Assignment 3

5 Card-Style Poker

***Due:*** *Tuesday, December 10th*

**Objectives:** Reference Types, Arrays, Loops

This assignment is to be done individually.

For this assignment we will be creating a simple poker style game. Players are dealt a hand of 5 cards, and get points based on how good their hand is. We will need to build up a deck of cards, deal those cards out, and then assess how good a hand is. Hands are combinations of cards that are ranked based on rarity – drawing 4 of the same card is less common than having 2 of the same card and is thus worth more points. A full list of poker hands can be found here: <https://en.wikipedia.org/wiki/List_of_poker_hands.> We will be using a simplified deck of cards, which only has values 1-10 (no Jacks, Queens, or Kings), and has no suits (no distinction between Clubs, Diamonds, Hearts, and Spades).  
  
**General considerations:**

Your code must follow the structure detailed below. You will be evaluated on how well you follow best practices when programming. Your code must be properly indented, contain good, descriptive variable names, have helpful inline comments, and ***use constants where appropriate***. You should use only classes and methods we’ve discussed in class – some parts of the assignment may be trivial when using other classes and methods. If you’re unsure whether something is ok to use, simply ask.

It is important to test your methods, you will be required to create and submit at least 3 test cases for **all methods** that do not use random numbers or user input. There are many methods here – it will be tedious to write all your test cases at the end. Instead, I highly recommend testing your code as you go and documenting your test cases as you run your tests. You should have 3 test cases for each method that does not use random numbers or user input.

Your program should be created in a class called Poker, with the following methods:

**Week 1 - Methods for preparing the game:**The following methods have to do with creating and dealing out cards. These methods will be essential to setting up the game.  
  
**int[] buildDeck():**

You should write a method buildDeck that takes no parameters and returns an array of integers representing the deck of 40 cards. Cards in our simplified game have no suits, and only values from 1-10. Your array should contain 4 copies of each number from 1 to 10.  
  
For example: {1,1,1,1,2,2,2,2,3,3,3,3,4,4, …. ,8,8,9,9,9,9,10,10,10,10}

**void shuffleDeck(int[] cardArray):**

You should write a method shuffleDeck that takes and int[] representing a list of cards as a parameter and returns nothing. This method should shuffle all elements of the array. If this problem is being solved using AI tools, the random numbers should be generated using ThreadLocalRandom   
You should do this by going through each element of the array and randomly generating a position in the array to swap with using the random number generator. The end result should be that the array passed as a parameter should have had all its values shuffled around randomly.

**int dealCard (int[] cardArray):**

You should write a method dealCard that takes an int[] representing a list of cards as a parameter and returns an int representing the value of a card. This method should go through the array until it finds the first element that isn’t 0. It should then set that element to 0 in the original array and return the integer value that was there.

**int[] dealHand (int[] cardArray):**

You should write a method dealCard that takes an int[] representing a list of cards as a parameter and returns an int[] representing a 5 card hand. This method should use the dealCard method to deal 5 cards into a new int[], and then return that int[].

**void redraw(int[] hand, int[] deck, int position)**

This method should take an array of ints representing a hand, an array of ints representing a deck, and an int position. The method should replace the value at the given position in the hand with a new value obtained via the dealCard method.

**TEST YOUR CODE! Before moving on, test these methods to ensure they work as expected.**  
  
  
**Week 2 - Methods for assessing hand:**  
The following methods have to do with evaluating the values in an array of cards representing a 5 card hand. These methods will be essential to determining points.

**int countOccurrences(int[] cardArray, int value)**

This classic method makes a return, here we are going to write a method called countOccurrences that will search the int[] sent in as a parameter for the int value, and return the number of times the value occurs in the array. (This method will be useful for several of the methods that follow)

**boolean isFourOfAKind(int[] cardArray)**

This method should take an array of cards, and returns a boolean representing whether the hand contains a four of a kind (four elements of the same value).

**boolean isThreeOfAKind(int[] cardArray)**

**boolean isPair(int[] cardArray)**

This method should take an array of cards, and returns a boolean representing whether the hand contains a pair (two elements of the same value).

