# COMP 2150 – Spring 2014
# Project 1: Go Fish!
**(Posted Feb. 11 - Due Mar. 7 by 11:59 pm)**

**Note: You have the option of working in teams of two for this assignment. If you choose to do so, please include both your name and your teammate's name on all your source code files. Only one person needs to submit the final product to eCourseware, but please include a note with the submission that lists both of your names. If an honors student and a regular student choose to work together, both agree to be graded on the honors scale.**

Submission: Please zip your source code into a single file (you can zip the entire project folder if you're using BlueJ) and upload it to the proper folder in the eCourseware dropbox at https://elearn.memphis.edu. The dropbox will cut off all submissions after the indicated deadline, so please don't wait until the very last minute to submit your work!

Grading: This assignment is worth a maximum of 80/75 points for regular students (75/75 for honors students). The assignment will be graded by the TA Vincent Nkawu (venkawu@memphis.edu). If you have questions or concerns about your grade, please contact him first. I'll be happy to look over your assignment myself if he's not able to resolve the situation to your satisfaction. Also remember that as stated on the syllabus, all submissions MUST compile and run to receive credit. The TA does not have time to find and correct your syntax errors!

Coding Style: Be sure to follow the good coding practices that were discussed in COMP 1900:

- Use descriptive variable and method names. Avoid using single-character names unless it's very obvious what the variable's being used for (like a loop counter).
- Follow standard Java programming conventions for **variableAndMethodNames**, **ClassNames**, **CONSTANT_NAMES**.
- Consistently indent your code.
- Use comments judiciously. Every source code file should include a comment block at the top that lists your name and the assignment number. Every non-trivial method should have a comment preceding it that specifies what the method does, what its parameters are, and what it returns. For simple methods like accessors or mutators this isn't necessary, but it doesn't hurt to put in a simple comment like `// accessor methods`. I also expect to see comments throughout your code explaining what actions are being taken!

## Background Information

*Go Fish* is a children's card game usually played by 3-6 players (although two can also play), using a shuffled standard deck of 52 cards. The objective of the game is to collect as many "books" (sets of 4 cards of the same rank) as possible.

The game starts by dealing 5 cards to each player. The player to the dealer's left takes the first turn. During Player *A*'s turn, *A* asks a specific player (call this Player *B*) to give *A* all cards of a specific rank. To do this, *A* must possess at least one card of that rank. If *B* has any cards of that rank, s/he must give all of them to *A*, and *A* then takes another turn.

If *B* does not have any cards of that rank, s/he tells *A* to "Go fish!" At this point, *A* draws the top card from the deck. If that drawn card matches the rank that *A* initially requested, *A* takes another turn. If the drawn card does not match the rank that *A* initially requested, it becomes *B*'s turn.

As soon as a player collects 4 cards of the same rank (whether from another player, or from the deck), s/he removes this "book" of cards from his/her hand and sets it aside. The game ends when there are no cards left in the deck, or when any player runs out of cards in his/her hand. The winner is the player with the most books. Ties are possible.

(Note: These rules were obtained from http://www.pagat.com/quartet/gofish.html)

## The Assignment

Write a text-based implementation of Go Fish. Your program should allow 2-6 players to play the game. Since there's no concept of "to the dealer's left" in this software version, you can make the first turn go to any player you wish. When a player takes his/her turn, your program should display a count of how many cards are left in the deck, as well as a summary of that player's current hand and collected books, if any. When the game ends, determine and show the winning player(s) on the screen.

To make things easier for you (and for the TA!), I've already developed a basic class design for you. Your project must include the following classes:

- **Card** and **Deck** classes to represent an individual card and a deck of cards. You can use the Card and Deck classes from the poker example discussed in lecture as a starting point. However, you may need to add a little bit more functionality.
- A **Player** class to represent each player. This class should include (but is not limited to) the following parts:
    - Instance variables
        - Player's name
        - Cards in the player's hand
        - Books that the player has collected
    - Methods
        - Adding a card to the hand
        - Removing cards from the hand that form complete books
        - Transferring all cards of a specific rank to another Player object
- A **GoFish** class to represent the entire set of players. This class should include (but is not limited to) the following parts:
    - Instance variables
        - A Deck object
        - An array of Player objects
    - Methods
        - Allowing a specific player to take his/her turn
        - Checking whether the game is over
        - Determining the winning player(s)
        - A **startGame()** method similar to what we wrote in the poker and Nim examples. This method is what will get executed when you run your game. It should include things like allowing the user to specify the number of players, collecting each player's name, allowing players to take turns until the game is over, etc.

Note that all user input should be located only in the **GoFish** class. The idea is that the core game objects (cards, deck, etc.) should be independent of the game's user interface. If you wanted to make a graphical version of Go Fish, you'd be able to use the existing classes and make changes only to **GoFish**!

Implement error checking on all user inputs, to the extent that was covered in COMP 1900. For example, you should include things like:

- Making sure the user can't select fewer than 2 or greater than 6 players
- Making sure a player can't ask him/herself for cards
- Making sure a player can't request a rank that s/he does not have in his/her hand

## Need Help?

If you choose to work with a teammate for this assignment, the two of you can jointly write your code. I don't mind if you consult with students outside your team, but the code you submit should be written exclusively by you and your teammate (or just you, if you opt to fly solo). If you find yourself stuck, please feel free to contact me (Top) anytime. The Computer Science Learning Center in Dunn Hall 208 is also open, where you can get help from graduate students. Hours are posted at www.cs.memphis.edu/cslc.

The most important thing is to start working on this project early. Despite the apparent simplicity of the game, this is definitely not a "wait until the night before the due date to start" assignment. It will require significantly more code than a regular homework.

If you run into issues, it often helps to take a break from programming to sleep on it and think about possible solutions. You will not have this luxury if you wait too long to start. You have nearly a month to write this – use that time wisely! I highly recommend that you have at least a basic working version of your game completed by Feb. 28. That gives you a full week to test and polish your code.