



## **Interim Report:** Rekop Poker

James Scully (4304469)  
psyjs20@nottingham.ac.uk  
G400 Computer Science

**Project Supervisor:** Milena Radenkovic

## Introduction

When one looks for a casual poker game that offers power to the actual end user, rather than attempting to fish money it can be quite difficult to find.

## Motivation

Rekop poker aims to provide an alternative to other multi-player applications on the Android app store as many of these prevent user customization and introduce in-app purchases, with no ability for players to play locally on their own network.

## Related Work

## Description

## Methodology

Test-Driven Development has been essential so far in the development of the poker algorithm. Evaluating Texas Hold'em hands has many vectors of errors and being able to write tests before and alongside implementation helped out immensely.

Take this snippet from the testing whether a hand is a straight or not:

```
assertTrue(new TexasEvaluator("0D 0D 2D KD AD QD JD").isStraight());
assertTrue(new TexasEvaluator("AD KD QD JD 0D QD JD").isStraight());
assertTrue(new TexasEvaluator("0D 2D 0D 9D 8D 7D 6D").isStraight());
assertTrue(new TexasEvaluator("2D 3D 4D 5D 6D QD JD").isStraight());
assertTrue(new TexasEvaluator("0D 0D 2D KD AD QD JD").isStraight());
assertTrue(new TexasEvaluator("KD QD JD JD JD 0D 9D").isStraight());
/* Falses */
assertFalse(new TexasEvaluator("0D 0D 0D 0D AD QD JD").isStraight());
assertFalse(new TexasEvaluator("2D 5D 9D JD 0D QD JD").isStraight());
assertFalse(new TexasEvaluator("kD 2D 0D 4D 8D 7D 6D").isStraight());
assertFalse(new TexasEvaluator("2D 3D 9D 3D 4D QD JD").isStraight());
```

Regarding the vectors of error, having instantenous feedback, whereby it was easy to debug providing breakpoints, the actual outcome and generally a sense of direction.

## Design

### Overview

The project itself is being broken down into three compartments - a backend that satisfies the core Texas Hold'em gamemode, the server which will provide the connections needed for players to play together on their own networks and the Android app, which links both of these together with the graphical interface.

## Backend

The backend is written solely in Java as this allows for good interoperability with multiple other operating systems - Android being the main one. It has been broken down in such a way that game modes other than Texas Hold'em can be added to it, enabling further development down the line. For example, we've created data classes for Faces and Suits, each containing values for easy comparison. Furthermore, we've created a Deck class that allows for the user to pull a random card from a standard 52-card deck and ensure that it hasn't been pulled before. These building blocks allow for future development by providing the basics needed in a standard card game.

## Algorithms

**Hand Strength** - The algorithm for hand-strength is not very efficient, and as such is comprised of simply checking a sorted array of cards to see whether they meet a certain condition. This has a performance impact as we often make repeated calls

**Win evaluation** -

## Implementation

### Progress

### Management

Management has been key in this project, of which I have not performed greatly. Time has been spent on modules which are lesser credits and although that coursework must be done, it shouldn't take as much time as it does. To combat this, I've essentially tried to keep doing atleast one task a day - whether it be debugging a function or atleast analysing what the error is, or writing a function.

## Contributions and Reflections

***Reflect - How hard evaluating a hand of poker can be; was not a simple process and underestimated. I.e. Talk about how detecting whether a straight may not be the most powerful, if a flush exists.***

A key goal of all software is that it must be efficient - both in terms of code complexity and performance. Upon undertaking this project it was naively assumed that, given how easy it is to recognize the outcome of a poker round in reality, it musn't be too difficult to implement efficiently through code.

Evaluating poker hands efficiently becomes very complex and convoluted once the amount of hands possible is realised and how determining the outcome of some hands can be tricky.

Reflecting upon how I assumed Test-Driven Development would take much of our time before and during up and that we did not have "too much time to write tests and develop a fully-functional program", I couldn't have been further from the truth. Having instant feedback on whether it passed, what the result was and the ability to debug into it was essential.

Although, initially I was under the assumption that each test case was going to have to be made up of newly created objects. Frustrated that test cases were taking too long and a nuisance to write, I created a factory pattern within the evaluator class itself to resolve, for example, "0D 0D 2D KD AD QD JD" into an actual hand + table.

I believe that the examples below really do speak for themselves.

```
TexasHand FLUSH_FOK = new TexasHand(  
    new Card(Suit.CLUBS, Face.FOUR),  
    new Card(Suit.CLUBS, Face.FOUR),  
    new Card(Suit.CLUBS, Face.FOUR),  
    new Card(Suit.CLUBS, Face.FOUR),  
    new Card(Suit.CLUBS, Face.ACE)  
);  
...  
public void isFlush() {  
    assertTrue(FLUSH_FOK.isFlush());  
}
```

**New:**

```
assertTrue(new TexasEvaluator("AD KD JD QD 0D QS JC").isRoyalFlush());
```

## Bibliography