School of Electronic Engineering and Computer Science

Final Report

Business Computing with Industrial Experience

Hold'em Poker Guide

a mobile application for Android

Student:
Anthony Schafer

Supervisor:
Professor Norman Fenton

Queen Mary
University of London

Submission date: 22/04/2013

Abstract:

There is a decent amount of poker game applications on the Android market. Some of those applications are designed for online multiplayer mode with real opponents. Others offer offline educational features without a functional game. To my knowledge, none of them offers an offline version of a fully functional game and an integrated set of learning features. Furthermore, many of the available applications do not have an optimised UI for tablet devices.

The objective is to develop a fully functional offline poker game with a series of learning features, which has an optimised UI for smartphones and tables. It is a mobile application which allows users to learn the rules, play the game and receive visual and textual decision support. It serves as a learning, entertainment and training tool for new players and players with little poker experience.

Although there are professional applications available for Windows OS, which have most of the implemented functionalities, this project focuses on Android OS and targets the digital distribution platform Google Play. Furthermore, the acquisition and evaluation of such professional applications is beyond the scope of this project.

The released application fulfils the main objective. It provides a fully functional offline poker game with a number of learning features. The application runs on Android devices with an operating system version 2.3.5 and higher. The UI was developed for a high definition resolution and is compatible with the majority of smartphone and tablet screen sizes.

Disclaimer

I understand the nature of plagiarism, and I am aware of the University's policy on this.

I certify that this paper reports original work by me during my University project except where indicated in the text and for the following technical aspects:

- One of the implemented AI strategies is based on the Sklansky & Malmuth starting hands table [1].
- The application framework used to develop this application is LibGDX [2].
- The classes *Card*, *Hand*, *Deck* and *HandEvaluator* were adapted from University of Alberta Computer Poker Research Group and modified as necessary for this project [3] [4].

Contents

Figures

Tables

Chapter 1 : Introduction

I remember being in a situation where I was travelling and decided to play Limit Texas Hold'em poker on my smartphone. As a new player, I was not able to understand the basic rules straightaway. The application did not provide a tutorial or any kind of learning features. I had to leave the game and look for other applications with informative features. Though that I found many tutorials on the web, I had no success in finding an Android application which offers access to educational features and a fully functional poker game in parallel. It got worse as my mobile device lost the signal. There was no way to access poker rules in the offline state. This led me to the idea of developing an offline application which allows users to play poker and have access to poker rules at the same time. This idea was extended with integration of additional educational features, which provide decision support to the player.

The finalised idea for my project is to design and implement an application for Android. The application should be a mobile version of the Limit Texas Hold'em (LTH) poker game. It should include educational features that help users to learn the game by playing it. No internet connection is required to run the application. The targeted users are the ones with no or little poker experience.

Previous to this project I have not developed any applications for the Android platform and have little experience in playing LTH poker. The challenge is to acquire both of those skills and apply them in order to achieve the objectives of this project.

1.1. Objectives

The high level objectives of this project can be summarised into 4 key components:

- Fully functional poker game application
- No need for internet access (AI development and implementation)
- Integrated Educational features
- Optimised UI for smartphones and tables

1.2. Scope

There are paid applications available for Windows OS, which have most of the implemented functionalities. However, this project focuses on Android OS and targets the digital distribution platform Google Play. In general, the acquisition and examination of non-free and professional applications is beyond the scope of this project.

At the time of state-of-the-art analysis, I could not identify any applications for Android which include the entire set of objectives and features defined in the requirements specification section of this report. However, considering the fact that Android market returns 6646 results when the keyword *Poker* is entered, it is clear that not all applications can be reviewed. Due to time limitations I have made the decision to look through the first 50 applications and choose 5 most popular poker game applications and 5 most popular educational poker applications. The selection criteria for chosen applications were rating score and number of active users. All data was retrieved from Google Play.

Poker has many variations, and those variations also have sub variations. Each of them has a different playing style and betting structure. Due to time restriction for this project, I decided to implement one particular sub variation. My choice is Limit Texas Hold'em. It is one of the most popular versions and has straightforward rules. The educational features of the game limit the application to be an offline (standalone) application. The opponents are controlled by a build in AI module. Unfair play against real players is avoided.

1.3. Overview

This report describes the state-of-the-art, the addressed problem, resolution methodologies including the description of features combination that allows this application to stand out. It also covers the high and low level design decisions of the actual implementation. Application screen shots are included where appropriate. The content of this report assumes that the reader has no knowledge about Android OS or Poker.

Chapter 2 : Research

In order to gather detailed requirements for my project and overcome the initial challenges, I have carried out an in depth research and analysed the literature about Android application development and LTH poker rules.

2.1. About Android

This subsection explains what Android is, describes the system Architecture, reflects on the digital distribution platform Google Play and shows the concluded technical details for my project.

2.1.1. What is Android?

Android is a Linux-based operating system primarily designed for mobile touchscreen handsets. The platform was created by Android Inc., a small software company from Palo Alto (California), which specialised in developing software for mobile phones. In 2005, Google acquired the company and released the platform as the Android Open Source Project (AOSP) [5].

Android is currently developed by Google in conjunction with the Open Handset Alliance. In the second quarter 2009, Android had a 2.8% share of global smartphone shipments [6]. By the third quarter of 2012 Android had a 75% share of the global smartphone market [7].

2.1.2. Android OS architecture

Figure 2-1 shows that Android OS can be subdivided into five layers. Green items are written in C/C++, whereas blue items are written in Java and run in the Dalvik VM [8].

- **Applications layer** contains all the native applications that Android comes with. Those are usually an e-mail, SMS, calendar, maps, contacts and browser application. Any installed third party application will be in this layer. Each application can use the services provided by the application framework layer [9].

- **Application framework layer** gives a fixed architecture which every developed application has to comply. The aim is to make sure that all applications are developed in accordance with certain guidelines. An example would be when a

background application wants to notify users about certain events, it has to pass the message to the Notification Manager, which will handle the message and display it to the user in a predefined format [9].

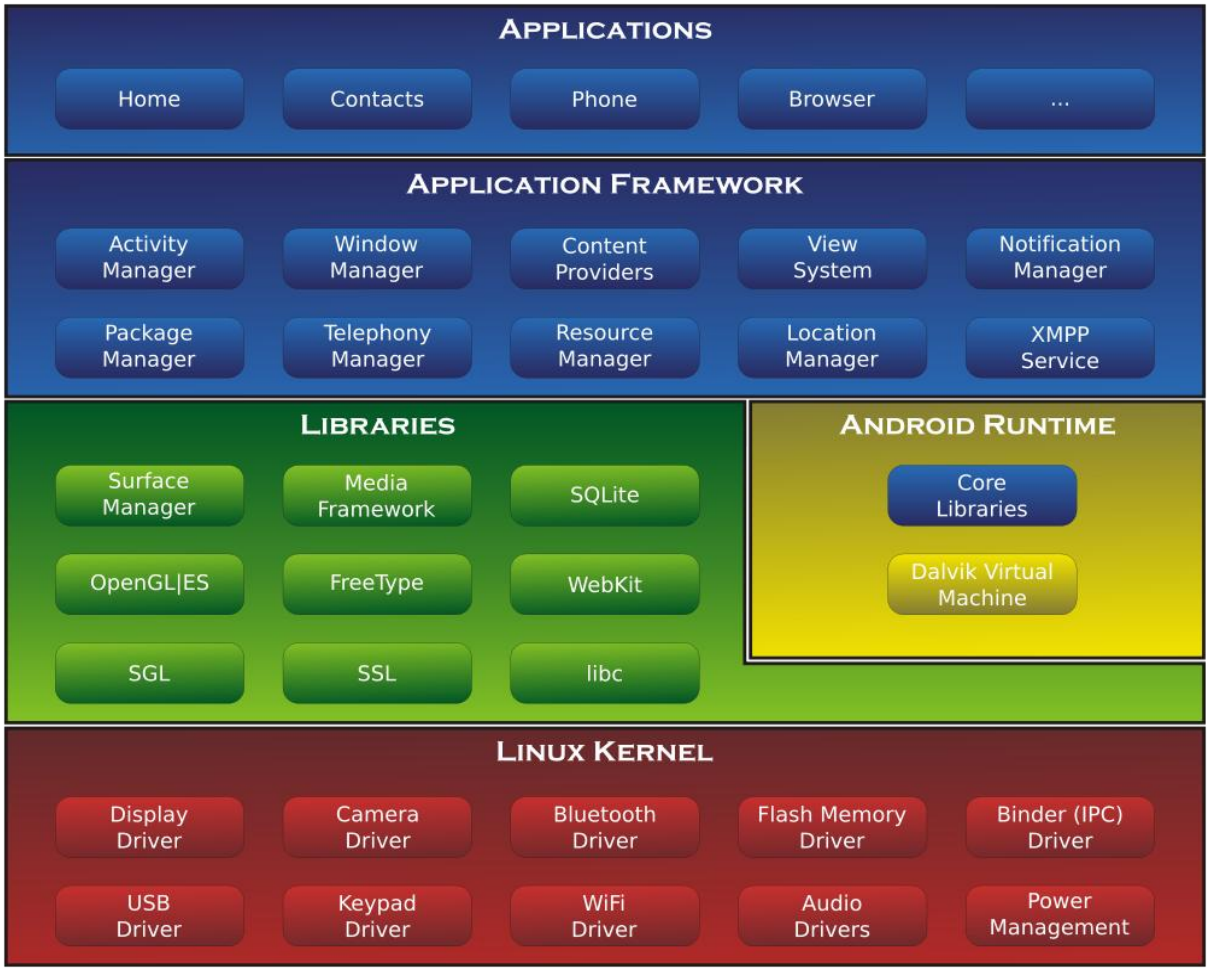


Figure 2-1: Android OS architecture [8].

- **Android Runtime Environment** consists of Dalvik virtual machine and java core libraries. Dalvik virtual machine is a customised Java VM. Every single application within the blue coloured layers runs in its separate Dalvik VM. Therefore, several VMs are running simultaneously on an Android OS. Dalvik VM uses the Java core libraries, which represent a light weight version of Java SDK libraries [9].

- **Native Libraries** are a set of C/C++ Libraries which are used by several operating system components. Furthermore, these libraries are available to application developers through the application framework. Those are also used by the Dalvik VM [9].

- **Linux Kernel:** consists of a modified Linux 2.6 series kernel and includes the low level drivers for hardware components like USB ports or Audio Drivers [9].

2.1.3. Android versions

Android 1.0, the first commercial version of the software, was released on 23rd of September 2008 [10]. The latest version, Android 4.2 (Jelly Bean), was released on 13th of November 2012 [11]. Figure 2-2 was taken from the official android developer website and shows the current distribution of Android versions on the market [12].

Version	Codename	API	Distribution
1.6	Donut	4	0.1%
2.1	Eclair	7	1.7%
2.2	Froyo	8	4.0%
2.3 - 2.3.2	Gingerbread	9	0.1%
2.3.3 - 2.3.7		10	39.7%
3.2	Honeycomb	13	0.2%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	29.3%
4.1.x	Jelly Bean	16	23.0%
4.2.x		17	2.0%

Data collected during a 14-day period ending on April 2, 2013.
Any versions with less than 0.1% distribution are not shown.

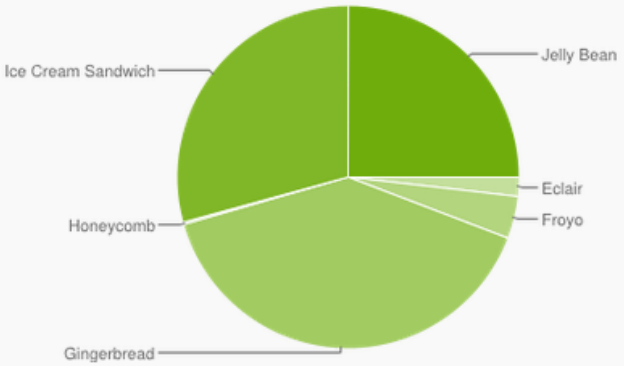


Figure 2-2: Android versions distribution on the smartphone market 04/04/2012

As we can see the, majority of devices are still running on Android 2.3 (Gingerbread). This is due to most low-end devices still being released with it. These statistics are collected every 14 days and based on devices accessing Google Play.

2.1.4. What is Google Play?

Google Play was formerly known as the Android Market. It is a digital application distribution platform for Android, which is developed and maintained by Google. The service allows users to browse and download digital content that was published through Google. In October 2012, Google announced that Google Play had 700.000 applications available to download [13].

2.1.5. Android applications

Android applications are written in Java. Each application is executed in its own sandboxed virtual machine. This means that each application is a separate process. With each Android version, Google releases a number of native applications. However, there is a large community of Android application developers who are writing third party applications to extend the functionality of Android devices. Android allows applications to use the hardware features through abstraction and enables a defined environment, the application framework layer.

2.1.6. Android as an open platform

The meaning of an open platform is that it is not tied to one hardware manufacturer. This allows Android to gain market share quickly. The source code for Android is freely available so all hardware manufacturers can make and sell Android devices. The openness of Android also introduces cross-compatibility. This means that an application developed for one device can be used on another device as soon as the target device is certified as being compatible [14]. For example, if an application requires a camera to work, it will be only available for devices which are certified to have a camera.

2.1.7. Android application development

Android provides access to services. This means that every application can use device hardware features like the camera, compass, GPS feature or Bluetooth transceiver. To enable an easy application development, it also provides several APIs, libraries and the application framework. Application development in Android is based on two main programming components. All logic is written in a customised version of Java, whereas the user interface layouts are defined through XML. The official and recommended IDE for Android development is Eclipse combined with the Android Development Tools (ADT) plugin [15].

2.1.8. Concluded technical project details

Due to a variety of available Android platform versions and Android device screen sizes I have decided on a certain range of devices that my application will support. Each platform version has its own set of functionalities. A newer platform includes the most features of

older platforms, but also has its own additional features and runs more stable. Figure 2-2 shows a whole range of available Android platform versions. It is clear that a certain range of versions has to be selected. The starting point for my decision would be the available testing devices. I own two Android powered devices described in Table 2-I.

