**COS Senior Project I**

**Title**

**Stochastic Card Game**

**Author, ID#**

**IRDI BALLA, 100080452**

**Author:**   **, Date:**

*signature*

**Supervisor:**   **, Date:**

*signature*

**Blagoevgrad, 2014**

**Title**: Stochastic Card Game

**Author**: Irdi Balla

**Abstract:**

This app is a simple computer game adaption of one of my favorite childhood card games, Xing. The purpose of the game is to give the player a chance to try his abilities against a computer that has different levels of difficulty. The game can be played by anyone who is looking to have some fun or just trying to spend some time. On the other hand experienced player who think they are very good at this game can try to play against the hard levels where they can be assured that it will not be easy at all.

**Declaration of authorship:**

“The Senior Project presented here is the work of the author solely, without any external help, under the supervision of Prof. Svetla Boytcheva. All sources, used in development, are cited in the text and in the Reference section.”

Author:

Table of Contents

**1) Introduction4**

1.1) Introduction ………………………………………………………………………………………………………………….……………..4 1.2) How to play4

1.3) Technologies6

**2) Specification and Analysis of the Software Requirements8**

2.1) Requirements8

2.2) Use Cases9

**3) Design of the Software Solution**16

3.1) UML Diagram16

3.2) Algorithms21

**4) Implementation**27

4.1) Technologies used27

4.2) Installation Requirements27

4.3) Code Snippets27

4.4) Security29

**5) Results and Conclusion**30

5.1) Functional Requirements not Implemented30

5.2) Test Version31

5.3) Bugs31

5.4) Statistics32

5.5) Conclusions33

5.6) Personal Gains33

5.6) The Game in Action33

**6) References**34

6.1) Books39

6.2) Web Pages39

**Part 1 – Introduction**

* 1. INTRODUCTION

This app is a simple computer game adaption of one of my favorite childhood card games, Xing. The purpose of the game is to give the player a chance to try his abilities against a computer that has different levels of difficulty. The game can be played by anyone who is looking to have some fun or just trying to spend some time. On the other hand experienced player who think they are very good at this game can try to play against the hard levels where they can be assured that it will not be easy at all.

1.1.1 Stochastic

## Definition of STOCHASTIC

1:  [random](http://www.merriam-webster.com/dictionary/random); specifically :  involving a random variable <astochastic process>

2:  involving chance or probability :  [probabilistic](http://www.merriam-webster.com/dictionary/probabilistic) <astochastic model of radiation-induced mutation> [Webster]

The definitions where taken from the Merriam Webster dictionary online. The stochastic nature of my game is in its usage of probability and chance. Cards are distributed randomly so this game after all is another game of luck

* 1. HOW TO PLAY
     1. POINTS

This is a kid’s game that everyone can play. The rules of the game are simple and if you are old enough to hold a card in your hand then you are old enough to play.

Xing (spelled tzi:ng) can be played by 2, 3 or 4 people. The aim of the game is to gather as many points as possible at the end of the game. There is a total of five (5) points that can be awarded to the players. Points are given if you complete specific tasks.

2 points – for the player with the most cards. In case of a draw the points are distributed equally between the players (i.e. if two players have 26 cards each then they both get 1 point),

1 point – for the player that has collected the most Clubs. In case of a draw the points are distributed equally between the players (i.e. if two players have 6 Club cards each then they both get 0.5 points). This rule applies only if 3 or 4 players are playing.

1 point – is awarded to the player that collects “the beautiful 10”. The “beautiful 10” is the 10 of Diamonds. The player that has it at the end of the game gets 1 point.

1 point – for the player that collects “the beautiful 2”. The “beautiful 2” is the 2 of Clubs. Note that the 2 of Clubs counts as a club as well.

The points are counted at the end of the game. The player with the most points is considered to be a winner. This game can also be played as a tournament and the one who reaches 21 points first is declared a winner. In case of a draw the game is extended to 31, 41 and so on until a player is victorious.

* + 1. GAME FLOW

When the game starts four cards from the bottom of the deck are placed face-up on the field one over the other. The game is composed of rounds and in each of the rounds, the players get four cards each. The players follow the same routine when they throw cards meaning that the one that throws first the first round will throw first the other rounds as well. When it is your turn you have to throw one of the cards in your hands. If the card you throw matches the suit of the card that is on top of the pile in the field than you take all the cards and you put them aside. If you can’t match the suit of the card than you throw any card you have and then your turn is over. The other player now has to match the card you threw and so on. There is also a special card. The J (Jack) when thrown matches any card on top of the pile and you automatically take all the cards that are on the field. When all players have used all for cards in the round then the round ends and another one starts. Each player gets four new cards and they start their turns all over again. The game is finished when all the cards are used. If at the end of the game there are some cards on the ground the players that has matched a card last takes all the remaining ones. At this point the game is considered over and all players count their points.

1.3 – TECHNOLOGIES

1.3.1 HARDWARE

For this project all the hardware I need is inside my personal lap-top. I will not be using any other hardware except for that.

1.3.2 SOFTWARE

I will develop this program in c# using visual studio 2013 and its forms and UI controls.

The Use case diagrams will be created using GenMyModel.com

1.3.3 ALGORITHMS

The game has three levels of difficulty, low, medium and high. On each level the AI will behave in a different way and thus the code for all levels will be different.

The low difficulty level will be very simple. The AI will just throw cards without analyzing them. This level will behave like a little kid who has just learned the game but still has no idea what he is doing.

The medium level would be the level of an everyday player, who has some experience in the game. This level differs from the previous one in that that this level will prioritize. By prioritization I mean that the computer will follow a certain path when deciding what card to throw. The prioritization for this level will be as follows: (top highest priority)

1. Beautiful exacts
2. Other exacts
3. Jacks
4. Other cards

The most interesting level is of curse the hard one. The AI in this case will play like a smart computer. Every card thrown throughout the game will be remembered and the AI will calculate the safest card to play at every specific hand. The memory will hold an index for every face and the index will be the number of cards with that face thrown so far plus the number of cards in the bots hand with that face. So the computer will throw the one that has the highest index which will result in the card that has the lowest possibility to be matched by the next player. This level of prioritization will be the very last after the computer has no chance of matching the card on the field momentarily. The prioritization for this level will be as follows:

1. Beautiful exacts
2. Spades exacts
3. Other exacts
4. Jack
5. Card with highest index

**Part 2 - Specification and Analysis of the Software Requirements**

2.1 REQUIREMENTS

2.1.1 FUNCTIONAL REQUIREMENTS

1. The game should have an interface where the player can interact

2. The player should be able to choose the level of difficulty

3. The player should be able to choose against how many AI they want to play (1, 2 or 3 AI)

4. The computer should calculate the safest card to play for the AI

5. The computer should hold data for the cards that are thrown in the hard level

6. The game should display the card that is the last on the field in the middle of the form

7. The game should also display the last card that is thrown by any of the players on the right of the field card

8. After the computer throws a card the game should pause for a second so that the player can see what card was thrown

9. The game should create a collection for each of the players so that the cards they have collected can be saved and used later

10. The game should count the points automatically at the end of the game

11. The game should display the points at the end of the game

12. The human player should be able to throw cards just by clicking them

13. The player should be able to exit the game whenever he/she wants

14. The game should create a deck of cards

15. The game should distribute the cards

*16. The player should be able to view statistics of the previous games he/she has played*

*17. The player should be able to play in tournament mode*

*18. The game should add up the points the players get each game.*

*19. The game should end the game once a player reaches 21 points.*

*20. Some sound should be played when a card is matched*

2.1.2 NON-FUNCTIONAL REQUIREMENTS

1. The game should run on windows 7 and later versions.

2. The game should be in English.

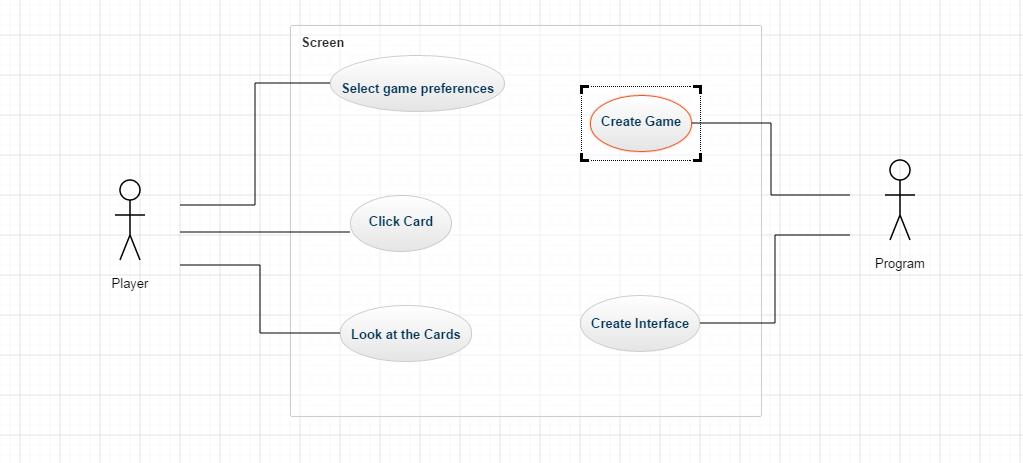
3. The computer should use less than 2 sec to make a move. 1 second should be reserved for point 8 in the functional requirements so that leaves 1 second for the computer to decide on throwing a card

