Dokumentation
zur Studienarbeit im Fach
Praktikum Softwareengineering

- Entwicklung einer Lernsoftware -
"myMemo"

von Susanne Kießling
Dozent: Prof. Dr. Martin Thost

13. Juli 2013

Inhaltsverzeichnis

1 Anforderungen des Kunden

Die Diakonie Hochfranken möchte das Angebot an Lernsoftware in ihren Kindergärten und Kindertagesstätten erweitern. Es soll eine Lernsoftware entwickelt werden, die Gedächtnistraining und die Erweiterung des Wortschatzes spielerisch umsetzt.

1.1 Systemvoraussetzungen

Aktuell verfügt der Kunde über Einzelplatz-PC's, die in den Jahren 2005 bis 2010 angeschafft wurden. Als Betriebssystem kommt Linux und Windows zum Einsatz.

1.2 Zielgruppe

Die Besucher der Einrichtung im Alter von fünf bis zehn Jahren bilden die Zielgruppe der Software.

2 Lastenheft

2.1 Zielbestimmung

Mit der Lernsoftware wird die Möglichkeit geschaffen, das pädagogisch wertvolle Prinzip des klassischen Memory-Spiel auf eine digitale Plattform zu übertragen. Zusätzlich zum Gedächtnistraining trägt die Vokabelfunktion zur Erweiterung des Wortschatzes bei. Die Führung einer Highscore ermöglicht den Vergleich der erreichten Punkte und steigert die Motivation der Benutzer, eine bessere Platzierung zu erreichen. Das erhöht wiederum den Lerneffekt. Neben dem Spielen an sich wird der Umgang mit dem Computer spielerisch erlernt. Vor Spielbeginn sind Attribute festzulegen und Meldungen der Software zu beachten. Das schult zusätzlich die Logik. Der 2-Player-Modus unterstützt außerdem die Kommunikation mit anderen Spielern.

2.2 Produkteinsatz

Die Lernsoftware kommt in den Kindertagesstätten der Diakonie Hochfranken zum Einsatz. Anwender der Software sind Besucher der Kindertagesstätte im Alter von fünf bis zehn Jahren.

2.3 Produktfunktionen

Priorität 1:

\LF10\	Spieler neu erstellen
\LF11\	Spieler laden
\LF12\	Spielerdaten ändern
\LF20\	Anzahl der Spieler wählen
\LF30\	Thema wählen
\LF40\	Spielfeldgröße wählen
\LF50\	Spiel starten
\LF60\	Highscore anzeigen

Priorität 2:

\LF70\	Urkunde drucken
\LF80\	Vokabeltraining

\LF90\ Audiodaten abspielen

2.4 Produktdaten

\LD10\ Spielerdaten

\LD20\ Highscoredaten

2.5 Produktleistungen

2.6 Qualitätsanforderungen

Funktionalität:	gut
Zuverlässigkeit:	sehr gut
Benutzbarkeit:	gut
Effizienz:	normal
Änderbarkeit:	normal
Portierbarkeit:	sehr gut
Spassfaktor:	sehr gut

2.7 Ergänzungen

Die Umsetzung der Software erfolgt in der Programmiersprache Java. Da der Kunde Linux und Windows als Betriebssystem einsetzt stellt dies die notwendige Portierbarkeit sicher.

3 Aufwandskalkulation

3.1 Function-Point-Analyse

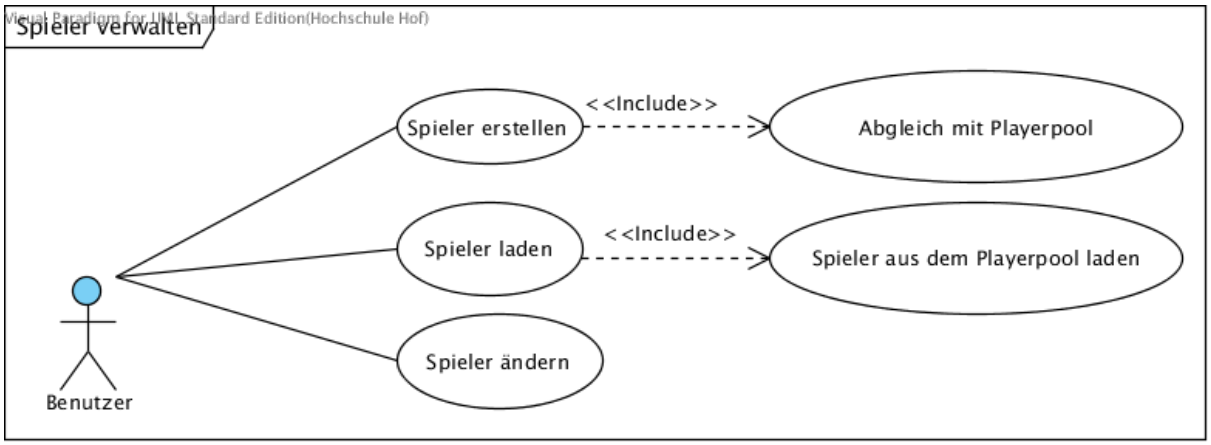
3.2

Function Point - Methode					
Kategorie	**Anzahl**	**Klassifizierung**	**Gewichtung**	**Zeilensumme**	
Eingaben	3	Einfach	3	9	
	4	Mittel	4	16	
	0	Komplex	6	0	
Abfragen	2	Einfach	3	6	
	1	Mittel	4	4	
	2	Komplex	6	12	
Ausgaben	8	Einfach	4	32	
	2	Mittel	5	10	
	3	Komplex	7	21	
Datenbestände	0	Einfach	7	0	
	1	Mittel	10	10	
	1	Komplex	15	15	
Referenzdaten	1	Einfach	5	5	
	0	Mittel	7	0	
	0	Komplex	10	0	
Summe			E1	140	
Einflußfaktoren		1 Verflechtung mit anderen Anwendungs-systemen (0-5)		0	
(ändern den Function Point-Wert um +/- 30%)		2 Dezentrale Daten, dezentrale Verarbeitung (0-5)		0	
		3 Transaktionsrate (0-5)		0	
		4 Verarbeitungslogik			
		A Rechenoperationen (0-10)		0	
		B Kontrollverfahren (0-5)		1	
		C Ausnahmeregelungen (0-10)			
		D Logik (0-5)		3	
		5 Wiederverwendbarkeit (0-5)		1	
		6 Datenbestandskonver-tierungen (0-5)		0	
		7 Anpaßbarkeit (0-5)		2	
Summe der 7 Einflüsse			E2	7	
Faktor Einflußbewertung = (E2/100) + 0,7			E3	0,77	
Bewertete Function Points: E1 * E3				107,8	

4 Use Case Diagramme

Folgend wird die Software als Use Case Diagramme dargestellt. Use Cases geben die Außensicht des Systems wieder. Es werden typische Funktionalitäten beschrieben, die der Benutzer mit dem System ausführt.

