Veronica Pratt

Mr. Ettlin

AP CS A, Period 3

February 27, 2019

Elevens Activity Questions

Activity 2 Questions:

1. A deck class, and therefore deck object, contains an ArrayList of cards.
2. The deck contains three cards.
3. You create ranks, suits, and pointValues arrays of length 52. The String ranks array has 4 of each rank, ranging from 1 to and Ace. The String suits contains 13 hearts, 13 spades, 13 diamonds, and 13 clubs. The int pointValues array contains has four of each value from 1 through 9, sixteen 10's, and four 11's.
4. The order does matter, because each card takes its characteristics from a single specific index in each array.

Activity 3 Questions:

1. public static String flip(){

   int randomNumber = (int)(Math.random()*3+1);

   if(randomNumber<=2){

   return “heads”; }

   else{

   return “tails”; }}

2.
```
public static boolean arePermutations(int[] array1, int[] array2){
    for(int i = 0; i<array1.length; i++){
        boolean test = false;
        for(int j=0; j<array2.length; j++){
            if(array1[i] == array2[k]){
                test = true;}}
        if(test){return true; }
        return false;}}
```

3. 1, 2, 2, 2

Activity 6 Questions

1. 5♠ 6♣, 5♣ 6♣
2. Yes because every time two cards are removed unless if it is a J, Q, and K. Ergo, the amount of cards remaining on the board without there being a full J, Q, and K set must always be an even number, otherwise there is a full J, Q, and K.
3. This game does not involve any strategy. It is pure luck based on which cards are drawn every time one replaces a pair.

Activity 7 Question

1. I would only need a deck of cards to play this at my desk. In the elevensBoard class, there is the board size variable, ranks, suits, and point values variables for the cards, a cards variable for cards on the board, a deck variable for the place where card are being pulled, and a debugging variable.

2. Create the game, lay out 9 cards from the deck, look for a pair that adds up to eleven, take them out and replace them, if that pair does not exist look for a set of jack, queen, and king, take them out and replace them, check to make sure at least 9 cards are left in the deck. Play until there is nothing to take out or there are less than 9 cards left.
3. Yes it does, but it doesn't have all the methods.
4. a) it is called in the constructor

   b) public boolean anotherPlayIsPossible(), and public boolean isLegal(List<Integer selectedCards)

   c) 0, 1, 3, 6, 7

   d) for (int i=0; i< cIndexes; i++){
   System.out.println(board.cards[i].toString());}

   e) anotherPlayIsPossible(), because if another play is not possible, then calling the other two methods would not even make sense.

Activity 8 Questions

1. Thirteens uses a ten-card board instead of a 9-card board, and takes out pairs of cards adding up to 13 and kings singly. Jacks and Queens are also worth 11 and 12 instead of nothing. Tens uses a 13-card board, and takes out pairs of cards adding up to 10. Tens,

jacks, queens, and kings are taken out as quartets of any combination. The games underlying similarity of all 3 games lies in the taking out of pairs of cards adding up to certain numbers. All are also mostly games of chance, with skill doing little to help one's chances of winning.

2. The instance variable is initialized in the Board class. Inside constructor of ElevensBoard, the values are passed into the constructor of the superclass.
3. They cover all the differences because all of the methods that are exactly shareable between the card games are implemented in the Board class while the overlapping functions that require different implementations (anotherPlayIsPossible and isLegal) are abstract, and thus implemented in the respective board game subclasses. Ergo, these methods can be easily adjusted to cover differences in Elevens, Thirteens, and Tens.

Activity 9 Questions

1. Size is not a method, it is an instance variable. It is defined within each subclass, so there is no need to make it an abstract method.
2. Removing and replacing cards is uniform regardless of which game is being played. Thus methods dealing with that can be implemented in the Board class and do not need to be made abstract.
3. isLegal() and anotherPlayIsPossible() would still be called polymorphically. This alternate design can still work but all of the methods would have to be implemented separately for each card game board class.