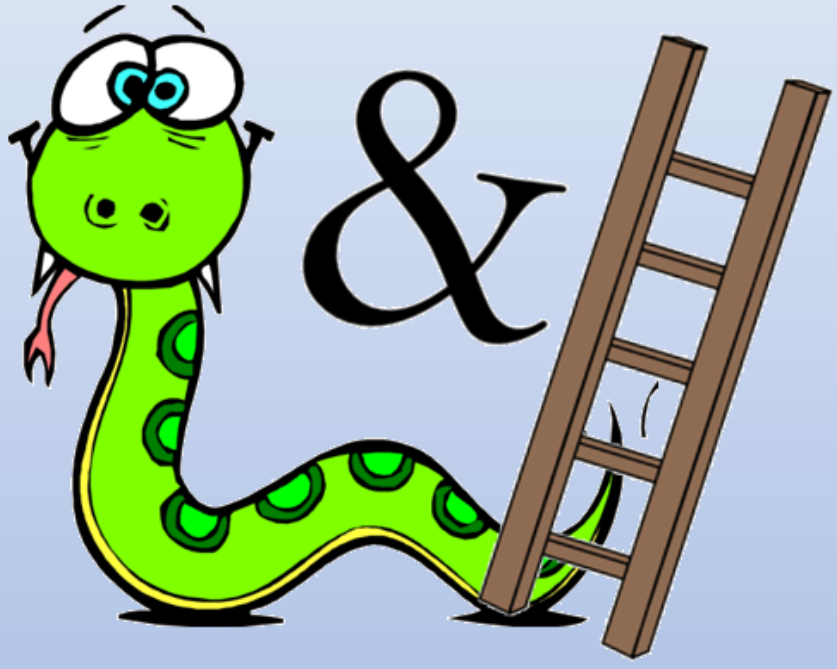
PSIT3 – Snakes and Ladders

Design-Resultate



Namen:	Milan Natkeeran Severin Roost Jan Sonderegger Dominik Steiner Philipp Wetzel
Klasse:	IT16a_WIN
Gruppe:	3
Schule:	ZHAW
Abgabetermin:	21.11.2017

Inhaltsverzeichnis

Versionierung

Version	Datum	Bearbeitet von	Änderung / Grund
1	07.11.17	Philipp	Erstellung des Dokuments
1.1	17.11.17	Philipp	Sequenz und Kommunikationsdiagramm für UC4 Spiel spielen hinzugefügt.
1.2	17.11.17	Philipp	Glossar erweitert.
1.3	19.11.17	Severin	Sequenz und Kommunikationsdiagramm für UC4.1 Spiel spielen - Würfeln hinzugefügt.
1.4	19.11.17	Severin	Klassenverantwortlichkeiten
1.5	19.11.17	Dominik	Klassenverantwortlichkeiten und Diagramme für UC3
1.6	19.11.17	Milan	Design-Klassendiagramm und GUI-Design hinzugefügt
1.7	20.11.17	Milan	Sequenz und Kommunikationsdiagramm für UC1 Spiel Umgebung starten hinzugefügt und Design-Klassendiagramm ergänzt
1.8	20.11.17	Jan	Hinzufügen der Beschreibung für die Architektur und der Diagramme für UC2
1.9	20.11.17	Severin	Anpassung und Ergänzungen zu den Klassenverantwortlichkeiten
2.0	20.11.17	Philipp	Übersichtsdiagramm für die Sequenzdiagramme der UC's hinzugefügt

Tabelle 1: Versionierung

1. Projektmanagement

1.1. 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**	***99.0***	***0.0***	-100.00%
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	*72.0*	*0.0*	-100.00%
7	12,13	03.12.2017	18.12.2017	15	4	- Schlussbericht verfassen (Projektidee, Analyse, Design, Implementation, Resultat und Anhang) - Abschluss Projekt	75	*27.0*	*0.0*	-100.00%

1.2. Risikoliste

Nr.	Name	Bschreibung	Wahrsch.	Schaden	Stufe	Prio	Massnahmen	Verantwortlich
1	Firewall	Server-Client-Kommunikation funktioniert nicht via Firewall	20%	Spiel ist nicht funktionsfähig und nicht brauchbar	M	1	Berechtigung im Netzwerk für das Spiel vergeben	Entwicklungs-team
2	Effizienz	Geforderte Antwortzeiten können nicht erreicht werden.	10%	Spiel ist nur bedingt Spielbar	M	1	Mögliche Verbesserung der Netzwerkkommunikation in betracht ziehen --> Protoyp entwickeln	Entwicklungs-team
3	Konkurenz	Es erscheint ein Produkt auf dem Markt welches unser Produkt obsolet ausehen lässt.	30%	Spiel lässt sich nicht mehr oder nur schlecht verkaufen	H	2	Projektskizze überarbeiten und Massgebende unterschiede deffinieren.	PL
4	Personell	Es fällt ein Mitarbeiter des Teams aus.	50%	Wissen über das Projekt und Ideen gehen verlohren.	L	3	Verteilung der Aufgaben auf andere Entwickler, Reduktion des Projektumfanges.	PL

Nr 1 Bei der Entwicklung und Implementierung der ersten Spielbefehle welche über das Netzwerk versendet wurden, tauchten bis heute keinerlei Schwierigkeiten mit der Firewall auf, daher wurde die Wahrscheinlichkeit dieses Risikos von 40% auf 20% reduziert. Jedoch kann es nicht auf 0% reduziert werden da mögliche Probleme in der Weiterentwicklung auftauchen könnten. Ebenfalls könnten Veränderungen an der Firewall (Updates oder andere Konfigurationen) Probleme hervorrufen.

Nr 2 Wie bereits beim Risiko Nr. 1 Firewall umschrieben konnten bis heute keinerlei Probleme mit der Netzwerkkommunikation festgestellt werden. Die Kommunikation war jederzeit stabil und wurde ohne Verzögerung versendet bzw. zugestellt. Daher wurde auch hier das Risiko von 30% auf 10% reduziert.

Nr 3 Ob ein Produkt auf dem Markt gebracht wird welches zu unserem Spiel eine Konkurrenz bietet, können wir auch in der nächsten Zeit nicht ausschliessen, daher bleibt die Wahrscheinlichkeit hier gleich wie beim Meilenstein 2.

Nr 4 In den vergangenen 2 Iterationen hatten wir mehrere, krankheitsbedingte Ausfälle von Mitarbeitern. Aus diesem Grund und der aktuellen Jahreszeit mit der höheren Gefahr durch eine Grippen Infektion wurde dieses Risiko drastisch von 10% auf 50% erhöht. Da in einem kleinen Projektteam ein Ausfall eine bedeutende Mehrbelastung für alle anderen im Team bedeutet wurde sie auf 50% gesetzt.

