n|w

IT Projekt FS2021 – Gruppe Rum und Ehre 2.0

David Schürch, Samuel David, Jannick Maurer

1 Inhalt

2 Informationen zum Start auf einen Blick

Um die Übersicht, das Testen und die Analyse unseres Claim Spieles zu vereinfachen sind hier die wichtigsten Informationen auf einen Blick dargestellt:

- Die Repository ist «public» und abrufbar unter https://github.com/jannickmaurer/RumUndEhre2.git
- Die gezeigten Prozessdiagramme in diesem Dokument sind ebenso als .bpmn Datei in der Repository vorhanden und lassen sich somit in bspw. Camunda oder direkt im Eclipse öffnen

- Das Spiel ist ausgelegt für zwei Spieler. Es lassen sich beliebig viele Clients mit dem Server verbinden, jedoch nur zwei können sich gleichzeitig anmelden – versucht sich ein dritter anzumelden, erhält er eine Fehlermeldung
- Sobald sich zwei Spieler angemeldet haben, werden die Karten automatisch ausgegeben – der restliche Spielfluss wird dann grösstenteils über Interaktionen im Client GUI gesteuert
- Es lässt sich ein vollwertiges Claims Spiel inkl. erster und zweiter Runde spielen
- Die wichtigsten sonstigen Zusatzfunktionalitäten sind das Account-Handling (Benutzername/PW), das erweiterte GUI, die Chat-Funktionalität sowie die Mehrsprachigkeit (EN/DE)

- Am Ende des Spieles wird beiden Usern ein Pop-Up mit dem Gewinner angezeigt, auf welchem der User dann aufgefordert wird, sich abzumelden
- Sobald sich einer der User während dem Spiel abmeldet, wird dem zweiten Benutzer eine Meldung angezeigt und dieser wird zum Abmelden aufgefordert – das Spiel auf dem Server wird beendet
- Wenn ein User das Fenster schliesst und der Server noch erreichbar ist, wird ein richtiger Logout durchgeführt
- Das Spiel lässt sich auch nach Logout wieder starten, weder der Server noch der Client muss neugestartet werden

- Um das beste Erlebnis mit dem GUI zu erhalten, empfiehlt es sich die Auflösung des Bildschirmes auf 1920 x 1080 einzustellen und, sofern möglich, den Zoom auf 150% einzustellen

3 Anforderungen

Wir verwenden bei der Implementation ein iteratives Vorgehen. Damit stellen wir sicher, dass wir einerseits die Arbeit besser untereinander aufteilen können. Andererseits können wir dem System so neue Funktionalitäten möglichst schnell und im Idealfall unabhängig voneinander hinzufügen. Ausserdem verringert es die Komplexität bei der Entwicklung und allfällige Fehler lassen sich besser lokalisieren. Um das Vorgehen entsprechend zu planen (und uns auch an die Planung zu halten), haben wir die Anforderungen auch gleich auf die jeweiligen Iterationen aufgeteilt, nach KANO Modell eingeordnet und beschrieben, wer die Anforderung umsetzt:

3.1 Implementierung/Arbeitsaufteilung

Sprint 1

Anforderung	KANO	Umsetzung - TBD
Client / Server Architektur	Basis	**Jannick**
Account erstellen / löschen ➔ Die Account-Verwaltung geschieht auf dem Server	Basis	**Jannick**
Login / Logout	Basis	**Jannick**
Client GUI MVP Funktionen: Account Handling & Login / Logout	Basis	**Samuel**
Sprachen Deutsch & Englisch werden unterstützt (Umsetzung in Sprint 1, da Implementierung bei Projektstart am sinnvollsten)	Leistung	**Samuel / David**

Sprint 2

Anforderung	KANO	Umsetzung
Erweiterung GUI - Spieltisch, Spieler fix = 2 - Anzeige Kartendeck und «Handkarten»	Basis	**Samuel**
52 Karten werden gemischt und jeweils 13 an die 2 Spieler verteilt	Basis	**David**
Spieler kann nur gültige Züge durchführen	Basis	**David**
Eine Runde gemäss Regeln durchführen	Basis	**Jannick**
Phase 1 (13 Runden) kann gespielt werden	Basis	**Alle**
Ermittlung des Gewinners gemäss Regeln (MVP: Gewinner Phase 1)	Basis	**Jannick**

Sprint 3

Anforderung	KANO	Umsetzung
Phase 2 kann gespielt werden	Leistung	**Alle**
Ermittlung des Gewinners gemäss Regeln (Spiel)	Leistung	**Jannick**
Spielraum-Verwaltung - Ein User kann einen neuen Spielraum erstellen - Ein anderer User kann diesem Spielraum beitreten - Spielraum kann wieder gelöscht werden	Leistung	**David / Samuel**

Sprint 4

Anforderung	KANO	Umsetzung
Chat-Funktionalität im Spielraum Innerhalb eines Chatraumes können die Teilnehmer chatten	Leistung	**Jannick**
Anbindung an Datenbank	Begeisterung	**David**

Die rot markierten Anforderungen wurden nicht umgesetzt.

4 Use Cases

4.1 Package Administration

Name	**Account erstellen**		
Use Case ID	UC101		
Beschreibung	Das System muss dem User die Möglichkeit bieten, einen neuen Account zu erfassen		
Auslöser	User hat noch keinen Account oder möchte einen neuen erstellen		
Vorbedingungen	Verbindung zum Server muss bestehen		
Eingaben	- Username - Passwort		
Szenario	Hauptszenario	Alternativszenario	Allgemein
	1. Button «Register» auswählen		*«X» auswählen und das Programm wird beendet*
	2. Scene «Registration» wird angezeigt		
	3. Informationen zu neuem Account eintragen: - Textfeld «Username» ausfüllen - PasswordField «Passwort» ausfüllen	3a. Nicht alle Felder ausgefüllt -> Button «Register» nicht anwählbar (grau)	
	4. Button «Register» auswählen	4a. Angaben stimmen mit einem bereits vorhandenen Account überein -> Pop-Up «Error» erscheint – zurück zu 3	
	5. Es wird ein Login mit den neuen Angaben durchgeführt und die Scene «Game» angezeigt.		
	6. Account wird mit allen Angaben auf dem Server gespeichert		
Ausgaben	-		
Nachbedingung	-		