Table 2-I: Available testing devices.

Device name	Android version	Screen properties
HTC Desire S	2.3.5	3.7" - 800 x 480 pixels
Google Nexus 7	4.2	7" – 1280 x 800 pixels

Considering those two Android versions as min and max boundaries according to Figure 2-2 my application will support 94.2% of all active devices. I think this is a good coverage.

2.2. Poker

I have chosen the website PokerStars.com as a starting point for my research about Poker. It is the largest online poker card-room in the world with nearly 50 million registered users. Though this is a privately held company, it is organising and sponsoring large scale poker tournaments all around the Europe and the UK. I think that it is a recognised and extremely reliable source of information regarding poker rules. The following sections describe the basic poker rules and the game play structure.

2.2.1. About poker in general

"Poker is the name given to a number of card games where players wager on the strength of the cards they hold. Poker is a game that involves a communal 'pot' consisting of the players' wagers, which is awarded to the player who either holds the highest ranking hand when all the cards are shown, or makes a wager which their opponents are unwilling to match." [16]

2.2.2. Limit Texas Hold'em (LTH)

Texas Hold'em is a variation of poker. There are different versions of Texas Hold'em. Each version has a fixed betting structure. My application is about Limit Texas Hold'em (LTH). The keyword *Limit* means that players' bets are limited to a certain value. LTH is one of the most popular versions of poker in the world. The special thing about LTH is that it is appealing to novice players. It takes a little time to learn the basic rules. The objective of the game is to

make the best 5 card combination out of 7 available cards and bet money that your combination will beat combinations of all other players sitting at the table. Each combination has a predefined rank. Combination ranks are based on ranks and suits of cards contained within a combination. The game is set up as shown in Table 2-II.

Table 2-II: LTH Poker game settings.

Players	2 - 10
Skill required	Probability, Psychology
Cards in deck	52
Play direction	Clockwise
Card rank (high to low)	A K Q J 10 9 8 7 6 5 4 3 2
Suit rank	All suits are valued equally

PokerStars.com provides a free glossary of poker specific terms[1]. In the following section, I will try to explain the key terms in my own words. For further information, please refer to the footnote.

In short, a complete game consists of several poker hands. In every hand, each player receives 2 *hole cards* and bets money during 4 betting rounds. Another 5 cards, called *community cards* are placed in the middle of the table. Community cards are visible to everyone. Hole cards are visible to the player only. In total, each player has access to 7 cards and has to make the best possible 5 cards combination out of those 7 cards.

A poker hand also has a nominal card *dealer*, a player who gives out the cards. This position is usually marked by the *dealer button* (a small circular marker), which is placed in front of a player and moves one position clockwise after each poker hand. The player at the button does not deal the cards for real. The cards are dealt, and the overall game is managed by an extra non-playing person, usually called the *table dealer*. In virtual poker, all dealer tasks are done automatically. The dealer button is also necessary to know which player starts a betting round.

Two players sitting next to the dealer button are required to post a forced bet at the beginning of each hand. The first left player sets a *small blind,* and the second left player

[1] http://www.pokerstars.co.uk/poker/how-to-play/dictionary/

sets a *big blind*. Those are forced wagers over which poker players compete initially. The forced bets also initiate the *pot* and determine the size of the *minimum bet* in the first betting round. Usually each player has to go through a *buy-in* stage. This is where a player takes a certain amount of money into the game. This amount becomes the player's *table stake*. A table stake is the amount of money each player can use to bet. Once the table stake runs out the player loses and leaves the table. The game starts with two cards being dealt face down to each player. Those are called the *hole cards* and belong to the players only. After that, 5 *community cards* are being placed face up in the middle of the table. This results in each player having access to seven cards to create any possible 5 card combination.

The community cards are placed in the following order:

1. A series of 3 cards is placed at first. Those are called the *FLOP.*
2. The fourth card is dealt after that. The community cards convert to the *TURN*.
3. At last, the fifth card in put on the table. All 5 community cards are called the *RIVER*.

The placing order of community cards creates several betting rounds which can be used by players to action. Each unveiled community card may increase or decrease chances of winning the current poker hand for each player. In each betting round players decide in turn about the action they want to perform. The available actions are *FOLD, CHECK/CALL* or *BET/RAISE*.

- **FOLD:** A player forfeits the interest in the pot and is not allowed to bet in further betting rounds of the particular hand.
- **CHECK:** If no bets were placed in a particular betting round or the bet of an acting player is equal to the maximum bet of this round, this player might choose to check and pass the action to the next clockwise player without any additional betting.
- **CALL:** If the current bet of a player is lower than the bet of another player, he has to call and match the highest bet to qualify to the next betting round.

- **BET:** If there were no bets in a particular betting round, or every previous player has checked in the same betting round, then the player might decide to be the first one to place a bet which has to be called by all other players.

- **RAISE:** A player may decide not only to match (call) another player's bet, but to put a higher bet (raise). Every other player who want to progress to the next betting round has to call the new highest bet.

As mentioned previously, one LTH game consists of several poker hands. One poker hand consists of several betting rounds. The four existing betting round are:

1. **PRE-FLOP** - The first round of betting

   Players' judgments are based on their hole cards, as no community cards have been dealt yet. The first two players next to the dealer button are forced to post the small and the big blind. Therefore, the first freely acting player is the third clockwise from the dealer button. The action options in this betting round are *FOLD, CALL* and *RAISE.* The option *BET* is not available because a bet has already been made by the blind posts.

2. **FLOP** - The second round of betting

   The first three community cards are available to all players now. Each player has to re-assess his chances based on 5 (2 individual and 3 community) cards. The first freely acting player is the first active player clockwise from the dealer button. Next active player means a player who has not folded in the previous betting round. The actions available in this round are *FOLD, CHECK and BET.* As soon as one of the players chooses to bet, all other player will have the actions *FOLD, CALL* and *RAISE* available. This is because if a player places a bet, other players have to call the bet (the same amount) or raise his bet (higher amount). The minimum bet is equal to the big blind. If a player wants to bet money in this and all subsequent betting rounds, he has to bet an amount equal to the big blind.

3. **TURN** - The third round of betting

   The next community card has been dealt, and players are re-assessing their winning chances based on 5 card combinations out of 6 cards. The first freely acting player is the

first active player clockwise from the dealer button. The actions available in this round are the same as in the second betting round (FLOP). The minimal bet doubles in this and all following betting rounds.

4. **RIVER** - The fourth round of betting

The last community card is unveiled. Each player has seven available cards to make the best possible 5 card combination. The first freely acting player is the first active player clockwise from the dealer button. The actions available in this round are the same as in the previous two rounds.

SHOWDOWN

The showdown happens when the final betting round is completed. All active players must expose their hole cards so that a winner can be determined. The player with the highest ranked 5 card combination wins the pot. The dealer button is moved to the next position, and the next poker hand begins. If, at any betting round, there is only one active player left (all others have folded), then the pot is awarded to the last active player and no showdown occurs.

2.3. Critical review of existing applications

By having the basic knowledge about Android application development and LTH rules, I can continue with the third step of my research. This section describes similar applications available on Google Play and the detailed comparison of their features and functionality. The testing devices would be a Google Nexus 7 tablet with Android 4.2. The secondary testing device would be a HTC Desire S. All application reviews will be based on features and functionalities found while testing on those two devices. The conditions defined in section apply. The features used for comparisons are explained in Chapter 3 in greater detail.

2.3.1. Comparison to standard poker applications

Google Play provides a search functionality which is based on keywords. I used the keywords *Hold'em* and *Poker* in my search query to obtain the most relevant results. The 5 most popular poker applications were reviewed and their features compared. The comparison is shown in Table 2-III.

Table 2-III: Comparison to standard poker applications

Application name	My Game	Live Hold'em Poker Pro	Zynga Poker	Texas Hold'em Poker Deluxe	Texas Poker	DH Texas Poker
1. Full gameplay	Yes	Yes	Yes	Yes	Yes	Yes
2. Interactive tutorial	Yes	-	-	-	-	-
3. Rules section	Yes	Yes	-	-	-	-
4. AI opponents	Yes	-	-	-	-	-
5. Game settings	Yes	Yes	Yes	Yes	Yes	Yes
6. Performance feedback	Yes	-	Yes	Yes	-	Yes
7. Interaction sounds	Yes	Yes	Yes	Yes	Yes	Yes
8. Animations	Yes	Yes	Yes	Yes	Yes	Yes
9. Play history log	Yes	Yes	Yes	-	-	-
10. Textual table representation	Yes	-	-	-	-	-
11. Instant winning percentages	Yes	-	-	-	-	-
12. Combination hints	Yes	-	-	-	Yes	-
13. Combination ranking table	Yes	Yes	Yes	Yes	Yes	Yes
14. Visual table navigation guide	Yes	-	Yes	Yes	Yes	Yes
15. Vibration notifications	Yes	Yes	-	Yes	Yes	Yes
16. Game speed adjustment	Yes	-	-	-	-	-
17. UI optimisation for tablets	Yes	Yes	Yes	Yes	Yes	Yes
18. Offline application	Yes	-	-	-	-	-
19. High quality graphics	Yes	Yes	Yes	Yes	Yes	Yes

2.3.2. Comparison to educational poker applications

Table 2-IV shows the comparison of my application to existing educational applications.

Table 2-IV: Comparison to educational poker applications

Application name	My Game	Poker No-Limit Trainer	Texas Touch'em	Poker simulator	My PokerTrainer	Shuffle Bot Hold'em Calc.
1. Full gameplay	Yes	-	-	-	-	-
2. Interactive tutorial	Yes	-	-	-	-	-
3. Rules section	Yes	-	-	-	-	-
4. AI opponents	Yes	-	-	-	-	-
5. Game settings	Yes	-	Yes	-	-	Yes
6. Performance feedback	Yes	-	Yes	-	Yes	Yes
7. Interaction sounds	Yes	-	Yes	-	-	-
8. Animations	Yes	Yes	-	-	Yes	Yes
9. Play history log	Yes	-	-	-	-	-
10. Textual table representation	Yes	-	-	-	-	-
11. Instant winning percentages	Yes	-	-	Yes	-	Yes
12. Combination hints	Yes	-	-	-	Yes	Yes
13. Combination ranking table	Yes	-	Yes	-	-	-
14. Visual table navigation guide	Yes	-	-	-	-	Yes
15. Vibration notifications	Yes	-	-	-	-	Yes
16. Game speed adjustment	Yes	-	-	-	-	-
17. UI optimisation for tablets	Yes	-	-	-	Yes	Yes
18. Offline application	Yes	Yes	Yes	Yes	Yes	-
19. High quality graphics	Yes	-	-	-	Yes	-

The Google Play search query with the keywords *Hold'em*, *Poker* and *Trainer* returned a list of 7 applications. Although those seemed to be the most relevant applications, after having a quick look through them, I decided to take a step back and use the keywords *Poker* and *Trainer* only. This resulted in a list of 121 applications, which gave me more research material and a better overview of existing educational poker applications.

2.4. Research conclusion

By looking at existing poker applications available on Google Play, I think that there is a real need to create an application which combines a fully functional game and a series of educational features. Other attraction points would be compatibility with most screen sizes and offline play mode. As shown in the Table 2-III and Table 2-IV, some similar applications are available. However, all of them specialise in one area and only provide a subset of features available through my application. The reviewed applications either provide a separate poker game with basic functionalities or educational features without a fully playable game. None of the reviewed applications has the whole set of features, which are defined in the specification section of this document. Furthermore, most educational applications have an extremely poor user interface, bland graphics and are not optimised for tablet users.

2.5. Summary

In this chapter, I have explained what Android is, described its system Architecture, reflected on the digital distribution platform Google Play and showed the concluded technical details for my project. I have also talked about the basic poker rules and the game play structure. Furthermore, I have presented a clear comparison of my application with similar applications available on the market.

Chapter 3 : Requirements specification

This chapter describes the identified requirements for my application. It starts with functional requirements, which are split into primary and secondary requirements. Those were captured through an in depth analysis and comparison of existing applications in sections 2.2 and 2.3. The second part describes non-functional requirements, which were identified in section 2.1. Those reflect extra aspects which are vital for the user experience, but not necessary for the application to function.

3.1. Primary functional requirements

These requirements are the core components that are needed for the application to function as intended. It is a minimum set of features required to achieve the objective of this project.

- **Play screen**: The play mode should take the user to a fully functional LTH game. This environment should function independent from the implementation of the educational features, but might include some of those. Player decision support should be accessible. It will have an interactive game table as the main area. Five community cards will be placed in the middle. Player icons will be placed around the table. There should be a notifications window which will show live game information in textual form. It should also provide an easily accessible settings and menu buttons. At the bottom of the screen, there should be a task bar with available poker actions.

- **Tutorial screen:** The tutorial screen should take the user to an interactive environment where the game rules are explained on a predefined scenario. This mode will be based on the play mode. The game flow will be explained step by step at a slow pace. This screen should also provide a settings button and a menu button.

- **Rules screen**: The rules screen should present a selection of educational features. This environment is the first one to be visited by a new user and should appear on top of the menu screen. Once the rules section was studies, the user should be able

to play the tutorial mode with a better understanding or even jump straight into play mode.

- **Educational features:** The following educational features should be implemented in either an independent way or in combination with the play, tutorial and rules screen.
    - Play table information represented in a clear textual form
    - Calculated individual winning percentages for hole cards
    - Card combination hints
    - Card combination ranking table
    - Visual identification of combinations in a set of cards
    - Easily accessible section of poker rules and basic poker terms
    - Visual play screen guide with explanations of screen buttons and visual concepts
- **AI module:** The game should have a build-in AI module to function properly. The AI module should control the opponents with a competent difficulty level. Several playing strategies should be implemented to enable a varied and exciting user experience. Randomly selected strategies should be assigned to each opponent at the beginning of the each game.
- **Critical actions confirmation:** It is easy to tap something on a touchscreen without intention. Therefore, critical actions like closing the application or restarting the game should be caught by a confirmation window.