1.3. Vergangene Iterationen

1.3.1. Iterationsplan #4

Task	Arbeitspaket	Aufwand geplant [h]	Aufwand effektiv [h]	Verantwortlich	Erledigt ?
1	Klassendiagramm für Logik-Aktionen erstellen	6	4	Jan (Dominik)	x
2	Klassendiagramm für Status erstellen	6	3	Milan	x
3	Klassendiagramm für Befehle erstellen	6	3	Dominik (Severin)	x
4	Klassendiagramm für GUI erstellen	6	5	Philipp (Milan)	x
5	GUI Architektur entwerfen	2	3	Philipp	x
6	SnakesAndLaddersUI	2	3	Philipp	x
7	GameCreateUI	2	2	Milan	x
8	GameUI	6	8	Philipp	x
9	QuestionUI	2	2	Philipp	x
10	Klasse Game	2	2	Dominik	x
11	Klasse QuestionList	2	1.5	Dominik	x
12	Klasse Player	2	1	Dominik	x
13	Klasse Gamefield	2	1	Severin	x
14	Klasse Field	2	1	Severin	x
15	Klasse Jumpfield	2	1	Severin	x
16	Klasse Questionfield	2	1	Severin	x
17	Klasse Server + Handler / Listener	5	3.5	Jan	x
18	Klasse Cleint + Handler / Listener	5	3	Jan	x
19	Projektmanagement	4	3	Philipp	x
20	Meeting / Aussprache / Diskussion	8	12	alle	x
21					
22					
23					
	Total:	74	63		

1.3.2. Iterationsplan #5

Task	Arbeitspaket	Aufwand geplant [h]	Aufwand effektiv [h]	Verantwortlich	Erledigt ?
1	Architektur entscheide beschreiben und begründen	4	3	Jan	x
2	Design-Klassendiagramm umschreiben	4	4	Milan	x
3	GUI-Design umschreiben	1	1	Milan	x
4	Klassendiagramm gemäss Java-Code erstellen	2	2	Philipp	x
5	Klassenverantwortlichkeiten Packages Game, util und gui)	2	1.5	Severin	x
6	Klassenverantwortlichkeiten Packages Network (Client, Server)	2	1	Dominik	x
7	Interaktionsdiagramme	4	2.5	Severin	x
8	Glossar weiter ausbauen	2	1.5	alle	x
9	Verbindung GUI / Handler	5	3.5	Jan	x
10	Verbindung Command / Game	3	2.5	Dominik	x
11	Commands	3	1	Dominik	x
12	GUI Ansichtenwechsel Main - Create Server	2	2	Milan	x
13	GUI Ansichtenwechsel Main - Game	2	2	Philipp	x
14	Server starten	2	2	Milan	x
15	Client starten	4	2	Philipp	x
16	GameCreateUI erweitern	2	2	Milan	x
17	GameConnectUI	3	3	Milan	x
18	Würfelfunktion implementieren + an Client übermitteln	3	2	Jan	x
19	Spielbewegung implementieren GUI	4	3	Philipp	x
20	Spielbewegung versenden	1	1	Philipp	x
21	Refactoring Code, literale entfernen	8	5	alle	x
22	Präsentation vorbereiten	1	1	alle	x
23	Projektmanagement	8	7	Philipp	x
24	Meeting / Aussprache / Diskussion	12	10	alle	x
25					
26					
27					
28					
	Total:	84	65.5		

1.4. Nächste Iterationen

1.4.1. Iterationsplan #6

Task	Arbeitspaket	Aufwand geplant [h]	Aufwand effektiv [h]	Verantwortlich	Erledigt ?
1	Serverliste abfragen und anzeigen	3		Milan	
2	GUI für wartenden Server auf Spieler erstellen	3		Milan	
3	Msg wer als nächstes würfeln darf versenden - einbauen	2		Philipp	
4	QuestionList vorbereiten (z.B. English)	1		Severin	
5	Testing Questionlist	2		Severin	
6	Dok für Bediehnungsanleitung entwerfen und beginnen (StartUI, ServerConfig, QuestionList)	3		Severin	
7	Testing Commands	5		Dominik	
8	Testing Server	5		Jan	
9	Testing Client	5		Milan	
10	Testing UI	5		Severin	
11	Play Test-Game	1		Alle	
12	Properties-Fiel für Befehle erstellen	3		Dominik	
13	Sicherstellen das min. Anz. Player = 1 ist	2		Philipp	
14	Logging & Exception-handling konzpet entwerfen und umsetzen	3		Jan	
15	Refactoring / Clean-Code	8		alle	
16	Code-Dokumentation (Wichtige Funktionen)	8		alle	
17	Projektmanagement	5		Philipp	
18	Meeting / Aussprache / Diskussion	8		alle	
19					
20					
21					
	Total:	72	0		

2. 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.

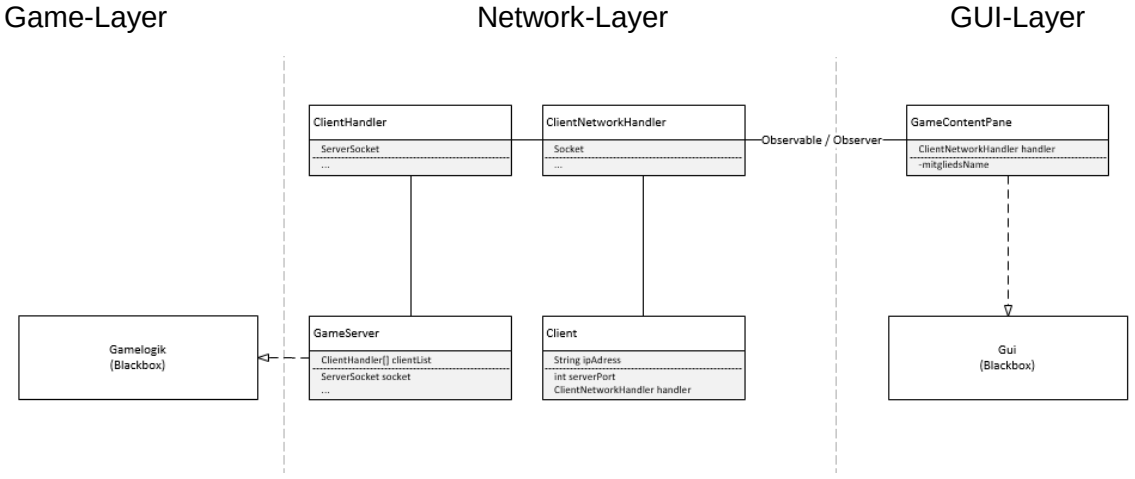
2.1. Umsetzung auf dem Server

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).

2.2. 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.

