Janet Zhang (BGC)

19 April 2023

1 Game of "Snap!" Constants

1.0.1 FACEVALUE: Each suit has 13 cards, with values ranging from two to ten, followed by face cards which are the Jack (J), Queen (Q), and King (K), and an Ace (A)

1.0.2 SUITS: four suits, which are clubs, diamonds , hearts and spades

1.0.3 CONDITIONS: suit, value or both

2 Game of "Snap!" Code Functions

2.0.1 shuffledCards: This function creates and shuffles a deck of cards based on the number of packs specified by the numPacks argument

2.0.2 snapGamePlay: This function plays a game of Snap between two players using the specified matching condition, which is passed as the matchCondition argument. The game is played by iterating over the shuffled deck of cards and adding each card to a common pile. If the length of the common pile is less than 2, the game continues. If the matching condition is 'suit', 'value', or 'both', the program checks whether the last two cards in the common pile match the specified condition. If they do, a snap winner is chosen randomly, and that player takes ownership of all the cards in the common pile. Otherwise, the game continues

2.0.3 declareWinner: This function tally up the total number of cards each player has accumulated and declare the winner/draw.

3 Game of "Snap!" Run Code

3.0.1 Script is ran with 2 inputs, the packChoice and conditionChoice under 'main'

4 Game of "Snap!" Improvements

4.0.1 Code could be optimized for performance, particularly when dealing with large numbers of packs, by using more efficient data structures and algorithms. List is performance slow, as alot of looping is required

4.0.2 Code could be made more interesting by adding in additional gameplay elements, such as allowing players to declare "Snap!" at any time and forcing the other player to match their card, or allowing players to play multiple cards at once and creating longer runs.