4. The software should have a fail ratio of 1:99, meaning that it can fail at most once every 100 games

*5. A video tutorial should be saved under a tutorial section in the first window*

2.2 USE CASES

2.2.1 MAIN USE CASE

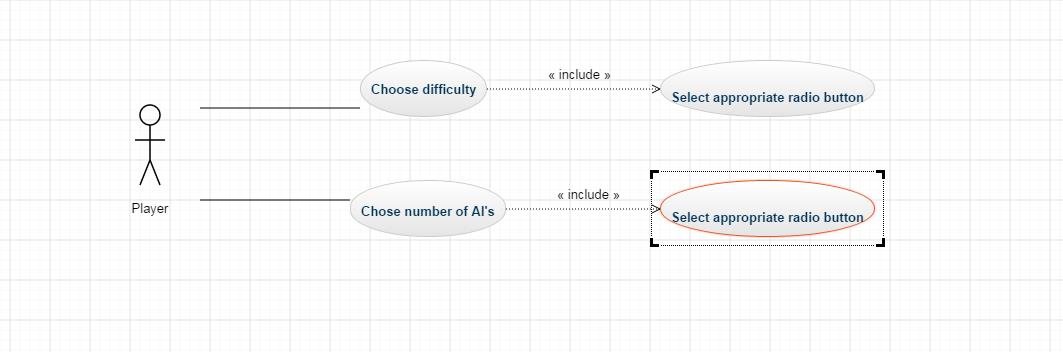


*This figure shows the main Use Case diagram*

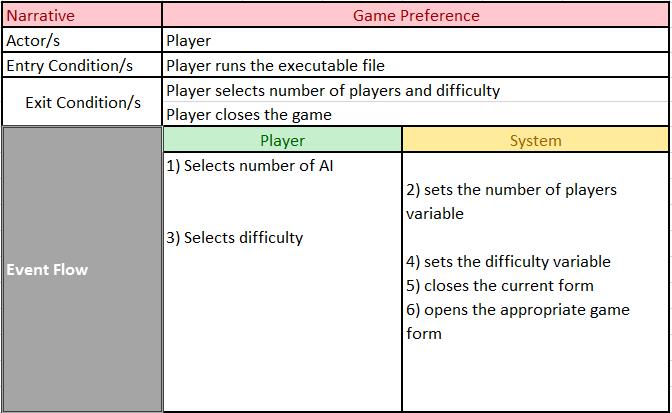
The diagram shown in the figure above describes the main functionalities of the game. It consists of 2 actors (the human player and the game) which play a role in the changing of the display of the screen. This main diagram can be broken down into several smaller ones which I will describe below.

2.2.2 GAME PREFERENCES

The functionalities included in this diagram are involved in selecting the game parameters. The Player can choose the number of bots he/she wants to play against and the difficulty they should offer.



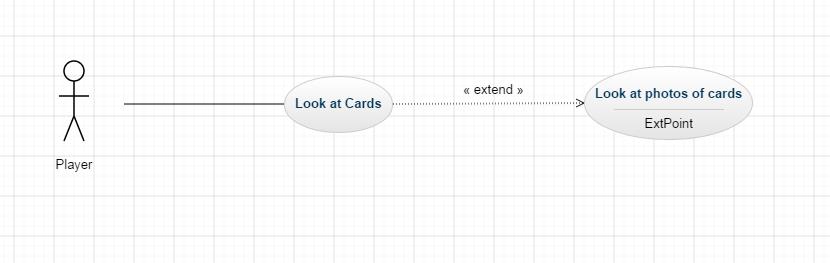
*This figure shows the Choose Preferences functionality*



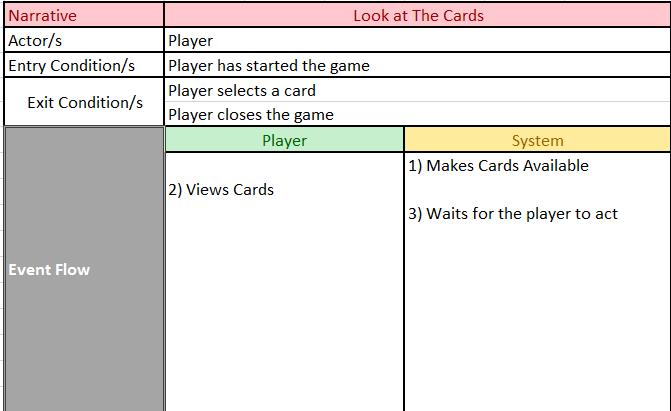
*This figure shows the narrative for the Game Preference case*

2.2.3 LOOK AT THE CARDS

This diagram is concerned with the player view the cards. The system on its side makes sure the cards are there so that the player can see them.



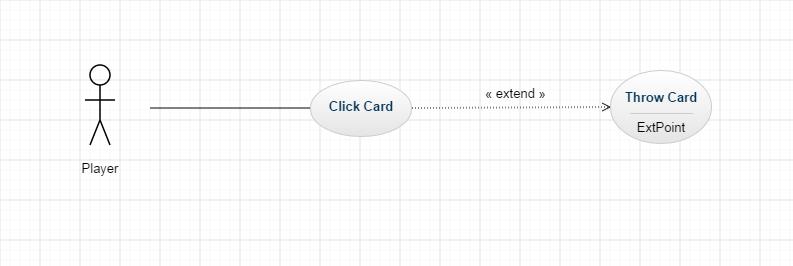
*This figure shows the Look at The Cards functionality*



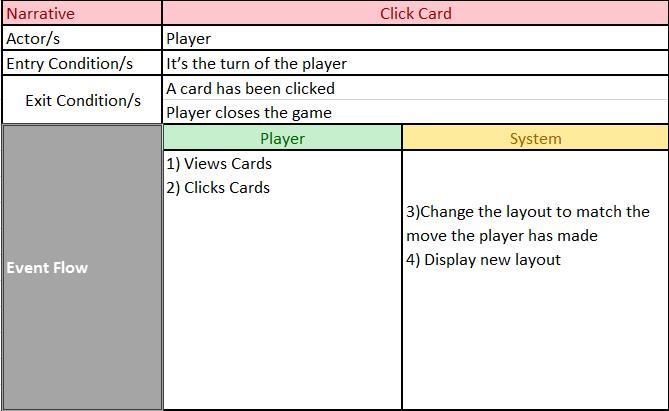
*This figure shows the narrative for the Look at the Cards case*

2.2.4 CLICK CARD

This functionality allows to the player to interact with the game throw the use of the picture boxes. By clicking one of the pictures of the cards in his hand he notifies the computer that that is the card he wants to throw.



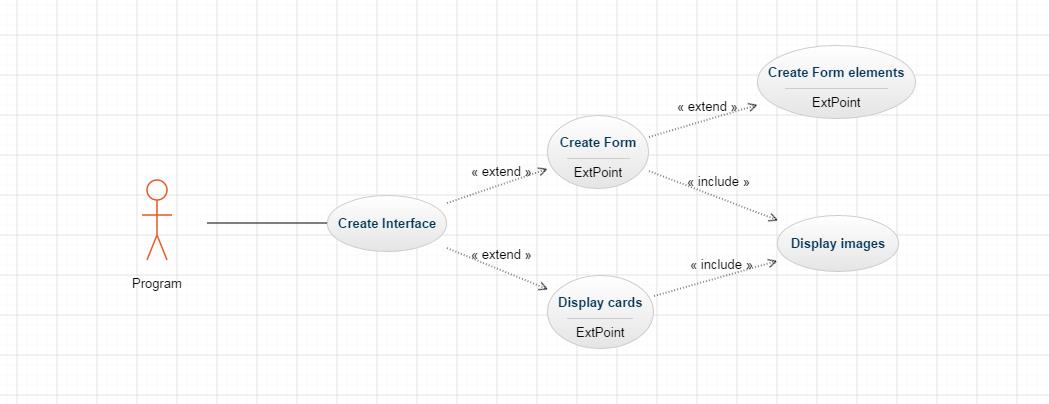
*This figure shows the Click Card functionality*



*This figure shows the narrative for the Click Card case*

2.2.5 Create Interface

This functionality involves the creation of an interface with all its elements, including picture boxes, images, and radio buttons.