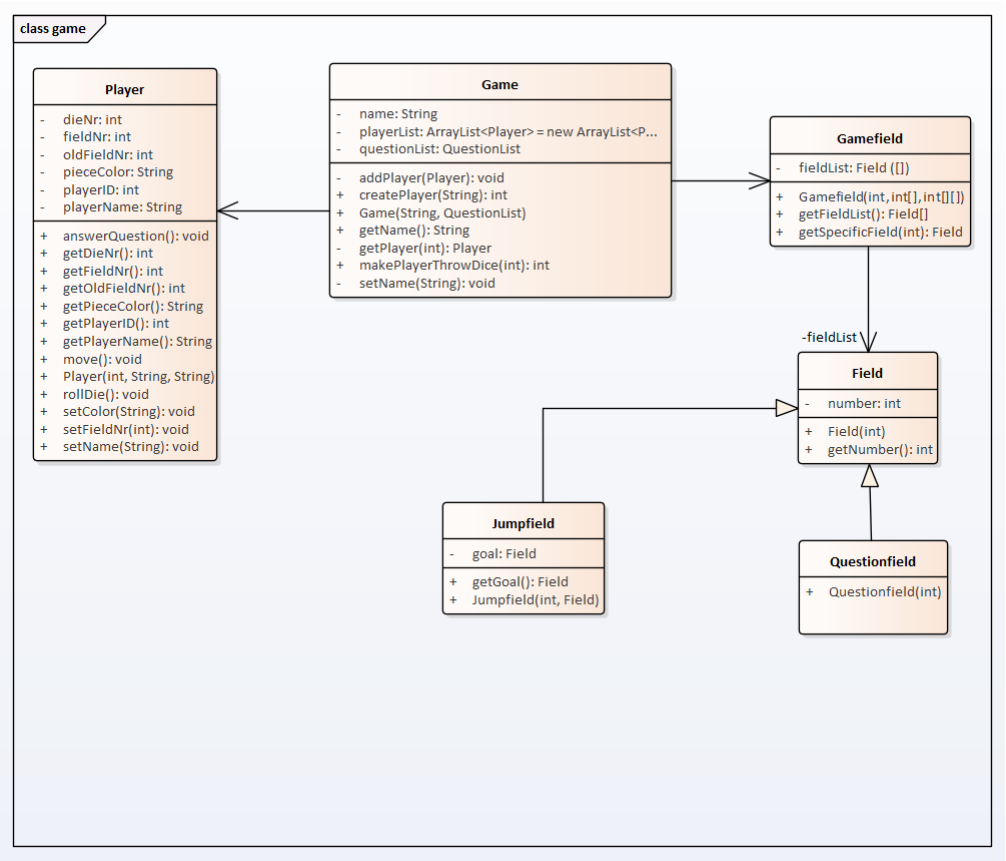


Die Grafik soll nicht die Klassen in ihrer endgültigen Form beschreiben, sondern lediglich die Abgrenzung zwischen Netzwerk-, Game-, und Gui-Logik verdeutlichen. Dabei laufen jeweils die Klassen GameServer, ClientHandler und ClientNetworkHandler in einem eigenen Thread.

3. Design-Klassendiagramm

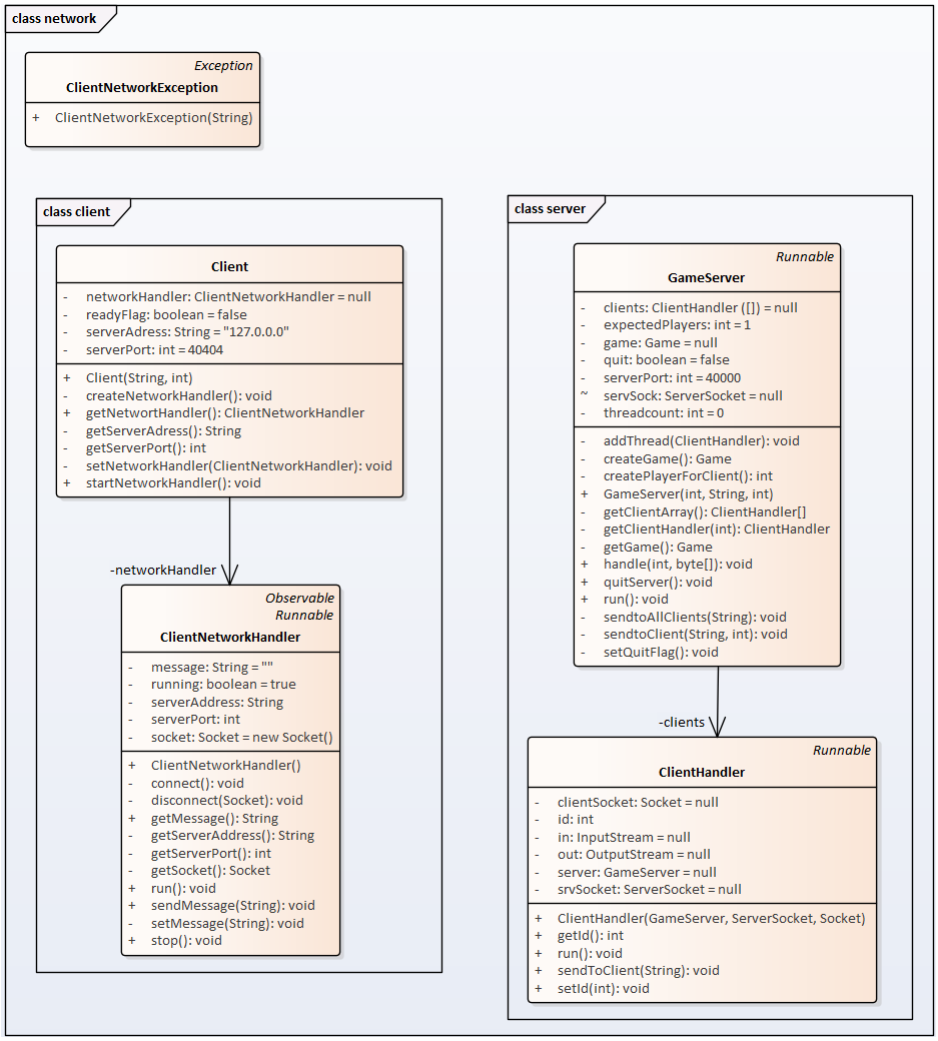
Für Snakes and Ladders wurden die vier Haupt-Packages Game, Network, Questionlist und GUI für die jeweiligen Klassen erstellt, zusätzlich haben wir noch das Package Util für die allgemeinen Einstellungen.

3.1. Package Game:



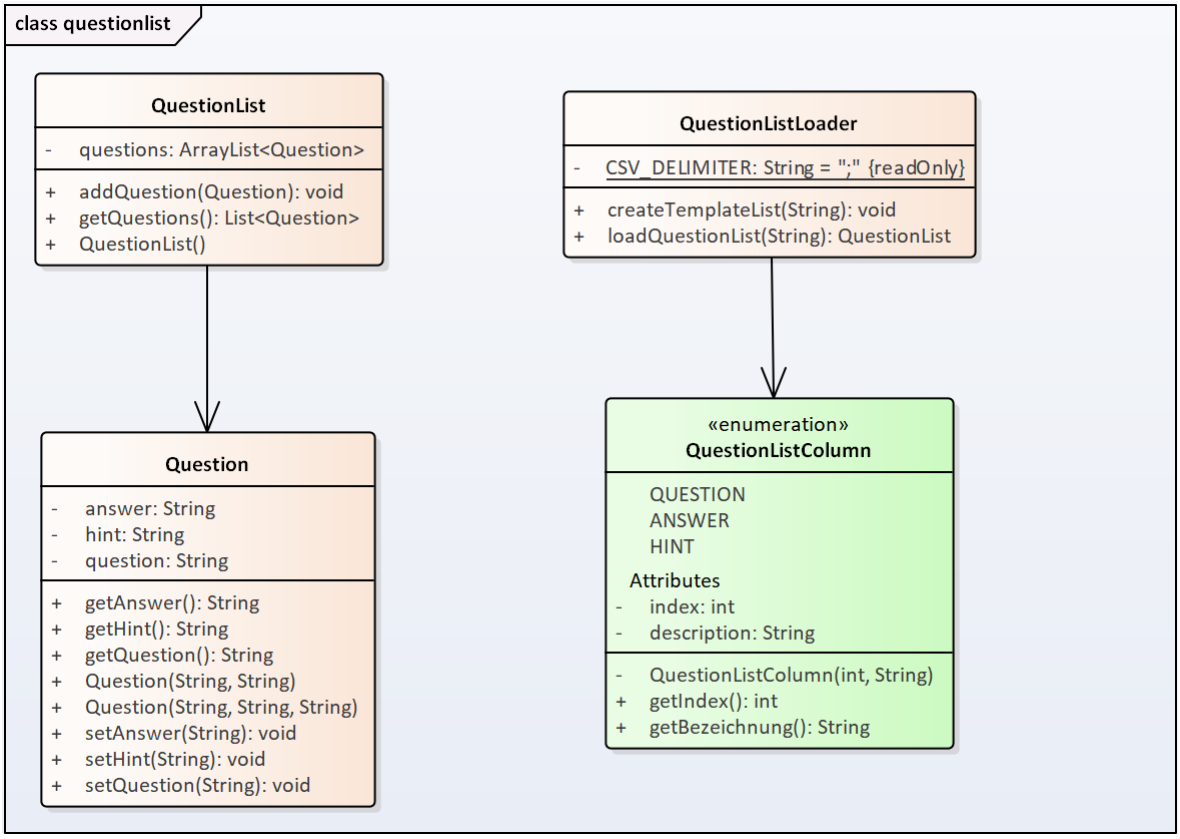
Im Package Game wurden die Klassen Player, Game, Gamefield, Field, Jumpfield und Questionfield erstellt. Die Klasse Game hat Zugriff auf die Klassen Player und Gamefield, damit das Spiel diese zwei Haupt-Objekte darstellen kann. Die Klasse Gamefield besteht aus der Klasse Field, welches entweder ein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

