Go Fish!
EE 422C

100 points

Due: Wednesday 2/14/2018 at 10:00pm

The goal of this assignment is to use some pre-defined classes to create more classes and to use these classes to play a simple card game. We will use the basic rules of [Go Fish](#). The requirement is that your program work with two players (starting with 7 cards).

The basic game

5 cards are dealt to each player, or **seven if there are only two players**, and the remaining card deck is placed between the players.

The player whose turn it is to play asks another player for their cards of a particular rank. For example, "Agatha, do you have any threes?" A player may only ask for a rank of which they already hold at least one card. The recipient of the request must then hand over a card of that rank (note the usual rules would state *all the cards of that rank*, but we are modifying the rules slightly). If the call was successful, the same player has another turn. If the player who was asked has no cards of that rank, they say "Go fish" (or simply "Fish"), and the asking player draws the top card from the pack. The turn then passes to the player who was asked.

When one player has two of the same cards of a given rank, they form a *pair*, and the cards are placed face up on the table (we will call the pair a *book*). The game ends when all twenty-six books are formed, and the player who won the most books wins.

If the player whose turn it is has no cards left in hand, the game is not over. They simply draw the top card from the pack and the turn passes to the next player in the rotation, or, in the case of a two-player game, the other player.

(adapted from Wikipedia)

Here are some classes that we will use for this project. You will need to implement most of these classes before you can get to the "Go Fish" game. We will be using the Arraylist class to hold the deck, the player's cards, and to hold the "books" that are formed from pairing cards. Each player will have a hand and a collection of books.

card.java - simulates one playing card (a suit and a value).

deck.java - simulates a deck of 52 cards

player.java - simulates a player in the game

card_demo.java - a demo of a main program that will deal some cards

---

For the output of the program, you should print each play to a text log file (you may want to echo to standard output while debugging) and the eventual winner.

Note: The computer will play the hands of both players. There is no input from people.

e.g. Joe asks - Do you have a J?
     Jane says - Go Fish
     Joe draws 4d

     Jane asks - Do you have a 4?
     Joe says - Yes. I have a 4.
     Jane books the 4..

     Jane asks - Do you have a 6?
     Joe says - Go Fish
     Jane draws Jc

etc.

Jane wins with 13 booked pairs.

(show the list of pairs)

Joe has 10 booked pairs.

(show the list of pairs)

Your output can certainly be different as long as it describes the game.

Notes:

1) Start by implementing the Card class and testing it, then the Deck class, then the Player, etc.

2) If you are unsure about how the flow of the game should proceed, use your best judgement and document your decision.

3) Start early!

Notes:

- We can do this program as individuals or with a partner (you can choose your partner, but he/she must be in the same discussion section).
- If you run into an ambiguous situation regarding the card game, use your best judgement and document your solution.
- Name the output file "**GoFish_results.txt**".
- **You <u>must name</u> your project "assignment2_EID1_EID2" (only add the EID2 if you have a partner).**
  - **Name the package gofish_assn.**
    - What to submit:
    - The package directory 'gofish_assn' and all Java files it contains
    - When unzipped, the folder structure should be:
    - assignment2_EID1_EID2(folder)
    - ->gofish_assn(folder)
    - --->GoFish.java
    - --->Card.java
    - --->Deck.java
    - --->etc.java
  - **Hand in a zipped file named assignment2_EID1_EID2.zip. Only turn in one copy of the zip file per group.**
- Appropriate values should be stored in constant variables.
- Be sure to follow the documentation standards for the course.
  - All methods should have Javadoc
  - Each variable must be declared on a separate line with a brief comment following the declaration
    - Use meaningful variable names

Submit: Turn in to Canvas

Updated: 1/29/18 rlp