**Requirements Document**

Case study: Ganderpoke Card Game

25-03-2018

**Author: Ahmed Aziz**

**25-03-2018**

## Requirements Document contents:

List of figures………………….………………………………………………………………..…………………………………2

1. Chapter 1: Requirements for Ganderpoke Card Game….………………………………………….…..3
   1. Domain description (i.e. how game is played…………………………………………………………3
   2. Functional requirements…………………………………….………………………………………………...4
   3. Quality requirements…………………………………….………………………………………………….…..4
   4. Use cases………….…….…………………………………………………………………………………………….5
   5. Workflow model.……..……………………………………………………………………………………………8
2. Chapter 2: Initial Conceptual Model…………………………………………………………………………….12
   1. Class diagram……………………………………………………………………………………………………….12
   2. Class description………………………………………………………………………………………………….13
3. Chapter 3: Dynamic Designs………………………………………………………………………………………..14
   1. Sequence diagram……………………………………………………………………………………………….14
   2. Object Specification….……………………………………………………………...............................18
   3. Object Diagram……...……………………………………………………………………………………………19
4. Chapter 4: User Interface Interaction…………………………………………………..........................22
   1. State diagrams………………………………………………………………………………………..…………..22
5. Chapter 5: Testing……………………………………………………………………………………………………….24
   1. Design and implementation verification…………………………………………….………………..24

List of figures

Figure 1.1 Workflow model of Ganderpoke game……………………………………………………….………..8

Figure 1.2 Setting up the Ganderpoke game……………………………………………………………….………..9

Figure 1.3 Player turn workflow………………………………………………………………………………………….10

Figure 1.4 Calculate score workflow…………………….……………………………………………………………..11

Figure 1.5 Class Diagram……………….…………………………………………………………………………………….12

Figure 1.6 Taking a turn.……………….…………………………………………………………………………………….14

Figure 1.7 Discard Joker card.….………………………………………………………………………………………….15

Figure 1.8 Get Joker card…………………………………………………………………………………………………….16

Figure 1.9 Check for winner.…….…………………………………………………………………………………………17

Figure 1.10 Valid Object Diagram for game with two players…………….…………………………….….19

Figure 1.11 Invalid Object Diagram for game with three players……………..………………………….19

Figure 1.12 Valid Object Diagram for player hand with 8 cards………………….……………………….20

Figure 1.13 Invalid Object Diagram for player hand with 7 cards………………….…………………….20

Figure 1.14 Detailed Class Diagram with methods……………………………………………..……………….21

Figure 1.15 State Diagram showing User Interface Interaction…………..……………………………….22

Figure 1.16 Substate of Game running………………………………………………………………………….…….23

Figure 1.17 Substate of Player Turn…………………………………………………………………………………….23

Chapter 1: Requirements for Ganderpoke Card Game

* 1. Domain description (i.e. how the game is played)

This game starts with a 52 card-pack, add one joker to the pack to make it 53 cards, and then remove all twos, threes and fours from the pack from all types of cards hearts, spades etc. This will then leave the game with total of 41 cards by removing 12 cards.

To deal the cards you must first deal 25 cards from the pack face down on the table in a square of five rows and columns. The divide the remaining 16 cards between the two players equally (8 for each player) face down.

The ultimate aim of the game for a player is to produce the highest or the lowest five-card poker combination from their hand of cards before all 25 cards on the table are turned faced upwards. Depending on the player’s hand they will score according to the total point-value of the 10-poker combination formed by the rows and columns of the square when all 25 cards are faced upwards.

The non-dealer of the cards will begin the turns, and then play continues in turns. With each player turn they may take any faced-down card from the square, then add it to their hand but must put a card from their hand back onto the square faced upwards. The card picked from the square can be put faced upwards if the card is not wanted by the player.

The square can only have one joker card which acts as a wildcard. If a player holds a joker on their turn of play they may take any of the faced upward cards from the square and replace it with the joker face up. If the joker is faced upwards in the square the player on their turn may pick the card and replace it with a card from their hand faced down. If a player holds the joker after all 25 cards are faced upwards they automatically lose the game.

The play ends as soon as all 25 cards from the square are faced upwards. The value of the square is then calculated as follows. Each row and column is considered a five-card poker combination and scored as:

* 1 pair = 1 point
* 2 pair = 2 point
* 3 pair = 3 point
* Straight, flush or full house = 5 point
* Four of a kind = 8 point
* Straight flush = 10 point

A joker in the square stands for whatever card is needed to make the highest possible combination in a row and for whatever other card is needed to make highest in a column. The score of the 5x5 square are summed to provide a total score (value) for the square.

After the score of the square is calculated then the players produce their best five-card poker combinations. The highest combination the player produces scores double the square value, and the lowest five-card poker combination can score 1 times the square value (which can be the same player would first produce their highest and lowest hand). This constitutes one round of play; players can decide how many rounds of play there should be when the game commences.

* 1. Functional requirements

From the perspective of the game server:

1. Initiate a game of Ganderpoke with two players
2. Allow client to connect to game server
3. Register a player with a game
4. Record a name for the player
5. Commence a game
   1. Shuffle the game pack
   2. Put 25 cards faced down in a 5x5 square
   3. Deal 8 cards to each player
   4. Last player to join game takes first turn
   5. Take turn
6. Check the move is legal according to game rules
7. Message to players to indicate of turns etc.
8. Calculate value of the 5x5 square
9. Check if player holds joker card in hand after all 25 cards faced
10. Produce players highest 5 card poker combination
11. Produce players lowest 5 card poker combination
12. Announce winner
13. Reset game to play again

From the perspective of a client:

1. Connect to a game
2. Join a game successfully
3. Enter a nickname for client
4. Play the game
   1. Flip a card from the 5x5 square
   2. Swap card with one from hand if its useful to hand
5. View other players in the game
6. Send messages to other players during game
7. Exit the game
   1. Quality requirements
8. The server application will be implemented using Java Standard Edition, Version 8
9. When clients are connecting they should not wait no longer than 20 seconds
10. The game server should handle multiple games concurrently
    1. Use cases

This section focuses on showing the different use cases that the system once built should be able to do in order to meet the client’s needs. Use cases are like samples which means that they are not near the modelling phase of the system. The dynamic modelling would start once we have a domain description and functional requirements of the system in order to get an idea of what the client is looking for in this system.

