System design report on

The '*Garbo*' card game

(Includes requirements, analysis and design and the delivery the project)

Contents page

List of Figures

1 Chapter 1: Background and requirements for the Garbo card game

1.1 Context

The Garbo game is a card based game for two players which uses 32 cards of a standard 52 card pack (7 through ace of each suit). This game is based on a classic variety of 'avoid matching cards' card games such as Zoo Party and 7-Safari. The idea of the game is to place cards in a 4x4 grid in such way that avoids duplication of the suit of an adjacent card or the rank of a card in the same row or column**.** Any such match of suit or rank gives away points to one's opponent. The game consists of two levels. Each suit and rank has a point value shown below in figure 1.1:

Spades	1
Hearts	2
Clubs	3
Diamonds	4
Jack	5
King	6
Seven	7
Eight	8
Nine	9
Ten	10
Ace	11
Queen	12

Figure 1.1.1 Cards and their value

1.2 How to play Garbo

When the game begins the dealer (one of the two players) shuffles the cards and deals four cards to him and the other player. The rest of the cards are placed of the side face down. They will come into play later on. The non-dealer makes the first move by placing a card face up on the field (all other cards must be placed as that they are next to a card). Then it is the dealer's turn. He/she places a card. The dealer and the non-dealer take turns in placing cards in such way that the first sixteen cards played form a square of four rows and four columns. Each subsequent card must go side by side with a card already placed.

If a card is placed so that it matches the suit of any adjacent card either diagonally or orthogonally your opponent can claim the value of the matching suit. If a card is placed so that matches the rank of any card in the same row, column or diagonal your opponent can claim the value of the matching rank. If a card is placed so that it makes more than one such match your opponent can score them all.

When each player has placed his/her four cards another eight cards are dealt to each player (from the face down stack). The two players continue as before till the square contains sixteen, face up, cards. This completes the first level.

When the first level is completed the remaining cards are dealt out so each player has eight in hand. The player with the higher score at the end of the first half goes first or whoever played last to Level One if both players have equal points.

At each turn the player places a card on top of a face-up card that was previously placed during level one. But before placing the card (for level 2) the face-up card (from level one) must be turned over and rotated 90 degrees to show that it cannot be played on again.

Scoring continues as before but with an additional penalty. If a player places a card on top of a card ,that was placed during level 1, that matches on either rank or suit then the opponent scores ten times the amount of points that would a normal match would be, in addition to the normal scores for matching cards that are adjacent (by suit) or in line (by rank).

The game is over when the two players run out of cards and the player with the most points is the winner.

1.3 Product description

1.3.1 Overview

The aim of this project is to take the card game Garbo described above and turn it into a computer game in such way that the players will play the game locally sitting side-by-side at a PC (or laptop) and take turns using the same keyboard or mouse, in other words a two-player non-networked version of the game.

1.3.2 Differences between the game and the project

The way the real game different from the final goal of this project raises some questions as to how the game will be implemented such as: How the cards will be displayed on the screen as the two players will be looking at the same monitor in a way that prevents them from seeing each other’s hands.

And how the scoring will work as in the real live version of the game the players have to pay close attention to how the cards are positioned in relation to each other and claim the points after a card has been placed, if there are any. And

1.3.3 Possible solutions

To prevent players from seeing each other's hands, only the current player's hand should be displayed. At the begging of each player's turn the previous player's hand should be hidden and to show the current player's hand a user would confirm that they are ready to take their turn.

At the end of each turn the current player could be asked how many points they would like to claim and add to their score. However this could lead to incorrect inputs and annoy players if an unreasonable score is entered.

Or the scoring could be automatic. Meaning that the game will calculate the maximum amount of points that can be claimed at the end of a player's turn and add them to the appropriate player's score. However, this might make the game less exiting to play, as players would have less control over the game (as this action would be automatic).

A hybrid of the two methods, described above, would be ideal. Where the current player is asked how many points would they like to claim and the maximum amount of points that the player can claim is calculated. So that the user input is compared to the maximum amount of claimable points and if the user input is less than the maximum points they can claim the user input is added to their score, else the user is prompted to enter a score that can be achieved given the current field's state.

Although, the project will be developed following the automatic calculation and updating of a player score, as that would meets the project requirements and it will not be very difficult to later improve on and fuse with a user input.

1.4 Functional requirements

The software version of the game will have to be able to do the following actions:

- Create the pack of card required for the game
- Shuffle the pack of cards
- Deal the appropriate amount of cards
- Allow a player to select and play a card
(adding it to the field and removing it from the hand)
- Calculate score
- Record a player move
- Check that the player move was legal

•Provide feedback if move was illegal

• Record each player's points

• Display current game state

•Announce a winner

1.5 Quality requirements

•The game needs to be written in an object oriented way using Java 8 Standard Edition

• The software will be operating on NetBeans 8 IDE

• It has to be implemented in Microsoft Windows environment.

1.6 Workflow illustration(s) using UML activity diagrams

Figures 1.6.1 (on the right) shows the basic work follow of the game. It is broken down into a series of activity models to aid simplicity and interpretation. It describes the processes in the game at a high level however, it will be of great help when the developing of the game begins

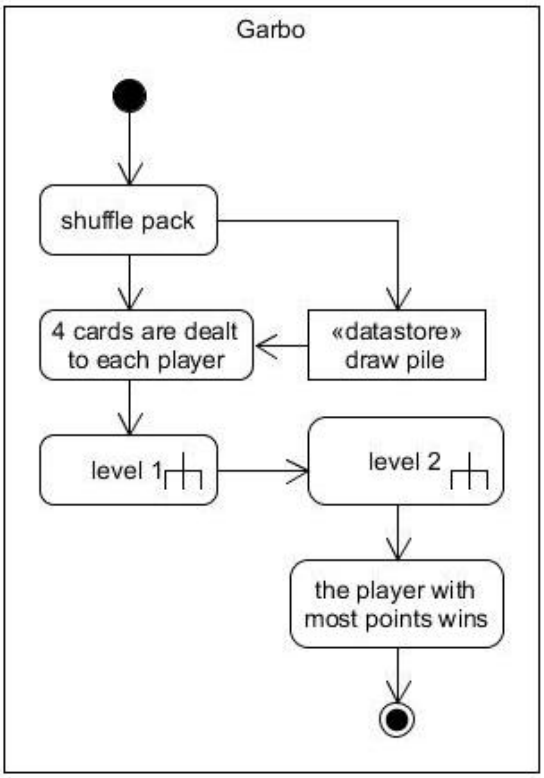

Figure 1.6.1 Workflow of the software version of the Garbo game

The workflow conveys shuffling the 32 card pack of cards and dealing four cards to each player, after which the players play through level one and two, at the end the player with most points wins.

The next workflow examples (see Figure 1.6.2 and 1.6.3) focuses in on the '*level1*' and '*level 2*' actions in the activity model shown on the right in Figure 1.6.1.

Figure 1.6.2 begins by letting player1 to take the first turn and play the first card. Then it is time for player2 to plaice a card. From now on after each player turn the score will be updated. The two players will continue to take turns until they both run out of cards. However, since the game will always start with player1 making the first move and both the players will have four cards in hand we only need to monitor the number of cards in player2's hand, as he would be the last to run out of cards . When player2 runs

out of cards eight cards are dealt to each player from the draw pail that originally had 32card in it. The two players continue to take turns until the 4x4 grid/field is full. This puts an end to level one.

At the beginning of level 2 (see figure 1.6.3) the reaming card are dealt out (the last 8). Then the player with the bigger score places a card or –if the two players have the same amount of points player2 is prompted to place a card. The two players continue to place card but this time this will place card on top of the cards placed during level 1. And if a card matches with the card underneath it gives away ten times the amount of points then it usually be

