2015

Schnaps2gether - Wartungshandbuch

DOKUMENTATION TEIL 2 - WARTUNGSHANDBUCH

EBERHARD, EDER, MAIER, PACHATZ, PLATTER

Inhalt

1. Architektur

Das Projekt ist wie für Android-Projekte üblich in einen Java-Teil und einen Ressourcenteil aufgeteilt. Dabei enthält der Ressourcenteil die Layouts für Activities, Popups und Menüs, und außerdem die verwendeten Bilder, Icons und Strings. Der Java-Teil ist wiederum in zwei Pakete aufgeteilt: „DataStructure" und „GUI". Ersteres enthält die Geschäftslogik, welche die Spielregeln implementiert, und die Datenstrukturen, welche die im Spiel vorhandenen Entitäten wie Karten und Spieler darstellen. Letzteres enthält die Klassen, welche auf Nutzerinteraktionen reagieren, die Funktionen der Geschäftslogik aufrufen und die zurückgelieferten Daten in die Präsentationsschicht übertragen; dementsprechend sind auch all diese Klassen mit einem Layout aus dem Ressourcenteil verbunden.

Klassendiagramme befinden sich im Anhang A

2. Design

a) Startmenü, Nickname, Spielstatistiken und Regeln

Das Startmenü ist nur eine statische Activity mit sechs Buttons, die entweder eine andere Activity oder ein Popup aufrufen oder die App beenden. Man kann in dieser Activity den eigenen Nicknamen ändern; dabei wird er lokal als „shared preference" abgespeichert. Auf dieselbe Weise werden auch die über das Startmenü erreichbaren Spielstatistiken abgespeichert. Die ebenso über das Startmenü einsehbaren Spielregeln enthalten nur einen statischen Text, der die Spielregeln beschreibt.

b) Lobby und Netzwerkverbindung

Das Layout der Lobby enthält einen ListView, welcher die Spiele im verbundenen WLAN anzeigt. Außerdem gibt es dort Buttons um ein Spiel zur Liste hinzuzufügen, die Liste zu aktualisieren oder zum Startmenü zurückzukehren. Sobald einem Spiel genügend Spieler beigetreten sind wird es automatisch gestartet. Da für die Lobby eine Verbindung verwendet werden muss, wurde diese in der entsprechenden GUI-Klasse „Lobby" implementiert. Konkret wurde dafür die Nearby Connections API von Google verwendet, mit der man sich über ein lokales Netzwerk mit anderen Geräten verbinden und Nachrichten austauschen kann. Dafür implementiert die Klasse „Lobby" eine Reihe von Interfaces, welche Methoden spezifizieren, die für den Verbindungsaufbau verantwortlich sind. Insbesondere werden folgende Methoden dafür verwendet:

- startAdvertising: Diese Methode wird dafür verwendet einen Service über das verbundene Netzwerk anzubieten. Konkret wird damit anderen Nutzern im Netzwerk ein offenes Spiel angezeigt, dem sie beitreten können. In dieser Methode wird über die Variable NO_TIMEOUT spezifiziert, dass das Anbieten des Services nur über die Methode stopAdvertising oder bei einem Verbindungsabbruch gestoppt wird.

- startDiscovery: Diese Methode wird dafür verwendet um über das verbundene Netzwerk angebotene Services zu finden. Konkret werden hier nur von der App Schnaps2gether angebotene Services gesucht, also offene Spiele. In dieser Methode wurde über die Variable DISCOVER_TIMEOUT der Timeout für die Suche nach einem Service auf eine Minute festgesetzt.

- connectTo: Diese Methode sendet eine Verbindungsanfrage zu einem Host. In dieser Methode wird auch die endpoint-ID des Hosts für den späteren Nachrichtenaustausch gespeichert.