|  |  |  |
| --- | --- | --- |
| Identifier and name | | UC01: start a game |
| Initiator | | Game administrator |
| Goal | | To successfully start the game |
| Start and stop points | | Two players join the game, record order of connection |
| Main success scenario | | |
| 1 | Players sent message game is about to initiate | |
| 2 | The game pack is shuffled | |
| 3 | Make a square of 25 cards faced down | |
| 4 | Player 1 receives 8 cards | |
| 5 | Player 2 receives 8 cards | |
| 6 | Player that joined last starts the turn | |

|  |  |  |
| --- | --- | --- |
| Identifier and name | | UC02: Initiate the game of Ganderpoke |
| Initiator | | Game administrator |
| Goal | | To successfully connect two players and start the game |
| Start and stop points | | Two players join the game, start game |
| Main success scenario | | |
| 1 | Game server accepts player 1 connection and sends message for other players to wait | |
| 2 | Game server accepts player 2 connection | |
| 3 | A new game is created and two players are linked to a game | |
| 4 | Start a game (Inclusion UC01) | |

|  |  |  |
| --- | --- | --- |
| Identifier and name | | UC03: Take a turn |
| Initiator | | Players |
| Goal | | To take a turn |
| Start and stop points | | Pick card from square if useful add to hand, put back in square face up |
| Main success scenario | | |
| 1 | Player picks a card from square | |
| 2 | Player checks value of card | |
| 3 | Useful to hand | |
| 4 | Player adds to hand | |
| 5 | Not useful to hand | |
| 6 | Player puts card back in square face up | |

|  |  |  |
| --- | --- | --- |
| Identifier and name | | UC04: Check for winner |
| Initiator | | Game administrator |
| Goal | | To monitor current game for a winner |
| Start and stop points | | Cards in square faced up, announce winner |
| Main success scenario | | |
| 1 | All 25 cards are faced up in the square | |
| 2 | Calculate value of square (Inclusion UC08) | |
| 3 | Produce highest 5 card poker combination (Inclusion UC09) | |
| 4 | Produce lowest 5 card poker combination (Inclusion UC010) | |
| 5 | Announce winner | |

|  |  |  |
| --- | --- | --- |
| Identifier and name | | UC05: discard joker card from hand |
| Initiator | | Player |
| Goal | | Discard joker card from hand and pick a new card from square |
| Start and stop points | | Select joker card to discard, replace with card from square face up |
| Main success scenario | | |
| 1 | Player selects joker card from hand | |
| 2 | Analyse the square | |
| 3 | Choose card to pick up from the square | |
| 4 | Pick up card from the square | |
| 5 | Replace with joker card face up | |

|  |  |  |
| --- | --- | --- |
| Identifier and name | | UC06: get joker card from square |
| Initiator | | Player |
| Goal | | Get joker card from square and replace with one from hand faced down |
| Start and stop points | | Player selects card from hand, pick up joker card |
| Main success scenario | | |
| 1 | Player selects card from hand | |
| 2 | Discard card from hand faced down in square | |
| 3 | Pick up joker card and add to hand | |

|  |  |  |
| --- | --- | --- |
| Identifier and name | | UC07: Player holds joker card after all cards are faced up |
| Initiator | | Players |
| Goal | | To check if player holds joker card after all cards have been faced up |
| Start and stop points | | Player checks hands, player loses |
| Main success scenario | | |
| 1 | Players check if they hold joker card in hand after all cards faced up | |
| 2 | Compare hands to see if other player holds joker card | |
| 3 | Player holds joker card in hand | |
| 4 | Player loses | |

|  |  |  |
| --- | --- | --- |
| Identifier and name | | UC08: calculate value of square |
| Initiator | | Game |
| Goal | | To accurately calculate the value of the square |
| Start and stop points | | Check rows and columns for five-card combination, present score |
| Main success scenario | | |
| 1 | Check row for 5 card poker combination | |
| 2 | The row contains joker card | |
| 3 | Use to make highest poker combination in the row | |
| 4 | Check row for 5 card poker combination | |
| 5 | The column contains joker card | |
| 6 | Use to make highest poker combination in the column | |
| 7 | Present total score for square | |

|  |  |  |
| --- | --- | --- |
| Identifier and name | | UC09: produce highest poker combination |
| Initiator | | Players |
| Goal | | To produce the highest 5 card poker combination |
| Start and stop points | | Players checks cards, players decide winner of highest 5 card combo |
| Main success scenario | | |
| 1 | Players check values of cards in hand | |
| 2 | Players decide which cards to use for highest 5 card poker combination | |
| 3 | Players compare poker hands | |
| 4 | Decide winner of highest 5 card poker combination | |

|  |  |  |
| --- | --- | --- |
| Identifier and name | | UC010: produce lowest poker combination |
| Initiator | | Players |
| Goal | | To produce the lowest 5 card poker combination |
| Start and stop points | | Players checks cards, players decide winner of lowest 5 card combo |
| Main success scenario | | |
| 1 | Players check values of cards in hand | |
| 2 | Players decide which cards to use for lowest 5 card poker combination | |
| 3 | Players compare poker hands | |
| 4 | Decide winner of lowest 5 card poker combination | |

* 1. Workflow models

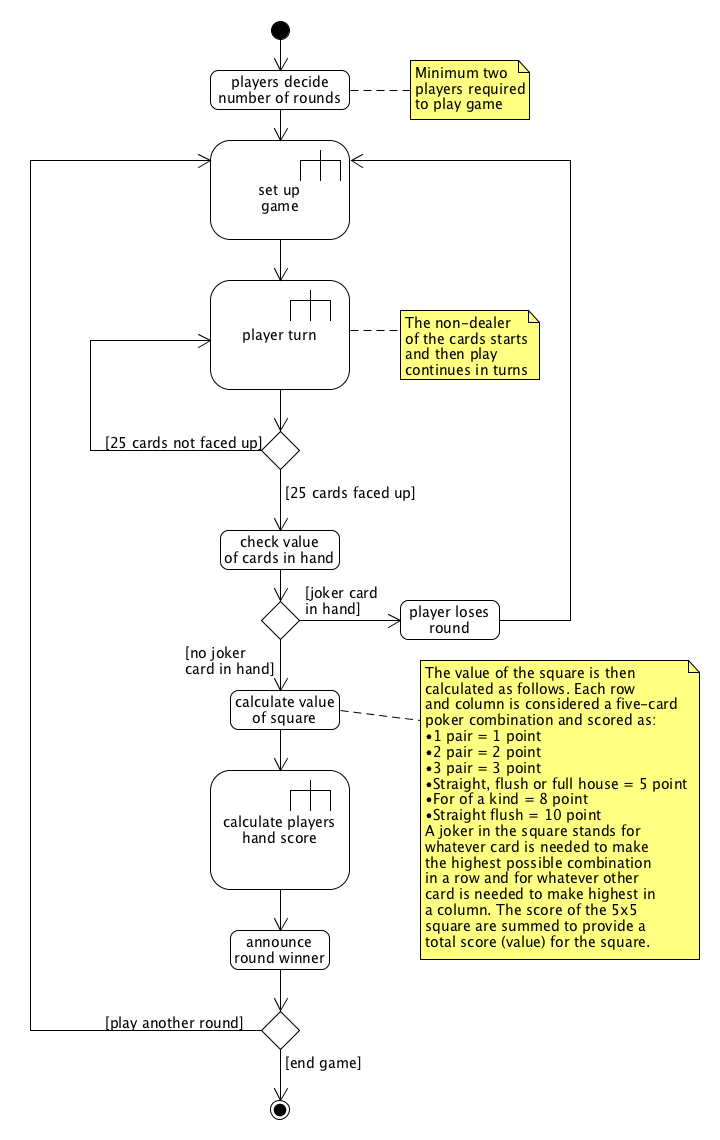
This section focuses on the workflow models of the Ganderpoke card game from analyst’s point of understanding what the client wants to build. The use of UML Activity diagrams clearly helps us understand how the system should be used in order to achieve goals asked by the client. The models presented in this section visualise the game being played in the real world with players playing the game and making decisions.

