Dokumentation Poker Project M. Berger

Inhalt

[Git Link : 2](#_Toc5343706)

[Features Übersicht 2](#_Toc5343707)

[Anforderungen gemäss Folien: 2](#_Toc5343708)

[Klassendiagramm 3](#_Toc5343709)

[Logik 4](#_Toc5343710)

[Handevaluation 4](#_Toc5343711)

[One Pair 4](#_Toc5343712)

[Two Pair 4](#_Toc5343713)

[Three of a kind 5](#_Toc5343714)

[Straight 5](#_Toc5343715)

[Flush 6](#_Toc5343716)

[Fullhouse 6](#_Toc5343717)

[Four of a Kind 6](#_Toc5343718)

[Straightflush 6](#_Toc5343719)

[RoyalFlush 7](#_Toc5343720)

[Score 7](#_Toc5343721)

[Beispiel: 8](#_Toc5343722)

# Git Link :

<https://github.com/MMB360A/Poker>

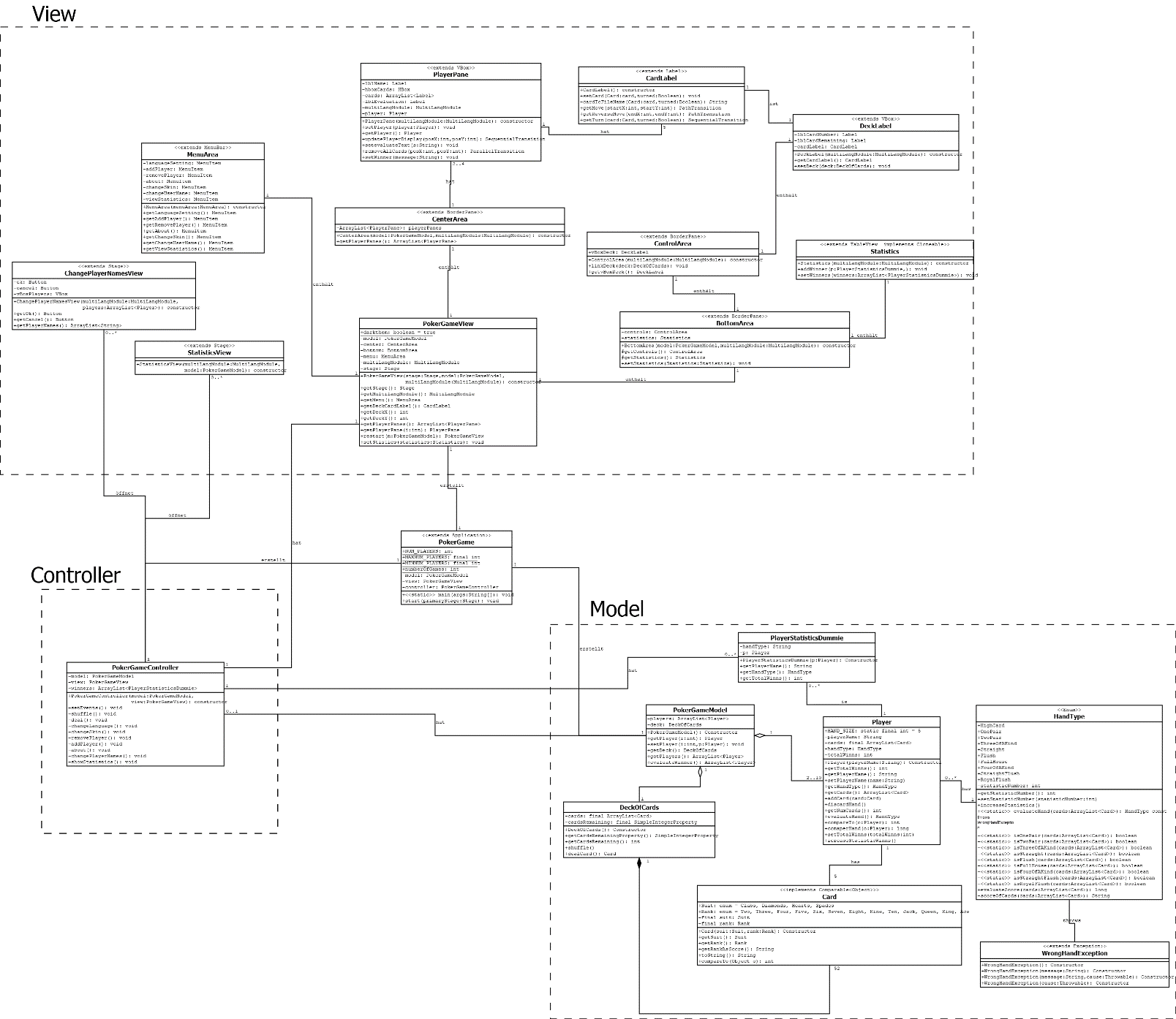
# Features Übersicht

* Voll implementierte Gewinnerermittlung inkl. Tie-Break
* Animationen der Karten
* Einfache Statistik in der Mainview
* Detaillierter Statistik über Gewinner und Hände
  + Die Statistik ist unter dem Menu Datei / Statistik einsehbar
* Mehrsprachig
  + Die Sprache kann unter Einstellungen / Sprache geändert werden
  + Um eine neue Sprache hinzu zu fügen kann einfach das translations.csv file angepasst werden
* Anpassbare Usernamen
  + Funktioniert auch wenn mehrere User denselben Namen haben
* Variierbare Anzahl Spieler
  + Minimale und maximale Anzahl werden im Programm vorgeschrieben es ist sinnlos weniger als 2 Spieler zu haben und unmöglich mehr als 10 zu haben da sonnst zu wenig Karten verfügbar sind.
* Day- und Nightmode
  + Unter Ansicht/Daymode kann der look der App in einen Tagesmodus gewechselt werden

# Anforderungen gemäss Folien:

|  |  |  |  |
| --- | --- | --- | --- |
| Nr | Anforderungen | Typ | Aus meiner Sicht erfüllt |
| 01 | GUI fertig und verschönert | Must have | Ja |
| 02 | Alle Handtype erkennen | Must have | Ja |
| 03 | Gewinner identifizieren | Must have | Ja |
| 04 | Menu einfügen | Must have | Ja |