Name	Login		
Use Case ID	UC102		
Beschreibung	Das System muss dem User die Möglichkeit bieten, sich einzuloggen		
Auslöser	User möchte sich einloggen		
Vorbedingungen	Account erstellt bzw. vorhanden (UC101)		
Eingaben	- Username - Passwort		
Szenario	Hauptszenario	Alternativszenario	Allgemein
	1. Informationen eintragen: - Textfeld «Username» ausfüllen - Textfeld «Passwort» ausfüllen	1a. Nicht alle Felder ausgefüllt -> Button «Login» nicht anwählbar (grau)	
	2. Button «Login» auswählen	2a. Angaben stimmen mit einem vorhandenen Account nicht überein -> Pop-Up «Error» – zurück zu 1	
Ausgaben	-		
Nachbedingung	User gelangt zur Scene «Game»		

Name	Logout		
Use Case ID	UC103		
Beschreibung	Das System muss dem User die Möglichkeit bieten, sich auszuloggen		
Auslöser	User möchte sich ausloggen		
Vorbedingungen	Im Programm einloggen (UC102)		
Eingaben	-		
Szenario	Hauptszenario	Alternativszenario	Allgemein
	1. Button «Logout» auswählen		
Ausgaben	-		
Nachbedingung	User gelangt zur Scene «Willkommen»		

Name	**Sprache ändern**		
Use Case ID	UC104		
Beschreibung	Das System muss dem User die Möglichkeit bieten, die eingestellte Sprache zu ändern		
Auslöser	User möchte Sprache ändern		
Vorbedingungen	Verbindung zum Server muss bestehen		
Eingaben	-		
Szenario	Hauptszenario	Alternativszenario	Allgemein
	1. Menu «Sprache» anwählen und im Dropdown zwischen «DE» und «EN» auswählen		
	2. Ausgewählte Sprache wird auf alle Textelemente angewendet	2a. Ausgewählte Sprache ist bereits aktiv -> Textelemente werden nicht aktualisiert	
Ausgaben	-		
Nachbedingung	-		

4.2 Package Spielsteuerung

Name	**Spiel starten**		
Use Case ID	UC201		
Beschreibung	Das System muss dem User die Möglichkeit bieten, das Spiel zu starten		
Auslöser	User möchte Spiel starten		
Vorbedingungen	Zwei User im Spielraum, User muss Besitzer des Spielraums sein		
Eingaben	-		
Szenario	Hauptszenario	Alternativszenario	Allgemein
	1. Button «Karte aufdecken» auswählen	1a. Nicht genügend Spieler im Spielraum -> Button «Karte aufdecken» nicht anwählbar (grau) und noch keine Hand-Karten verteilt	
	2. Oberste Karte des Stapels von der Mitte wird aufgedeckt		
Ausgaben	-		
Nachbedingung	Besitzender User des Spielraums ist an der Reihe (Karten nicht mehr grau)		

Name	**Karte spielen**		
Use Case ID	UC202		
Beschreibung	Das System muss dem User die Möglichkeit bieten, eine Karte zu spielen		
Auslöser	User ist an der Reihe		
Vorbedingungen	-		
Eingaben	Karte auswählen		
Szenario	Hauptszenario	Alternativszenario	Allgemein
	1. Zur Auswahl zugelassene Karten werden hervorgehoben		
	2. User wählt eine Karte		
	3. Karte wird in die Mitte des Tisches verschoben		
	4. Nächster User ist an der Reihe	4a. Gespielte Karte war die letzte einer Runde 4b. Sieger wird ermittelt 4c. Verlierer sieht welche Anhänger-Karte er bekommt 4d. Button «Karte aufdecken» anwählbar	
Ausgaben	-		
Nachbedingung	-		

Name	Karten einsammeln		
Use Case ID	UC203		
Beschreibung	Das System muss dem Rundensieger und -verlierer, eine Karte zuweisen		
Auslöser	Beide User haben gespielt		
Vorbedingungen	-		
Eingaben	-		
Szenario	Hauptszenario	Alternativszenario	Allgemein
	1. Karte mit dem höheren Wert wird ermittelt	1a. Werte sind gleich stark -> Führender Spieler bekommt die Karte 1b. Zweite gespielte Karte war nicht von der gleichen Fraktion -> Führender Spieler bekommt die Karte 1c. Fraktionsfähigkeit ist aktiv -> Höherer Wert gewinnt	
	2. Gewinner bekommt Karte vom Stapel und ist führender Spieler		
	3. Verlierer bekommt oberste Karte des Stapels von der Mitte		
Ausgaben	-		
Nachbedingung	-		

Name	Spiel beenden		
Use Case ID	UC204		
Beschreibung	Das System muss dem User die Möglichkeit bieten, ein Spiel zu beenden		
Auslöser	- User möchte Spiel beenden - Runde wurde beendet		
Vorbedingungen	-		
Eingaben	-		
Szenario	Hauptszenario	Alternativszenario	Allgemein
	1. Button «Logout» auswählen		
	2. Alle Spieler gelangen in Scene «Lobby».		
Ausgaben	-		
Nachbedingung	User gelangt zur Scene «Willkommen»		

4.3 Package Zusätze

Name	**Chatfunktion verwenden**		
Use Case ID	UC301		
Beschreibung	Das System muss dem User die Möglichkeit bieten, Nachrichten an andere User zu senden		
Auslöser	User möchte Nachrichten versenden		
Vorbedingungen	Login (UC102)		
Eingaben	Textnachricht		
Szenario	Hauptszenario	Alternativszenario	Allgemein
	1. Textnachricht in Textfeld eingeben		
	2. Button «Send message» auswählen	2a. Kein Text in Textfeld -> Button «Send message» nicht anwählbar (grau)	
	3. Textnachricht wird zusammen mit Username in Textarea «Nachrichten» angezeigt		
Ausgaben	- «Username»: «Textnachricht»		
Nachbedingung	-		