4.1 Spieler verwalten - \LF10\,\LF11\,\LF12\



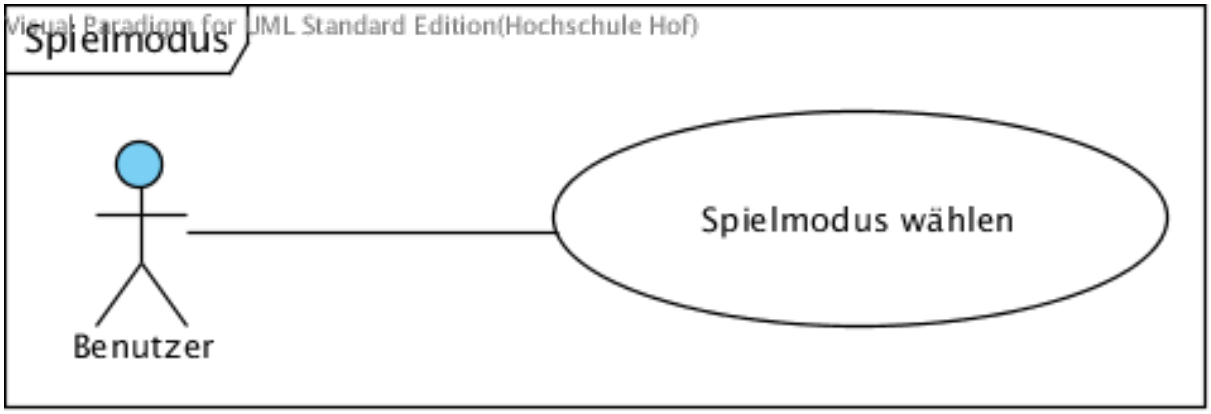
4.1.1 Beschreibung

Spieler erstellen: Der Benutzer erstellt durch die Angabe des Namens einen Spieler. Bei jeder Erstellung eines Spielers wird automatisch geprüft, ob bereits ein Spieler mit gleichem Namen existiert. Dazu wird ein Abgleich mit dem Playerpool durchgeführt.

Spieler laden: Ist der Benutzer bereits als Spieler gespeichert, kann er durch die Auswahl des Spielernamens seine Spielerdaten laden.

Spieler ändern: Gespeicherte Spieler können geändert oder gelöscht werden.
Bemerkung: Diese Lastenheftfunktionen wird in den weiteren Diagrammen und Ausführungen nicht weiter betrachtet, weil es letztendlich eine Kombination aus Laden und Erstellen ist und aufgrund des Umfangs nicht extra betrachtet wird.

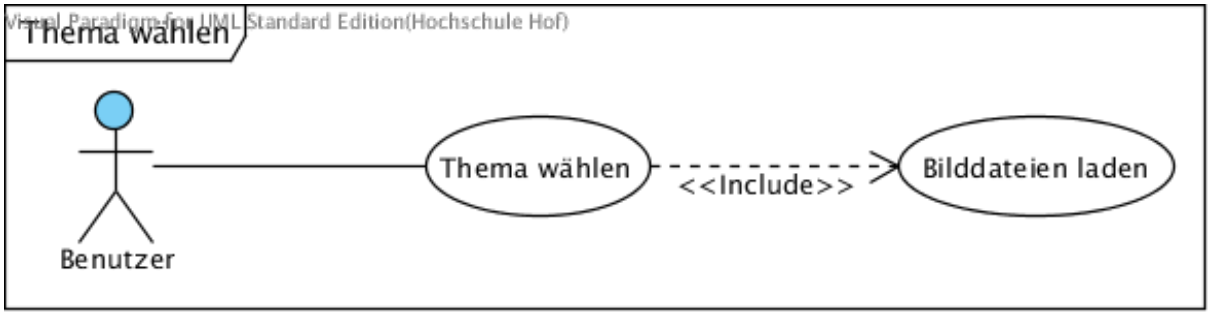
4.2 Anzahl der Spieler wählen - \LF20\



4.2.1 Beschreibung

Spielmodus wählen: Der Benutzer kann zwischen dem Modus „Spieler gegen Computer“ und „Spieler gegen Spieler“ wählen.

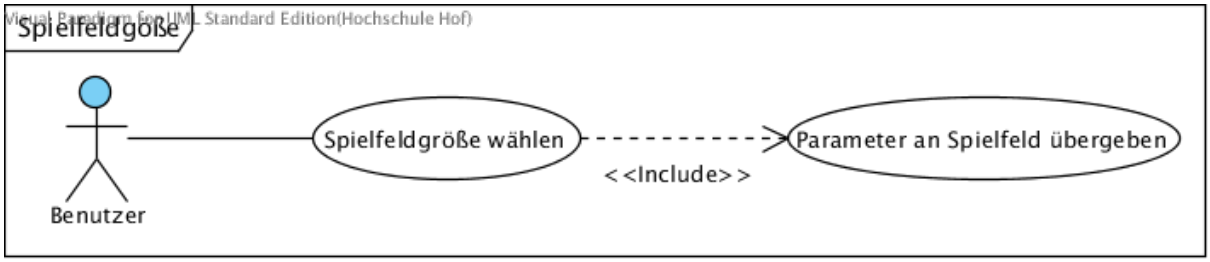
4.3 Thema wählen - \LF30\



4.3.1 Beschreibung

Thema wählen: Dem Benutzer stehen die Themen „Tiere“, „Natur“ und „Flaggen“ zur Auswahl. Je nach gewähltem Thema werden die jeweiligen Bilddaten geladen.

