Contributor - Qi Liu

- App startup introduction slides
- Use popup dialog to show game rules for players
- Implement Game Rules
    - GameState
    - Player
- Use JUnit to test functions

App Introduction

```
boolean isFirstStartup =
readStartupFlagFromDisk();
Log.d(TAG, "IsFirstStartup: " + isFirstStartup);

if (isFirstStartup) {
   addSlides();
} else {
   setContentView(R.layout. activity_intro);
   Intent myIntent = new
Intent(IntroActivity. this, LoginActivity. class);
   IntroActivity. this.startActivity(myIntent);
   finish();
}
```

```
private void writeStartupFlagToDisk() {
   SharedPreferences sharedPref =
this.getSharedPreferences( "preference_file_key",
this.MODE_PRIVATE);
   SharedPreferences.Editor editor =
sharedPref.edit();
   editor.putBoolean( "isFirstStartup", false);
   editor.commit();
}
private boolean readStartupFlagFromDisk() {
   SharedPreferences sharedPref =
this.getSharedPreferences( "preference_file_key",
this.MODE_PRIVATE);

   return sharedPref.getBoolean( "isFirstStartup",
true);
}
```

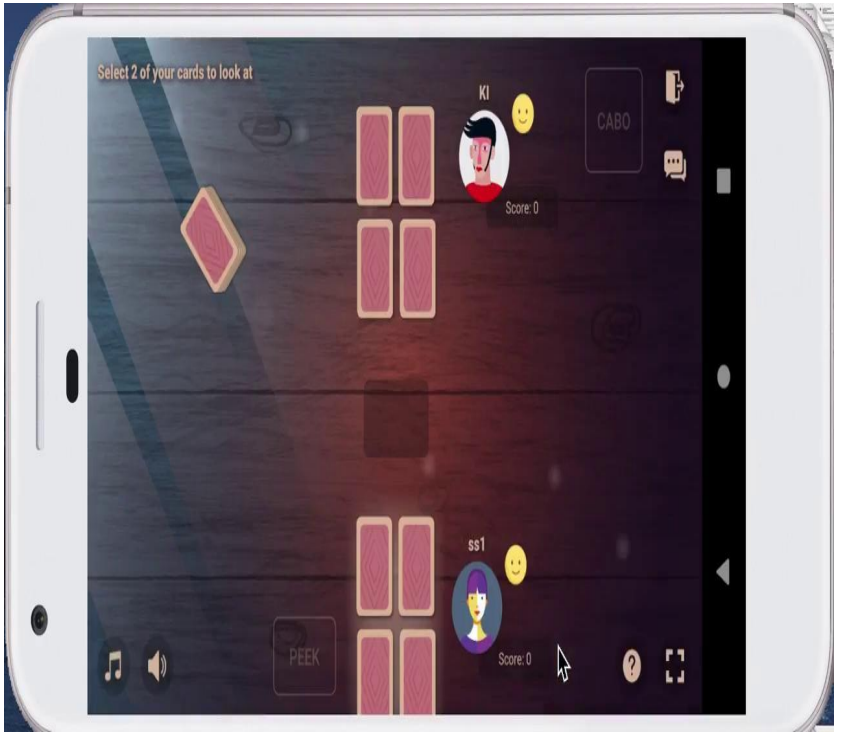
App Introduction

```
boolean isFirstStartup =
readStartupFlagFromDisk();
Log.d(TAG, "IsFirstStartup: " +
isFirstStartup);

if (isFirstStartup) {
   addSlides();
} else {
   setContentView(R.layout.activity_intro);
   Intent myIntent = new
Intent(IntroActivity.this,
LoginActivity.class);

IntroActivity.this.startActivity(myIntent);
   finish();
}
```

SharedPreferences,
AppIntro library

Popup Dialog for Game Rules

```
private AlertDialog.Builder builder = null;
private AlertDialog ruleDialog = null;

builder = new AlertDialog.Builder(this);
builder
        .setTitle("Cheat Sheet")
        .setMessage(msg)
        .setCancelable(false)
        .setPositiveButton("Got it", new
DialogInterface.OnClickListener()
        {
            public void onClick(DialogInterface dialog, int id)
            {
                dialog.cancel();
            }
        })
;
ruleDialog = builder.create();
ruleDialog.getWindow().setBackgroundDrawableResource(R.color.beige);
```