| 05 | Spieleranzahl verändern | Must have | Ja |
| 06 | GUI dynamisch an Anzahl Spieler anpassen | Must have | Ja |
| 07 | Tie-Break | Nice to have | Ja |
| 08 | Noch schöneres GUI | Nice to have | Ja |
| 09 | Animationen | Nice to have | Ja |
| 10 | USW. | Nice to have | ja |

# Klassendiagramm



# Logik

## Handevaluation

Die Logik überprüft zuerst ob die Liste den Anforderungen entspricht:

* Nicht null
* Enthält exakt 5 Karten
* Keine Karte Null

Danach wird die Hand geklont und Sortiert und anschliessend analysiert, da die Hand sortiert ist gibt es für jeden Hand type nur eine sehr begrenzte Anzahl möglicher Kombinationen die überprüft werden müssen:

### One Pair

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Pos 0 | Pos 1 | Pos 2 | Pos 3 | Pos 4 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| Die Karten eines Paares in einer sortierten Liste sind immer nebeneinander | | | | |
| for(int i = 0; i< cards.size -1; i++) if(cards.get(i).getRank() == cards.get(i+1).getRank()) return true; | | | | |

### Two Pair

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Pos 0 | Pos 1 | Pos 2 | Pos 3 | Pos 4 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| In jedem möglichen Fall sind Pos1 und Pos3 Teil eines der beiden Paare aber nicht desselben | | | | |
| If((pos1 == pos0 || pos 1 == pos 2) && (pos3 == pos 2 || pos3 == pos4)) return true;  //Also true if pos1 == pos2== pos3 but this is 3 of a Kind so no problem | | | | |

### Three of a kind

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Pos 0 | Pos 1 | Pos 2 | Pos 3 | Pos 4 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| In jedem Fall ist die pos2 Teil des Drillings | | | | |
| for(int i = 2; i< cards.size; i++)  if(cards.get(i).getRank() == cards.get(i-1).getRank() &&  cards.get(i).getRank() == cards.get(i-2).getRank()) return true; | | | | |