Figure 1.1 illustrates the high level Workflow of Ganderpoke card game

Figure 1.1 shows us how the game is set up (see figure 1.2) and played in the real world with them making decisions. It also shows players taking turns until the win condition is met by one of the players, where after they have a condition of either ending the game or option to play another round. The next workflow model (see figure 1.3) focuses on ‘player turn’ action in the model above figure 1.1. The action has a call to behaviour which means that there are many behaviours associated with this action, so there is another model to show this (see figure 1.3).

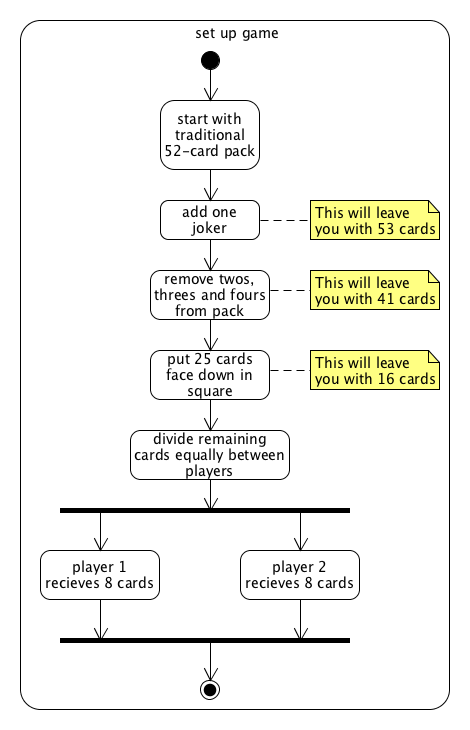


Figure 1.2 illustrates how to set up the game

Figure 1.2 shows how the game is set up with the deck, adding and removing the cards and also dealing the cards to the players.

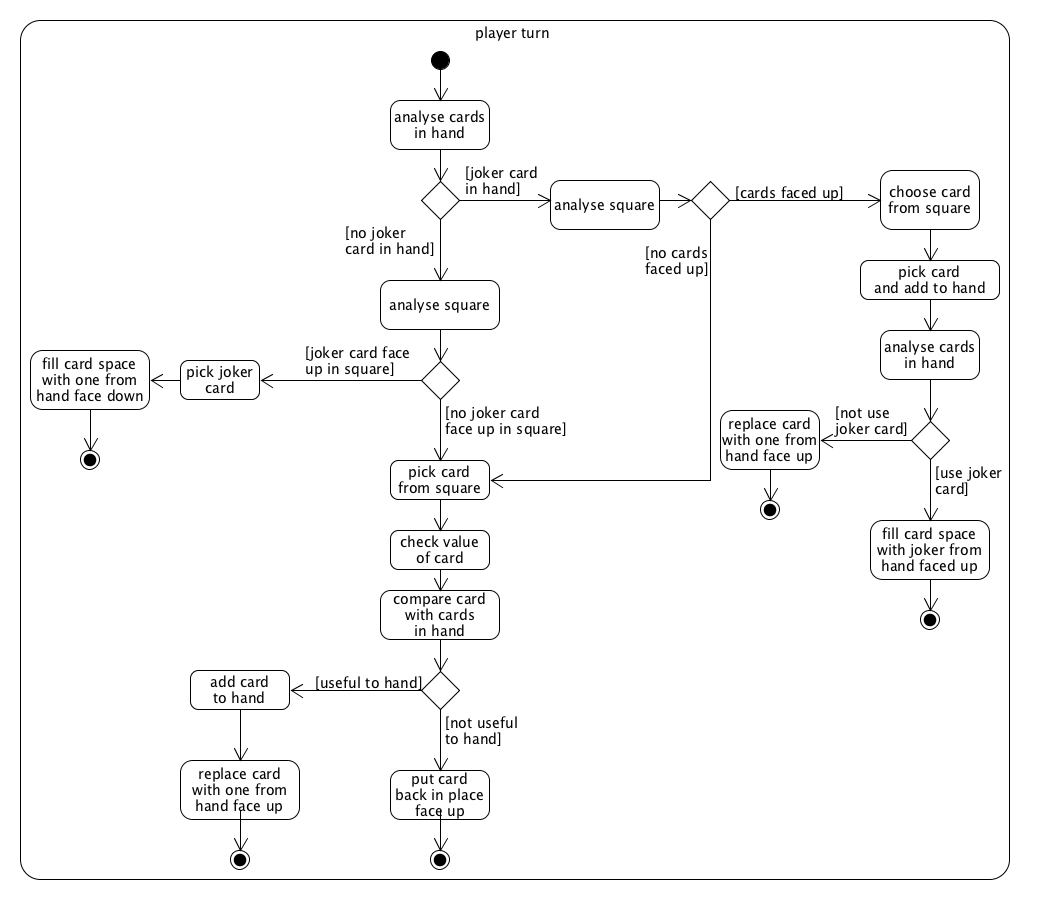


Figure 1.3 illustrates Player turn workflow

Figure 1.3 is a called behaviour ‘player turn’ in the first model (see figure 1.1), which is a detailed action that is explained in more detail in this model above. It also gives us an insight into how the players will be taking turns in the game, where they would have many decisions to make when they are picking cards.