3.2. Package Network:



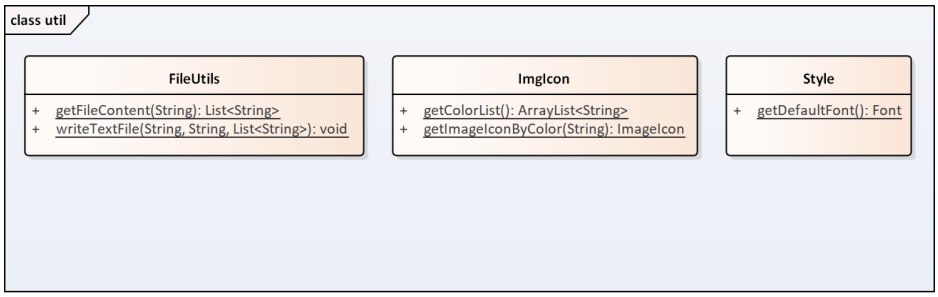
Im Package Network wurden die zwei Unterpackages Client und Server erstellt. Diese zwei Unterpackages beinhalten die Klassen Client, ClientNetworkHandler, GameServer und ClientHandler. Diese sind für die Kommunikation für das Spiel Snakes and Ladders verantwortlich.

3.3. 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.

3.4. Package Util:



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

4. Klassenverantwortlichkeiten

4.1. Package Game

Klasse	Gesamtverantwortlichkeit	Knowing	Doing
Field	Modelliert ein einzelnes Feld auf dem ganzen Spielfeld	Kennt seine Nummer	Kann seine Nummer ausgeben
Game	Kennt den aktuellen Status/Informationen vom Spiel	Kennt Name des Spiels, Frageliste und Spielerliste	Spieler erstellen oder hinzufügen, einen Spieler aus der Liste ausgeben, einen Spieler würfeln lassen, eigenen Name ändern oder ausgeben
Gamefield	Modelliert das komplette Spielfeld, kennt die einzelnen Felder	Kennt alle Felder eines Spielfeldes	Kann alle Felder eines Spielfeldes erzeugen
Jumpfield	Erweiterung eines einfachen Felds	Kennt das Zielfeld des Sprunges	Kann das Zielfeld ausgeben
Player	Hat die aktuellen Information zu einem Spieler	Kennt ID, Name, Farbe seiner Figur, momentanes Feld, ursprüngliches Feld, den letzten gewürfelten Wert	Kann alle Knowing Eigenschaften einzeln ausgeben, sich auf dem Feld bewegen, einen Würfel werfen, eine Frage beantworten
Questionfield	Erweiterung eines einfachen Felds	Weiss, dass es ein Fragefeld ist	-

4.2. Package GUI

Klasse	Gesamtverantwortlichkeit	Knowing	Doing
CreateGameContentPane	Modelliert das GUI, welches ein Admin sieht beim Erstellen eines neuen Spiels	Kennt diverse Parameter zur Darstellung des GUI, Frageliste, Spieleranzahl, Passwort	Kann GUI visualisieren, Frageliste importieren, Passwort und maximale Spieleranzahl definieren und startet den GameServer
GameContentPane	Modelliert das GUI, welches ein User während eines laufenden Spiel sieht	Kennt diverse Parameter zur Darstellung des GUI, insbesondere die Spielernamen, Figuren und deren Position	Kann GUI visualisieren, übermitteln dass ein Spieler würfeln möchte, berechnet die Position der Figur auf dem Spielfeld
JoinGameContentPane	Modelliert das GUI, welches ein User sieht, wenn er einem Spiel beitreten möchte	Kennt diverse Parameter zur Darstellung des GUI, kennt alle bereiten Spiele	Kann GUI visualisieren, verfügbare Spiele suchen, Spiel beitreten
MainContentPane	Modelliert das GUI, welches das Hauptmenü der Applikation zeigt	Kennt diverse Parameter zur Darstellung des GUI	Kann GUI visualisieren, weiterleiten zu Spiel erstellen oder Spiel beitreten
Piece	Enthält die aktuellen Informationen zu einer Figur	Kennt ID und Name des Spielers und deren Spielfigur	Kann Name, ID oder Spielfigur ausgeben
SnakesAndLaddersUI	Modelliert das GUI, welches beim Start der Applikation angezeigt wird	Kennt diverse Parameter zur Darstellung des GUI	Kann GUI visualisieren, weiterleiten zum Hauptmenü

4.3. Package Util

Klasse	Gesamtverantwortlichkeit	Knowing	Doing
FileUtils	Enthält nützliche Funktionen zum Arbeiten mit Files	-	Kann File lesen oder in File schreiben

ImgIcon	Enthält Bilder der Spielfiguren	-	Kann Liste aller möglichen Farben ausgeben, ein Bild der Farbe entsprechend ausgeben
Style	Enthält Default Design	-	Kann Standard Font ausgeben