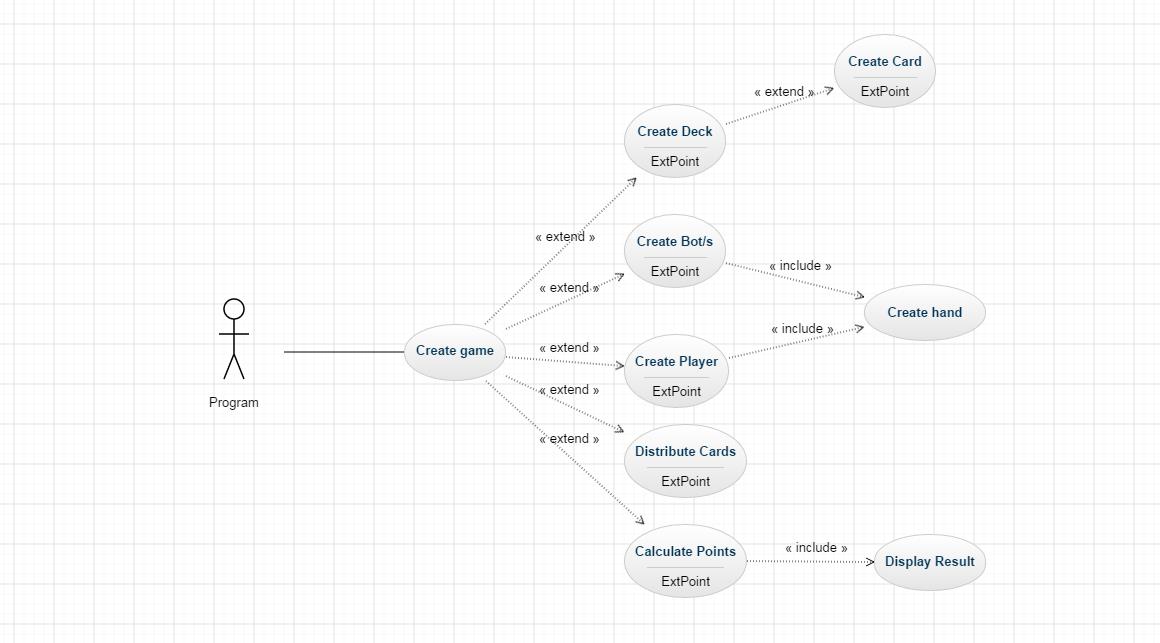
*This figure shows the Create Interface functionality*



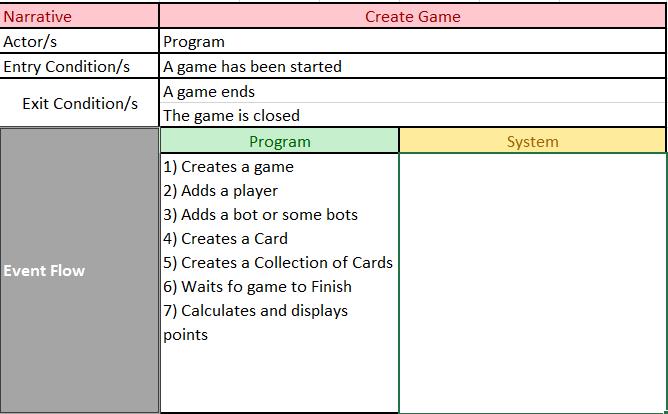
*This figure shows the narrative for the Create Interface case*

2.2.6 CREATE GAME

This is the most interesting and complicated Use case derived from the main Use case diagram. The program creates the game by creating all the virtual representations of the objects, such as player, bot/s, deck, card, hand, and it also takes care of jobs like calculating points and displaying the result.



*This figure shows the Create Game functionality*



*This figure shows the narrative for the Create Game case*

**3. Design of the software solution**

**3.1 UML DIAGRAM**



*This figure shows the main UML Diagram with only the dependencies shown*

The diagram above shows a representation of a layered system which describes how the game works in general.

In the top layer it’s the Program class which calls the form that will be first displayed when the game is run.

The second level consists of the first form that is open where the player has to select difficulty and nr of bots. This layer includes the resources and the settings as of the application as well

The next layer is composed of three parts, the different GameForms. These are the forms that will be displayed after the user leaves the NewGameForm. Depending on the number of bots he decided to play against the appropriate form will be displayed. Two of these forms cannot be called at the same time by the program.

Based on what form was loaded a Game instance will be created (either Game, Game2 or Game3). The game Class holds the most important methods for the game. This class is the engine that makes the application run. It creates instances of all the other classes in the level below (except Hand).

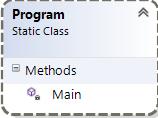
The fifth level contains the instances needed from the Game class to do its work. It contains classes such as the Player class, the Deck class and the AI class which is a base class for the AIEasy, AIMedium and AIHard classes.

The lowest level is the Hand and the Card class. These classes hold the implementation of instances that hold a single card and a hand. These classes are very important to the ones on top as they are the basis for the game (especially the Card class).

Now I will analyze each layer of the diagram and show the connections between them.

3.1.1 Top Layer

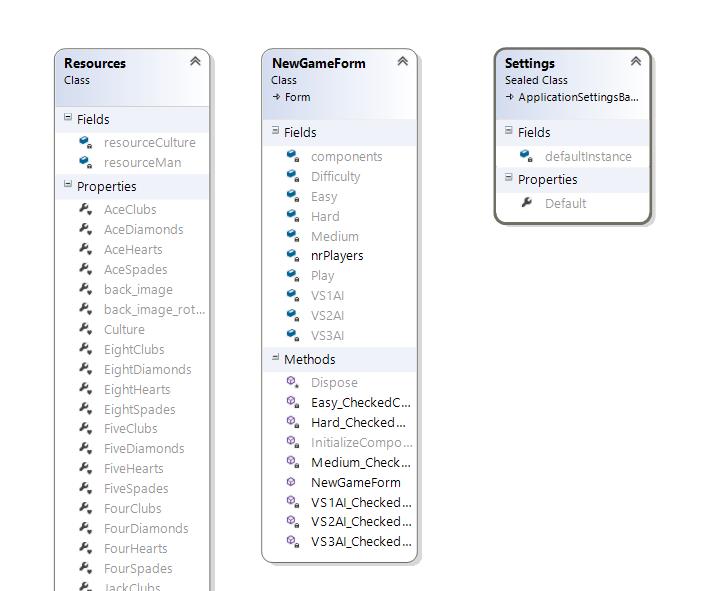
The top layer is generated by Visual studio when a new project is created. It contains a single main method which specifies the entry point of the program, in my case the NewGameForm form.



*This figure shows the Program class*

3.1.2 Second Level

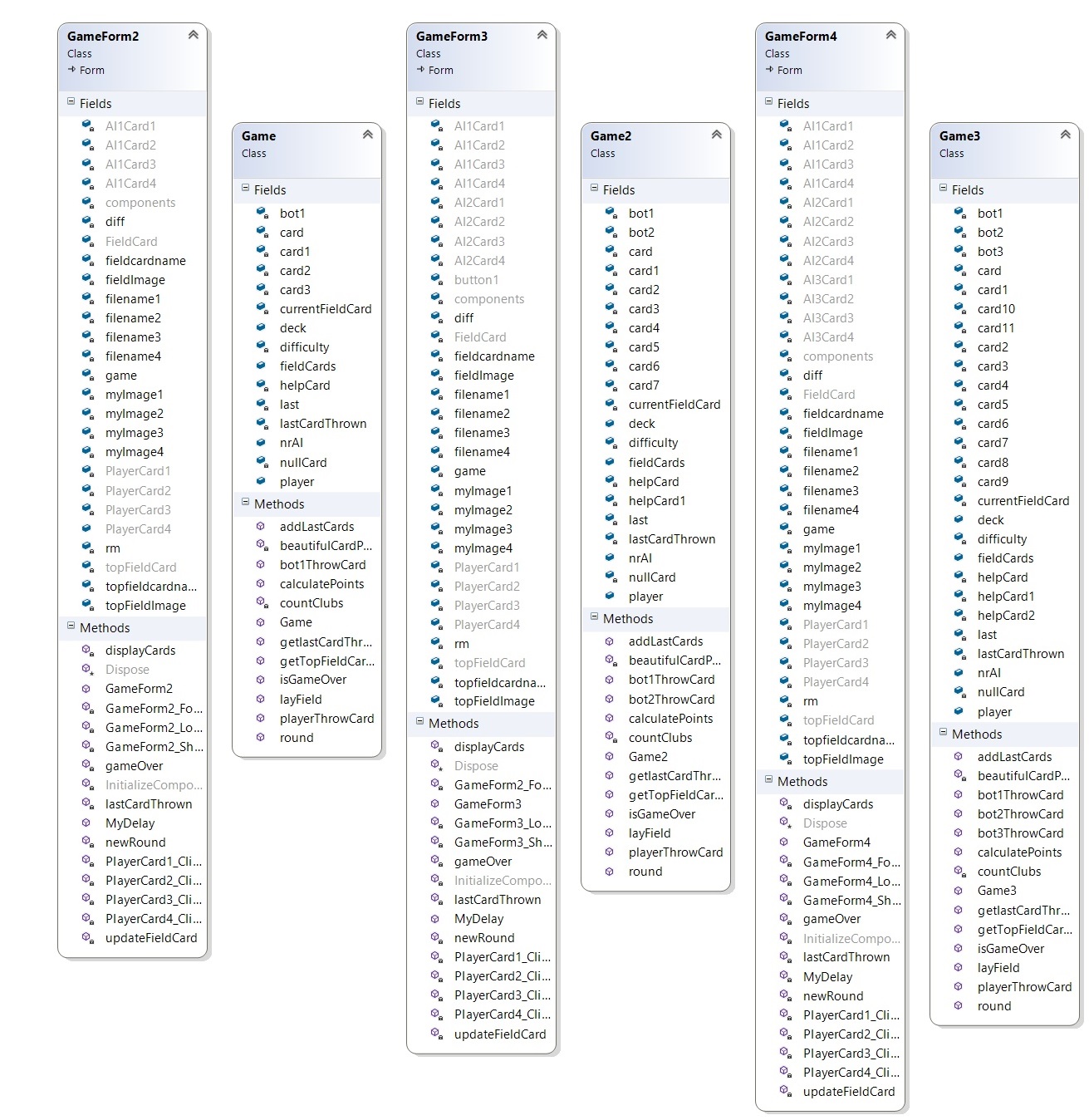
This layer contains the Resources class which contains the resources that are needed for the game such as all the images used. It is way longer that displayed here as I have an image for every one of the 52 cards in the deck. It also hold the background image, that table image and the image for the back of the card. The NewGameForm is the form opened when you start the program. It contains two group boxes each with three radio buttons and some methods to respond to different events. The Settings class was created by Visual Studio at the start of the project and it holds different data about the settings of the program itself.



