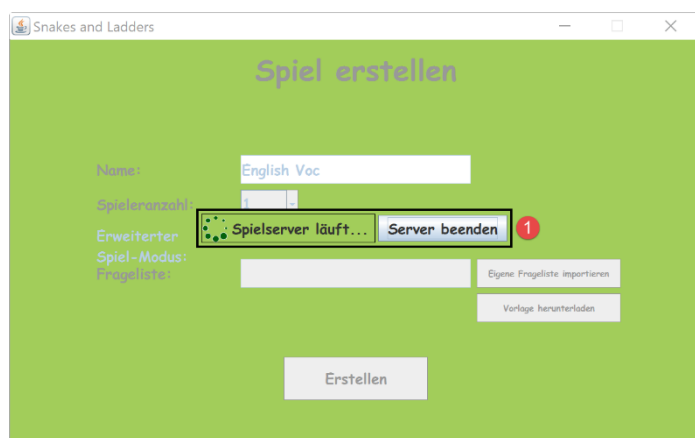
Wenn alle benötigten Daten gefüllt wurden, kann das Spiel mit dem Knopf «Erstellen» bereitgestellt werden.



1

Das Spiel kann mit dem Knopf «Server beenden» beendet werden und die Spieler können jederzeit das Spiel verlassen.

Wenn ein Spieler das Spiel verlässt und der Spielserver noch läuft, ist es nicht mehr möglich dem Spiel wieder beizutreten und weiter zu spielen.



### 6.1.3. Spiel beitreten

Sobald über das Menü auf «Spiel beitreten» geklickt wird, erscheint ein neues Fenster (rechts abgebildet).

- 1 Als erstes muss auf das bereitgestellte Spiel in der Liste geklickt werden.
- 2 Anschliessend muss auf den Knopf «Beitreten» geklickt werden, um dem Spiel beizutreten.

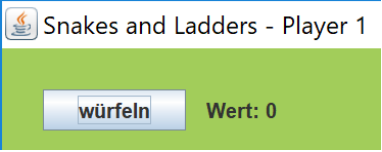
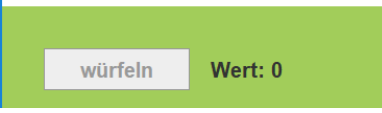


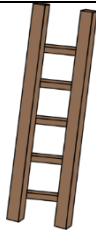


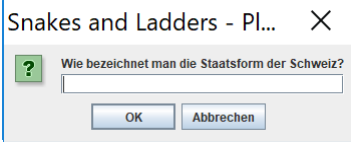
Erst nachdem alle Spieler beigetreten sind, kann das Spiel gespielt werden.