5 GUI-Mockup

Spielausschnitt in der 1 Phase

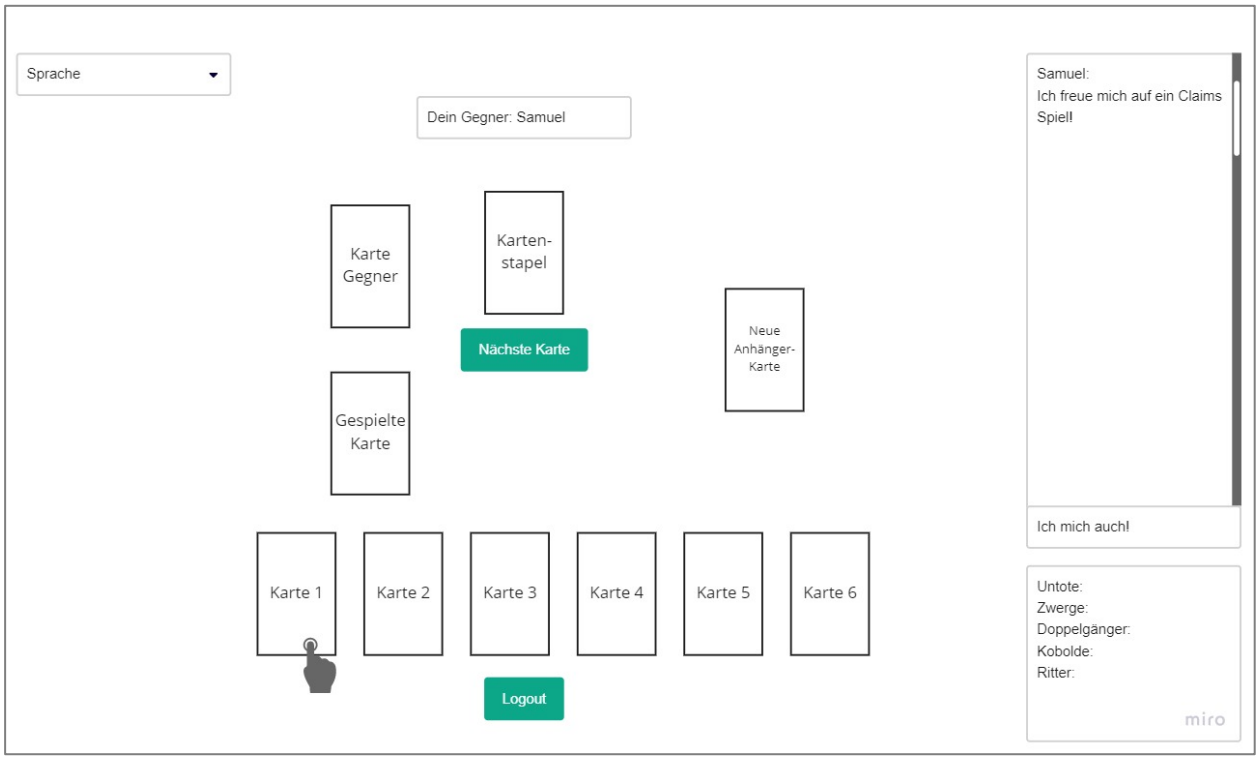


Abbildung 1: Mockup Claim Runde 1

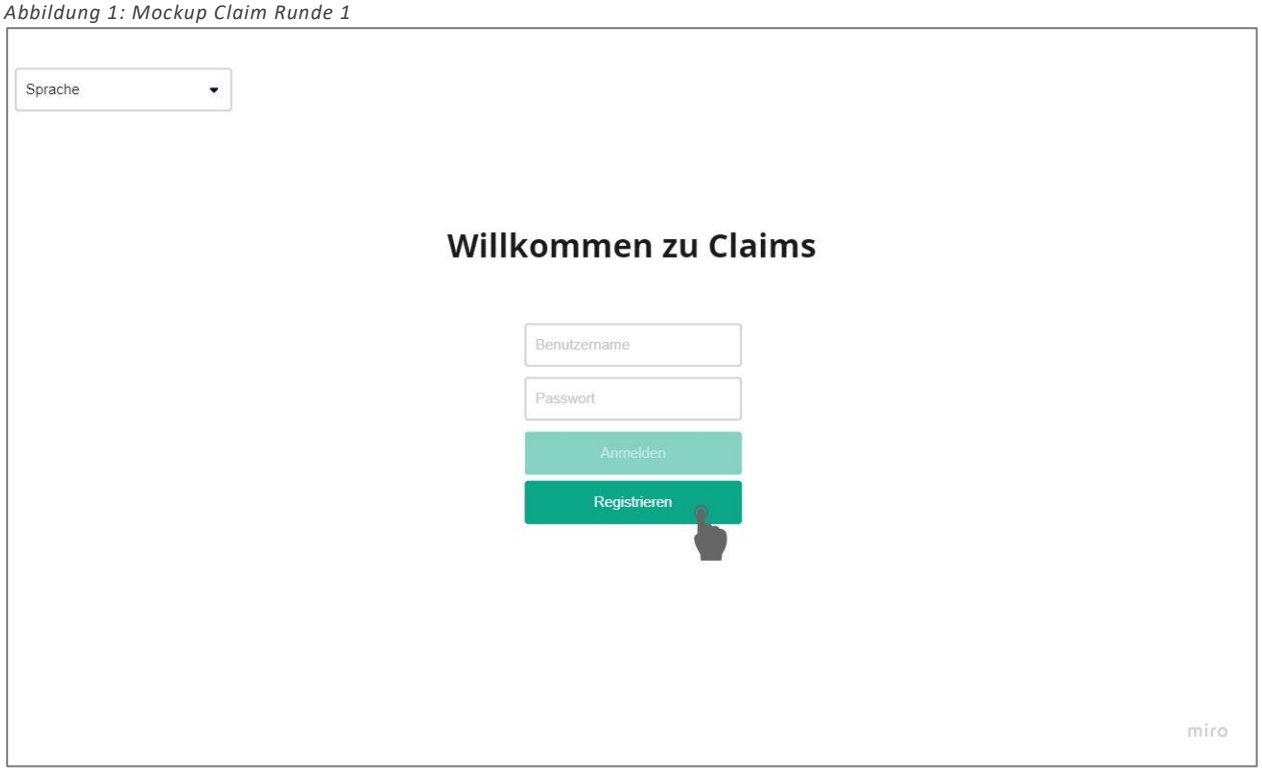


Abbildung 2: Mockup Claim Login

6 Test Plan

Folgend werden die wichtigsten Testfälle beschrieben. Analog zu diesen können auch andere beschriebene Use-Cases getestet werden.

