# Homework 2: Texas Hold'em
# COSC150-02 Spring 2017

Assigned: 1/31/2017
Due: 2/14 at 11:59pm ET

## Description

The goal of this homework is to familiarize students with Java programming and JUnit testing. For this assignment, you will write a program that calculates the winner of a Texas Hold'em game, given a provided board and a set of players (and their cards).

In a Texas Hold'em game, the board contains five **community cards** and each player has two **pocket cards**. Each card has a rank among A, K, Q, J, 10, 9, 8, 7, 6, 5, 4, 3, 2; cards don't have suits (i.e., spade, heart, club, and diamond) in this simplified game. Each player plays the best hand they can make from the seven cards comprising their two pocket cards and the five community cards. A player may use both of their own two pocket cards, only one, or none at all, to form the final five-card hand.

The ranking of a hand is as follows (with ranked examples for each type):

- **Four of a kind**. Example: [7 7 7 7 K] > [7 7 7 7 3] > [3 3 3 3 A]
- **Full House**. Example: [5 5 5 K K] > [5 5 5 3 3] > [3 3 3 A A]
- **Straight**. Example: [7 8 9 10 J] > [3 4 5 6 7]
- **Three of a kind**. Example: [5 5 5 A K] > [5 5 5 A 7] > [3 3 3 A K]
- **Two pairs**. Example: [A A 3 3 10] > [A A 3 3 7] > [K K Q Q A]
- **Pair**. Example: [A A 9 6 3] > [A A 9 4 3] > [Q Q A K J]
- **None**. Example: [K Q 10 7 3] > [K Q 7 6 5] > [Q 10 8 5 4]

Consider a sample setting:

- Board community cards: 4 K 4 8 7
- Bob: A 4
- Carol: 7 8
- Ted: K K
- Alice: 5 6

Bob's best hand is [4 4 4 A K]
Carol's best hand is [7 7 8 8 K]
Ted's best hand is [K K K 4 4]
Alice's best hand is [4 5 6 7 8]

In this case [K K K 4 4] > [4 5 6 7 8] > [4 4 4 A K] > [7 7 8 8 K]. So Ted is the winner.

**Part 1. Implementing Texas Hold'em.**

In part 1, you are asked to develop from the provided skeleton code to complete the implementation of a Texas Hold'em game. There are four major functions:

- *int Hand.compareTo(Hand otherHand).* It computes how the current hand compares to a given otherHand, and returns 1 if the current hand wins against otherHand, -1 if it loses, or 0 if there is a tie. **Hint: Try not to write all code within this single function; use other functions specified in the skeleton code.**

- *void Game.readSettings(String filename).* It reads board and player settings from an input file and set the corresponding member fields (i.e., board and players). The file is in the following format: The first line contains five integers, representing the ranks of five community cards; the second line contains one integer, indicating the number of players; then each of the following line contains a string and two integers, representing the name and the two pocket cards of each player.

- *ArrayList<Hand> Game.generateCandidates(ArrayList<Card> board, Player player).* It computes all possible hands of the given player, that is, it computes all possible 5-card combinations from the available 7 cards (5 from board's community cards, and two pocket cards of the player). The results are returned as an ArrayList.

- String Game.announceWinner(). It decides who is the winner of the game, based on the configuration of the current game. It returns "<playerName> wins" if there is a single winner, or "<playerName1> <playerName2> … <playerNameK> tie" if there is a tie. Use single space between in the output line.

**Part 2. Test with JUnit.**

In part 2, you are asked to write a JUnit test fixture to test the four functions specified above. **Please write all your code in a single file TestGame.java.** We will grade by using your JUnit code to check intentionally buggy reference implementations.

## What to turn in

You will turn in a single zip file that contains TestGame.java, Game.java, Hand.java, along with any other supporting .java files (for example, the Player.java and Card.java programs, and any other classes that you created). Your .zip file should not have any directories; that is, everything should be in the "root" of the .zip file.

If you have trouble creating .zip files, please post a note on Piazza.

Once you have created a .zip file that contains your code (again, in the top (and only) directory of the zip file), upload it to Autolab at https://autolab.georgetown.edu/. In Autolab, look for the "TexasHoldem" assignment.

## Where to turn it in:

Turn in the file using Autolab at https://autolab.georgetown.edu/

Where to go for help:
For questions about this assignment, please post to Piazza at
https://piazza.com/class/ixumh06s7vo2ez