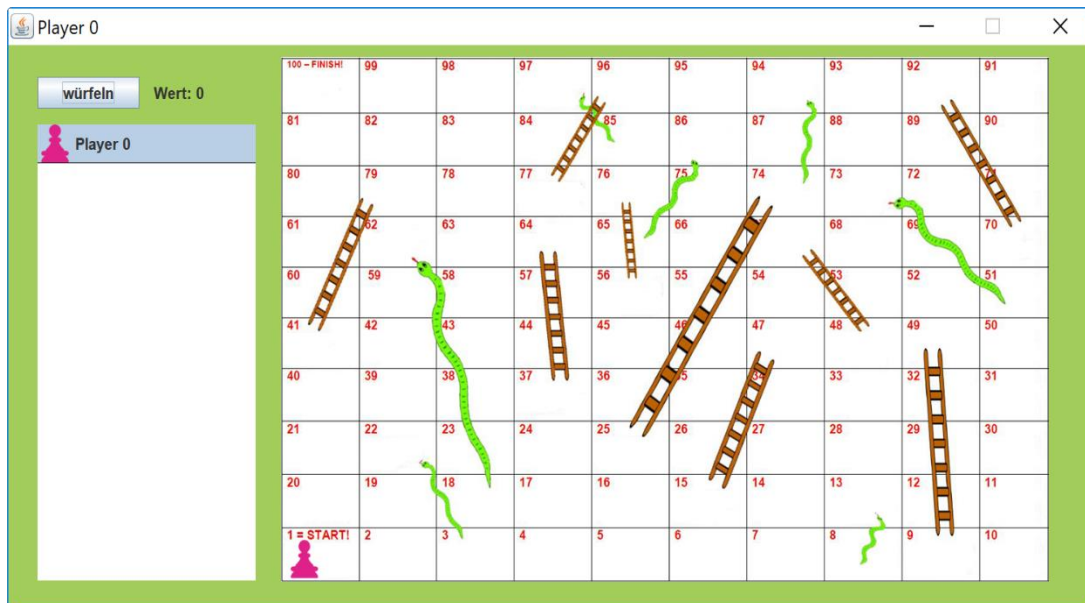
#### 6.1.4. Spielablauf

<p>Das Spiel zeigt an, welcher Spieler beginnt, in dem der Knopf „würfeln“ ausgewählt werden kann. Durch diesen kann der Spieler würfeln und seine Spielfigur fährt die entsprechende Augenzahl des Würfels auf dem Spielfeld. Nach dem Würfeln ist der nächste Spieler am Zug und kann diesen Schritt ebenfalls tun.</p> <p>Kommt nun ein Spieler auf ein spezielles Feld (Leiter-, Schlange- oder beim erweitertem Modus auf ein Fragefeld, siehe <i>Kapitel 3. Spiel Umgebung starten</i>) so folgen die nachträglich umschriebenen Aktionen.</p>	<p><b>Spieler 1 kann würfeln.</b></p>  <p><b>Es ist ein anderer Spieler am Zug.</b></p> 
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

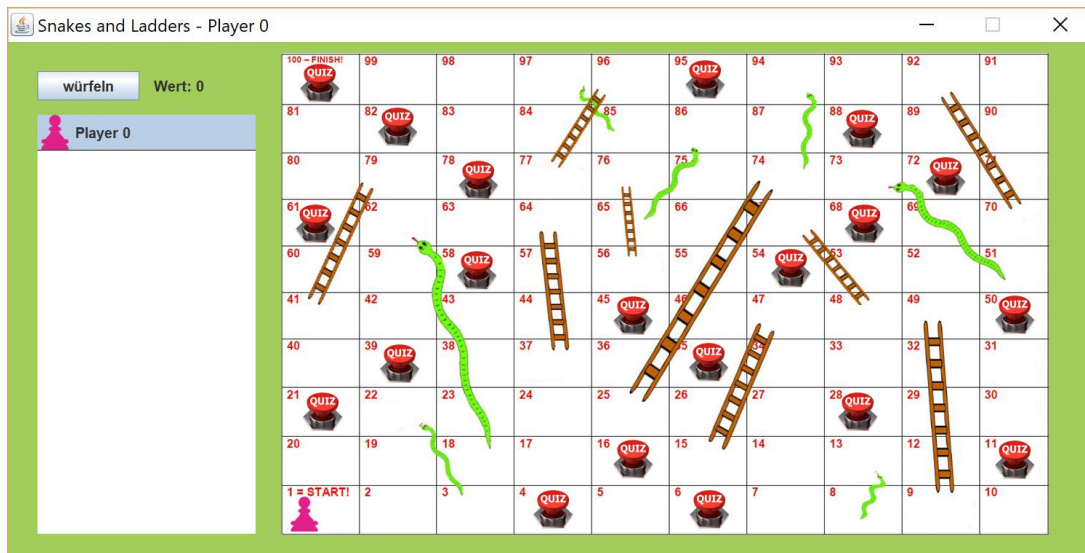
<p><b>Leiterfeld:</b></p> <p>Glückwunsch, Sie haben gut gewürfelt. Sie stehen am Fusse einer Leiter und können diese nun nach oben gehen. Ihre Spielfigur wird automatisch an das Ende (Kopf) der Leiter gesetzt.</p> <p>Die Leiter kann nur von unten nach oben benutzt werden.</p>	
<p><b>Schlangenfild:</b></p> <p>„AUTSCH“, die Schlange hat Sie erwischt. Sie rutschen die Schlange hinunter und landen auf dem Feld, auf dem sich das Ende der Schlange befindet. Sie spielen nun vom Ende der Schlange weiter aber achten Sie darauf, dass die Schlange Sie nicht noch einmal erwischt.</p> <p>Die Schlange ist das Gegenstück der Leiter und kann nur von oben nach unten benutzt werden.</p>	
<p><b>Fragefeld:</b></p> <p>Hier ist Ihr Wissen gefragt. Können Sie die Frage richtig beantworten, können Sie auf dem Feld verweilen. Ist die Antwort falsch müssen Sie zurück auf Ihr vorheriges Feld.</p> <p>(Fragefelder sind nur im erweiterten Spiel-Modus vorhanden.)</p>	  <p>Beispielfrage</p>

