Activity 5:

Buggy 1: isEmpty()

The deck returns that it is empty even though it is not. They could return size > or < 0 instead of size == 0 or they could have accidentally typed in a number other that 0 for return size == 0. So they could have typed in size == 9 since it is right next to the 0 key on the keyboard.

Buggy 2: size()

It said when they put in a deck of 1 card the tester returns that the deck has 0 cards in it even though there is 1. When they defined size in the constructor they could have misunderstood what cards.size() was and they could have thought that they needed to subtract 1 card like is you were running through a loop and needed to access the first one because it has an index of 0. They would have written it as size = cards.size() – 1; instead of just size = cards.size();.

Buggy 3: shuffle()

When the shuffle method is called it doesn’t do anything and the shuffled deck is the exact same as the original. This could be caused by the shuffle method returning the beginning deck instead of the new updated one.

Buggy 4: testOneCard()

The program returns null for the first card dealt in a 1 card deck. This could be caused by the deck only being one card and the program wont deal a deck with only one card because the programmer put a -1 in some place to count the amount of cards to deal in the deck.

Activity 6:

1)5+6, 6+5,

2) yes they have to be J,Q, and K because that is the only combo of three. The only other way to get rid of cards is in pairs of 2 which will always leave one card.

3) when more than one play is possible if you have the option to get rid of a card you have two of you want to take that play so for the next one you have as many options as possible depending on what cards they deal to you afterwards

Activity 7:

Q1:

Deck, how big the board is, how many cards are on the board at a time, what cards are on the board at a certain time.

Q2:

First you would have to shuffle the board, then lay out 9 cards face up and the deck to the side upside down, then you have to be able to combine two cards to make 11 or a jack, a queen, and a king and then it gets rid of those cards if they meet the conditions and deals the appropriate number of cards from the face down deck. If all the cards have been used it should then tell you that you won.

Q3:

Yes the deck class implements all of these.

Q4:

1. In the constructor and newGame method
2. The isLegal method and the anotherPlayIsPossible method

d)

public static printCards(ElevensBoard board) {

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

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

{

String card = cIndexes.get(j);

System.out.println(card);

)

e) anotherPlayIsPossible

Activity 8:

1. Each is very similar the only actual differences are how many cards are face up at a time with in tens uses a thirteen card board, elevens uses a nine card board, and thirteens uses a ten card board. The other main difference is how many points you have to add up to in order to get rid of a set of cards with in tens you need to get 10, elevens is eleven, and thirteens is thirteen.
2. They come from the card and deck classes
3. Yes they do because it is malleable enough to handle the settle differences between the three classes