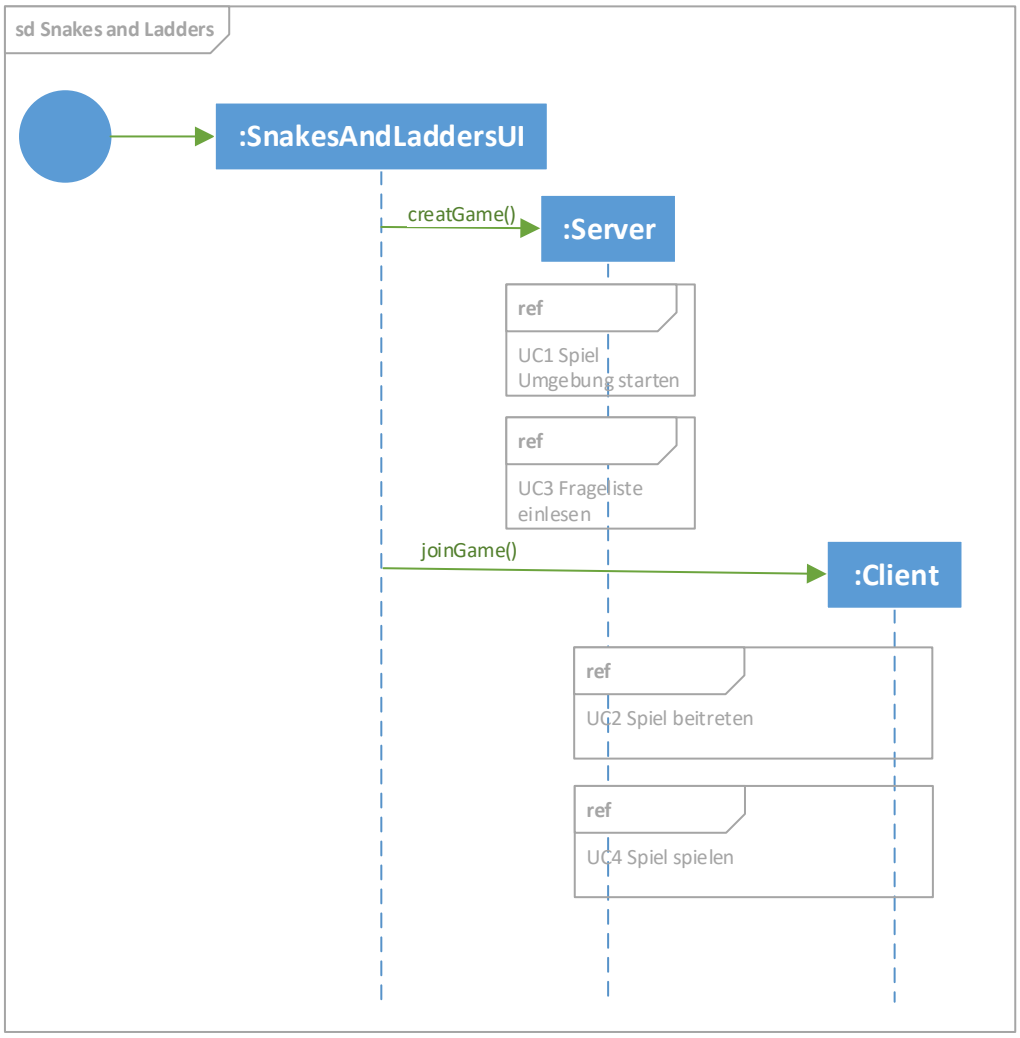
4.4. Package QuestionList

Klasse	Gesamtverantwortlichkeit	Knowing	Doing
Question	Hat den Inhalt einer einzelnen Frage	Kennt Frage, Antwort und Hinweis	Kann Frage, Antwort und Hinweis setzen oder ausgeben
QuestionList	Hat alle notwendigen Informationen zur Frageliste	Kennt alle Fragen einer Frageliste	Kann eine Frage ausgeben oder eine neue hinzufügen
QuestionListColumn	kennt die einzelnen Spalten der Frageliste-Datei	kennt die einzelnen Informationen zu den Spalten der Frageliste	-
QuestionListLoader	ermöglichst das Laden einer Frageliste oder das generieren einer Beispielfrageliste	-	kann bestehende Frageliste einlesen oder eine

4.5. Package Network

Klasse	Gesamtverantwortlichkeit	Knowing	Doing
ClientNetworkHandler	verwaltetet und kennt eine Verbindung zum Server	kennt die Verbindung zum Server	kann vom Server Informationen empfangen und senden
GameServer	verarbeitet den Netzwerk-Input	kennt die einzelnen Verbindungen und andere Netzwerkbezogene Informationen	verwaltet den Server und erledigt andere Netzwerkbezogene Aktionen
ClientHandler	verwaltet eine Verbindung	kennt eine Verbindung	kann von der Netzwerkverbindung lesen und schreiben
ClientNetworkException	ist eine Fehlermeldung	kennt Informationen bezüglich des Fehlers	-

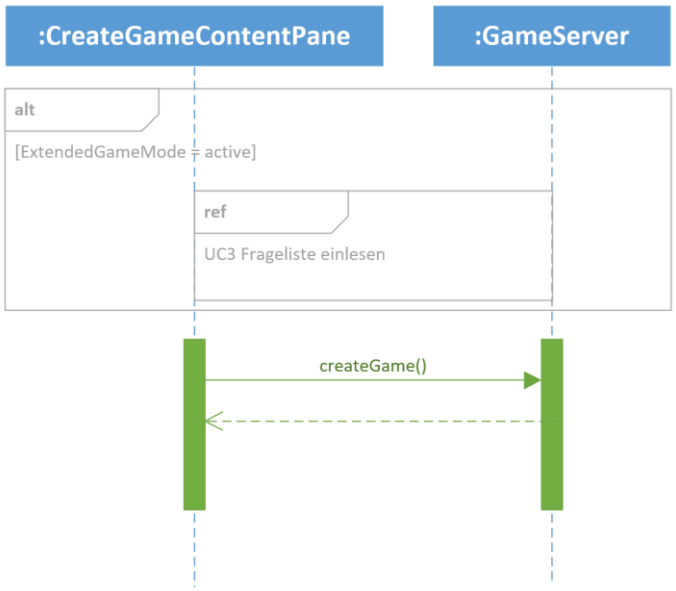
5. Interaktionsdiagramme



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.

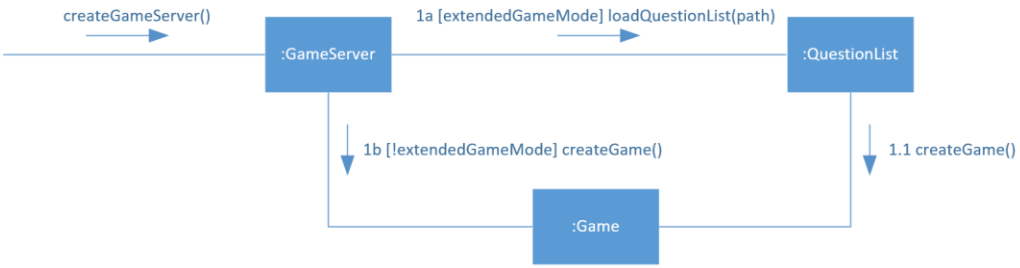
5.1. UC1 Spiel Umgebung starten

5.1.1. Sequenzdiagramm



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.