Alle weiteren Felder sind neutral und haben keine speziellen Eigenschaften. Auf diesen können Sie sich ausruhen und das Spiel geniessen.

Sobald ein Zug vom Spieler beendet wurde, kommt automatisch der nächste Spieler zum Zug und dieser darf anschliessend auch würfeln.



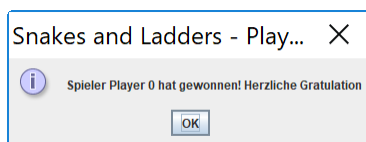
Spielfeld (Standard Spiel-Modus)



Spielfeld (Erweiterter Spiel-Modus)

### 6.1.5. Spielende

Das Spiel wird beendet, sobald ein Spieler das Zielfeld (Feldnummer 100) exakt erreicht hat. Beim erweiterten Spiel-Modus muss beim letzten Spielfeld die Frage noch korrekt beantwortet werden.



Wenn der Spieler beispielsweise bei Feld 99 die 6 würfelt, so wird er auf das Feld 95 positioniert, da man zu hoch gewürfelt hat. Das bedeutet, dass der Spieler die genaue Zahl würfeln muss, um auf das letzte Feld zu kommen.

## 6.2. Projektmanagement

### 6.2.1. Grobplanung

Iteration	Semester	Woche	Start	Ende	Anz Tage	Milestone	Ziele, Release	Aufwand geplant [h]	Aufwand aus Iter # [h]	Aufwand effektiv [h]	geplant vs effektiv
			25.09.2017	02.10.2017	7	1	Inception Phase	20	22.5	24.3	21.25%
1	2		25.09.2017	02.10.2017	7	1	- Projektthema, Idee, Funktionalität, Umfang im Projektteam definieren sodass allen die Vision klar ist.	20	22.5	24.3	21.25%
			03.10.2017	23.10.2017	20	2	Elaboration Phase	110	105.0	106.8	-2.95%
2	3,4		03.10.2017	13.10.2017	10	2	Absprache und klare definition des Ablaufs des Spiels (Anwendungsfälle)	55	50.0	55.8	1.36%
3	4,5		13.10.2017	23.10.2017	10	2	Kommunikation zwischen Client und Server herstellen Spielregeln definieren	55	55.0	51.0	-7.27%
			23.10.2017	20.11.2017	28	3	Construction Phase	160	158.0	128.5	-19.69%
4	6,7		23.10.2017	06.11.2017	14	3	- Klassendiagramme für die Logik, Status, den Befehlen und des GUI erstellen - Erstellen der Klassen und Strukturen gemäss Klassendiagrammen	80	74.0	63.0	-21.25%
5	8,9		06.11.2017	20.11.2017	14	3	- Implementierung der Logik, Befehlen und fertigstellung des GUI in den vorbereiteten Klassen.	80	84.0	65.5	-18.13%
			21.11.2017	18.12.2017	27	4	Transiton Phase	150	178.5	193.3	28.83%
6	10, 11		21.11.2017	03.12.2017	12	4	- Tests mit einer Testgruppe durchführen - Erkenntnisse aus Testgruppe einfließen lassen - BETA-Version erstellen	75	88.5	96.5	28.67%
7	12,13		03.12.2017	18.12.2017	15	4	- Schlussbericht verfassen (Projektidee, Analyse, Design, Implementation, Resultat und Anhang) - Abschluss Projekt	75	90.0	96.8	29.00%
(Pro Kopf 18h - 2 Tage) Reserve:								90			
Total:								530	464	452.8	-14.58%
Vorgaben:								500-600h			

