# Activity 2

1. A Deck is a list of cards
2. 6
3. ranks = {”ace”, ”king”, ”queen”, ”jack”, ”10”, ”9 ”, ”8 ”, ”7”… ”2”};

suits = {”spades”, ”hearts”, ”diamonds”, ”clubs”};

pointValues = {11, 10, 10, 10, 10, 9, 8, 7… 2};

1. ranks and pointValues order matters, suits order doesn’t

# Activity 3

1 and 2(In Blue Jay)

3. 3 and 2

# Activity 4

None

# Activity 5

None

# Activity 6

1. 5/5, 6 or
2. Yes. Otherwise you’re left with one all alone and no single card can make an 11
3. No. You never know what cards are coming next so you can’t strategize

# Activity 7

1. Spots for the stacks of cards. Cards. A discard pile.
2. Pick two cards. Check if they add to 11. If they do then remove them. Add new cards in their spots. If 3 cards are selected they must be JQK.
3. No
4. Constructor
5. isLegal(), replaceSelectedCards(), anotherPlayIsPossible()
6. J 6 2 A 4 at 0, 1, 3, 6 and 70
7. For(int c = 0; c < cIndexes.length; c++)

{

If(cIndexes != null)

System.out.println(cards[c]);

}

1. replaceSelectedCards(). To know what to swap and where to deal

# Activity 8

1. All have cards, a deck, replacement cards, algorithms for acceptable moves, game ending detection methods/interfaces
2. Constructor
3. No. there are different point amounts for the cards and they are built to do eleven’s specific things

# Activity 9

1. Because it simply returns the size
2. Because the abstract methods are not going to be as helpful
3. Yes this would work, but not as well because the abstract design has more, useful options to it’s class than the interface version. Yes they could be called polymorphically.