Entwicklung eines “Kniffel“ Computer-Spiels

Teil II

Design

1 Allgemeines

Das vorliegende Dokument beschreibt das Design des Kniffel – Computerspiels. Grundlage für das Design ist das in der ersten Phase erarbeitete Analysedokument Kniffel-Analyse.doc.

2 Schichtenmodell

Die primäre Architektur des Kniffel-Spiels beruht auf einem Drei-Schichtenmodell. Die einzelnen Schichten nutzen lediglich die Funktionalität der eigenen Schicht oder die der darunter liegenden.

2.1 Bedienoberfläche

Diese Schicht realisiert die grafische Oberfläche und stellt die Schnittstelle zum Anwender bereit(GUI). Hier werden die Tastatur- und die Mauseingaben in Dienstanforderungen an die Applikationsschicht gewandelt und die aktuelle Spielsituation angezeigt. Die Bedienoberfläche verfügt selbst über keine weitere Funktionalität der Anwendungslogik. Ein oberflächenorientiertes Bedienkonzept wird im nächsten Abschnitt beschrieben. Dies ist die Grundlage für die weitere Betrachtung der Use-Cases. Die daraus resultierende Zustandsmaschine wird nicht in der GUI, sondern ebenfalls in der Applikationslogik der Applikationsschicht realisiert. Die GUI wird lediglich über die Zustandswechsel informiert.

2.2 Anwendungsschicht

Die Anwendungsschicht ist die Schicht, die die eigentliche Funktionalität erbringt. Sie verfügt über eine funktionale Schnittstelle, die es ermöglicht, das Spiel zu spielen ohne auf eine spezielle GUI angewiesen zu sein. Aufgabe dieser Schnittstelle ist es darüber hinaus, die darunter liegenden Subsysteme zu kapseln und deren Details zu verbergen. Dies entkoppelt die einzelnen Schichten und erleichtert lokale Änderungen.

2.3 Persistenzschicht

In dieser Schicht sind die grundlegenden Dienste angesiedelt, die den Zugriff auf Daten der Highscoreliste realisieren und deren persistente Speicherung.

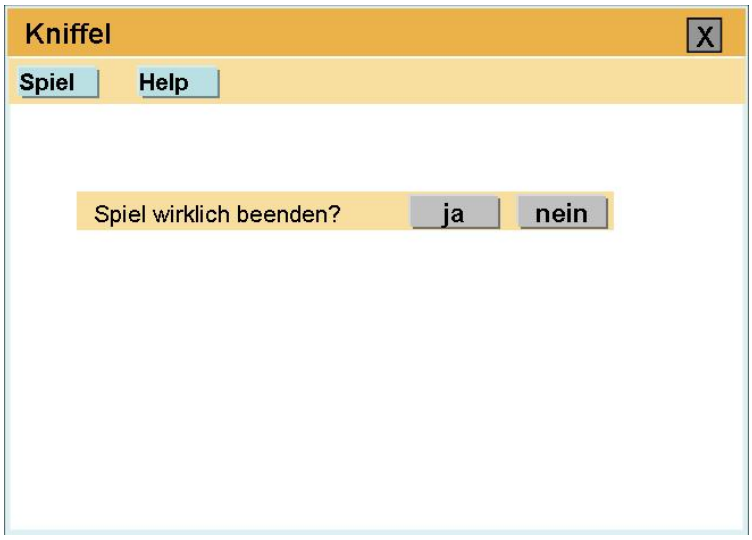
3 Bedienkonzept

Die Aufteilung des Bedienkonzepts erfolgt gemäß der Einteilung der Use-Cases in die Dialoge für „Kniffel starten“, „Kniffel Ende“, Kniffel spielen“.

3.1 Spiel beginnen und Spiel beenden

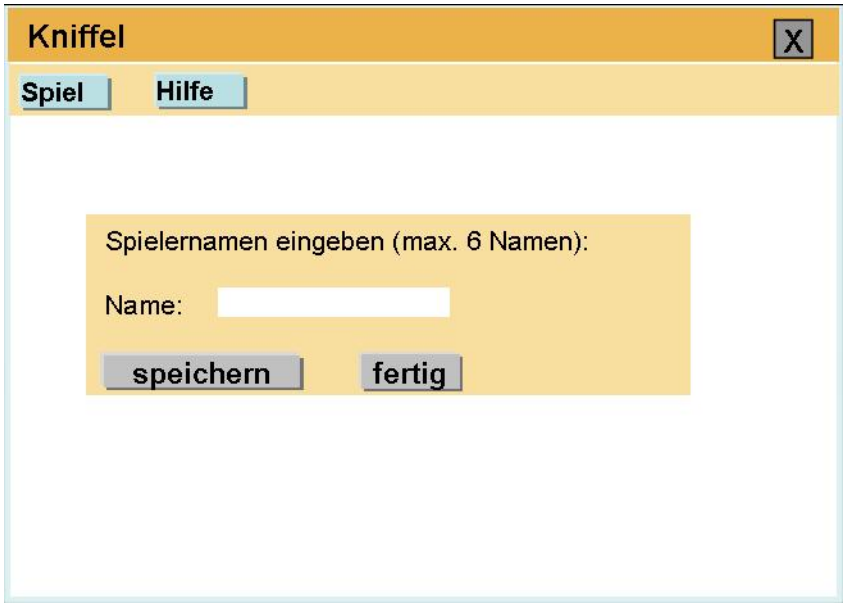
Nach dem Start des Spiels steht dem User ein Dialog zur Verfügung, der ihm die Möglichkeit bietet

- Neu …:
  ein neues Spiel zu beginnen, siehe Use-Case “Kniffel starten“
- Beenden :
  das Spiel zu beenden, hier wird nochmal nachgefragt, ob das Spiel tatsächlich beendet werden soll.



Spielernamen eingeben

Nach der Wahl des Menüpunktes “Neu“ wird den Spielern über einen Dialog die Möglichkeit gegeben ihre Namen in den Spielplan einzutragen.



Spielende

Nachdem alle Runden gespielt sind und der Sieger bestimmt ist, wird der Sieger gefragt, ob er sich in die Highscoreliste eintragen möchte.

Falls der Spieler mit ja geantwortet hat, wird er an der entsprechenden Stelle in die Highscoreliste eingefügt und die Highscoretabelle angezeigt.

3.2 Kniffel spielen

Für die Durchführung von “Kniffel spielen“ (siehe Use Case “Kniffel spielen“) wird der Standarddialog verwendet.

Im Dialog “Kniffel spielen“ wird angezeigt: die Runde, in der gerade gespielt wird, der Name des Spielers, der an der Reihe ist, und dessen Wurfanzahl.

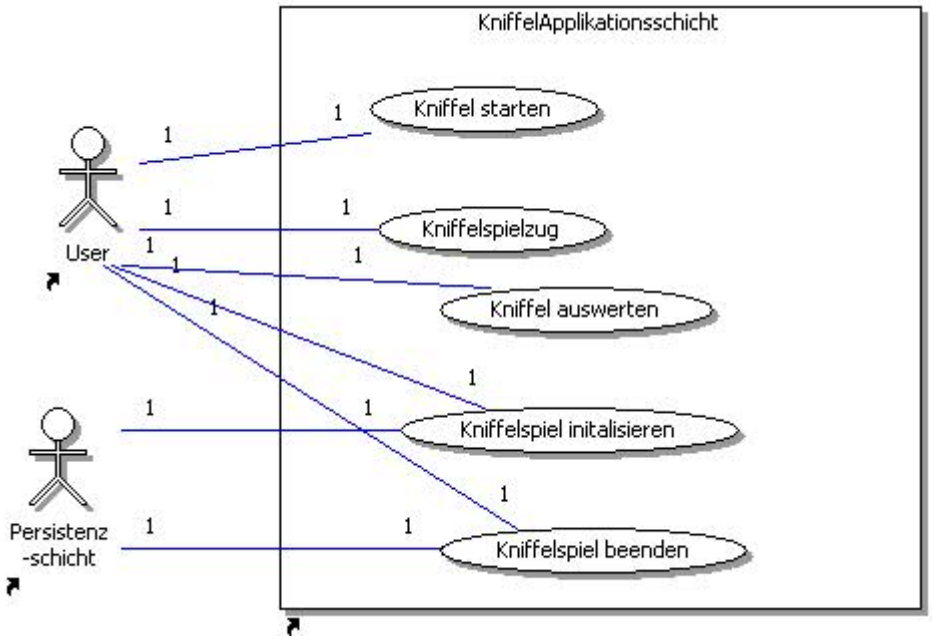
Zu Beginn klickt der Spieler, der an der Reihe ist, auf den Button “Würfeln“. Damit wird für alle 5 Würfel das Würfeln ausgelöst und anschließend die Augenzahl angezeigt. Nun kann der Spieler durch klicken auf die Buttons “Halten“ die Würfel auswählen, mit denen nicht mehr gewürfelt werden soll. Anschließend kann er wieder auf den Button “Würfeln“ drücken und den nächsten Wurf mit den nicht zurückgehaltenenen Würfeln starten oder, falls er mit den gewürfelten Punkten zufrieden ist auf einen der Punktebuttons drücken.Durch klicken auf einen Punktebutton werden die erzielten Punkte in das entsprechende Punktefeld eingetragen, falls die Augenzahlen den geforderten Würfelbildern entsprechen, andernfalls wird “0“ als Ergebnis eingetragen.
Nach dem zweiten Wurf kann der Spieler wieder aus allen 5 Würfeln diejenigen auswählen, die er halten möchte bzw. diejenigen mit denen er noch mal würfeln möchte. Nach dem 3. Wurf muss der Spieler sich für ein Punktefeld durch Drücken eines Punktebuttons entscheiden. Nach dem Eintrag der Punkte wird die Wurfanzahl auf 0 gesetzt und der nächste Spieler kommt an die Reihe. Dies wird in der Runde solange wiederholt bis alle Punktefelder ausgefüllt sind.