AlertDialog

Player - Overview

Class Members

```
// The cards belong to this player
private ArrayList<Card> cards = new ArrayList<>();
@JsonIgnore
private Gamestate gamestate;
@JsonIgnore
private boolean calledCabo = false;
private int score = 0;
```

Class Functions

```
public void drawCard();
public void tryDiscardCards(ArrayList<Card> cardsToRemove);
public void swapWithOtherPlayer(Player other, Card myCard, Card otherCard)
public void swapWithAvailableCards(Card myCard, Card otherCard);
public void swapWithDiscardedCards(Card myCard, Card otherCard);

public int calculatePoints();

public void setCalledCabo(boolean calledCabo);
public void callCabo();
public void setScore(int score);
public int getScore();
```

Player - Discard Cards

```java
public void tryDiscardCards(ArrayList<Card> cardsToRemove) {
    if (this.calledCabo) return;

    Card card = cardsToRemove.get(0);
    for (int i = 1; i < cardsToRemove.size(); i++) {
        if (card.getValue() != cardsToRemove.get(i).getValue()) {
            // not equal, return
            return;
        }
    }

    // Remove all of them
    for (int i = 0; i < cardsToRemove.size(); i++) {
        this.gamestate.getDiscardedCards().add(cardsToRemove.get(i));
        this.cards.remove(cardsToRemove.get(i));
    }
}
```

Game Rule ! Make sure all the cards to be removed are the same.

Player - Swap Cards

```java
public void swapWithOtherPlayer(Player other, Card myCard, Card otherCard) {
    if (this.calledCabo) return;

    for (int i = 0; i < other.cards.size(); i++) {
        if (otherCard.equalsCard(other.cards.get(i))) {
            other.cards.set(i, myCard);
            break;
        }
    }
    for (int i = 0; i < this.cards.size(); i++) {
        if (myCard.equalsCard(this.cards.get(i))) {
            this.cards.set(i, otherCard);
            break;
        }
    }
}
```

Game Rule ! Make sure the cards to be swapped are owned by the players

GameState - Overview

Class Members

```
// The cards in available pile, face-down, never exposed these cards to clients
private ArrayList<Card> availableCards = null;
// The cards drawn from available card pile, they may be in discarded card pile, clients'
decks.
private ArrayList<Card> playedCards = null;
// The cards discarded by clients, face-up
private ArrayList<Card> discardedCards = null;
```

Class Functions

```
public void generateCards(boolean shouldShuffle);
public void distributeCardsAtBeginning();
public void calcScores();
```

GameState - Calculate Scores

```
public void calcScores() {
// Case1: Checking the special case, (0, 0, 13, 13)
…
for (Player  player : players.values()) {
   if (player.getId() !=  player.getId()){
        player.setScore( player.getScore() + 50);
       if ( player.getScore() == 100 ||  player.getScore() == 50){
           _player.setScore(_player.getScore() / 2);
       }
       if ( player.getScore() >= maxPoints) {
           this.terminate();
       }
   } else {
       if (player.getScore() == 50) {
           player.setScore(player.getScore() / 2);
       }
   }

}
```

… **Game Rule !** Other players get 50 score in this round.

GameState - Calculate Scores

```
public void calcScores() {
// Case 2: common case
…
      for (Player player : players.values()) {
         if (player.getCalledCabo()) {
             if (smallestPoint != player.calculatePoints()) {
                 player.setScore(player.calculatePoints() * 2 + player.getScore());
             } else {
                 // get zero score this time
             }
         } else {
             player.setScore(player.getScore() + player.calculatePoints());
         }
         if (player.getScore() == 100 || player.getScore() == 50) {
             player.setScore(player.getScore() / 2);
         }
         if (player.getScore() > maxPoints) {
             this.terminate();
         }
      }
…
}
```

Game Rule ! The player called cabo get 0 (if he won) or double score (if he lost)

JUnit

```
@Test
public void testDrawCard() {
    this.tony.drawCard();
    ArrayList<Card> cards = this.tony.getCards();
    assertEquals(cards.size(),1);

    Card drawnCard = this.gamestate.getPlayedCards().get(0);
    Card tonyCard = cards.get(0);
    assertEquals(drawnCard, tonyCard);

    Card topCardOnAvailableCards = this.gamestate.getAvailableCards().get(0);
    //Does tony really draw the first Card from AvailableCards?
    assertNotEquals(tonyCard, topCardOnAvailableCards);
}
```