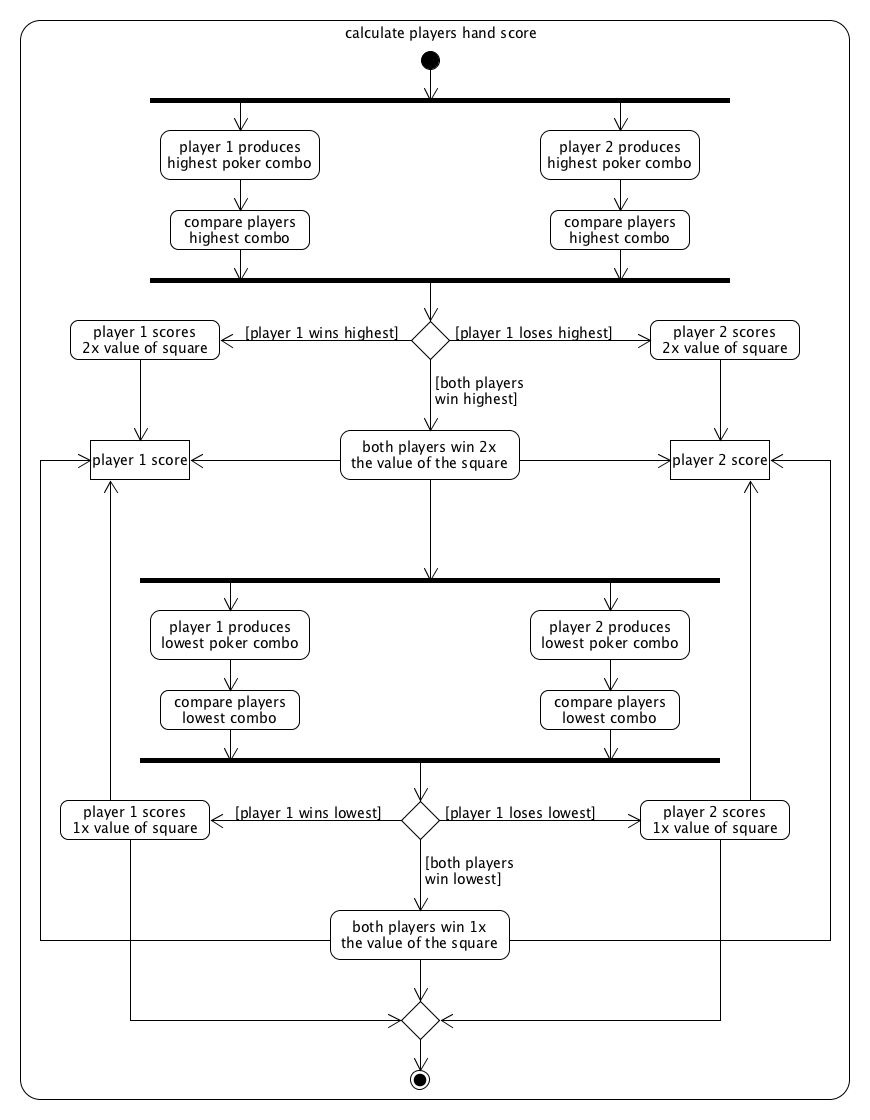


Figure 1.4 illustrates Calculating player’s hand score

Figure 1.4 shows us another called behaviour ‘calculate player’s hand score’ from the model in figure 1.1 where the players would need to calculate the score of the round. This is demonstrated using this model, which clearly states the actions that the players would need to do in order to successfully calculate the scores of the round.

Chapter 2: Initial Conceptual Model

This chapter focuses on the Initial Conceptual Model that shows what the possible classes there will be for this system, as well as the things that it would be in charge off (attributes of classes). I will also give a class description in order to show the behaviours of each class to show their responsibilities within this system.

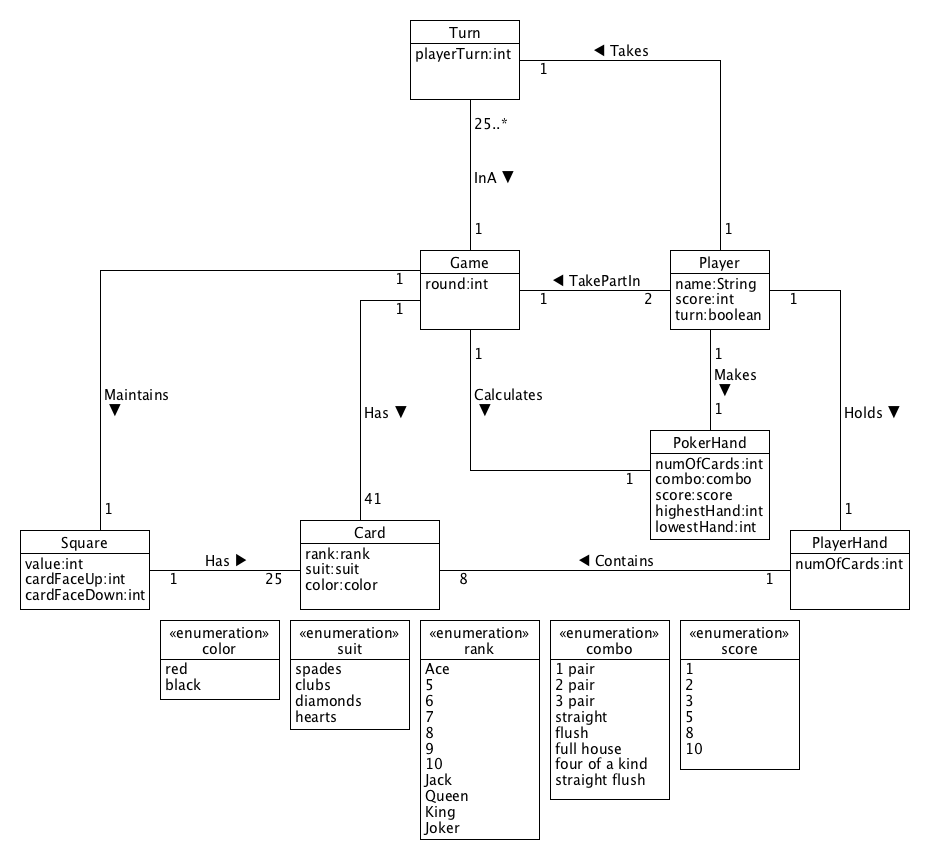
* 1. Class diagram

Figure 1.5 Initial Class diagram for Ganderpoke game

* 1. Class description

**Class** Game A game of Ganderpoke

**Attributes**

round Current round in the game

**Class** Player A player in the game

**Attributes**

name Name of the player

score A player has a score

turn A player has a turn in the game

**Class** PlayerHand A collection of cards currently held by the player

**Attributes**

numOfCards The number of cards currently held in players hand

**Class** PokerHand A poker hand of cards for the player

**Attributes**

numOfCards The number of cards currently held in poker hand

combo Different poker hand combos (1 pair, 2 pair, 3 pair, straight, flush, full house, four of a kind, straight flush)

score A score for each combo (1, 2, 3, 5, 8, 10)

highestHand A player makes a highest 5 card poker combination

lowestHand A player makes a lowest 5 card poker combination

**Class** Turn A players turn in the game

**Attributes**

playerTurn Decide which player turn it is

**Class** Card A card in the Ganderpoke game

**Attributes**

rank The type of card it is (Ace, 5,6,7,8,9,10,Jack, Queen, King, Joker)

suit The suit of card (spades, clubs, diamonds, hearts)

color A color of the card (black, red)

**Class** Square A square of cards that players pick up from

**Attributes**

value A value of the square once all cards are faced up

cardFaceUp Number of cards faced up in the square

cardFaceDown Number of cards faced down in the square

Invariants:

* The **Game** class is responsible for controlling the game flow and all classes.