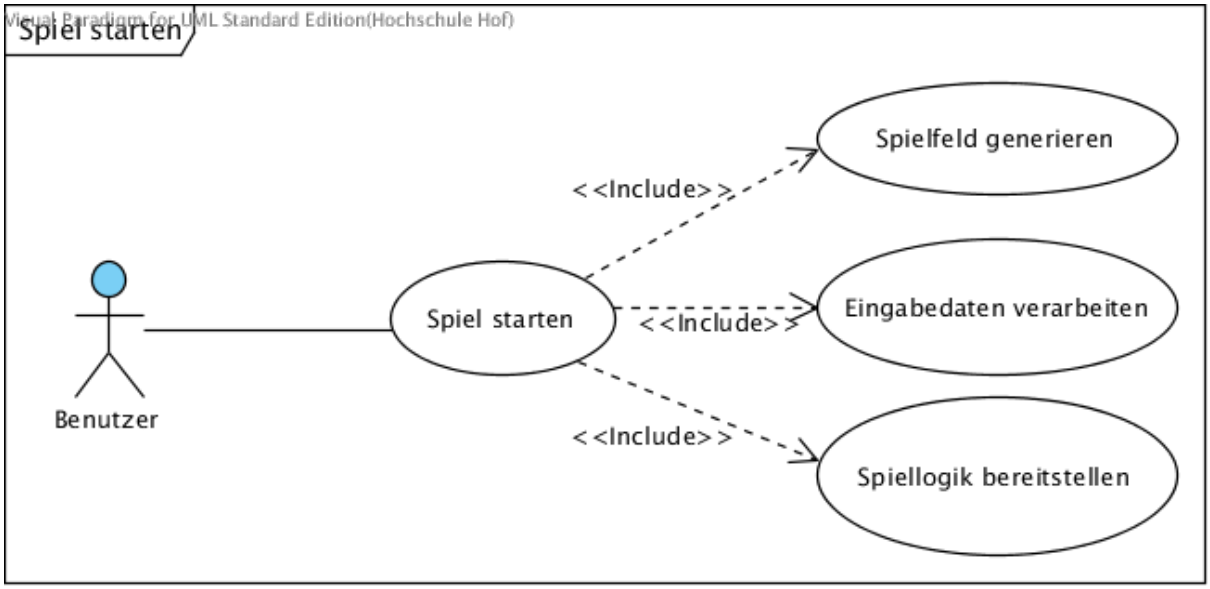
4.4 Spielfeldgröße wählen - \LF40\



4.4.1 Beschreibung

Spielfeldgröße wählen: Bei der Spielfeldgröße können die Größen 4x4 oder 8x8 ausgewählt werden. Anhand der Wahl wird die größe des Spielfeldes bestimmt.

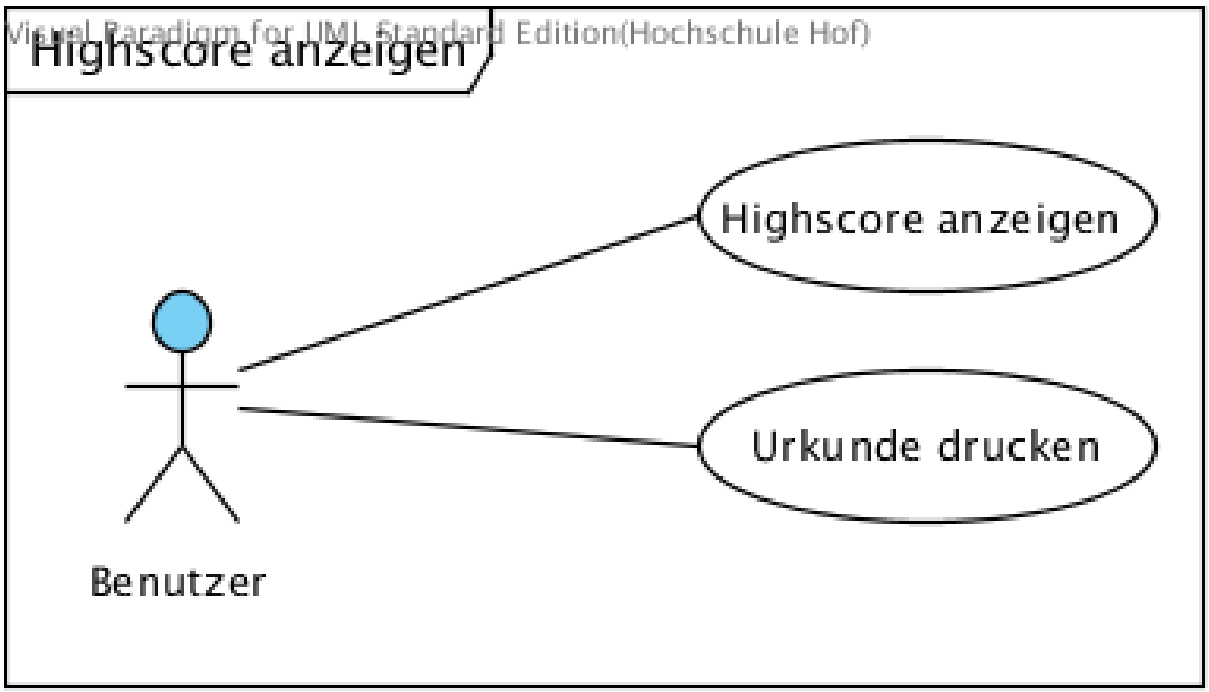
4.5 Spiel starten - \LF50\



4.5.1 Beschreibung

Spiel starten: Durch den Anwendungsfall „Spiel starten“ wird das Spiel gestartet. Es werden folgende Prozesse ausgelöst: Das gewünschte Spielfeld wird generiert, die Eingabedaten Name, Spielmodus, Thema werden verarbeitet und die Spiellogik wird bereitgestellt.

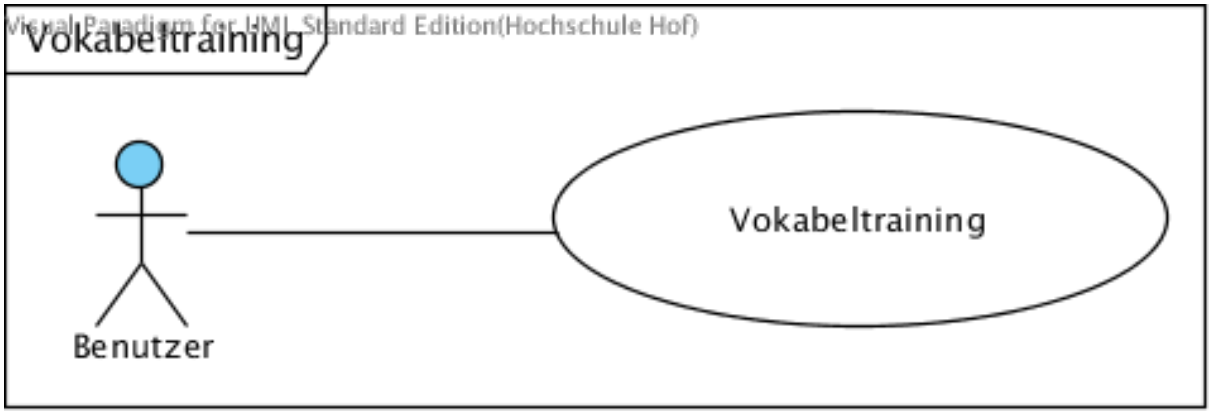
4.6 Highscore - \LF60\,\LF61\



4.6.1 Beschreibung

Highscore anzeigen: Der Benutzer hat die Möglichkeit, sich eine Highscore anzeigen zu lassen. Es werden die besten 10 Spieler sortiert nach erreichter Punktzahl angezeigt.
Urkunde drucken: Die Spieler aus der Highscore haben die Möglichkeit sich eine Urkunde zu drucken.