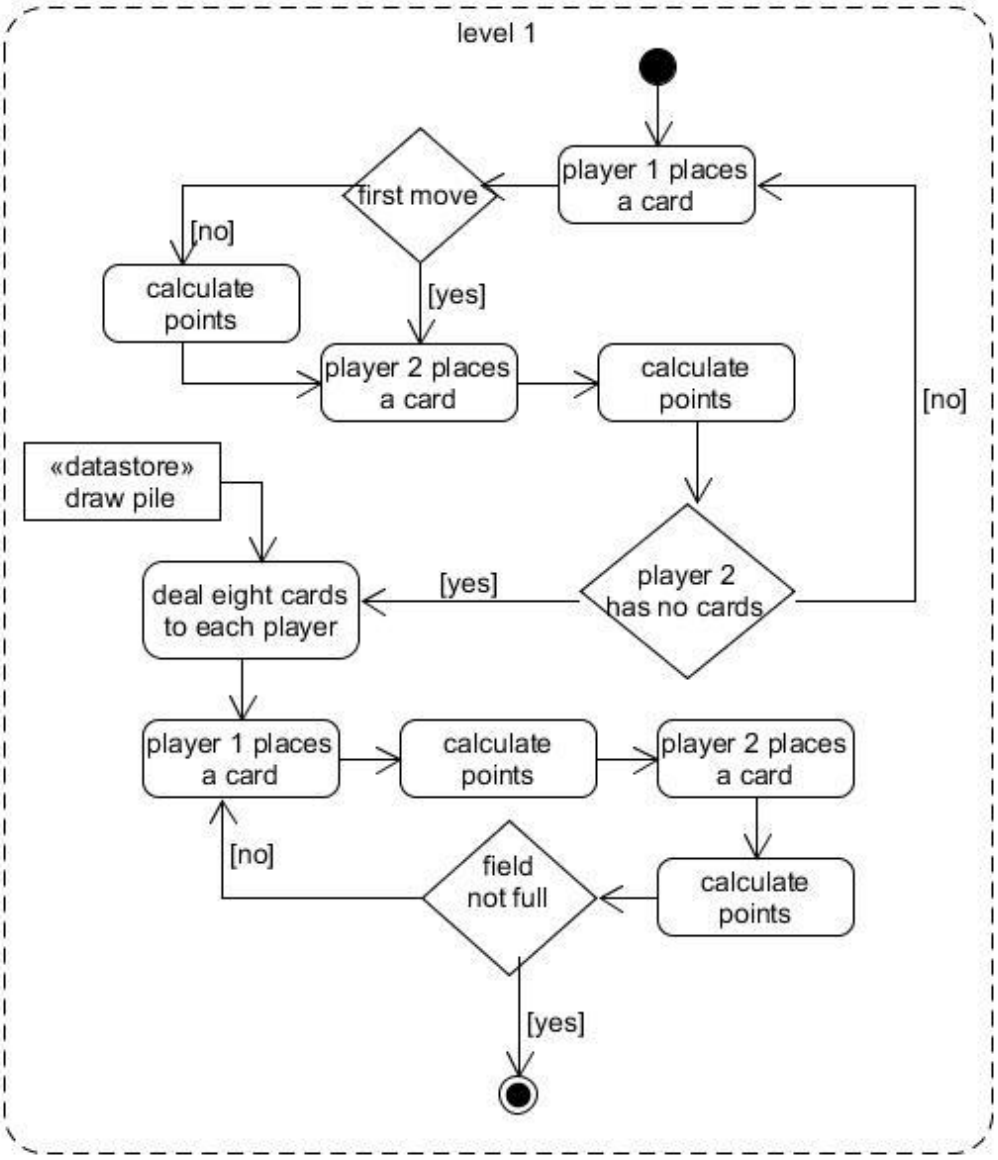

Figure 1.6.2 Work flow of level 1

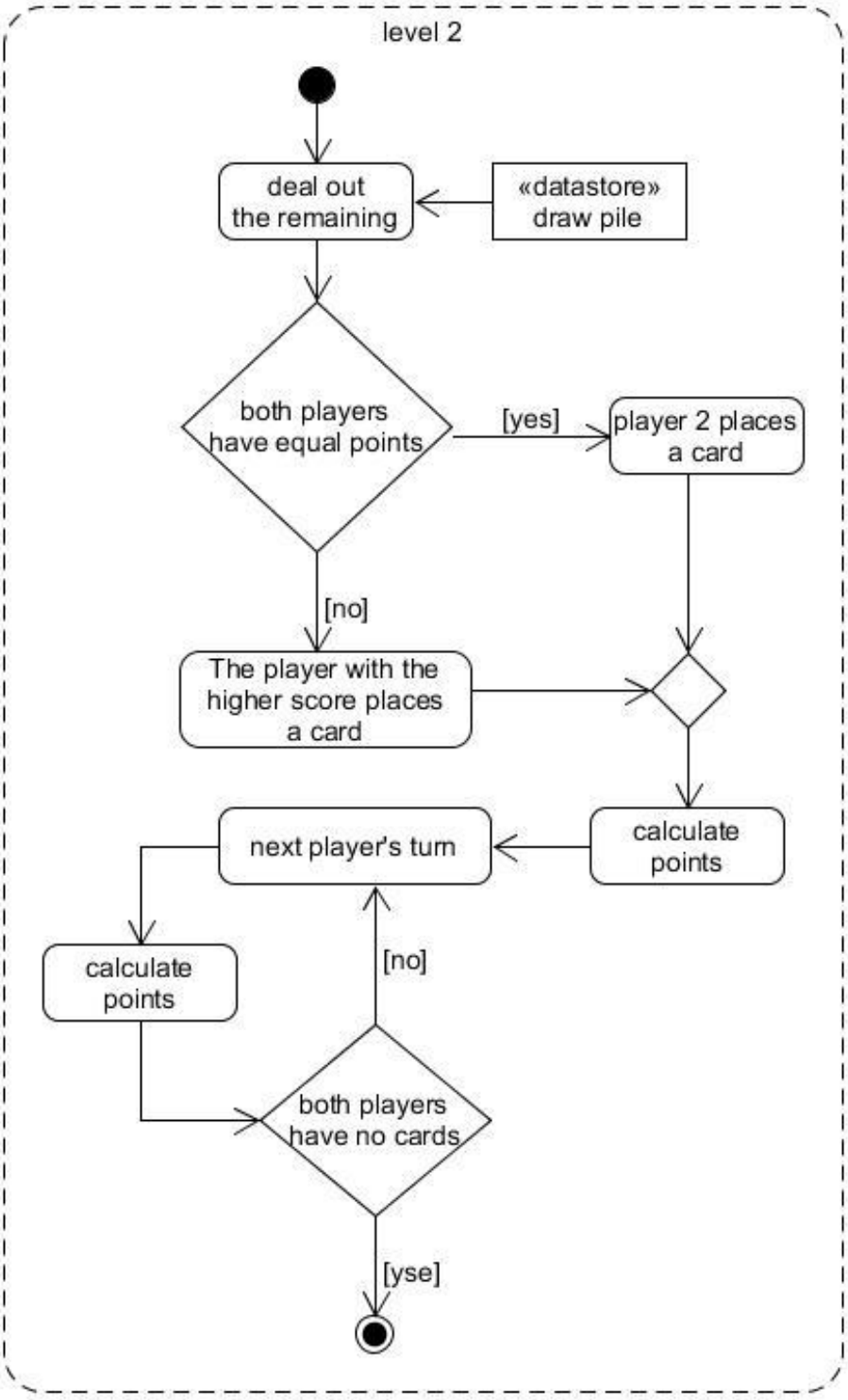

Figure 1.6.3 Work flow of level 2

1.7 Use cases for a main success scenario

In this section will take a look at the use cases involved in the Garbo game assuming each player plays appropriately and does not make any illegal moves.

1.7.1 Use case: Start a Garbo game

Name identifier version: Start a game UC01 Version 1.0
Initiator: Garbo
Goal: Start a game
Assumptions: two players have joined the game they know how to play, they have decided how will take the first turn, the pack of 32 cards is available.
Main scenario:

1. A 'welcome message' is displayed
2. The game pack is shuffled
3. Eight cards from the game pack are dealt between the two players(4 each)
4. The field is displayed
5. Player one is invited to take the first turn

1.7.2 Use case: play a card

Name identifier version: Play a card UC02 Version 1.0
Initiator: Garbo
Goal: To play one of the cards in the player's hand onto the field
Assumptions: it is the current player's turn and no player has yet won the game
Main scenario:

1. Player selects a card and where to place it
2. The card is added to the filed
3. The card is removed from player hand

1.7.3 Use case: calculate suit points

Name identifier version: calculate suit points UC03 Version 1.0
Initiator: Garbo
Goal: to calculate points for a player
Assumptions: a game is in progress a player has just placed a card and checks are made after each player turn

Main scenario:

1. Get the suit of placed card
2. Check if card matches the suit of any adjacent card
3. Calculate how many points the suit matching is worth

1.7.4 Use case: calculate rank points

Name identifier version: calculate rank points UC04 Version 1.0
Initiator: Garbo
Goal: to calculate points for a player
Assumptions: a game is in progress a player has just placed a card and checks are made after each player turn

Main scenario:

1. Get the rank of the placed card
2. Check if card matches the rank of any card in the same row , column and diagonally
3. Calculate how many points the rank matching is worth

1.7.5Use case: calculate card underneath points

Name identifier version: calculate card underneath points UC05 Version 1.0
Initiator: Garbo
Goal: to calculate points for a player
Assumptions: a game is in progress a player has just placed a card and checks are made after each player turn

Main scenario:

1. Check if there is a card underneath
2. Check if the two card match suits
3. Calculate how many points the suit matching is worth
4. Check if the two card match in rank
5. Calculate how many points the rank matching is worth

1.7.6Use case: calculate points

Name identifier version: calculate points UC06 Version 1.0
Initiator: Garbo
Goal: to calculate points for a player
Assumptions: a game is in progress a player has just placed a card and checks are made after each player turn

Main scenario:

6. Get the current player's points
7. Calculate suit points UC03
8. Calculate rank points UC04
9. Calculate card underneath points UC05
10. Calculate how many points should be added to the player's points
11. Add points to player's current points

1.7.7 Use case: beginning of Level 2

Name identifier version: beginning of Level 2 UC07 Version 1.0
Initiator: Garbo
Goal: To end level 1 and start level 2
Assumptions: the game is in progress and checks are made after each player turn
Main scenario:

1. The playing field is full
2. The rest of the dealing pile is dealt out(the last 8 cards)
3. Players will now plays cards on top of existing cards

1.7.8 Use case: hide/show hand

Name identifier version: hide/show hand UC08 Version 1.0
Initiator: Garbo
Goal: To prevent players from seeing each other's hands
Assumptions: The game is in progress, the hide/show cards is performed after each player turn, the game is played on a laptop
Main scenario:

1. Current player's turn begins
2. Current player interacts with the game to display hand
3. Current player plays a card
4. Current player's hand is hidden
5. Next player turn

1.7.9 Use case: Announce winner time

Name identifier version: Announce winner time UC09 Version 1.0
Initiator: Garbo
Goal: To determine when a winner needs to be announced
Assumptions: The game is in progress and checks are made after each player turn
Main scenario:

1. A check is made with regard to the number of cards remaining in the dealing pail
2. A check is made with regard to the number of cards remaining in the hand of player1
3. A check is made with regard to the number of cards remaining in the hand of player 2
4. If the number of cards remaining in the dealing pack, player1's hand and player2's hand is zero. It is time to check who won the game

1.7.10 Use case: Who won

Name identifier version: Who won UC10 Version 1.0
Initiator: Garbo
Goal: To identify a winner
Assumptions: The game is in progress and will end after the winner is announced
Main scenario:

1. The score of player 1 is retrieved
2. The score of player 2 is retrieved
3. The two scores are compared
4. The player with more points is announced as the winner
5. Game over

2 Chapter 2: Initial conceptual model

2.1. Class diagram

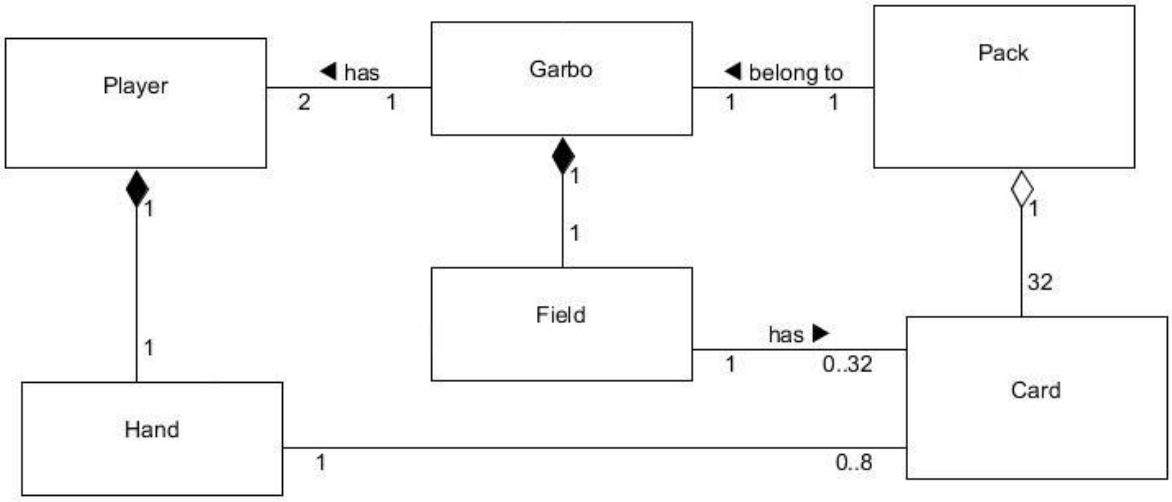

Figure 2.1.1 Class Diagram for Garbo game

2.2 Class description

Class Card A card in the Donut game

Aggregation to pack

Attributes

rankValue	The points for matching by rank
suitValue	The points for matching by suit
suit	The suit of the card (Hearts Spades Clubs Diamonds)
rank	The rank of the card (seven thought ace)

Class Pack The dealing pack

Attributes

Cards The cards that the pack has

Class Garbo A game of Garbo

Attributes

currentPlayer The player how's turn it is

notCurrentPlayer The player how's turn it is not

Class Filed The 4x4 grid that cards will be placed on

Composition to Garbo

Attributes

grid The current state of the play filed

Class Player A person (client) who is playing the game

Attributes

Score The amount of points a player currently has

Class hand A collection of cards currently held by a player in a game

Composition to player

Attributes

none

2.3 Invariants.

•The card objects of class Cards that will be linked to an object of class Hand must be obtained from an object of class Pack.

•The cards that are linked to an object of class Hand must not be linked to any other object.

•The cards that are linked to an object of class Pack must not be linked to any other object.

• The card objects of class Card that will be linked to an object of class Field must come from on object of class Hand.

• The card objects of class Card that will be linked to an object of class Field must not be linked to any other object.

•If aPlayer object of class Player is linked to aHand object of class Hand and the hand object aHand is linked to a number of card objects of class Card then the

3 Chapter 3: Dynamic designs

Earlier we designed the static modelling of the system and some use cases. In this chapter we will take a look at the use cases we have and design sequence diagrams for them. Starting with use case 1 'Start a game'

3.1 Use Case 1 'Start a game'

Name identifier version: Start a game UC01 Version 1.0
Initiator: Garbo
Goal: Start a game
Assumptions: two players have joined the game they know how to play, they have decided how will take the first turn, the pack of 32 cards is available.
Main scenario:

1. A 'welcome' message is displayed
2. The game pack is shuffled
3. Eight cards from the game pack are dealt between the two players(4 each)
4. The field is displayed
5. Player one is invited to take the first turn

Action 1 of use case can be achieved with a console output method or a pop up window.
Action 2 could be a simple function that randomises the order of cards in the pail
Action 3 needs retrieve a certain amount of cards (in this case eight) from the pile and add then to a player's hand and remove the cards from the collection of cards. As this act requires the complaint of several tasks (that we will have to complete several times throughout the game) a specific function will be created, to improve code readability and reduce code duplication with the fallowing details:

Identifier: dealCards version 1.0
Context : Garbo
Signature : dealCards(int numOfCards) : void
Preconditions : there are cards in the pile, it is time to deal cards
Postcondition : the cards in each player's hand has increased by the necessary amount