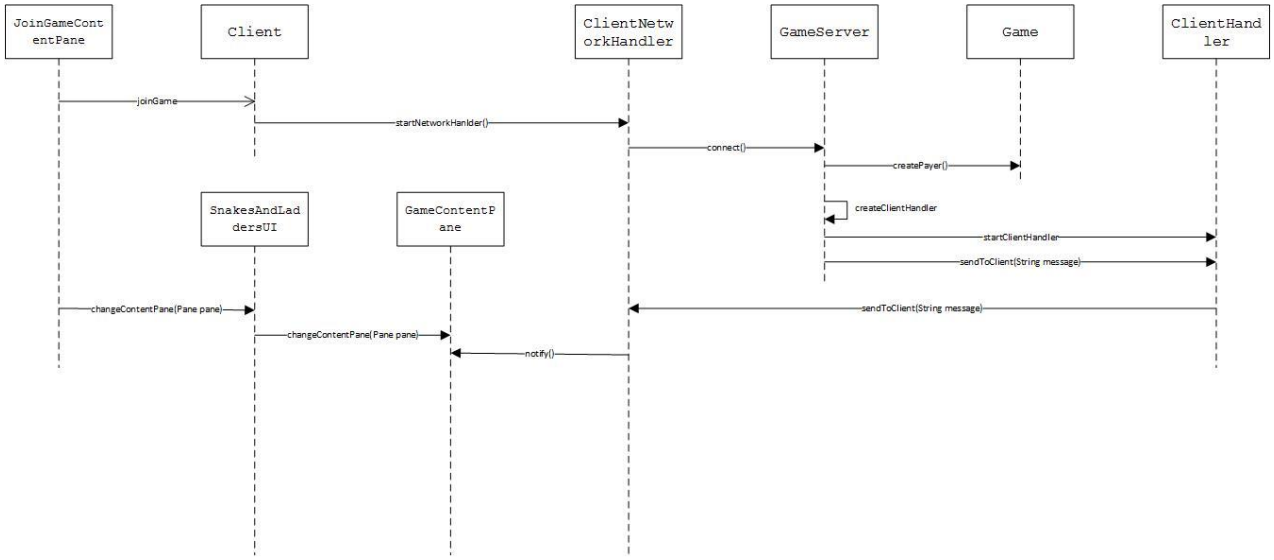
5.1.2. Kommunikationsdiagramm



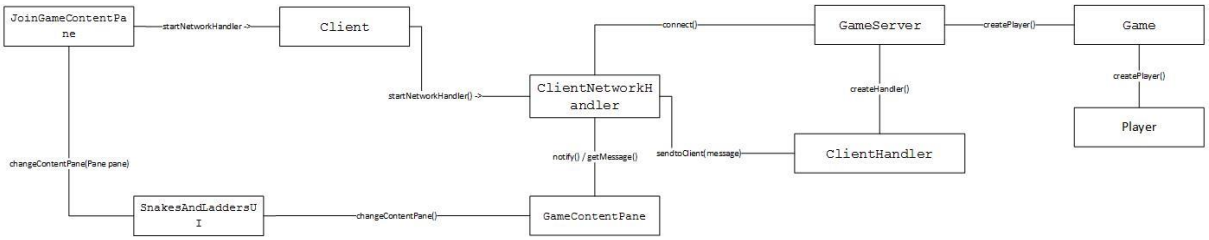
Sobald der Administrator das Spiel erstellt, wird die Kommunikation analog dem Sequenzdiagramm ausgelöst.

5.2. UC2 Spiel beitreten

5.2.1. Sequenzdiagramm

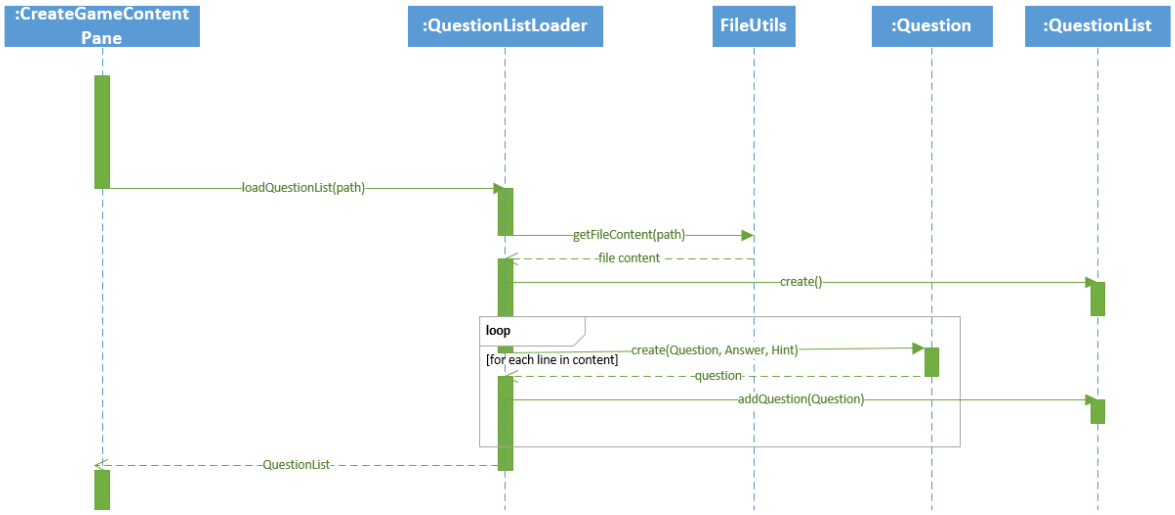


5.2.2. Kommunikationsdiagramm



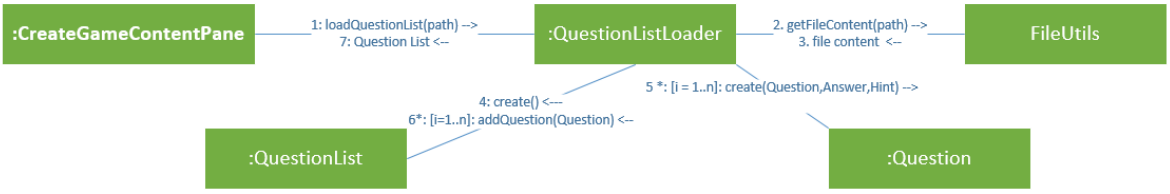
5.3. UC3 Frageliste einlesen

5.3.1. Sequenzdiagramm



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.

