Lab Elevens Questions – Activity 6, 7, 8, 9

Questions A6:

1. List all possible plays for the board 5♠ 4♥ 2♦ 6♣ A♠ J♥ K♦ 5♣ 2♠

5♣ & 6♣

5♠ & 6♣

1. If the deck is empty and the board has three cards left, must they be J, Q, and K? Why or why not?

Yes, because the only pairs of 3 on the board can be J, Q, and K.

1. Does the game involve any strategy? That is, when more than one play is possible, does it matter which one is chosen? Briefly explain your answer.

Yes, waiting to remove a Jack, King, and Queen until all other choices have been eliminated can give a greater possibility of getting another match.

Questions A7:

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.

If I were playing Elevens, I would need a card deck and a place to set the cards.

The private instance variables needed for the ElevensBoard are cards, suits, ranks, point values, the board size, and the deck.

1. Write an algorithm that describes the actions necessary to play the Elevens game.

Create a playing board. Deal a certain number of cards. Replace pairs of two cards that add up to 11, or replace a King, Queen, or Jack. Repeat this process until there are no pairs of two cards that add up to 11 and there are no pairs containing a King, queen, and jack, or until there are no cards left.

1. Now examine the partially implemented ElevensBoard.java file found in the Activity7 Starter Code directory. Does the ElevensBoard class contain all the state and behavior necessary to play the game?

Yes, but the code segments for the isLegal, anotherPlayIsPossible, containsPairSum11, and containsJQK methods are missing.

1. ElevensBoard.java contains three helper methods. These helper methods are private because they are only called from the ElevensBoard class.
2. Where is the dealMyCards method called in ElevensBoard?

The constructor and newGame().

1. Which public methods should call the containsPairSum11 and containsJQK methods?

isLegal() and anotherPlayIsPossible().

1. It’s important to understand how the cardIndexes method works, and how the list that it returns is used. Suppose that cards contains 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.

0, 1, 3, 6, 7

1. Complete the following printCards method to print all of the elements of cards that are indexed by cIndexes.

public static printCards(ElevensBoard board)

{

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

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

{

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

}

}

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?
2. anotherPlayisPossible() because cardIndexes removes null entries.

Questions A8:

1. Discuss the similarities and differences between Elevens, Thirteens, and Tens. They all use the same deck size, ranks, suits, and point values. They also replace cards that match a criterion with undealt cards. There is a difference in that criterion per game though. In elevens, the point values of a pair must add up to 11 or the pair to be removed must be a pair of 3 faces. In tens, the point values have to add up to 10, or the pair to be removed must be a pair of face cards with the same rank. In Thirteens, the point values have to add up to 13 or the card to be removed must be a king. There is also a difference in the cards per board.
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?

The variables are initialized on the Board class, but the elevens board passes the values into the Board class.

1. 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?

Abstract methods cover all the differences between Elevens, Thirteens, and Tens because those boards initialize the abstract methods with separate codes that differ from game to game.

Question A9:

1. The size of the board is one of the differences between Elevens and Thirteens. Why is size not an abstract method?

Because size is a class initialized instance variable.

1. Why are there no abstract methods dealing with the selection of the cards to be removed or replaced in the array cards?

Because selection and removal of cards is initialized the same way, no matter the game.

1. Another way to create “IS-A” relationships is by implementing interfaces. Suppose that instead of creating an abstract Board class, we created the following Board interface, and had ElevensBoard implement it. Would this new scheme allow the Elevens GUI to call isLegal and anotherPlay IsPossible polymorphically? Would this alternate design work as well as the abstract Board class design? Why or why not?

public interface Board

{

boolean isLegal(List<Integer> selectedCards)

;

boolean anotherPlayIsPossible()

;

}

Yes, but all the methods that were implemented in the abstract Board would need to be implemented in separate classes.