This figure shows the *second* layer of the main UML Diagram

3.1.3 Third and Fourth Layer

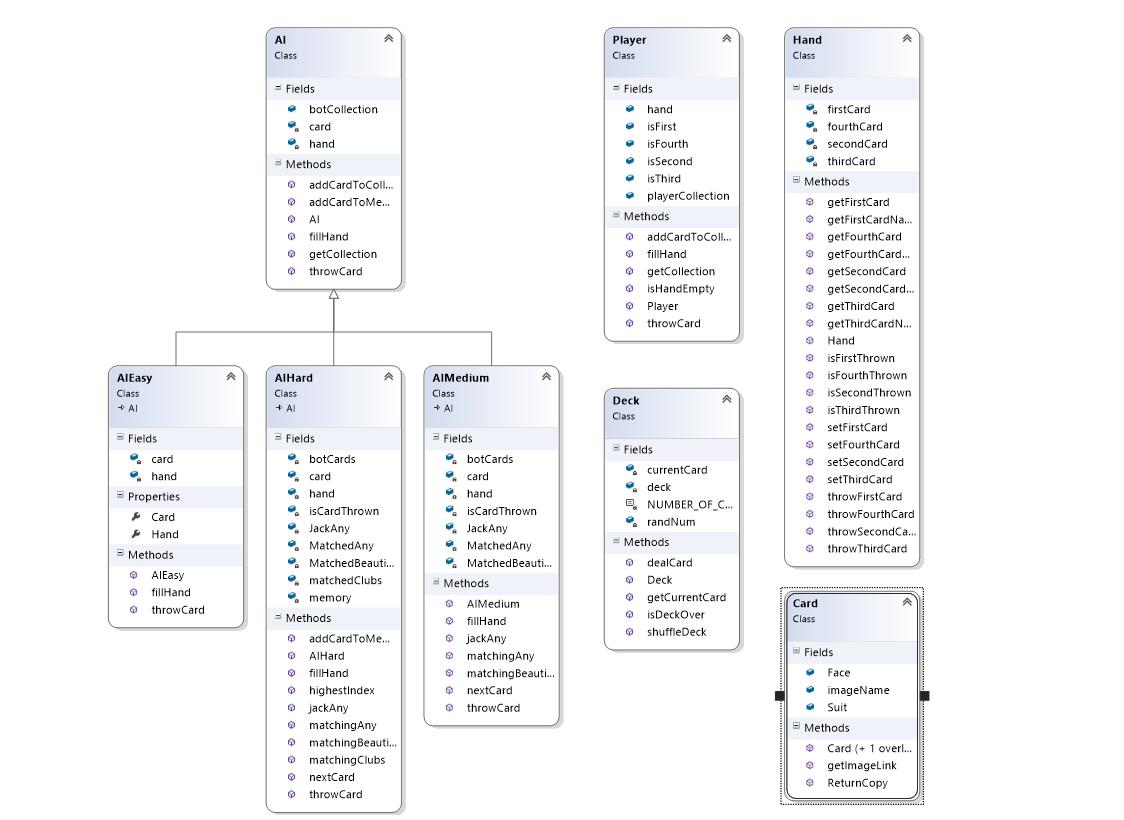
These two layer are very connected to each other. In case GameForm2 was opened then an instance of Game will be created, if GameForm3 was the one to be loaded then an instance of Game2 will be created and so on. The GameForms in general hold the layout of the Form for different number of players selected, GameForm2 when you play against 1 AI, GameForm3 when you play against 2 AI etc. The Game Classes on the other side hold the logic behind the forms. How the game works out.



This figure shows the third and fourth layer of the main UML Diagram

3.1.4 Fifth and Sixth Levels

These two levels define the way the different objects used in the Game class act. The AI class contains definition for the bots. These class contains several virtual methods which are overridden by the three derived classes AIEasy, AIMedium and AIHard. The most important method in these classes is the throwCard() method which is the algorithm that the computer obeys to throw a card. The Player class hold the methods needed for the human interaction with the game. The Deck card creates and maintains the usage of the Deck in game. An instance of the Hand class is created in both AI and Player classes and it contains four cards which represent the cards the players have in their hand. The Card class is used by all other class and it is a very simple class that simply creates a card with a certain Face and Suit.

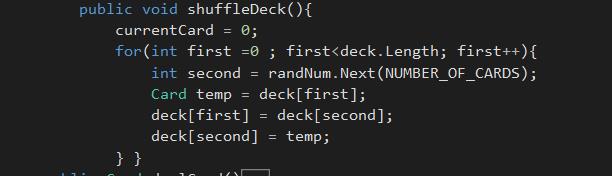


This figure shows the fifth and sixth layer of the main UML Diagram

3.2 Algorithms

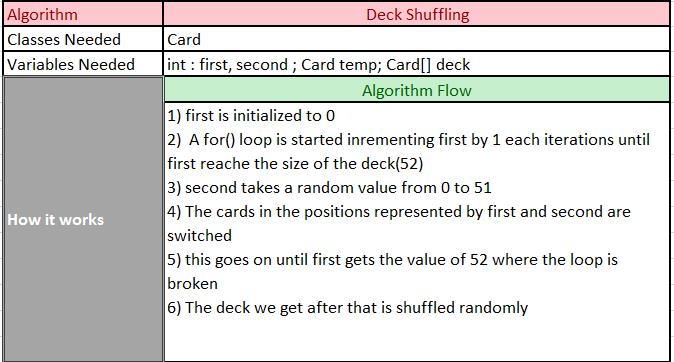
All the algorithms used in this game are my own creation. They are written in c# programming language using only the libraries provided by the Visual Studio Professional 2013 IDE. I will list and describe the functionality of the most important algorithms below.

3.2.1Deck Shuffling Algorithm



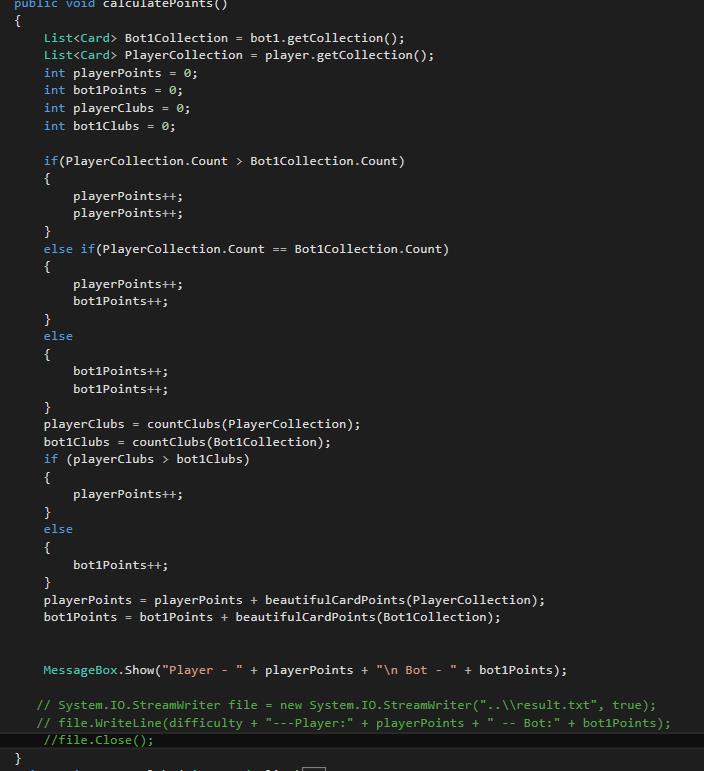
*This figure shows the code for the Deck Shuffling algorithm*

For this algorithm to be performed I first need a Card class which hold the data and creates the Card. The algorithm can be found on the ShuffleDeck() method in the Deck class



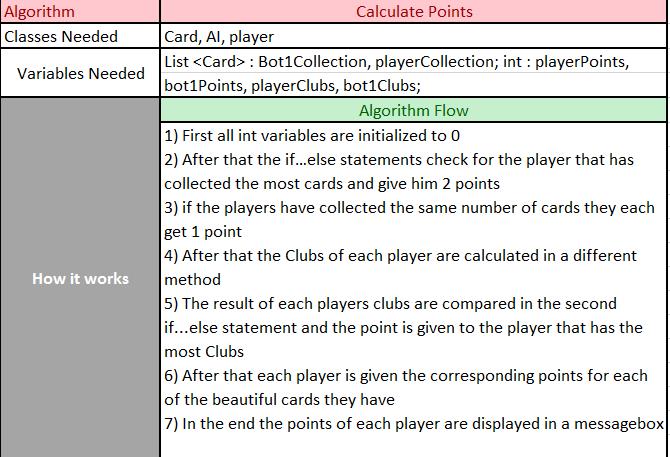
*This figure shows the way the Deck Shuffling algorithm works*

**3.2.2 Calculate Points**



*This figure shows the code for the Calculate Points algorithm*

For this example I will take the game of 1 Human vs 1 Bot. This algorithm is in the calculatePoints() method in the Game class. It makes use of other classes such as AI, Player and Card. It also displays a textbox with the result in the end. The last commented lines are used for a testing version which will be explained later.