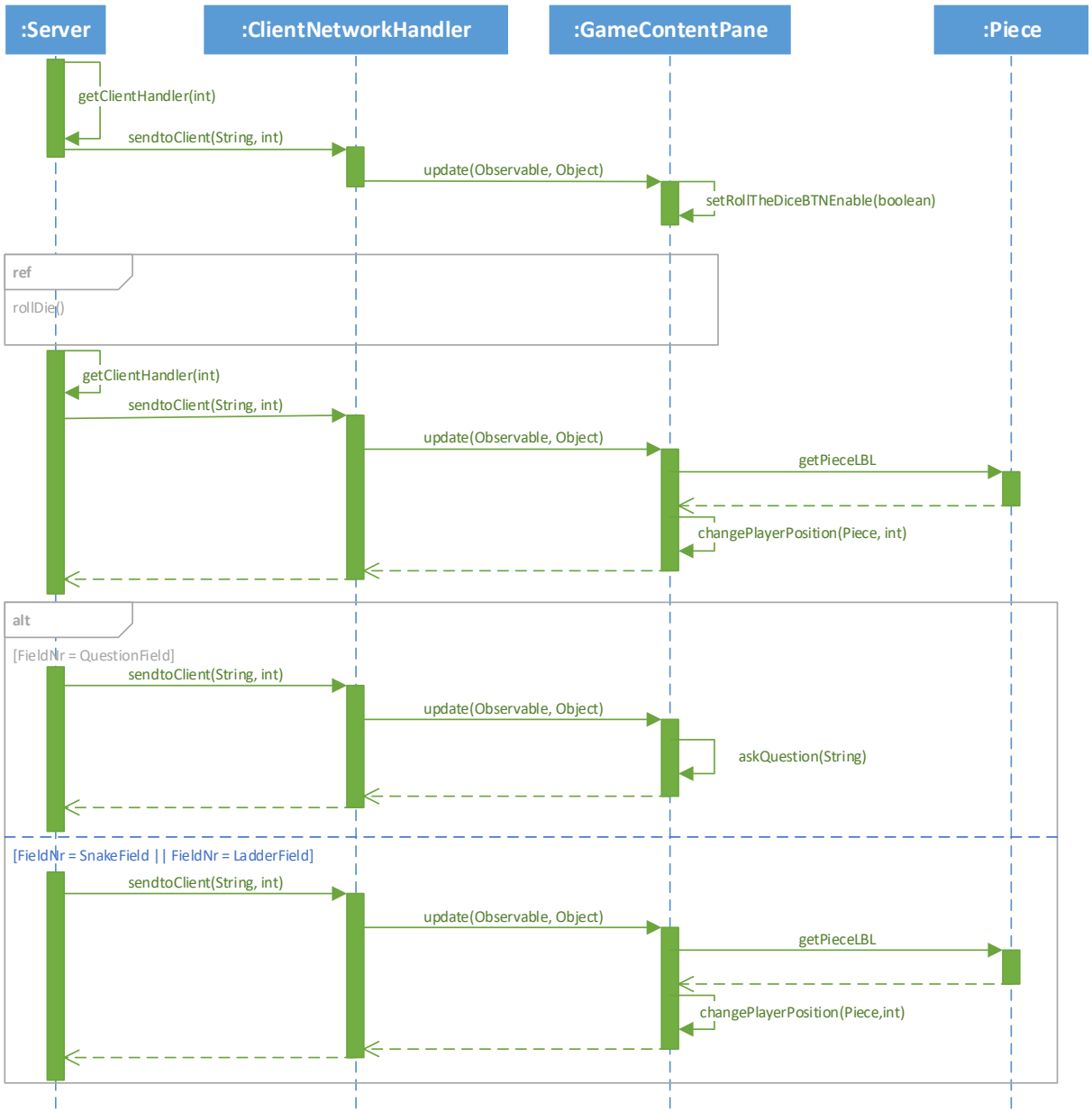
5.3.2. Kommunikationsdiagramm



Die Kommunikation wird durch den Benutzer ausgelöst, wenn er die Datei zum Importieren bestätigt. Der Ablauf der Kommunikation ist ansonsten genau derselbe wie im Sequenzdiagramm.

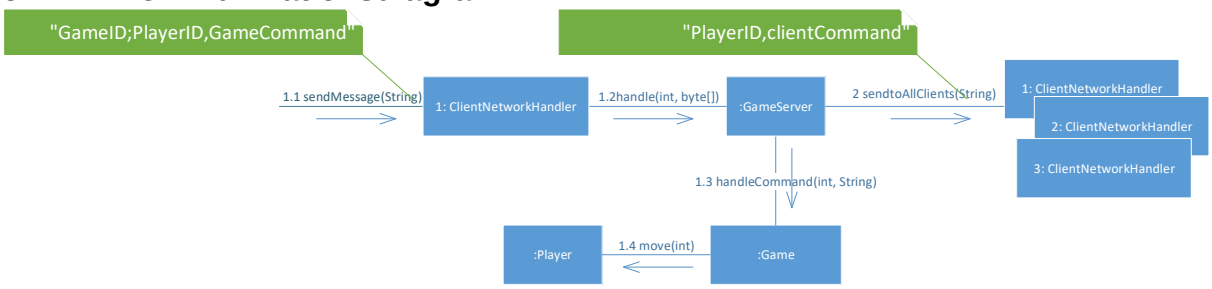
5.4. UC4 Spiel spielen

5.4.1. Sequenzdiagramm



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.

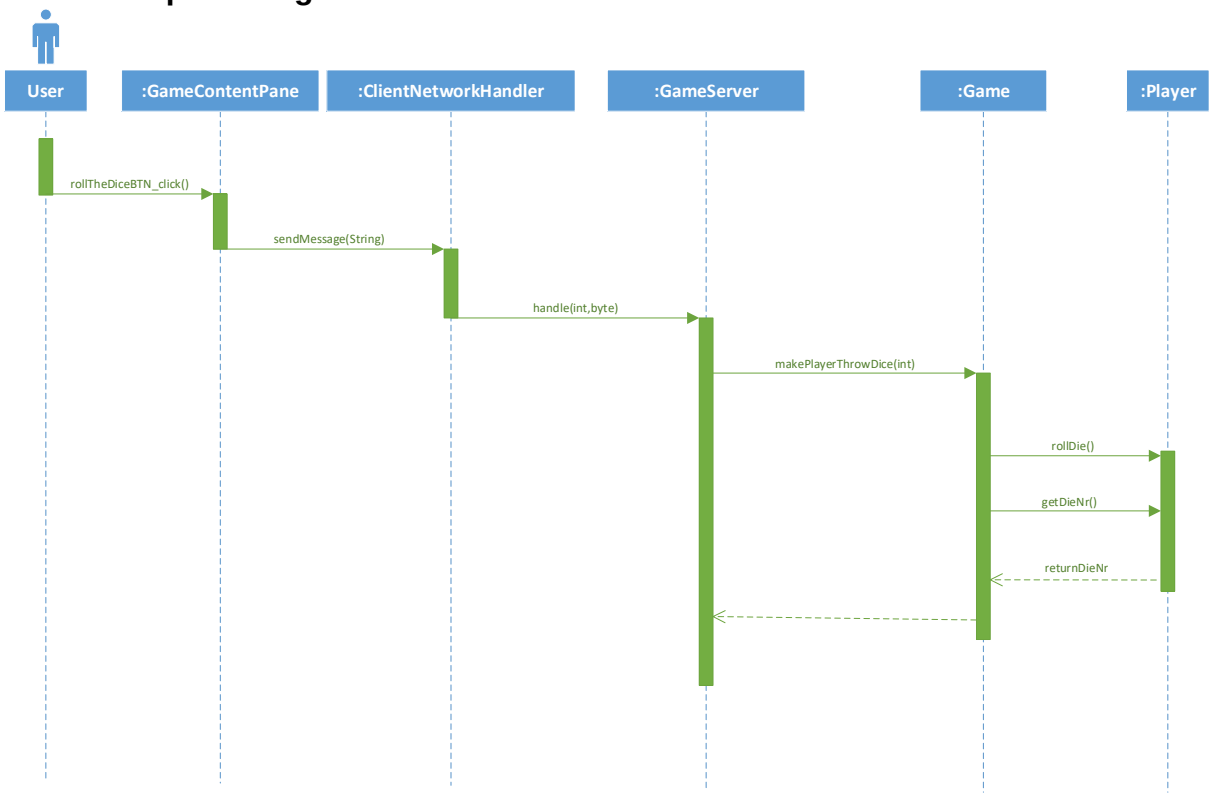
5.4.2. Kommunikationsdiagramm



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.