Chapter 3: Dynamic Designs

This chapter focuses on Dynamic Designs where I will start to use my Conceptual Class Diagram (see chapter 2) in order to show how my classes will be working together to interact within the game. I will be producing Sequence Diagrams, which will be based on the use cases that I have produced. This chapter also introduces object diagrams and object specifications to show my object instances working to achieve a desired goal.

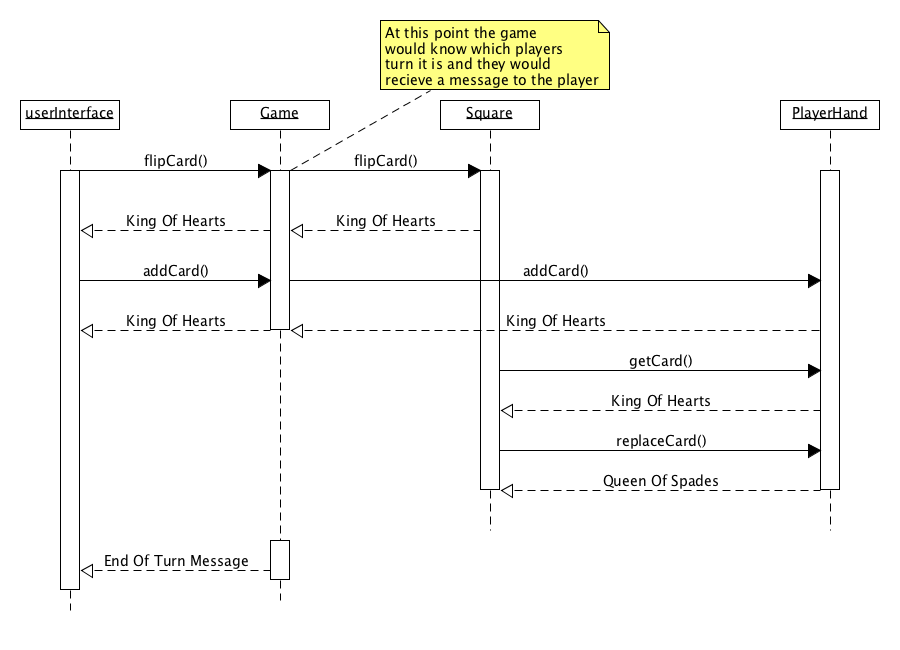
* 1. Sequence diagram

Figure 1.6 Illustrates taking a turn

Figure 1.6 shows us how the system classes would interact in order to allow the players to successfully take a turn in the Ganderpoke Game. For this sequence diagram above the player would call the **flipCard()** function in order to flip a card from the square, which would go through the **Game** class. Once the card is flipped it’s returned as *King Of Hearts* to the **userInterface** and then the card is added to the allocated **PlayerHand** using the **addCard()** function that would add the card into the player hand. Then to physically move the card, the player retrieved from their turn into their hand we would use the **getCard()** function to move the card and then use **replaceCard()** functionin order to replace the card space with a card from the **PlayerHand**. Then the last stage is where there is an end of turn message to the player.

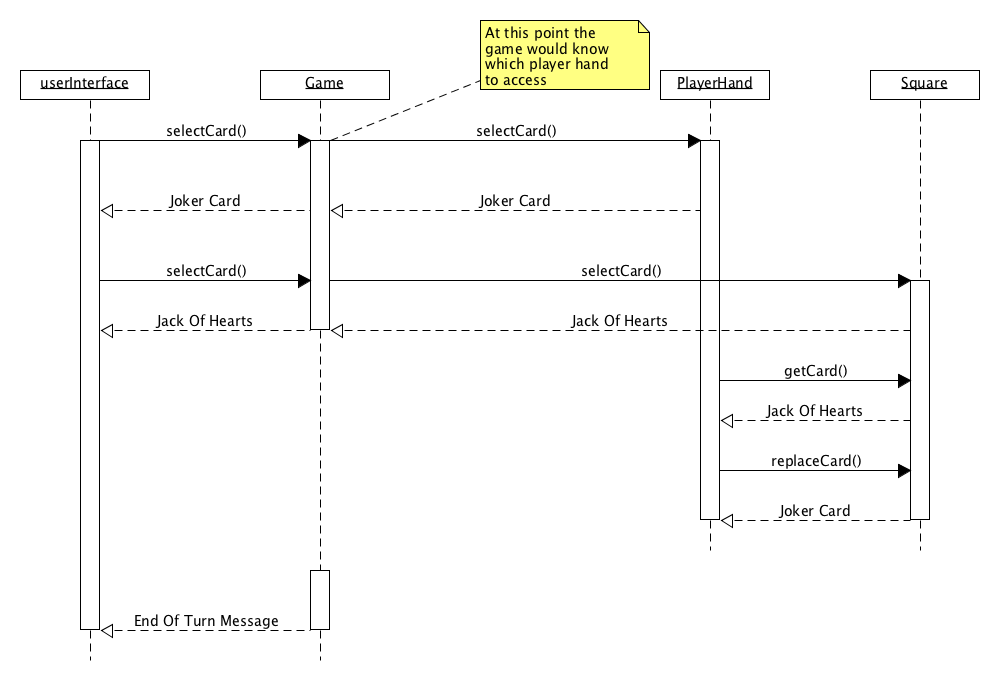
Figure 1.7 Illustrates discarding a joker card from player hand

Figure 1.7 shows us how the joker card is discarded from **PlayerHand** in order to replace it with a card from the **Square**. The player would need to first select a card from the **PlayerHand**, this is done using **selectCard()** function which would go through the **Game** class to access the **PlayerHand** class from where they can select a card to discard. From the diagram the *Joker* is selected and returned to the **userInterface** and then the player would need to select a card from the grid to swap with using **selectCard()** function again in order to select a card from the square this time (at this point the card that the player wants is already faced up) that the player wants to swap the joker card with, which in this case is the *Jack Of Hearts* card. Then to physically move the card, the player retrieved from their turn into their hand we would use the **getCard()** function to move the card and then use **replaceCard()** functionin order to replace the card space. Then the last stage is where there is an end of turn message to the player.