Sind alle 13 Runden gespielt, werden die Gesamtpunktzahlen und der Sieger ermittelt.

4 System Use-Case-Modell

Im Folgenden werden die zentralen Spielsituationen in Form von System Use Cases beschrieben. Jeder Use-Case umfasst neben einer kurzen Beschreibung des Anwendungsfalls und den beteiligten Akteuren eine Referenz auf die den Anwendungsfall betreffenden Anforderungen und den entsprechenden Anwendungsfall des Analysemodells.

4.1 System-Use-Case Diagramm



4.2 System-Use-Case Beschreibungen

4.2.1 System-Use-Case “Kniffel starten“ (SUC-1)

Akteure: User
Priorität: Hoch
Referenz: UC-1(Kniffel starten), F-1,F-2,F-23,
E1, E2, E3,E4
Beschreibung: Nach der Wahl des Menüpunktes „Neu …“, muss der Spielplan (E1) mit den einzelnen Punkte- (E2) und Summenfeldern (E3) initialisiert werden. Die User bekommen einen Dialog präsentiert, der es ihnen ermöglicht Namen einzugeben, diese werden in die Namensfelder (E4) eingetragen. Nach jeder Namenseingabe, kann der User den Eingabedialog beenden, durch klicken auf „fertig“ oder einen weiteren Namen eingeben und speichern durch click auf „speichern“, bis max. 6 Namen eingegen wurden.
Vorbedingung: Das Spiel ist initalisiert.

4.2.2 System-Use- Case “Kniffelspielzug“ (SUC-2)

Akteure: User
Priorität: Hoch
Referenz: UC-2 (Kniffel spielen),
F-3, F-4, F-5, F-6, F-7, F-8, F-9, F-19, F-20, F-21, F-22, F-24,F-25
E5, E6, E7, E2

Beschreibung: Den Usern wird im Feld „Runde:“ angezeigt in welcher Spielrunde sie sich gerade befinden, im Feld „Spieler:“ wird der Name des Users, der an der Reihe ist angezeigt, und im Feld Würfe, wird die Anzahl der bereits durchgeführten Würfe angezeigt. Der User klickt auf den Button “Würfeln“ und löst damit das Würfeln für alle 5 Würfel aus. Die erzielte Augenzahl wird durch die Würfel angezeigt. Nun kann der User durch klicken auf die Buttons “Halten“ die Würfel auswählen, mit denen nicht mehr gewürfelt werden soll. Anschließend kann er wieder auf den Button “Würfeln“ drücken und den nächsten Wurf mit den nicht zurückgehaltenenen Würfeln starten oder, falls er mit den gewürfelten Punkten zufrieden ist auf einen der Punktebuttons drücken.Durch klicken auf einen Punktebutton werden die erzielten Punkte in das entsprechende Punktefeld eingetragen, falls die Augenzahlen den geforderten Würfelbildern entsprechen, andernfalls wird “0“ als Ergebnis eingetragen. Nach dem Eintragen der Punkte werden die entsprechenden Summenfelder aktualisiert.
Nach dem zweiten Wurf kann der Spieler wieder aus allen 5 Würfeln diejenigen auswählen, die er halten bzw. diejenigen mit denen er noch mal würfeln möchte. Nach dem 3. Wurf muss der Spieler sich für ein Punktefeld durch Drücken eines Punktebuttons entscheiden, der Button „Würfeln“ wird gesperrt. Nach dem Eintrag der Punkte wird die Wurfanzahl auf 0 gesetzt und der nächste Spieler kommt an die Reihe. Dies wird in der Runde solange wiederholt bis alle Punktefelder ausgefüllt sind. Sind alle 13 Runden gespielt, werden die Gesamtpunktzahlen und der Sieger ermittelt.

offene Punkte:

4.2.3 System-Use Case “Kniffel auswerten“ (SUC-3)

Akteure: User
Priorität: Hoch
Referenz: F-10, F-11, F-12, F-13, F-14,F-27, F-28, F-29, F-30
E2, E3, E5, E8
Beschreibung: Sind alle 13 Runden (E5) gespielt, ist das Spiel zu Ende. Die verschiedenen Zwischensummen und die Gesamtsumme (E3) jedes Spielers müssen aus den Punktefeldern (E2) berechnet und der Sieger bestimmt werden. Sieger ist derjenige Spieler, der am meisten Punkte hat. Der Sieger wird gefragt, ob er in die Highscoreliste eingetragen werden möchte. Wenn ja, wird er nach einem Namen gefragt und entsprechend der Punktzahl in die Liste eingetragen.
offene Punkte:

4.2.4 System-Use-Case “Kniffelspiel initialisieren“ (SUC-4)

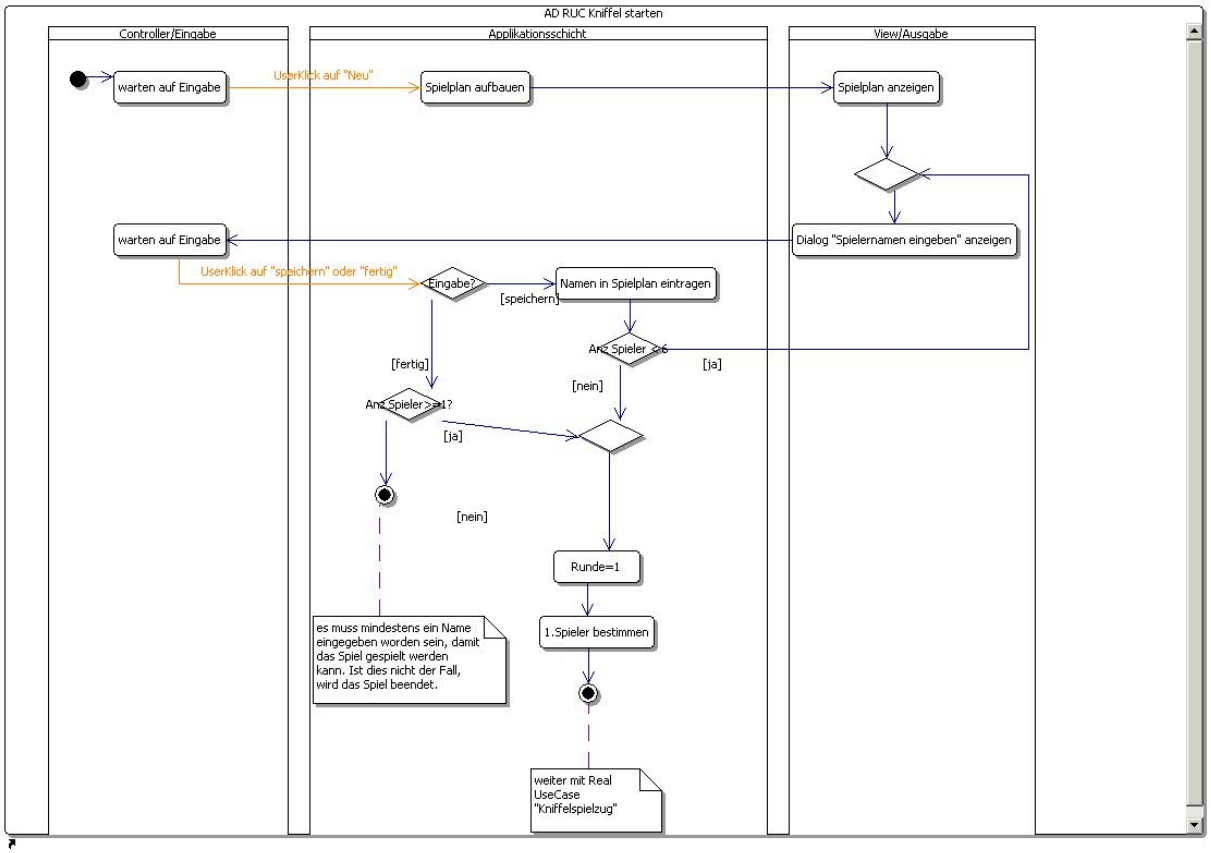
Akteure: Persistenzschicht
Priorität: Hoch
Referenz: F-26, F-31, E8
Beschreibung: Nach dem Programmstart müssen Klassen initalisiert und die Highscoreliste aus dem Textfile “Highscoreliste.txt“ eingelesen werden..
offene Punkte:

4.2.5 System-Use-Case “Kniffelspiel beenden“ (SUC-5)

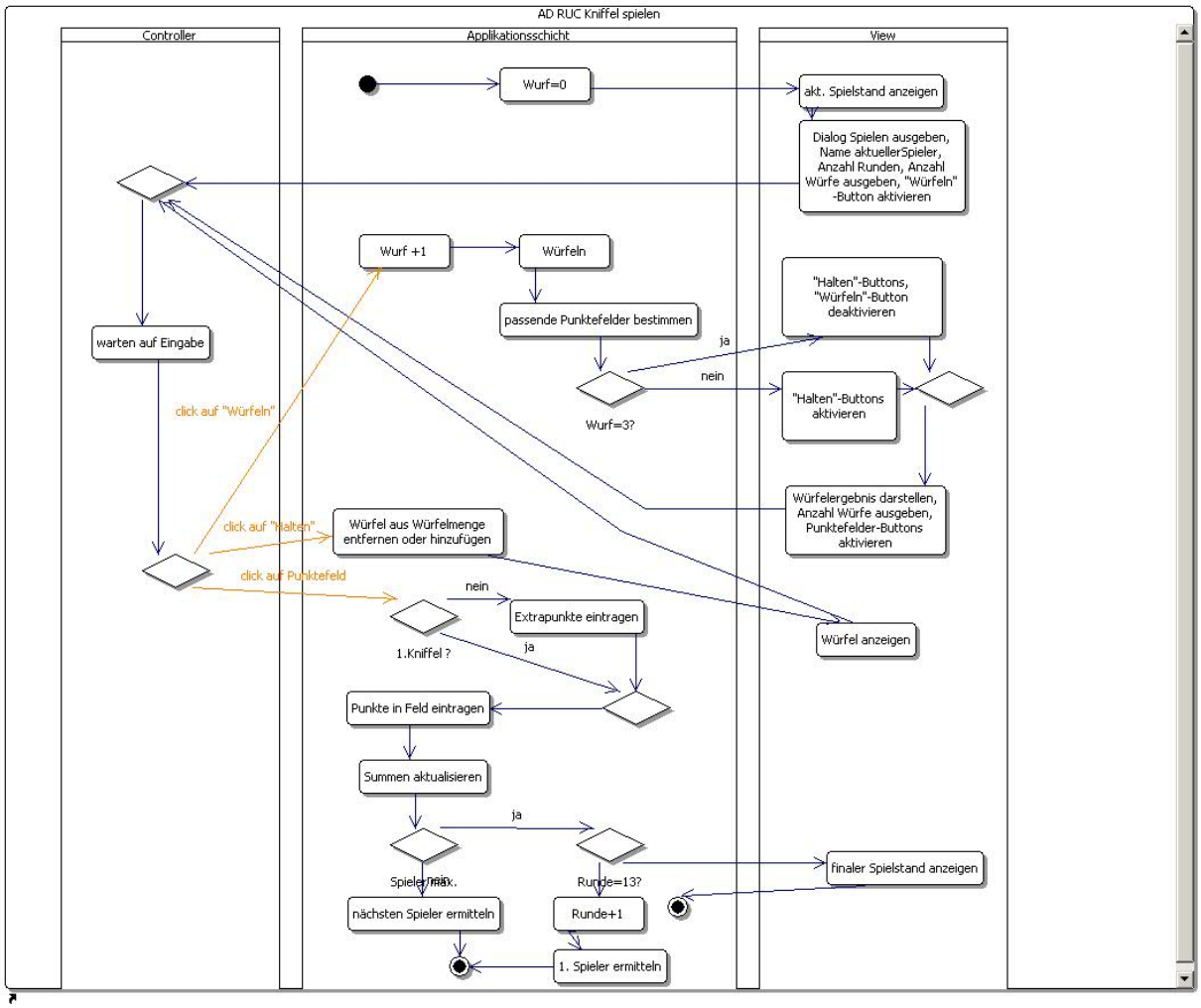
Akteure: User, Persistenzschicht
Priorität: Hoch
Referenz: F-26, F-31, E8
Beschreibung: Wenn der User das Spiel beendet durch klicken auf den Menüpunkt “beenden“ bzw. auf “ja“, dann wird die Highscoreliste in dem Textfile “Highscoreliste.txt“ abgespeichert und anschließend werden alle Resourcen freigegeben.
offene Punkte:

4.3 Detaillierte Beschreibung der System Use Cases durch Aktivitätsdiagramme

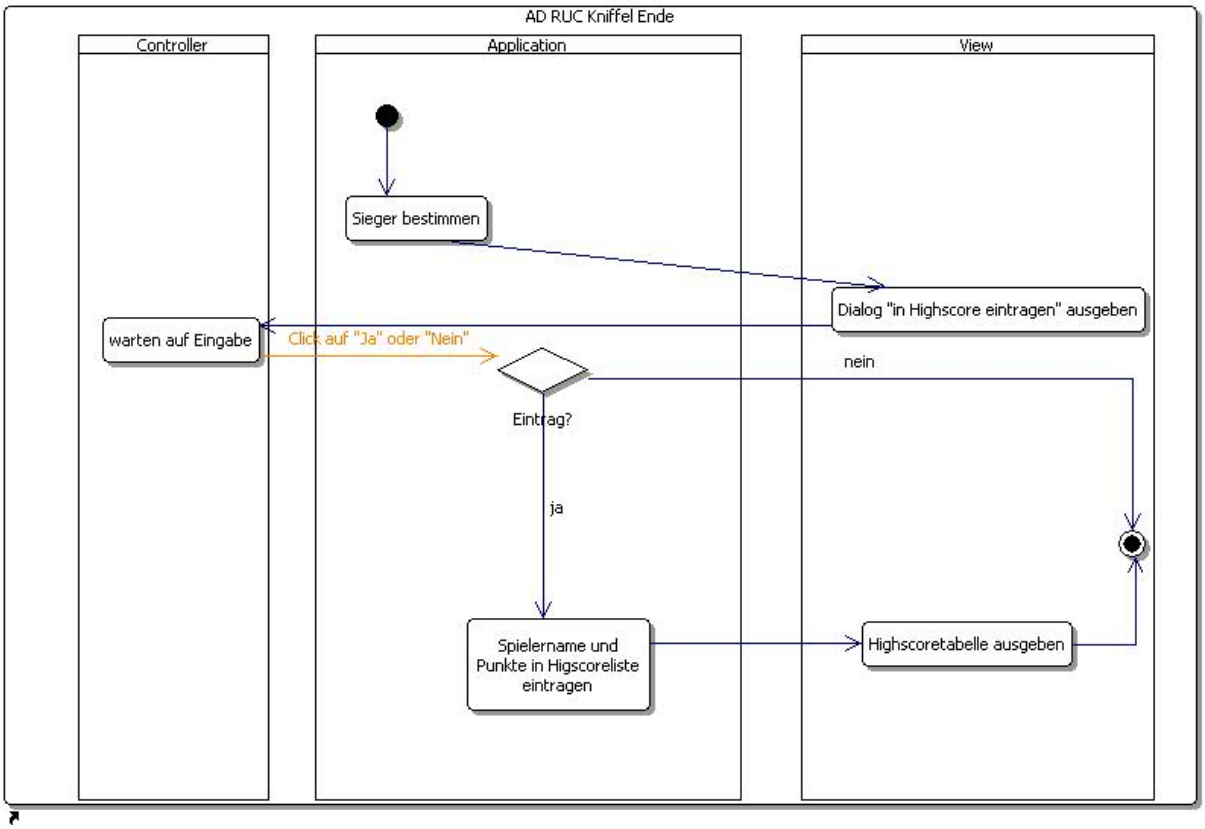
4.3.1 Aktivitätsdiagramm zu System Use Case “Kniffel starten“ (SUC-1)