Die letzten zwei Iterationen wurden nun umgesetzt und verlangten nochmals einiges ab. Denn durch das ausgiebige Testen wurden nochmals Fehler aufgedeckt und auch Fehlverhalten festgestellt, welche viel Zeit für deren Analyse und auch Behebung in Anspruch genommen haben. Sei es die Animation der Spielfigur, die nicht immer funktionierte oder die Netzwerkkommunikation, welche die Befehle nicht immer korrekt versendete. Weitere Details befinden sich in den anschliessenden Iterationsplänen.

Nun ist das Projekt vorbei und wir haben zu Beginn einen Aufwand von 530 Stunden geplant, davon wurden 464 Stunden als Aufwände in den einzelnen Iterationen geschätzt und schlussendlich davon 452.8 Stunden in Anspruch genommen. Dies bedeutet, dass wir 14.58% unter dem geplanten Aufwand geblieben sind. Die nicht genutzte Zeit ging hauptsächlich im 2. und 3. Meilenstein verloren, da das gesamte Team sich zuerst mit den Iterationen und den Diagrammen einstimmen musste. Auch wenn es nun den Anschein hat, die Zeit nicht komplett genutzt zu haben, sind die geplanten Reserven von 90h zu berücksichtigen. So sind ohne Reserven 440h geplant gewesen, dazu kommen 24h, welche wir für die Umsetzung dieses Prototyps zusätzlich benötigten.

Die Ziele der beiden Iterationen wurden erreicht, nur die Erstellung der BETA-Version wurde um eine Iteration verschoben, da das Testing und die daraus entstandenen Tasks noch nicht komplett implementiert werden konnten.

### 6.2.2. Risikoliste

Nr.	Name	Beschreibung	Wahrsch.	Schaden	Stufe	Prio	Massnahmen	Verantwortlich
1	Firewall	Server-Client-Kommunikation funktioniert nicht via Firewall	20%	Spiel ist nicht funktionsfähig und nicht brauchbar	M		2 Berechtigung im Netzwerk für das Spiel vergeben	Entwicklungsteam
2	Effizienz	Geforderte Antwortzeiten können nicht erreicht werden.	10%	Spiel ist nur bedingt spielbar	M		2 Mögliche Verbesserung der Netzwerkkommunikation in betracht ziehen --> Prototyp entwickeln	Entwicklungsteam
3	Konkurrenz	Es erscheint ein Produkt auf dem Markt welches unser Produkt obsolet aussehen lässt.	10%	Spiel lässt sich nicht mehr oder nur schlecht verkaufen	H		3 Projektskizze überarbeiten und Massgebende unterschiede definieren.	PL
4	Personell	Es fällt ein Mitarbeiter des Teams aus.	30%	Wissen über das Projekt und Ideen gehen verloren.	L		3 Verteilung der Aufgaben auf andere Entwickler, Reduktion des Projektumfanges.	PL
5	Netzwerk-kommunikation	Die Netzwerkpakete kommen nicht korrekt an.	40%	Befehle können nicht verstanden werden und sind somit unbrauchbar.	H		1 Mit Buffern bzw genügend grossen Netzwerkpakete arbeiten.	Entwicklungsteam

Bei der Risikoliste ist ein neues Risiko hinzugekommen (Nr. 5), da beim Testen der Applikation festgestellt wurde, dass gewisse Zeichen der Befehle nicht richtig oder ganz versendet wurden. Durch die Massnahme, siehe Massnahme bei Nr. 5, konnten wir den Fehler vorerst nicht mehr erzeugen jedoch können wir ihn nicht ganz ausschliessen. Auch Testen ist bei einer Kommunikation über das Netzwerk schwierig oder hier schlecht möglich. Ansonsten gibt es nur kleinere Herabstufungen der Wahrscheinlichkeiten der anderen Risiken.

