Lab 4

CSE 438

Spring 2020

1 Setup

Navigate to the repository for this assignment and make a copy for yourself. To submit your project, commit and push your code to the repository by the due date.

2 Introduction

After successfully making a great dice simulator in Lab 2, you have convinced your friend to not drop out of college to become a professional craps player. Unfortunately, your friend now wants to become a world champion blackjack player. You, an even better educated computer science student with an even stronger understanding of discrete math and probability realize how bad of an idea this is. To help convince your friend to stay in school, you will once again be using your skills in mobile app development, but this time to make a blackjack game using Firebase, gestures, and animations.

3 Blackjack Rules

For this assignment, we will be using these rules: There will be two players, a player (referred to as Player) and a dealer (the computer). At the beginning of the game two cards will be dealt to both players. Both cards dealt to the Player will be dealt face up. One of the cards dealt to the dealer will be left face down. The Player will then choose what move they want to make. The Player can choose to hit to receive a new card, or stand so that it is the dealer's turn. If the Player's total goes above 21, they will bust and the dealer will win. If the Player is able to stand, the turn goes to the dealer. The dealer's moves are controlled by certain rules. If the dealer was dealt 21, they will automatically win the game. If the dealer's total is above 17, they will stay, and the resulting winner will be whoever has a larger total. If the dealer has below 17, they will keep hitting until they have a total above 17, or they bust. All face cards have a value of 10, and an Ace can change between 11 and 1 depending on the current total. Cards values are based on the number visible on the card. All royal cards (jacks, queens, and kings) are worth ten points and Aces are worth EITHER 11 or 1.

4 Dealing the cards

You will be given the cards as images in the drawables resource folder. There will also be one image for the back of a card which you can use for the dealers face down card and the deck. You will be given a class called CardRandomizer. The CardRandomizer object contains one method that will return a list of all of the ID's for the cards in the resource folder. Here is an example of generating a random card:

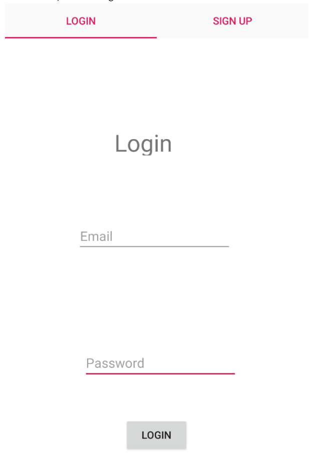
```
val randomizer: CardRandomizer = CardRandomizer()
var cardList: ArrayList<Int> = randomizer.getIDs(this) as ArrayList<Int>
val rand: Random = Random()
val r: Int = rand.nextInt(cardList.size)
val id: Int = cardList.get(r)
val name: String = resources.getResourceEntryName(id)
```

To display the cards, you will need to pass the card id to an ImageView. There are a couple of different ways you can set this up:

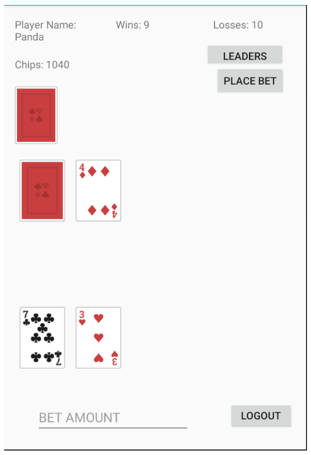
- One approach would add some ImageViews to the layout ahead of time. Once you want to deal a card,you would grab the next image view, set the approprate image resource using the id, then move the card to the appropriate position.
- You can also dynamically create ImageViews in your code. In order to do this, you need to make sure to add the ImageView to the layout after it is created.
-

5 Assignment

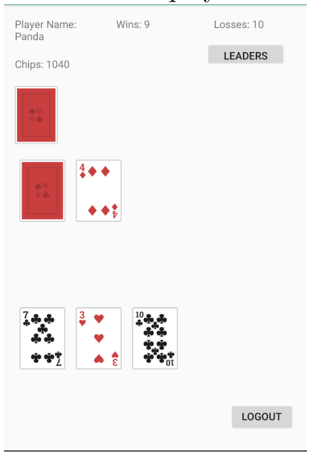
When the app is started, the app should direct the user to a login screen. From there they can login to their account using a valid email. The Firebase login feature should be helpful here. If the user does not have an account, they should be able to register to make a new one. Below is an example of a login screen.