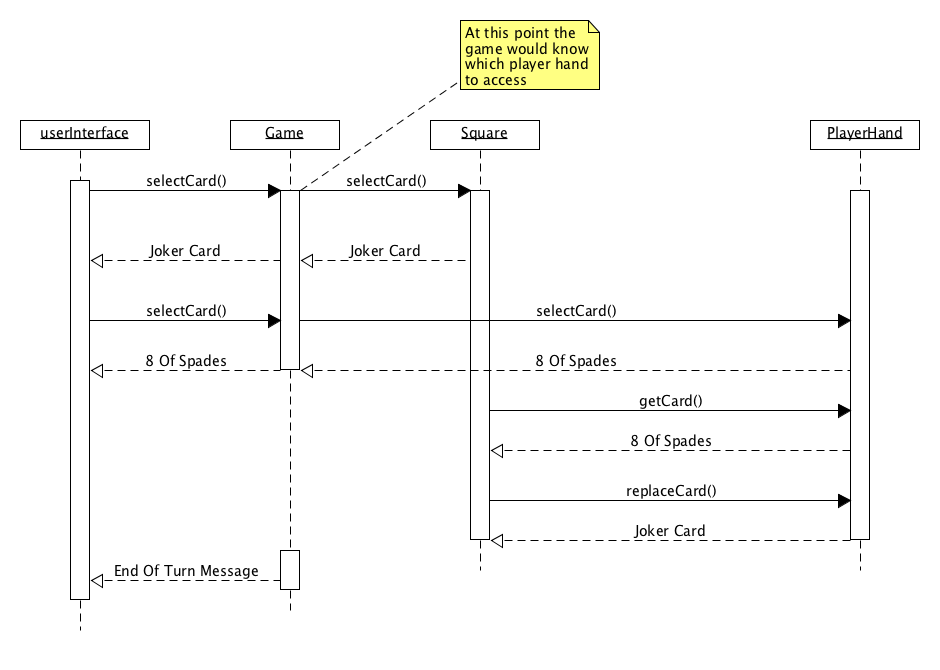
 Figure 1.8 Illustrates getting a joker card from the square

Figure 1.9 shows us how to retrieve a joker card from the square, which is a similar process to discarding a joker card. The **userInterface** would select a card using **selectCard()** function that go through to the **Game** class and access the **Square** where the *Joker Card* would be selected and then the player would need to select a card from their **PlayerHand** to swap with. To access the **PlayerHand** the player again selects a card using **selectCard()** function in order to access the **PlayerHand** and select a card from there to swap the *Joker Card* with, which in this case is *8 Of Spades*. Then to physically move the card, the player retrieved from their turn into their hand we would use the **getCard()** function to move the card and then use **replaceCard()** functionin order to replace the card space. Then the last stage is where there is an end of turn message to the player.

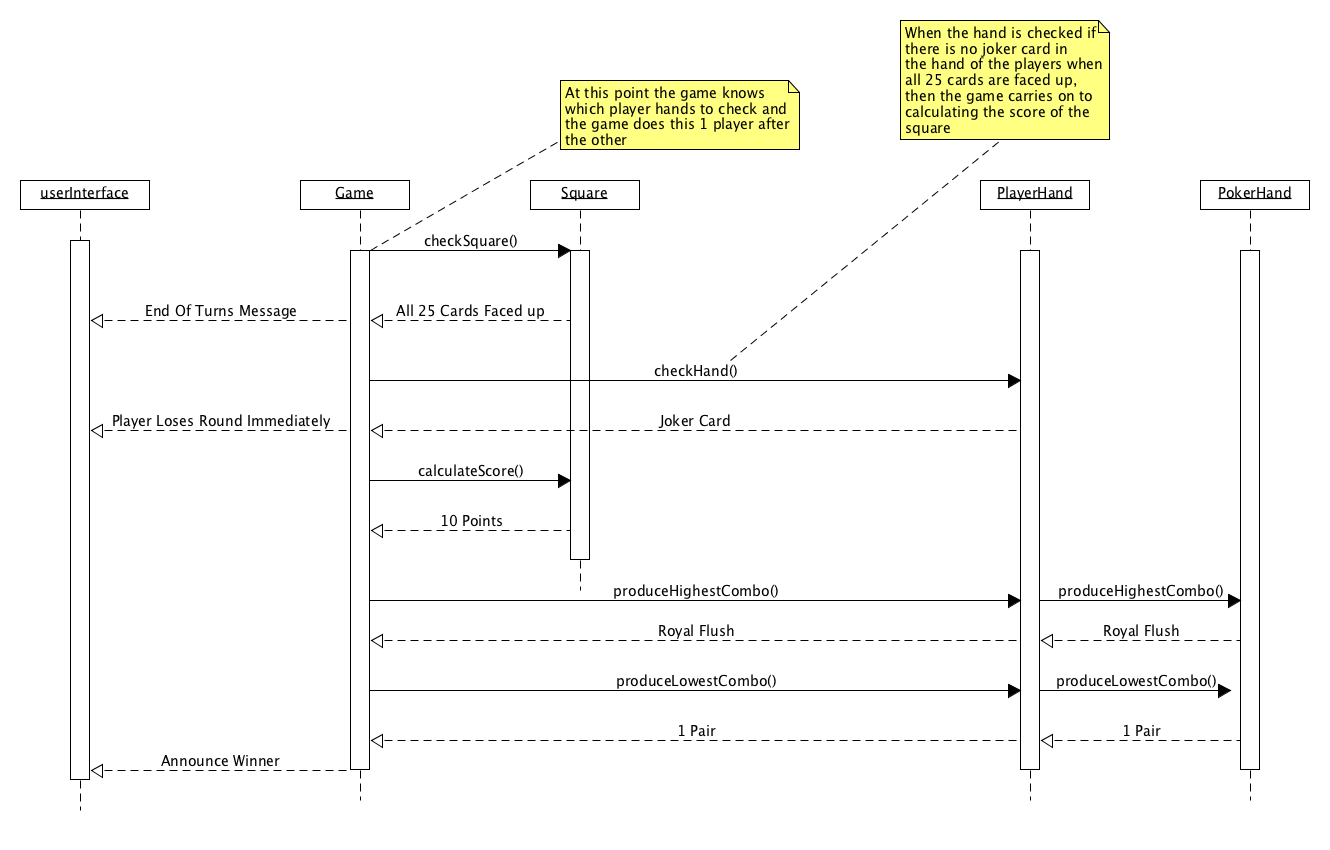


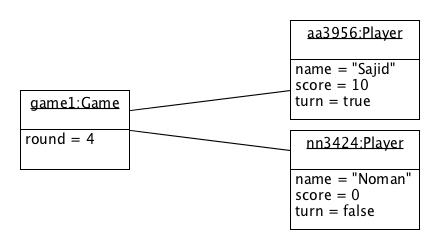
Figure 1.9 Illustrates the game checking for a winner

Figure 1.9 shows how the game would check for a winner, the first thing the game does is check the square to see if all 25 cards are faced up, if its true then it moves onto checking 1 players hand at once to see if they have a joker card in their hand. If the player holds the joker card, they lose immediately otherwise if no players have a joker card in their hands the game carries onto calculating the value of the square. Once that is done it produces the highest and lowest 5 card poker combination for the player and then announce the winner accordingly. If the player wins highest hand then they score 2x the value of the square, but they lose the lowest then other player wins 1x the value of the square.

