Interactive Teaching of Poker Strategies

Anton Rand
Supervisor: Dr Peter Lewis

Submitted in conformity with the requirements
for the degree of BSc Computer Science
School of Computer Science,
University of Birmingham

Acknowledgements

I would like to thank Dr Peter Lewis, for his invaluable guidance, support and feedback throughout the year, to which I owe a considerable debt.

I would also like to express my gratitude to Dr Mark Lee for his feedback during the project inspection and after the demonstration.

Finally, I would like to thank my friends and family for the support and feedback offered throughout the year, and for giving up hours of their time so that I could run usability tests with them.

0.1 Abstract

The purpose of this project was to design a software system to provide an all in one solution for users looking to learn poker strategies. The poker strategies taught are for the most popular variant of poker called Texas Hold'em. This report documents the stages taken in the development process including analysis, design, implementation and testing. Which produced a solution involving:

- 6 lessons on poker rules and strategy.
- A customisable free play poker game.
- The recording of various user statistics.
- An achievement system able to recognise when strategies are correctly applied.
- A difficult to beat AI player.

0.2 Poker Terms

For convenience a glossary of poker terms can be found in Appendix B and an explanation of how poker is played can be found in Appendix C.

Table of Contents

Chapter 1

Introduction

Poker is a game where players bet on the strength of the cards dealt to them, there have been so many variants of the game that historians have had difficulty determining where and when it was invented [1].

Texas Hold'em Poker is currently the most popular variant, it saw it's first increase in popularity when the first ever World Series of Poker tournament was played in Las Vegas in 1970 [1]. The popularity of Texas Hold'em was bolstered significantly further thanks to the Internet and three key reasons explaining this stand out:

- Poker became **easier to play**, players could play from their living rooms any time of the day instead of having to travel to a casino and it even became quicker to play, because players could be reseated and decks reshuffled instantly.
- Players were offered a **better deal**; some online casinos host over 300,000 players at a time [2], whilst real casinos would do well to host 300 players. This huge difference in scale allows online casinos to charge a fraction of the commission you would expect to pay in actual casinos, yet online casinos also offer bigger prize funds because thousands of players can take part in tournaments at any given time.
- Players were handed **better opportunities** to become professionals. In 2003, Chris Moneymaker (his real surname) was an amateur poker player who deposited $39 online in an attempt to qualify for the World Series of Poker. He qualified and went on to defy the odds, knocking out numerous professional players and winning the tournament along with $2,500,000 [3].

Chris Moneymaker's big win combined with other success stories encouraged millions of people to try their luck at poker. As strategy is key in poker, a market emerged of players looking to learn new strategies to improve their game.

1.1 Motivation

My motivation for this project comes from my own personal experience as an amateur player looking to improve. I couldn't find a solution that was capable of teaching me new strategies and testing that

I correctly understood the content. As a result I struggled to stay motivated to learn new strategies and it took me several months before I had even learnt the basics.

This gave me the motivation to develop a solution so that other players need not have the same bad experiences as me. It gave me the determination to make the learning process more effective and enjoyable, thus giving users the incentive to learn.

This project also gave me the incentive to further my own poker knowledge, as I developed some lessons in which I had little experience of, so I had conduct research before having the knowledge to implement those lessons.

1.2 Aims

My motivation for developing teaching system is based on the inadequacies of other solutions, therefore it was important to deconstruct the failings of existing solutions so that project aim's could be defined and documented. These aims gave the project a focus and helped ensure that the developed solution didn't suffer the same downfalls as current solutions.

1.2.1 Evaluation of Current Solutions

Three existing methods for learning poker strategies were explored; reading books/websites, watching videos and using software. Their strengths and weaknesses were then evaluated.

Reading Books/Websites

Books and websites are difficult platforms to offer users with feedback; users need to process their own input. For example, a book may offer a multiple-choice question but the burden is on the user to check if they were correct by looking up the answer at the back of the book.

Books and websites are unable to offer users much functionality. Neither of which can offer users the functionality of a playable poker game, nor can they recognise users progress or record user statistics. Consequently, the longevity of the product suffers.

An advantage of learning through books or websites however is that it is a relatively portable option, as users can read up on lessons anywhere throughout the day.

Watching Videos

Videos share many of the same downfalls as reading books/ websites, it is difficult to process user input and they have a low level of functionality.

The development of smartphones has made it relatively easy for users to watch videos on the move however, whilst another positive is that videos can provide more visual information than books, which may be more suitable for some learning styles.

Using Software

Software is an ideal platform to provide users with feedback, but of the few software solutions available many of which choose not to provide users with any feedback.

Software can also provide much more functionality and interactivity than the other two solutions. It is possible to simulate games of poker, recognise user progress and record user statistics. Yet nobody has implemented such a solution yet.

Portability of software is worse than other options, smartphone screens are too small to develop a software app for and this is demonstrated by the lack of apps available through the Apple or Android stores. This means that users who want to learn through software are required to do so through their computer or laptop.

1.2.2 Conclusions

After evaluating the existing methods, three conclusions were made:

- Current methods are **uninvolving**, books and videos are impractical for input, whilst existing software chooses not to provide users with feedback.
- Current methods are **unenjoyable**, users can not see the progress they make and they do not have the opportunity to play games for fun.
- Current methods are **ineffective** , users who do not find learning involving or enjoyable are less likely to remember what they are taught, which was the case for me.

1.2.3 Project Aims

Taking all of this into account, 4 project aims were determined which shaped the final solution:

1. Make teaching more **involving** than current methods.
2. Make teaching more **enjoyable** than current methods.
3. Make teaching more **effective** than current methods.
4. Develop an easy to use solution.

1.3 Solution

The end result of the project is a software system called 'Poker Coach', an all in one system comprised of 5 components:

- The implementation of a playable poker game, which conforms to the rules of Texas Hold'em. Users can customise the game and set the difficulty of their opponents.
- A lesson center to teach players 6 lessons on the rules and strategies of poker, lessons are filled with enough content to last between 5 and 10 minutes. Throughout the lessons users are provided with explanations, simulations, tests, hints and feedback.
- A statistics center which informs players of their progress, telling them how much profit they have made, their return on investment and their win percentages amongst other statistics.
- An achievement center, which recognises when players successfully apply a taught strategy and rewards them on their progress with achievements to keep them interested in the learning process.
- An advanced AI player, that implements the Monte Carlo method of sampling to determine which action to take. The player is also capable of applying some of the strategies taught to players in the lessons and has 3 levels of difficulty.

1.4 Report Structure

Having talked about the current methods of learning new strategies and justifying why there is a requirement for my solution, the rest of the report documents the processes taken and decisions made which influenced the final solution.

The project started with analysis of the problem, a user specification was documented to determine the required functionality of the project.

Design and Implementation followed, several key algorithms are explained and justifications are made for the chosen approaches. Test strategies are also explained.

The process of developing the Graphical user interface (GUI) is then explained, followed by details on project management and survey results.

Finally, the project is critically evaluated.

Chapter 2

Analysis and Specification

Analysis of the problem was required before beginning development of system, it was important to understand the required functionality of the system, so that the problem could be broken down and managed effectively. This began with the development of a user specification, which expanded on the initial 4 high-level requirements, allowing me to increase the clarity on what the system should do.

Design of a user specification was crucial to ensuring the system was designed as originally planned, without clearly defined requirements there would be a risk of the project not achieving it's key aims.

The user specification was also useful for validating the success of the project as well as helping to gain a further understanding into how the system should behave.

2.1 User Specification

Analysis of existing systems was useful in determining the user specification, formed of functional and non-functional requirements. These clearly defined requirements would be useful in evaluating the differences between the initial design and the final product.

2.1.1 Functional Requirements

The following functional requirements were documented to define the expected behaviour of the system:

1. The system must allow new users to register an account.
    (a) The system must request username input.
        i. The system must validate that the username entered is unique.
    (b) The system must request forename and surname input.
        i. The system must validate the format of the forename and surname.
    (c) The system must require users to specify their experience level.
        i. The system must provide a selection between beginner, amateur or experienced.
    (d) The system must require completion of an introductory lesson before completing registration.

i. The system must interact with the database.
   A. The system must add the user record to the database.
   B. The system must record lesson and hand data.

2. The system must allow existing users to log in.
   (a) The system must provide an interface for users to log into upon start up.
      i. The system must validate log in credentials against formatting conventions.
      ii. The system must validate log in credentials against the database entry.
      iii. The system must provide error feedback to users.
         A. The system display the relevant profile screen upon successful login.

3. The system must display a customised profile screen to users providing a summary of their progress.
   (a) The system must display a screen showing relevant details.
      i. The system must display the users name.
      ii. The system must display achievements.
         A. The system must show unlocked achievements.
         B. The system must show locked achievements and the explain unlock conditions.
      iii. The system must display user statistics.
         A. The system must show a users best hand.
      iv. The system must show the list of lessons available to users.

4. The system must allow users to play a game of poker.
   (a) The system should provide support for up to 7 AI opponents.
   (b) The system should display the users hole cards.
   (c) The system should display the community cards.
   (d) The system should provide feedback on the actions taken by opponents.
      i. The system should show the action taken for each player (call, check, fold, raise).
      ii. The system should display the minimum bet for users to remain in a hand.
   (e) The system should provide an interface for users to take actions.
      i. The system should allow users to call, fold, check and raise when allowed.
      ii. The system should ensure a player cannot carry out an invalid action.
   (f) The system should detect when the game is over.
      i. The system should display a summary of the game.
      ii. The system should take users back to the main menu.

5. The system must teach users new strategies.
   (a) The system must provide a range of lessons for users to take.
      i. The system must provide an introduction to poker.
         A. The lesson must explain how a typical game is played.
         B. The lesson must explain big and small blinds.

C. The lesson must explain split pots.
D. The lesson must explain the different combinations of hands.

ii. The system must provide a lesson on 'bluffing'.
A. The lesson must explain the reason for 'bluffing'.
B. The lesson must explain the best conditions for 'bluffing'.
C. The lesson must explain the types of 'bluffing'.

iii. The system must provide a lesson on determining hand strength.
A. The lesson must explain the importance of having strong cards.
B. The lesson must explain the importance of having suited cards.
C. The lesson must explain the importance of having connected cards.
D. The lesson must explain what to do with pocket pairs.

iv. The system must provide a lesson on 'slow playing'.
A. The lesson must explain the types of player you could maximise your profits off by 'slow playing' against.
B. The lesson must explain the conditions for 'slow playing'.

v. The system must provide a lesson on the different stages of a hand.
A. The lesson must explain the difference between betting pre-flop and post flop.
B. The lesson must explain which strategy it is best to adopt for each stage of betting.

vi. The system must provide a lesson on position.
A. The lesson must explain the concept of early, middle and late position.
B. The lesson must explain the positions of the small and big blinds.
C. The lesson must explain how late position can benefit players.
D. The leson must explain how to act based on position.

vii. The system must provide a lesson on when to go 'all in'.
A. The lesson must explain what kind of players to go 'all in' against.
B. The lesson must explain the type of hand that warrants going 'all in'.