3.2. Secondary functional requirements

Secondary requirements are considered as additional features, which will be implemented once the primary requirements are in place.

- **Game settings:** The basic game settings should allow the user to control the audio output, switch the vibration on/off and adjust the game speed. The latter

functionality is extremely useful because players might have a different speed of thinking and would like to adjust the game speed to their personal preference.

- **User profile:** The game should store user profile information. It should contain some basic statistical information like the number of games played and the proportion of won/lost games. These statistics will display the user's progress within the game. It should be also possible to reset the stored data.

- **Interaction sounds and background music:** The game should be entertaining. Background music and interaction sounds (audio feedback) should be in place.

- **Animations:** The application should have animations for most game actions to look more dynamic. Typical animations would be moving cards when dealt, moving chips when a bet is set, highlighting card combinations and the highlighting winning player.

- **About section:** This section should take the user to a details description about the game development and copyright notice. It will show all necessary copyright and licensing information.

- **Loading screen:** Considering the high graphics quality, it surely will take a longer time to load all assets on the low-end devices than on high-end devices. The application should provide a loading screen where user gets visual feedback about the current progress of the loading process.

- **Play history log:** If a player missed a certain step within the game, he should be able to review this information by analysing the log. The log should be an extra window which contains the entire play history of a game from the first step till finish.

3.3. Non-functional requirements

- **Portability:** The application has to run on both testing devices and on Android 2.3.5 to 4.2 in general. It also has to be optimised for smartphone devices and tablet devices. It should provide the same functionality on both testing devices regardless their respective technical specifications. For all other devices, the minimal hardware

requirements would be a 1GHz CPU, touchscreen and a screen size of minimum 800x480 pixels and maximum 1280x800 pixels.

- **Re-usability**: The game should be implemented by using the MVC design pattern. The core classes have to be developed in a way that allows re-usability in other cards games. The application design should be flexible such that more educational features could be added on demand. UI and game logic threads should be synchronized but not dependent on each other.

- **Efficiency (speed)**: I was not able to identify official guidelines about standardised user feedback response time on Android OS. According to the GNOME developer guidelines[2] switching between main screens (settings, tutorial, rules and play) should not take longer than 1 second. Simple button clicks, like switching on/off sounds, should not take longer than 0.1 seconds. These figures should be used as standards. The application efficiency should be as close as possible to those numbers. AI decision should take no longer than 1 second to respond.

- **Efficiency (storage):** Considering that the application graphics will be designed for a resolution of 1280x800 pixels, and looking into storage requirements of the reviewed applications, I have identified that the average application size is 12 megabytes. The application will also contain sounds which will contribute to the storage size. Therefore, the packaged application should be of an approximate size of 13 megabytes.

- **Efficiency (memory)**: The memory allocations will depend on the screen size. The more pixels have to be drawn on screen, the more memory will this require. Having the extreme case of all graphics being loaded in memory at the same time, it should require approximately 12 megabytes of memory. For smaller screen sizes, the number is expected to be lower by 3-4 megabytes.

[2] https://developer.gnome.org/hig-book/3.5/feedback-response-times.html.en

3.4. Use case examples

The application will have two key use cases shown in Figure 3-1.

- **Learn rules:** A new player who does not know anything about Limit Texas Hold'em will be able to learn the rules. The main menu should offer a *Rules* button. Clicking on it will take the player to an environment where rules are explained in a textual form. Visual examples of poker combination ranking will also be accessible. The rules section will also include a visual table guide, which explains the poker table components. Once the rules are studied, the player can explore the *Tutorial* section, where the acquired knowledge can be applied in a step by step manner within a pre-defined game session.

- **Play game:** A player who already knows the rules of Limit Texas Hold'em will be able to improve his poker skills by simply playing the game. The main menu will offer a *Play* button. Clicking on it will launch a new game session with several AI controlled opponents. The player will receive help by clicking the *Information* button. The game will continue until one player is left at the table. After the game, the Player will be presented with the basic play statistics in the main menu.

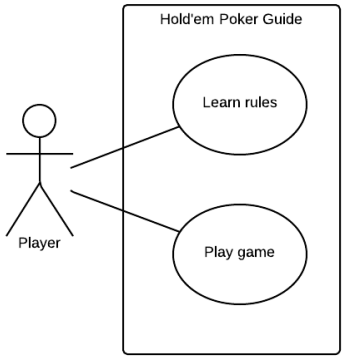


Figure 3-1: Key use cases

3.5. Summary

This chapter described the identified requirements for my application. It showed the primary and secondary functional requirements. In the second part I described non-functional requirements which reflect extra aspects. Those are important for the user experience but not essential for the application to function.

Chapter 4 : Design and Implementation

This chapter describes the high level design of the application and implementations details. I will reflect on the design patterns used and the content of the application packages. This will allow reviewing the frontend concepts and backend concepts separately. I will discuss the general application user interface and the screen flow. Non-trivial implementations decision and algorithms will be explained in detail.

4.1. Development environment

Application development in Android is based on Java and XML. I used Eclipse combined with the Android Development Tools (ADT). The implementation of the project is realised by adapting the LibGDX Framework [2]. LibGDX is a cross platform java based framework which allows developers to run their application on a desktop machine with a compatible java installation. This is especially convenient as it saves a lot of time while debugging as there is no need to deploy your application to a mobile device after every small change.

4.2. High level design

The MVC design pattern was chosen in conjunction with the observer/observable patter because it allows a clear system design structure and takes the processing load from the controller. When adapting those patterns the code was divided into four different packages shown in Figure 4-1. This figure also shows how those packages interact with each other at a high level.

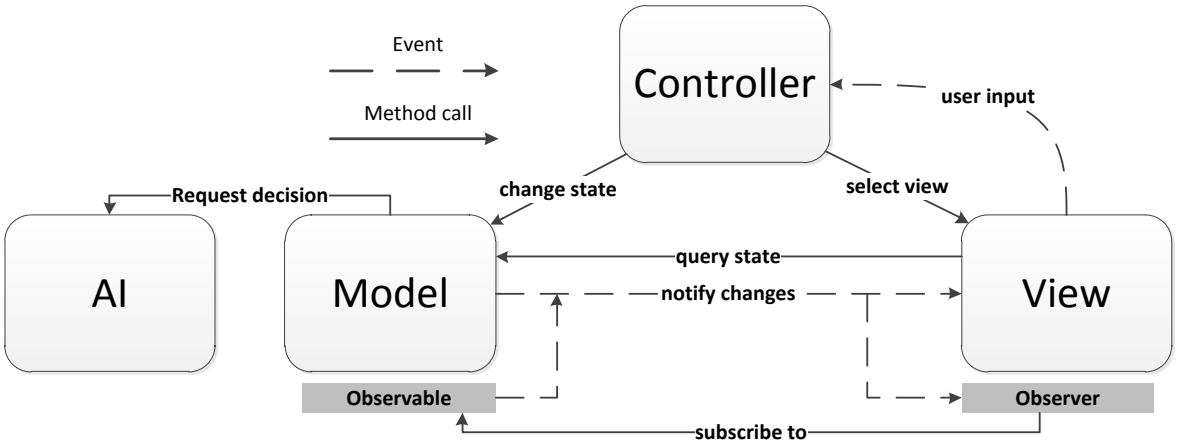


Figure 4-1: MVC Pattern with Observer/Observable extension

4.2.1. Model package

The model package contains classes which represent real entities like *Cards, Deck, Player*, and logical entities like *Hand* and *DealerBoard*. The classes within this package are updated by the controller package. Certain observable model classes have the ability to update their respective observers in the view package. When asked by the controller, a player model is able to request a decision from the AI package. The high level class diagram is shown in Figure 4-2.

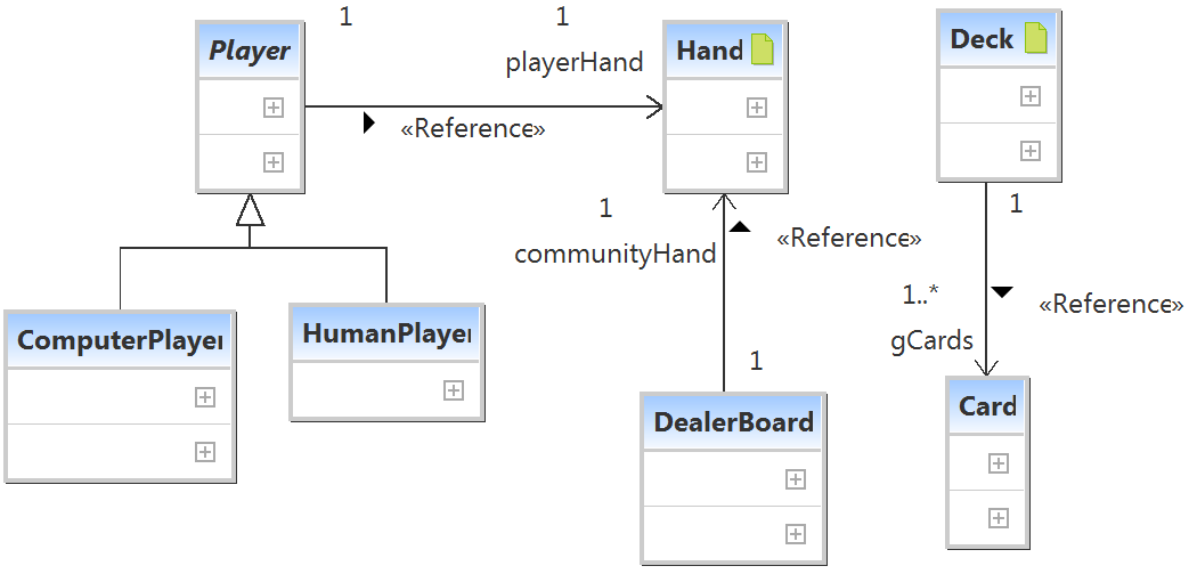


Figure 4-2: Model Package - high level class diagram

4.2.2. View package

The view package consists of UI elements like screens and widgets. It also contains user interaction elements, which allow user input to be captured and diverted to the controller package for processing. The link between the view and the controller package is extremely important because most mobile devices do not provide physical input peripherals. Consequently, the UI (view package) has the role of the input and output. Certain classes in the view package have the ability to subscribe to models and get notified directly whenever a model state changes. The controller package navigates which screen from the view package is currently visible. The high level class diagram is shown in Figure 4-3.

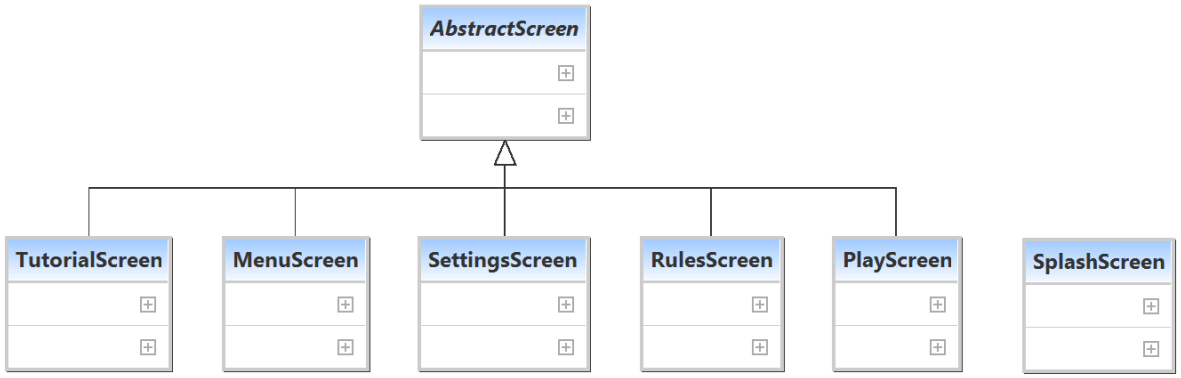


Figure 4-3: View package - high level class diagram

4.2.3. Controller package

The controller package is responsible for controlling the game logic and table play logic. When a model state has to be updated due to some game logic events, the controller package changes the state of the model. This package also processes user input. By controlling the game logic and processing the user input, which affects the game logic, it is easy to combine those two tasks within one package. Furthermore, it controls the transitions between application screens. The controller package also contains a number of utility classes. The high level diagram of this package is shown in Figure 4-5 and Figure 4-5.

