Documentation - Full Time Project - Julia Ulsh

Description:

For this project, I followed specification as set out for the Full Time Project Technical interview.

I wrote my solution in Java, using the Eclipse IDE (Version: 2021-09 (4.21.0)). I created a project called "cardGame" which contains classes "CardGame", "Player", and "Card". The CardGame class holds the main method which runs the game, and various classes to support the logic of the game. The Card class is a class that defines a new object of type Card which has attributes of a face value, a suit, a suit rank, and points. The Player class also defines a new object of type Player, which has attributes of a name, a hand (of cards), and a score. I have also included JUnit 5 unit tests for my project.

CardGame Classes:

createDeck()

* returns array of Card objects, the *deck*
* creates Cards with faces and suits as defined in project specification

shuffleDeck()

* returns array of Card objects, the *deck*
* shuffles the deck using the Set structure to prevent duplicate cards in the shuffled deck

scoring()

* returns array of Player objects, the *players*
* manages the scoring after each round:
  + Player with highest face value card - increases score by 2 points
    - if more than one player has the same highest face value, points are awarded by suit ranking in the order: Spade > Heart > Diamond > Club
  + If a player receives a penalty card:
    - their score is reduced by one, only if the resulting score will be 0 or greater
  + All other players - score does not change

findMax()

* returns a two integer array, the maximum value of the array passed to the method and the location of that value within the original array
* helps with finding the highest scoring card among the players

scoreboard()

* returns a String that displays the current scoring of the game

getPlayerNames()

* returns a String array that contains the unique identifiers for players in the game

playGame()

* manages the logic of the game
* Prompts user for number of players in game
* Players, scores, and deck are initialized
* Prompts players to take their turns
* Game is played until winning player reaches the winning score and maintains at least a 2 point lead over the next closest player

Player Classes:

setName()

* assigns the String passed in to be the name of the Player getName()
* returns the String *name* of the Player setScore()
* assigns the integer passed in to be the score of the Player getScore()
* returns the integer score of the Player setHand()
* adds the Card passed in to the player's List of Cards at index 0 getHand()
* returns the List of Cards that belong to the player toString()
* returns the String of the player's name along with the card that they drew

Card Classes:

setSuit()

* assigns the String passed in to be the Suit of the card getName()
* returns the String Suit of the card setFaceValue()
* assigns the String passed in to be the Face Value of the card getFaceValue()
* returns the String Face Value of the Player setPoints()
* sets the integer points of the Card to be the integer equivalent of the Face Value getPoints()
* returns integer points of the Card getSuitRank()
* returns the integer ranking of the different suits, for comparing cards with the same face value

getCardName()

* concatenates the Face Value and Suit Strings of the card

Running Instructions

To Run the Game: Open the CardGame.java file in a java IDE such as Eclipse

* The Scanner accepts the number of players
  + the program will wait for an integer between 2 and 4 to be entered
  + if a non-integer character is entered, the program will halt and prompt the user to re-run the program, entering a valid number of players
* The Scanner will wait for a character to be entered and then the enter key to be pressed in between players' turns
* The output will be visible in the Console

To Run the Tests:

Add Junit 5 to build path in IDE like Eclipse

\*The testPlayGame() test will require a run of the game until completion to run.

Sample Run:

Enter number of players as an integer between 2 and 4: 3

Beginning Game...

player1 Press any key and then enter to take turn:a

player1 drew card: Nine of Hearts

player2 Press any key and then enter to take turn:d

player2 drew card: Nine of Diamonds

player3 Press any key and then enter to take turn:m player3 drew card: Two of Clubs

Score Board after round 1 player1: 2

player2: 0

player3: 0

player1 Press any key and then enter to take turn:a

player1 drew card: Two of Spades

player2 Press any key and then enter to take turn:d

player2 drew card: Ten of Diamonds

player3 Press any key and then enter to take turn:m player3 drew card: Jack of Clubs

Score Board after round 2 player1: 2

player2: 0

player3: 2

...

player1 Press any key and then enter to take turn:a player1 drew card: Nine of Diamonds

player2 Press any key and then enter to take turn:d

player2 drew card: Four of Diamonds

player3 Press any key and then enter to take turn:m player3 drew card: Queen of Hearts

Score Board after round 27 player1: 13

player2: 14

player3: 21

Player player3 wins!

Winning Score: 21

Notes

For this project, I actually really enjoyed working through the process of creating the game! The only places I really got hung up were the shuffling of the deck and handling when two players drew cards with the same face.

For shuffling, I knew the basics of how to rearrange the cards with the Random class, but I was of course getting repeats during this process. After a little research, I found that using the Set structure would help me keep track of which indexes I had already used since Sets do not allow repeats. I used this source to help me understand sets:

* [https://javarevisited.blogspot.com/2015/06/3-ways-to-find-duplicate-elements-in-array- java.html](https://javarevisited.blogspot.com/2015/06/3-ways-to-find-duplicate-elements-in-array-java.html#axzz77oSEr800)

For the ranking based on Suits, I spent a good bit of time trying to figure out the best way to approach this and finally the solution occurred to me after taking some time away from thinking about it by crocheting! I have found in my college career that sometimes focusing at something completely different from the code allows me to think of things in a different way and find a solution I had not considered before.

I also referenced the Java 8 Documentation from Oracle throughout the process of coding to refresh myself on Java Syntax

Overall, I am proud of my solution and hope you enjoy playing the game! Thank you for your time and consideration!