(b) The system must determine when users have successfully learnt a lesson.
i. The system must unlock a new lesson when users successfully complete a previous one.

(c) The system should grade each lesson.

6. The system must detect when users apply different strategies.

(a) The system must check for plays at the end of each hand.
i. The system must detect 'slow plays' and 'bluffs'.

(b) The system must check hands against achievement conditions.
i. The system must detect when an achievement has been unlocked.
ii. The system must update the database when an achievement has been unlocked.

7. The system must store data in a database.

(a) The system must store user data.
i. The system must store usernames.
ii. The system must store full names.

(b) The system must store deck data.
   i. The system must store each players hole cards.
   ii. The system must store each players gains or losses.
   iii. The system must store the stage of play when the hand ended.

(c) The system must store achievement data.
   i. The system must store a list of users who have achieved each achievement.

(d) The system must store lesson data.
   i. The system must store the grades of each lesson took by each user.

8. The system must interact with a database.

(a) The system must allow SQL UPDATE queries to be executed on the database.
   i. The database must allow entries to be updated.
   ii. The database must allow new entries to be added.
      A. The database must allow achievements to be added for users.
      B. The database must allow lesson grades to be added.
   iii. The database must allow entries to be deleted.

(b) The system must allow SQL SELECT queries to be executed on the database.
   i. The database must allow player statistics and data to be retrieved.
      A. The database must allow a players best hand to be retrieved.
      B. The database must allow a players achievements to be retrieved.
      C. The database must allow a players available lessons to be retrieved

2.1.2 Non-Functional Requirements

The following non-functional requirements describe general system requirements. To allow each requirement to be validated easily values are given where applicable. The requirements are ordered by Sommerville's requirement categories where applicable.

Usability

- Appropriate colour scheme should be used to make reading text easy.
- 3 click maximum to carry out any poker action.
- Average lesson time between 5-10 minutes.

Performance/Efficiency

- 100% of AI players should make a decision within 5 seconds.
- Log in response within 5 seconds.
- Lessons loaded within 5 seconds.

- Profile displayed within 5 seconds.
- The system should not allow concurrent use.

Resource Constraints

- The system should be a similar size to other available software solutions.
- The system should store all images in highly optimised PNG format.
- The system should be compatible across all platforms.

Standards

- The system should conform with official Texas Hold'em rules.

Delivery

- The system should be completed by 19th March 2012.

Implementation

- The system should be easy to maintain.
- The system should contain many basic classes as opposed to few complex ones.
- The system should be easy to test.
  - JUnit tests should be implemented to test classes where possible.
- The system should be designed for a minimum 1024 x 768 resolution.

Security

- 100% of SQL queries executed with prepared statements.

2.2 Analysis

2.2.1 Use Case

The following use case diagram was designed to model the basic functionality of the system.

Figure 2.1: Use case diagram modelling system functionality.

2.2.2 Feasibility Study

A feasibility study in the project proposal concluded that the project should be coded using Java, although SQL could also be used to implement the database. This decision was reached because I

have a strong understanding of both languages and because both languages are cross-platform. It was also concluded that online functionality would be achievable through the schools server, although implementation of this functionality is not planned.

2.2.3 Validation of Aims

The user specification would play an important role in validating that the system does what is required, but it was also decided that validation criteria should also be set for the 4 project aims in the introduction, to see how the project performs against the alternatives.

It is proposed that users are asked to learn a new poker strategy through each of the different methods, including my software solution. Users will then be given a survey aferwards.

Validating Increased Involvement

- Survey should show that users preferred, paid most attention to and felt more involved with my solution.
- Comparison of my solution against alternatives, shows that my solution requires more user input.

Validating Increased Enjoyment

- Survey results should show a preference to learning again through my solution.
- Survey results indicate that users agree that the system is fun and that achievements add to the enjoyment.

Validating Increased Effectiveness

- Ask users to take a tournament after registration, after users have took several lessons ask them to take another tournament, a noticeable improvement should be observed.
- Logged statistics show an improvement in player judgements after progressing through the lessons.
- Survey results indicate that users agree that they performed better after taking lessons.

Validating Ease of Use

- Regular usability testing held to determine how to improve ease of use.

- Prototyping used for GUI design.
- Survey results indicate that users agree that the system is easy to use.
- Personas developed for different user abilities to emphasise and understand their interaction with the system.

2.2.4 Timetable & Risk Management

Before beginning the project I knew it was going to be challenging, therefore the decision was taken to break the problem down and timetable the expected completion times of project components. For the initial timetable please see Appendix D.1.

The action was also taken to identify components in the timetable, which carried significant amounts of risk. Fallback positions were then defined for these components so that the project could be effectively managed and given the best chance of success whilst remaining ambitious. The core focus focus of teaching could also be preserved.

The identified risks are:

Development of the command line game carries risk as too much time could be taken in developing a solution which will later become obsolete. If development ran behind schedule unimportant functionality of the command line game should be left out.

Development of Hi-Fi prototypes is often time consuming. If the project is running behind schedule only one Hi-Fi prototype should be developed.

Development of lessons could also be time consuming. As they are expected to last between 5-10 minutes a lot of research is required. If the development of lessons runs behind schedule the content should be condensed and the number of lessons reduced. So that the quality of the lessons developed does not suffer.

Development of achievements is not the core focus of the project, so if the development of recognition algorithms becomes an issue, the complexity can be reduced.

Development of the statistics could be more difficult than anticipated, if this is the case the volume of recorded data should be reduced. As a minumum; profit, best hand and number of wins should be recorded.

Chapter 3

Design

The following chapter documents the design and architecture of the finished product, describing how the key components of the project have been designed and how the system functions. Decisions made, which altered the planned design of the system, are explained in the implementation chapter.

3.1 System Overview

3.1.1 System Architecture

The core system architecture has represented through class diagrams shown in figures 3.1 and 3.2. For simplicity only the key fields and methods have been included.

The architecture of the system is based on a view and model. Figure 3.1 shows that the View class contains a Game object as a field, which represents the model.

Games are played within this Game class by constructing new Hand instances. Inside this Hand instance the players, pots and deck are contained. It is in this Hand class where users take action and progress through the different stages of hands.

The Game class was initially very complicated upon design, so in accordance with the non-functional requirement of simplifying classes, a GameDetails class was designed to simplify the architecture and reduce complexity.

The lesson mode is built from the above architecture, where abstract classes were designed so that the implementations could differ. The different implementations are highlighted in figure 3.2, which shows that lessons require the functionality to display lesson content, hints, arrows and the players score. Which differs from the implementation of free play, which requires functionality to display the time remaining until the blinds are increased and to check after each hand whether players have unlocked new achievements.

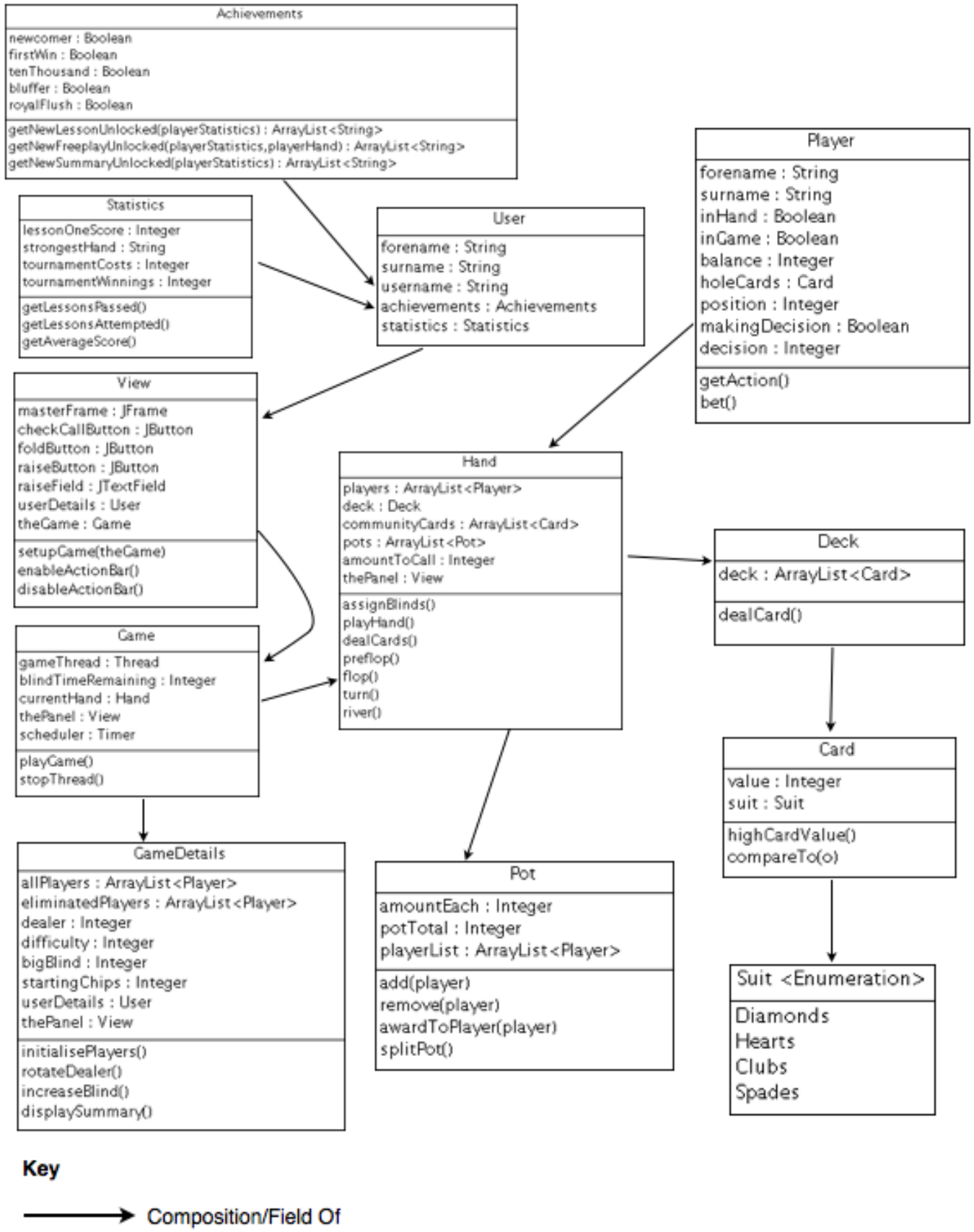

Figure 3.1: Class diagram representing the core system architecture.