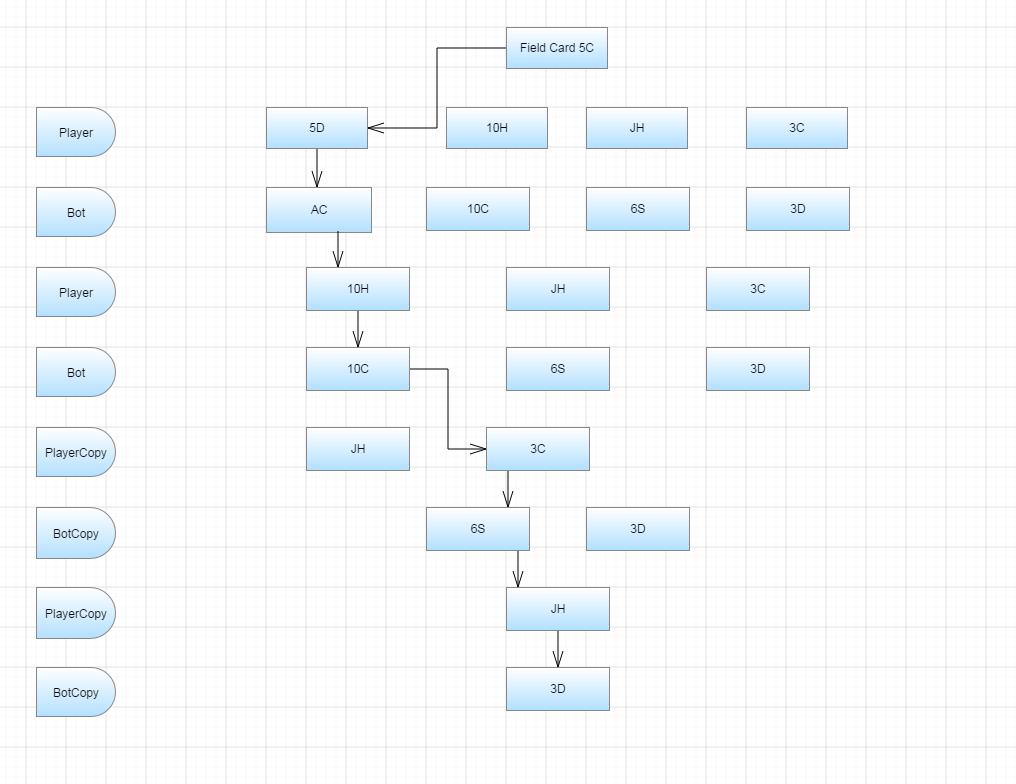
*This figure shows the way the Calculate Points algorithm works*

**3.2.3 Throw Card Decision**

In this section I will describe the logic that stands behind the card throwing for the bot. The three difficulty levels implement a different card throwing method which is inherited from the base AI class.

**3.2.3.1 Easy Level**

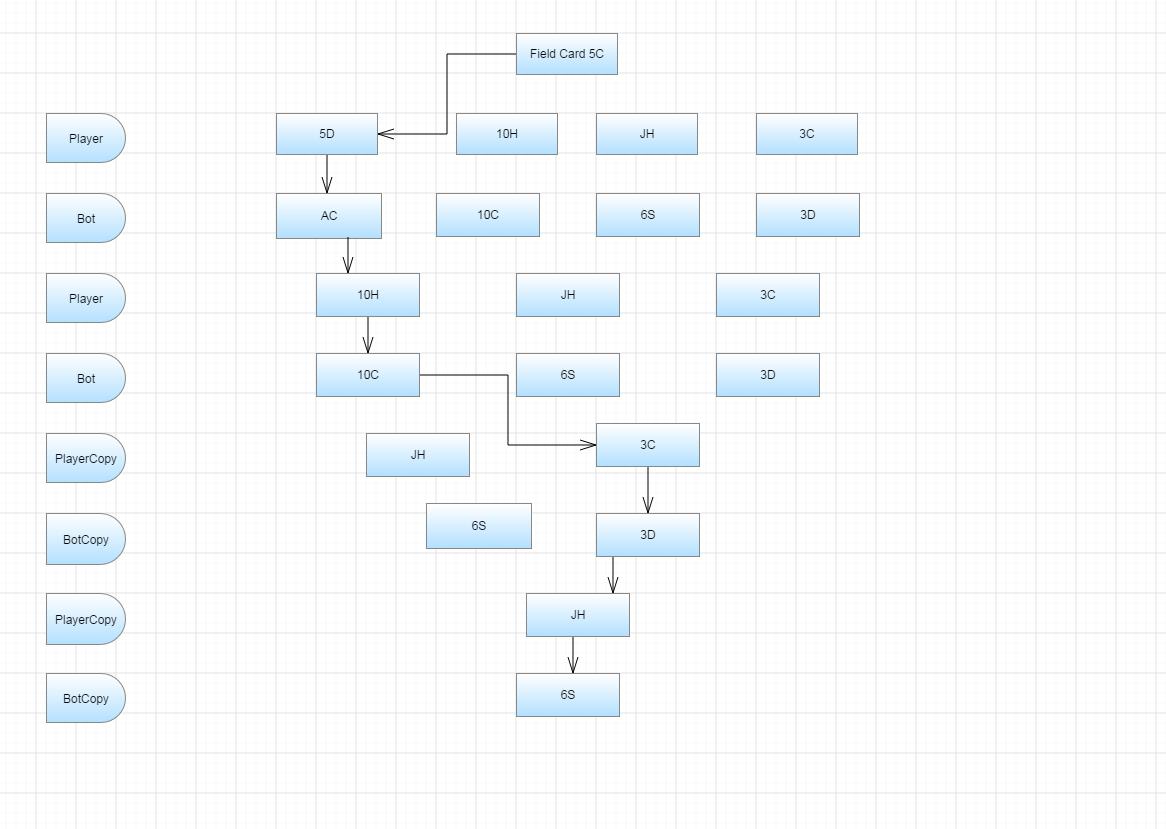
The easy level has a very simple card throwing technique. It will first check whether the first card is thrown. If not it will throw it. If the card is thrown then it will check for the second card and so on. This decision making represents that of a 5-year old kid who was just taught the game. He can’t yet decide what the best card to be thrown is and he will throw the card he likes the most even if he has a matching card in his hand. An example of such a round is shown in the figure below.