## 6.2.3. Vergangene Iterationen

### 6.2.3.1. Iterationsplan #6

Task	Arbeitspaket	Aufwand geplant [h]	Aufwand effektiv [h]	Verantwortlich	Erledigt ?
1	Serverliste abfragen und anzeigen	3	4	Milan	x
2	GUI für wartenden Server auf Spieler erstellen	3	2	Milan	x
3	Msg wer als nächstes würfeln darf versenden -	2	2	Philipp	x
4	QuestionList vorbereiten (z.B. English)	1	0.5	Severin	x
5	Testing Questionlist	2	1.5	Dominik	x
6	Dok für Bedienungsanleitung entwerfen und beginnen (StartUI, ServerConfig, QuestionList)	3	4	Severin	x
7	Testing der Befehle, evt ergänzen und verbessern.	5	6.5	Dominik	x
8	Frage beantworten Handling ergänzen und fertig erstellen	3	3.5	Jan / Philipp	x
9	Handling des letzten Feld einbauen - definieren	1	2.5	Philipp	x
10	Umstellung Standard und Erweiterter Modus im GUI implementieren	1	0.5	Jan	x
11	Spielernamen im Client-Window anzeigen lassen	0.5	0.5	Jan	x
12	Testing Server, Thread-Handling und Messages	5	7	Jan	x
13	Testing Client, Animation / Messages / Interaction	5	8	Jan / Philipp	x
14	Testing UI, Ablauf, Verhalten der Fehler, Status, Informationen	5	6	Alle	x
15	Schlussdokumentation erstellen, Kapitel zuweisen, Übersicht verschaffen und bereits Kapitel ergänzen.	5	5	Severin	x
16	Properties-Fiel für Befehle erstellen	3	2.5	Dominik	x
17	Sicherstellen das min. Anz. Player = 1 ist	2	1	Philipp	x
18	Logging & Exception-handling konzept entwerfen und umsetzen	3	2.5	Jan	x
19	Refactoring / Clean-Code	8	8	alle	x
20	Code-Dokumentation (Wichtige Funktionen)	8	5	alle	x
21	Projektmanagement	5	7	Philipp	x
22	Meeting / Aussprache / Diskussion	15	17	alle	x
23					
24					
Total:		88.5	96.5		

Die letzten Befehle und Funktionalitäten wurden eingebaut und getestet. Bei den Tests stellten wir folgende 3 Fehler oder falsches Verhalten fest, welche neben ein paar anderen Kleinigkeiten sehr viel Zeit für die Analyse und deren Behebung in Anspruch genommen haben.

1. Netzwerk-Befehl kommt nicht immer vollständig und korrekt an.
2. Animation funktioniert nicht immer korrekt.
3. Serverliste aus dem Netzwerk auslesen (Broadcasting) fehlt.

Diese Fehler wurden im Team besprochen und es wurde begonnen, sie zu beheben. Jedoch wurden alle 3 Task am Iterationsende nicht fertig umgesetzt und somit in die nächste Iteration weitergezogen.

Ansonsten hat das Testen mehr Zeit in Anspruch genommen als vorgängig geplant. Dies und die Behebung der Fehler führten hauptsächlich dazu, dass wir über den geplanten Aufwand von 75 Stunden gekommen sind.