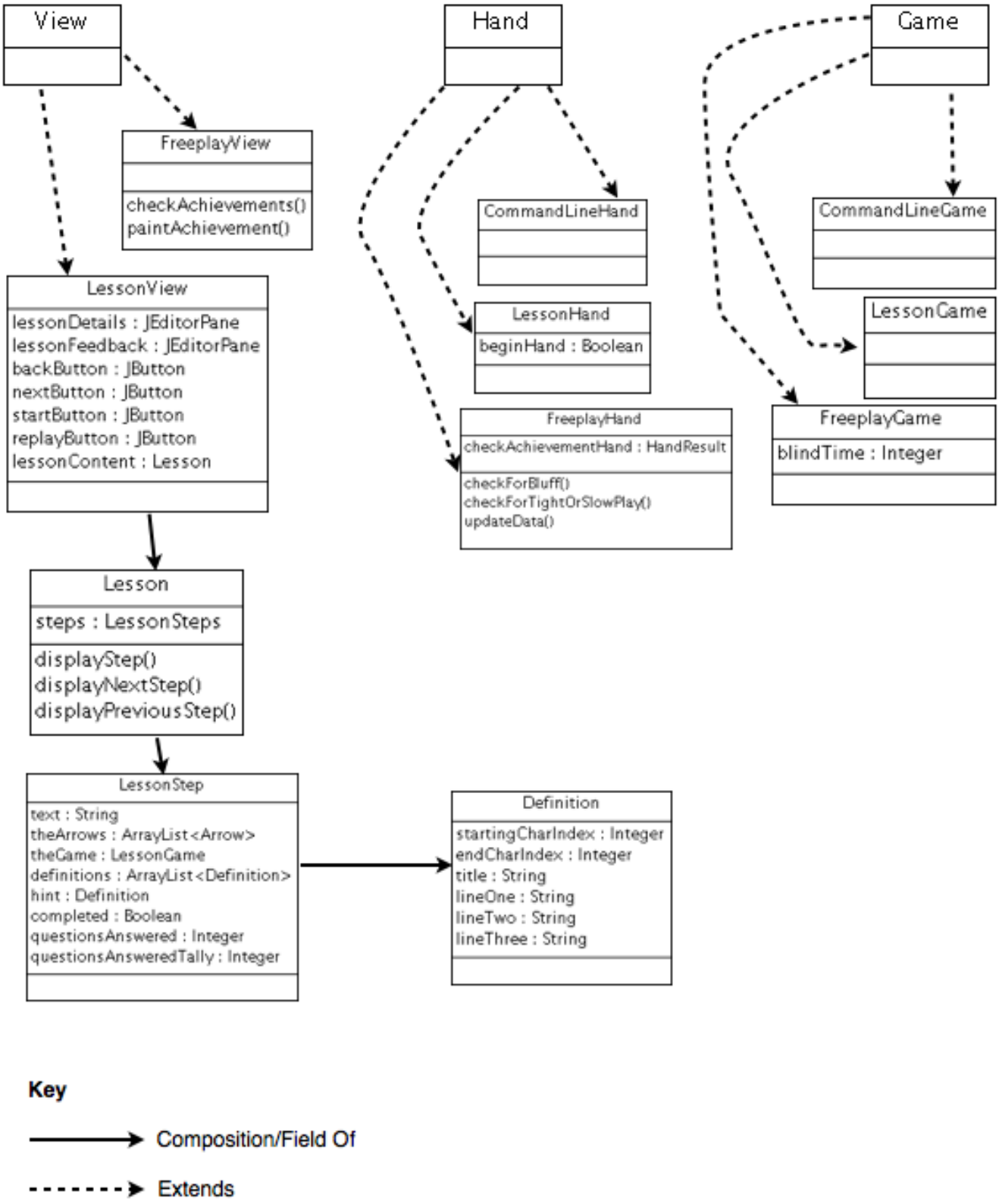

Figure 3.2: Class diagram representing different implementations of 'View', 'Hand' and 'Game'.

3.1.2 Player Architecture

The game features different types of players; therefore an abstract 'Player' class was designed to model the attributes all players have in common. Each different player was designed in a separate class which extends the 'Player' class, which includes an implementation of the abstract 'getAction()' method. This method is responsible for determining the action taken by the player and differs for each implementation of a player.

The differences between each player are itemised below:

AIPlayer - The first AI player designed for users to play against, which makes decisions based on rules.

AdvancedAIPlayer - An improved AI player which makes decisions based of simulations.

CommandLinePlayer - The first implementation requiring user input. Allows users to play by typing in an action through keyboard input.

GUIPlayer - Implementation of 'Player' class which allows users to play by detecting input through the action panel buttons.

GUILessonPlayer - Implementation of 'GUIPlayer', but with additional functionality. Lists of acceptable actions are passed into the constructor and the action taken is detected by detecting input through the action panel buttons. The action taken is determined based on this input and it is decided whether the player answered correctly and can take that action, or whether the answer was incorrect and should choose a different action to take.

LessonPlayer - Used to instruct computer players what action to take so that lessons can be simulated. 'ArrayLists' of integers are passed into the constructor and the player pops and returns the first value from the list each time 'getAction()' is called. Values such as '-1' instruct the 'LessonPlayer' to call any amount and once the list becomes empty the player will check/fold.

The player architecture is shown through a class diagram in figure 3.3. It displays the differences between player classes and the implementations.

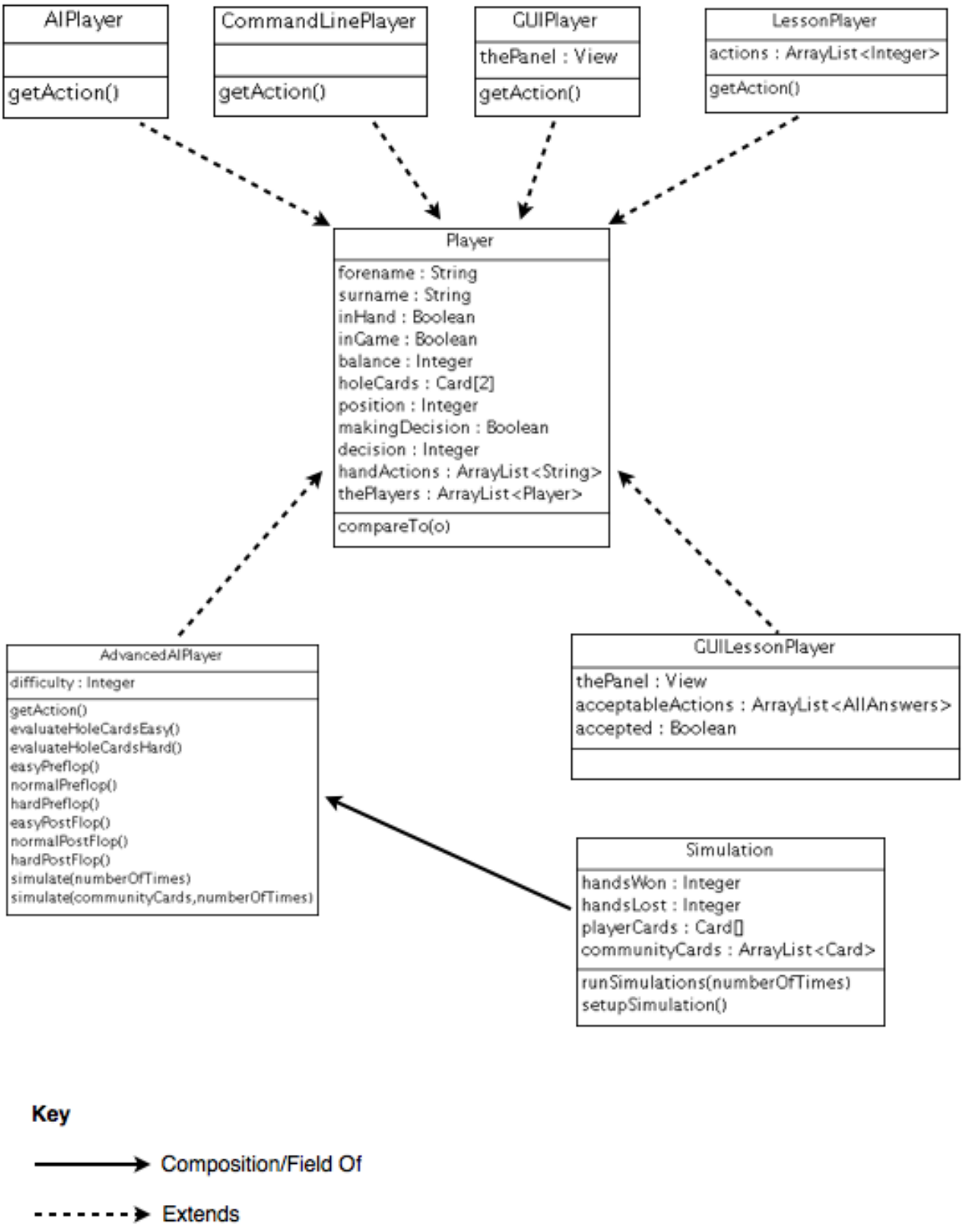

Figure 3.3: Class diagram showing the different implementations of the 'Player' class.

3.2 Key Components

3.2.1 Game Design

Pots

Pots are used to represent the winnings each player is entitled to. If a player goes all in and wins, they cannot win more than this amount from each other player. The 'Hand' class uses an array of pots to ensure that players are only capable of winning what they are entitled to.

Each 'Pot' instance is constructed with an integer, which represents the amount each player must contribute into the pot. An 'add()' method is used to add players to the pot, where the amount to contribute is deducted from their balance and they are added to the list of players entitled to win the pot.

When winners are evaluated at the end of hands, losing players are removed from this list through a 'remove()' method and the pot is distributed evenly amongst the remaining players.

The situation can arise where a pot cannot be split evenly, e.g. Splitting $101 between 2 players. In accordance with the rules of Texas Hold'em any remainder is awarded to the player with the highest hole card, or the player with the best-ranked suit in the event of a tie.

Hand Comparer

To determine the winning players at the end of each hand, 'HandComparer' and 'HandResult' classes were designed.

When the 'compare()' method of the 'HandComparer' class is called, a 'HandResult' instance is returned which consists of an integer to represent the hand category and an 'ArrayList' holding the 5 card representation of the hand.

The 'compare()' method functions by checking if each category can be made, starting with the strongest. For each category that cannot be made a null object is returned. The 'compare()' method checks each category until a 'HandResult' object is returned that is not null, this object represents the players strongest hand as is subsequently returned by the 'compare()' method.

A player will always have 1 of the 9 possible hand categories, so the 'compare()' method is guaranteed to return a valid 'HandResult' object.

Four of the hand detection methods are summarized below. Full explanations of all methods are available in the documentation folder of the provided CD.

Note : Cards are sorted in descending order before each method is called, this allows the highest cards to be taken into consideration first.

Four of a Kind

- Create 4 empty 'ArrayLists'.
- For each card in the hand, add it to one of the 'ArrayLists' if the rank matches, otherwise add it to the next empty list.
- Four of a Kind occurs if one of the lists contains 4 cards. The list with 4 cards is returned along with the first card from the earliest remaining list. This 5^{th} card is known as a 'Kicker'.

Full House

- Create 4 empty 'ArrayLists'.
- For each card in the hand, add it to one of the 'ArrayLists' if the rank matches, otherwise add it to the next empty list.
- Full House occurs if one of the lists contains 3 cards and another contains 2 or 3 cards. The first list containing 3 cards is returned along with the first two cards from the other list.

