**Activity 2:**

1. A deck is a container object for a bunch of card objects that represents a deck
2. 6
3. –
   1. Ranks: {“two”, “three”, “four”, “five”, “six”, “seven”, “eight”, “nine”, “ten”, “jack”, “queen”, “king”, “ace”}
   2. Suits: {“Diamonds”, “Spades”, “Clubs”, “Hearts”}
   3. Values: {2, 3, 4, 5, 6, 7, 8, 9, 10, 10, 10, 10, 11}
4. Yes

**Activity 3:**

1. Done
2. Done
3. –

**Activity 6:**

1. (5 of spades, 6 of clubs) and (5 of clubs, 6 of clubs)
2. Yes because any triplet must be the tree face cards because normally you cannot match more than two cards
3. No because the game is completely determined by the sequence of cards in the deck because you cannot manipulate the cards that add up to 11

**Activity 7:**

1. A deck of cards
2. --
   1. Create new game
   2. Check if there are at least 9 cards remaining in the deck
   3. If so, draw 9 cards. Otherwise draw the rest of the cards in the deck.
   4. Look for a set of Jack, Queen, and King. If it exists, replace the three cards.
   5. Look for a set of two cards that adds to 11. If it exists, replace the two cards.
   6. If nothing was replaced in the last turn, then the game is lost.
   7. Repeat steps 2-6 until there are no cards left remaining in the deck or on the board.
   8. Game is won.
3. Yes
4. –
   1. In the newGame() method and in the constructor

for (Integer i : cIndexes)

{

System.out.println(board.cards[i].toString());

}

* 1. anotherPlayIsPossible(), isLegal()
  2. 0, 1, 3, 6, 7
  3. anotherPlayIsPossible()

**Activity 8:**

1. They are all designed around a 1-player system and all involve a full deck of cards. Also, none of them involve strategy.
2. The instance variable is initialized in the Board class. Inside constructor of ElevensBoard, the values are passed into the constructor of the superclass.
3. They cover all the differences because all of the methods that are exactly shareable between the card games are implemented in the Board class while the overlapping functions that require different implementations (anotherPlayIsPossible and isLegal) are abstract, and thus implemented in the respective board game subclasses.

**Activity 9:**

1. Since size is an instance variable (for each object) it gets defined and runtime and is accessible this way. This means it does not need setters and getters because it is already defined.
2. Because removing and replacing cards is a features of all games, whether 13, 11, etc. This means it does not need to be an abstract method and instead can be defined completely in the Board class.
3. All the methods would still work as they would all be called polymorphically, but you would have to define the methods for each subclass implementing the interface.