4.3.2 Aktivitätsdiagramm zu System Use Case “Kniffelspielzug“ (SUC-2)



4.3.3 Aktivitätsdiagramm zu System Use Case “Kniffel Ende“



4.4 Systemoperationen

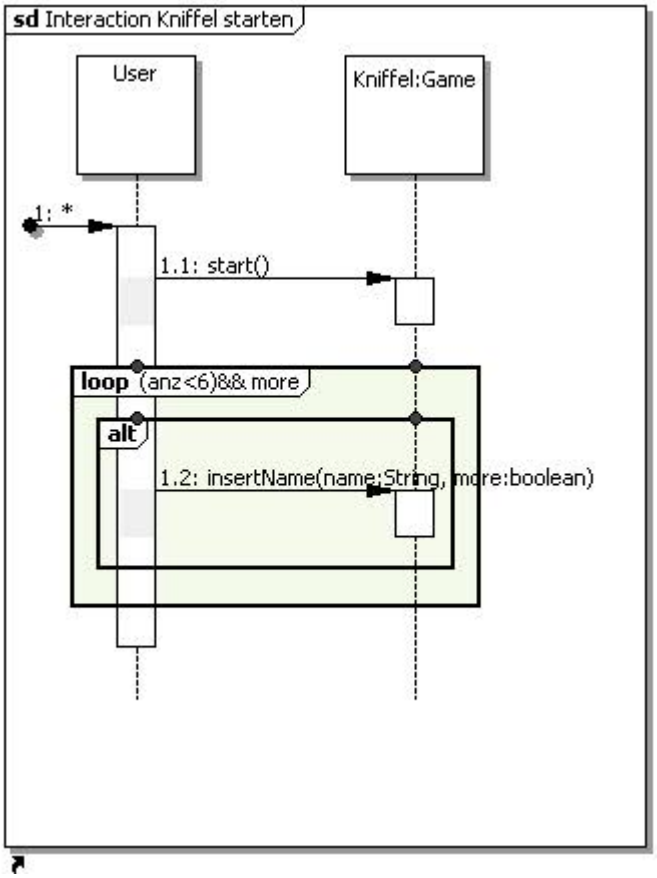
Systemsequenzdiagramme stellen Systemoperationen eines System Use Cases dar.

Eine Systemoperation wird aufgerufen als Reaktion auf ein Systemereignis, sie verändert den Zustand des Systems.
Ein Systemereignis ist eine Nachricht, die ein Akteur an das System sendet.

Beschreibung der Systemoperationen

Name:	Name der Operation
Verantwortlichkeit:	Aufgabe/Ziel der Operation
Referenzen:	Operation gehört zu welchem UseCase
Bemerkungen:	
Ausnahmen:	Ausnahmefälle, die zum Abbruch der Operation führen
Output:	Ausgaben an andere Aktuere
Vorbedingungen:	Anforderungen an den Anfangszustand, Bedingungen, die nicht geprüft werden, die aber gelten müssen, damit die Systemoperation erfolgreich ist, z.B. das Vorliegen der Ergebnisse von vorherigen Systemoperationen
Nachbedingungen:	beschreibt Änderungen zu Anfangszustand, es wird nicht die Aktion beschrieben, es wird nur ein ”Schnappschuss“ des Systems nachher mit einem vorher verglichen. Es wird nicht beschrieben wie der neue Zustand erreicht wird.

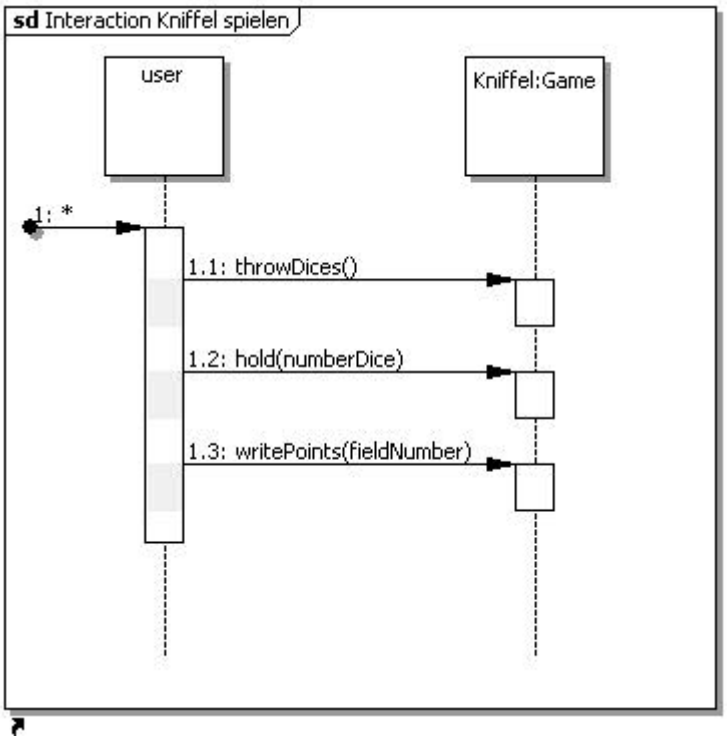
4.4.1 Systemoperationen im Rahmen des System Use Case “Kniffel starten“



Name:	**SYSOP-ST1 start():void**
Verantwortlichkeit:	Die Operation initalisiert ein neues Kniffelspiel, erzeugt den Spielplan. Wird die Operation „start“ während des Spielverlaufs aufgerufen, muss die bisherige Spielsituation bereinigt werden.
Referenzen:	System Use Case SUC-1 “Kniffel starten“
Bemerkungen:	Ein Neustart des Spiels ist jederzeit möglich.
Ausnahmen:	
Output:	
Vorbedingungen:	
Nachbedingungen:	Das Spiel ist initalisiert, die Container für die Spielerspalten stehen bereit. Das Spiel ist im Zustand „Start“.

Name:	**SYSOP-ST2 insertName(name: String, more: boolean):void**
Verantwortlichkeit:	Die Operation legt eine neue Spalte in der Spieltabelle an und trägt den Namen in die Spalte ein.
Referenzen:	System Use Case SUC-1 “Kniffel starten“
Bemerkungen:	
Ausnahmen:	es wurden keine Zeichen im Namensfeld eingegeben
Output:	
Vorbedingungen:	Das Spiel ist initialisiert und es sind max. 5 Spieler eingetragen.
Nachbedingungen:	Eine neue Spalte in der Spieltabelle wurde angelegt, solange noch nicht 6 Spieler eingetragen wurden und der Parameter more = true ist, bleibt das System in diesem Zustand. Ist das Anlegen der Spielerspalten abgeschlossen, d.h es sind 6 Namen eingetragen oder der Parameter more = false, und es sind mindestens 1 Spielername eingetragen, wechselt das System in den Zustand „play“.

4.4.2 Systemoperationen im Rahmen des System Use Case „Kniffelspielzug“



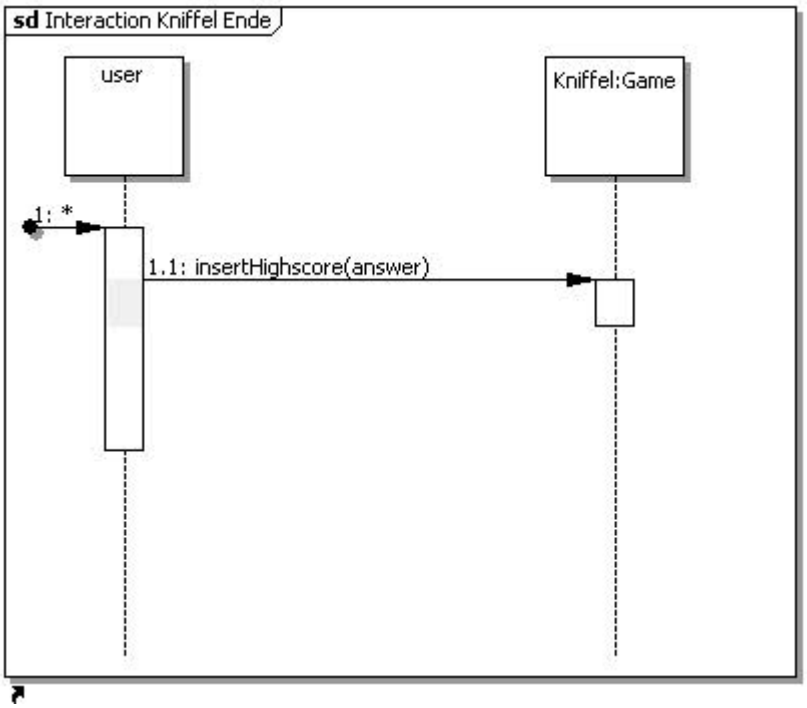
Name:	**SYSOP-SP1 throwDices():void**
Verantwortlichkeit:	erzeugt Augenzahlen für die Würfel mit denen gewürfelt werden soll durch einen Zufallsgenerator
Referenzen:	System Use Case SUC-1 “Kniffel spielen“
Bemerkungen:	
Ausnahmen:	es sind keine Würfel mehr zum Würfeln vorhanden
Output:	gewürfelte Augenzahlen
Vorbedingungen:	die Namenseingabe ist abgeschlossen, es ist mindestens ein Name eingetragen, das System befindet sich im Zustand „play“, es wurden noch keine 3 Würfe für den aktuellen Spieler getätigt und es haben noch nicht alle Spieler in allen Runden gespielt
Nachbedingungen:	die gewählten Würfel haben eine neue Augenzahl, die Anzahl der Würfe wurde um 1 erhöht.

