A2.1 Go Fish

Announcements and Clarifications

None

Description

Your first task is to implement a simple two player card game called "Go Fish". You **must** use a linked list data structure of your own implementation to store the cards contained in each player's hand as well as the cards that remain in the deck. The rules of the game are listed below. The text of a sample run of our solution is here. You need not match the output exactly, though your program should produce output that is in the same spirit.

The first player in the game is the human player, they will enter moves their moves from the console. The second player is the computer, each turn the computer must choose a valid move. How the computer chooses its moves is up to you. The computer can choose a random valid move, the computer can choose the lowest or highest rank it has, the computer can even cheat and look at the human's hand and pick moves that way it is entirely up to you.

Your design should be object-oriented using natural objects in the problem space: Deck, Card, LinkedList, ... You should not use any Java collection classes in your implementation (if you are unsure whether you have done this or not please ask).

Goals / Requirements

- Implement a simple linked list data structure
- Use your linked list to manage cards in the player's hands and those remaining in the deck
- Implement the rules of Go Fish
- Review program implementation and design in Java:
    - Loops
    - If-statements
    - Object oriented design
    - Design choices: coupling and cohesion
    - Console input using Scanner and handling bad input
    - Generating random numbers

Rules for Go Fish

1. Shuffle a standard deck of 52 playing cards.
2. Deal 7 cards to each of the two players.
3. The players place all pairs of cards with the same rank on the table.
4. The players alternate turns:
    a. The current player says to the other player give me all cards in your hand of rank X. The current player must hold a card of rank X.
    b. If the other player has cards of rank X those cards are given to the current player.
    c. If the other player does not have cards of that rank that player says "Go Fish". The current player draw a card from the top of the unused cards in the deck.
    d. If the current player has a new pair of cards they place that pair on the table and repeat their turn.

    e. If the current player does not have a pair the other player takes their turn.
5. The game continues until the current player has no cards in their hand at the beginning of their turn.
6. The player with the most pairs placed on the table wins.

The standard deck of cards contains 52 cards. Each card has a suit and a rank. There are four suits: Clubs, Diamonds, Hearts, and Spades; we represent the suits by a letter C, D, H, and S respectively. The thirteen ranks are Ace, Two, Three, Four, Five, Six, Seven, Eight, Nine, Ten, Jack, Queen, King; our program will represent them by the numbers 1, 2, 3, ..., 12, 13 respectively. For example the King of Spades will be displayed as 13S, the Ace of Diamonds as 1D and the Seven of Clubs as 7C.

Commenting and Style

- Your program should be written in a style that makes it easy to read and understand.
- At the beginning of each .java file you should include a description of the class and how it interacts with the other parts of your program.
- You are **not** required to use javadoc style comments.
- If your code is doing something complex or non-standard please comment that portion heavily.

Handin

Please hand all necessary files into your handin directory in a subdirectory named **GoFish**. Your application class should be called **GoFish.java**. There should be exactly one class declared within each .java file. If you program does anything strange (bugs), awesome(extra features) or has a non-intuitive interface please include a file called README.txt which explain them. If there are bugs in your program but you do not describe them in your README you will lose more credit than if you had described them.

Hints

- Develop incrementally:
    - Get your linked list working
    - Use your linked list with the basic classes in the program (Card)
    - Develop the rest of the program
- Test your program using smaller decks of cards so the game plays faster (fewer ranks, fewer suits, smaller starting hands).
- Reduce randomness in your program by initially testing with a fixed random seed.