### Straight

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Pos 0 | Pos 1 | Pos 2 | Pos 3 | Pos 4 |
|  |  |  |  |  |
|  |  |  |  |  |
| Sortiertes Blatt = alle Karten der Reihe nach aufsteigend ausnahme 2 - Ass | | | | |
| Boolean isStreet = true;  For(int i == 0; i < cards.getSize() && isStreet; i++)  if (cards.get(i-1).getRank().ordinal() + 1 != (cards.get(i).getRank().ordinal())){  if(cards.get(i-1).getRank().ordinal() != 12 || cards.get(0).getRank().ordinal() != 2)  isStreet = false;  } | | | | |

### Flush

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Pos 0 | Pos 1 | Pos 2 | Pos 3 | Pos 4 |
|  |  |  |  |  |
| Der Rank der Carde ist egal die Suite nicht, einfacher Loop | | | | |
| for(int i = 0; i< cards.size; i++) if(cards.get(i).getSuite() != cards.get(0).getSuite()) {return false;} return true; | | | | |

### Fullhouse

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Pos 0 | Pos 1 | Pos 2 | Pos 3 | Pos 4 |
|  |  |  |  |  |
|  |  |  |  |  |
| Die Pos 2 gehört immer zum drilling des Fullhouses und es ist unmöglich, dass pos0 = pos2 wenn pos1 != pos2 | | | | |
| If((pos2 == pos0 && pos3 == pos4)|| (pos2 == pos4 && pos0== pos1)) return true ; | | | | |

### Four of a Kind

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Pos 0 | Pos 1 | Pos 2 | Pos 3 | Pos 4 |
|  |  |  |  |  |
|  |  |  |  |  |
| Wenn pos1 = pos 4 muss pos2 und pos3 ebenfalls denselben Rank wie pos1 und 4 haben | | | | |
| If(pos0 == pos3 && pos1 == pos4) return true; | | | | |

### Straightflush

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Pos 0 | Pos 1 | Pos 2 | Pos 3 | Pos 4 |
|  |  |  |  |  |
|  |  |  |  |  |
| Gleichzeitig Strasse und Flush | | | | |
| Return(isStraight && isFlush); | | | | |

### RoyalFlush

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Pos 0 | Pos 1 | Pos 2 | Pos 3 | Pos 4 |
|  |  |  |  |  |
| Gleichzeitig Straightflush und die tiefste Karte muss eine 10 sein | | | | |
| Return(isStraightFlush && pos0 == ten); | | | | |

## Score

Jede Hand bekommt zusätzlich noch einen individuellen Score basierend auf dem Handtype und allen Karten. Der Score erlaubt es 2 Hände exakt miteinander zu vergleichen und den Gewinner zu ermitteln.

Die erste Stelle ist für den Handtype:

|  |  |
| --- | --- |
| **Blatt** | **Score** |
| Hightcard | 0 |
| Pair | 1 |
| Two Pairs | 2 |
| Three OK | 3 |
| Straight | 4 |
| Flush | 5 |
| Full House | 6 |
| Four OK | 7 |
| Straight Flush | 8 |
| Royal Flush | 9 |

Danach wird jede Karte mit einer 2-stelligen Nummer der Relevanz und der Grösse nach aufgelistet:

Es wird ein seperater Score erstellt und nicht der Ordinal verwendet aus 2 Gründen.

1. Im Spezialfall Strasse mit Ass – 5 soll das Ass die niedrigste Karte (Score 01) und nicht die höchste sein
2. Für mich als Programmierer sind diese Scores übersichtlicher da sie denn Nummern auf den Karten entsprechen

|  |  |
| --- | --- |
| **Karte Score** | **Score** |
| 2 | \*02 |
| 3 | \*03 |
| 4 | \*04 |
| 5 | \*05 |
| 6 | \*06 |
| 7 | \*07 |
| 8 | \*08 |
| 9 | \*09 |
| 10 | 10 |
| Jack | 11 |
| Queen | 12 |
| King | 13 |
| Ace | 14 |
| Ace on Beginning of a Straight | \*01 |

### Beispiel:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Pos 0 | Pos 1 | Pos 2 | Pos 3 | Pos 4 | Score |
|  |  |  |  |  | 51108050302 |
| 02 | 03 | 05 | 08 | 10 | *Gewinner* |
|  |  |  |  |  | 10303050402 |
| 02 | 03 | 03 | 04 | 05 | *Dritter* |
|  |  |  |  |  | 30202020403 |
| 02 | 02 | 02 | 03 | 04 | *Zweiter* |