This figure describes how a round between 1 Human and 1 easy-level Bot can go

**3.2.3.2 Medium Level**

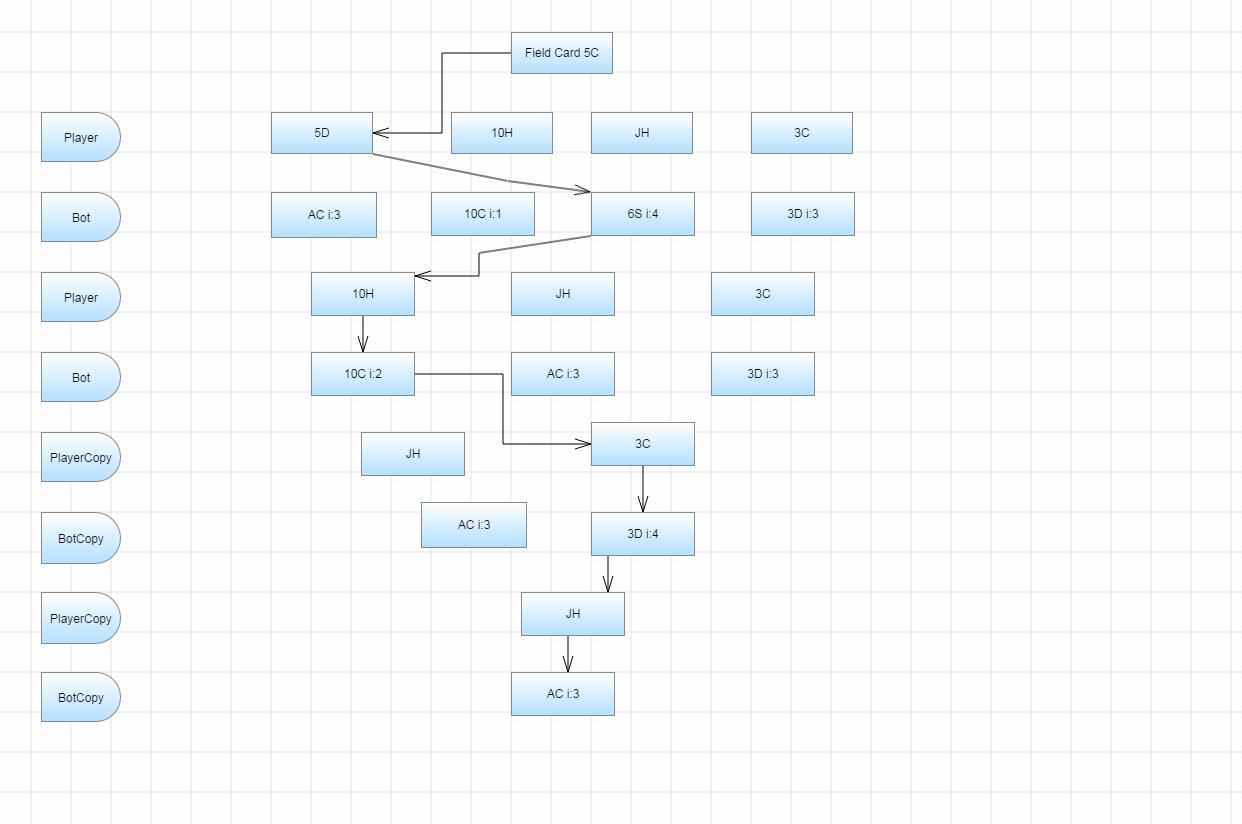
As described earlier the medium level has some priorities. He tries to match what there is on the ground so that he can get points. This behavior matches that of a normal player who knows what the purpose of the game is. The same round shown above for the easy level is now adapted for the medium level below. The “mistake” the computer made in round 3 where he didn’t match the 3 is now fixed by the medium level.



This figure describes how a round between 1 Human and 1 medium-level Bot can go

**3.2.3.3 Hard Level**

The most important and difficult level to implement is the hard level. In the medium level when the computer cannot match the card on the field it will throw the first card that is not thrown in his hand. This level improves that by throwing the card based on an index. There is an index for each card face and it is equal to the number of copies of the same face throw. If there have been thrown 3 eights so far and the computer has the fourth one in his hand, unless he can match the card that is on the field he will throw the eight so that the opponent (the human player) has the lowest chance to match it. The computer reserves a place in its memory for the collection of the thrown cards and the program uses this collection when deciding what card to throw. The same round is played against the hard level. Next to the bot cards the index of that face is shown. In this case instead of throwing the Ace of Clubs the computer throws the Six of Spades which has a higher index, thus the player is less likely to match it (in this case the probability for the player to match it is 0). This algorithm is an adapted version of the famous alpha-beta pruning.



This figure describes how a round between 1 Human and 1 hard-level Bot can go

**4. Implementation**

**4.1 TECHNOLOGIES USED**

The game was created in c# programing language using the Microsoft Visual Studio Profession 2013 IDE. No external libraries were used except for the ones offered by this software. The video tutorial was created using Windows Live Movie Maker. I haven’t used any platform for my game. I have only created some derived classes from the form class which is already defined.

**4.2 INSTALLATION REQUIREMENTS**

In order to play the game you need to run the SeniorProject.exe file in a Windows Operating System.

The game will run on all Windows Operating systems.

The main forms are not very big {(W, H), (820, 550) and (750, 800)} so they can fit most of the screen resolutions that exist today.

**4.3 CODE SNIPPETS**

4.3.1 Card clicking

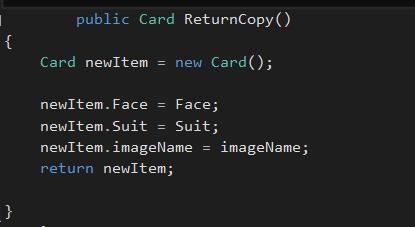
Below there is a picture of the code inside a method that is called when the player clicks his left-most card. This method at first throws the card and makes appropriate changes in the layout, such as hiding the picture box of the card selected, changing both images of the field card to the new one. After that the form is refreshed and a delay of 1 second is called. This delay will make sure that the user can see the card that was thrown by the computer especially when there are 3 or more players



Code snippet for the method that control the clicking of the first card

**4.3.2 Return Copy**

Whenever I want to get a specific card, for example when I throw a card or the computer throws I never the card but instead I pass a copy of that card. This method is in the card class and it will make sure that my cards don’t mix with each other by returning a copy of the card and not the real one.



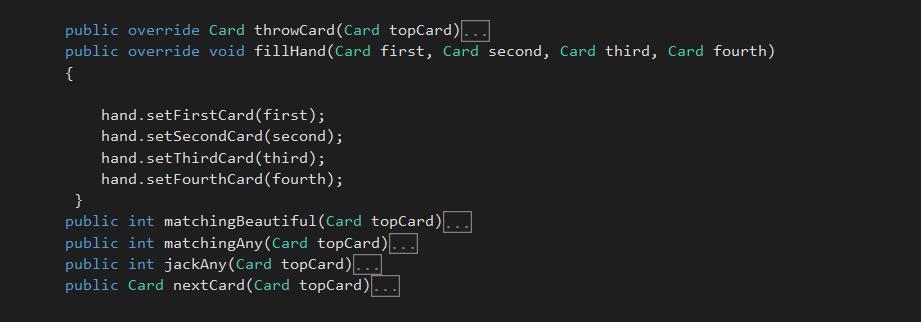
*Code snippet for the method that returns a copy of the card*

**4.3.3 AIMedium METHODS**

The AIMedium class contains 6 methods 2 of which override a base method in the AI class.

ThrowCard() is the most important one. In its implementation it makes calls to all other methods except for fillHand() which just sets the cards of the hand. Of the bot. ThrowCard() returns a card that the computer thinks is the best option to throw. matchingBeautiful() will return a number from 0 to 3 if it finds a beautiful card in the hand that can match the field card. matchingAny() and jackAny() will return same results the first one if it can match the card on the field and the second one if the bot has a jack and there is at least 1 card on the field.

nextCard will return the first card it checks that is not thrown. So first it will try to return the first card, if that is thrown then the second card and so on.



*Code snippet for the methods of the AIMedium class*

4.4 SECURITY

The security level that this game requires is minimal. It uses no database, neither it is connected to the internet. Unless the user clicks a card nothing will happen. A card already thrown can’t be clicked again. In the form at the beginning one of the groupboxes has already a selected value while the other is needed in order to play the game. Unless u select a difficulty you can’t play.

5. Results and Conclusion

**5.1 FUNCTIONAL REQUIREMENTS NOT IMPLEMENTED**

The functional requirements from point 2.1.1 that are not implemented yet are the ones that follow.

*16. The player should be able to view statistics of the previous games he/she has played*

*17. The player should be able to play in tournament mode*

*18. The game should add up the points the players get each game.*

*19. The game should end the game once a player reaches 21 points.*

*20. Some sound should be played when a card is matched*

I have divided them in two groups and plan to finish them in the next two iterations.

The first iteration or the extras feature will contain:

*16. The player should be able to view statistics of the previous games he/she has played*

*20. Some sound should be played when a card is matched*

The implementation of these features is not that hard and I started the first one but I have not included that in this report. The testing version which I will explain later is the first step towards the statistics tab.

The second round will consist of the following requirements to implement:

*17. The player should be able to play in tournament mode*

*18. The game should add up the points the players get each game.*

*19. The game should end the game once a player reaches 21 points.*

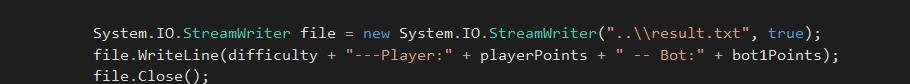
The tournament mode will be another game mode similar that will consist of a list of instances from the Game class.

I predict that the implementation of these two features will not take more than two weeks and that by the end of the year 2014 the full version will be available.

When the iteration that I mentioned are completed I will study the probability of adding a multiplayer online version of the game so that everyone can play and have fun with their friends.

**5.2 TEST VERSION**

The test version of the game is a special version that I created and gave to some of my friends so that they can play and test the game. I believe that the more testers there are even if unexperienced the better it is. I helped them install the game in their computer and asked them to play as many games as possible using as many combinations as possible of difficulty and player number as possible. I also asked them to take a screenshot if something unexpected would happen. The test version differs from the other one in that it has some lines of code added which open a file, write to the file and then close the file.