### 6.2.3.2. Iterationsplan #7

Ta	Arbeitspaket	Aufwand geplant [h]	Aufwand effektiv [h]	Verantwortl. h	Erledigt ?
1	Animation für Spielbewegung implementieren	3	3	Philipp	x
2	Bedienungsanleitung beginnen	4	3	Severin	x
3	Fehler Behebung – Lost Server Message	4	5	Jan	x
4	Multicast Serverping für Serverliste	4	5	Dominik	-
5	Verbesserung der Animation	4	3	Philipp / Dominik	x
6	Refactoring – Nicht gebrauchte Funktionen und Felder entfernen.	3	2.5	Philipp / Milan	x
7	Testing GUI und allenfalls Verbesserungen gleich einbauen.	5	4.5	Philipp	x
8	Junit-Testing Game	4	4	Dominik	x
9	UC für Schlussbericht verbessern und einbauen	5	4	Alle	x
10	Punkte der Projektidee für Schlussbericht zusammenfassen	1	1	Severin	x
11	Resultate des Domänenmodells und der Anwendungsfälle, efit. Problemanalysen dokumentieren und in den Schlussbericht einfügen	2	2.5	Dominik	x
12	Architektur nochmals überarbeitet und dokumentiert, im Schlussbericht eingefügt	3	2	Jan	x
13	Server sauber beenden (Alle Threads schliessen)	1	2	Jan	x
14	Server läuft – Information an Benutzer ausgeben	1	0.75	Philipp	x
15	Client sauber beenden (Alle Threads schliessen)	1	1.5	Jan	x
16	Schlussbericht Erweiterungen schreiben	1	1.5	Severin	x
17	Schlussbericht Pers Erkenntnisse / Fazit	3	3.5	Alle	x
18	Refactoring / Cleancode / Javadoc	10	14	Alle	x
19	Schlussbericht zusammenfügen, Struktur	2	3	Severin	x
20	Benutzerhandbuch Server erstellen	2	1.5	Milan	x
21	Benutzerhandbuch Spiel beitreten	2	1.5	Milan	x
22	Benutzerhandbuch Spiel handling	2	1	Philipp	x
23	Rechtschreibkorrektur Benutzerhandbuch	1	1	Milan / Philipp	x
24	Rechtschreibkorrektur Schlussbericht	2	1.5	Milan / Philipp	x
25	Projektmanagement	5	6.5	Philipp	x
26	Schlusspräsentation und Demo vorbereiten	2	2	Philipp	x
27	Abschluss des Projekts, Code-ZIP erstellen, ausführbare Datei erzeugen, das ganze auf OLAT ablegen.	1	1	Alle	
28	Meeting / Aussprache / Diskussion	12	15	alle	x
<b>Total:</b>		90	96.75		

Die Behebung der Fehler, welche in der letzten Iteration festgestellt wurden, war nicht ganz so einfach und nicht immer erfolgreich.

1. Netzwerk-Befehl kommt nicht immer vollständig und korrekt an.  
→ Hier wurde sichergestellt, dass der Buffer in der Netzwerkkommunikation immer geflutet wird, sprich alle Daten sind versendet und der Buffer ist leer. Dies hatte zur Folge, dass der Fehler bei unserer Kommunikation nicht mehr reproduziert werden konnte, jedoch können wir ihn nicht ganz ausschliessen da es sehr schwierig ist, dies zu Testen.
2. Animation funktioniert nicht immer korrekt.  
→ Hierfür wurde ein eigener Pool mit Threads angelegt, welcher der Reihe nach die Animationen abarbeitet. Dadurch können Animationsaufträge, auch wenn bereits neue eintreffen, gespeichert und sicher dargestellt werden. Somit konnte der Fehler einer «verloren gegangenen Animation» behoben werden.
3. Serverliste aus dem Netzwerk auslesen (Broadcasting).  
→ Beim mehrmaligen Ausprobieren eines UDP Multicast Client-Server Konstruktes in verschiedenen Netzwerken und in einem eigenständigen Projekt konnte kein positives Ergebnis erzielt werden. Aus diesem Grund entschied man sich für eine statische Variante der Serverliste. Sprich die IP des Servers muss manuell eingegeben werden.

#### 6.2.4. Feedback als Projektteam

Am Anfang war es neu für uns, auch die Art und Weise des Vorgehens von UP zu verstehen und zu verwenden. Dies brauchte Zeit, doch als grosse Hilfe empfanden wir die Grobplanung und auch die einzelnen Iterationspläne, denn diese halfen uns jederzeit die Übersicht zu behalten.