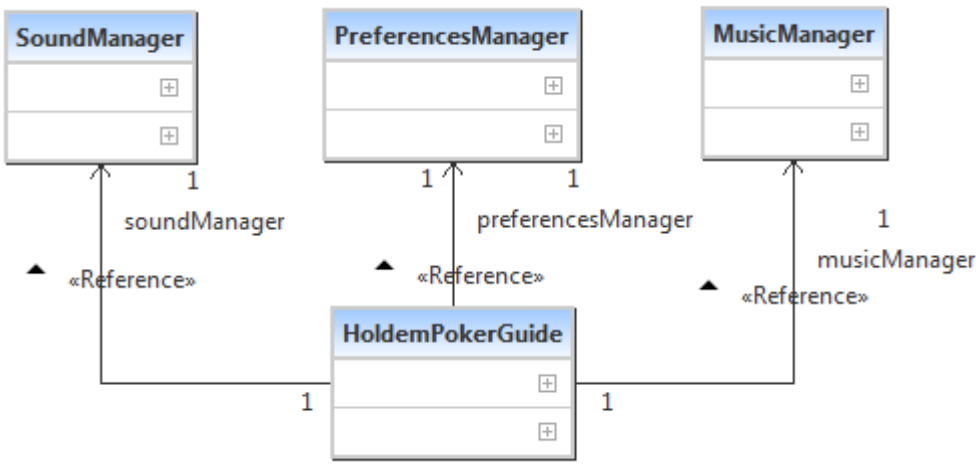


Figure 4-4: Controller Package - high level class diagram part 1

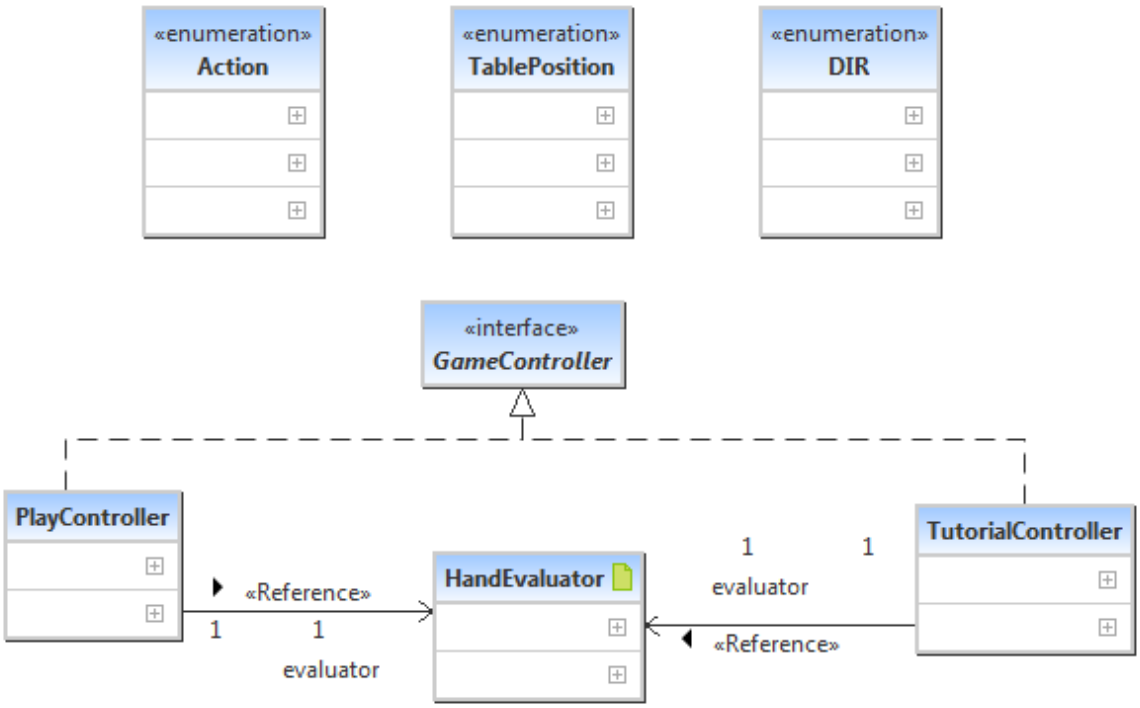


Figure 4-5: Controller Package - high level class diagram part 2

4.2.4. AI package

Figure 4-6 shows the high level class diagram for the AI package.

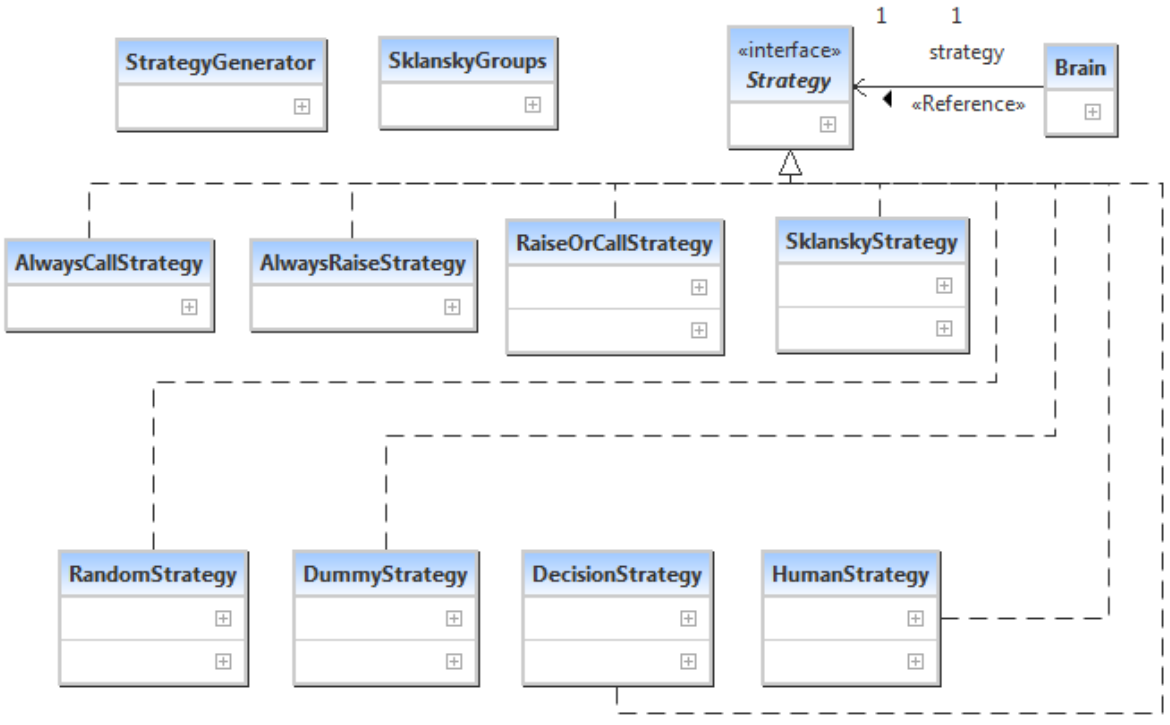


Figure 4-6: AI Package - high level class diagram

The AI package contains several AI strategies, which control computer players. Each computer player model gets an instance of the brain object. This brain instance is assigned with one strategy object. Each strategy is capable of taking a collection of available decisions and the context. Based on the context values it chooses and returns one action.

4.2.5. Application model and screen flow

At the frontend, the application consists of six main screens. The abstract overview of the screen flow is shown in Figure 4-7, where each screen is represented by a separate state. Once the application has started, the Android OS integrated back-button can be used to exit the application in any inner state.

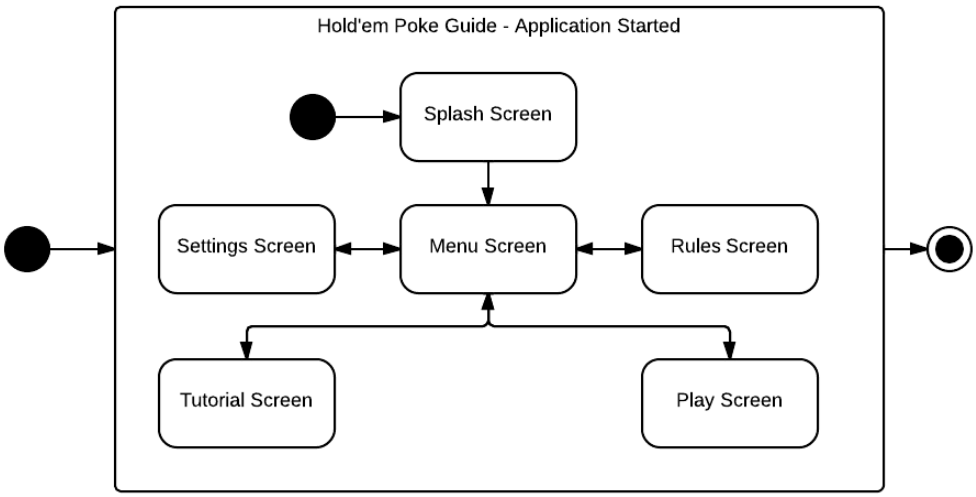


Figure 4-7: Application screen flow

At the backend, there are 2 parallel threads running. The first one handles the game logic and takes care of the models. The second thread is responsible for updating the UI and allows smooth animations. This structure delivers a dynamic and responsive user experience.

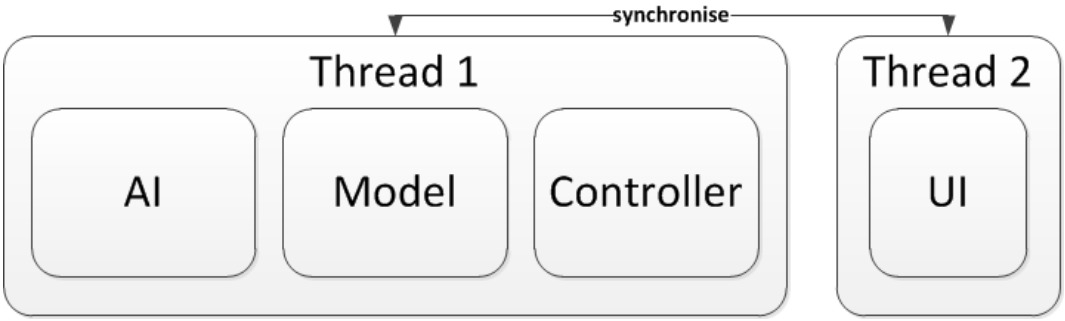


Figure 4-8: Implementation Structure

4.3. Low level design

This section describes the implementation details. I will discuss each package in a separate section and explain decisions made while writing the code for my application. Non-trivial class functionalities are explained.

4.3.1. Model package

The model package contains classes which represent real entities and logical entities. Models simply store information and have a certain state which is used by the view and controller package.

- ***Card:*** This class represents a playing card from a deck of 52 cards. It stores a card suit and card rank. It also stores a card code that is two characters long. Cards ranks are encoded as '2', '3', '4', '5', '6', '7', '8', '9', 'T', 'J', 'Q', 'K', 'A'. Card suits are encoded as 'c', 'd', 'h', 's'. For example, Two of Clubs has the code '2c'. *(adapted from UoA* [3]*)*

- ***Deck:*** This class represents a collection of 52 unique cards. There are 4 different suits and 13 different ranks. It stores the amount of cards left in the deck which represents a deck size. The deck class also allows shuffling cards and taking a card from the deck. *(adapted from UoA* [3]*)*

- ***Hand:*** This class represents a collection of up to 7 cards. It could be used to store just 2 cards (hole cards), 5 cards (community cards) or 7 cards (2 hole cards + 5 community cards). It essentially contains a set of cards which can be assigned to any player or used to find a valid combination within this set. *(adapted from UoA* [3]*)*

- ***Player:*** This is an abstract class which represents a player. It has two subclasses, *HumanPlayer* and *ComputerPlayer*. A player class stores the player name, amount of cash, bet and the player's cards.

- ***ComputerPlayer:*** This class represents a computer player. It inherits all attributes and methods from the player class. A computer player object can be asked to pick one action when given a list of available actions and the context. Cards of a computer player are not visible to the user.

- ***HumanPlayer:*** This class represents a human player. It inherits all attributes and methods from the player class. A human player object can be asked for a decision, which will initiate decision buttons to be displayed on the screen. Cards of the human player are visible to the user.

- ***DealerBoard:*** This class represents the table area which is manager by the table dealer in real life. It contains the community cards and the betting pot. It also contains a message window which is visible to every player and replaces the announcements of a real table dealer.

4.3.2. View package

The view package consists of UI elements like screens and widgets. It also contains user interaction elements, which allow user input to be captured and diverted to the controller package for processing. Furthermore, the view package is responsible for animations and dynamic appearance of the application.

- ***AbstractScreen:*** This is an abstract class which represents a generalised screen object with all its attributes and behaviour. It defines standard animations and screen transition methods. It catches the Android back button and displays a confirmation window to the user. The abstract screen has access to all game assets which can be used by its subclasses. It has a background image which is passed on to all other screens. Furthermore, this class defines the target resolution which is also valid for all sub classes.

- ***SplashScreen:*** User response feedback is extremely important. When the application starts, all assets have to be loaded. This process might take up to 10 seconds on a low-end device. The role of the splash screen is to present a loading progress bar. Instead of just seeing a black screen and waiting till the application starts, the user is presented with visual information about the loading progress. The splash screen is displayed once when the application is started and is never visited again. This class does not extend the abstract class because its purpose is only to display the progress of loading without any additional functionality being required.

- ***MenuScreen:*** This is the logical centre of the application. It extends the abstract screen class. It has a bi-directional transition to every other screen except the splash screen. The essence of this screen is to provide a menu of buttons which lead to one of the other four screens. This screen also presents the basic user statistics recorded during play sessions.

- ***RulesScreen:*** The rules screen shows the game rules in a textual form. It also extends the abstract screen class. Furthermore, a poker table navigation guide and graphical card ranking examples are shown. This screen also contains the "About" section of the application which shows the basic copyright and licensing information.

- ***SettingsScreen:*** The settings screen extends the abstract screen and contains the basic application settings described in the requirements chapter of this document. Additionally, this screen includes a reset button for the statistics show in the menu screen.

- ***TutorialScreen:*** This screen is a pre-defined scenario play. It extends the abstract screen and has a similar functionality as the play screen. The restriction is that the user cannot act freely and must follow the guide. The screen should be visited by players who already studied the rules screen so that the acquired knowledge can be applied here.

- ***PlayScreen:*** This screen is the most interactive environment of the game. It allows the player to participate in a fully functional poker game against AI opponents of various difficulty levels. It also provides access to hints and live play log where every step of the game can be reviewed. Quick access to settings and rules is also integrated.

- **Observers:** This term describes all classes whose purpose is to display the state of the observable models. They are widgets. Observer classes subscribe to their according models when the application starts. Whenever the model state has changed, the observers are notified. Following they grab the new information from the model and translate it into a visual representation. Observer classes are *DealerBoardPanel*, *HumanStrategyPanel* and *PlayerPanel*.

4.3.3. Controller package

The controller package contains the application controller, table play controller, tutorial controller, enumerations and utility classes. In this section, I will go into details and explain the purpose and the general functionality of each class.

- ***Action:*** This class is a public enumerator of actions that can be performed during gameplay. Each action has a name and a verb representation. This way, whenever a player performs an action within the game, the verb representation of this action can be retrieved so that a meaningful sentence can be build and presented to the user.

- ***DIR:*** This class is a public enumerator which contains information about all assets which have to be loaded before the game starts. For each asset, it stores the asset category, type, path to the actual resource and an asset name. This structure is especially convenient as it provides an extra layer between client classes who use the assets. A strong integrity is achieved. For example, when an asset is required, we simply use the name of it and do not care about the actual path as this is handled by the DIR class.