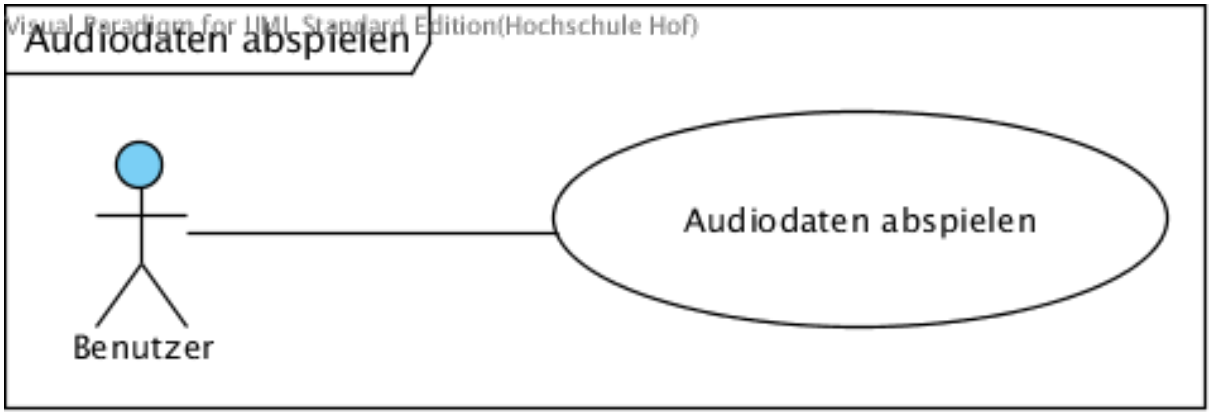
4.7 Vokabeltraining - \LF70\



4.7.1 Beschreibung

Vokabeltraining: Während des Spiels wird bei übereinstimmenden Karten das englische Equivalent zum jeweiligen Kartenmotiv abgefragt.

4.8 Audiodaten abspielen - \LF80\



4.8.1 Beschreibung

Audiodaten abspielen: Zum gewonnenen Kartenpaar kann ein Sound abgespielt werden. Beim Thema Tiere werden die Tierlaute wiedergegeben, bei den Flaggen die jeweilige Hymne des Landes.

5 Use Case Beschreibungen

5.1 Spieler erstellen

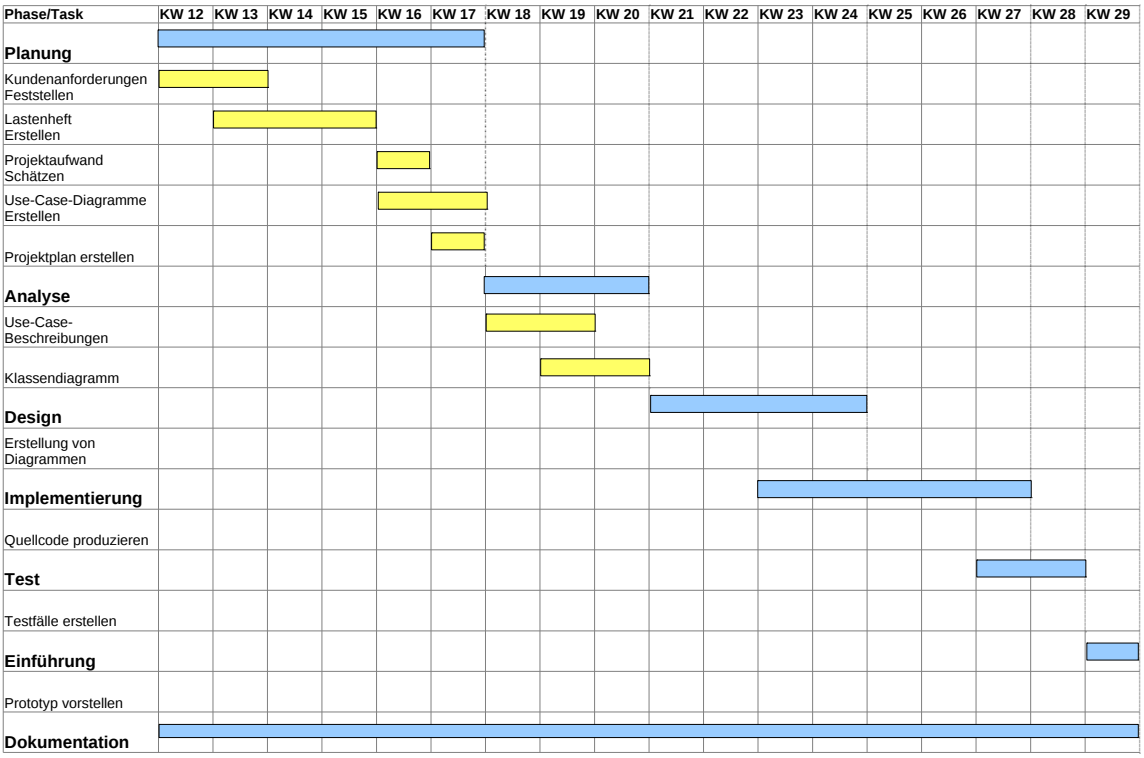
Use Case:	Spieler erstellen	
Actors:	Benutzer	
Purpose:	Benutzer erstellt durch Eingabe seines Namens einen Spieler	
Entry Cond:		
Overview:	Für den Benutzer ist noch kein Spieler erstellt. Der Benutzer trägt seinen Namen ein und e	
Exit Cond:		
Includes:		
Special Req:		
Category:		
Cross Ref:	auf /LF10/ aus Lastenheft	
Ablauf:	Actor Action:	System Response:
	1. Spieler erstellen wählen	
	2. Spielernamen eintragen	