6.1 Testfall: Registrierung

Prozess:
Der Benutzer kann auf den Button Registrierung klicken und anschliessend erscheint die Ansicht um sich mit einem Benutzernamen und einem Passwort zu registrieren.

Funktionalität:
Der Benutzer kann sich nur mit einem Benutzernamen registrieren, welcher noch nicht besteht. Er muss einen Benutzernamen und ein Kennwort zur Registrierung festlegen. Falls sich der Benutzer nicht anmelden will, kann er über den Button zurück zum Startbildschirm kehren, wo er sich anmelden kann, sofern er einen registrierten Account hat.

Voraussetzung:
Das Clientfenster wurde zuvor über Eclipse geöffnet und ausgeführt und ist in seinem Startzustand auf dem Bildschirm ersichtlich.

6.2 Testfall: Login

Prozess:
Der Benutzer drückt auf Login und gibt seine Accountdaten, Benutzername und Kennwort ein und meldet sich an. Anschliessend erscheint das Spielbrett.

Funktionalität:
Der Benutzer muss sich mit seinen korrekten Accountdaten beim Server anmelden. Nach der Eingabe seines Benutzernamens und seines Kennworts werden die Accountdaten an den Server versendet, um den Benutzer zu verifizieren. Er kann sich nur beim Server anmelden, wenn er sich zuvor einmal registriert hat. Bei einem falschen Kennwort wird der Benutzer nicht angemeldet, jedoch kann er das Login erneut versuchen.

Voraussetzung:
Der Benutzer hat sich zuvor beim Server registriert und einen Account mit Passwort erstellt. Das Clientfenster wurde zuvor über Eclipse geöffnet und ausgeführt und ist in seinem Startzustand auf dem Bildschirm ersichtlich.

6.3 Testfall: Logout

Prozess:
Der Benutzer kann sich jederzeit mit Hilfe des Abmeldebuttons oder mit dem Kreuz zum Fenster schliessen abmelden, ausser das Spiel ist am Ende angelangt und die Auswertung hat stattgefunden. Dann kann sich der Benutzer mit dem Abmeldebutton des erschienene Pop-Ups oder mit dem Kreuz zum Fenster schliessen abmelden.

Funktionalität:
Beim Logout muss sich der Client beim Server abmelden. Die Prozesse müssen beim Logout zurückgesetzt werden. Speziell die Handkarten und das GUI, sowie der Chat müssen gelöscht werden. Nach dem Logout muss sich das GUI anpassen und auf das Login GUI zurückkehren. Beim Server muss der Client abgemeldet und die Spielstände gelöscht werden. Der Spieler muss sich jederzeit nach dem Login abmelden können.

Voraussetzung:
Der Spieler hat sich mit seinem Client durch den Login-Prozess angemeldet und ist im Spiel. Der aktuelle Spielstand spielt für das Logout keine Rolle.

6.4 Testfall: **Karte spielen**

Prozess:
Der Benutzer kann im Fall das er den ersten Zug hat eine beliebige Handkarte seiner Wahl spielen, nachdem die Tischkarte ausgegeben wurde. Der Benutzer kann im Fall das er den zweiten Zug hat nur die evaluierten Spielkarten ausspielen, welche nicht ausgegraut sind.

Funktionalität:
Die Tischkarte muss währen den ersten 13 Runden gespielt sein, bevor der Client mit dem ersten Zug spielen darf. Für den ersten Client sind alle seine Handkarten auf playable gesetzt, da er als erster Spieler die freie Kartenwahl hat. Nachdem der Client die Karte gespielt hat wird diese an den Server versendet, welcher die Tischkarte an den zweiten Client weiterleitet.
Der zweite Client bekommt die zuerst gespielte Karte des Gegners und wertet diese auf dem Board aus und setzt alle Karten, die mit einem gültigen Zug gespielt werden können auf playable. Die Karten, die nicht gespielt werden können, werden ausgegraut und können nicht angeklickt werden. Nachdem der zweite Spieler seine Karte gespielt hat wird diese an den Server gesendet, welcher sie dem ersten Spieler weiterleitet.
Nachdem eine Handkarte gespielt wurde wird diese immer in der Tischmitte angezeigt.

Voraussetzung:
Der Benutzer hat sich zuvor erfolgreich über das Login angemeldet. Ein zweiter Client hat sich angemeldet, welcher über den Server mit ihm am selben Spiel teilnimmt. Vom ersten eingeloggten Benutzer wurde der Button für die nächste Tischkarte gerückt und diese wird nun angezeigt. Nach den ersten 13 Karten kann der Benutzer keine Tischkarte mehr anfordern, jedoch kann er auf den Button für eine neue Spielrunde drücken, wonach der erste Client spielen kann. Der Client mit dem zweiten Zug darf immer erst spielen, wenn der Client mit dem ersten Zug seine Karte gespielt hat.

6.5 Testfall: **Sieger nach einer Runde evaluieren**

Prozess:
Beide Spieler haben je eine Handkarte gespielt und dadurch an den Server gesendet. Die Auswertung wer die Runde gewonnen hat startet automatisch.

Funktionalität:
Der Server prüft ob er zwei Handkarten hat und wenn das der Fall ist, wertet er den Sieger über die finishRound aus und gibt mit einer Message den Sieger Benutzername der Runde an beide Spieler weiter. Die Spieler prüfen nun, ob der Benutzername Ihrem Benutzername entsprechen. Wenn das der Fall ist, darf der Spieler der gewonnen hat den Button drücken für einen neue Tischkarte (innerhalb der ersten 13 Runden) oder auf den Button für die nächste Runde (ab der 13 Runde) drücken.
Bei der Auswertung des Siegers wird geprüft, ob man sich in der ersten Runde (erste 13 Karten) oder in der zweiten Runde befindet (ab der 13 Karte). Je nachdem in welcher Runde man sich befindet gelten andere Regeln. Im Fall eines Patts gewinnt immer der erste Spieler, ansonsten gelten die Regeln der ersten, beziehungsweise der zweiten Runde, wonach der Sieger evaluiert wird.