Außerdem gehören auch einige Handler für bestimmte Ereignisse zu den implementierten Methoden. Nennenswert sind davon:

- onEndpointFound: Diese Methode fügt einen gefundenen Service als offenes Spiel zur Spieleliste hinzu.

- onMessageReceived: Diese Methode verarbeitet alle über die Nearby Connections API empfangenen Nachrichten. Dementsprechend delegiert sie auch Nachrichten, welche sich auf bereits laufende Spiele beziehen, an „receiveFromLobby" genannte Methoden in den entsprechenden Klassen (die „Spielfeldklassen").

- onConnectionRequest: Diese Methode verarbeitet alle eingehenden Verbindungsanfragen. Falls das aktuelle Gerät ein Host ist, wird die Anfrage angenommen und die endpoint-ID des anfragenden Clients für den späteren Nachrichtenaustausch gespeichert. Falls sich genügend Clients für den gewählten Spieltyp verbunden haben, wird das Anbieten des Services gestoppt und ein Spiel gestartet.

Die Art der Netzwerkverbindung wurde über die statische Klassenvariable NETWORK_TYPES auf WIFI oder Ethernet beschränkt, wobei angenommen wird, dass im Allgemeinen ein WLAN verwendet wird. Die tatsächliche Überprüfung des Netzwerks geschieht in der Methode isConnectedToNetwork.

c) Spielregeln und KI

Die Spielregeln der Spielmodi 2er-, 3er- und 4er-Schnapsen wurden in den entsprechenden Klassen „spiel2", „spiel3" und „spiel4" implementiert. Dabei wird jeweils im Konstruktor alles für den Spielbeginn vorbereitet, soll heißen die Reihenfolge der Spieler wird festgelegt, das Kartendeck gemischt und die ersten Karten ausgeteilt. Die weiteren Methoden verändern den Spielzustand entsprechend den Nutzerinteraktionen, die von einer GUI-Klasse übertragen werden. Außerdem enthält die Klasse „spiel2" auch die Implementierung des entsprechenden Computergegners. Der Computergegner entscheidet deterministisch aufgrund des aktuellen Spielzustands, welche Aktionen er durchführt. Er wehrt auch Schummelversuche ab, dies geschieht allerdings zufallsbasiert.

d) Karten, Spieler, Bummerl und Rufspiel

Kartenobjekte haben einen Wert, eine Farbe und eine Punktezahl. Außerdem enthält sie auch Methoden die entsprechende Bildressourcen zurückliefern und eine Methode, die ein ganzes Kartendeck erstellt. Um Karten sortieren zu können wurde außerdem ein Kartenkomparator erstellt.
Ein Spielerobjekt speichert unter anderem die Handkarten und die Punkte der aktuellen Spielrunde des entsprechenden Spielers.
Es gibt für jeden Spielmodus eine eigene Bummerlklasse, welche die Gesamtpunkte

der einzelnen Spieler sowie die Anzahl der bereits gespielten Spielrunden abspeichert. Außerdem enthalten diese Klassen auch eine Methode, über die man abfragen kann, ob das ganze Spiel zu Ende ist.
Ein Rufspielobjekt speichert den Namen und die dafür bei einem Sieg erhaltenen Punkte ab.
Für Karten und Bummerl wurden auch Methoden und Konstruktoren für die Serialisierung in Strings und die Rekonstruktion aus solchen Strings implementiert. Dies wird für den Nachrichtenaustausch verwendet.

e) Spielfeld2

Die Klasse „Spielfeld2" wird für das Spielen gegen einen Computergegner („Schnelles Spiel") verwendet. Sie kümmert sich darum, dass ein Nutzer nur erlaubte Aktionen durchführen kann, indem sie die entsprechenden Elemente der Präsentationsschicht bezüglich Sichtbarkeit und über Aktivierung oder Deaktivierung kontrolliert. Davon abgesehen verarbeitet sie Nutzerinteraktionen sowie die Reaktionen der KI und aktualisiert die Präsentationsschicht entsprechend deren Ergebnisse. Hierfür implementiert sie OnClick-Handler für klickbare Layoutelemente, aktualisiert Layoutelemente, ruft bei Bedarf Popups oder Menüs auf und verwendet die Klassen aus dem Paket „Datastructure". Außerdem werden am Ende einer Spielrunde die Spielstatistiken aktualisiert. Auch die möglichen Schummelaktionen und das Abwehren derselben wurden in dieser Klasse implementiert.