Action 4 and 5 are console output method

A sequence diagram that fallows use case 1 can be seen below in figure3.1.1

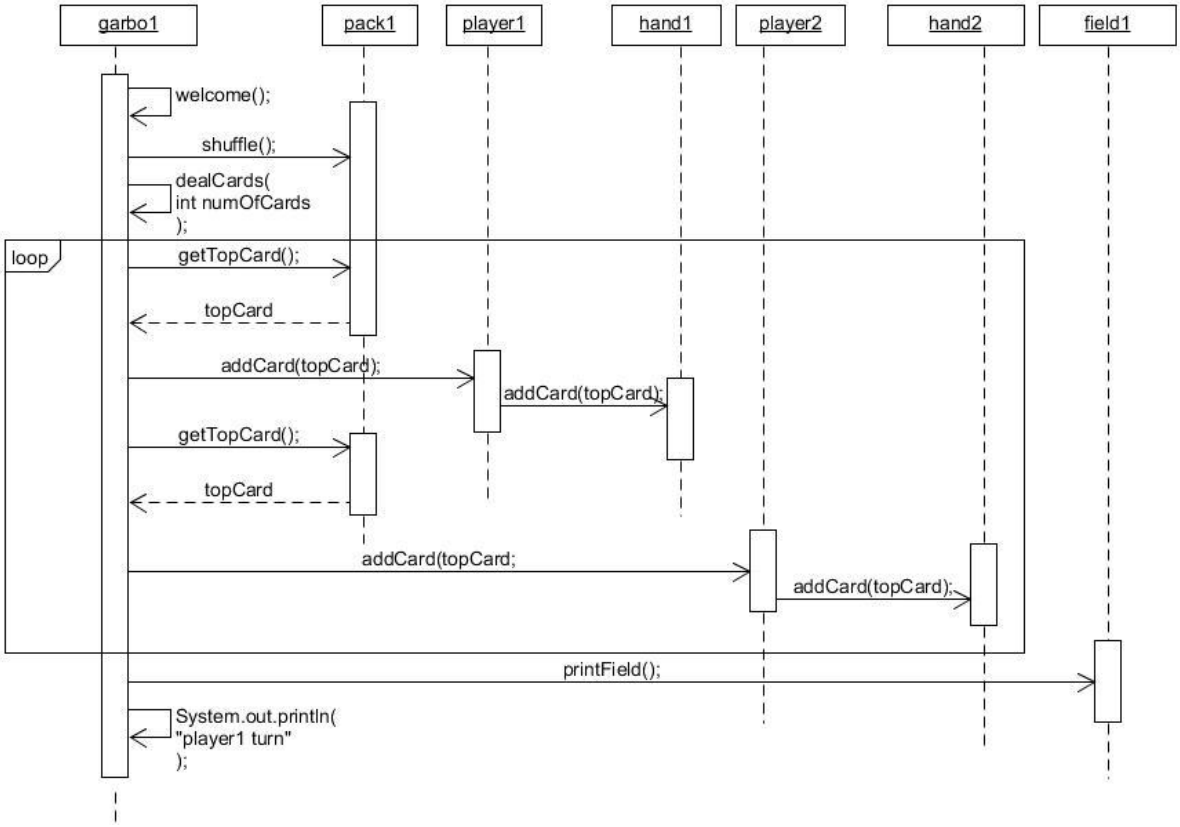

Figure 3.1.1 start a game sequence diagram

3.2 Use Case 2 'take a turn'

Name identifier version: play a card UC02 Version 1.0
Initiator: player
Goal: To play one of the cards in the player's hand
Assumptions: It is the current player's turn and no player has yet won the game
Main scenario:

1. Player selects a card and where to play it
2. The card is added to the filed
3. The card is removed from player hand

In figures 3.2.1 and 3.2.2 we can see how the object diagram of the Garbo game would look like before and after a card is placed.

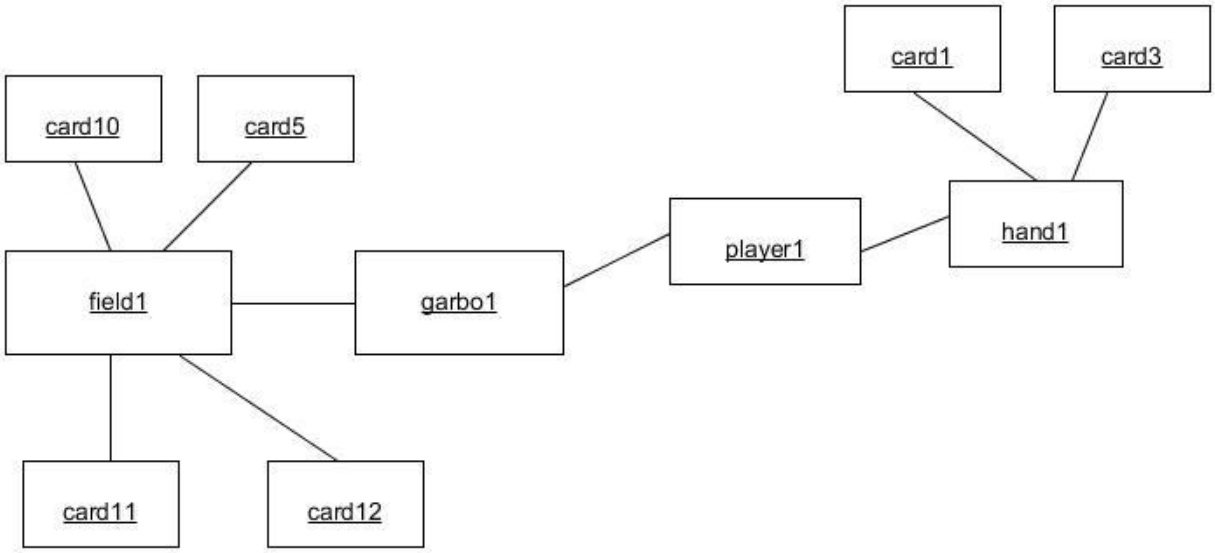

Figure 3.2.1 Before a Card is Played

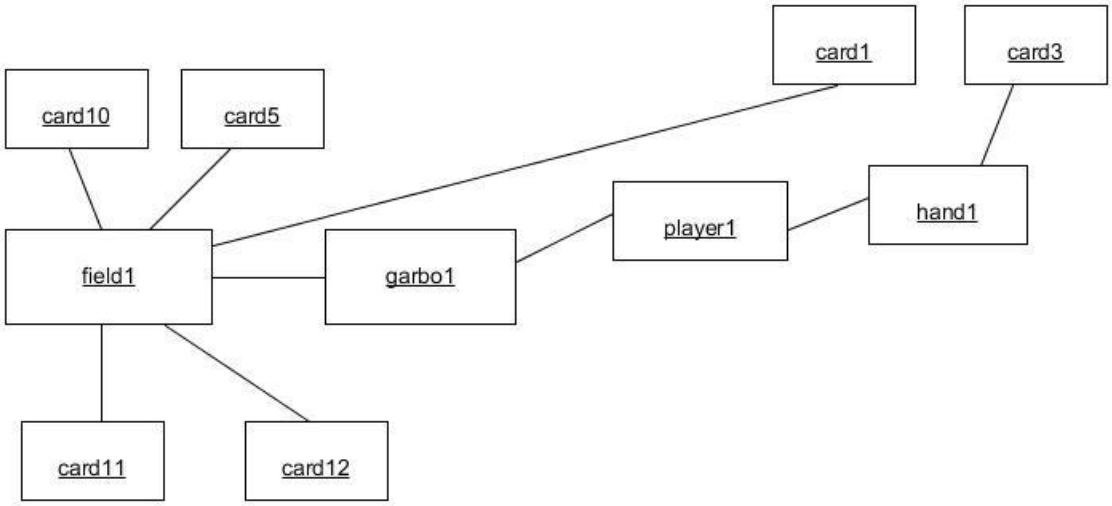

Figure 3.2.2 After a Card is Played

As the use case impales we are going to need a method that removes a specific card from a ‘hand’ and adds that specific card to a specific plaice in ‘field’. The method is described in farther detail below

Identifier m3 version 1.0
Context: Player
Signature: play(Card : aCard int row int columns) : void
Preconditions : the player must have at least one card in his hand, no one has won the game yet, it is the current player’s tern
Postcondition : the card is added to the specified plaice in ‘field, the card is removed from the hand

The sequence diagram can be seen in figure 3.2.3

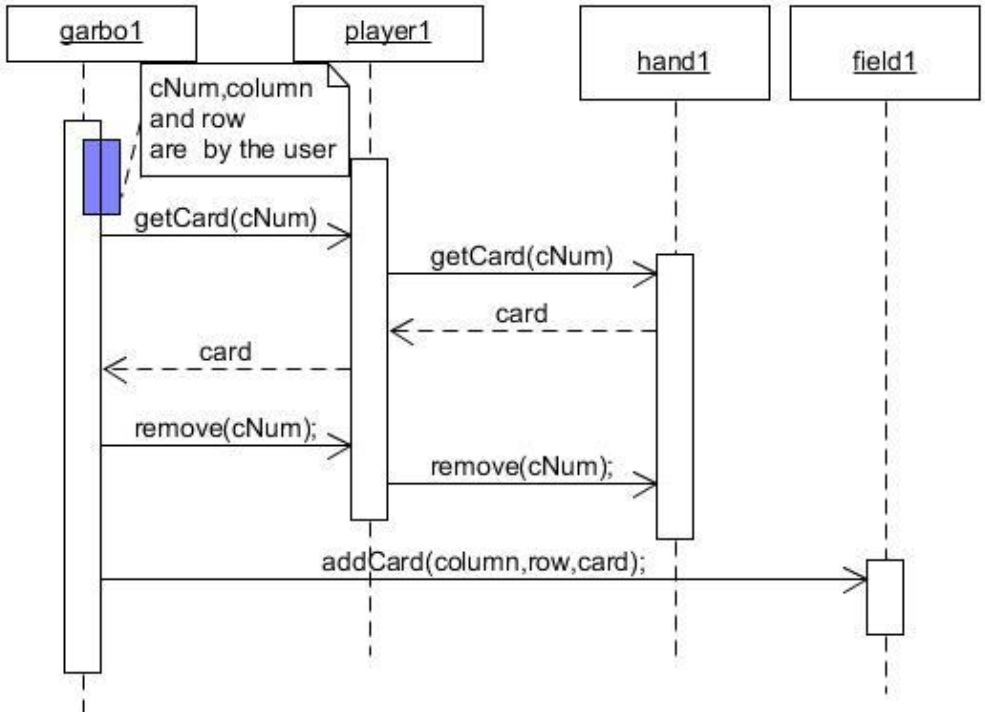

Figure 3.2.3 - ’Play a Card’- Sequence Diagram

3.3 Use case 6 ‘calculate points’

Name identifier version: calculate points UC06 Version 1.0
Initiator: Garbo
Goal: to calculate points for a player
Assumptions: a game is in progress a player has just placed a card and checks are made after each player turn