Flush

- Create 4 empty 'ArrayLists', one for each suit.
- For each card in the hand, add it to one of the 'ArrayLists' if the suit matches, otherwise add it to the next empty list.
- Flush occurs if one of the lists contains at least 5 cards. The 5 cards from the list (highest) are returned.

Straight

- If the hand contains an Ace (Card of rank 14), add it's low value to the list (1) as well and re-sort the list.
- For each card that is followed by 4 others, enter a nested loop to check that the rank of the next 4 cards is 1 less each time.
- If this is true then a Straight has occurred. Return the 5 cards that were checked.

3.2.2 Lesson Design

Research

Research was conducted before each lesson was designed to determine the structure and content, this ensured that the quality of lessons was high and by using numerous sources I could be sure of the correctness of each lessons content and identify the most important topics, as they appeared more often across sources.

A plan was created for each lesson, denoting the lesson structure through summarised key points. Lesson One's plan is shown in Appendix E and all plans can be viewed from the documentation folder of the CD.

Implementation

An overview of the lesson architecture is shown in figure 3.2 through a class diagram. The lessons were designed so that users are taught through 3 ways:

- Reading about the strategy.
- Watching scenarios unfold.
- Determining the correct action to take in test.

Appendix item G.3 shows the final design of the lesson system. Key features of the lesson design include:

- Content displayed through a 'JEditorPane' on the right hand side bar, formatted in HTML. HTML was used so that the content could be neatly formatted and coloured.
- Details of player actions shown through a 'ColorPane' class reused from an example on the Internet [4]. This class was used as it allowed actions to be formatted in colour, making details easier to read for users.
- Definitions that appear when words are hovered over. Code was reused from an Internet example to detect the character index in the 'JEditorPane' that is being hovered over [5]. The definition of a word is drawn when the current character index falls between values specified in the Definition class.
- 40 pre-programmed playable lessons. Each with a pre-determined deck, set of players and list of correct answers. When a test/simulation is available two buttons appear allowing players to start, pause and replay the hand. Users answers are recorded to provide a score and feedback is given when players fail to answer a question correctly 3 times in a row.

3.2.3 AI Player

Basic AI Player

A rule based AI player was first designed, with the hope of implementing a better player during later stages of development. The rules followed by the basic AI player are explained below:

Pre-flop

- If the player has a pocket pair or a combined hole card rank above 23 (rank of Ace is 14):
  - **Call**, if the amount to call is 15% or more of balance.
  - **Raise**, if the amount to call is less than 15% of chip balance.

Post flop

- If the player has better than a Three of a Kind:
  - **Call**, if the amount to call is 50% or more of chip balance.
  - **Raise 3 times**, if the amount to call is less than 50% of chip balance.
- If the player has better than a High Card:
  - **Bet**, one times the big blind if amount to call is $0.
  - **Call** otherwise.
- If the player has worse than a High Card.
  - **Check/Fold**

Advanced AI Player

During later stages of development an advanced AI player was implemented, providing users with a tougher challenge. Instead of using rules, this solution uses the Monte Carlo method by randomly simulating the outcome of hands. A focus was put on making the player less predictable than the previous AI player and on implementing 3 different levels of difficulty.

For each of the three difficulties the AI player has a method for determining what action to take pre-flop and one for determining what action to take post flop. These methods are called through the 'getAction()' method which behaves as follows:

- If difficulty is **easy**.
  - Determine action using easy pre-flop/post flop method 70% of the time.
  - Determine action using average pre-flop/post flop method 20% of the time.
  - Call 10% of the time.
- If difficulty is **average**.
  - Determine action using hard pre-flop/post flop method 65% of the time.
  - Determine action using easy pre-flop/post flop method 25% of the time.

  - Call 10% of the time.

- If difficulty is **hard**.
  - Determine action using difficult pre-flop/post flop method 80% of the time.
  - Determine action using average pre-flop/post flop method 15% of the time.
  - Call 5% of the time.

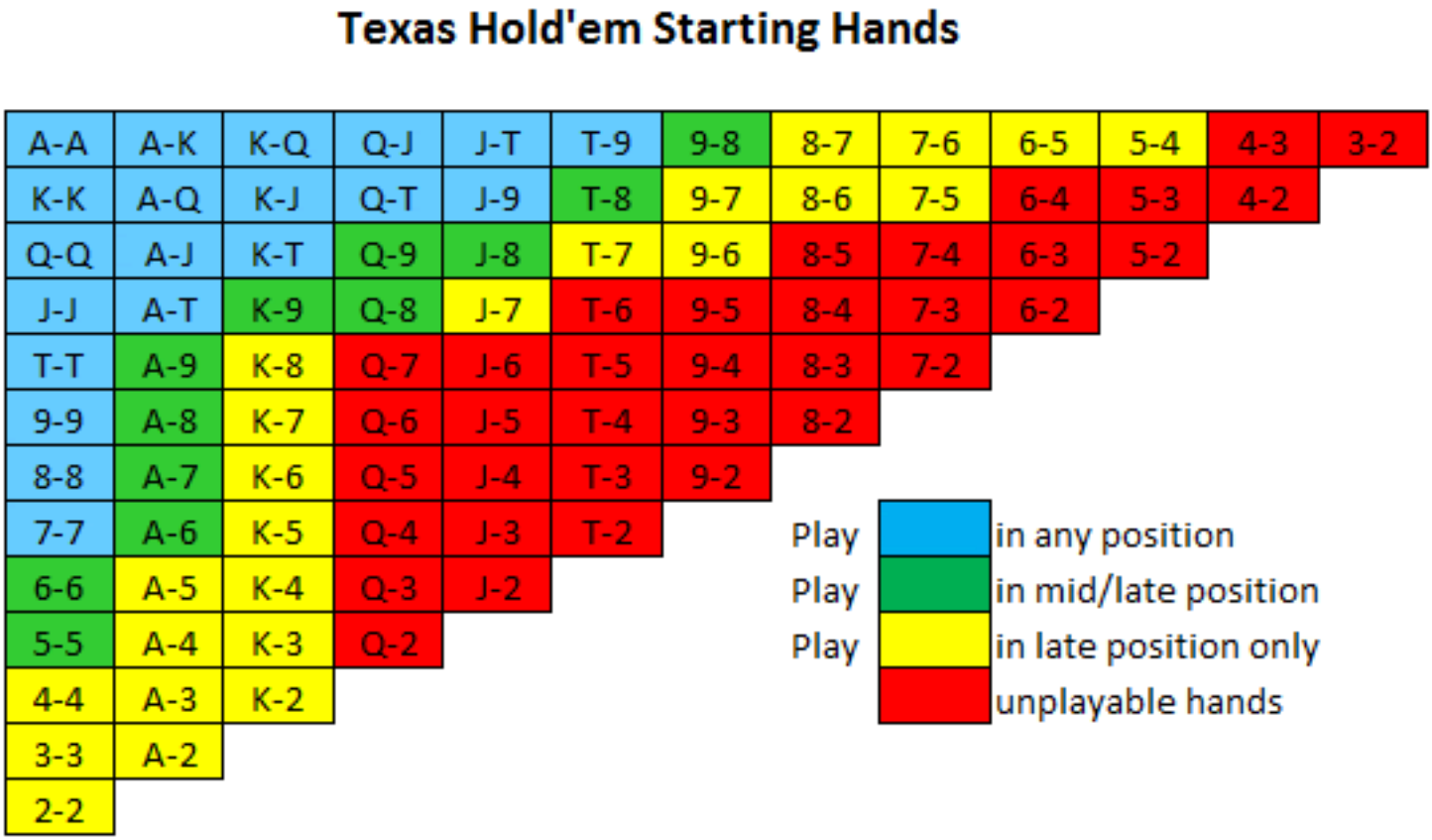

Figure 3.4: Shows which hole cards a player should go in a hand with. [6]

Pre-flop

The table of acceptable hole cards shown in figure 3.4 is widely used throughout lessons. It helps players to determine whether they have adequate hole cards to participate in a hand based on the strength of their cards and position. Each pre-flop method adds a selection of these cards to a list and then checks for a match with their hole cards.

Easy Pre-flop

Players are easier to beat when they take unnecessary risks. The easy pre-flop method doesnt consider position and instead uses a random selection of cards from the table:

- The method always checks whole cards against 'any position' cards.
- 50% of the time, check for match with the 'mid/late position' cards.
- 30% of the time, check for match with a random sample of 20-60% of the 'late position' cards.

Random numbers are generated each time the 'getAction()' method is called, this is used to determine which cards to check. By using random numbers the player becomes unpredictable, users cannot be sure if the player has called with good or bad cards.

Hard Pre-flop

The hard method takes position into account and follows the table accordingly, although predictable this method makes the player consistant and less likely to be beaten.

Average Pre-flop

The average method uses a combination of the easy and hard pre-flop methods:

- Determine action using hard pre-flop method 70% of the time.
- Determine action using easy pre-flop method 25% of the time.
- Call 5% of the time.

The average method represents an inconsistant player who is mostly intelligent but also prone to making the mistakes of an easy player.

Post flop

The number of card combinations increases dramatically after the flop, as a result the optimal strategy to take cannot be represented through a table. The AI player was designed to follow a Monte Carlo strategy where simulations are made to detemine the players chance of winning from a random sampling.

Simulations are constructed by using the players hole cards, the community cards and the number of opponents. A random deck is initialised and the known cards (hole/community cards) removed from this deck, cards are then dealt from this deck to estimate what hole cards opponents have and what the remaining community cards are.

A 'HandChecker' is used to compare players different hand results, and result of each simulation is recorded to determine a win percentage.

A player determines what action to make based on their simulated win percentage and the expected win percentage. The expected win percent is equal to 100 divided by the number of players involved in the hand. So in a hand with 4 players, each player would expect to win 25% of the time.

It is not enough to just call any bet where the simulations were favourable, the amount a player raises is supposed to indicate how confident they are of winning. The easy and hard post flop methods determine what action to take based on the amount to call.

Easy Post flop

100 simulations are ran, the player then makes a decision based on the percent of hands won from the simulations. Some of the required percentages to call/raise are shown below:

Number of Opponents	**Amount to Call (Big Blinds)**	**Simulation Win % to Call/Raise**
1	$0 \leq$ Amount to call < 1	50%
1	$1 \leq$ Amount to call < 2	55%
1	$2 \leq$ Amount to call < 3	58%
1	Amount to call > 3	60%
2	$0 \leq$ Amount to call < 1	33%
2	$1 \leq$ Amount to call < 2	38%
2	$2 \leq$ Amount to call < 3	41%
2	Amount to call > 3	43%
3	$0 \leq$ Amount to call < 1	25%
3	$1 \leq$ Amount to call < 2	30%
3	$2 \leq$ Amount to call < 3	33%
3	Amount to call > 3	35%

Hard Post flop

The hard post flop method is more conservative than the easy method. The table shows that the method will only call/raise when the odds are extremely favourable:

Number of Opponents	Amount to Call (Big Blinds)	Simulation Win % to Call/Raise
1	$0 \leq$ Amount to call < 1	60%
1	$1 \leq$ Amount to call < 2	65%
1	$2 \leq$ Amount to call < 3	70%
1	Amount to call > 3	75%
2	$0 \leq$ Amount to call < 1	43%
2	$1 \leq$ Amount to call < 2	48%
2	$2 \leq$ Amount to call < 3	53%
2	Amount to call > 3	58%
3	$0 \leq$ Amount to call < 1	35%
3	$1 \leq$ Amount to call < 2	40%
3	$2 \leq$ Amount to call < 3	45%
3	Amount to call > 3	50%

Both methods follow these tables, but random numbers are generated so that AI players sometimes raise when the odds are not favourable. This is an implementation of bluffing and makes the player more difficult to beat, as a raise by an AI player doesn't necessarily indicate that they have strong cards.