f) Spielfeld2Host und Spielfeld2Client

Die Klasse Spielfeld2Host wird vom Host eines Mehrspielerspiels im Spielmodus 2er-Schnapsen verwendet, Spielfeld2Client vom Client. Grundsätzlich erledigen die Klassen dieselben Aufgaben wie die Spielfeld2Klasse mit dem Unterschied, dass sie auch noch die Kommunikation mit dem jeweils anderen Spieler implementieren. Für die Verbindung werden die entsprechenden Objekte aus der Klasse „Lobby" weiterverwendet. Für die Kommunikation zwischen Spielern werden Nachrichten aus Konstanten und Strings mit Informationen über den Spielzustand zusammengestellt, über die Nearby Connections API verschickt und auf dem Gerät des anderen Spielers von einem MessageListener (siehe Lobby) geparst. Die für die Nachrichten verwendeten Konstanten sind am Beginn der sie verwendenden Klassen definiert. Die restlichen verwendeten Strings werden entweder nur als Literale verwendet oder über Methoden in den entsprechenden Datenstrukturen erstellt und geparst. Davon abgesehen greift die Klasse „Spielfeld2Client" nicht direkt auf die Methoden der Hintergrundlogik („spiel2") zu, sondern erhält alle relevanten Informationen vom Host. Dies wird von Sequenzdiagrammen, die das Ausspielen einer Karte vom Host und vom Client abbilden, illustriert.

g) Spielfeld3Host und –Client und Spielfeld4Host und –Client

Die Spielfeld3- und Spielfeld4-Klassen sind grundsätzlich analog zu den Spielfeld2-Klassen aufgebaut. Allerdings implementieren sie natürlich entsprechend unterschiedliche Spielabläufe. Insbesondere muss der Host die Nachrichten, welche er von einem Client erhält, an die anderen Clients weiterleiten, da er als Bindeglied zwischen ihnen dient. Im Fall von Spielfeld4 wurde ein Großteil des noch auf Spielabläufe bezogenen Codes in die „Datastructure"-Klasse „Spielfeld4Logik"

extrahiert, sodass die Spielfeld4-Klassen fast nur mehr die Kommunikation mit der Präsentationsschicht und mit den anderen Spielern enthalten. Abgesehen davon implementieren diese Klassen weder das Schummeln, noch das Abspeichern von Spielstatistiken.