Name:	**SYSOP-SP2 hold(numberDice:IDicenumber):void**
Verantwortlichkeit:	je nachdem in welchem Set der Würfel ist, wird der gewählte Würfel aus dem Set der Würfel, mit denen gewürfelt werden soll, entfernt oder hinzugefügt
Referenzen:	System Use Case SUC-1 “Kniffel spielen“
Bemerkungen:	
Ausnahmen:	
Output:	
Vorbedingungen:	
Nachbedingungen:	die Sets der Würfel haben sich entsprechend geändert

Name:	**SYSOP-SP3 writePoints(fieldnumber:IFieldnumber):void**
Verantwortlichkeit:	die Punkte von den passenden Würfeln werden in das gewählte Feld eingetragen, bzw. die eingebene Punktezahl, die Summen des

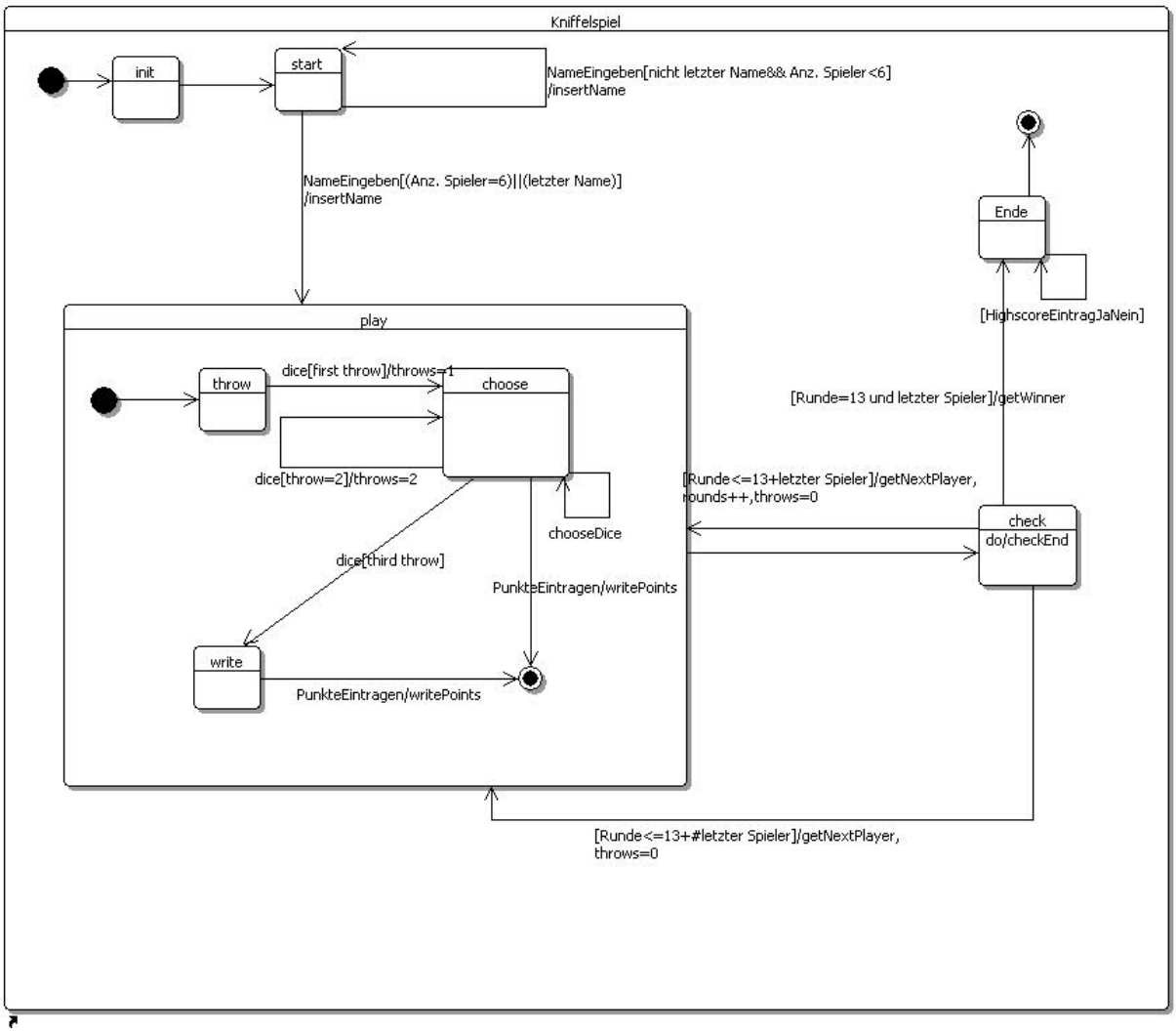
	entsprechenden Bereichs neu berechnet, der nächste Spieler wird bestimmt. Falls eine neue Runde beginnt, die Anzahl der Runden erhöht.
Referenzen:	System Use Case SUC-1 “Kniffel spielen“
Bemerkungen:	
Ausnahmen:	
Output:	
Vorbedingungen:	ein Feld wurde ausgewählt
Nachbedingungen:	die Punkte sind eingetragen, die Summen aktualisiert. Falls noch nicht alle Runden gespielt sind, ist der nächste Spieler ist an der Reihe, das Spiel bleibt im Zustand spielen. Sind alle Runden gespielt, wechselt der Zustand nach „end“

4.4.3 Systemoperationen im Rahmen des SystemUseCase „Kniffelende“



Name:	**SYSOP-E1 insertHighscore(answer:boolean):void**
Verantwortlichkeit:	Ist der Parameter answer = true wird der Name und die erreichte Punktzahl des Spielers in die Highscoreliste an der der Punktzahl entsprechenden Stelle eingetragen
Referenzen:	System Use Case SUC-1 “Kniffel Ende“
Bemerkungen:	
Ausnahmen:	
Output:	
Vorbedingungen:	Spielzustand = „end“, Highscoreliste muss initialisiert sein, das Spiel muss beendet sein, d.h. alle Runden müssen gespielt sein, der Sieger muss bestimmt sein
Nachbedingungen:	es wurde ein Eintrag in die Highscoreliste erzeugt

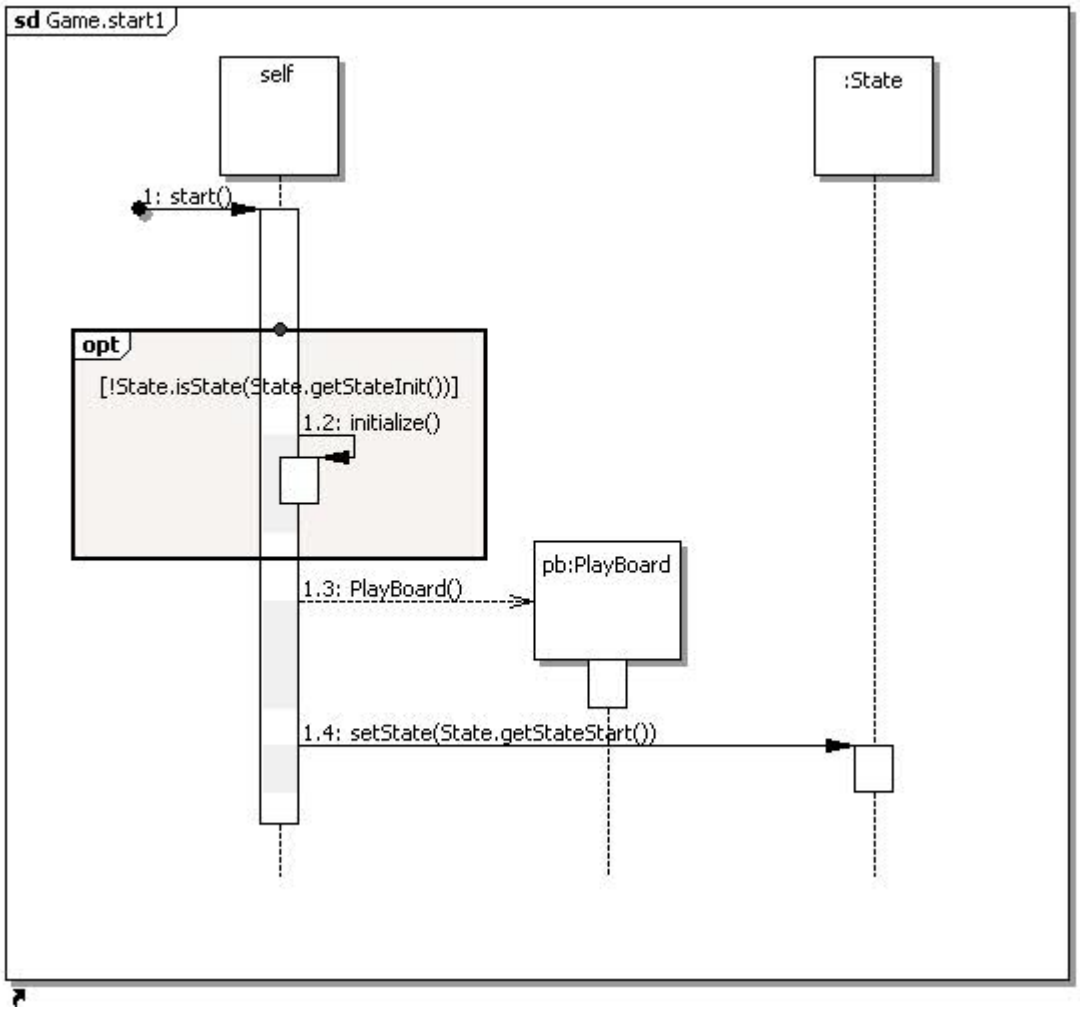
5 Zustandsmodell



6 Applikationssschicht

6.1 Die Funktionen des Kniffelstarts

6.1.1 SYSOP-ST1 start():void



Klassen:

Game entspricht der Klasse Spiel aus dem Analysemodell

Operationen:

start():void

Parameter:	keine
Rückgabewert	keiner
Vorbedingung:	Das Spiel ist erzeugt.
Nachbedingung:	Der Container für die Spieler ist erzeugt, das Spiel befindet sich im Zustand „start“.
Ausnahmen:	
Beschreibung:	Nach dem Start wird geprüft, ob das Spiel neu ist (!State.isState(State.getStateInit()) oder nur neu gestartet werden soll. Falls das Spiel neu gestartet werden soll, muss der Initialzustand hergestellt werden. Danach wird ein leerer Container für die Spalten angelegt und das Spiel in den Zustand „start“ versetzt.
Gui:	Die Gui muss den Dialog „Spielernamen eingeben“ anzeigen.

initialize():void

Parameter: keine
Rückgabewert: keiner
Vorbedingung: Das Spiel ist erzeugt.
Nachbedingung: Das Spiel ist wieder in der Ausgangsituation.
Ausnahmen:
Beschreibung: Den Container für die Spalten der Spieler löschen, die Anzahl der Runden und die Anzahl der Würfe auf 0 setzen

PlayBoard Container, in dem die Spalten für die Spieler gespeichert werden.

Operationen:

constructor() **erzeugt einen leeren Container**

Parameter: keiner
Rückgabewert: der erzeugte Container
Vorbedingung:
Nachbedingung: ein leerer Container ist erzeugt.
Ausnahmen:
Beschreibung:

State die möglichen Zustände (stat. Attibute dieser Klasse)

Operationen:

static setState(s:State):void Setzt den internen Zustand gemäß dem übergebenen Zustand „s“

Parameter:
Input: s:State der neue Zustand, in den das System wechseln soll
Rückgabewert: keiner
Vorbedingung:
Nachbedingung: der aktuelle Zustand ist „s“
Ausnahmen:
Beschreibung:

static getStateInit():State Diese Operation liefert den Zustand “init”

Parameter: keine
Rückgabewert: **State** der Zustand „init“
Vorbedingung:
Nachbedingung:
Ausnahmen:
Beschreibung:

static getStateStart():State Diese Operation liefert den Zustand „start“

Parameter: keine
Rückgabewert: **State** der Zustand „start“
Vorbedingung:
Nachbedingung:
Ausnahmen:
Beschreibung:

static getStateChoose():State Diese Operation liefert den Zustand „choose“

Parameter: keine
Rückgabewert: **State** der Zustand „choose“
Vorbedingung:
Nachbedingung:
Ausnahmen:
Beschreibung:

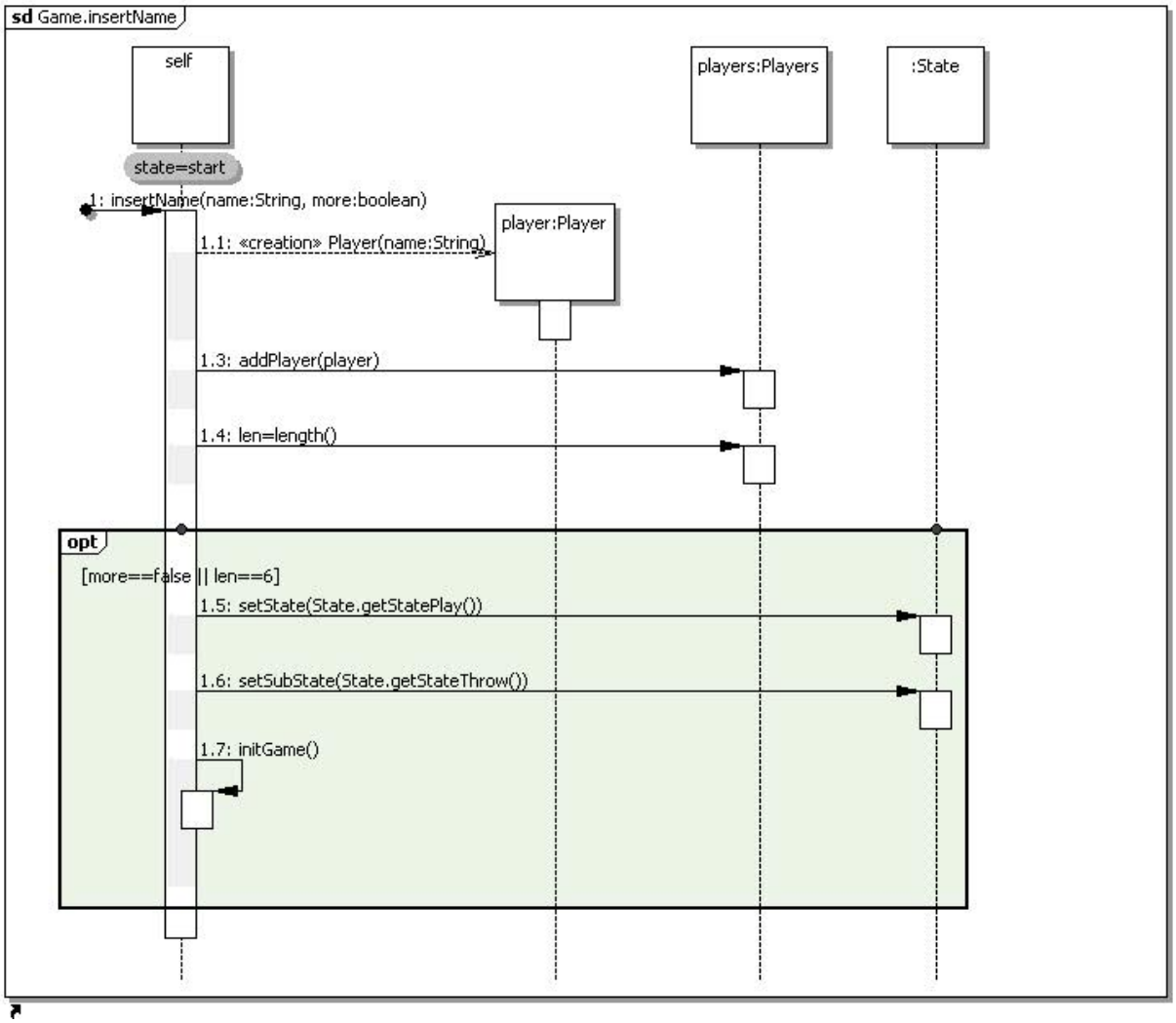
static getStatePlay():State Diese Operation liefert den Zustand „play“

Parameter: keine

Rückgabewert: **State** der Zustand „play“
Vorbedingung:
Nachbedingung:
Ausnahmen:
Beschreibung:

static isState(s:State):boolean Diese Operation liefert das Ergebnis des Vergleichs des aktuellen Zustands mit dem abgefragten Zustand „s“.
Parameter: keine
Rückgabewert: **true oder false**
Vorbedingung:
Nachbedingung:
Ausnahmen:
Beschreibung:

6.1.2 SYSOP-ST2 insertName(name: String, more: boolean):void



Klassen:

Game entspricht der Klasse Spiel aus dem Analysemodell

Operationen:

insertName(name:String,more:boolean):void Beschreibung siehe 4.4.1

Parameter:

Input: name:String Name des Spielers
more:boolean Flag, das anzeigt, ob weitere Spieler angelegt werden sollen

Rückgabewert keiner

Vorbedingung: Das Spiel ist erzeugt und befindet sich im Zustand „start“, ein Container für die Spielerspalten ist angelegt (pb)

Nachbedingung: Eine Spalte mit dem Namen des Spielers wurde erzeugt und in das Playboard pb eingetragen. Das Spiel befindet sich im Zustand „start“ oder „play“.

Ausnahmen: Es wurden keine Zeichen mit dem Parameter name übergeben und more = true,
der Parameter more = true und die Anzahl der Spalten = 6.

