INTRODUCTION TO COMPUTER SCIENCE II
PROJECT 1
Poker Hands

We are going to create classes to represent both standard playing cards and decks of those cards. With such objects, you will use code of mine to draw hands of five cards, and then test those hands to see what they hold (e.g., full house, flush, three of a kind). By doing so repeatedly, we'll try to measure the frequency with which each type of hand occurs.

1 Poker hands

The card game of poker—which is really a family of card games—is widely played and simple to explain.[1] Given a standard deck of 52 cards, shuffled, we want to examine what kinds of poker hands we would expect to be dealt.

This assignment addresses those questions, all while exercising our ability to decompose the problem into a set of interacting objects.

2 Getting started

If you are doing your work on `remus/romulus`, login with Remote Desktop, open a terminal window, create a directory for your work, and change into it. Next, grab the initial source code for this assignment:

- **On `remus/romulus`:** Copy the source code with the following command...

```
$ cp ~sfkaplan/public/COSC-112/project-1/PokerHand.java .
```

- **On your own computer:** Download the source code by clicking here.

Examine the code in `PokerHand.java`. Notice that it assumes the existance of a `Card` class and a `Deck` class. These two classes are described below, in Section 3.1. Additionally, there are a number of methods, named things like `hasPair()` and `hasFullHouse()` for which no bodies have yet been written.

3 Your assignment

In short, **write the `Card` and `Deck` classes, and complete the methods of the `PokerHand` class**. Below is some more detail...

[1] Playing it well, however, is quite challenging. That, however, is not our concern here.

3.1 New classes

The `Card` class : This entire exercise depends on representing and manipulating playing cards. Thus, each `Card` object should hold the two characteristics that define a playing card, namely its *suit* and its *rank*. The suits are *spades, hearts, clubs,* and *diamonds*, while the ranks are *ace, 2* through *9, 10, jack, queen,* and *king*.

You have some latitude for deciding how each `Card` stores its information. However, it should be possible to do the following with your `Card` objects:

- `public Card (String suit, char rank)`
  Where the `suit` is given as the suit name, and the `rank` is one of `'A', '2', '3', ..., '9', 'T', 'J', 'Q', 'K'}`.
- `public String getSuit ()`
  Return the suit as provided in the constructor.
- `public char getRank ()`
  Return the rank as provided in the constructor.

You may want to add methods to this set of capabilities.

The `Deck` class : A deck contains 52 cards, one of each combination of the four suits and thirteen ranks. It should be capable of doing the following:

- `public Deck ()`
  Create a new deck with 52 cards.
- `public void refill ()`
  Reset the deck to its initial 52 card state, just as it was after being constructed.
- `public void shuffle ()`
  Randomly permute the order of the cards in the deck.
- `public Card draw ()`
  Remove and return one card from the top of the deck.[2]

You must write these two classes to fulfill what is described above and what the `PokerHand` class relies upon.

3.2 Completing the `PokerHand` class

You will notice, inside this class, that there are a number of methods with names such as `hasPair()` and `hasFullHouse()`, each of which returns a `boolean` result that indicates whether the proposition, as implied by the method name, is true.

You must complete these methods such that a `PokerHand` object can determine whether it contains these card combinations. The `main()` method in this class assumes that these methods work.

[2]Implied in many of these methods is that the deck represents not only a collection, but also an order to the cards. Thus, a deck must keep track of where the "top" of the order is.

4 How to submit your work

Use the CS submission systems to submit your work. Specifically, you will need to submit your `PokerHand.java`, `Deck.java`, and `Card.java` source code files. You may use either of the following two methods, while connected to `remus` or `romulus`, to use the submission system:

- **Web-based:** Visit the submission system web page.
- **Command-line based:** Use the `~lamcgeoch/submit` command at your shell prompt.

This assignment is due on Sunday, Oct-16, 11:59 pm.