Project 4 – Collectively, how do they rank?

Objectives:

- Practice the new stuff: OOP - constructors, instance variables, instance methods, recursion
- Practice the old stuff: helper methods, information hiding, refactoring, good variable and function names, unit testing, debugging

Overview

You will continue modeling the game of poker, but we're going to make a version that's closer to the popular Texas Hold'em. This will entail one main difference: Instead of normal 5-card poker hands, we will have **community cards** (five cards that all hands can use) and **hole cards** (2-card **stud poker hands**) from which to make the best possible 5-card hand.

Here's how the game in your `main` method will work. We first draw five community cards from a deck and then repeatedly:

1. Draw two new 2-card hands from a given deck. These are the hole cards.
2. Print the community cards and the hands.
3. Ask the user who the winner is (or if there's a tie), taking into account the community cards.
4. If the player is correct, they get one point and the game continues.
5. If the player is incorrect, the game ends and the player's total points should be displayed.
6. The game is also over if there are not enough cards left to play another round.

Each hand uses both the hole cards and the community cards to get the best 5-card hand possible. In other words, choose the best 5 card hand from 7 cards. A hand is not required to use its hole cards as part of its hand (i.e. any 5 cards from the available 7, including just the 5 community cards, is a possibility)..

Here is some sample output. Yours doesn't have to look exactly like this.

```
The community cards are:
Queen of spades | 5 of diamonds | 2 of spades | 6 of clubs | 7 of diamonds

Which of the following hands is worth more?
Hand a:
6 of spades, 8 of clubs
or
Hand b:
```

```
4 of clubs,  8 of spades

Enter a or b (or SPACE to indicate they are of equal value)a
got input: a
CORRECT!!!
-----------------------------------------------

The community cards are:
Queen of spades | 5 of diamonds | 2 of spades | 6 of clubs | 7 of diamonds

Which of the following hands is worth more?
Hand a:
King of spades, 2 of hearts
or
Hand b:
6 of hearts, Jack of spades

Enter a or b (or SPACE to indicate they are of equal value)b
got input: b
CORRECT!!!
-----------------------------------------------

The community cards are:
Queen of spades | 5 of diamonds | 2 of spades | 6 of clubs | 7 of diamonds

Which of the following hands is worth more?
Hand a:
9 of clubs, Jack of diamonds
or
Hand b:
9 of hearts, Jack of clubs

Enter a or b (or SPACE to indicate they are of equal value)
got input:
CORRECT!!!
```

Details

Here are the required classes.

- **Card**. You get to choose what the internal representation will be, but you are limited to a maximum of **two** instance variables. Your Card class must have the following:
    - getter methods for getting the rank and suit (your choice for return types)

    - `toString` method to return the "Jack of Clubs" readable version of the card. Whole cards should be represented numerically, like "2 of Hearts", not "Two of Hearts".
    - the following **<u>two constructors</u>**. Read the Javadocs carefully. You may have additional constructors if you want.

```
/**
 * Creates a Card with a given rank and suit.
 *
 * @param rank whole cards (2-10) can either be spelled
 *             out like "two" or numeric like "2". Case
 *             insensitive.
 * @param suit "Spades", "Hearts", "Clubs", or "Diamonds"
 */
public Card(String rank, String suit)

/**
 * Creates a Card with a given rank and suit.
 *
 * @param rank The rank of the card, which must be between
 *             2 and 14, inclusive.
 * @param suit The suit of the card, which must be
 *             0=SPADES, 1=HEARTS, 2=CLUBS, or 3=DIAMONDS
 */
public Card(int rank, int suit)
```

- **Deck.** Implement this as an ArrayList again. It should have the following public methods with the same behavior as in Project 3 (except for `isEmpty`, which is new). The words in `Courier New font` tell you what the required name is.
    - a default constructor
    - `shuffle` method
    - `deal` method
    - `gather` method
    - `size` method
    - `isEmpty` method that returns true or false if the deck is empty or not
    - `toString` method
- **PokerHand**. This should be the same as in Project 3 (if your PokerHand class worked well in Project 3, this should require no changes other than possibly method names). Required public methods:
    - a non-default constructor that takes an ArrayList of Cards as a parameter
    - `addCard` method (adds a given Card to the hand)
    - `getIthCard` method (gets a Card by index in the hand)
    - `toString` method
    - `compareTo` method

Here are the new classes:

- **CommunityCardSet.** An object of this class represents the collection of community cards. It should contain
  - a non-default constructor that takes an ArrayList of Cards as a parameter
  - `addCard` method
  - `getIthCard` method
  - `toString` method

  Note that a CommunityCardSet is not a PokerHand, even though it might have 5 Cards – A CommunityCardSet isn't a flush or a pair and doesn't need to be able to compare to a PokerHand.

