**Activity 2:**

1. The relationship would be aggregation because the cards can be in a deck or somewhere else.
2. 2 because you would have a null value for the suit of the 3rd card.
3. Ranks would have 4 of each rank and you would have 1 suit for each of the suits. Then the values for all the aces would be 11, king, queen, jack, and 10 would all be value of 10 then from there we go down until we get to two. You would just set them up so that each rank, suit, and value is matched up for every card.
4. Yes because if they are out of order, the card will be assigned the wrong suit/value.

**Activity 3:**

1. public static String flip()

{

String returnThis = “ “;

Int num = (int)\*(Math.random()\*100);

If(num>33)

{

returnThis = “heads”;

return returnThis;

}

else

{

returnThis = “tails”;

return returnthis;

}

}

1. public static Boolean arePermutations(int[] array1, int[] array2)

{

For(int I = 0;i<array1.length;i++)

{

If(array1[i] != array2[i])

{

Return false;

}

}

Return true;

1. 3,2,1,0

**Activity 5:**

Buggy1:

Constructor or Method: isEmpty

Describe possible code Error: It could just be with their == or != and how they returned it because the way I checked for if it was empty was if the size == 0 then the deck was empty and I returned that. So, if they used anything other than == then they probably just returned false when they meant to return true.

Buggy2:

Constructor or Method: testOneCard

Describe possible code error: Maybe they are forgetting to increment size because when they return it when there is only 1 card in the deck, size is still 0.

Buggy3:

Constructor or method: shuffle

Describe possible code error: I’m guessing that they never reassigned the values of the new deck back to the old deck. Because the new deck is still unchanged because it is the same as the old deck.

Buggy4:

Constructor or method: testOneCard

Describe possible code error: If there is a card in the one card deck then the problem is probably the indexing. But, if there is no card in the deck then that is obviously the problem.

**Activity 6:**

1. The only play is 5 and 6. Everything else added does not equal 11.
2. Yes they do because the only way to get rid of 3 cards in one turn is for them to be king jack and queen. Also, there is an even number of cards(52) in a deck of cards and if you have no more cards in the deck and all the cards are on the field then they have to be king queen and jack so that you can get rid of them
3. Absolutely not. It doesn’t matter what time you pair certain things because you will end up getting the new cards sooner or later. Also the pairs that you make don’t affect any of the other pairs because you can only make pairs using the same 2 or in the one case 3 cards.

**Activity 7:**

1. The board, the cards on the board, the deck
2. What you really need to do is find if there are cards that add up to equal 11 or a king, queen, and jack on the board all at once. If there is, you select the cards and it removes them. Then their spots are filled by new cards from the deck.
3. I am pretty sure that it contains all the state and behavior necessary to play the game. It has the replace the selected cards. And checks if the move is legal(which gets implemented in activity 9) it also checks to see if the cards add up to 11. And has another to check if the selection contains Jack Queen King.
4. a) instructor and newGame method,

b) anotherPlayPossible, isLegal, replace selected cards

c) jack of hearts, 6 of clubs, 2 of spades, ace of spades, 4 of hearts,

d)

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

{  
 System.out.println(cIndexes.get(i));

}

e) anotherPlayPossible would probably need to call cardIndexes first because it probably iterates through the list of the cards on the board and checks whether or not there are any that add up to 11 or jqk, so I null value could easily mess that iteration and checkup.

**Activity 8:**

1. similarities: They all have a deck and cards on a board. In all of them you have to add up the cards to get to a certain number. Object of the game is to get rid of all the cards.

Differences: in tens you add up to ten or you can select all the same values (all the kings, ect..) and remove them. You also have a 13 card board. In elevens you add up to 11s but you can also to jack queen king. You have a 9 card board. In thirteens, you add up to 13 and you can just discard kings. You have a 10 card bored.

1. I would probably just make a get method and set method for the each instance variable that would be different between boards. Because instance variables are private so we wouldn’t be able to reach them without a public method to do so with.

(in the board class)

Private int boardSize;

Public int getBoardSize()

{

return boardSize;

}

Public void setBoardSize(int newSize)

{

boardSize = newSize;

}

1. IsLegal() and anotherPlayPossible() are abstract.

They aren’t implemented.. It says that they will be implemented in activity 9. But, as for the documentation goes, it does seem like to covers all the differences because it looks for the specifics that elevens needs for the selections to work. And that is the difference that those all look at. The other difference has to do with the answer to part 2.

**Activity 9:**

1. Because it is used in the constructor to build the new board so it needs to be pre-defined which makes it easy to make it an instance variable that is final because you can’t change the board size.