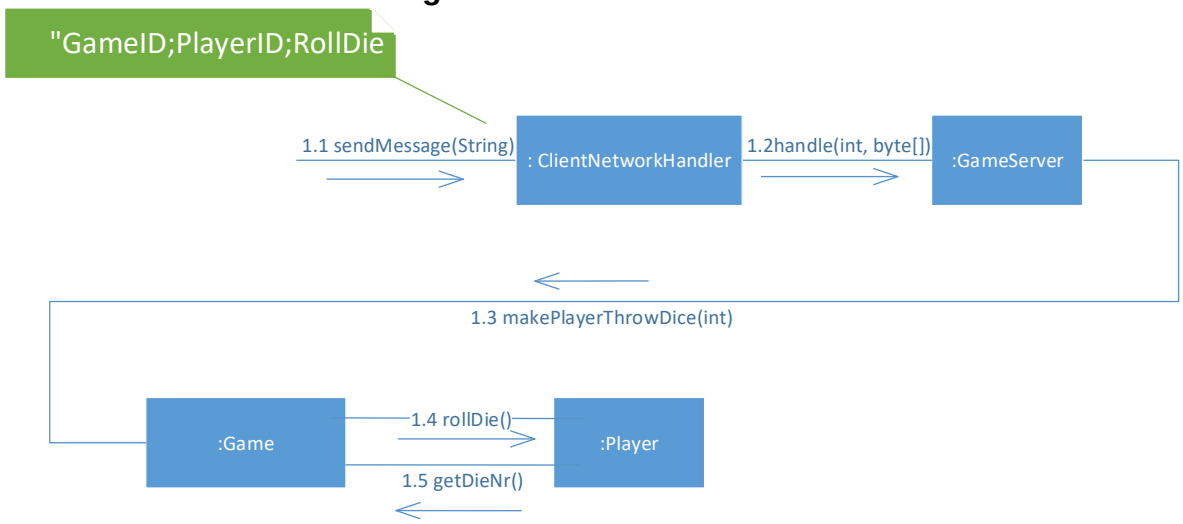
5.5. UC4.1 Spiel spielen - Würfeln

5.5.1. Sequenzdiagramm



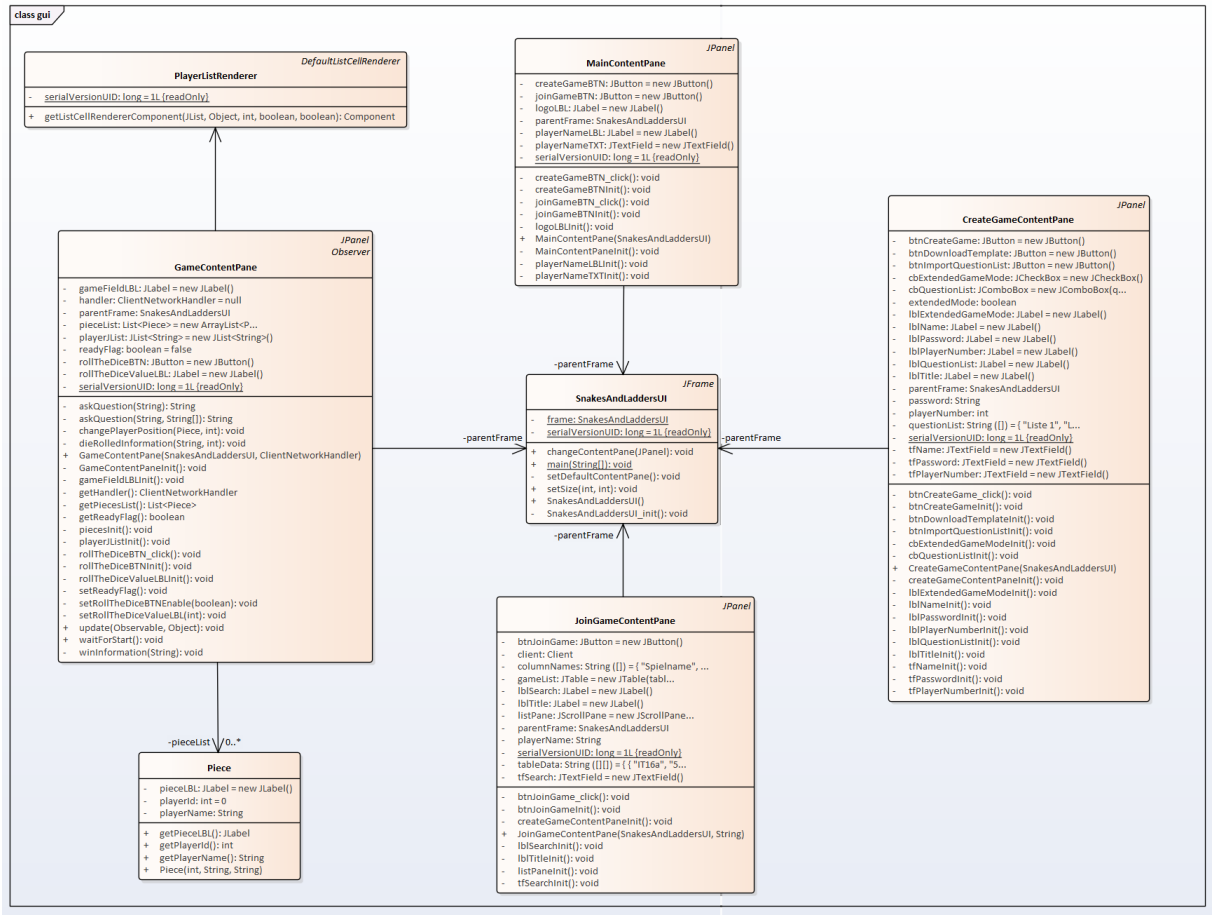
Um einen Würfel zu werfen muss der Benutzer auf den entsprechenden Button auf dem GUI klicken. Diese Aktion wird vom GameContentPane wahrgenommen und per ClientNetworkHandler an den GameServer weitergegeben. Der GameServer übergibt dem Game den Befehl zum Würfeln und die ID des Benutzers. Das Game lässt nun den entsprechenden Player den Würfel werfen. Zum Schluss wird der gewürfelte Wert an das Game und den GameServer zurückgegeben damit er den entsprechenden Klassen für die Weiterverarbeitung zur Verfügung steht (siehe UC4 Spiel spielen).

5.5.2. Kommunikationsdiagramm



6. GUI-Design

6.1. Package GUI:



Im Package GUI wurden die Klassen PlayerListRenderer, GameContentPane, Piece, MainContentPane, SnakeAndLaddersUI, JoinGameContentPane und CreateGameContentPane erstellt. SnakesaandLaddersUI ist das Hauptfenster, welche die einzelnen Oberflächen anzeigt. MainContentPane ist die Oberfläche, in der ausgewählt werden kann, ob ein Spiel erstellt werden soll oder beigetreten werden soll. CreateGameContentPane ist die Oberfläche in der das Spiel konfiguriert und bereitgestellt werden kann. JoinGameContentPane ist die Oberfläche, welche die verfügbaren Server bzw. Spiele anzeigt und wo ausgewählt werden kann in welchem Spiel beigetreten werden soll. GameContentPane ist das Spielbrett wo das Spiel stattfindet. Im GameContentPane werden die einzelnen Spieler (Spielfiguren) mithilfe der Klasse Piece angezeigt. Der PlayerListRenderer dient lediglich dazu, dass die Spielfigur ebenfalls in der Spielerliste anzeigt werden kann.

7. Glossar

Begriff	Definition und Information	Weitere Erklärungen	Aliase, auch in anderen Sprachen	Beziehungen zu anderen Elementen
Administrator	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
Spieler	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
Server-Applikation	Ist ein Programm, welches ein «Snakes and Ladder» Spiel verarbeitet und laufend den einzelnen Spielern ihre nächsten Schritte anzeigt.			
GUI	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
UI	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
Piece	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	