Main scenario:

1. Get the current player’s points
2. Calculate suit points UC03
3. Calculate rank points UC04
4. Calculate card underneath points UC05
5. Calculate how many points should be added to the player’s points
6. Add points to player’s current points

Here we will need several methods to calculate the total amount of points that a made during a turn.(match with suits, match with rank and match with the suit or rank of the card underneath)

Identifier calSuitPoints version 1.0
Context: Garbo
Signature: calSuitPoints (Card : aCard int row int columns) : void
Preconditions : the game is in progress, the player must have just played a card, it is the current player’s tern
Postcondition : the not current player’s score has increased with the amount of points that the current player’s move are worth

Identifier calRankPoints version 1.0
Context: Garbo
Signature: calRankPoints (Card : aCard int row int columns) : void
Preconditions : the game is in progress, the player must have just played a card, it is the current player’s tern
Postcondition : the not current player’s score has increased with the amount of points that the current player’s move are worth

Identifier level2calPoints version 1.0
Context: Garbo
Signature: level2calPoints (Card : aCard int row int columns) : void
Preconditions : the game is in progress, the player must have just played a card, it is the current player's tern
Postcondition : the not current player's score has increased with the amount of points that the current player's move are worth

The sequence diagram can be seen in figure 3.3.1

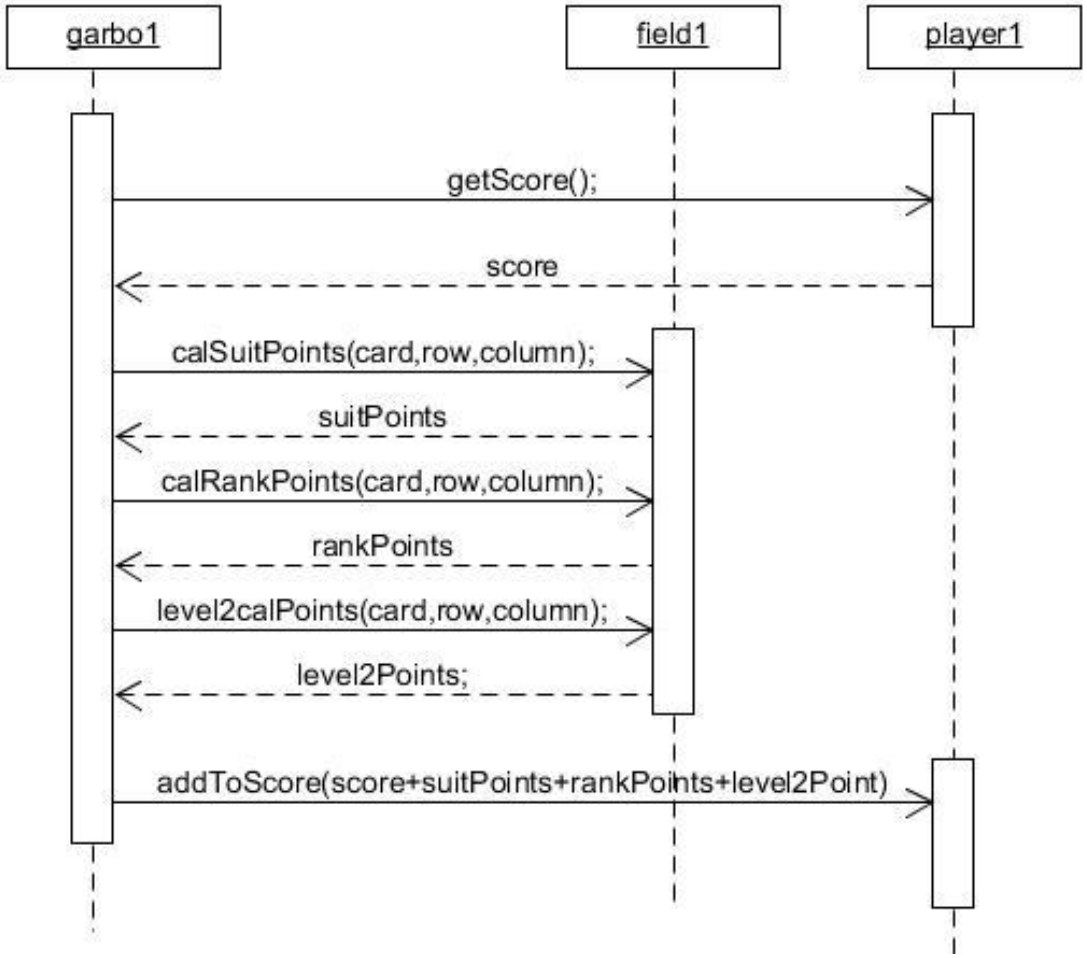

Figure3.3.1' Calculates Points'- Sequence Diagram

Although the sequence in figure 3.3.1 will calculate the new player's score improvements can be made. The individual calculation of the points acquired from suit and rank is not really required. It will be preferred to have a single method that calculates the points.

A sequence diagram for that purpose can be seen in figure 3.3.2

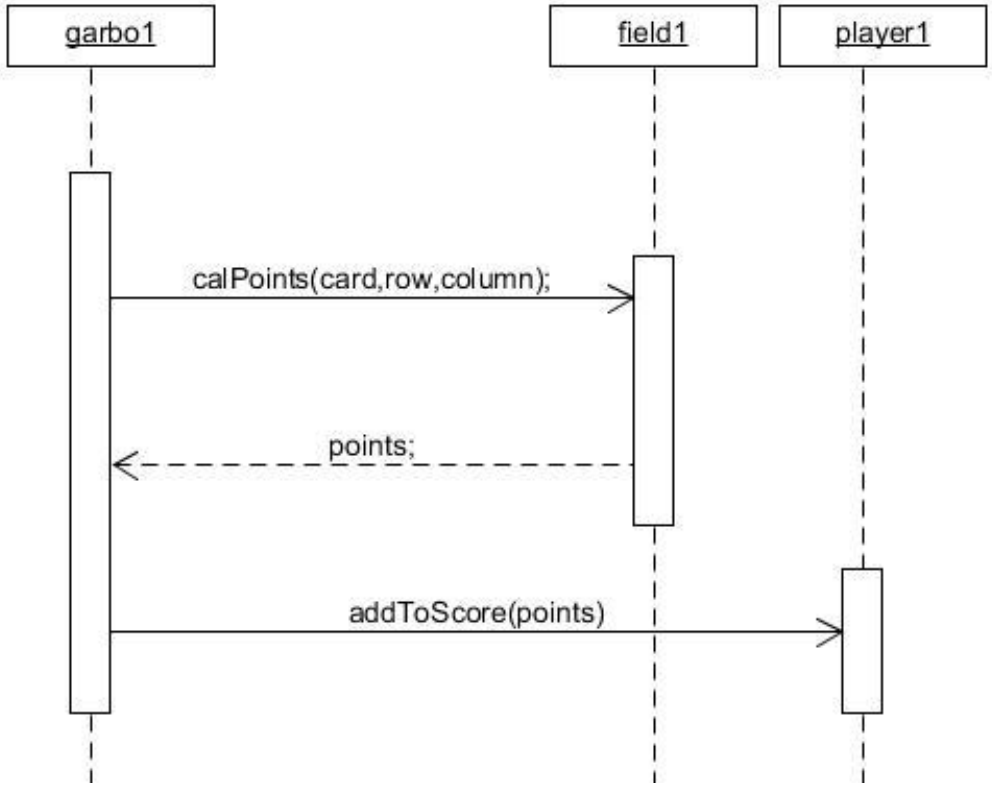

Figure 3.3.2 'Improved Calculate points'- Sequence Diagram

3.4 Use case Win time

Name identifier version: Win time UC09 Version 1.0
Initiator: Garbo
Goal: To monitor the current game for a win situation
Assumptions: The game is in progress and checks are made after each player turn
Main scenario:

1. A check is made with regard to the number of cards remaining in the dealing pail
2. A check is made with regard to the number of cards remaining in the hand of player1
3. A check is made with regard to the number of cards remaining in the hand of player 2
4. If the number of cards remaining in the dealing pack player1's hand and player2's hand is zero it is time for a winner to be announced

As the use case impales we are going to need a method that checks if the number of cards in the dealing pail and in the hands of player1 and player2 is zero. If that is the case it would be time to find out who won the game.