Average Post flop

The average post flop method is a combination of the easy and hard post flop methods, as previous explained.

3.2.4 Achievement System & Recognition

Users have their log in credentials, statistical data and achievements stored in a 'User' class which is saved to a YAML file. The achievement class is formed of 18 different achievements represented as booleans which are all initialised to false.

Achievements can be unlocked during 4 stages:

- **Log In** - When a user logs in for the first time they unlock the newcomer achievement.
- **At the end of each hand** - It is checked what the players winning hand was and if they successfully applied a strategy.
- **At the end of each game** - It is checked whether a player has earnt a set amount or won a given number of tournaments.
- **At the end of a lesson** - It is checked whether a player has completed a set amount of lessons or achieved a combined score of 100%.

Checking for Bluffs

The algorithm designed to check for bluffs works for many cases but is not perfect. I believe detection of bluffs could be improved by using more rules. The current implementation is sufficient however and works as follows:

- If the player won the hand
  - If there were less than 4 players in the hand.
    - If (50% of pot) > Amount Bet < (100% of pot)
      - If best hand is high card —> Bluff occurred

Checking for Slow Plays

To check if a player has slow played a list of their actions for each hand is recorded, this is then analysed at the end of each hand. A player slow plays when the following conditions are true:

- If the player won the hand
  - If the player did not raise at any time during the hand
    - If the player called at least one raise during the hand.
      - The player successfully slow played if they had a two pair or better.

Checking for Tight Plays

- If the player won the hand
  - They played with tight cards if both of their hole cards have a rank of 10 or above.

Chapter 4

Implementation and Testing

This chapter gives an insight into the decisions during development which shaped the final product and explains how I overcame problems. I justify the data structures used and present the test strategy.

4.1 Decisions

4.1.1 Prototyping

System

During the Analysis phase, the problem was broken down into manageable work packages. One of the key decisions made was to first implement a working prototype of a poker game to run through command line. This decision was taken to prevent over complicating the project from the start and to allow the initial focus to be on implementing an accurate representation of the game. By developing the core of the project first, it was then easier to implement additional functionality later.

GUI

The decision was also taken to create Low-Fi and Hi-Fi prototypes of the GUI so that feedback from users could be acquired before the actual GUI was implemented. This ensured that the implementation time of the GUI was reduced as changes were identified prior to implementation. However, some decisions were made during the implementation phase which altered the final design of the GUI:

- It was decided that player ability levels be removed from the system, This is because the ability of the user has no influence on the lesson content and caused too much confusion for users during registration.

- During implementation of the free play mode, sliders were used to replace text boxes. This was due to the confusion caused to users who were unsure what to enter. Sliders have fixed values which help guide users through the free play setup.

- Changes were made to the action bar. When users type invalid raise amounts the raise button used to display the last valid raise typed, to prevent users from raising by an amount they didn't intend to raise by this is no longer the case. If a user types an invalid amount the raise button is disabled until a valid amount is entered.

- A hand strength meter to display the users current hand category was initially identifed as a non-essential component. Feedback from users prompted me to re-evaluate its importance and the decision was taken to implement this feature.

- Feedback from an experienced player led to the implementation of game details in the free play mode. The details displayed are the current blind level, next blind level and time remaining until the next increase.

4.2 Problems

A few problems arose during implementation, the main problems with the project are summarised below.

4.2.1 Game Implementation

The development of the command line game was delayed due to difficulties implementing the pots. Analysis did not give an indication that the implementation of a 'Pot' class was required, but it was analysed that development of the command line game may be harder than first predicted. I took the decision to spend extra time developing the game, as opposed to rushing through development and allowing the problem to escalate. The problem was managed by revising the timetable and setting new completion dates.

4.2.2 GUI Implementation

During development of the Hi-Fi prototype, it was observed that the task was taking longer than expected to complete and that the project was falling behind schedule. The decision was made to revise the number of Hi-Fi prototypes down to 1. This enabled me to develop a high quality prototype instead of 3 prototypes of reduced quality.

Development of the GUI began behind schedule so I took the decision to catch up with work during the Christmas break. This decision was key to the GUI being completed by the start of January and allowed me to catch up with the timetable.

4.2.3 Lesson Implementation

Lessons were identified as carrying risk due to the amount of research and coding needed to be undertaken before any implementation is possible. Implementing a new lesson weekly was the most challenging part of the project, and as I encountered difficulty with meeting this aim I decided to take action and revise the timetable further, as shown in Appendix D.3.

The revised timetable shows that I managed the problem of implementing lessons by condensing the content into 4 lessons. Later evaluation revealed that there was enough content to increase the number of lessons up to 6. I kept the lessons on slow play and going all in merged, as there wasnt enough content for each to become a separate lesson.

4.2.4 Implementation of Statistics & Achievements

During the implementation phase, it was decided that statistics and achievements could be better implemented through YAML, along with user data. The decision was taken to implement through this format instead of through a database using SQL because the implementation process would be made easier and the development time reduced.

It was also decided that both these features be implemented at the same time, as they were both similar and relied on each other to function.

e.g. To unlock the achievement of earning $10,000 in free play tournaments the 'Achievements' class needs access to the 'Statistics' class.

4.3 Data Structures

The data structures used to represent the model of the poker game have an influence on the stability, robustness and efficiency of the system.

One of the most important decisions made was to represent the suit of cards through enumerations instead of strings. This is is because enumerations are more efficient and avoid having to carry out time-consuming string comparisons. The rank of cards is represented as an integer, where 11, 12, 13 and 14 represents Jacks, Queens, Kings and Aces. This also eliminated the need for string comparisons but upon reflection I believe representing ranks through enumerations too would have made implementation a little easier, as the code would have been more readable.

Players have cards stored in an array of size 2. I decided use an array as opposed to an 'ArrayList' as a players number of hole cards is fixed. They are always dealt 2 hole cards. Conversely, I decided that community cards should be represented through an 'ArrayList' because the length can vary between 0, 3, 4 and 5 cards depending on the stage of the hand.

'HandComparers' are constructed with an 'ArrayList' of cards because AI players use the method to check the strongest hand they are capable of making by using 5, 6 or 7 combined cards, so the size is not fixed.

In my experience 'ArrayLists' are more suitable than arrays when sorting needs to be done and arrays are more likely to throw a null pointer exception.

4.4 Testing

Ensuring the system is accurate and behaves as expected is critical to the projects success, In order to determine that the system is was stable, robust, accurate and easy to use I adopted various strategies through the implementation phase:

4.4.1 Unit Testing

JUnit testing was performed on each class where possible. Complexities of various objects including the 'Hand' class made unit testing difficult, so main method testing was conducted throughout implementation and the components fields were JUnit tested where possible.

Therefore, because JUnit testing the 'Hand' class was difficult, I focused on JUnit testing the 'Deck', 'Pot' and 'HandComparer' classes which are all fields of 'Hand' as well as testing the class through its main method.

The method of JUnit testing the 'HandChecker' class is summarised below, full test documentation can be views on the supplied CD.

Hand Checker Testing

With over 2,500,000 unique combinations of poker hands, it was unreasonable to test every possibility. I therefore applied the test strategy of testing edge cases, along with a few typical cases. The results of JUnit tests for straight flushes and straights are shown in Appendix F.

Errors Discovered

The JUnit tests in Appendix F show that tests 4 and 38 failed when ran. Closer inspection revealed a minor bug with the straight checker algorithm.

The error with the algorithm was when cards of the same rank ocurred, because cards were organised by rank the following combination:

Ac, Kc, Ks, Qc, Jh, 10c, 2d

Would never be detected as a straight, because when the loop reaches the first King, the next card is not 1 lower in rank. This error was fixed by removing duplicate ranks from the list when checking for straights.

4.4.2 Integration Testing

After each new class was implemented, integration testing was carried out to ensure that the classes correctly function together. Every time a new player class was developed a game was simulated to test the players behaviour.

4.4.3 Usability Testing

Ensuring that the system is easy to use is especially important for this project. A lot of time and effort went into conducting usability tests with users of different abilities to ensure any annoyances were eliminated.

Users were asked to carry out tasks for each usability test where I wrote down any issues observed with the user interaction and users also explained their thoughts about aspects of the system, including what they would change. An experienced player was asked to focus on the lesson content so that minor changes could be made about parts that were incorrect, misleading or unimportant.

Appendix item F.3 shows the usability test summary for lesson one and complete documentation of usability testing is available on the CD.

All problems discovered through usability testing were fixed before implementation of the next lesson could begin, this ensured that errors could not be carried through and allowed to escalate.

Usability testing identified an issue with sound during games, sound was initially implemented but often caused the system to crash after a random number of hands on the Max OS X operating system. The error was not java specific but the decision was taken to remove sound from the project as it was not a requirement and the robustness of the system could not be guaranteed with the implementation of sound included.

Chapter 5

User Interface

Good use of HCI was essential to the projects success, if users have difficulty interacting with the system they will experience difficulty learning new strategies effectively. To achieve one of the key aims of developing an easy to use solution, elements of the user-centered design process were followed which resulted in the development of Lo-Fi and Hi-Fi prototypes.

5.0.4 Lo-Fi Prototype

The first stage of GUI design was to develop a Lo-Fi paper prototype. A paper prototype is a series of sketches designed to model the basic functionality of the system. As the prototype was made from quick sketches the functionality could be modelled quickly and without using any code. Appendix item G.1 shows the initial paper prototype modelling the lesson center. All other paper prototypes can be viewed from the supplied CD.

Usability testing was conducted with 3 users of differing abilities. Each player was asked to carry out 4 different tasks and their behaviour and thoughts were noted.

Results

Paper prototyping proved to be a useful tool for gathering feedback. The following changes were took forward to the Hi-Fi prototype:

- The registration process should be done through a separate screen.
- Ability levels required for registration should be explained, so that users can determine their ability without uncertainty.
- The layout should be modified so that lessons are above options, this would ensure that lessons are the main focus.
- Buttons to unlocked lessons should be green, whilst buttons to locked lessons should be red.
- An additional raise button should be included for minimum raises.

5.0.5 Hi-Fi Prototype

Proposed changes from the paper prototype were made during the design of the Hi-Fi prototype. The Hi-Fi prototype was designed as a series of slides in PowerPoint which modelled the functionality more realistically than the paper protoype. Users could click on the buttons to navigate to the relevant screen. A problem I experienced with developing the Hi-Fi prototype was the increased time taken to design.

