Activity 2:

Q1: A deck can be composed of many cards, but a card cannot be composed of decks.

Q2: 6

Q3:

{"ace", "2", "3", "4", "5", "6", "7", "8", "9", "10", "jack", "queen", "king"};

{"spades", "hearts", "diamonds", "clubs"};

{11, 2, 3, 4, 5, 6, 7, 8, 9, 10, 0, 0, 0};

Q4: yes the order of ranks and pointvalues has to be the same.

Activity 3:

Q1:

public static String flip()

{

Random gen = new Random();

int num = gen.nextInt(3);

if (num == 1){return "heads"; }

else if (num == 2){return "heads";}

else {return "tails";}

}

Q2:

Q3: {3,2,1,4}

Activity 6:

Q1: 5s and 6c, or 6c and 5c

Q2: It must be jack, queen, and king because you would have to have pairs otherwise

Q3: I played it so that I would try to not use my jacks, queens, or kings because I thought that was only a last resort because it would open up more possibilities rather than just getting rid of 2 cards

Activity 7:

Q1:

* current cards
* cards in deck
* variable to see if there are no more possible solutions

Q2:

* draw board
* initialize deck
* deal cards
* have user pick cards and swap
* repeat 4th until either wins or loses

Q3: yes

Q4a: in the beginning after everything is initialized

Q4b: anotherPlayIsPossible() and isLegal()

Q4c: 0, 1, 3, 6, 7

Q4d:

public static printCards(ElevensBoard board)

{

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

/\* Your code goes here. \*/

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

{

String rank = board.cardAt(cIndexes.get(i).rank());

String suit = board.cardAt(cIndexes.get(i).suit());

int pV = board.cardAt(cIndexes.get(i).pointValue());

System.out.println("Rank: " + rank + ", Suit: " + suit + ", Point Value: " + pV);

}

}

Q4e: anotherPlayisPossible()

Activity 8:

Q1:

State — a deck of cards and the cards “on the” board.

Behavior— to deal the cards, to remove and replace selected cards, to check for a win, to check if selected cards satisfy the rules of the game, to see if there are more legal selections available, and so on.

^^^similarities

differences – they have different ways to check for possibilities and different legal moves, they also have a different number of cards on board and different card values

Q2: using “super()” which calls the super class to initialize everything

Q3: yes they cover the main differences, but differences like different rank values are already covered because they can be initialized differently

Activity 9:

Q1: because it is an instance variable that is initialized separately from everything else

Q2: because it is not necessary. These tasks are achieved the same way

Q3: it would less efficient considering that you have many more methods that are the same within these 3 games