**boolean isFullHouse(int[] cardArray)**

This method should take an array of cards, and returns a boolean representing whether the hand contains a full house (two elements with one value, three elements with another value).

**boolean isTwoPair(int[] cardArray)**

This method should take an array of cards, and returns a boolean representing whether the hand contains two pair (two elements with one value, two elements with another value).

**boolean isStraight (int[] cardArray)\*\*\***

This method should take an array of cards and returns a boolean representing whether the hand contains a straight (all elements have consecutive rank, ie: {6,3,7,4,5}).  
**\*\*\***This method is meant to be challenging - it is recommended that you do this only after completing everything else. For a simpler version that is worth most of the marks, you can assume that the array is in sorted order, from lowest to highest

**TEST YOUR CODE! Before moving on, test these methods to ensure they work as expected.**  
  
**Finishing Touches: Methods for running the game**  
The following methods have to do with evaluating the values in an array of cards representing a 5-card hand. These methods will be essential to determining points.

**int assessHand(int[] cardArray)**

This method should take an array of cards representing a hand, and should return the number of points that hand is worth. The points should be calculated using the hand checking methods written above. Points are awarded based on the highest possible scoring hand that can be found for the given values. (Ie: {3,3,3,1,1} returns 700)

|  |  |  |
| --- | --- | --- |
| **Hand** | **Example** | **Points** |
| Four of a Kind | {4,4,3,4,4} | 4500 |
| Full House | {7,7,7,2,2} | 700 |
| Straight | {2,3,4,5,6} | 300 |
| Three of a Kind | {2,5,7,5,5} | 50 |
| Two Pair | {6,6,8,8,3} | 20 |
| Pair | {1,4,6,3,1} | 17 |
| High Card (Nothing) | {3,6,8,1,2} | 0 |

**int runRound()**

The method runRound plays a single hand of poker, and returns the int representing the number of points earned. This method should do the following using the methods you’ve written:

* Create and shuffle a new deck of cards
* Deal a hand of cards 5 into an array, and print it out to the player
* Give the player 3 opportunities to redraw. For each redraw, ask them if they would like to discard a card by specifying a position. If they type -1, do not redraw any cards, otherwise replace the card at that position.
* Calculate the point value of the hand
* Return the number of points

**void runGame()**

The method runGame plays 10 rounds of poker, adding up the number of points accumulated by the player for each hand. Once all 10 hands have been played, print out the player’s final score.

**void main()**

Should be one line of code that calls runGame. That’s it!

**To submit:**

* Poker.java – Source Code
* Tests.txt - 3 test cases for each of: dealCard, dealHand, redraw, checkOccurrences, isFourOfAKind,isThreeOfAKind,isPair,isFullHouse,isTwoPair,isStraight, and assessHand
  + (This is 33 test cases total – it's going to be incredibly tedious to do this all at once after you’ve finished the assignment, so I very strongly recommend documenting these as you go!)

**What next:**  
This assignment is the culmination of all the skills we’ve learned in Programming 1. Great job on making it this far! If you’d like to practice a bit more, you should feel free to add additional features onto this game. The following are some ideas to get you started. **Make sure you submit your working code first before trying any of these ideas, they are not worth extra marks:**

* Add Aces, Jacks, Queens and Kings (AJQK) to the deck. This may require using a different datatype for values than simply ints!
* Add suits to the game (Hearts, Diamonds, Spades, Clubs). For instance, you would have 4 different versions of each card: 5H, 5D, 5S, 5C.
  + Add new hand types, Flushes and Straight Flushes to the game. A Flush occurs when all cards have the same suit (for example: {5D,6D,2D,9D,8D}). A Straight Flush occurs when you have both a straight AND a flush.
* Create options for adding a second player. This could take many forms:
  + Players could compete to see who gets the higher score
  + Alternatively, players could bid on each hand as in regular poker, with the player with the higher hand winning all the points bid
* Use your imagination to come up with other ideas!