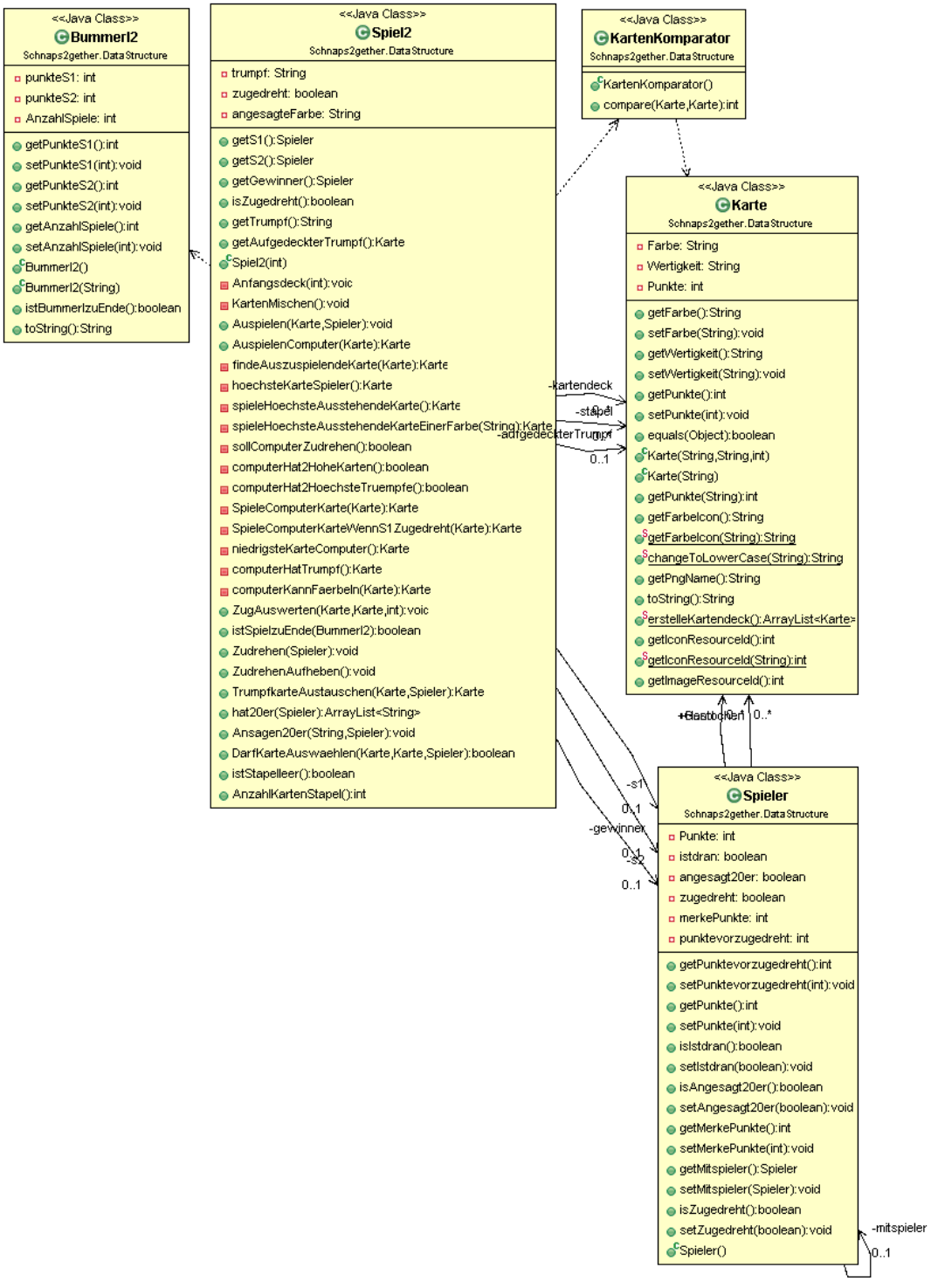
3. Anhang A – Klassendiagramme

a) Klassendiagramm GUI

- Klassendiagramme\pdf\GUI-logic.pdf

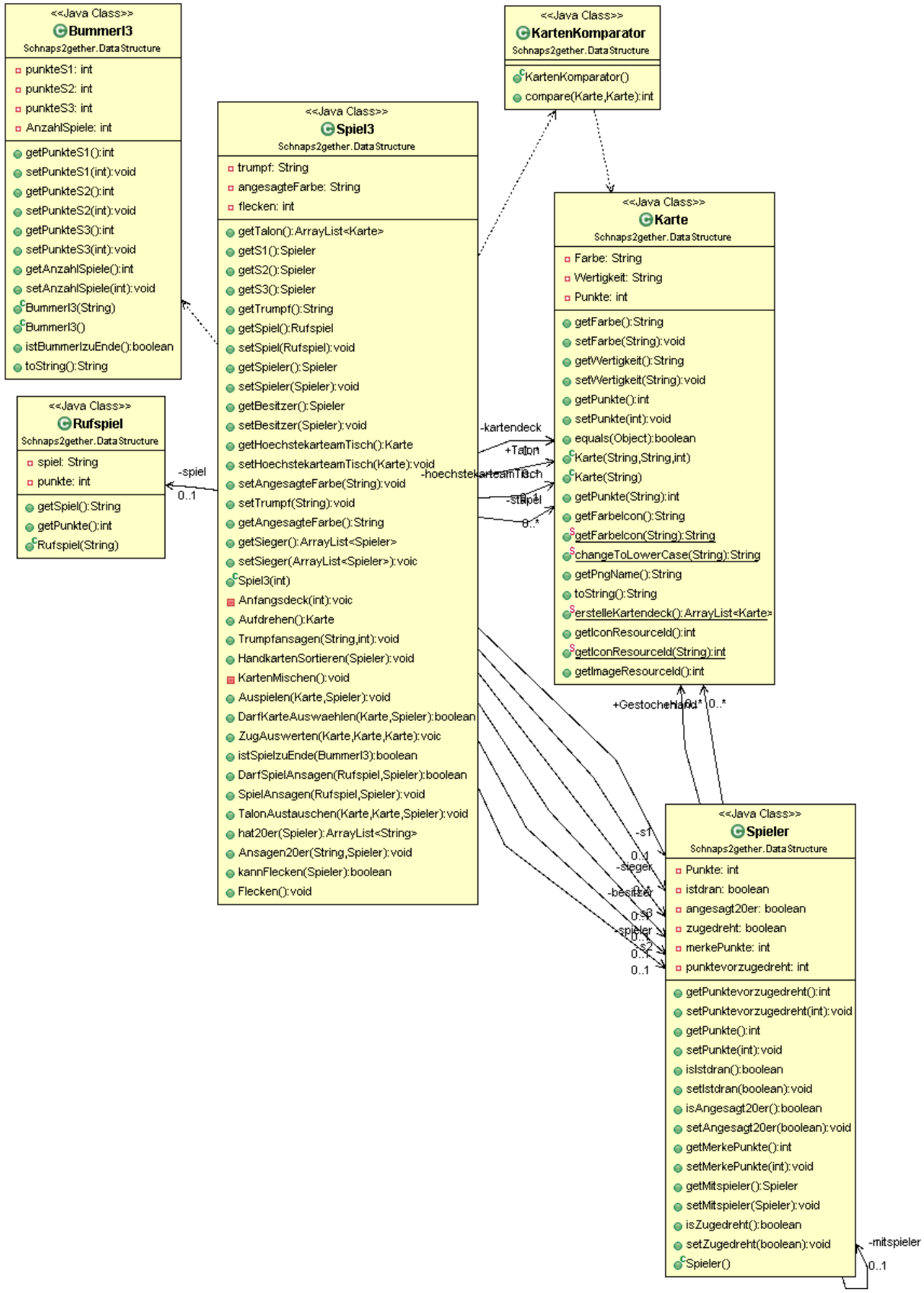
b) Klassendiagramm Spiel 2

- Klassendiagramme\pdf\Spiel2.pdf