* 1. Object Specification

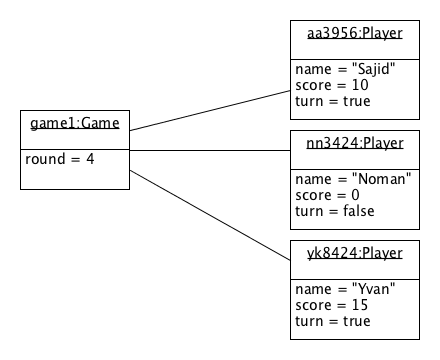
|  |  |  |
| --- | --- | --- |
| Identifier: | | OS01 |
| Context: | | CardController |
| Signature: | | getCard(String:rank, String:suit, String:color) : void |
| Pre-condition | The specified card that the player wants is to be retrieved from the square once flipped faced up and added to players hand | |
| Post-condition | The card is retrieved from square  The card is added to hand | |

|  |  |  |
| --- | --- | --- |
| Identifier: | | OS02 |
| Context: | | CardController |
| Signature: | | discardCard(String:rank, String:suit, String:color) : void |
| Pre-condition | The specified card that the player wants to discard from hand is swapped with from one from the square | |
| Post-condition | The card from player hand is discarded from hand and moved into the square | |



* 1. Object Diagram

Figure 1.10 Illustrates a valid object diagram for a game with two players

Figure 1.11 Illustrates an invalid object diagram for a game with three players

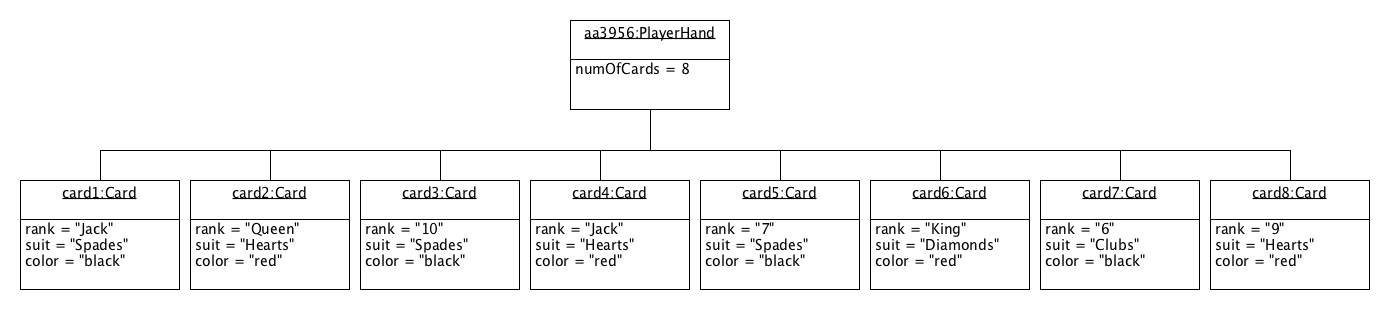


Figure 1.12 Illustrates a valid object diagram for a player hand containing 8 cards

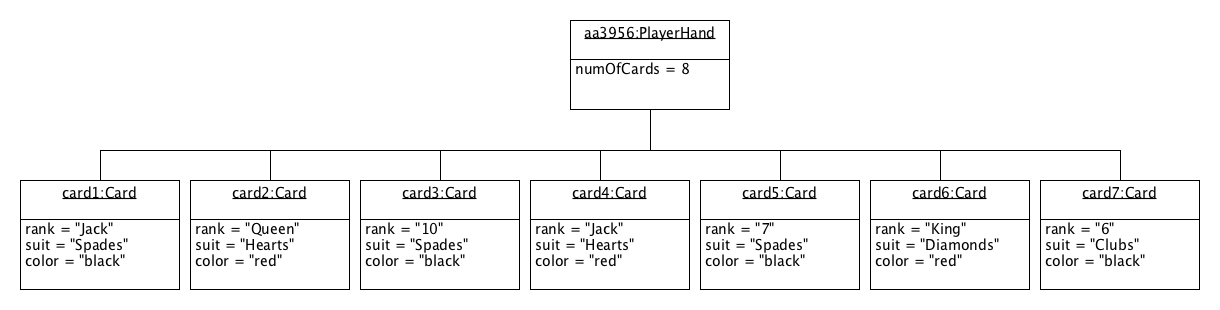


Figure 1.13 Illustrates an invalid object diagram for a player hand with 7 cards

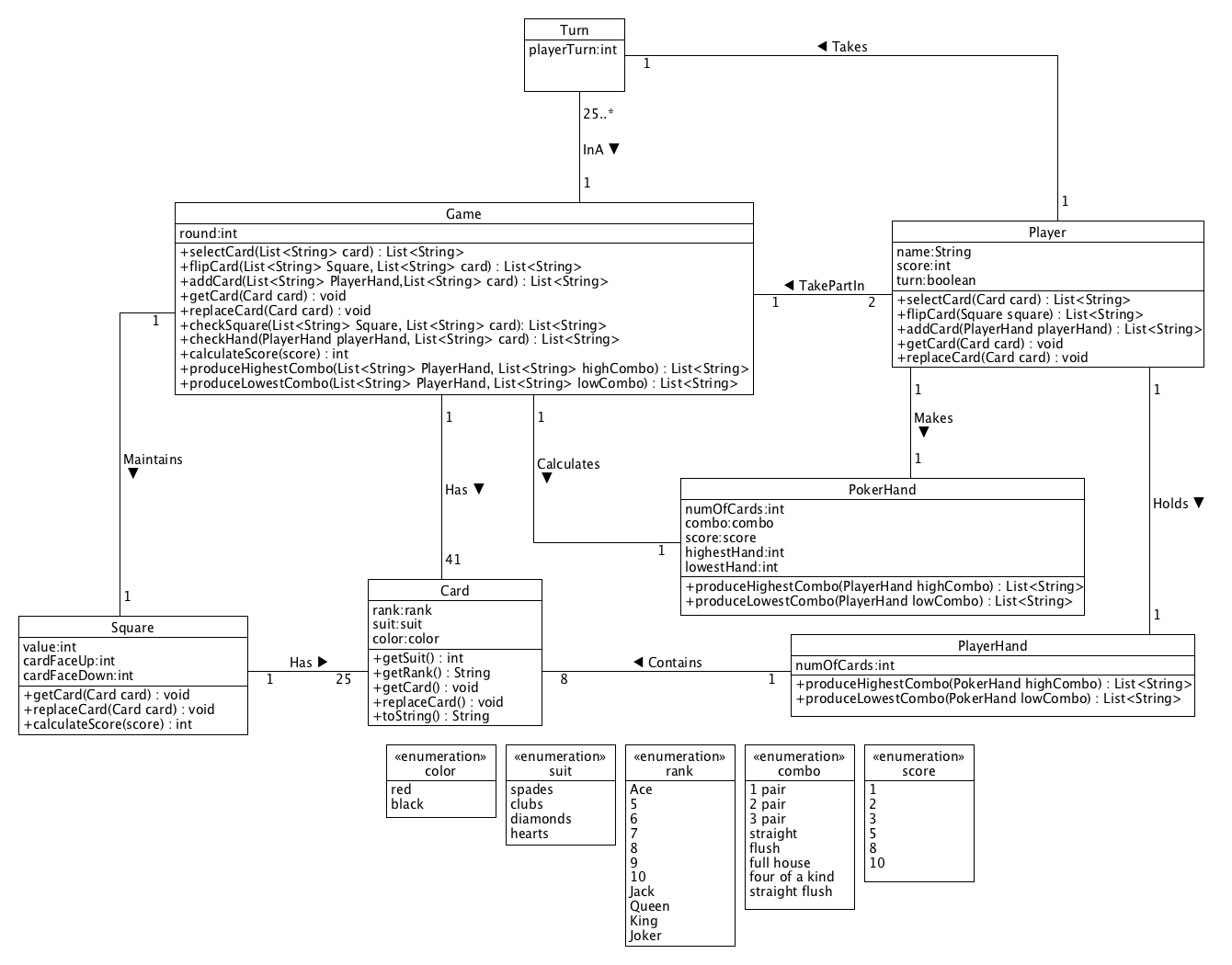
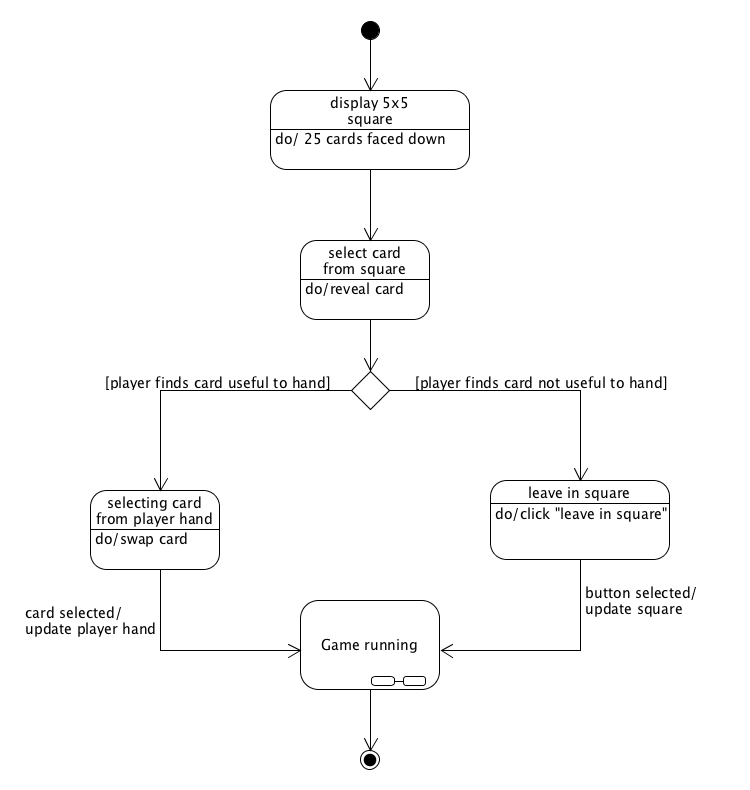


