THIS ASSIGNMENT IS TIME-CONSUMING AND DIFFICULT. YOU SHOULD START THE DAY IT IS ASSIGNED SO YOU HAVE TIME TO DEBUG

Reading

This assignment assumes you have completed the reading for weeks 1-9 on zyBooks.

Why?

You will learn how our OO framework for playing Cards can be adapted into a variety of different card games by re-using the same classes in different ways. You will also get significantly more practice with Java Collections and a deeper understanding of Java's reference types.

The Assignment

Now that you've built an awesome OO framework for playing cards, let's use what you've done to program an actual card game: ERS!

You will not be getting user input to play the game, but instead simulating the game play using probability and printing out a record of the game play along with the winner of the game. ERS is played according to the following rules (you can also watch this game play video to get a better understanding):

https://www.youtube.com/watch?v=VGp1CDK71PM

Game Set Up

ERS is played with 2 or more Players arranged in a circle. The deck is divided evenly between all players to start the game.

Game Play

The game starts by dealing the deck evenly amongst all players. Player 1 then places a card from the top of their hand into the center pile. There is no decision making involved in this, they simply put down whatever card happens to be there without looking at it. The players then sequentially add cards from the top of their hands on top of the center pile until one of two things happen:

A Face Card is Played

When a face card is played, the next player will have a certain number of cards to put down to try and play a face card. If the player plays a face card, it moves on to the next player's turn. If their number of chances run out without a face card being played, the player who played the face card takes the center pile and adds it to the bottom of their hand. The number of chances depends on the face card that was played:

Jack: 1 chance

Queen: 2 chances

King: 3 chances

Ace: 4 chances

A Special Pattern is Played

If a special pattern is played, players can "slap" the center pile. Whichever player slaps the pile first gets to take the entire pile and add it to the bottom of their hand. There are many different patterns that are played with ERS, but we will play with the following:

Double: two cards of the same value are played in a row

Sandwich: two cards of the same value are separated by one card of a different value

Top Bottom: the card that was just played (top of the pile) matches the first card played that round (bottom of the pile)

Once a player takes the center pile, whether through slapping or through winning a face card round, that player then starts the next pile.

Ending the Game

The game ends when one player has accumulated all of the cards in the deck. This player is declared the winner.

Simplifications

To make our lives easier for this project, we will be simplifying the game in a few ways. For this implementation, you can make the following assumptions:

Players cannot "slap back in" once they have run out of cards. If they no longer have any cards to play, they are out and removed from the game.

optional: Each Player will only evaluate the pile for a single pattern (ie. "doubles" or " top bottom". This will be chosen at random when the Player is initialized).

If multiple Players recognize the same pattern and slap the pile, one Player should be chosen at random to be the "first" Player to slap.

Players cannot "slap" on a pattern during a face card face-off

Players always recognize their pattern and slap for it, and they never accidentally slap on something that is not a valid pattern.

Designing a Solution

As with MP3A, the best way to build your card game is to rely heavily on object orientation. You will want to create the following classes:

Player

A player consists of the following member variables:

playerNum, an integer (1, 2, etc) to identify which player it is

hand, a LinkedList of cards in their hand. The first element in the list is considered to be on "top" and new cards should be added to the "bottom" (the end of the list)

pattern, a String representing the type of pattern this player watches for, chosen at random from the Game class's patterns array

Additional member variables as you deem appropriate

Players consist of the following methods:

An overloaded constructor that takes a player number, a LinkedList of Cards representing their entire hand, and the pattern they should recognize

public Card playCard() removes a Card from the top of the player's hand (position 0) and returns it

public boolean slaps(LinkedList<Card> pile)- evaluates the Player's pattern member variable and calls the appropriate Game static method to check for that pattern. If the pattern is in play, the method should return true to slap. If not, the method should return false.

Accessor methods for all member variables: getPlayerNum(), getHand(), getPattern() etc

toString that returns the Player's number, pattern, and current hand of cards

Other private helper methods as required to implement game play rules

... [Content truncated for brevity]