Name:   
Date:

**19.03 Elevens Lab Worksheet**

**Directions**: Make note of your responses to the following questions as you work through the activities and exercise in the lesson.

**Activity 6 Questions**

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

5♠ 6♣ 6♣ 5♣

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. Every card has a corresponding pair except for JQK, which can only be matched with all three.

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.

Not necessarily. It is based on luck.

**Activity 7 Questions**

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.

A deck of cards will be necessary if I were playing a game of Elevens on my desk. Arrays for ranks, suits, and point values.

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

if(2 cards are chosen && sum =11){

replace cards and deal two;

}

if(3 cards are chosen && points value = 11+12+13){

replace cards and deal three;

}

if(card.size != 0 and no moves left){

lose game;

}

if(card.size == 0){

win game;

}

1. In the partially-implemented ElevensBoard.java file, does the class contain all the state and behavior necessary to play the game? Explain.

Yes. The methods contain the algorithms above.

1. 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?

in newGame() and in the ElevensBoard constructor.

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

methods?

in islegal() method and in anotherPlayIsPossible().

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
| **cards** | J♥ | 6♣ | null | 2♠ | null | null | A♠ | 4♥ | null |
| **returned**  **list** | 0 | 6 |  | 2 |  |  | 1 | 4 |  |

* 1. 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 nine elements.
  2. 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?

islegal() , we have to count the points of each card in the selected list and determine if a group of 11 or a group of JQK can be made.

**Activity 8 Questions**

1. Discuss the similarities and differences between the games *Elevens*, *Thirteens*, and *Tens*.

Similarities would be the board superclass. They all have deal and shuffle method. The differences would be the abstract methods they would inherit from the board class. These abstract methods are checking islegal()

1. The instance variables for cards and deck 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?

super() could be used in elevensboard. ElevensBoard would then inherit the instance variables of the board class.

1. List the abstract methods in Board.java. 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?

isLegal() and anotherPlayIsPossible() are both abstract methods. These are different between Elevens, Thirteens, and Tens because they will have different rules. They cover all the differences as the other methods such as deal and shuffle can be used by all.

**Activity 9 Exercise Results**

1. After running the ElevensGUIRunner.java class, describe what you see and experience. Take a picture of the screen and paste it below, if you like, along with the description.

it runs smoothly similar to the elevens.jar.

**Activity 9 Questions**

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

This is because size is an instance variable. size returns card.length, thus the size depends on the length of the card.

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

removing cards or replacing still works the same way for all three versions of the game. Thus elevensboard, thirteenssboard, etc. can just inherit this method.

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 anotherPlayIsPossible 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();

}

isLegal() and anotherPlayIsPossible() can both be called polymorphically. I think this alternate design would work similarly to the abstract board class design.