Once the user inputs a valid login, they should stay logged in in between uses of the app so long as they do not logout of the app. The user should then be directed to a screen that will allow them to play blackjack. Upon logging in a new game should start and two cards should be dealt to the player and to the computer. Both of the players cards should be face up and one of the dealers cards should be face up. The cards should be animated to move from a deck of face down cards or off screen to appropriate places on the screen. Upon being dealt the cards, the user should be prompted the ability to bet chips. The user can bet any number of chips they currently have on the game that they will win. They can also bet 0 chips and play a practice game. An example of the the starting screen is shown below after the cards have been dealt and the player is in their betting phase. Note that the chip total is always on display and updated relative to what the player bets.

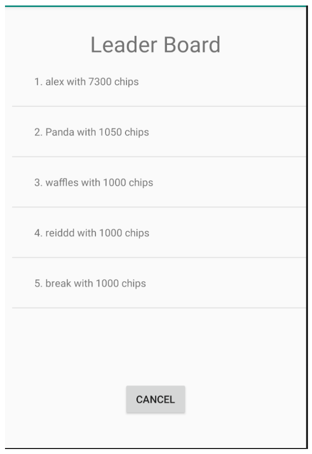


Once the player has placed a bet, their turn begins. The player has the option to "hit" and receive another card by double tapping the screen. When this happens, another card should be moved from the deck to the player's hand using an animation. This can be done continuously until the player busts. Remember that hands can get very long in blackjack. Make sure you account for cases where a second row of cards may be needed for either player. An example of a hand with an extra card is shown below.



The player's turn ends when the player swipes right on the screen or the player's total goes over 21. The computer's turn then begins if the player did not bust. The computer should continuously "hit" until their card total is at least 17 at which point they will "stay". The winner of the game is determined by the player with the total closest to but not over 21. When the winner is determined, the player's chips should be updated. If they lost, they receive no chips back, but if they won, whatever they bet is doubled and returned to them. It is up to you what happens when and if a player runs out of chips. When a game ends, a screen should be reset, updated, and a new game should be re dealt automatically.

When the user presses the leader button, they should be taken to a leader board that sorts ALL users based on their current chip total with the player with the most chips at the top of the leader board. You will want to use a RecyclerView and adapter for this, although LastAdapter may make your life a bit easier if you choose to use it.



1. alex with 7300 chips
2. Panda with 1050 chips
3. waffles with 1000 chips
4. reiddd with 1000 chips
5. break with 1000 chips

6 Last Adapter

Are you sick or writing adapters and RecyclerViews yet? If not, keep writing them by hand, its important to know how to. However, if you are sick of them, you may want to read this article which shows how to use a Kotlin library to create RecyclerViews with significantly less code than we have done previously.

7 Creative Portion

For every homework assignment, you will be asked to think of an additional feature to be added to the application that will improve the user experience and provide you an opportunity to learn about concepts that you are personally interested in. Put yourself in the shoes of your users: what features would they like to see in an app like this? Try to make it something new and substantially different from what the app already does - do not just rehash existing requirements. Firebase has a lot of cool features that we have not explored: social media authentication, machine learning, real time database, cloud functions to name a few. Additionally, the Android OS has a lot of features we have not looked at yet: cameras, sensors, additional gestures and there are a plethora of Kotlin and Java libraries online available to you all. Please use some of them. When you submit your assignment, please include a ReadMe.txt file that explains your creative portion. You should explain what the feature is, why you chose to implement that particular feature, and how you went about implementing it.

8 Rubric 100 Points

- (10 points) Creative portion
- (10 points) Player can login and player data is stored in Firebase as information unique to the user
- (10 points) Players have a running amount of chips that they can bet on games. If they win, the amount they put in is doubled, but if they loose, they loose the money they bet. What happens when the player runs out of chips (if that is possible) is up to you
- (10 points) A leader board is shown in a separate tab or activity and is consistent among installations
- (10 points) Cards being dealt are animated, and all cards are visible on every hand with none of the cards being off screen
- (5 points) Games can be bet on, but do not have to be. Zero is an acceptable bet, but bets should not bet negative or beyond the amount of chips a user currently has
- (5 points) Player's chip counts are displayed on startup and updated

- (5 points) The Player receives two cards face up, and the dealer receives one card face up and one card face down when the game starts
- (5 points) Swiping right allows the Player to hit
- (5 points) Double tapping allows the Player to stand
- (5 points) If the Player goes over 21, they automatically bust and a new game starts automatically
- (5 points) The dealer behaves appropriately based on the rules described above
- (5 points) Once the game is complete, the winner should be declared and the Firebase database should be updated appropriately
- (5 points) The player is able to bet before they ask for additional cards
- (5 points) Player can logout of the app and their data is preserved