- **StudPokerHand.** An object of this class represents a 2-card poker hand that also has access to the community cards. It should contain
  - a non-default constructor with the following prototype:
    `public StudPokerHand(CommunityCardSet cc, ArrayList<Card> cardList)`
  - `addCard` method (adds to just the 2-card hand)
  - `getIthCard` method (from just the 2-card hand)
  - `toString` method (should include the community card set too)
  - `compareTo`, which compares this hand to another StudPokerHand. Here's the required signature:

```
/**
 * Determines how this hand compares to another hand, using the
 * community card set to determine the best 5-card hand it can
 * make. Returns positive, negative, or zero depending on the comparison.
 *
 * @param other The hand to compare this hand to
 * @return a negative number if this is worth LESS than other, zero
 * if they are worth the SAME, and a positive number if this is worth
 * MORE than other
 */
public int compareTo(StudPokerHand other)
```

The algorithm for StudPokerHand's compareTo method is to find all possible 5-card PokerHands one can make out of the 7 cards the StudPokerHand has access to. Let's call the helper method that does this **`getAllFiveCardHands()`**, which returns an **`ArrayList`** of **`PokerHands`**. Once you have the list of all possible PokerHands, do this:

```
private PokerHand getBestFiveCardHand()
{
      ArrayList<PokerHand> hands = getAllFiveCardHands();
      PokerHand bestSoFar = hands.get(0);

      for (int i = 1; i < hands.size(); i++) {
          if (hands.get(i).compareTo(bestSoFar) > 0) {
                bestSoFar = hands.get(i);
          }
      }
      return bestSoFar;
}
```

Study what the above code is doing and it should look familiar. It's just a version of the old "find the biggest number in a list" algorithm that you wrote in intro. Only now, we're using PokerHand's compareTo method to "find the *biggest* PokerHand in an ArrayList". (You're welcome to copy/paste this method into your project.) Once you know this StudPokerHand's best 5-card hand and the `other` StudPokerHand's best 5-card hand, you can compare them to see who wins.

Of course, this requires that you've written the `getAllFiveCardHands` method. I won't give you the code for that one, but we will discuss a couple of possible algorithms in class.

Other Requirements

1. Every public method should be unit-tested now. Make a separate class for each class you are testing (CommunityCardSetTester, StudPokerHandTester, etc.)
2. We're again using Gradescope to test output. The Card constructors (0.4 points), PokerHand's compareTo (7.6 points), and StudPokerHand's compareTo (12 points) will all be tested.
3. All code should be in a package called "**`proj4`**". Gradescope expects this. I'm not giving you any starter code this time. You should just start with a copy of Project 3 and work from there.
4. The `main()` method with the game described on the first page should be in a class by itself (**Main**).

Grading
This project is worth 100 points. Here's the breakdown:

- 20 points for output. This comes directly from your Gradescope score.
- 20 points for thorough unit testing
- 60 code/design quality, including information hiding, good use of OOP, reusability, Javadocs for all public methods, meaningful names for parameters, variables, and methods, good readability using whitespace, a blank line to separate method definitions, having required elements described in this document, etc.

Turning it in
1. Put the honor code affirmation in *main*:
   I affirm that I have carried out the attached academic endeavors with full academic honesty.
2. Upload all of your .java files to gradescope (don't upload a zip file --- that will likely cause problems).

Gentle Reminder
Programming projects are *individual* projects. I encourage you to talk to others about the general nature of the project and ideas about how to pursue it. However, the technical work, the writing, and the inspiration behind these must be substantially your own. You must cite anyone else who contributes in any way to the project by adding appropriate comments to the code. Similarly, if you include information that you have gleaned from other sources, you must cite them as references. Looking at, and/or copying, other people's code is inappropriate, and will be considered an honor code violation.