Results

Usability testing of the Hi-Fi prototype had a positive influence on the design of the GUI, the following proposed changes were took forward to implementation:

- Sliders replaced text fields for game setup, increasing the systems robustness and simplicity as the user has a choice between select values.
- A text box was added to the player action bar so that players can input specific raises easier.
- The colouring of buttons received a negative response, buttons were all made the same colour.
- The font was changed as users found it difficult to read.

5.0.6 Final GUI

During the development of the GUI, usability testing was conducted for each lesson as well as for the main functionality. The changes made during the implementation of the GUI were minimal as the Lo-Fi and Hi-Fi prototypes proved useful in identifying the major issues. The following changes were made during implementation:

- Passwords and secret questions/answers were added to the registration process. Passwords were encrypted and functionality was added to reset forgotten passwords.
- A hand strength meter was added.
- The colour of feedback messages was changed from red to white, colour blind users had issues reading red text on the black background.
- The log out button was only made available on the home screen to prevent users accidentally logging out during games.
- The text box used for player raises was more popular than the slider, therefore the slider was removed from the GUI and the two raise buttons merged into one.
- Added keyboard listener so users could type a raise and press enter to carry out the action.

Chapter 6

Project Management & Software Engineering

6.1 Project Lifecycle

For the design of the poker game a waterfall approach was used, as the functionality and rules for playing poker are well defined and unlikely to change. However, components such as the GUI, lessons and statistics required a more iterative approach. An iterative approach was used for these as additional user feedback could be gained during the development cycle to determine additional requirements. During the development of lessons, previous experience could applied to improve the quality of the system [7].

6.2 Timetabling

The project was broken down into workable components and scheduled through timetables. When problems occurred, causing the project to fall behind schedule, revisions were made to the timetables to manage the situation. All timetables are shown in Appendix D.

6.3 Risk Management

During analysis of the project, components of carrying significant risk were identified along with fallback positions. General risks to the project were also identified along with a suitable management strategy.

6.3.1 Data Loss

Data loss is a significant risk with the potential to harm the quality of the finished product. The probability of data loss occurring can be reduced by using anti-virus software and by using subversion to store regular versions of the system on the schools secure server.

6.3.2 Illness

Illness could restrict the amount of time available to spend on the project. The timetable should be rescheduled where short term illness causes the project to fall behind schedule. In the event of a more serious illness, functionality of the system should be reduced.

6.3.3 Project Too Difficult

The risk of the project being too difficult to complete can be reduced by following good software engineering practice to analyse the problem and develop clearly defined requirements. Risk analysis of project components should also be carried out to determine fallback positions to reduce the projects complexity.

6.4 Project Meetings

Over the course of the academic year I kept in regular contact with my supervisor Dr Peter Lewis. Weekly logs were sent detailing the progress made on the project and weekly meetings were also held to discuss issues with the project and to gain recommendations and feedback off Peter on approaches to take, which helped me get the most from the project in the time available.

6.5 Code Reuse

Following good software engineering practice, I reused parts of code where applicable. The following list summarises which parts of the project involved code reuse:

- A 'ColorPane' class was reused which allows coloured text to be appended to a JTextPane. This class was taken from an example on the O'Reilly website [4] .
- 7 lines of code from a Stack Overflow example was used to change the default gradient of JButtons [8].
- The JYAML API provides several lines of code to convent YAML files into java classes and vice versa.
- An example on Java Developer was used to create the skeleton structure to the 'PlayerComparator' class. [9]
- Skeleton code for document, mouse and action listeners was used throughout the project from relevant examples and API's which are referenced in the code. Code was also reused to detect and take action when a user presses the enter key.

- A few lines of code were reused from an example on Stack Overflow which allowed me to convert the mouse position over a JEditorPane into the character index being hovered over. This was used for the implementation of definitions and hints [5].
- A method to encrypt passwords was taken from an example by Dev Bistro, this method was later improved with the addition of salting passwords for added security [10].

6.6 Images Used

Some royalty free images were used with the design of the GUI, the list and source of these images can be found on the provided CD.

Chapter 7

Validation of Aims

At the beginning of the project, I determined 4 aims to help determine how successful the project was. Users with no knowledge of the project were surveyed and the results used to summarise how well each aim was met based on the evaluation criteria set during analysis. The survey given to users can be viewed from the CD supplied.

7.0.1 Involvement Results

- Users were asked to take a lesson through my software, a website and through a video. The results showed that:
  - 75% paid most attention to the software system.
  - 100% of users felt most involved with the software system.
- Comparison with the alternatives proves that my solution required more user input. This was also one of the reasons for why users said they paid most attention to the software.

These results validate that the developed solution is more involving than the alternatives.

7.0.2 Enjoyment Results

- Users were asked to take a lesson through my software, a website and through a video. The results showed that:
  - 100% of users would choose to learn through my software again, because it was the most interactive and they felt they made the most progress through this solution.
  - The statement 'Earning new achievments is fun' scored 87.5%
  - The software solution scored highest when each solution was given a score between 1 and 10 for enjoyment.

These results validate that the developed solution is more enjoyable than alternatives.

7.0.3 Effectiveness Conclusions

- As part of the survery, users were asked to take a tournament before taking any lessons, and after taking 3 lessons. The results showed that:
  - 75% of users improved on their final position after taking 3 lessons, whilst the remaining user came in the same position.
  - The statement 'I made better judgements after completing all of the lessons' scored 85%.
  - The statement 'I successfully applied strategies that I learnt in the free play mode' scored 87.5%.
- The survery also revealed:
  - The statement 'I made better judgements after completing all of the lessons' scored 85%.

These results indicate that the software may be more effective than alternative solutions. To be sure that this aim was successful I believe further research is required. Given the time, I would like to have recorded users statistics at different stages for further analysis. Users also felt that explanations should be given after each question is answered correctly, just to be sure that they understood the logic behind the question.

7.0.4 Ease of Use Conclusions

3 out of 4 of the ease of use aim's were met. Regular usability testing occured and both Lo-Fi and Hi-Fi prototypes were created to determine the end design of the GUI. Survey results showed that the statement 'I found the system easy to navigage' scored 82.5%.

Personas were not created to emphasise with users because regular contact with users was made during usability testing anyway.

I feel that these results indicate that the system is easy to use, but room for improvement still exists. To improve ease of use in future versions I would like to implement a help center explaining the main functionalities of the system.

Chapter 8

Appraisal

It is my belief that the project was largely successful. At the beginning my plan was to develop a software solution that could teach users new strategies and the final solution surpassed original expectations. Although I feel that the project could have been improved by using the structure of the MVC model, the complexity of the project made this difficult given the time constraints and the quality of the project has not suffered without this implementation.

8.1 Project Approach

I feel that the project was approached well, I managed the project effectively by rescheduling the timetable when problems occurred and I took the workloads of other modules into account so that I could balance time. I used proven philosophies such as the user-centered design approach to develop the GUI and remained in constant contact with users, asking them for feedback as each new component was implemented. I took decisive action when I felt system features could be implemented better, such as the decision to use YAML instead of SQL.

8.2 Comparison to Specification

Nearly all of the requirements documented in the user specification were met, the requirement of using a database was not met because it was felt that YAML was a more suitable solution. I believe this decision was justified and YAML is capable of storing all of the data described in the requirements document without any precautions needed, such as prepared statements. The following requirements:

- 'The system must require users to specify their experience level.'
- 'The system must require completion of an introductory lesson before completing registration.'
- 'The system must unlock a new lesson when users successfully complete a previous one.'

Were not met because usability testing later revealed that they caused confusion or restricted the user experience. Implementation of these requirements would have been detrimental to the system.

8.3 Quality of Product

The approaches took during projects lifecycle helped create a high quality product. JUnit and usability testing was used to ensure that the system is reliable and robust. Once the AI player had been developed, I was able to quickly simulate games with up to 8 players, to test that games always ended and that the pots were always correctly calculated.

The performance of the product is also favourable. All performance requirements were met, players even made decisions much quicker than anticipated so the thread needs to be slept for seconds to give the impression that the player is thinking what decision to make.

I believe the efficiency of the product is good overall, but minor improvements could be made. The algorithm to determine whether a player has made an action yet is called 3 times a second. This could be improved by setting this algorithm to wait until notified, this is only minor however and the current implementation is not an issue. Repainting of the screen is done efficiently as most of my methods specify a rectangle to paint where possible, preventing processing power from being wasted in painting the whole frame when unnecessary.

8.4 Problems

As explained in the implementation chapter, problems occurred during the project which I had not envisaged. The development of lessons took a lot longer than expected to get right but once the first lesson had been implemented the time to complete the others was reduced. I estimate that future lessons would take just under a week to successfully implement and test.

Chapter 9

Conclusions

This chapter summarises the achievements and limitations of the project. Future opportunities to improve the functionality and the experiences gained are discussed.

9.1 Achievements

The system achieves what it was designed to do. It teaches users how to play poker with 6 lessons lasting between 5-10 minutes. It allows players to play against difficult to beat AI players through a customisable free play mode and statistics as well as achievements are recorded and detected. I was also able to implement additional functionality such as animated achievements and a hand strength meter. User feedback shows a preference to learning through this system than with the alternatives.

9.2 Limitations

I do believe that some aspects of the system are limited and could be improved. During evaluation I learnt that even more feedback should be given to users taking tests, the lessons could be improved by explaining the logic behind each test once answered correctly. I would also implement a help center as suggested from user feedback, so that important aspects of the system such as definitions and hints could be explained to users.

9.3 Future Development

The project has a lot of potential for future development, some of the most interesting possibilities include:

- **Online Functionality** - An opportunity exists to implement the system through Facebook. Allowing users to play against each other and see how they rank amongst friends. This implementation would also simplify the registration process as users could log in through their Facebook credentials.

- **Lesson Creator** - As part of online functionality, users could create and share their own lessons. The best ranked lessons could be downloaded and professional players could even create and sell lessons through a lesson store.
- **Increased Freeplay Functionality** - The freeplay option currently allows sit and go tournaments to be played where players remain in the game until knocked out. Other options exist, In the future cash games could be implemented where a user has the option to leave at any time with their current balance and there is even the possibility to expand the system to cater for other variants of poker.
- **Improvement Monitoring** - Users statistics could be recorded each time a lesson is taken, their improvement could be shown visually through graphs showing how their ability has improved after taking lessons.

9.4 Experience Gained

The project gave me a lot of experience in management. Although I felt the project was timetabled well I found myself needing to work harder in the second term. I learnt to prioritise specific components of the project and to analyse alternative options before jumping to conclusions. I feel my programming skills have improved further by developing a substantial project and I even learnt a new language in YAML.

9.5 Summary

To summarise, I am happy with the quality of the project and proud that the many hours of time and effort have paid off with the final result. I feel that this project reflects well on what I have learnt throughout the degree. I found the project challenging and had to overcome numerous problems, but it was equally rewarding.

Bibliography

[1] Cards Chat. History of Poker and Texas Hold'em. Available: http://www.cardschat.com/poker/history/. Last accessed 1st April 2012.