5.2 Spieler laden

5.3 Spielmodus wählen

5.4 Thema wählen

6 Projektplan



7 Klassendiagramm (vorläufig)

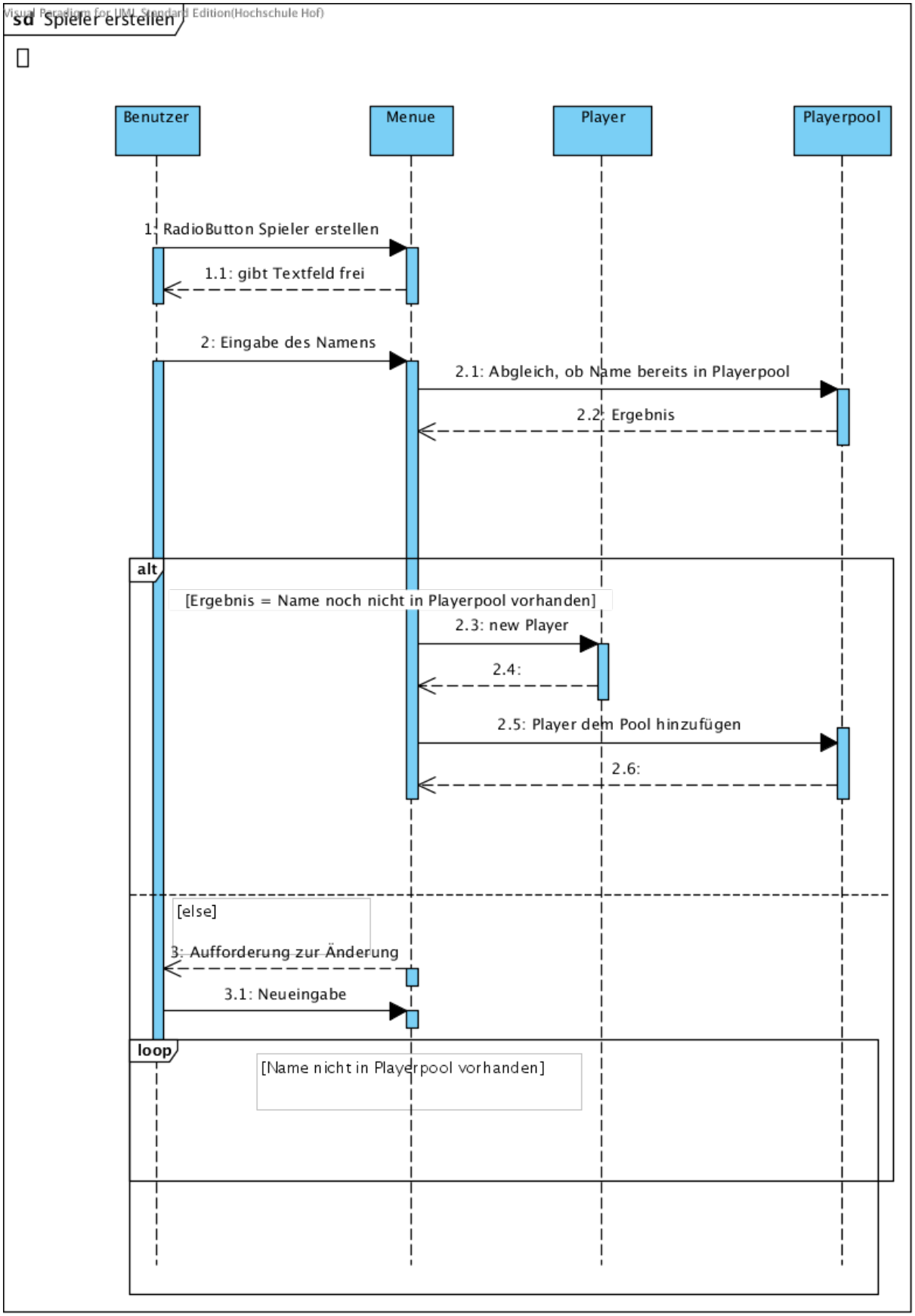
7.1

7.2

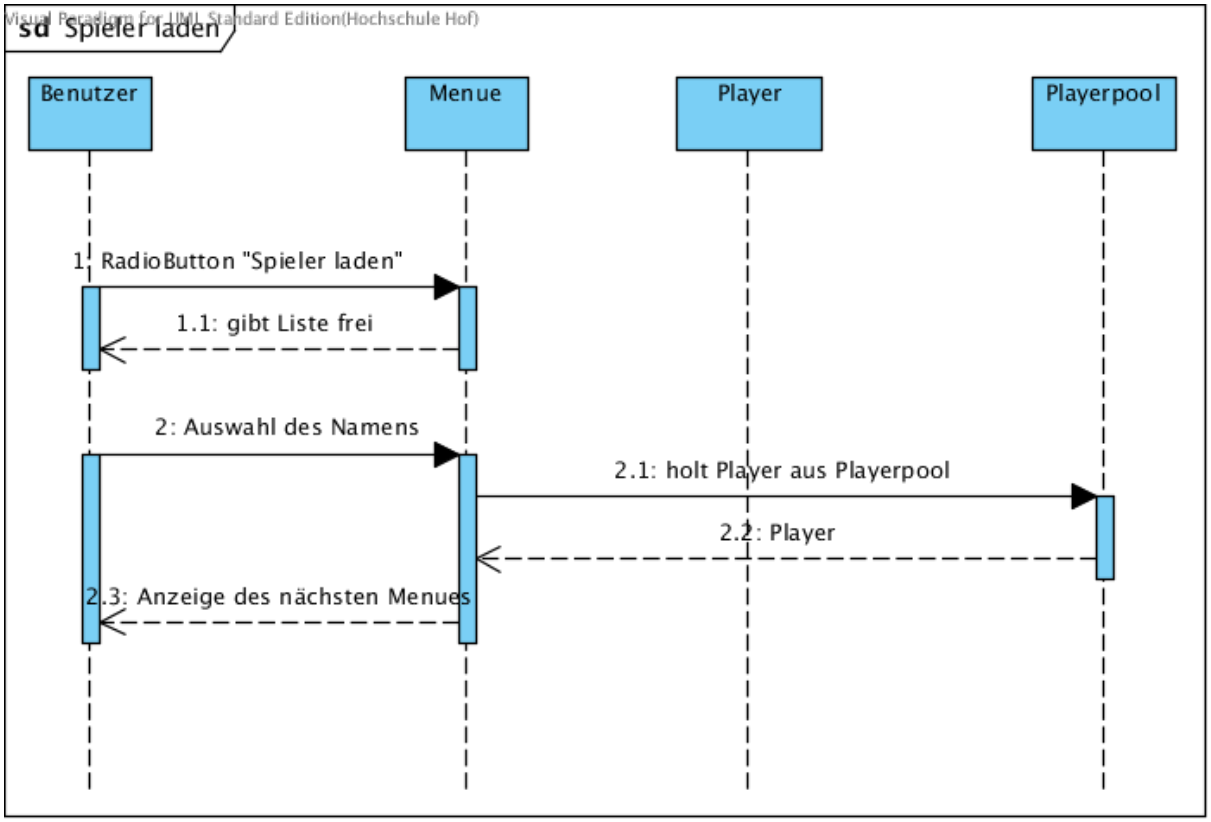
8 Sequenzdiagramme

Sequenzdiagramme bla bla bla .....