Voraussetzung:
Beide Spieler sind im Spiel und jeder hat bereits eine Handkarte gespielt, welche an den Server gesendet wurde.

6.6 Testfall: Spiel Sieger evaluieren

Prozess:
Nach der letzten gespielten Karte in der zweiten Runde kann der Spieler, welcher in dieser Runde als erster am Zug war den Button für die Siegesauswertung des ganzen Spieles drücken. Anschliessen erscheint ein Pop-Up, welches den Benutzernamen des Siegers ausgibt.

Funktionalität:
Nachdem der Button für die Siegesauswertung des gesamten Spiels gedrückt wurde, wird auf dem Server die winner aufgerufen, welche den Benutzernamen des Spielers zurückgibt, der das ganze Spiel gewonnen hat. Bei einem Patt gibt es die Meldung NoWinner aus, dass es keinen Sieger gibt.
Für die Siegerevaluierung werden die gewonnenen Karten jedes Spielers sortiert nach Ihrer Fraktion und anschliessend mit derselben Fraktion des Gegners verglichen. Sobald ein Spieler mehr Karten einer Fraktion hat, hat er gewonnen. Falls es einen Gleichstand gibt, gewinnt der Spieler mit der höchsten Karte dieser Fraktion. Wenn keiner der Spieler diese Fraktion hat, gibt es ein Patt und die Fraktion wird nicht gewertet. Am Ende wird ausgewertet, welcher Spieler wie viele Fraktionen gewonnen hat. Der Spieler mit den meisten Fraktionen, aber mindestens drei Fraktionen gewinnt das gesamte Spiel.

Voraussetzung:
Alle 26 Runden wurden des Spiels wurden gespielt und der Button für die Auswertung des Siegers erscheint bei dem Spieler, der die letzte Runde begonnen hat.

7 System Spezifikation

7.1 Komponenten

Das System wird mit einer Client/Server Architektur implementiert. Der Server ist in java programmiert und wird ohne GUI implementiert. Für den Client wird ein GUI implementiert, welches mit javaFX programmiert wird – die Backend Funktionalitäten werden ebenfalls in java programmiert. Die Spiellogik wird primär server-seitig implementiert, während der Client die Befehle entgegennimmt und das GUI entsprechend anpasst. Es werden so wenige Berechnungen und Evaluationen wie möglich auf der Client-Seite durchgeführt – dadurch stellen wir sicher, dass die beiden Clients zu jederzeit dieselbe Ausgangslage betreffend des Spielstatus besitzen.

7.2 Kommunikation

Die Kommunikation zwischen Server und Client geschieht mittels des Versands von String Objekten, welche den Namen (Zweck) einer Message enthalten. Der Client sowie der Server erstellen dann jeweils beim Empfangen eines Strings ein „Message“ Objekt, welches dann wiederum verschiedene Methoden beinhaltet, die die notwendigen Funktionalitäten ausführen. Eine Message vom Client erfordert in jedem Fall eine Antwort des Servers. Sprich, sendet der Client eine Message and den Server, sendet der Server nach dem Prozessieren zwingend eine Antwort. Diese kann entweder eine aus der Funktionalität resultierende Message sein, die wiederum beim Client Funktionalitäten abruft, oder aber eine „Error“ Message, sollte der Server die ursprüngliche Message nicht verarbeiten können.
Nachfolgend sind verschiedene Prozessdiagramme gezeigt. Das erste verdeutlicht anhand des Beispiels „CreatePlayroom“ wie die Kommunikation und das Konzept der Messages im Detail funktioniert. Diese generelle Kommunikationslogik kann so für jede Art Message übernommen werden, wurde jedoch nicht für jeden Typ modelliert, da dies unnötige Redundanzen mitbringen würde. Das weiteren Prozessdiagramme zeigen dann den High-Level Ablauf des Spieles und fokussieren dabei auf die relevanten Aktionen und Messages. Es wird zur besseren Verständlichkeit bei allen Diagrammen eine vereinfachte BPMN Notation verwendet. Wie einleitend erwähnt, sind alle Prozesse in der Repository auch als BPMN Datei vorhanden, diese lässt sich wiederum beispielsweise in Camunda oder direkt im Eclipse öffnen.