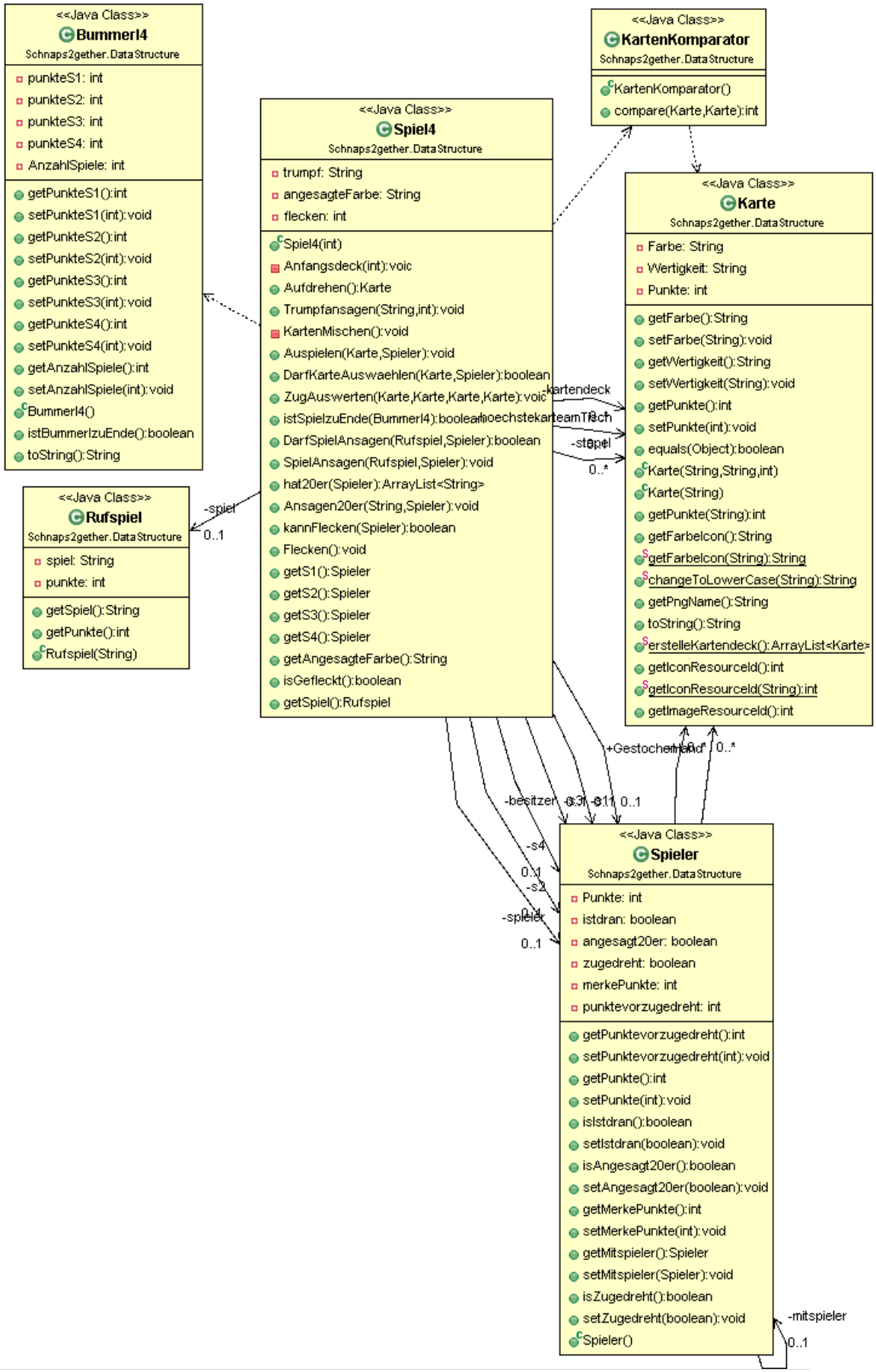
c) Klassendiagramm Spiel 3

- Klassendiagramme\pdf\Spiel3.pdf



d) Klassendiagramm Spiel 4

- Klassendiagramme\pdf\Spiel4.pdf



4. Anhang B – Sequenzdiagramme

a) Sequenzdiagramm – Spielfeld2 Host – Zug ausführen

- Sequenzdiagramme\Sequenzdiagramm-Spielfeld2Host-zugAusfuehren.pdf

b) Sequenzdiagramm – Spielfeld2 Client – Zug ausführen

- Sequenzdiagramme\Sequenzdiagramm-Spielfeld2Client-zugAusfuehren.pdf