Figure 1.14 Illustrates a more detailed Class Diagram with methods

Figure 1.14 shows us how the Initial Class Diagram has been developed over the last chapter where have modelled our sequence diagrams in order to see how the classes will interact with each other.

Chapter 4: User Interface Interaction

In this chapter I will be focusing on the User Interface Interaction using UML State Diagrams. State diagrams allow us to be able to map out the user interface in a diagram in order to understand what type of interface is needed in order to successfully achieve the aim of the game for the user.



* 1. State diagrams

Figure 1.15 illustrates User Interface Interaction to start the Ganderpoke game

Figure 1.15 shows us how the user would interact with the User Interface in order to successfully start the game. The first thing the user would be shown is the 5x5 square from which the user can select a card to flip (reveal) and then decide if its useful to their hand or not. If its useful the player would select a card from their player hand and as soon as card is clicked in the player hand it would automatically switch with the card they just flipped from the square. The next state is “Game running” which has a substate (see figure 1.16).

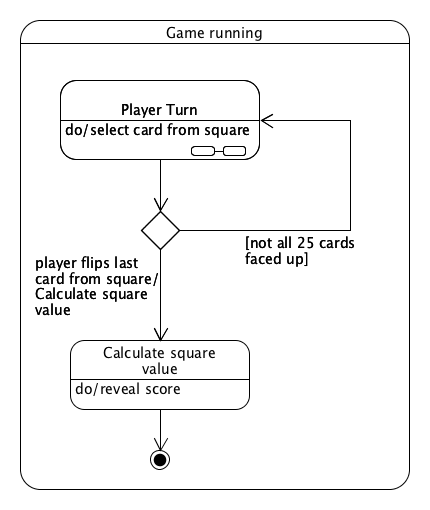


Figure 1.16 illustrates a substate of Game running

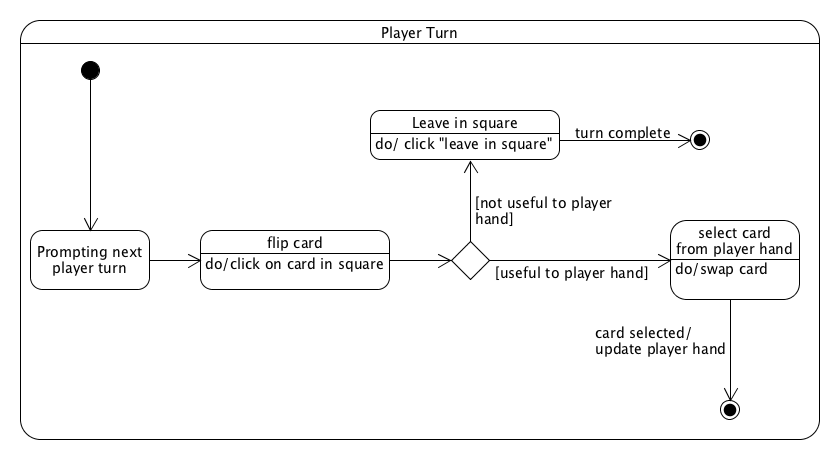
Figure 1.16 shows us what happens when the game is running and straight away the first state is “Player Turn” which is another substate (see figure 1.17) in the diagram. This diagram shows once the last card from the square is flipped the value of the square is calculated, however if there are still cards that are left in the square the player keeps taking turns until there no cards left faced down.

Figure 1.17 illustrates a substate of Player Turn

Figure 1.17 shows us the player takes a turn using the User Interface. The first step is the interface would display a message to the player’s turn it is and then they would go ahead and flip a card from the square. Once they have done so they would check the value if its useful to player hand they will select a card from their player hand and it would immediately swap into the square with the card they just flipped. However, if the player does not find the card useful to player’s hand they would leave it in the square by selecting the “leave in square button” and then end their turn.

Chapter 5: Testing

In this chapter I will be testing my design of the game in order to fix any bugs or errors that I would have made during the design and implementation stage. This would allow me to be able to make the system meet the needs of the client without missing any of the requirements that the system needed during the first stages of design.

* 1. Design and implementation verification