William Nguyen

4/25/16

Mrs. Ozbun 1st Per. APCS

ElevensLabQ

Activity 2:

1. Relationship of Deck and Card?:

The Deck class uses an array of different values equivalent to suits, ranks, and pointValues and creates an ArrayList of cards out of them.

1. How many cards does the initialized deck in the packet contain?:

6 cards. # of ranks, suits, and pointValues matches the example, containing 6 total cards

1. Game of Twenty One explained in packet. Specify contents of ranks, suits, and pointValues

String[] ranks = new String[] {"jack","queen","king" “A” “1” “2” “3” “4” “5” “6” “7” “8” “9” “10”};

String[] suits = new String[] {"spades”, “clovers”, “hearts”, “diamonds”};

int[] pointValues = new int[] {10, 10, 10, 10, 11, 1, 2, 3, 4, 5, 6, 7, 8, 9,10};

1. Does the order of the ranks, suits, and pointValues matter?

As long as the ranks, and pointValues match position in the array list, it should functionally work fine. Suits don’t matter as they work from a different for loop, but the point values and rank need to match in order. But other from that matching, ranks, suits, and pointValues can go in any order. And be rearranged/shuffled.

Activity 3:

1. Write a static named flip. Simulates heads or tails. Twice as likely heads over tails.

public static string flip(){

    int x = (int)(Math.random() \*2 +1);

    if(x >=2){

    return “heads”;

}

else{

    return “tails”;

}

}

      2. Static method arePermutations, given 2 int arrays of same length and no duplicate elements. Returns true if arrays have same elements but in same/different order.

public static boolean arePermutations(int[] first, int[] second){

int count = 0;

for(int i = 0; i < first.length; i++){

    for(int j = 0; j < first.length; j++){

        if(first[i] == second[j]){

    count++;

}

    }

}

if(count == first.length){

    return true;

}

else{

return false;

}

}

   3. Suppose that the initial contents of the values array in the Shuffler.java are {1,2,3,4}. For what sequence of random integers would the efficient selection shuffle change values to contain {4,3,2,1}?

Activity 6:

1. List all possible plays for the board 5,4,2,6,J,K,5,2

5&6,

     2. If the deck is empty and the board has 3 cards left, must they be J,Q,K:

Yes, just because the only time it would offset to 3 is from a previous JQK, so the only way it would arrive there is when there is still 3 left. If it has made it to the end of the deck, it becomes increasingly likely that the cards will successfully fit the Elevens parameter.

     3. It is very unlikely that strategy is involved, as the cards it provide for you afterwards are randomized, so whatever order you are choosing the pairings may lead to success or defeat based on what it randomly chooses next to fill in, as there may be no pairs or another pair. The pairings are specifically meant to arrive at 11 so there are specific number matchings and thus don’t overlap, so there is no issue when you choose since it’s a random card.

Activity 7:

1. What items would be necessary if you were playing a game of Elevens at your desk(not on the computer)? List the private instance variables needed for the ElevensBoard class.

Deck of cards, 9 positions for cards. Pile of pairs.

1. Write an algorithm that describes the actions necessary to play the Elevens game.

Deal initial 9 from deck to positions on field of play.

Choose pair, if they are equal then remove from board, otherwise do nothing

Replace combos with successful chosen pair and replaces 2 cards that were just used.

1. Examine the partially implemented Elevensboard. Does it contain everything necessary to play the game?

No. It doesn’t run and a lot of methods aren’t implemented yet.

     4.

1. Where is the dealMyCards method called in ElevensBoard?:

The ElevensBoard Constructor, newGame.

     b.  Which public methods should call the containsPairSum11 and containsJQK methods?

isLegal, anotherPlayIsPossible,

     c. Activity c in blue packet. You basically remove all the nulls because it only adds everything but nulls to a new list and returns that list. There are also no extra spaces because it is a brand new constructed arrayList

     d.Complete the following printCards method to print all of the elements of  method to print all of the elements of cards that are indexed by cIndexes.

public static printCards(ElevensBoard board) {

List<Integer> cIndexes = board.cardIndexes();

for(int i = 0; i< cIndexes.size(); i++){

if(cIndexes[i] != null){

System.out.println(cards[cIndexes[i]]);

}

}

}

e. which one of the methods that you identified in the question 4b above needs to call the cardIndexes method before calling the containsPairSum11 and contains JQK methods? Why?

anotherPlayIsPossible does because it’s checking the entire board to find another play.

isLegal doesn’t need it because it’s specifically checking two cards.

Activity 8:

1. Discuss the similarities and differences between Elevens, Thirteens, and Tens.

All of them are boards, They use different card point values and have a different number of cards on the board. They also have different ways to remove cards out of the game. They still functionally play the same though, with choosing certain cards and then checking if they are removed corresponding to the rules and then doing so. Similar methods and variables.

1. As discussed previously, all of the instance variables declared in the Board class. But it is the ElevensBoard Class that “knows” the board size, and the ranks, suits, and point values of the cards in the deck. How do the Board instance variables get initialized with the ElevensBoard values? What is the exact mechanism.

Since the Board class is the superclass of ElevensBoard, we can super the variables back up to the Board constructor. ElevensBoard can use all of the values and methods of its super class so while it effectively has these values in the ElevensBoard object, it is still applying the Board fields and methods.

1. Now examine the files Board.java, and ElevensBoard.java found in the Activity 8 Starter Code directory. Identify the abstract methods in Board.java. See how these methods are implemented in ElevensBoard. Do they cover all the differences between Elevens, Thirteens, and Tens as discussed in question 1?

It accommodates what plays are possible and what the parameters for a successful play is, which is the major changes in Elevens. It seems to also cover the change in the size of the board with the constructor and the size statement.

Activity 9:

1. Size of the board is one of the differences between Elevens and Thirteens. Why is size not an abstract method?

Returns card.length so it already varies based on cards, which cards is based on the constant in each separate board and thus already varies based on board.

1. Why are there no abstract methods dealing with the selection of the cards to be removed or replaced in the array cards?

There is no need because the games work functionally the same. Cards are removed in the same simple way and can already be polymorphed for a similar result.

1. Interfaces, instead of an abstract Board Class, created a board interface. Would it be able to call isLegal and anotherPlayisPossible polymorphically? Could it work overall?

This new scheme would be able to polymorphically add the methods, because interfaces only implement the header and then provide the class it implements with the opportunity to implement the method how they want it to. The issue with actually implementing it is that Board has more than some empty methods, having some ones that are being used and some variables that are being used. It then becomes an issue of also including what else needs to be implemented from Board into ElevensBoard.