[2] Cypra, D. (2010). PokerStars Deals 40 Billionth Online Poker Hand. Available: http://www.pokernewsdaily.com/pokerstars-deals-40-billionth-online-poker-hand-8482/. Last accessed 1st April 2012.

[3] (2012). Biography and Career Highlights. Available: http://www.chrismoneymaker.com/wiki/. Last accessed 01/04/2012.

[4] Loy, M. Eckstein, R. Wood, D. Elliot, J. Cole, B.. (2002). ColorPane. Available: http://examples.oreilly.com/jswing2/code/ch22/ColorPane.java. Last accessed 3rd April 2012.

[5] (2009). How to convert from a mouse position to a character position in a JEditorPane. Available: http://stackoverflow.com/questions/1180390/how-to-convert-from-a-mouse-position-to-a-character-position-in-a-jeditorpane-in. Last accessed 4th April 2012.

[6] (2011). Poker Position Explained. Available: http://uspokerwebsite.com/poker-position-explained/. Last accessed 5th April 2012.

[7] (2011). Iterative Design. Available: http://en.wikipedia.org/wiki/Iterative design. Last accessed 8th April 2012.

[8] (2010). Repaint swing button with different gradient.. Available: http://stackoverflow.com/questions/4458982/repaint-swing-button-with-different-gradient. Last accessed 8th April 2012.

[9] (2010). Java Comparator Example. Available: http://www.javadeveloper.co.in/java-example/java-comparator-example.html. Last accessed 6th April 2012.

[10] Shvarts, J. Password encryption: rationale and Java example. Available: http://www.devbistro.com/articles/Java/Password-Encryption. Last accessed 10th March 2012.

Appendix A

CD Data Structure & Instructions

A.1 Data Structure

- Documentation
  - Analysis - Contains project proposal, user specification and estimated class structure.
  - Design Screenshots - All screenshots of Lo-Fi and Hi-Fi prototypes.
  - Implementation - Further explanation of algorithms.
  - Lesson Plans - Shows how each lesson was researched prior to implementation.
  - Survey - Survey given to users and scanned results.
  - Testing - JUnit test tables.
  - Timetables - Revised timetables.
  - Usability Tests - Results from usability testing.
- Project - This contains the NetBeans project root.
- Required Files - Netbeans may require the JYAML .jar file.

A.2 Running the Software

- Copy the 'Project' and 'Required Files' folders from the CD onto computer.
- Open NetBeans and click File -> Open Project, Navigate to the copied 'Project' folder and open the project.
- Locate jyaml-1.3.jar to resolve reference problems by navigating to the 'Required Files' folder.
- Set the working directory to be the Data directory of the project.
  e.g. /Users/Anton/NetBeansProjects/FinalYearProject/FinalYearProject/src/Data
- The project can be ran by right-clicking on RunGUI.java and clicking Run.
- You can register for a new account or log in with the following credentials:
  Username: Anton1990
  Password: password

Appendix B

Glossary

All In - A player goes all in if they bet their remaining balance.

Big Blind - The player who places the forced bet, usually two places left of the dealer.

Bluff - A bluff is when a player bets to give the impression that they have strong cards when they don't. A bluff is successful if all other players fold and the pot is won by the player who bluffed.

Call - A player calls if they match the required bet to stay in the hand.

Connected Cards - This is when a player can combine their hole cards with 3 other cards to form a straight.

Dealer - This is the player who deals the cards and acts last.

Flop - This is the stage of betting after the pre-flop, 3 community (shared) cards are revealed.

Fold - A player folds when they choose not to match the required bet to stay in the hand, or if they no longer wish to remain in the hand.

Hole Cards - These are the two cards dealt to you face down each hand.

Kicker - A kicker is used to determine winners. e.g. If two players have the same ranked pair, the player with the next highest card (the kicker) wins the pot.

Pocket Pair - This is when both of a players hole cards are the same rank.

Position - This refers to the position a player is sat with respect to the dealer. The dealer is in the best position because they have the most information available to them before they have to act.

Post Flop - This refers to any stage after the pre-flop.

Pre-flop - This is the stage of betting before any community cards are turned.

River - This is the stage of betting after the turn, 1 community card is revealed.

Slow Play - This is when a player knows they have the best cards and chooses not to raise in the hope of earning more through other players raises.

Small Blind - This is the player who places a forced bet half the size of the big blind, once place left of the dealer.

Suited Cards - This is when both of a players hole cards are the same suit.

Tight Play - This is when players goes into a hand with strong hole cards, where the lowest card is of rank 10 or above.

Turn - This is the stage of betting after the flop, 1 community card is revealed.

Appendix C

Texas Hold'em Background

The rules of playing Texas Hold'em poker are explained here, or alternatively you could run the software and take Lesson 1.

In Texas Hold'em players can enter tournaments with an equal amount of chips, representing the amount of money they have. The aim of the game is to acquire all of the chips on the table, at which point the game is over and that player becomes the winner.

The game is made up of numerous hands where chips are exchanged between players until the game is over.

A hand begins with a dealer (otherwise known as the button) who traditionally deals the cards, although this is automatically done in software implementations of Poker. The dealer plays an important part in determining how players should act.

Before the cards are dealt, the two players to the left of the dealer are required to post a forced bet known as a blind. The player two places to the left (clockwise) of the dealer posts the big blind while the player one place to the left of the dealer posts the small blind, which is usually half the value of the big blind. This prevents players from folding every hand which would lead to never ending games.

The deal then begins, each player is dealt 2 cards face down, known as their hole cards. Which only they are allowed to look at. The action begins with the player to the left of the big blind and ends with the big blind. In any following round of betting the action begins with the player to the left of the dealer.

Each player has 4 potential actions to take:

- **Check** - If a player has already matched the current bet they can check, meaning they bet $0 and the action continues to the next player.
- **Call** - If a player has not matched the current bet, they can match the current bet in order to stay in the hand.
- **Fold** - If a player does not continue in a hand they can fold, they give up their hole cards without showing them and are inelligible to win anything.
- **Raise** - If a player wants to bet more than the required amount they can do so, this is known

as a raise. Raises must be double the amount of any previous raise or at least the size of the big blind if the amount to bet is $0.

Note that as well as these actions players can also go all in at any time, in doing so they bet the entirety of their chip stack and play no further part in subsequent rounds of betting. Also when a player doesn't have their bet matched and there are no players who have gone all in they win the pot without having to show their hole cards.

If all bets are successfully matched the game proceeds to the flop stage, where 3 community cards are dealt face up on the table, players decisions are based on their confidence of winning based on the hand they can make with these cards, combined with their hole cards. All stages end when all bets are matched.

If bets were successfully matched the game proceeds to the 'turn' and the 'river' stages which are similar to the flop stage apart from there being only one card turned face up on the table.

If there is more than one player left in the game after the river, players show their cards and the winner or winners are determined based on who has the strongest hand. Hands are detailed below:

Hand Name	**Hand Description**
Straight Flush	5 cards of the same suit that all succeed each other.
Four of a Kind	4 cards of the same rank + the next highest card.
Full House	3 cards of the same rank + 2 different cards of the same rank.
Flush	5 cards of the same suit.
Straight	5 cards which all succeed each other.
Three of a Kind	3 cards of the same rank + the next 2 highest.
Two Pair	2 different sets of pairs + the next highest card.
Pair	2 cards of the same rank + the next three highest.
High Card	5 highest cards.

Table C.1: Table illustrating hand categories ordered by strongest first.

Appendix D

Timetables

Activity	**Completion Date**
Design of class structure	12/10/11
Implementation of poker hand recognition	14/10/11
Testing of poker hand recognition	16/10/11
Implementation of basic AI player	19/10/11
Testing of AI player	20/10/11
Implementation of command line playable game	27/10/11
Structure for logging statistics planned	30/10/11
Testing of game and scenarios	02/11/11
Design paper prototypes	04/11/11
Usability testing of prototypes	08/11/11
Design 2-3 Hi-Fi prototypes and evaluate the best one	16/11/11
Implementation of final GUI for poker play	21/11/11
Usability testing of GUI	23/11/11
First lesson planned	25/11/11
First lesson implemented	01/12/11
First lesson usability testing	03/12/11
Prototype with first lesson and play against AI player	03/12/11
Inspection preperation	04/12/11
Inspection	05/12/11
User profile and achievement system design	08/12/11
Implementation of user profile and achievement system	20/01/12
Testing of user profile and achievement system	23/01/12
New lesson planned	Every 7 days from early January
New lesson implemented	Every 7-10 days from early January
New lesson usability testing	Every 2 days after implementation
Implement achievement recognition	12/02/12
Test recognition accuracy	15/02/12
Implementation of additional AI players	18/02/12
AI player testing	20/02/12
Implementation of statistical imformation	30/02/12
Testing of statistical information	01/03/12
Final testing and bug fixes	06/03/12
Final release	14/03/12
Demonstration preperation	18/03/12
Demonstration	19/03/12

Table D.1: Initial Timetable

Activity	**Completion Date**
Implementation of command line playable game	05/11/11
Design paper prototypes	11/11/11
Testing of game and scenarios	15/11/11
Usability testing of prototypes	16/11/11
Design 1 Hi-Fi prototype	18/11/11
Implementation of final GUI for poker play	22/11/11
Usability testing of GUI	23/11/11
First lesson planned	25/11/11
First lesson implemented	01/12/11
First lesson usability testing	03/12/11
Prototype with first lesson and play against AI player	03/12/11
Inspection preperation	04/12/11
Inspection	05/12/11
User profile and achievement system design	08/12/11
Implementation of user profile and achievement system	20/01/12
Testing of user profile and achievement system	23/01/12
New lesson planned	Every 7 days from early January
New lesson implemented	Every 7-10 days from early January
New lesson usability testing	Every 2 days after implementation
Implement achievement recognition	12/02/12
Test recognition accuracy	15/02/12
Implementation of additional AI players	18/02/12
AI player testing	20/02/12
Implementation of statistical imformation	30/02/12
Testing of statistical information	01/03/12
Final testing and bug fixes	06/03/12
Final release	14/03/12
Demonstration preperation	18/03/12
Demonstration	19/03/12

Table D.2: Revised Timetable 1

Activity	**Completion Date**
Second lesson implemented	03/02/12
Second lesson usability testing	04/02/12
Third lesson planned	04/02/12
Third lesson implemented	09/02/12
Third lesson usability testing	10/02/12
Fourth lesson planned	10/02/12
Fourth lesson implemented	14/02/12
Fourth lesson usability testing	15/02/12
User profile and achievement system design	16/02/12
Implementation of user profile and achievement system	19/02/12
Implement achievement recognition	19/02/12
Testing of user profile and achievement system	21/02/12
Test recognition accuracy	23/02/12
Implementation of additional AI players	25/02/12
AI player testing	26/02/12
Implementation of statistical imformation	30/02/12
Testing of statistical information	01/03/12
Final testing and bug fixes	06/03/12
Final release	14/03/12
Demonstration preperation	18/03/12
Demonstration	19/03/12

Table D.3: Revised Timetable 2

Appendix E

Lesson Plan

Interactive Teaching of Poker Strategies

Lessons 1 Plan – Introduction to Poker

Lesson 1 needs to give users an introduction and refresher into the rules of the game. It should also provide a walkthrough of various hands where there is uncertainty into how the hand is dealt with.

