Floaties
Augie Murphy, Sasha Fomina, Gabi Newman

APCS pd3
01/04/16

Final Project Proposal

GO FISH!!!

INTRODUCTION

This is a simulation of the card game Go Fish! that can be played via terminal prompt. It allows the user to choose single or multiplayer, the number of players, and the number of computer opponents.

*****To Play*****

The object of the game is to acquire the most books.

Setup: *Each player is dealt five cards. If they have four cards of the same rank, they are removed from their hand, and counted as one book.*

Rules: *On their turn, a player can ask any other player if they have a card of a specific rank. If they do, the other player gives them all cards that they have of that rank. Otherwise, they draw one card from the remaining cards in the deck. If there are no cards left to draw and a player has no cards, they are skipped.*

To Win: *The game ends when there are no cards left in play. The player with the most books wins.*

****

Our Implementation

Each **Player** has a **Hand** of **Card**s. Each player is ***dealt*** five cards from the **Deck** and the remaining cards become the "pool". This is where the players will fish from.

The first user will be prompted for their ***_name*** and assigned the first ***_turn***. Their ***_hand*** will be displayed in the terminal. Then, they will be prompted what card they'd like to ***ask for*** and who they want to ask. The askee will ***check*** their cards and if they have it, ***remove*** all cards of that rank from their ***_hand*** and the asker ***adds*** those cards to their ***_hand*** . Otherwise, the asker must ***fish***.

...this gives us candidates for classes and methods!!!

1) A OrderedCards class, which is a wrapper class of ArrayList with an addBinary() that takes a Card as a parameter
2) A **Hand** class with an OrderedCards<Cards> known as the _hand that can ***check*** (using binary search), ***add***, and ***remove***.
3) A **Card** class with a _rank and _suit , both being represented by integers
4) A **Player** a _name, _hand, _turn number, and keeps track of their _numBooks. ***Each player can*** declare their name, and implement the interface deck. They should sort their hands as they receive cards. We'll need an accessor for the name.
    a) **Human** extends **Player**
    b) **AI** extends **Player**
5) A **Deck** class that holds an ArrayList of 52 **Cards** and can ***shuffle*** and ***deal***.

PLAYING THE GAME

1) To start the game: shuffle and deal the deck; show each player(not computer), make any possible books and refill their hand if necessary
2) In each player's turn: they get to see their hand and ask any player for a card in their hand. If the player has the card, repeat. If they don't, go Fish! and end turn.
3) To move from player to player: players have a turn number. The game class keeps track of the total number of turns. (Total number of turns mod current players number + 1) = x; if get(player's turnNum == x) theirTurn; there are no players left, end the game. If a player has no cards, go to the next player.

The player with the most books wins.