- ***GameController:*** This is an interface which defines a set of public methods required for a poker table controller. It is implemented by the *PlayController* and the *TutorialController*. It provides methods for all the public information like the amount of players, the current pot, currently active player or the community cards. Hence, whenever the AI is asked for a decision, we pass on a *GameController* as context so that the AI strategy can get the public info and decide on an action.

- ***PlayController:*** This class implements the *GameController* interface. It controls a poker table game and manages all related public and private information. Public information can be accessed by the methods defined in the *GameController*.

- ***TutorialController:*** This class implements the *GameController* interface. It controls a poker play table. The difference is that it runs the game flow on a predefined scenario. However, all public *GameController* methods are still available, and it can be integrated with the AI in the same way if necessary.

- ***HandEvaluator:*** This is a utility class which compares, identifies and ranks hand object. It is used to determine the winner at the end of each play round. It is also used to calculate the strength of a given card combination. Furthermore, this class is essential for the winning chance percentage calculations. It works in combination with an evaluation algorithm and returns a rough percentage of player's winning chances. It also influences the decisions of the AI package. *(adapted from UoA* [3]*)*

- ***HoldemPokerGuide:*** This is the main controller class which initialises the whole application. It contains information about the game settings and all screen instances. It processes the user input to switch between screens and handles the Android OS system calls. Utility class instances like *AssetManager*, *MusicManager*, *SoundManager* and *PreferencesManager* are also stored in this class.

- ***MusicManager:*** This class handles the background music and provides public methods to play, stop, change the volume and enable/disable audio output. It constantly reads the music file bits from the storage.

- ***SoundManager:*** This class handles the sounds within the application. Basic options like changing the sound volume and enable/disable audio output are also provided. The difference to the *MusicManager* is that because of the small file size sound files are loaded into and read from the memory rather than from the storage.

- ***TablePosition:*** This is a public enumerator which stores coordinates of objects in the *TableScreen*. One usage example is the circular sitting order of players at the table. Great flexibility is provided here. For example, we can change the coordinates of players and let them sit at a square table without touching any of the game classes.

4.3.4. AI package

The AI package is responsible for the decision making of computer players. It includes several strategies, which all behave differently. Each player object is assigned with an AI at the beginning of the game. Some of them behave educated. Others just chose random actions. The point is that the user does not know which strategy was assigned to each of the computer players. That brings a lot of variety and unexpected behaviour into the gameplay.

- ***Brain:*** This class represents the brain of a player object. Each brain object is assigned with a strategy. The brain class provides few public methods. Those allow requesting a decision when a set of available actions and a context are given.

- ***Strategy:*** This is an interface which defines the public methods required by a brain object. All types of strategy classes implement this interface and define the logic for the decision request methods. Most strategies include a random value which affects the decision and makes the opponents more unpredictable.

- ***AlwaysCallStrategy:*** This is a strategy class which acts aggressively and always matches the bets of opponents.

- ***AlwaysRaiseStrategy:*** This strategy always tries to place higher bets than its opponents.

- ***DecisionStrategy:*** This strategy makes its decisions based on the actual cards strength value provided by the *HandEvaluator* class. When the value is high enough it will bet. Lower hand values result it folding the cards and miss out a playing round.

- ***DummyStrategy:*** This strategy performs a predefined sequence of choices. It is used for the computer player objects in the tutorial game play of the application.

- ***HumanStrategy:*** This strategy is assigned to the human player. It has a direct connection to the UI. Whenever asked for a decision, this strategy notifies the UI which options are available and blocks the thread while waiting for user input. When user input was received. This strategy returns the chosen action.

- ***RaiseOrCallStrategy:*** This strategy is a mixture of the *AlwaysRaiseStrategy* and *AlwaysCallStrategy.* Whenever asked for action, it chooses one of those two strategies with a chance of 50/50.

- ***RandomStrategy:*** This strategy randomly decides which available action to pick.

- ***SklanskyStrategy:*** This strategy makes decisions based on the *SklanskyGroups* class. Sklansky groups categorise all possible hole cards combinations. Groups are ordered by their strength with 1 being the strongest. There are 9 groups in total [1].

- ***StrategyGenerator:*** This class provides a public method which returns a randomly selected strategy. Before each play round, this class returns a random strategy for every computer player. This means that most likely computer player would behave different from what the user has seen in the previous round.

4.4. Card statistics algorithms

Instant calculation of rough winning percentages requires a simulation to be run in the background. A lot of CPU power is needed, which is a limited resource on mobile devices. To calculate the winning percentage for a certain combination of cards it has to be compared to all other possible combinations. The class *HandEvaluator*, which was written by the Computer Poker Research Group from University of Alberta in year 2000, provides few excellent methods which allow a given combination of cards to receive a unique number representing its rank. Two equally strong card combinations would receive the same unique rank value. I have implemented two algorithms. Both use these ranking value methods.

4.4.1. Hand strength

Hand strength is the probability that a given hand is better than that of an opponent. We assume that an opponent is equally likely to have any possible two hole card combinations. All of the possible opponent hands can be enumerated. For example, if the user has 2 hole cards and 3 community cards are already on the table, there are 47 cards remaining in the deck. Therefore {47 choose 2} would give us 1081 possible 2 card combinations that an opponent could have. To determine the winning chance of the user's card combination, we have to calculate its rank value and compare it to the rank values of all possible opponent combinations. While looping, we count how many of those combinations have a higher, equal or lower value. By using those counters and dividing them by the total number of comparisons, we receive a value between 0 and 1 which represents the likelihood that the user's combination has a higher value than the opponent's combination [17]. The pseudo code for this algorithm is shown in Figure 4-9. The actual implementation can be seen within the *HandEvaluator.getEquity()* method. The resulting number has to be raised to the power of the number of opponents at the table. There are other factors that might affect the final result. However, for the scope of this project, the formula was generalised. It serves as a rough guideline for the user.

```
HandStrength(ourcards,boardcards)
{
   ahead = tied = behind = 0
   ourrank = Rank(ourcards,boardcards)
   /* Consider all two-card combinations
      of the remaining cards.  */
   for each case(oppcards)
   {
      opprank = Rank(oppcards,boardcards)
      if(ourrank>opprank)         ahead += 1
      else if(ourrank==opprank)    tied += 1
      else /* < */               behind += 1
   }
   handstrength = (ahead+tied/2) / (ahead+tied+behind)
   return(handstrength)
}
```

Figure 4-9: Hand Strength algorithm - pseudo code [17]

This algorithm was successfully implemented and instantly provides a winning percentage to the user while playing the game.

4.4.2. Hand potential

As soon as the first 3 community cards are received, there are 2 more community cards to be unveiled. The potential impact of those cards can be pre-calculated. As a result, we would receive 2 probabilities. The first one would be the positive potential (PPos). It represents the chance that the card combination a player already has (2 hole cards + 3 community cards) will develop into a combination with a higher value after the last 2 community cards are placed on the table [17]. The second value we would receive is the negative potential (NPot). It essentially gives us the chance that the card combination a player already has will develop into a combination with a lower value after the last 2 community cards are placed on the table [17].

The difference to the hand strength algorithm is that we not only look at potential card combinations of the opponent, but also take into account the potential community cards still to be placed on the table. The simulation required for this calculation will need much more time to be performed. Previously, we had to look at {47 choose 2} = 1081 possible combinations. To calculate the potential, for each of those 1081 calculations, we will have to run another simulation of {45 choose 2} = 990 possible combinations. This means, it costs us

1070190 (1081*990) comparisons for the potential value calculation of a single player. Hand strength required just 1081 comparisons. Figure 4-10 shows the pseudo code. The actual implementation can be reviewed in *HandEvaluator.get2LevelFlopPotential().*

```
HandPotential(ourcards,boardcards)
{
   /* Hand Potential array, each index represents
      ahead, tied, and behind.  */
   integer array HP[3][3]  /* initialize to 0 */
   integer array HPTotal[3] /* initialize to 0 */

   ourrank = Rank(ourcards,boardcards)
   /* Consider all two-card combinations of the
      remaining cards for the opponent.  */
   for each case(oppcards)
   {
      opprank = Rank(oppcards,boardcards)
      if(ourrank>opprank)      index = ahead
      else if(ourrank=opprank) index = tied
      else /* < */             index = behind
      HPTotal[index] += 1

      /* All possible board cards to come.  */
      for each case(turn)
      {
         for each case(river)
         {  /* Final 5-card board */
            board = [boardcards,turn,river]
            ourbest = Rank(ourcards,board)
            oppbest = Rank(oppcards,board)
            if(ourbest>oppbest)       HP[index][ahead] += 1
            else if(ourbest==oppbest)  HP[index][tied] += 1
            else /* < */              HP[index][behind] += 1
         }
      }
   }

   /* PPot:  were behind but moved ahead.  */
   PPot = (HP[behind][ahead] + HP[behind][tied]/2
        + HP[tied][ahead]/2) / (HPTotal[behind]+HPTotal[tied]/2)
   /* NPot:  were ahead but fell behind.  */
   NPot = (HP[ahead][behind] + HP[tied][behind]/2
        + HP[ahead][tied]/2) / (HPTotal[ahead]+HPTotal[tied]/2)
   return(PPot,NPot)
}
```

Figure 4-10: Hand potential algorithm - pseudo code [17]

This algorithm was successfully implemented but not included in the release of the application. The reason for exclusion is that it takes too much time to perform the entire simulation on the low-end devices. In detail, one successful hand potential calculation for a single player took 5 seconds on a Google Nexus 7 (NVIDIA® Tegra® 3 quad-core processor) device and 20 seconds on a HTC Desire S (1 GHz Scorpion CPU) device. Such delays are not acceptable during gameplay.

4.4.3. Effective hand strength

By having the potential values and the hand strength value, we could calculate the effective hand strength. The following formula was defined by Darse Billings in his paper [17] :

$$Pr(win) = Pr(ahead) \times Pr(opponent\ does\ not\ improve)$$

$$+ Pr(behind) \times Pr(we\ improve)$$

$$= HS \times (1 - NPot) + (1 - HS) \times PPot$$

This formula produces a more precise winning chance percentage for the user. However, the potential values are not available due to reasons explained in 4.4.2.

4.5. Application walkthrough

The first screen the user will see is the splash screen. It shows an animated loading bar which provides visual feedback about the loading progress of the assets. Figure 4-11 shows the loading screen on the right side and position within the screen flow on the left side.

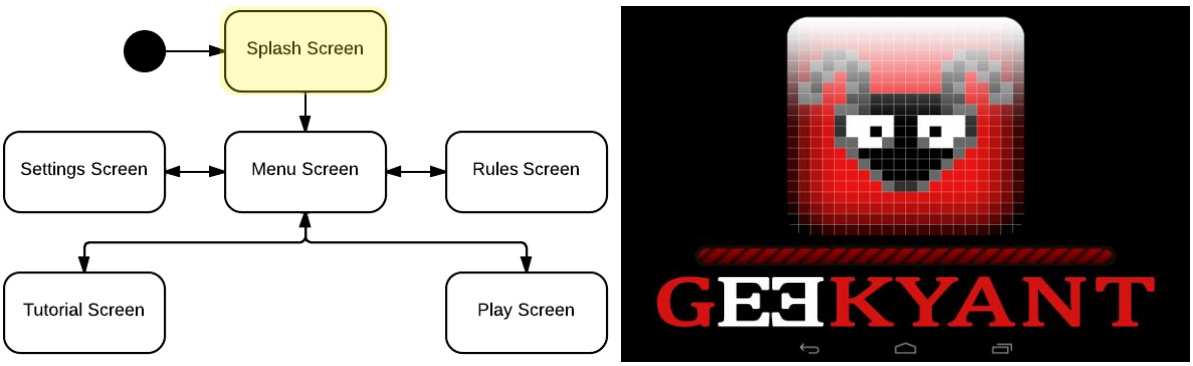


Figure 4-11: Loading Screen

The next screen is the main menu shown in Figure 4-12. It shows the four options in the main menu on the right side. User statistics are shown on the left side.

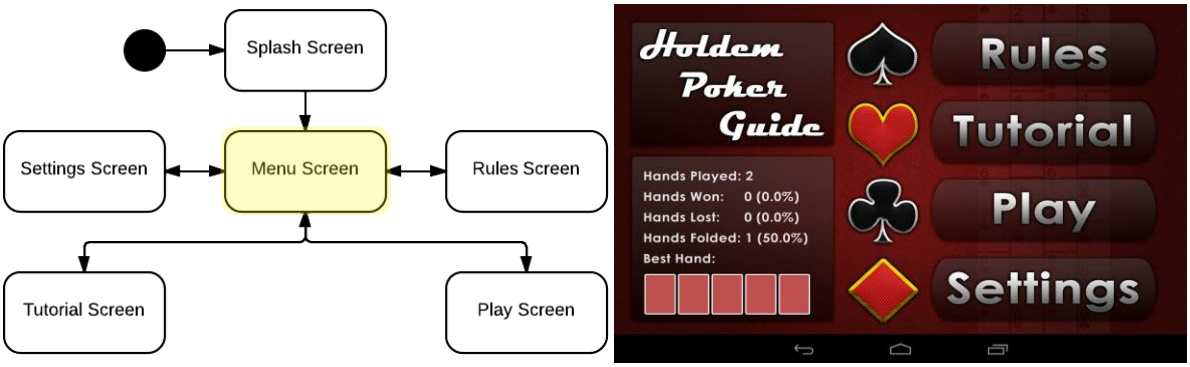


Figure 4-12: Menu screen

The rules screen is shown in Figure 4-13. It offers the user 4 subsections. There is a visual guide which explains the elements of a poker play table. The subsections are shown in Figure 4-14. There is a scrollable list of poker rules, visual examples of combination ranks and copyright information.