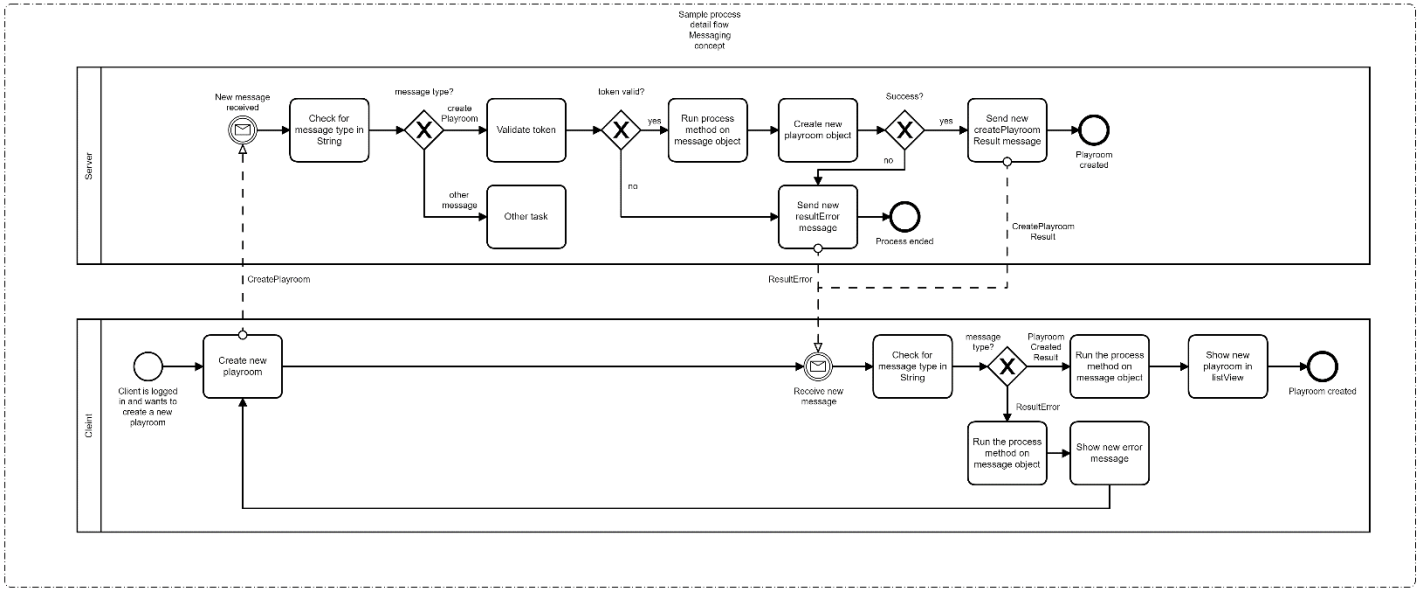


Abbildung 3: Prozess Messaging Detail

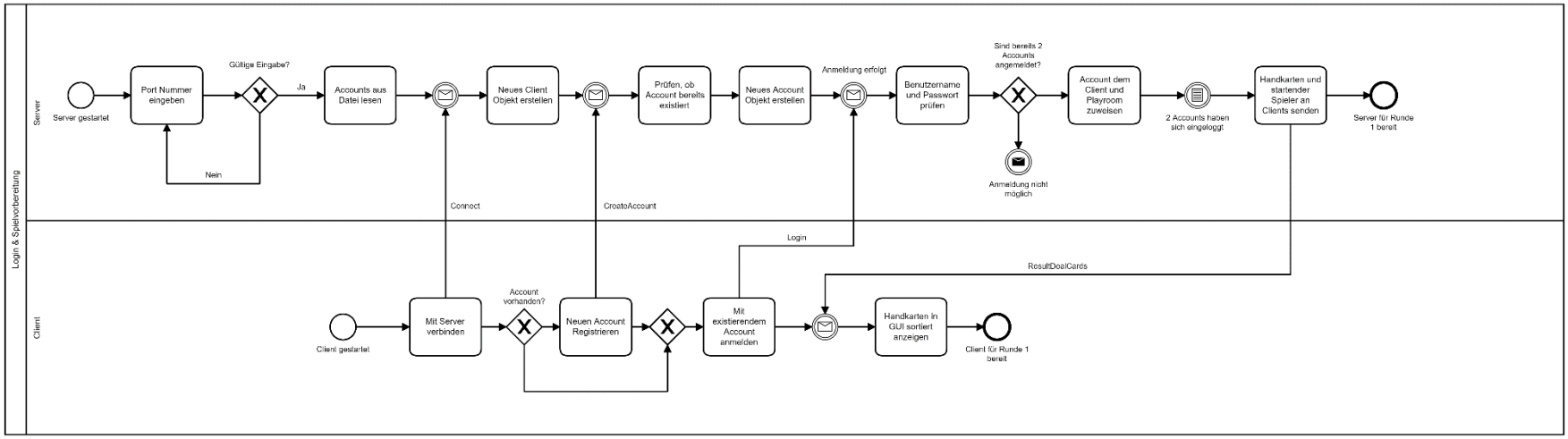


Abbildung 4: Prozess Spielvorbereitung

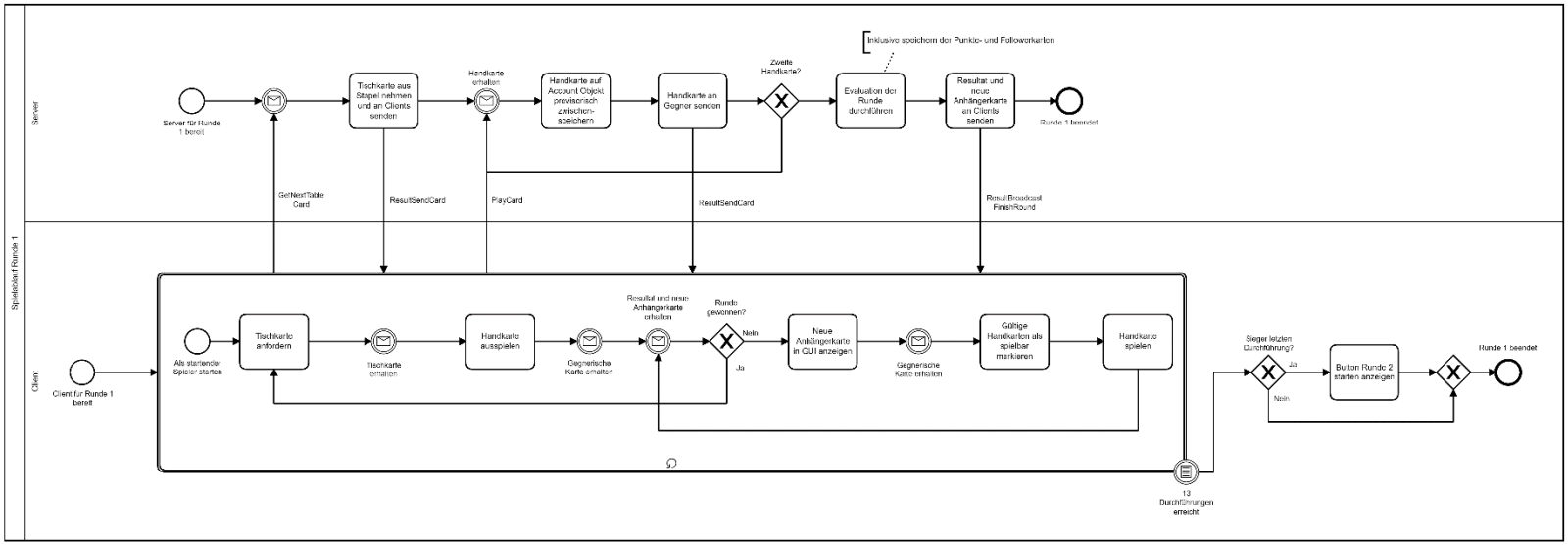


Abbildung 5: Prozess Runde 1

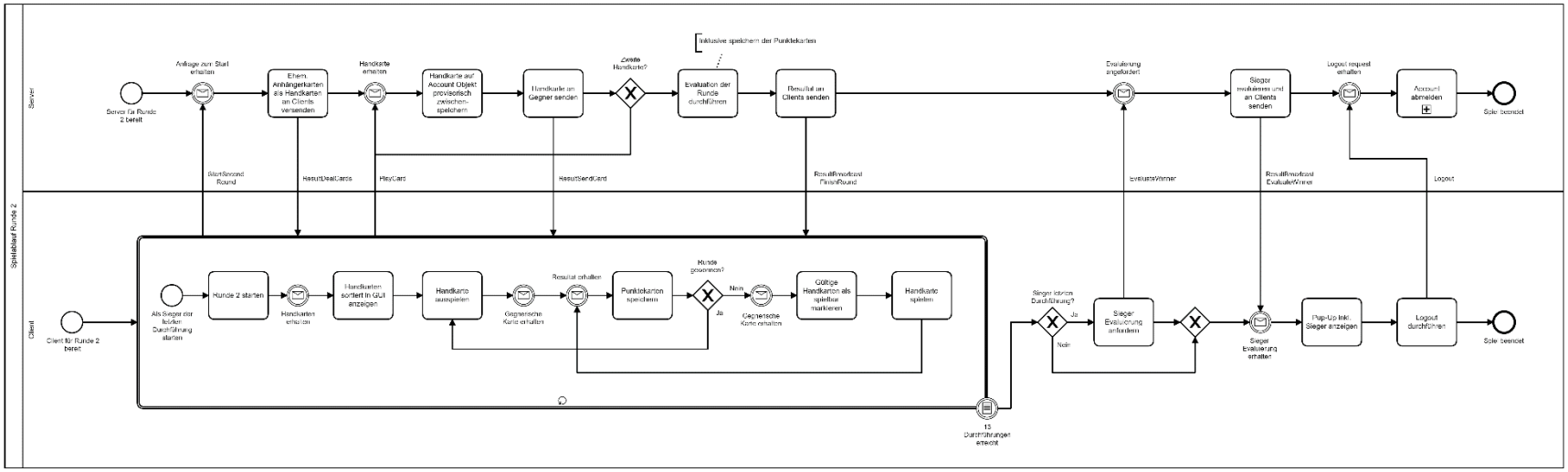


Abbildung 6: Prozess Runde 2

7.3 Anwendungslogik

Im Spiel Claim ist es an verschiedenen Stellen notwendig, dass Clients nur bestimmte – also „erlaubte“ – Handlungen vornehmen. Beispielsweise darf ein Spieler nur gültige Züge spielen, bspw. also nur eine Karte derselben Fraktion spielen, sofern er Karten dieser Fraktion auf der Hand hat. Solche Restriktionen in der Anwendungslogik werden wir auf dem Client durch Anpassungen im GUI sicherstellen. So stellen wir sicher, dass anwenderseitig nur Handlungen vorgenommen werden können, die der Server zum gegebenen Zeitpunkt auch verarbeiten kann und darf – dadurch ersparen wir uns eine erneute Validierung auf dem Server (wir sind der Meinung, im Rahmen dieses Projektes genügt das). Einige Beispiele:

- Ist der User nicht eingeloggt, wird im ihm GUI nie die Möglichkeit gegeben, ein Spiel zu starten
- Der Server sendet an den Client, welche Fraktion gespielt wurde. Der Client wiederum evaluiert, welche seiner Handkarten gespielt werden können – und der User kann im GUI nur diese auswählen.
- Wartet der User auf den Zug des Gegenspielers, werden im GUI alle nicht wählbaren Buttons ausgegraut