Der Start mit der eigentlichen Programmierung im Meilenstein 3 würden wir im Nachhinein vorziehen bzw. parallel zur Dokumentation und der Ausarbeitung der UML-Diagramme machen. Denn bis anhin wurden zuerst die Diagramme erstellt, besprochen und dann umgesetzt, welches einfach mehr Zeit in Anspruch nahm. Damit ist nicht der Zeitaufwand gemeint, sondern die Tage welche vergingen, bis alle Projektmitglieder auf dem gleichen Stand waren.

Zum Schluss können wir nur sagen, dass wir sehr zufrieden sind. Wir waren ein tolles Team, welches gut zusammenarbeitete. Jeder konnte sein Wissen einbringen und seine Fähigkeiten gezielt einsetzen. Auch delegierte Verantwortungen wurden wahrgenommen und umgesetzt, so wurde durch den Projektleiter zum Beispiel bei den Schlussdokumentationen und der Benutzeranleitung einen Verantwortlichen definiert, der wiederum die Aufgabe hatte, die Kapitel oder benötigte Informationen bei den anderen Mitgliedern im Team einzufordern oder zuzuweisen. Dies funktionierte super und entlastete den Projektleiter diesbezüglich gut.



### 6.3. Anmerkung zum Design der Server-Architektur

Während der Entwicklung des Projekts wurde festgestellt, dass die Schnittstelle zwischen Gameserver als Netzwerkkomponente und der Game-Logik nicht optimal designt worden war. Dies äussert sich beispielsweise darin, dass der Server in der Methode `handleTurn()` grosse Teile eines Spielzuges behandelt.

Beim Design des Servers haben wir entschieden, dass die Game-Klasse seinen Server nicht als Attribut kennen soll. Denn der Server hat bereits das Spiel als Attribut und eine zu enge Kopplung der beiden Klassen wollten wir verhindern. Da die Netzwerkschicht durch das Versenden und Empfangen von Befehlen den Takt des Spieles vorgibt, wurde entschieden, dass der Server aufgrund von erhaltenen Befehlen aktiv Aktionen in der Game-Logik anstossen soll.

Es kann nun im Spiel passieren, dass der Benutzer nach dem Würfeln eine Frage beantworten muss. Es müssen also manchmal mehrere Befehle und Antworten innerhalb eines Zuges versendet werden, bevor ein weiterer Zug eingeleitet werden kann. Da in diesem Design der Server jeweils die entsprechenden Methoden der Game-Klasse aufruft, entscheidet der Server in welcher Phase eines Zuges sich ein Spieler befindet.

Idealerweise müsste diese Vermischung von Netzwerk-Aufgaben und spielspezifischen Aufgaben des Servers aufgelöst werden. Allerdings haben wir erst im Verlaufe des Projektes die nötigen Designpattern kennengelernt und keine Zeit mehr gehabt, diese auf unser Server-Design anzuwenden. Im Team haben wir die Möglichkeit diskutiert, diese Beziehung durch ein Observer-Pattern zu verbessern und so die Verantwortlichkeiten besser aufzuteilen. Mit dem heutigen Wissensstand würden wir diese Option favorisieren, aus zeitlichen Gründen ist diese Verbesserung jedoch nicht mehr vorgenommen worden.



## 6.4. Literaturverzeichnis

Bundesamt für Statistik (2017): *Bildung und Wissenschaft*.

URL: <https://www.bfs.admin.ch/bfs/de/home/statistiken/bildung-wissenschaft.html>

[Stand: Dezember 11.12.2017]

o. A. (o. J.): *Snakes and Ladders spielen*.

URL: <https://de.wikihow.com/Snakes-and-Ladders-spielen> [Stand: 26.09.2017].

o. A. (o. J.): o. T.

URL: <http://www.agame.com/game/snakes-and-ladders> [Stand: 26.09.2017].

o. A. (o. J.): o. T.