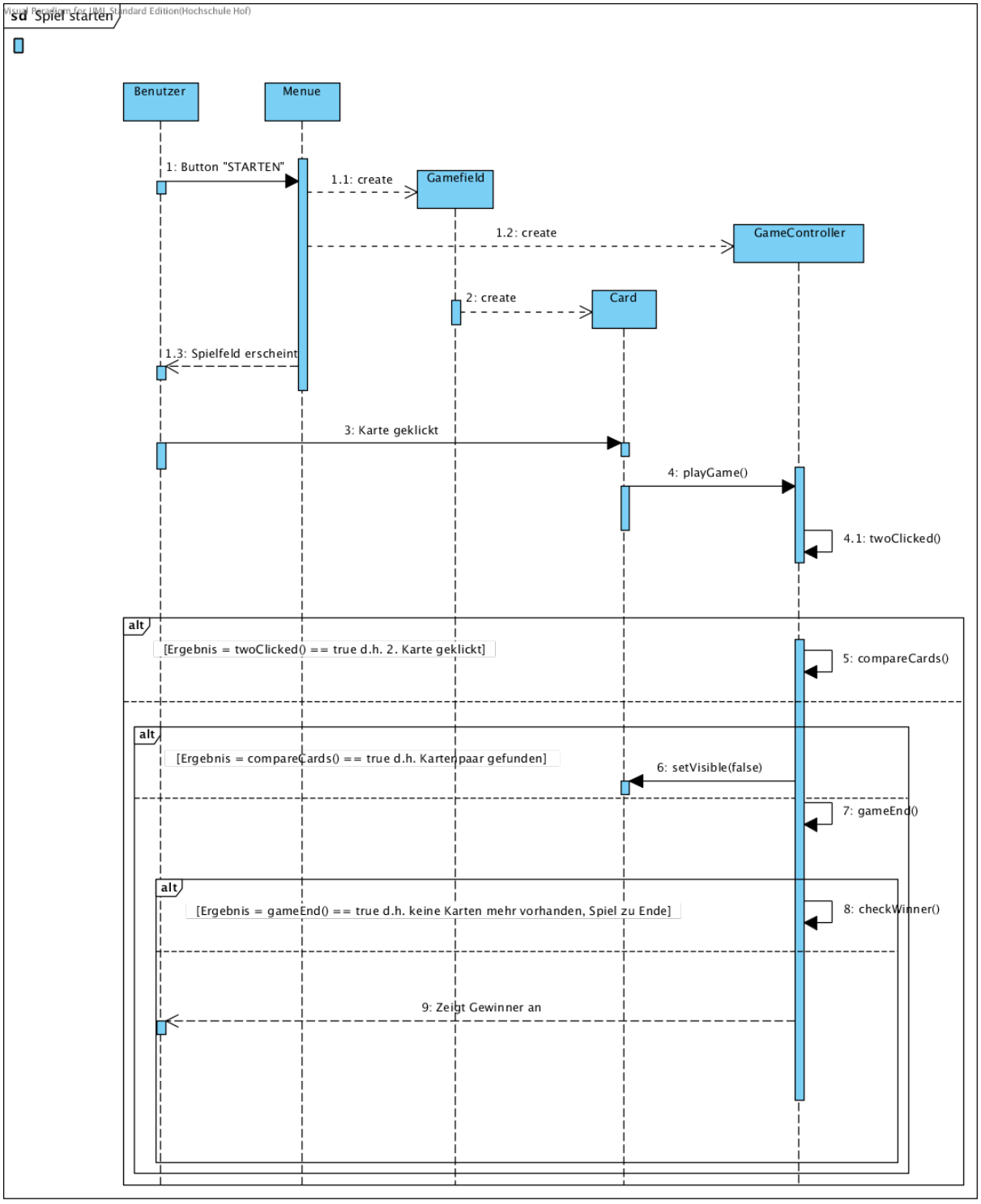
8.1 SD Spieler erstellen


Visual Paradigm for UML Standard Edition(Hochschule Hof)
sd Spieler erstellen
Benutzer
Menue
Player
Playerpool
1: RadioButton Spieler erstellen
1.1: gibt Textfeld frei
2: Eingabe des Namens
2.1: Abgleich, ob Name bereits in Playerpool
2.2: Ergebnis
alt
[Ergebnis = Name noch nicht in Playerpool vorhanden]
2.3: new Player
2.4:
2.5: Player dem Pool hinzufügen
2.6:
[else]
3: Aufforderung zur Änderung
3.1: Neueingabe
loop
[Name nicht in Playerpool vorhanden]