*Code snippet of the extra code in the test version*

**5.3 BUGS**

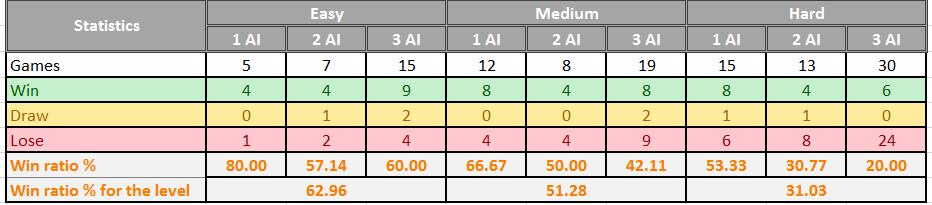
To my surprise no tester reported any bug during their gameplay. I thought they might have just been lazy to report something so I asked them one by one to tell even the smallest problem the have had. Some of the complained about the “flashing of the screen” when a card is thrown. That is caused by the Refresh() method of the form. I am currently trying to find an alternative way for this method.

Overall I was very happy as one of the requirements in the beginning was that the game should crash more than once every 100 games. At the end I had the result of more than 100 games and no reports of any crash.

**5.4 STATISTICS**

**5.4.1 SAMPLE**

The sample for the following statistics I will display is group of 124 games played in different difficulties and different number of computer players. These games were played by me and 6 other game testers who had the test version of the game. I considered all the players of a medium level as they all knew the rules of the game and had played before. Except for me no one new how the computer reacted. So they could not have a strategy on how to win.



*Table with the results of all the games*

**5.4.2 EASY**

27 easy games were played in total. 5 games were against a single AI, 7 against 2 AI and 15 against 3 AI. The player winning ratios from these games were respectively 80%, 57% and 60%. The total winning ratio for this level was about 63%.

**5.4.3 MEDIUM**

39 medium games were played in total. 12 games were against a single AI, 8 against 2 AI and 8 against 3 AI. The player winning ratios from these games were respectively 67%, 50% and 42%. The total winning ratio for this level was about 51%. If the medium level would represent a normal player this result is very pleasing.

**5.4.4 HARD**

58 hard games were played in total. 15 games were against a single AI, 13 against 2 AI and 30 against 3 AI. The player winning ratios from these games were respectively 53%, 30% and 20%. The total winning ratio for this level was about 31%. The hard level offers a very difficult game and winning in this level is pretty hard.

**5.5 Conclusions**

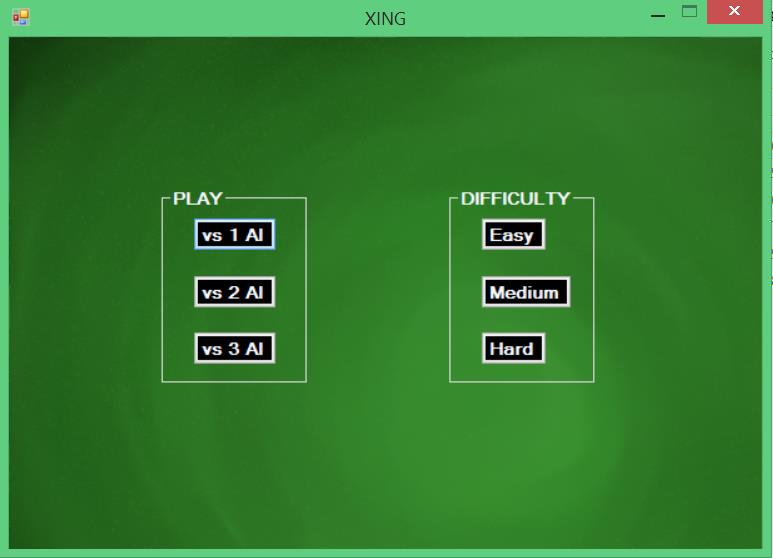
After analyzing all the data I came up with some conclusions.

1. The program is in a very good condition when it comes to being bug-free
2. The three levels offer different game experience and they qualify for what they represent
3. Playing against a single opponent tends to raise your chances of winning. For example let’s take the hard level. If an imaginary player is the same level as the hard AI he would have respectively 50%, 33% and 25% chances of winning. Instead based on the data I have these probabilities are 53%, 30% and 20%. So when you play against 3 hard-level bots than you are below their level, but when you play against 1 bot the average player is better than the bot. I attributed this to the throw of the jack. The computer tends to throw the jack at the beginning of a round while the players usually tend to keep it for the end. The human strategy is very good when you are playing 1 v 1 but the computer strategy tends to do better when there are 3 or 4 player playing
4. The programs works on all Windows computers I have tried it so far including windows XP, windows 7, windows 8 and 8.1.

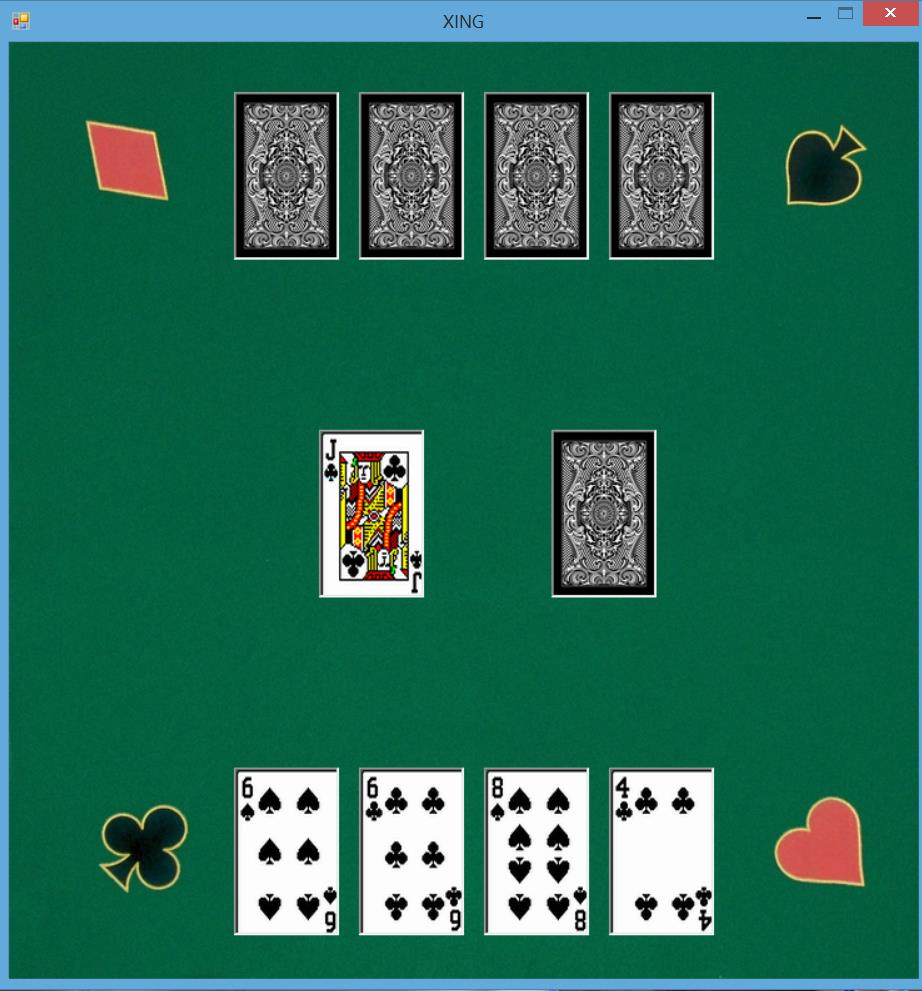
**5.6 PERSONAL GAINS**

With this game I managed to finally create a game I learned in my childhood that I loved so much. I also polished the basic concepts of OOP which I will definitely need in the future. Furthermore I am now more familiar with the Visual Basic IDE and the c# programing language. In the future I will be able to create more complex programs and why not use some of the code I have already created here.