Identifier m5 version 1.0
Context: Garbo
Signature: winTime () : Boolean
Preconditions : a game of Garbo is in progress , 32 cards have been played from a player's hand onto the field
Postcondition : the output of the function needs to link to a function that determinates how the winner is.

The sequence diagram can be seen in figure 3.4.1

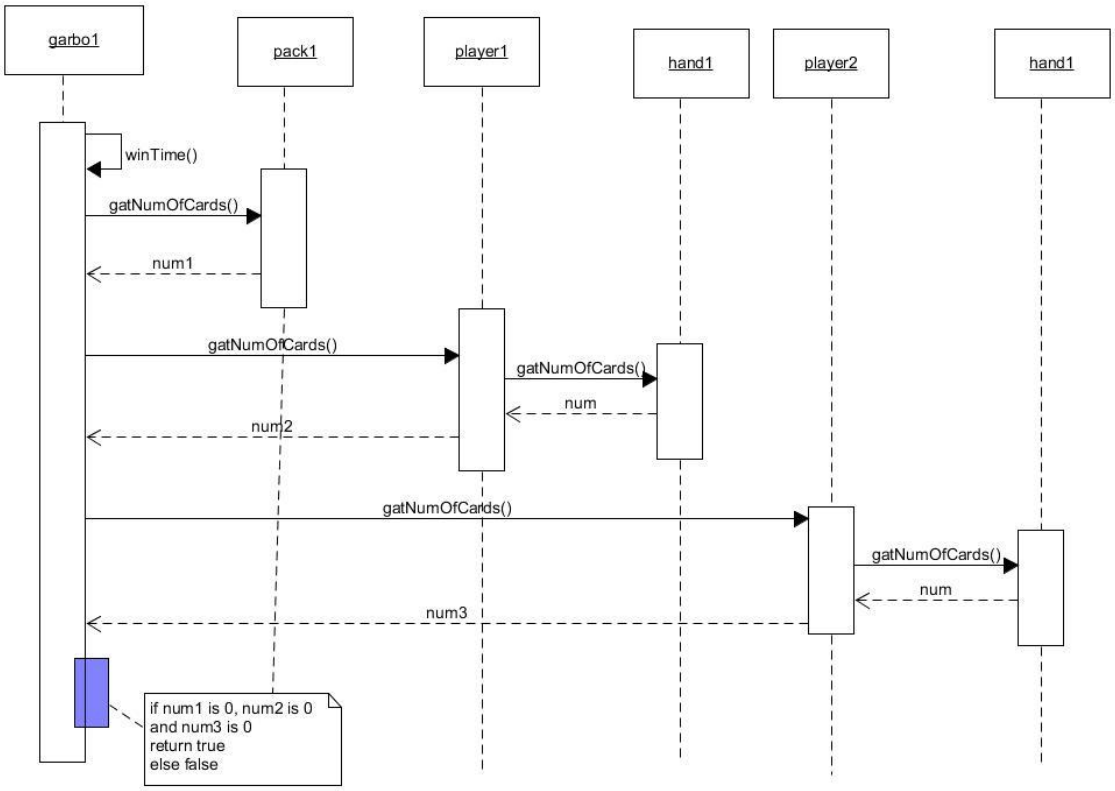

Figure 3.4.1 winTime - Sequence Diagram

3.5Use case who won

Name identifier version: Who won UC10 Version 1.0
Initiator: Garbo
Goal: To identify a winner
Assumptions: The game is in progress and it is announce winner time
Main scenario:

1. The score of player 1 is retrieved
2. The score of player 2 is retrieved
3. The two scores are compared
4. The player with more points is announced as the winner
5. Game over

As the use case impales we are going to need a method that retrieves the score of the two players and compares them. The player with the higher score will be announced as the winner in the console or in a popup window if a graphical user interface is used.

Identifier m5 version 1.0
Context: Garbo
Signature: winner() : void
Preconditions : a game of Garbo is in progress , winTime() is true
Postcondition : the player with the higher score is announced as the winner

The sequence diagram can be seen in figure 3.5.1

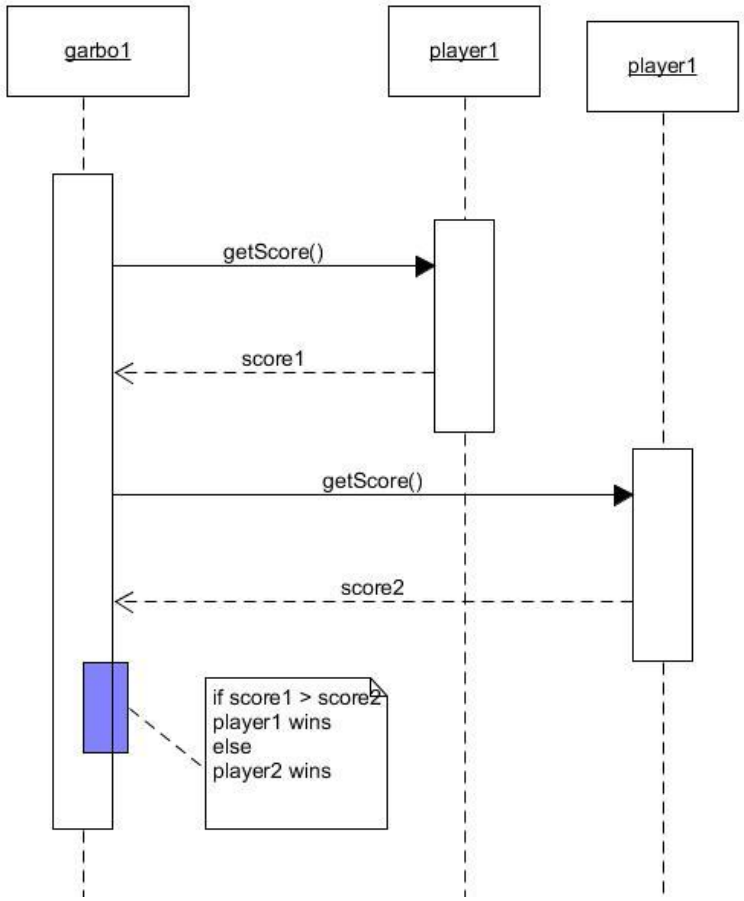

Figure3.5.1 - howWon- Sequence Diagram

3.6 New class diagram

By considering the UML sequence diagrams discoursed above we can conclude that the mentioned methods will be a necessary addition to the game

Figure 3.6.1 below shows how the class diagram looks after adding in the methods and attributes seen in the sequence diagrams

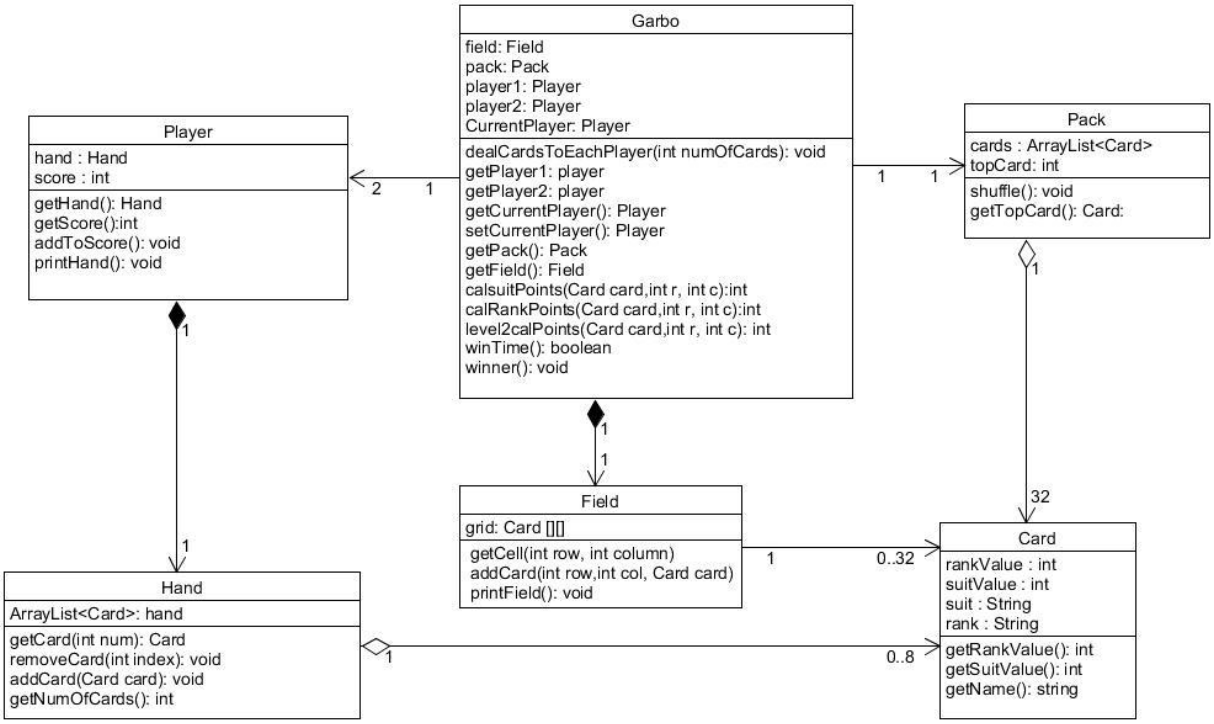

Figure 3.6.1 Updated Class Diagram

Chapter 4: User interface interaction

4.1 UML state machine diagrams of the game logic

If we look at the Garbo game as a state machine we would see that there are four main states the game can be in: deal cards player turn filed update and calculate points. Now since we are designing a software version of the game we will also need three more states for beginning of the game (dealing the first set of cards) error messages i.e. when a player attempts to make an illegal move and one more for declaring a winner.