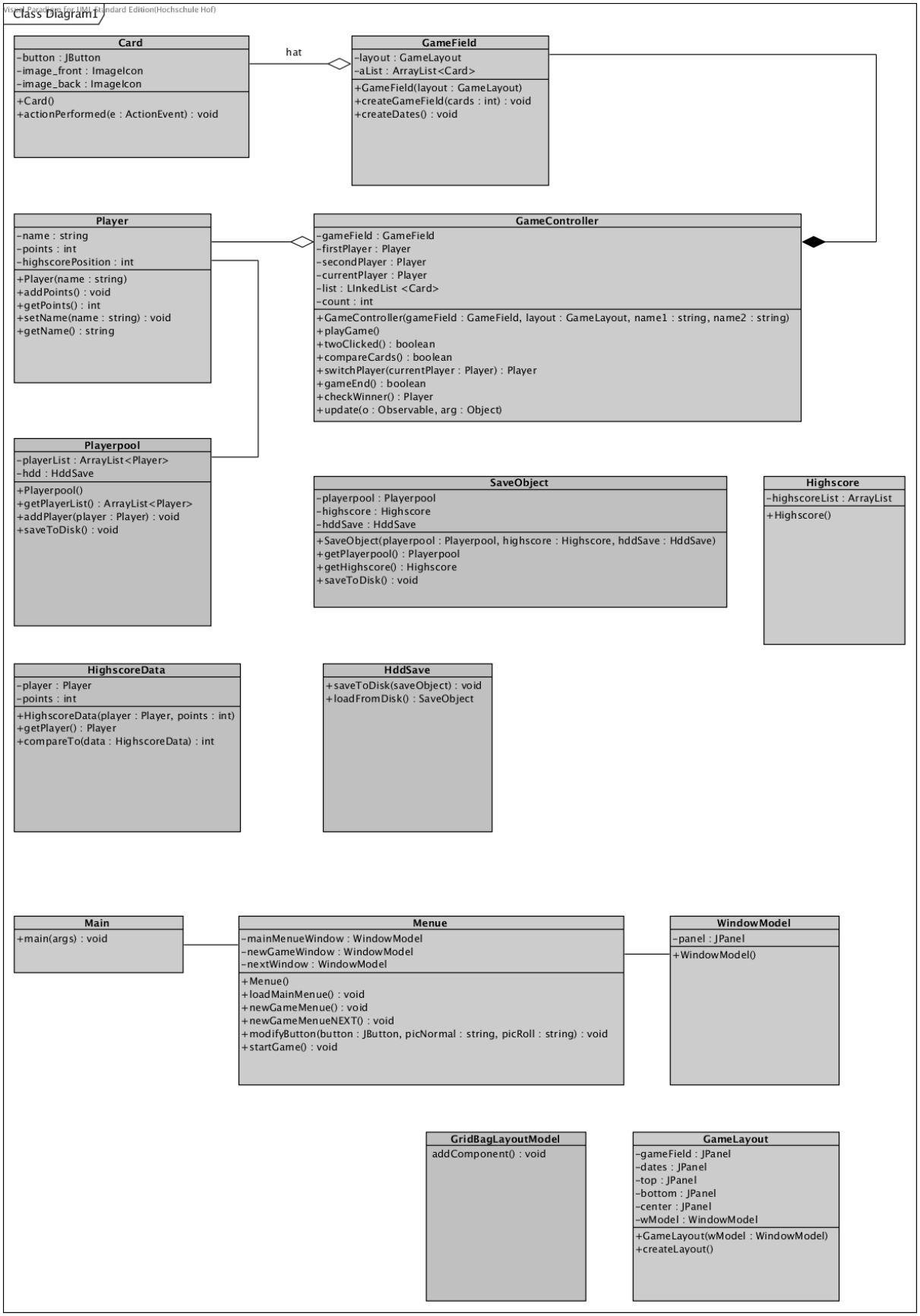
8.2 SD Spieler laden



8.3 SD Spiel starten - LFX



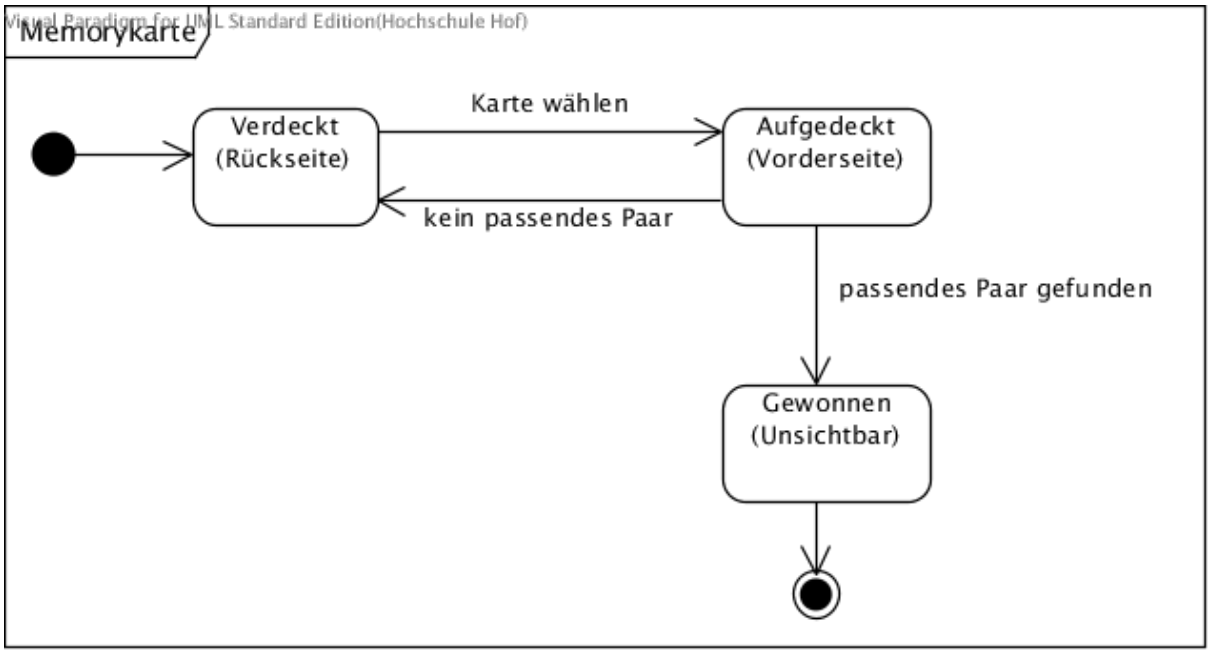
9 Klassendiagramm (final)


Class Diagram1
Card
-button : JButton
-image_front : ImageIcon
-image_back : ImageIcon
+Card()
+actionPerformed(e : ActionEvent) : void
hat
GameField
-layout : GameLayout
-aList : ArrayList<Card>
+GameField(layout : GameLayout)
+createGameField(cards : int) : void
+createDates() : void
Player
-name : string
-points : int
-highscorePosition : int
+Player(name : string)
+addPoints() : void
+getPoints() : int
+setName(name : string) : void
+getName() : string
GameController
-gameField : GameField
-firstPlayer : Player
-secondPlayer : Player
-currentPlayer : Player
-list : LInkedList <Card>
-count : int
+GameController(gameField : GameField, layout : GameLayout, name1 : string, name2 : string)
+playGame()
+twoClicked() : boolean
+compareCards() : boolean
+switchPlayer(currentPlayer : Player) : Player
+gameEnd() : boolean
+checkWinner() : Player
+update(o : Observable, arg : Object)
Playerpool
-playerList : ArrayList<Player>
-hdd : HddSave
+Playerpool()
+getPlayerList() : ArrayList<Player>
+addPlayer(player : Player) : void
+saveToDisk() : void
SaveObject
-playerpool : Playerpool
-highscore : Highscore
-hddSave : HddSave
+SaveObject(playerpool : Playerpool, highscore : Highscore, hddSave : HddSave)
+getPlayerpool() : Playerpool
+getHighscore() : Highscore
+saveToDisk() : void
Highscore
-highscoreList : ArrayList
+Highscore()
HighscoreData
-player : Player
-points : int
+HighscoreData(player : Player, points : int)
+getPlayer() : Player
+compareTo(data : HighscoreData) : int
HddSave
+saveToDisk(saveObject) : void
+loadFromDisk() : SaveObject
Main
+main(args) : void
Menue
-mainMenueWindow : WindowModel
-newGameWindow : WindowModel
-nextWindow : WindowModel
+Menue()
+loadMainMenue() : void
+newGameMenue() : void
+newGameMenueNEXT() : void
+modifyButton(button : JButton, picNormal : string, picRoll : string) : void
+startGame() : void
WindowModel
-panel : JPanel
+WindowModel()
GridBagLayoutModel
addComponent() : void
GameLayout
-gameField : JPanel
-dates : JPanel
-top : JPanel
-bottom : JPanel
-center : JPanel
-wModel : WindowModel
+GameLayout(wModel : WindowModel)
+createLayout()