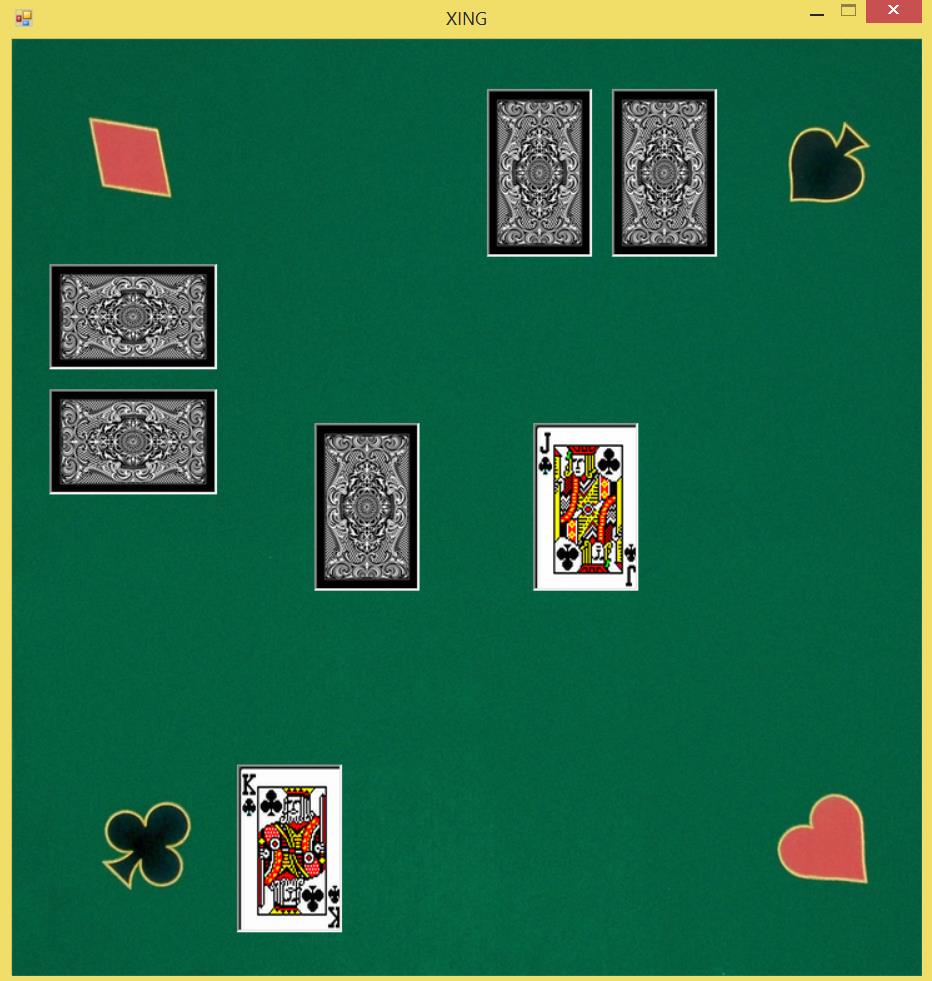
**5.7 THE GAME IN ACTION**



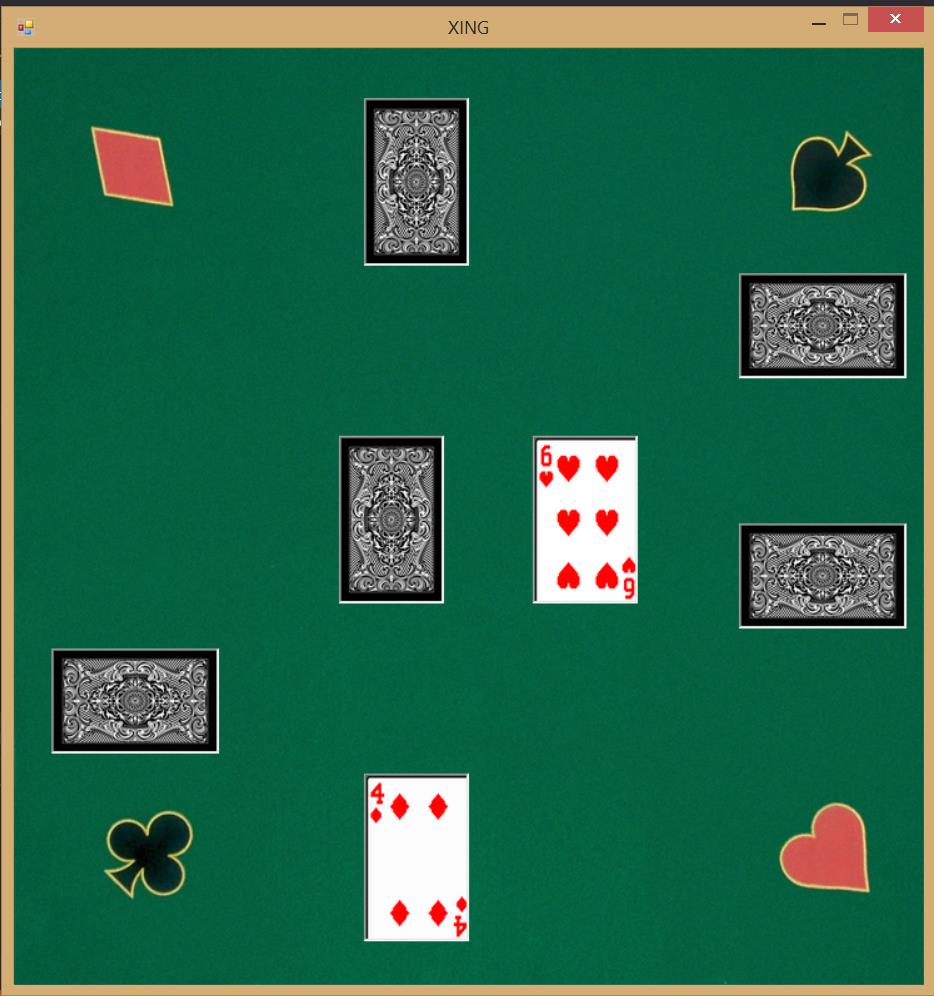
This is the first form that will be shown once you open the game. Two group boxes with radio buttons that look like buttons are displayed and you have to select the number of AI you want against and the difficulty of the AI.



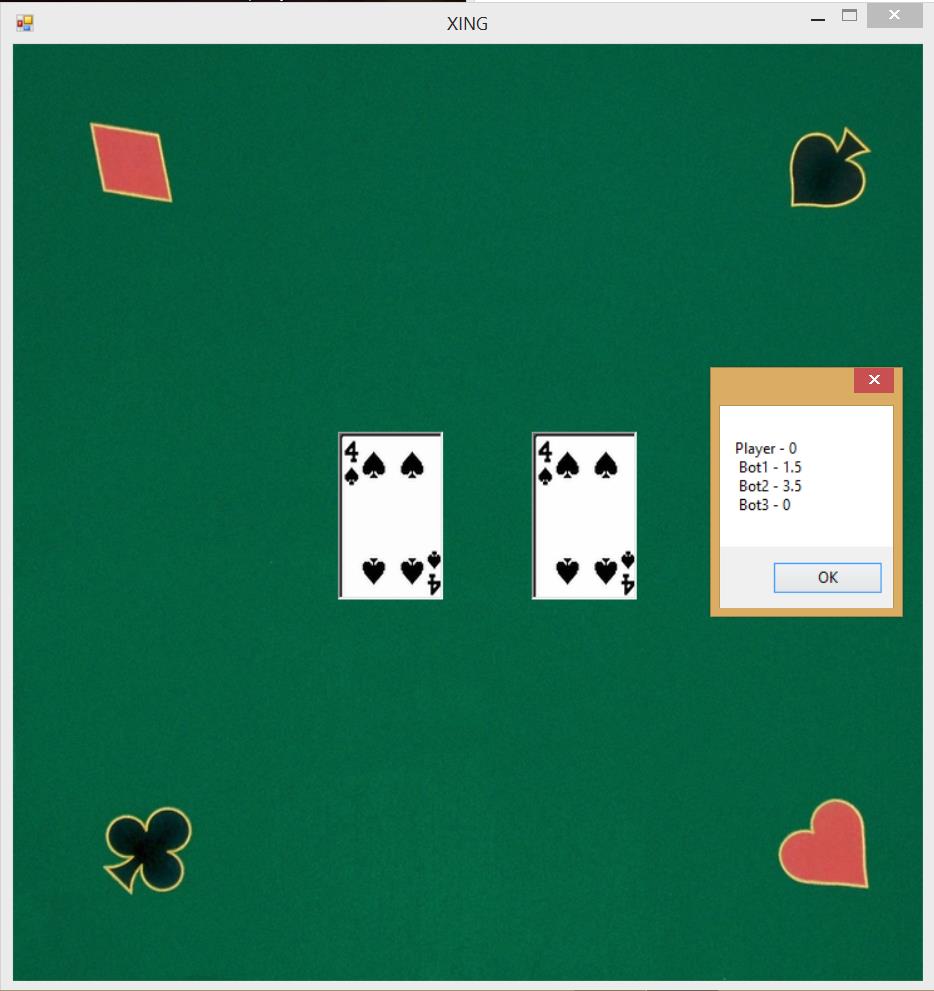
This form will be displayed if you decide to play against 1 bot in either of the levels. The face-up cards in the bottom are you cards while the face-down ones on top are the bots cards. The cards in the middle are the field cards. The one on the left is the collection of all the cards on the field and the card on top is the currently top card. The left image is reserved for the last card thrown by either of the players and it will remain there even if you have matched it.



This is a game for 3 player in action (when you select 2 bots). The last card was thrown by the player and he threw the universal matcher (Jack) so he has collected all the cards on the field that is why the left card on the field is empty. The jack on the left remains there to show the last card that was thrown.



This is the field for a game of 4 (1 human and 3 bots). The bot on top has matched the card that was on the field and he has collected them. It is now the turn of the bot on the right who will throw a card.



In the end a message box will be displayed with the points of every player. This is just another loss from me and a bad one actually. But this loss just makes me happy is it proves that the computer is winning and the the algorithms are working fine.

**6. References**

**6.1 Books**

*Beginning Visual C# 2012 Programming written by Karli Watson, Jacob Vibe Hammer, Jon D. Reid, Morgan Skinner, Daniel Kemper, Christian Nagel*

*Artificial Intelligence: A modern Approach third edition written by Stuart Russell and Peter Norvig*

**6.2 WEB PAGES**

*Webster, Merriam, Merriam Webster Dictionary*

*http://www.merriam-webster.com/dictionary/stochastic*