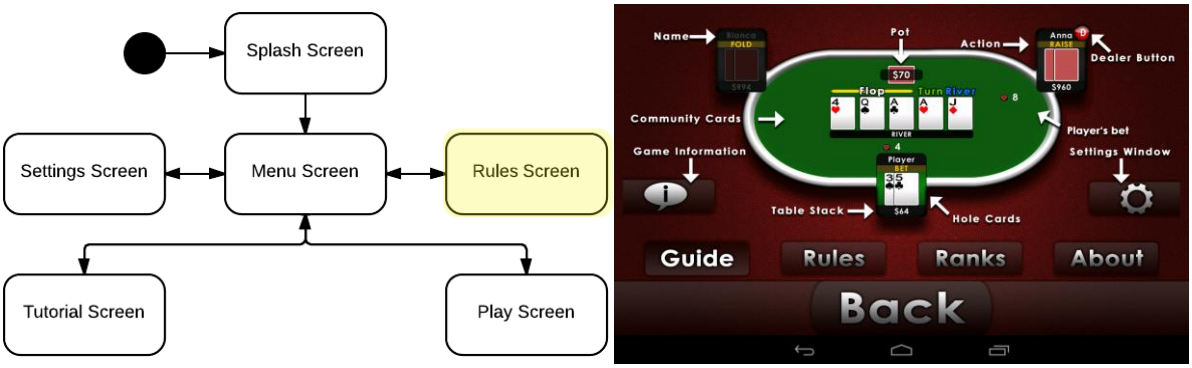


Figure 4-13: Rules screen

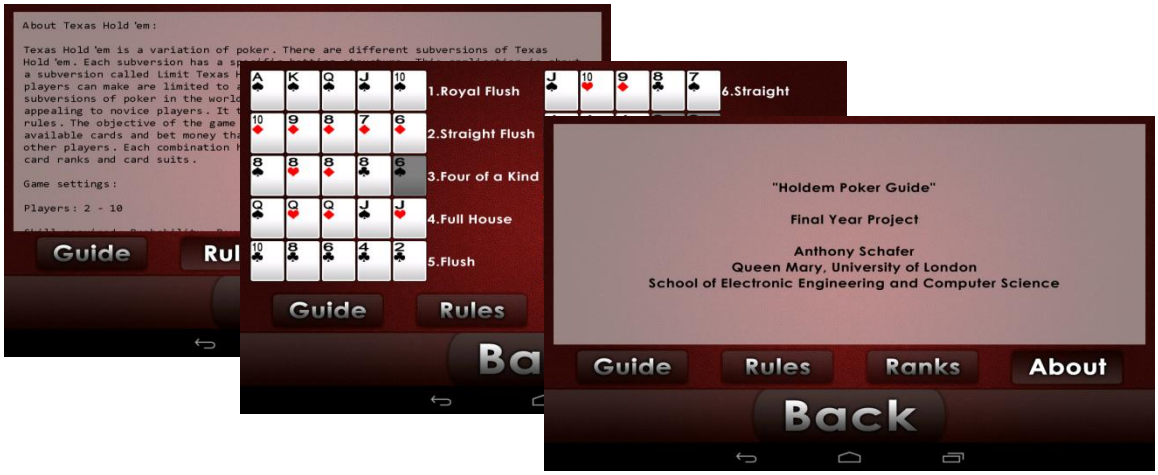


Figure 4-14: Rules screen - subsections

Another screen which the main menu leads to is the settings screen shown in Figure 4-15. It offers audio output controls, game speed slider and vibration settings. In the bottom left corner, there is a statistics-reset button which resets all values shown on the left side of the menu screen.

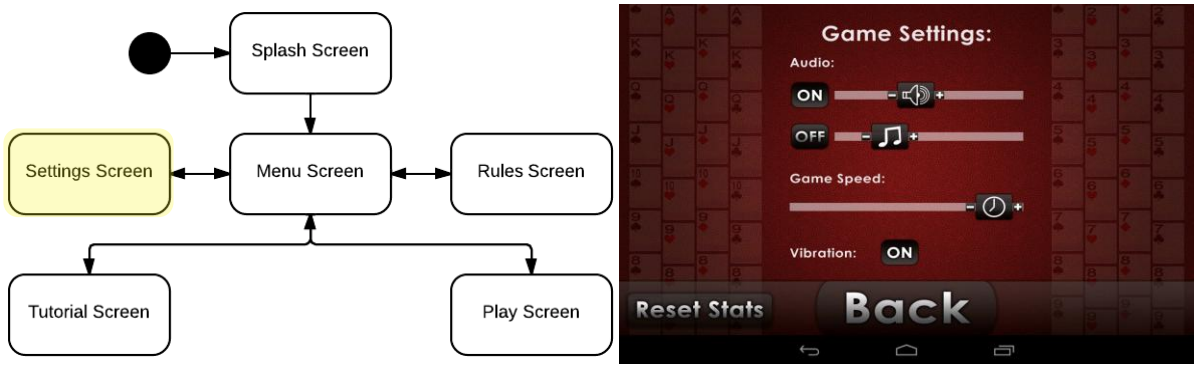


Figure 4-15: Settings Screen

The play screen is shown in Figure 4-16. It includes a full gameplay and is the most interactive environment within the game. At the bottom, there is an action bar where player decision is requested. Just above the action bar, on the left and right side, there are two buttons, which lead to the information or settings subsections shown in Figure 4-17. The Information subsection on the left side offers detailed information about the game. It also shows the player decision support, a scrollable game log and all the subsections of the rules screen shown in Figure 4-14. The settings subsection on the right side offers the most settings available in the settings screen, a table restart button and a back to the menu button.

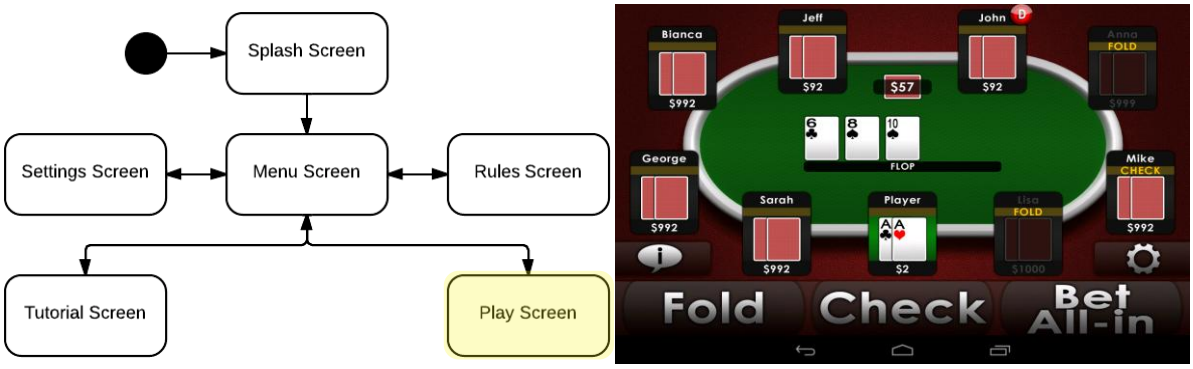


Figure 4-16: Play Screen

Figure 4-17: Play Screen - subsections

The last screen is the tutorial screen shown in Figure 4-18. This screen represents a full game play table similar to the play screen. One difference is that the tutorial screen executes a predefined scenario, which explains poker rules to the user in an interactive way. The other difference is that there is no information subsection available. This is because that information is redundant when the user follows a predefined scenario. However, the settings subsection is available and offers practical options like the adjustment of the game speed. In the middle of the screen is an instructions window navigates the user.

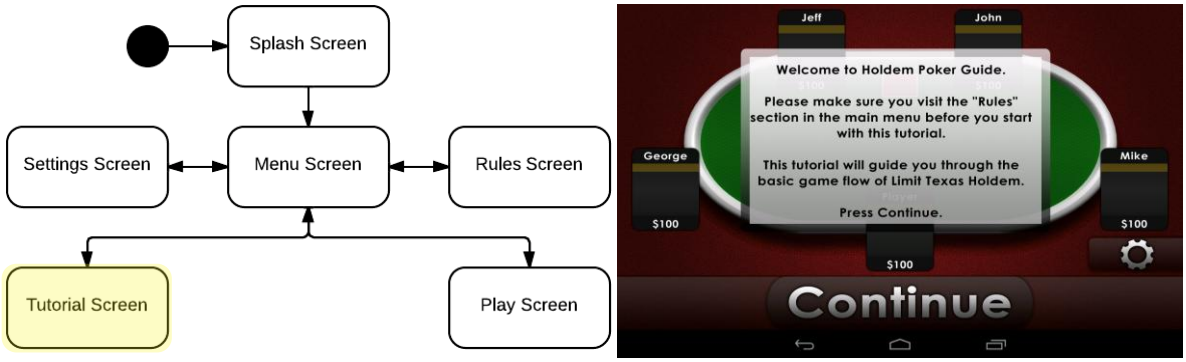


Figure 4-18: Tutorial Screen

When the Android back button is presses, or a table is about to be restarted, the user is asked for confirmation with a screen overlay shown in Figure 4-19.

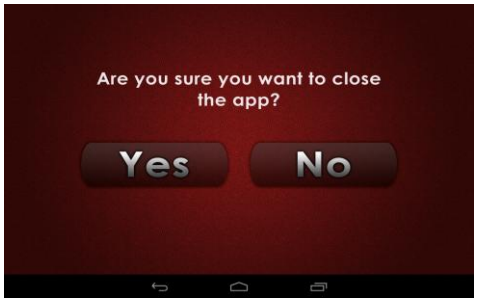


Figure 4-19: Confirmation Window

4.6. User interface design

The UI was designed to be used in landscape mode. The colour scheme is grey and dark red. All buttons have a suitable size to be easily accessible on a big screen (tablets), as on small screens (smartphones). While designing the graphics, care has been taken that every little detail is clearly visible on all supported screen sizes. I think that a good looking UI contributes a lot to a better user experience.

4.7. Summary

In this chapter, I talked about the high level design of the application and implementations details. The applied design patterns were explained, and the content of the application packages was shown in high level class diagrams. Low level class descriptions were also provided. I have discussed the statistics calculations algorithms and shown a complete application walkthrough.

Chapter 5 : Testing

The development process of my application was of incremental nature. One feature was developed and thoroughly tested once completed. This methodology allowed me to concentrate on one thing at a time and build up on solid ground when the next step had to be taken. The application development was also supported with unit tests. This guaranteed code integrity when changes were applied.

5.1. Portability

One of the non-functional requirements was portability. This requirement was matched to the fullest extent. The application was tested and is running on both testing devices with Android 2.3.5 and 4.2. The UI is fully optimised for smartphone devices and tablet devices with various screen sizes. In fact, the application is scalable to any logical screen size. Even when the UI gets rescaled, user input coordinates are translated correctly, and the application stays fully functionally. Furthermore, the LibGDX framework allows the application to be run on any system which is able to start up a java virtual machine. The application was also successfully run tested on Windows OS.

5.2. Game flow stability

The game flow stability was tested in the following way. The human player had the role of a spectator. A table with 8 computer players was set up. Each computer player got a large amount of money. Computer players successfully continued to play for 10 consecutive hours without any anomalies.

5.3. User acceptance testing

I have asked 6 people with different backgrounds and poker skills to test my application. The aim was to test if the application is fit for purpose and matches the use cases defined in 3.4. The received feedback was positive. Some suggested improvements were implemented. It seems like overall I have achieved the aims. Two example feedbacks are quoted below.

"I've never played poker before, but I found the app very easy to use. Tutorial and description of the rules were extremely helpful, and I found myself playing the game much longer than I planned! Large buttons and easy navigation added to a very smooth experience." - Student

"I have been playing poker over 2 years now, and I actually like the application overall. It provides a clear user interface and so that it's a real pleasure to use it on my tablet. I find it always exciting to guess which strategies computer players have in each round." - Professional

5.4. Efficiency testing

Another non-functional requirement was to have reasonable response times of the UI. This was tested on both testing devices. It was not a problem for the high-end device Google Nexus 7. However, the low-end device HTC Desire S takes approximately 1 second to switch to the heaviest (graphics) screen.

Both devices were also tested for user input response time. Button clicks and audible feedback happen almost immediately. The smartphone HTC Desire S shows a slight delay in processing user input. It responds with an average time of 0.3 seconds, which is still an acceptable response time for a low-end device.

The AI decisions should not take longer than 1 second. In fact, the modified version of the *HandEvaluator* class almost immediately returns the required value so that it takes no longer than 1 second to react for each computer player. In fact, a manual delay had to be introduced so that the pace of computer player responses adjusts to the game speed settings.

As specified in the requirements section, the total application size is expected to be around 13 megabytes. The application was packaged into one file with a size of 13 megabytes.

The memory efficiency requirement is satisfied. It was measured and does not exceed 13 megabytes.

5.5. Summary

This chapter reflected on the testing strategies applied to my project. Sample user feedback was provided, and testing procedures of the non-functional requirements were explained in detail.

Chapter 6 : Conclusions and recommendations

This chapter describes what I have learnt from this project. I will discuss the positives and negatives, talk about things that worked well and did not work well and what I would do different next time.

6.1. What I learned and achieved

Overall I am exceptionally pleased with the outcome of my project. It was the biggest personal software development project so far, and I have learnt a lot about poker and Android application development. Poker is a tremendously complex game, and I feel proud that I managed to implement a fully functional game play. On top of that, developing the educational features certainly helped me to understand the rules and different methods of card statistics calculations.

I was able to apply the knowledge gained from the Probability & Matrices and Software Risk Assessment university modules. More importantly, I believe that the Software Engineering module played a crucial role in my personal and academic progress as a software developer, which allowed me to complete this project on time and to my fullest satisfaction.

I am also extremely proud of the UI graphics. I put a lot of effort in designing the UI so that it looks qualitative, accurate and professional on devices with different screen sizes.

By using the MVC design pattern, I my application design became highly flexible. For example, the current source code of my application could be used as a framework for another card game. All required models are in place. The only thing I would have to add is the logic of the new game and the appropriate user interface of the play table. Most classes from the view package could be reused as they do not depend on the game logic. Furthermore, the majority of designed graphics like the card set and background images could be reused.

Certainly there is always room for improvement. However, the application does what it is supposed to do. It satisfies all functional and non-functional requirements. Personally, I consider this as my biggest academic achievement so far.