10 Weitere Diagramme

Sequenzdiagramme bla bla bla .....

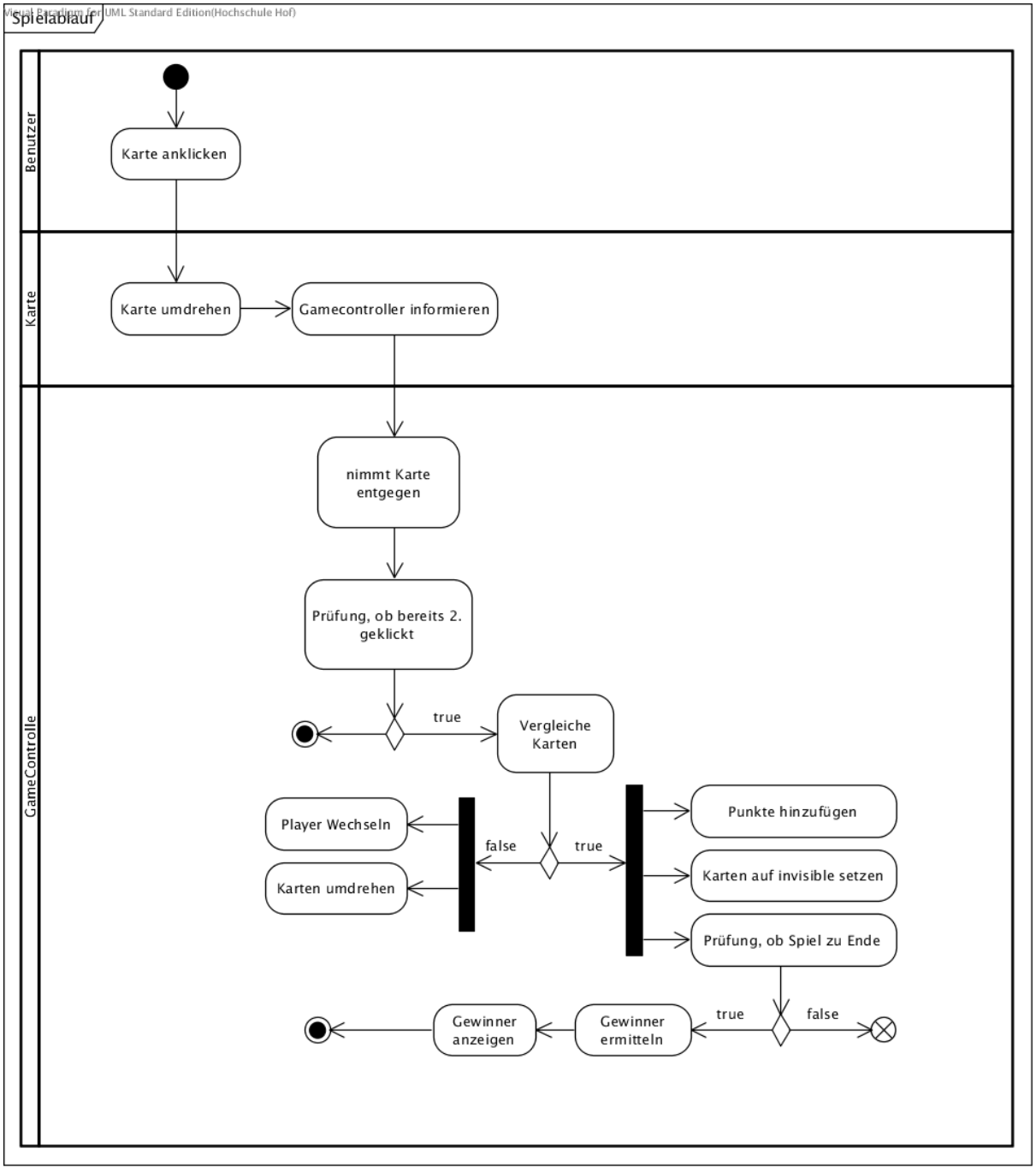
10.1 Zustandsdiagramm - Memorykarte



10.1.1 Beschreibung

Die Karte kann die drei Zustände Verdeckt, Aufgedeckt und Gewonnen annehmen. Wird eine verdeckte Karte ausgewählt, wechselt sie in den Zustand Aufgedeckt. Der Zustand Aufgedeckt kann entweder durch die Feststellung, dass ein passendes Kartenpaar gefunden wurde in den Zustand Gewonnen übergehen, oder falls es sich um nicht zwei übereinstimmende Karten handeln, wieder in den Zustand Verdeckt wechseln.

10.2 Aktivitätsdiagramm - Spielablauf



11 Implementierung

11.1 Allgemeines

Die Implementierung wurde für alle Lastenheftfunktionen der Priorität 1 umgesetzt. Alle Elemente der grafischen Benutzeroberfläche sind ohne GUI-Builder erstellt worden. Der Programmcode liegt als .java Dateien vor und kann über den Aufruf von Main.java ausgeführt werden.

11.2 Umfang

Es wurden 16 Klassen mit insgesamt X Lines of Code implementiert.

```
http://cloc.sourceforge.net v 1.53  T=0.5 s (32.0 files/s, 3324.0 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
Java                            16            342             39           1281
-------------------------------------------------------------------------------
SUM:                            16            342             39           1281
-------------------------------------------------------------------------------
```

11.3

12 Test