Activity 7

1. What items would be necessary if you were playing a game of Elevens at your desk (not on the computer)? List the private instance variables needed for the ElevensBoard class.
   1. Answer: A deck of cards, and a list of cards that are being shown on the game board. The private instance variables needed for the ElevensBoard class are Deck[] and Card[].
2. Write an algorithm that describes the actions necessary to play the Elevens game.
   1. Answer:
      1. Shuffle the deck
      2. Deal 9 cards
      3. While cards can be removed::
         1. If 2 number cards can be added up for a value of 11:
            1. Remove those 2 cards
            2. Replace those 2 empty spots
         2. Else if there is a J, K, and Q:
            1. Remove those 2 cards
            2. Replace those 3 empty spots
      4. If there are more cards:
         1. Game over and lose
      5. Else:
         1. Game over and win
3. Now examine the partially implemented ElevensBoard.java file found in the **Activity7 Starter Code** directory. Does the ElevensBoard class contain all the states and behavior necessary to play the game?
   1. Answer: No, because there is no behavior/method to remove the cards.
4. ElevensBoard.java contains three helper methods. These helper methods are private because they are only called from the ElevensBoard class.
   1. Where is the dealMyCards method called in ElevensBoard?
      1. Answer: It is called in the ElevensBoard constructor and the newGame() method.
   2. Which public methods should call the containsPairSum11 and containsJQK methods?
      1. Answer: containsPairSum11 and containsJQK should be called in anotherPlayIsPossible() and isLegal() methods.
   3. It’s important to understand how the cardIndexes method works, and how the list that it returns is used. Suppose that cards contain the elements shown below. Trace the execution of the cardIndexes method to determine what list will be returned. Complete the diagram below by filling in the elements of the returned list, and by showing how those values index cards. Note that the returned list may have less than 9 elements.
      1. Answer (index in parentheses):
         1. J (0), 6(1), 2(2), A(3), 4(4)
   4. Complete the following printCards method to print all of the elements of cards that are indexed by cIndexes.
      1. Answer:

| public static printCards(ElevensBoard board){  List<Integer> cIndexes = board.cardIndexes();  for (Integer kObj:cIndexes){  System.out.println(board.cardAt((int) kObj.intValue()));  } } |
| --- |

* 1. Which one of the methods that you identified in question 4b above needs to call the cardIndexes method before calling the containsPairSum11 and containsJQK methods? Why?
     1. Answer: isLegal() needs to call the cardIndexes method before calling the containsPairSum11 and containsJQK methods because before placing cards on the board from the deck, it should be checked to see if that is even possible. Otherwise, the null values would be moved around in the lists.

Activity 8

1. Discuss the similarities and differences between Elevens, Thirteens, and Tens.
   1. Answer: The differences between Elevens, Thirteens, and Tens is that there are different moves and values needed per game, and there are different cards and numbers of cards. The similarities between Elevens, Thirteens, and Tens is that they all utilize cards and decks, there are winners and discard decks, and you need to get either a specific value or pair of face cards.
2. As discussed previously, all of the instance variables are declared in the Board class. But it is the ElevensBoard class that “knows” the board size, and the ranks, suits, and point values of the cards in the deck. How do the Board instance variables get initialized with the ElevensBoard values? What is the exact mechanism?
   1. Answer: The Board instance variables get initialized with the ElevensBoard values through the constructors. The exact mechanism for this is the keyword “super.”
3. Now examine the files Board.java, and ElevensBoard.java, found in the Activity8 Starter Code directory. Identify the abstract methods in Board.java. See how these methods are implemented in ElevensBoard. Do they cover all the differences between Elevens, Thirteens, and Tens as discussed in question 1? Why or why not?
   1. Answer: The differences between Elevens, Thirteens, and Tens are covered. That is because the isLegal() and anotherPlayIsPossible() methods are mainly about the rules and card numbers. The specific details are implemented in Board.