URL: <http://www.counton.org/games/virtualmathfest/snakesladders.html> [Stand: 26.09.2017].

o. A. (o. J.): o. T.

URL: <http://www.playonlinedicegames.com/snakesandladders> [Stand: 26.09.2017].

## 6.5. Glossar

Begriff	Definition und Information	Weitere Erklärungen	Aliase, auch in anderen Sprachen	Beziehungen zu anderen Elementen
<b>Administrator</b>	Der Benutzer der Server-Applikation welche das Spiel konfigurieren kann.	Ist von einem Administrator die Rede so kann dieser sich nur auf der Server-Applikation befinden.	Admin, Host	Spieler
<b>Spieler</b>	Der Benutzer der Client-Applikation welche das Spiel spielt.	Ist von einem Spieler die Rede so kann dieser sich nur auf der Client-Applikation befinden.	Player, User	Administrator
<b>Server-Applikation</b>	Ist ein Programm, welches ein «Snakes and Ladder» Spiel verarbeitet und laufend den einzelnen Spielern ihre nächsten Schritte anzeigt.			
<b>GUI</b>	Die Grafische Benutzeroberfläche ist eine Form der Benutzerschnittstelle (UI), mithilfe von grafischen Symbolen und Steuerelementen wird die Anwendungssoftware dargestellt.	Eingaben auf dem GUI könnten mittels Touch (falls vorhanden), per Maus oder über die Tastatur erfolgen.	Graphical user interface, Benutzeroberfläche	UI
<b>UI</b>	Die Benutzerschnittstelle ist die Stelle oder Handlung, mit der ein Mensch mit einer Maschine in Kontakt tritt.	Mensch ↔ Mensch-Maschine-Schnittstelle ↔ Maschine	Benutzerschnittstelle / Nutzerschnittstelle	GUI
<b>Piece</b>	Englische Bezeichnung für eine Figur, hier stellt sie die Spielfigur welches auf dem Spielfeld angezeigt wird dar.	Im Spiel Snakes and Ladders wird die Schachfigur «Bauer» verwendet und für die Spieler in unterschiedlichen Farben dargestellt.	Figur / Spielfigur	
<b>Ladder</b>	Spezielles Feld auf dem Spielfeld. Landet ein Spieler am unteren Ende einer Leiter bewegt er sich ans obere Ende der Leiter und überspringt somit einige Felder.	Bestandteil des originalen Snakes & Ladders Brettspiel.	Leiter / Leiterfeld	Jumpfield

<b>Snake</b>	Spezielles Feld auf dem Spielfeld. Landet ein Spieler auf dem Kopf einer Schlange bewegt er sich nach unten, ans Ende der Schlange.	Bestandteil des originalen Snakes & Ladders Brettspiel.	Schlange / Schlängelfeld	Jumpfield
<b>Jumpfield</b>	Oberbegriff für Schlangen- und Leiternfelder, da sich beide Arten auf die gleiche Art implementieren liessen.		Sprungfeld	Ladder Snake
<b>Questionfield</b>	Spezielles Feld auf dem Spielfeld. Landet ein Spieler auf einem solchen Feld, wird ihm eine Frage gestellt. Bei korrekter Antwort darf er auf dem Feld verweilen, ansonsten muss er zurück auf sein ursprüngliches Feld.	Erweiterung des originalen Snakes & Ladders Brettspiel.	Fragefeld	
<b>Erweiterter Modus</b>	Option beim Erstellen eines neuen Spiels. Ist der erweiterte Modus aktiviert befinden sich Fragefelder auf dem Spielfeld	Erweiterung des originalen Snakes & Ladders Brettspiel die je nach Präferenz aktiviert oder deaktiviert werden kann.		Questionfield
<b>Zielfeld</b>	Der Spieler, der als erstes genau auf diesem Feld landet, gewinnt das Spiel.	Beim originalen Snakes & Ladders Brettspiel jeweils das Feld 100	letzte Feld, goal	