Eleven’s Lab Answers

Activity 2:

1. A deck contains cards. It can exist without them, but it would be severely impacted if any major changes were made to the card class, so the deck class depends on the card class
2. This deck would contain 6 cards
3. ranks = {“2”,”3”,”4”,”5”,”6”,”7”,”8”,”9”,”10”,”Jacks”,”Queens”,”Kings”,”Aces”}

suits = {“Hearts”,”Spades”,”Aces”,”Clubs”}

pointValue = {2,3,4,5,6,7,8,9,10,10,10,10,11}

1. Yes, each element in the ranks array must correspond with the correct point value in the pointValue array. But for the suits array, order does not matter

Activity 3:

1. public static String flip()

{

Random generator = new Random();

int value = generator.nextInt(3);

if (value == 0){

return "tails";

} else {

return "heads";

}

}

2. public static boolean arePermutations(int[] array1, int[] array2)

{

ArrayList<Integer> godArray = new ArrayList(1);

while (godArray.get(godArray.size()) == null)

{

for (int j =0; j<=(array1.length-1); j++)

{

for (int k=0; k<=(array1.length-1); k++)

{

if (array1[j] == array2[k])

{

godArray.add(array1[j]);

}

}

}

}

if (godArray.size() == array1.length)

{

return true;

} else {

return false;

}

}

3. First 0, and then 1

Activity 6:

1. Matching either of the fives with the six
2. Yes because for there to only be three cards left on the board without you losing, they would have to be a Jack, Queen, and King because there are no other triplets that give you an eleven
3. The only strategy there that exists is to always match the triplet of the Jack, Queen, and King first whenever you can, because you will then draw three new cards instead of two, giving you a greater chance of getting more cards for a possible eleven

Activity 7:

1. There would need to be to static variables, the amount of games played, and the amount of games won. The deck would also be an instance variable.
2. Create deck

Deal 7 times

If two cards are selected and the point values equal 11 or three cards are selected and are Jack, Queen, and King then remove them from the board and deal 2 or three more respectively

If no pairs can be made, display “You Lose”

If all cards have been dealt, once the final pair/triple is made, display “You Won”

1. No, because half the code still hasn’t been implemented
   1. When the board is initialized, and when a new game is started
   2. In the isLegal method and anotherPlayIsPossible method
   3. {0,1,3,6,7}
   4. For (int x: cIndexs)

{

System.out.println(x);

}

* 1. In the anotherPlayIsPossible method because you need to know all of the board spaces that still have cards before verifying whether or not another move exists

Activity 8:

1. All three of the games need methods to determine whether a move is allowed, and whether there are moves available, but other than that, each game’s methods are too specific to share a commonality. They have different board sizes, and have different plays that are considered “legal”.
2. Because ElevensBoard inherits from the Board class, all of the specifics for the ElevensBoard class are passed into the constructor for the Board class, which then sets the instance variables to their specific values
3. Though the two abstract methods, isLegal and anotherPlayIsPossible have not been implemented in the ElevensBoard class for activity 8, the comments on what the methods should do indicates that they have been implemented specifically for elevens and not for the other games because it has been implemented with the specific pairs/triples allowed only in elevens.

Activity 9:

1. Because size in an abstract private instance variable, so it should be defined within the class that it is being used in
2. Because each game has a different way in which these are determined, so you making them all abstract methods would leave multiple methods for each class that apply to a different game, which wouldn’t be very efficient
3. This would cause the methods to be called polymorphically, however this would be less efficient as all of the shared methods as well as all of the different methods would have to be re-implemented