Lesson aims from Requirements Document

- To explain a typical hand.
- Explain big and small blinds.
- Explain the different categories of hands.

I have also decided that it should meet the following aim:

- Test the user's ability to call, fold, check and raise.

Structure

After researching the introductions from various videos and websites, I was able to determine the structure of the lesson and the key points to be mentioned.

Rules

• The dealer chip rotates around the table.
• Blinds prevent players from excessively folding.
• Players not participating have no chance of winning because their chip amount will diminish over time from posting blinds.
• Blinds control the length of games; they are regularly doubled to force players to bet more. This ensures that the blinds take up a larger percentage of a players balance and make it more likely a player goes all in and gets knocked out of the game.
• The small and big blinds are to the left of the dealer.
• At the end of each round the bets are placed into the pot.
• Each player is dealt two cards, known as hole cards. Only they can see these.
• Pre-flop Betting begins with the player to the left of the big blind and finishes on the big blind. The big blind has the option to check or raise if his blind has been matched. If he raises the action goes around again until all bets are matched.
• At every stage action continues until bets are matched.
• Players who go all in may not have met the amount to call, if other players have matched a higher amount then the 'all in' Player will not be eligible to win this additional amount. If there are only two players then the player who bet more will get the difference refunded.
• The next stages after pre-flop are the flop, turn and river. Betting happens at each stage and the small blind gets first choice on what action to take, they may check, bet or fold. If the small blind isn't in the hand then it goes to the next available player to the small blinds left.
• 3 Cards are revealed during the flop and 1 card during both the turn and the river. The players involved will know their best 5 cards at the river.
• If a players bet isn't matched at any stage they win the pot and don't have to show their cards.
• If bets have been matched at the river we go to the showdown stage where players reveal their cards and the winning player/s receive their share of the pot.
• A new hand begins after each hand ends until a winner is decided.

Hand Categories

Here the different hand categories should be explained, along with terms such as 'kickers' and 'high cards'.

- Straight Flush
- Four of a Kind
- Full House
- Flush
- Straight
- Three of a Kind
- Two Pair
- Pair
- High Card

Edge Cases

Simulations of the following scenarios should be developed, as well as teaching the player what happens it will allow us to test that the game performs correctly under these situations:

- Lesson should show what happens when a player/s go all in.
- Lesson should show what happens when a player/s go all in posting the blind.
- Lesson should show what happens with a 'dead' blind.
- Lesson should simulate a player winning by high card, as determined by the suit of their hole cards.

Sources Used

- http://www.pkr.com/en/raise-your-game/quick-start/
- http://en.wikipedia.org/wiki/List_of_poker_hands
- http://en.wikipedia.org/wiki/Texas_hold_%27em

Appendix F

Testing

Straight Flush			
Test ID	**Cards**	**Description**	**Result**
Edge Cases			
1	10♠ J♠ Q♠ K♠ A♠ 7♣ 2♦	Royal Flush, Best Hand in the Game.	Passed
2	A♠ 2♠ 3♠ 4♠ 5♠ 7♣ 2♦	Lowest Possible Straight Flush	Passed
3	A♠ A♥ 3♠ 4♠ 5♠ A♣ 2♠	Straight Flush with Three of a Kind	Passed
4	5♠ 6♠ 7♦ 8♠ 9♠ 7♣ 7♠	Straight with Three of a Kind	Failed
5	5♥ 6♥ 7♥ 8♥ 9♥ 7♣ 7♠	Straight Flush with Three of a Kind	Passed
6	3♦ 4♦ 5♦ 6♦ 7♦ 8♦ 9♦	7 Card Straight Flush with Three of a Kind	Passed
Typical Cases			
7	6♥ 2♥ 4♥ 3♥ 9♣ 7♠ 5♥	Mid Strength Straight Flush	Passed
8	8♦ 6♦ 1♠ 7♦ 5♦ 7♣ 4♦	Mid Strength Straight Flush	Passed
9	A♣ 7♠ 3♠ 5♠ 1♥ 6♠ 2♠	Flush But Not Straight Flush	Passed
10	5♠ 6♠ 7♦ 8♠ 9♠ 7♣ 7♥	Straight	Passed

Figure F.1: JUnit test results for Straight Flush

Straight			
Test ID	**Cards**	**Description**	**Result**
Edge Cases			
34	A♦ K♦ Q♦ J♦ 10♦ 9♦ 8♦	Straight Flush	Passed
35	A♥ K♣ Q♣ J♦ 10♠ 9♣ 8♥	Ace High Straight (7 Cards)	Passed
36	A♥ K♣ Q♣ J♦ 10♠ 2♣ 3♥	Ace High Straight (5 Cards)	Passed
Typical Cases			
37	A♥ 2♣ 3♣ 4♦ 5♠ 9♣ 8♥	5 High Straight	Passed
38	3♣ 4♥ 4♦ 4♥ 5♥ 6♣ 7♥	Mid Level Straight	Failed
39	9♥ 7♣ 8♦ 5♥ 6♠ A♣ 2♥	Mid Level Straight	Passed

Figure F.2: JUnit test results for Straight

Interactive Teaching of Poker Strategies

Lessons 1 Usability Test

After successful implementation of lesson 1, feedback was crucial to allow me to evaluate my aims of the system being easy to use and enjoyable. I asked 3 different users to take lesson one. Without any further help I monitored user's behaviour and at different points I would ask users what they were thinking.

The 3 users all had differing levels of knowledge with poker, one had never played the game before, one had a basic understanding of the rules and the final player was experienced.

The inexperienced players were useful at finding uncertainties that an experienced player may not have noticed and the experienced player was asked to focus on the content and scenarios to ensure the lesson was accurate.

Notes

- User attempted resizing of JFrame which increased the size of the frame but didn't draw the panel any larger. This was a bug, JFrame shouldn't be resizable.
- User unaware that they needed to click "Start" to run the scenario, suggested the text on the right panel prompting the user to begin the lesson.
- When the user was prompted to click "Start" the click was not being registered.
- I observed that the final scenario did not display the Start buttons, meaning it wasn't playable.

Actions Taken

- JFrame set to fixed size.
- Text prompt added to the content.
- Button click wasn't registering because the TextPane was covering the 'Start' button; reduction in the Y size fixed this problem.
- Further investigation showed a problem with the method to determine if a lesson snippet had a runnable scenario. "StartEnd" is used to indicate that game can be started but also that it was the end of the lesson. By changing the method to contains instead of matches this problem was fixed.

Content Changes

- Players believed that an overview should be given for the lesson so that they will know what they can expect to learn.
- Experienced player suggested some explanations be made clearer; some were misleading or too wordy.
- Experience player recommended using 'community cards' instead of 'board cards' as this was correct terminology, other terminologies were also recommended.
- Experience player also recommended that the tests be made simple, felt it was important not to make the first lesson too difficult.

Appendix G

Development of GUI

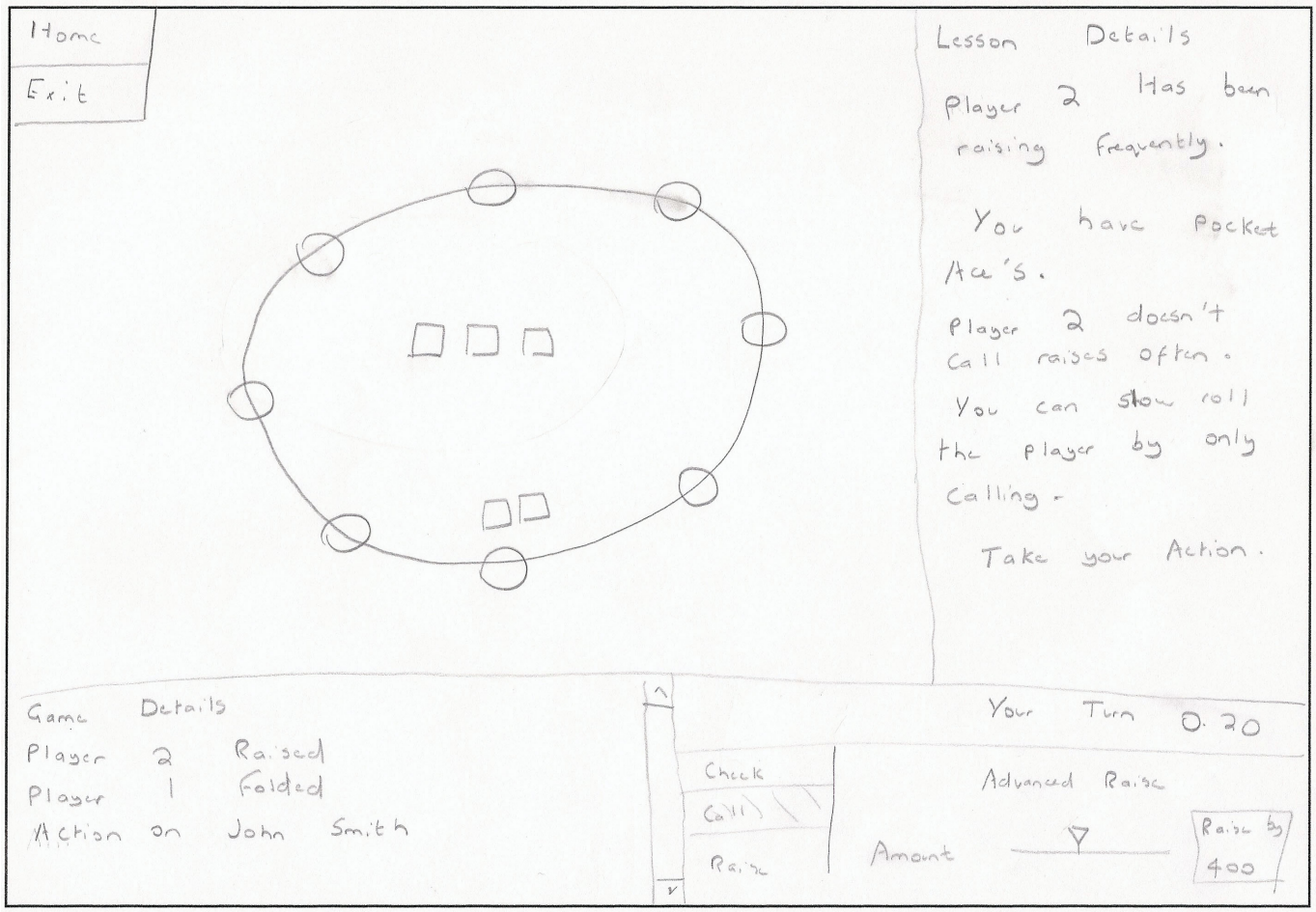

Figure G.1: Lo-Fi design of lesson system.

Figure G.2: Hi-Fi design of lesson system.

Figure G.3: Final design of lesson system, a definition can also be seen.