Beschreibung: Das Spiel befindet sich im Zustand “start“, mit dem übergegebenen Namen wird eine neue Spielerspalte (player:Player) erzeugt und in die Liste der Spieler aufgenommen (add (player)) , danach wird die Anzahl der Spieler bestimmt und der Parameter more getestet, um den Folgezustand festzulegen.

initGame( ):void Anzahl Runden=1, Anzahl Würfe=1 setzen, ersten Spieler ermitteln

Parameter:

Input:

Rückgabewert

Vorbedingung:

Nachbedingung:

Ausnahmen:

Beschreibung:

Gui:

Column Klasse, die die lnformation verwaltet, die mit einem Spieler verbunden ist.

Operationen:

constructor(name:String) erzeugt eine Spielerspalte mit dem eingegebenen Namen

Parameter:

Input: name:String Name des Spielers

Rückgabewert: die erzeugte Spalte

Vorbedingung:

Nachbedingung: eine Spalte ist erzeugt.

Ausnahmen:

Beschreibung:

State die möglichen Zustände (stat. Attibute dieser Klasse)

Operationen:

static setState(s:State):void Setzt den internen Zustand gemäß dem übergebenen Zustand „s“

Parameter:

Input: s:State der neue Zustand, in den das System wechseln soll

Rückgabewert: keiner

Vorbedingung:

Nachbedingung: der aktuelle Zustand ist „s“

Ausnahmen:

Beschreibung:

static setSubState(s:State):void Setzt den internen Zustand gemäß dem übergebenen Zustand „s“

Parameter:
Input: s:State der neue Zustand, in den das System wechseln soll
Rückgabewert: keiner
Vorbedingung:
Nachbedingung: der aktuelle Zustand ist „s“
Ausnahmen:
Beschreibung:

static getStatePlay():State Diese Operation liefert den Zustand „choose“

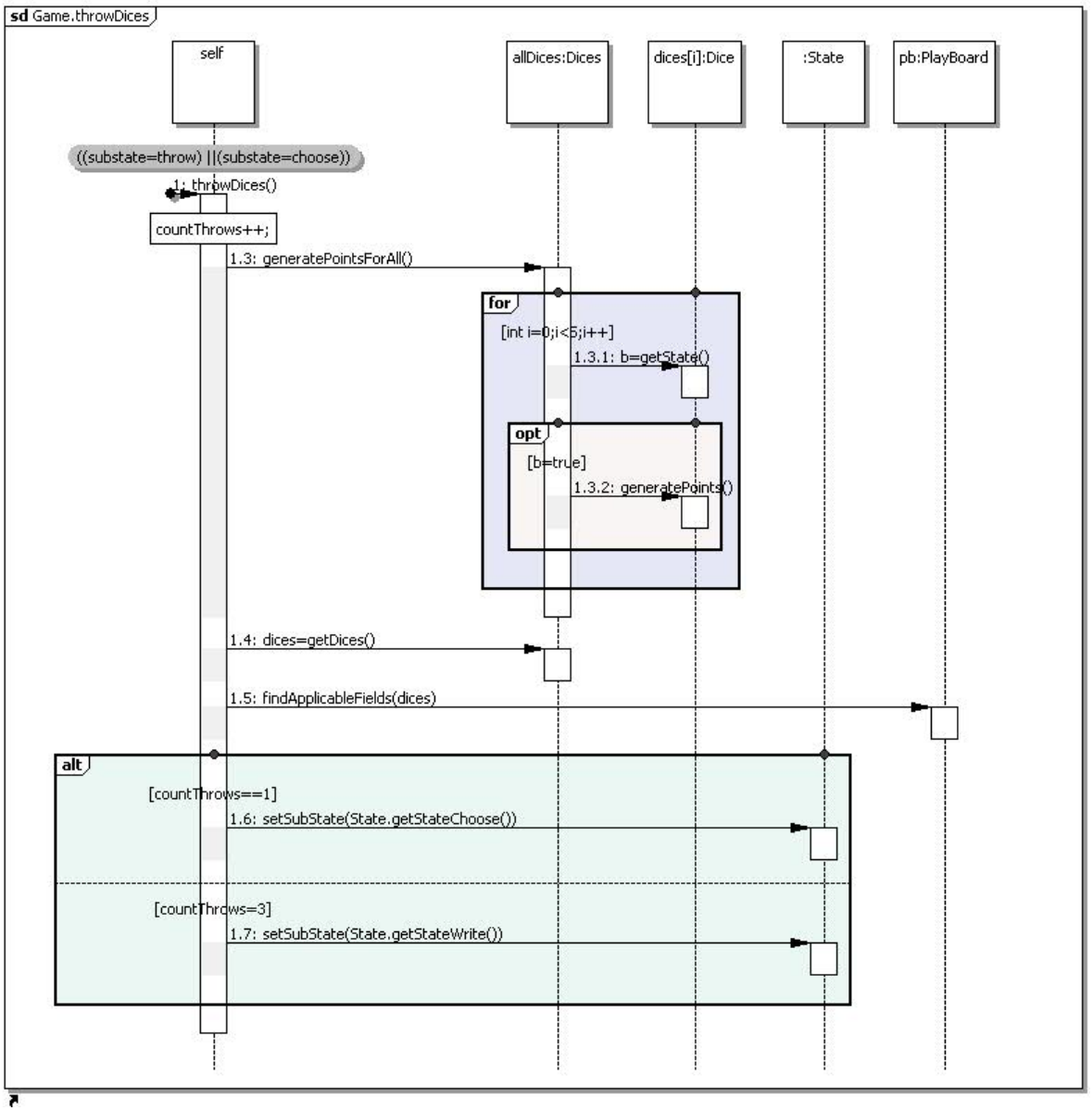
Parameter: keine
Rückgabewert: **State** der Zustand „play“
Vorbedingung:
Nachbedingung:
Ausnahmen:
Beschreibung:

static getStateThrow():State Diese Operation liefert den Zustand „throw“

Parameter: keine
Rückgabewert: **State** der Zustand „throw“
Vorbedingung:
Nachbedingung:
Ausnahmen:
Beschreibung:

6.2 Die Funktionen des Kniffelspielzugs

6.2.1 SYSOP-SP1 throwDices():void



Klassen:

Dices Container, in dem alle Würfel gespeichert werden.

Operationen:

getDice(diceNumber:int):Dice

Parameter:

Input: ***diceNumber:int*** die Nummer des Würfels

Rückgabewert: ***Dice*** der Würfel

Vorbedingung:

Nachbedingung:

Ausnahmen:

Beschreibung: gibt den Würfel mit der Nummer diceNumber aus dem Container zurück

getDices():Dice[]

Parameter:

Input:

Rückgabewert: ***Dice[]*** die Würfel

Vorbedingung:
Nachbedingung:
Ausnahmen:
Beschreibung: gibt die Würfel aus dem Container zurück

Dice enthält alles was zu einem Würfel gehört, Status, gewürfelte Augenzahl

Operationen:

getState():boolean

Parameter: keiner
Input:
Rückgabewert: true oder false
Vorbedingung:
Nachbedingung:
Ausnahmen:
Beschreibung: State = true → mit dem Würfel kann gewürfelt werden
State = false → die Augenzahl des Würfels soll nicht verändert werden

generatePoints():void

Parameter: keiner
Input:
Rückgabewert:
Vorbedingung: Status des Würfels muss true sein
Nachbedingung: der Würfel hat einen neuen Wert **points**
Ausnahmen:
Beschreibung: es wird eine neue Augenzahl durch einen Zufallsgenerator erzeugt

PlayBoard Container, in dem die Spalten für die Spieler gespeichert werden.

Operationen

findApplicableFields(allDices:Dices):void

Parameter:
Input: **allDices:Dices**
Rückgabewert: keiner
Vorbedingung:
Nachbedingung: alle noch nicht ausgefüllten Punktefelder, haben den Wert possible = true oder = false
Ausnahmen:
Beschreibung: es wird geprüft, für welches Punktefeld, das noch keinen festeingetragenen Wert hat, die gewürfelte Punktkombination zulässig ist.