6.2. Challenges that I faced

When I started to work on my project, I did not have any knowledge about Android application development. Neither did I have much experience in playing poker. Developing an application which teaches other people how to play poker required me to understand poker rules to their fullest extent. The next step was to learn how to implement the logic of the game to a development framework which I never worked with before. The final product of this project proves that these challenges were achieved.

Another challenge I faced was to understand and implement the algorithms for hand strength and hand potential calculations. I experienced that the theory of poker statistics is a large field of studies. Furthermore, entire research groups from universities, like the Poker Research Group from University of Alberta [17], are actively developing new methodologies and artificial intelligence for computer players. Unfortunately, I was limited in time for this project and could examine only a limited range of algorithms.

As I already mentioned, poker is a tremendously complex card game. There are thousands of micro situations which could occur. As a very abstract calculation, we could take a table with 9 players where each player can act in 3 different ways at his turn at each of the 4 betting rounds gives us 9*3*4 = 108 possible situations. This number is very optimistic and assumes that the game play is straight forward, and each player acts only once. That almost never happens as players re-raise the bets of their opponents very often. It took me a long time to find a bug as the exact gameplay had to be manually reconstructed.

6.3. What would I add or do different

There are not many things that I would do different as I developed the application in the way I wanted it to be developed. However, I certainly would add more features to the application, if more time would be available. In fact, it is likely that I will continue working on this application after my graduation.

- **Multiple user profiles:** Most commonly, a smartphone device is used by a single person. However, I think that multiple user profiles would be a good feature for tablets. This could be useful when two people use the same tablet in a family.

- **Save/Load:** Though that some basic user data is stored, and a reset option for this data is provided, there is currently no way to save or load in the middle of a game session. This feature could be a valuable addition for my application. However, at the current stage the application should serve as a quick entertainment session, rather than an environment where long term goals have to be achieved.

- **Mode educational features:** There is a large number of educational features which could be added to my application. Some examples include tools where specific game situation can be set up or poker specific quizzes.

- **More game settings:** More game settings could be added. For example, the player could choose the number of opponents and how much money each of the opponents will have.

6.4. Summary

This section has reflected on my personal experiences. It described what I have learnt from this project and discussed the positives and negatives about my project. I also expressed my future plans regarding the application.

References

[1] www.thepokerbank.com. (2013, Apr.) www.thepokerbank.com. [Online]. http://www.thepokerbank.com/strategy/basic/starting-hand-selection/sklansky-groups/

[2] http://libgdx.badlogicgames.com/. (2013, Apr.) badlogicgames.com. [Online]. http://libgdx.badlogicgames.com/

[3] University of Alberta Computer Poker Research Group. (2013, Apr.) http://spaz.ca/poker/. [Online]. http://spaz.ca/poker/

[4] http://webdocs.cs.ualberta.ca/~games/poker/. (2013, Apr.) http://webdocs.cs.ualberta.ca/~games/poker/. [Online]. http://webdocs.cs.ualberta.ca/~games/poker/

[5] http://source.android.com/about/index.html. (2012, Nov.) About the Android Open Source Project. [Online]. http://source.android.com/about/index.html

[6] http://appleinsider.com. (2012, Nov.) iPhone outsold all Windows Mobile phones in Q2 2009. [Online]. http://appleinsider.com/articles/09/08/21/canalys_iphone_outsold_all_windows_mobile_phones_in_q2_2009.html

[7] http://www.microwave-eetimes.com. (2012, Nov.) Android powers 75% of smartphones in Q3. [Online]. http://www.microwave-eetimes.com/en/android-powers-75-of-smartphones-in-q3.html?cmp_id=7&news_id=222903359

[8] Stefan Brahler. (2012, Nov.) Analysis of the Android. [Online]. http://os.ibds.kit.edu/downloads/sa_2010_braehler-stefan_android-architecture.pdf

[9] Sebastian Kroop. (2012, Nov.) Evaluierung der Android-Plattform. [Online]. http://opus4.kobv.de/opus4-fhbrb/files/27/bachelorarbeit.pdf

[10] http://android-developers.blogspot.co.uk. (2012, Nov.) Announcing the Android 1.0 SDK, release 1. [Online]. http://android-developers.blogspot.co.uk/2008/09/announcing-android-10-sdk-release-1.html

[11] http://www.techradar.com. (2012, Nov.) Android 4.2 release date, news and features. [Online]. http://www.techradar.com/news/phone-and-communications/mobile-phones/android-4-2-release-date-news-and-features-1107255

[12] http://developer.android.com. (2012, Nov.) Platform Versions. [Online]. http://developer.android.com/about/dashboards/index.html

[13] http://www.businessweek.com. (2012, Nov.) Google Says 700,000 Applications Available for Android. [Online]. http://www.businessweek.com/news/2012-10-29/google-says-700-000-

applications-available-for-android-devices

[14] http://source.android.com. (2012, Nov.) Compatibility Program Overview. [Online]. http://source.android.com/compatibility/overview.html

[15] http://developer.android.com. (2012, Nov.) ADT plugin. [Online]. http://developer.android.com/tools/index.html

[16] http://www.pokerstars.com. (2012, Nov.) Poker Rules. [Online]. http://www.pokerstars.com/poker/games/rules/

[17] Darse Billings. (2006, Sep.) http://webdocs.cs.ualberta.ca/~games/poker/publications.html. [Online]. http://webdocs.cs.ualberta.ca/~games/poker/publications/billings.phd.pdf

[18] https://play.google.com. (2012, Nov.) Google Play Store. [Online]. https://play.google.com/store/apps

Appendices

This section includes the class diagrams, task list, Gantt chart and information about supporting materials.

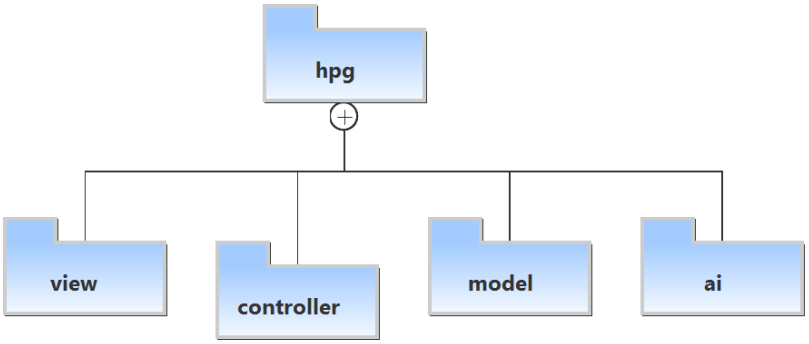
Supporting materials

The full source code and an executable file for the application can be found within the ***supporting material zip file*** under the following directories:

- **Source code:** see folder "./src"
- **Executable file:** see file "./hpg.jar"

Class diagrams

Application packages:

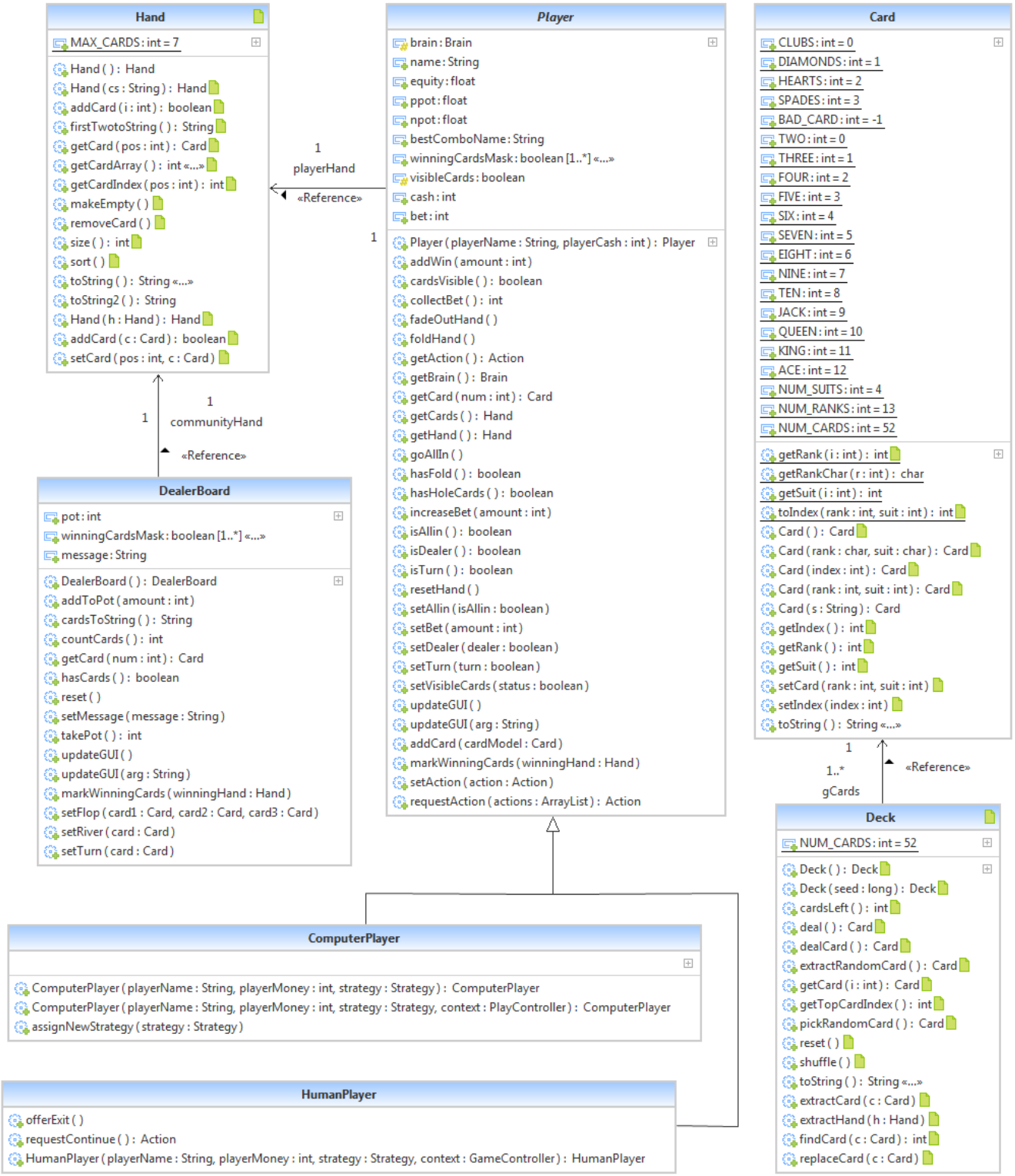


Appendix 1: Application packages

A specific class diagram with only public methods and public variables can be found below in Appendix 2, Appendix 3, Appendix 4 and Appendix 5. Full class diagrams with public and private elements can be found within the ***supporting material zip file*** under the following file names:

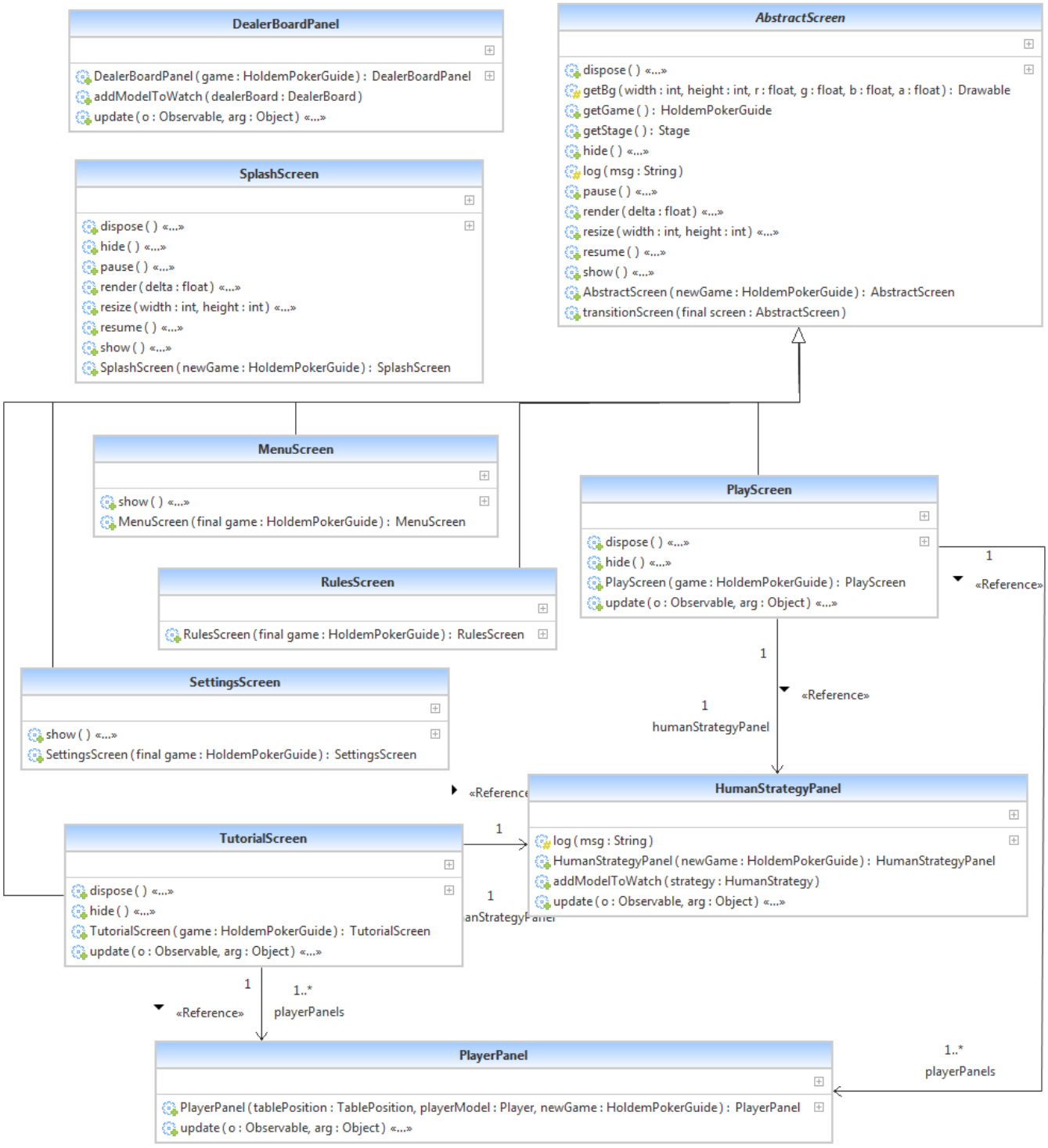
- **View package:** see file "./class_diagram_view_full.png"
- **Model package:** see file "./class_diagram_model_full.png"
- **Controller package:** see file "./class_diagram_controller_full.png"
- **AI package:** see file "./class_diagram_ai_full.png"