The UML state machine diagram for the software version of the Garbo game can be seen in figure 4.1.1

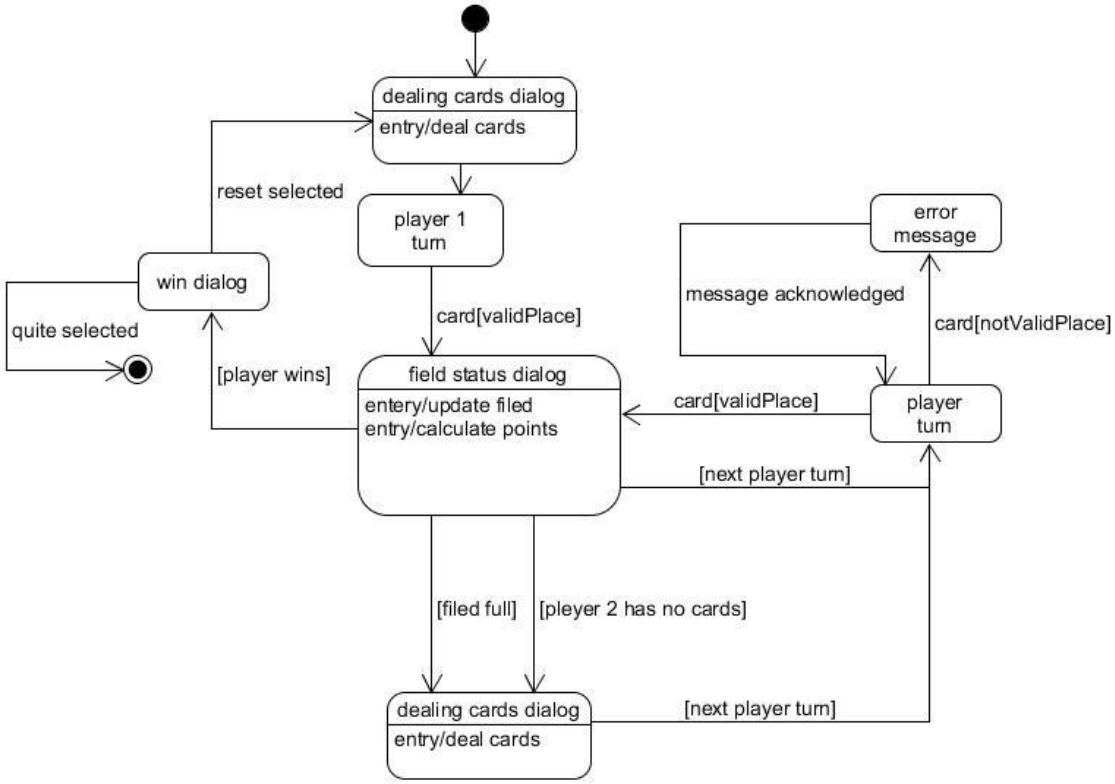

Figure 4.1.1 UML State machine diagram for the software version of the Garbo game

4.2 User-system interaction

When designing the user interface there a few rules to keep in mind: The interface needs to be easy to understand and navigate though. The widgets need to be positioned logically and their prepuce should be intuitive.

The user interface needs to provide a clear and easy to understand view of the grid, the player's hand, the game's state , the dealing pile, the points that each player has acquired in their last turn and though out the game.

A dialog box is also necessary as there is a need to display messages i.e. when a player tries to make an illegal move or why a certain amount of points has been added to a players score.

A sample of how the user interface, for the Garbo game, cloud look like is provided in figure 4.2.1, below.

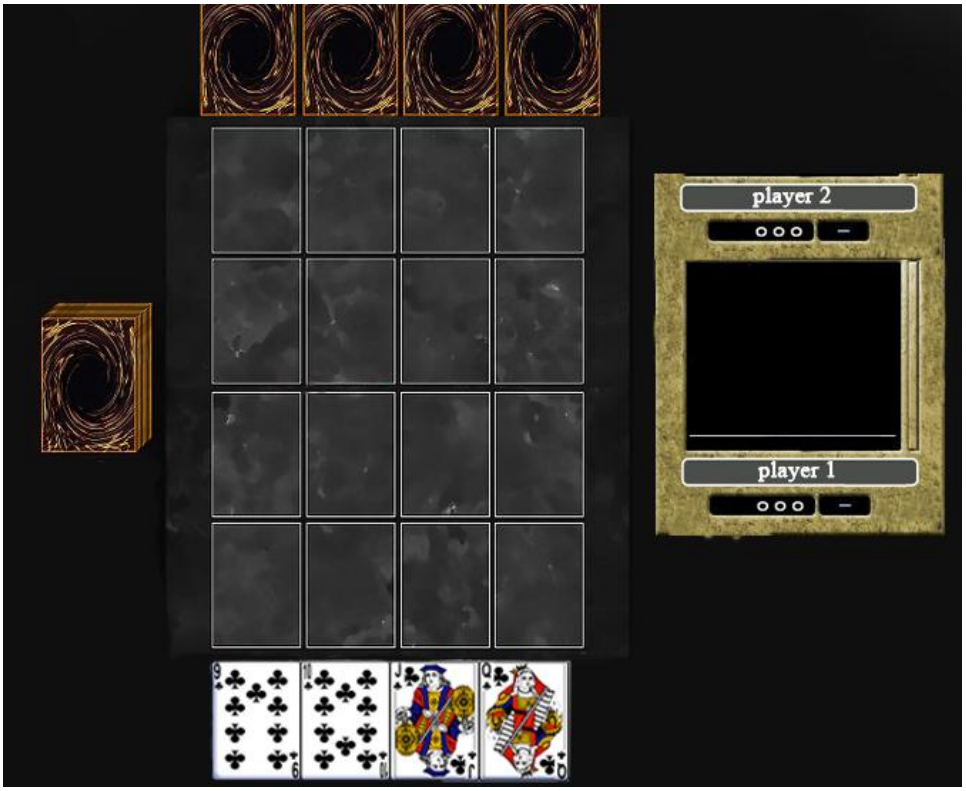

Figure 4.2.1 User interface

When a player places a card their card are flipped over a message is displayed in the dialog box prompting the user to change seats with the next player. When the next player sits in front they interact with the dialog box to reveal his/her cards and complete their turn. If there are other cards on the field the player's score will be updated. A state machine diagram of this process can be seen in figure 4.2.2

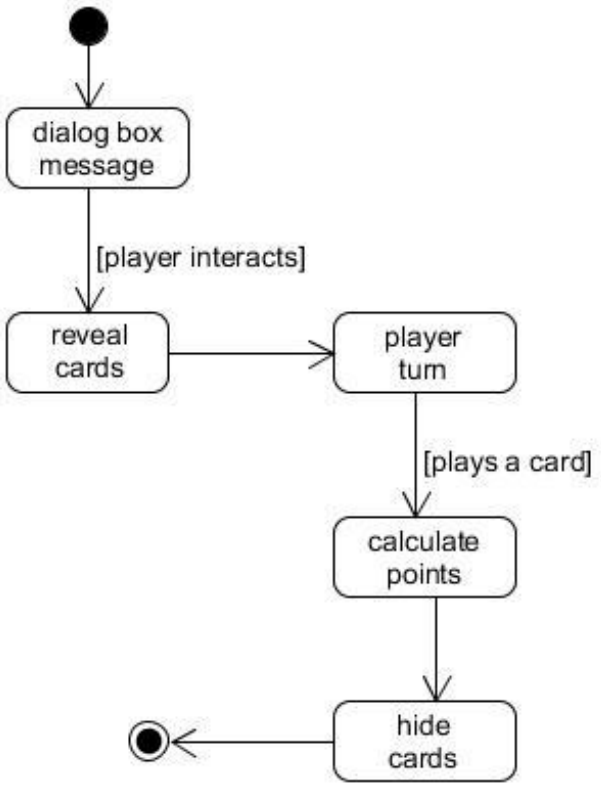

Figure 4.2.2 User interface interaction

Chapter 5: Testing

5.1 Verification with respect functional requirements

The functional requirements were as follows:

• Create the pack of card required for the game

• Shuffle the pack of cards

•Deal the appropriate amount of cards

• Allow a player to select and play a card
(adding it to the field and removing it from the hand)

• Calculate score

• Record a player move

•Check that the player move was legal

•Provide feedback if move was illegal

• Record each player's points

• Display current game state

•Announce a winner

5.1.1 Create the pack of card required for the game

For the creation of the pack the card constructor was called 32 times with different arguments to create the required cards. Tests were run to make sure that the card constructor will create the required cards and that will they behave in the way they have to. The pale of card was also tested as to if it has the needed cards with the required values both in rank and suit. As well as if it has the appropriate amount of cards in it

5.1.2 Shuffle the pack of cards

For the shuffling of the cards the native java method Collections.shuffle() was used The method was verified by printing the pack's content of the pile in the console before and after it was shuffled.

5.1.3 Deal the appropriate amount of cards

The dealing of cards was achieved by using the dealCardsToEachPlayer(int numOfCards), this method was verified by unit testing each individual method involved in it individually, as well as a whole.

5.1.4 Allow a player to select and play a card

This requirement was archived by combining several methods. The included methods are:

- garbo.getField().addCard(int r, int c, Card tempCard)
- garbo.getCurrentPlayer().getHand().removeCard(Card cardNum)
- garbo.getNotCurrentPlayer().addToScore(addToScore)

Each of the methods above was tested using a unit testing approach. Also, upon completion, the game was played through several times with the intention to find any errors or bugs

5.1.5 Calculate score

The calculation of a player's score is done by using two main methods

- calPoints(Garbo garbo, Card card, int r, int c)
- addToScore(int addToScore)

The 'aclPoints' method was tested by playing the game and paying close attention to how the player score rises The 'addToScore' method was unit tested.

5.1.6 Record a player move

This requirement was meat by storing any changing to the state of the game in objects that have the appropriate data types. For instance the state of the field i.e. when a card is played is stored in a field object what has a two dimensional array which can hold each played card and its place or storing each player score in a Player class. The methods used were unit tested or user tested

5.1.7 Check that the player move was legal and provide feedback if move was illegal

These requirements were met and user tested as when an illegal move is made the state of the does not change and a user prompt is printed in the console.

5.1.8 Record each player's points

Each player's points are stored in an object of class Player so the there is no confusion as to which points belong to which player. The recording of each player's score works fine and the method (addToScore(int score)) involved were user and unit tested

1.5.9 Display current game state and announce a winner

These criteria were also met and user tested as they print to the console

5.2 Validations with respect to the use cases

5.2.1 Use case: Start a Garbo game

Name identifier version: Start a game UC01 Version 1.0
Initiator: Garbo
Goal: Start a game
Assumptions: two players have joined the game they know how to play, they have decided how will take the first turn, the pack of 32 cards is available.
Main scenario:

1. A 'welcome message' is displayed
2. The game pack is shuffled
3. Eight cards from the game pack are dealt between the two players(4 each)
4. The field is displayed
5. Player1 is invited to take the first turn

All the actions introduced in this use case were implemented into the game. Slight changes were made to action five. In the beginning of the game the first turn is given to player1 and they are prompted to play the first card. Every method involved is the use case

5.2.2 Use case: play a card

Name identifier version: Play a card UC02 Version 1.0
Initiator: Garbo
Goal: To play one of the cards in the player's hand onto the field
Assumptions: it is the current player's turn and no player has yet won the game
Main scenario:

1. Player selects a card and where to place it
2. The card is added to the filed
3. The card is removed from player hand

Each method involved in the completion of the tasks involved in this use case was unit tested and upon completion of the project the game was played several times to

5.2.3 Use case: calculate suit points

Name identifier version: calculate suit points UC03 Version 1.0
Initiator: Garbo
Goal: to calculate points for a player
Assumptions: a game is in progress a player has just placed a card and checks are made after each player turn

Main scenario:

1. Get the suit of placed card
2. Check if card matches the suit of any adjacent card
3. Calculate how many points the suit matching is worth

Each method involved in the completion of the tasks involved in this use case was unit tested and upon completion of the project the game was played several times to

5.2.4 Use case: calculate rank points

Name identifier version: calculate rank points UC04 Version 1.0
Initiator: Garbo
Goal: to calculate points for a player
Assumptions: a game is in progress a player has just placed a card and checks are made after each player turn

Main scenario:

1. Get the rank of the placed card
2. Check if card matches the rank of any card in the same row , column and diagonally
3. Calculate how many points the rank matching is worth

Each method involved in the completion of the tasks involved in this use case was unit tested and upon completion of the project the game was played several times to

5.2.5 Use case: calculate card underneath points

Name identifier version: calculate card underneath points UC05 Version 1.0
Initiator: Garbo
Goal: to calculate points for a player
Assumptions: a game is in progress a player has just placed a card and checks are made after each player turn

Main scenario:

1. Check if there is a card underneath
2. Check if the two card match suits
3. Calculate how many points the suit matching is worth
4. Check if the two card match in rank
5. Calculate how many points the rank matching is worth

Each method involved in the completion of the tasks involved in this use case was unit tested and upon completion of the project the game was played several times to

5.2.6 Use case: calculate points

Name identifier version: calculate points UC06 Version 1.0
Initiator: Garbo
Goal: to calculate points for a player
Assumptions: a game is in progress a player has just placed a card and checks are made after each player turn

Main scenario:

1. Get the current player's points
2. Calculate suit points UC03
3. Calculate rank points UC04
4. Calculate card underneath points UC05
5. Calculate how many points should be added to the player's points
6. Add points to player's current points

Each method involved in the completion of the tasks involved in this use case was unit tested and upon completion of the project the game was played several times to

5.2.7 Use case: beginning of Level 2

Name identifier version: beginning of Level 2 UC07 Version 1.0
Initiator: Garbo
Goal: To end level 1 and start level 2
Assumptions: the game is in progress and checks are made after each player turn
Main scenario:

1. The playing field is full
2. The rest of the dealing pile is dealt out(the last 8 cards)
3. Players will now plays cards on top of existing cards

There is no method that checks if the field is full. Instead the software relies on the number of cards that have been played and the number of cards in each player's hand to determine when level 2 should begin.

At this point we know that there are only eight cards left in the dealing pile so we just deal four cards to each player

Each method involved in the completion of the tasks involved in this use case was unit tested and upon completion of the project the game was played several times to

5.2.8 Use case: hide/show hand

Name identifier version: hide/show hand UC08 Version 1.0
Initiator: Garbo
Goal: To prevent players from seeing each other's hands
Assumptions: The game is in progress, the hide/show cards is performed after each player turn, the game is played on a laptop
Main scenario:

1. Current player's turn begins
2. Current player interacts with the game to display hand
3. Current player plays a card
4. Current player's hand is hidden
5. Next player turn

Each method involved in the completion of the tasks involved in this use case was unit tested and upon completion of the project the game was played several times to

5.2.9 Use case: Announce winner time

Name identifier version: Announce winner time UC09 Version 1.0
Initiator: Garbo
Goal: To determine when a winner needs to be announced
Assumptions: The game is in progress and checks are made after each player turn
Main scenario:

1. A check is made with regard to the number of cards remaining in the dealing pail
2. A check is made with regard to the number of cards remaining in the hand of player1
3. A check is made with regard to the number of cards remaining in the hand of player 2
4. If the number of cards remaining in the dealing pack, player1's hand and player2's hand is zero. It is time to check who won the game

Each method involved in the completion of the tasks involved in this use case was unit tested and upon completion of the project the game was played several times to

5.2.10 Use case: Who won

Name identifier version: Who won UC10 Version 1.0
Initiator: Garbo
Goal: To identify a winner
Assumptions: The game is in progress and will end after the winner is announced
Main scenario:

1. The score of player 1 is retrieved
2. The score of player 2 is retrieved
3. The two scores are compared
4. The player with more points is announced as the winner
5. Game over

Each method involved in the completion of the tasks involved in this use case was unit tested and upon completion of the project the game was played several times to