State die möglichen Zustände (stat. Attibute dieser Klasse)

Operationen:

static setSubState(s:State):void Setzt den internen Zustand gemäß dem übergebenen Zustand „s“

Parameter:
Input: s:State der neue Zustand, in den das System wechseln soll
Rückgabewert: keiner
Vorbedingung:
Nachbedingung: der aktuelle Zustand ist „s“
Ausnahmen:
Beschreibung:

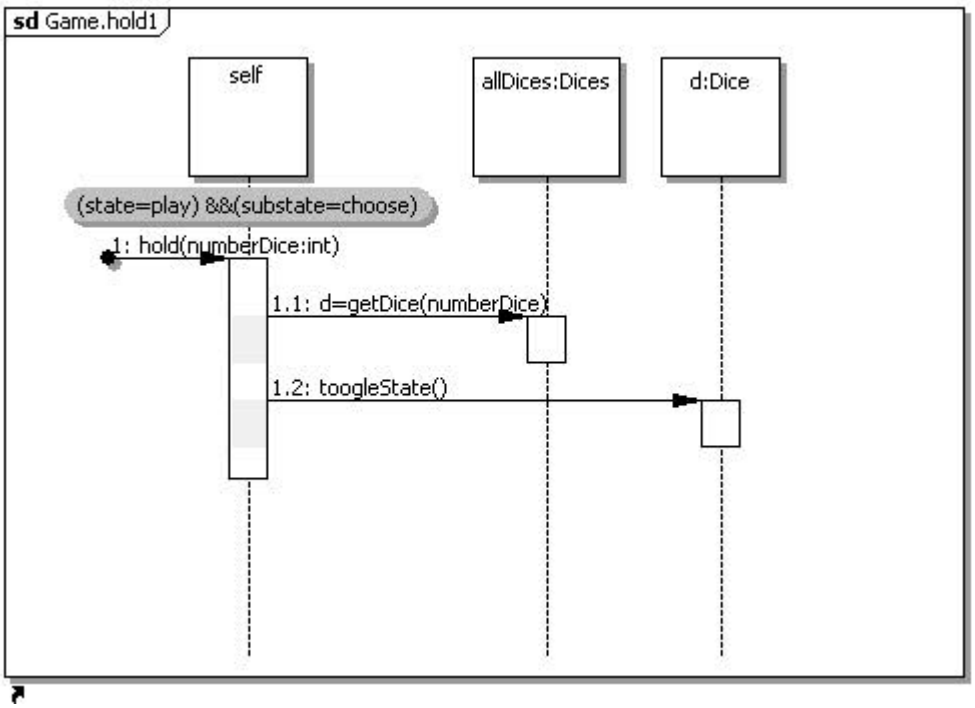
static getStateChoose():State Diese Operation liefert den Zustand „choose“

Parameter: keine
Rückgabewert: **State** der Zustand „choose“
Vorbedingung:
Nachbedingung:

Ausnahmen:
Beschreibung:

static getStateWrite():State Diese Operation liefert den Zustand „write“
Parameter: keine
Rückgabewert: **State** der Zustand „write“
Vorbedingung:
Nachbedingung:
Ausnahmen:
Beschreibung:

6.2.2 SYSOP-SP2 hold(numberDice:int):void



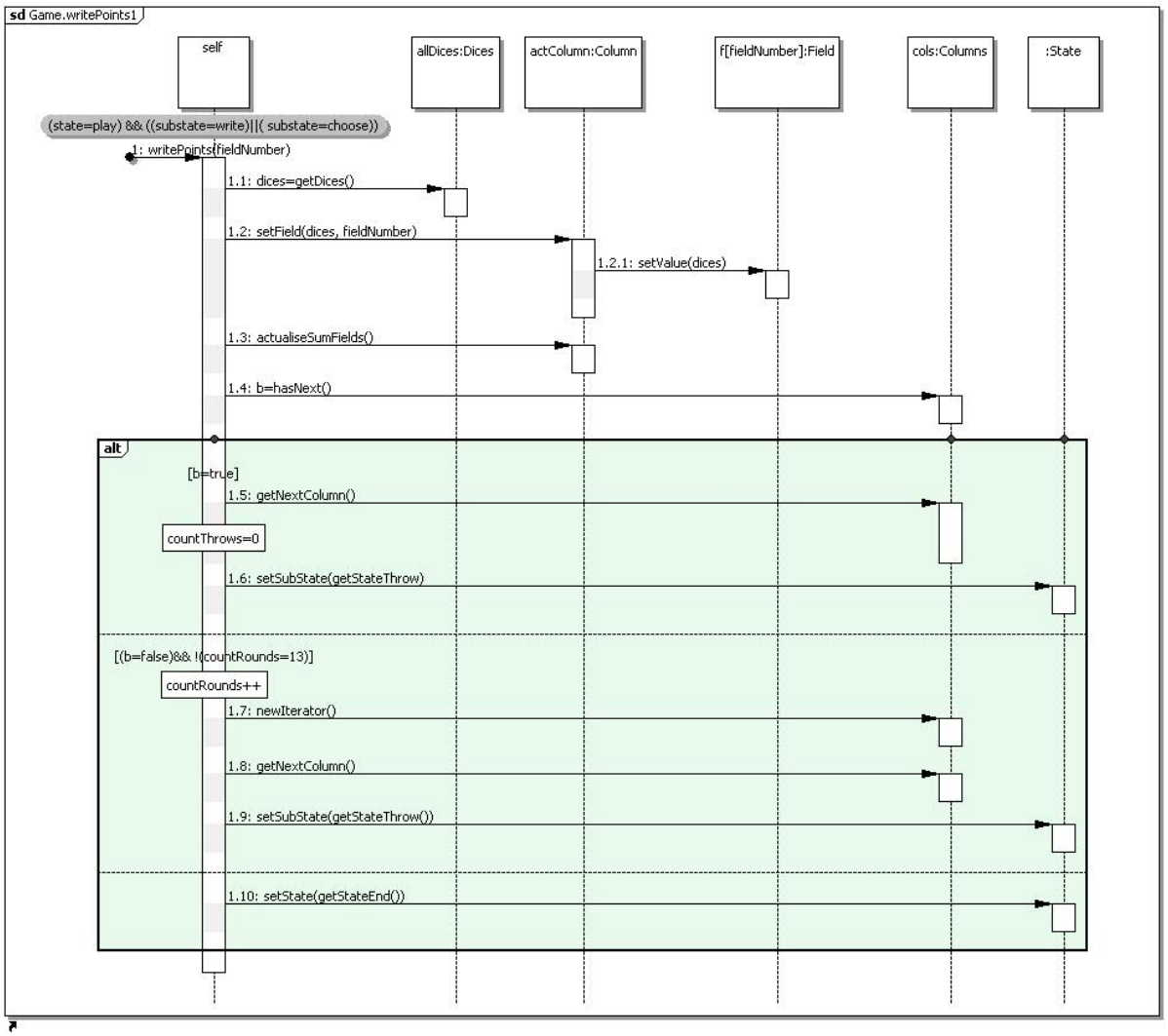
Klassen:

Dices Container, in dem alle Würfel gespeichert werden.
Operationen:
getDice(numberDice:int):Dice
Parameter:
Input: ***numberDice:int*** die Nummer des Würfels
Rückgabewert: ***Dice*** der Würfel
Vorbedingung:
Nachbedingung:
Ausnahmen:
Beschreibung: gibt den Würfel mit der Nummer diceNumber aus dem Container zurück

Dice enthält alles was zu einem Würfel gehört, Status, gewürfelte Augenzahl
Operationen:
toogleState():void
Parameter: keiner
Input:
Rückgabewert: keiner
Vorbedingung:
Nachbedingung: der Zustand des Würfels hat sich geändert von true → false, von false → true

Ausnahmen:
Beschreibung:

6.2.3 SYSOP-SP3 writePoints(fieldnumber:IFieldnumber):void



Klassen:

Dices Container, in dem alle Würfel gespeichert werden.

Operationen:

getDices():Dice[]

Parameter:

Input:

Rückgabewert: ***Dice[]*** Array aller Würfel

Vorbedingung:

Nachbedingung:

Ausnahmen:

Beschreibung: gibt den Würfel mit der Nummer diceNumber aus dem Container zurück

Column Klasse, die die Information verwaltet, die mit einem Spieler verbunden ist.

Operationen:

setField(dices:Dices, fieldNumber:IFieldNumber) :void

Parameter:
- Input: dices alle Würfel
  fieldNumber das Feld das mit dem Würfelergebnis besetzt werden soll
- Rückgabewert: keiner
- Vorbedingung:
- Nachbedingung: der Zustand des Würfels hat sich geändert von
  true → false, von false → true
- Ausnahmen:
- Beschreibung: das Feld mit der Feldnummer wird mit dem Ergbenis der Würfel belegt.

actualiseSumFields():void

- Parameter: keine
  Input:
- Rückgabewert: keine
- Vorbedingung: neuer Wert wurde in ein Punktefeld eingetragen
- Nachbedingung: die Summenfelder sind aktualisiert
- Ausnahmen:
- Beschreibung: es werden die Summenfelder neu berechnet

Field Klasse, die die Informationen verwaltet, die zu Punktefeldern gehören

Operationen:

setValue(dices:Dices):void

Parameter:
- Input: **dices:Dices**
- Rückgabewert:
- Vorbedingung:
- Nachbedingung: der Wert im Feld wurde gesetzt
- Ausnahmen:
- Beschreibung:

6.3 Die Funktionen des Kniffelende

6.3.1 SYSOP_E1 insertHighscore(answer:boolean):void

Noch nicht ausgearbeitet.