Model package class diagram showing all public variables and methods:



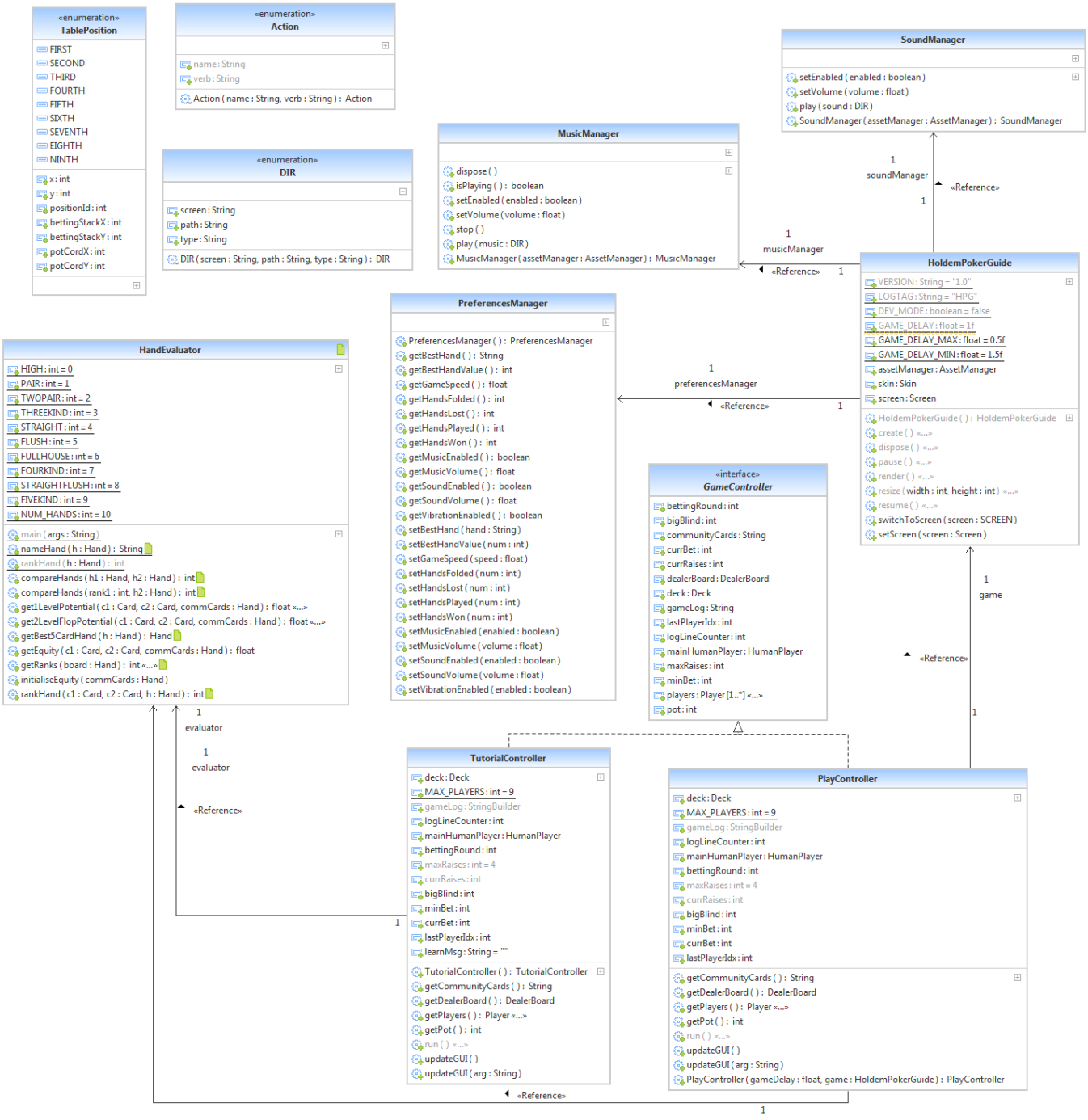
Appendix 2: Model package class diagram

View package class diagram showing all public variables and methods:



Appendix 3: View package class diagram

Controller package class diagram showing all public variables and methods:

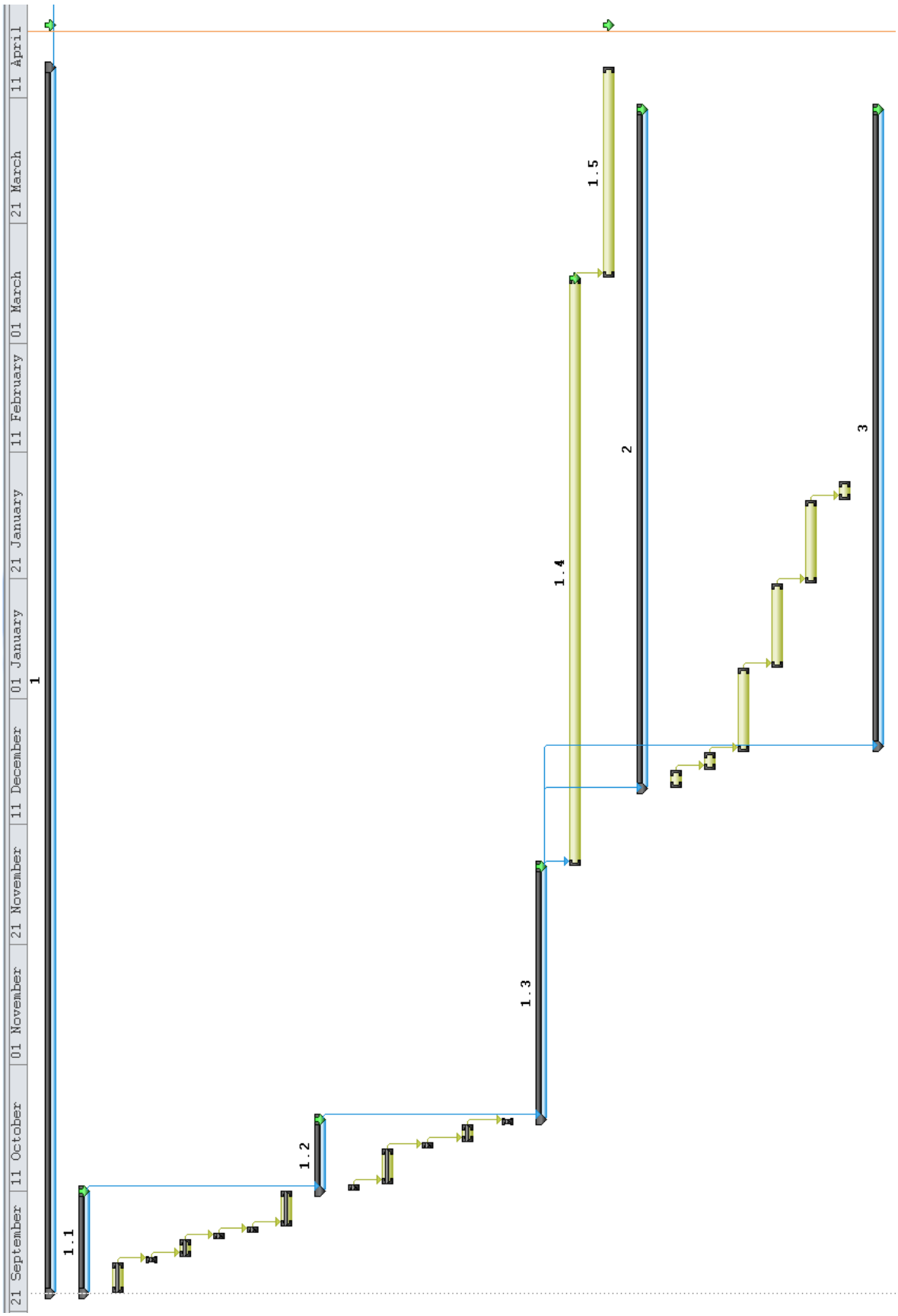


Appendix 4: Controller package class diagram

AI package class diagram showing all public variables and methods:

Appendix 5: AI package class diagram

Project Gantt chart:



Appendix 6: Gantt chart

WBS	ID	Task Name	Start	Finish	Deadline	Predecessors	Duration
1	**1**	**Writing**	**Mon 24/09/12**	**Mon 15/04/13**	**Mon 22/04/13**		**204 days**
1.1	**2**	**Supervisor/Project Choice**	**Mon 24/09/12**	**Wed 10/10/12**	**Wed 10/10/12**		**17 days**
1.1.1	3	Research on available projects	Mon 24/09/12	Fri 28/09/12	NA		5 days
1.1.2	4	Create a list with interesting projects	Sat 29/09/12	Sat 29/09/12	NA	3	1 day
1.1.3	5	Communicate with potential supervisors	Sun 30/09/12	Tue 02/10/12	NA	4	3 days
1.1.4	6	Decide on a project	Wed 03/10/12	Wed 03/10/12	NA	5	1 day
1.1.5	7	Visit the supervisor and apply for adoption	Thu 04/10/12	Thu 04/10/12	NA	6	1 day
1.1.6	8	Get officially adopted and start to specify the project idea	Fri 05/10/12	Wed 10/10/12	NA	7	6 days
1.2	**9**	**Specifications**	**Thu 11/10/12**	**Mon 22/10/12**	**Mon 22/10/12**	**2**	**12 days**
1.2.1	10	Develop the basic document structure	Thu 11/10/12	Thu 11/10/12	NA		1 day
1.2.2	11	Specify the technical details of the project and send the first draft to prof. Fenton	Fri 12/10/12	Wed 17/10/12	NA	10	6 days
1.2.3	12	Meeting with all other project students supervised by prof. Fenton	Thu 18/10/12	Thu 18/10/12	NA	11	1 day
1.2.4	13	Use the feedback and the proposed report structure	Fri 19/10/12	Sun 21/10/12	NA	12	3 days
1.2.5	14	Submit the project specifications	Mon 22/10/12	Mon 22/10/12	NA	13	1 day
1.3	**15**	**Interim report**	**Tue 23/10/12**	**Mon 03/12/12**	**Mon 03/12/12**	**9**	**42 days**
1.3.1	16	Await Feedback for project specs	Tue 23/10/12	Fri 26/10/12	NA	9	4 days
1.3.2	17	Understand and practice TH rules	Sat 27/10/12	Sun 28/10/12	NA	16	2 days
1.3.3	18	Find out about existing applications	Sun 28/10/12	Sun 28/10/12	NA		1 day
1.3.4	19	Find out which elements could be reused	Mon 29/10/12	Wed 31/10/12	NA		3 days
1.3.5	20	Write a chapter about Android OS	Mon 29/10/12	Sun 04/11/12	NA	18	7 days
1.3.6	21	Write a chapter about poker	Mon 05/11/12	Sun 11/11/12	NA	20	7 days

1.3.7	22	Identify the core game features and create requirements	Mon 12/11/12	Sun 18/11/12	NA	21	7 days
1.3.8	23	Write the risk assessment	Mon 19/11/12	Sun 25/11/12	NA	22	7 days
1.3.9	24	Finalise the interim report according to requirements	Mon 26/11/12	Sun 02/12/12	NA	23	7 days
1.4	25	Draft report	Tue 04/12/12	Mon 11/03/13	Mon 11/03/13	15	98 days
1.5	26	Final report	Tue 12/03/13	Mon 15/04/13	Mon 22/04/13	25	35 days
2	**27**	**Coding**	**Mon 17/12/12**	**Mon 08/04/13**	**Mon 08/04/13**	**15**	**113 days**
2.1	28	Go through the Android SDK tutorials	Mon 17/12/12	Wed 19/12/12	NA		3 days
2.2	29	Implement the first prototype	Thu 20/12/12	Sat 22/12/12	NA	28	3 days
2.3	30	Implement the core game features	Sun 23/12/12	Sat 05/01/13	NA	29	14 days
2.4	31	Develop and implement the AI	Sun 06/01/13	Sat 19/01/13	NA	30	14 days
2.5	32	Implement the didactic features	Sun 20/01/13	Sat 02/02/13	NA	31	14 days
2.6	33	Revise the coding and make efficiency adjustments	Sun 03/02/13	Tue 05/02/13	NA	32	3 days
3	**34**	**Testing**	**Mon 24/12/12**	**Mon 08/04/13**	**Mon 08/04/13**	**15**	**106 days**
3.1	35	Define test objectives and tasks	Mon 24/12/12	Mon 24/12/12	NA		1 day
3.2	36	Define the testing scope	Mon 24/12/12	Mon 24/12/12	NA		1 day
3.3	37	Define the testing strategies	Mon 24/12/12	Mon 24/12/12	NA		1 day
3.4	38	Define the hardware requirements	Mon 24/12/12	Mon 24/12/12	NA		1 day
3.5	39	Test cycle 1	Mon 31/12/12	Mon 31/12/12	NA		1 day
3.6	40	Test cycle 2	Mon 28/01/13	Mon 28/01/13	NA		1 day
3.7	41	Test cycle 3	Mon 25/02/13	Mon 25/02/13	NA		1 day
3.8	42	Test cycle 4	Mon 25/03/13	Mon 25/03/13	NA		1 day
3.9	43	Test cycle 5	Mon 08/04/13	Mon 08/04/13	NA		1 day
4	44	Viva	Mon 29/04/13	Fri 07/06/13	Fri 07/06/13	1	40 days