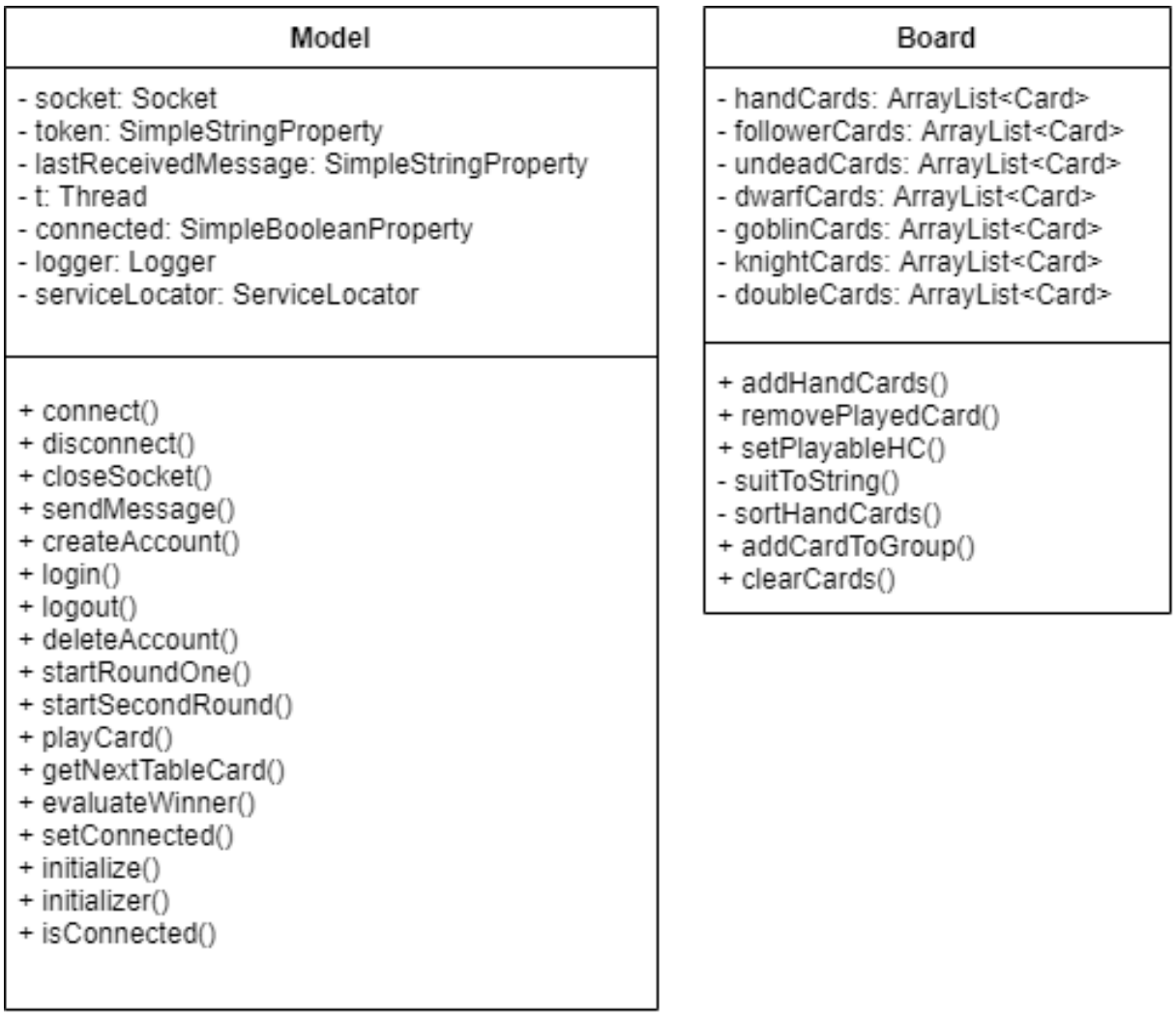
7.4 Klassendiagramme UML

Auf den nachfolgenden Seiten werden einige UML Klassendiagramme gezeigt. Dabei wurden lediglich die für das Spiel sowie die Client-Server Verbindung notwendigen «funktionalen» Klassen illustriert. Der Server wird kein GUI besitzen, der Client wird mittels MVC implementiert – die Klassen für die View und den Controller werden jedoch nicht im Klassendiagramm abgebildet. Die View ergibt sich aus den Mock-Up, wobei der dafür zuständige Entwickler dann selbst entscheiden kann, welche JavaFX Elemente er dafür verwendet. Ausserdem fehlen aus Gründen der Einfachheit (und da diese eher standardisiert eingesetzt werden) in den Klassendiagramme folgende Elemente:

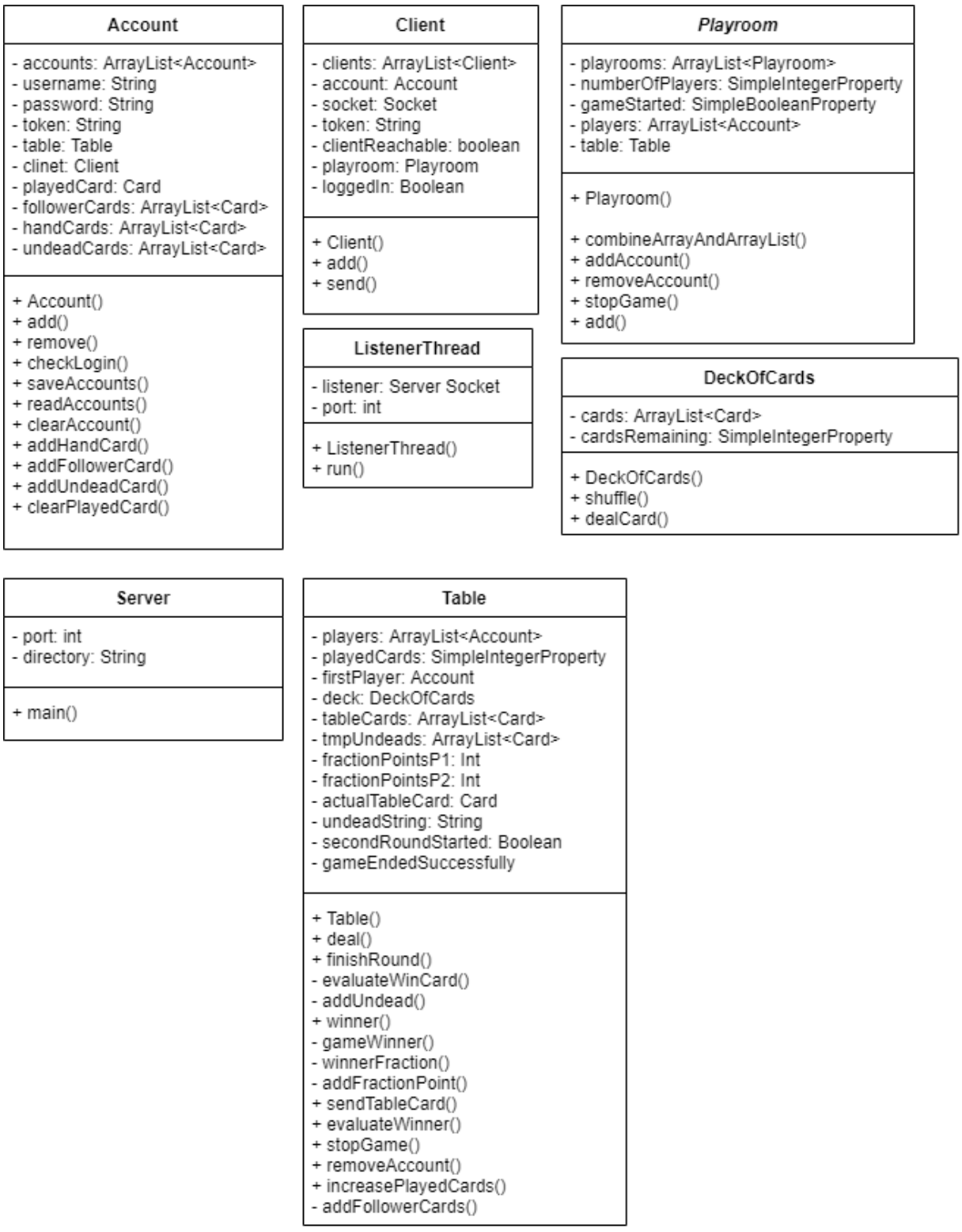
- Umstellung Sprache
- Logging
- ServiceLocator allgemein
- Passwort Security (wird mittels Hash realisiert)
- ALLE Getter und Setter der Instanz-Variablen

Die Klassendiagramme sind gemäss dem Package, in dem die Klassen implementiert sind, illustriert.

7.4.1 Package Client.Model



7.4.2 Package Server



7.4.3 Package Message