AP® Computer Science A Elevens Lab Student Guide

The AP Program wishes to acknowledge and thank the following individuals for their contributions in developing this lab and the accompanying documentation.
Michael Clancy: University of California at Berkeley
Robert Glen Martin: School for the Talented and Gifted in Dallas, TX
Judith Hromcik: School for the Talented and Gifted in Dallas, TX

Activity 9: Implementing the Elevens Board

Introduction:

In Activity 8, we *refactored* (reorganized) the original `ElevensBoard` class into a new `Board` class and a much smaller `ElevensBoard` class. The purpose of this change was to allow code reuse in new games such as Tens and Thirteens. Now you will complete the implementation of the methods in the refactored `ElevensBoard` class.

Exercises:

1. Complete the `ElevensBoard` class in the **Activity9 Starter Code** folder, implementing the following methods.

   Abstract methods from the `Board` class:

   a. `isLegal` — This method is described in the method heading and related comments below. The implementation should check the number of cards selected and utilize the `ElevensBoard` helper methods.

```
/**
 * Determines if the selected cards form a valid group for removal.
 * In Elevens, the legal groups are (1) a pair of non-face cards
 * whose values add to 11, and (2) a group of three cards consisting of
 * a jack, a queen, and a king in some order.
 * @param selectedCards the list of the indexes of the selected cards.
 * @return true if the selected cards form a valid group for removal;
 *         false otherwise.
 */
@Override
public boolean isLegal(List<Integer> selectedCards)
```

↑ not Board

note that the helper methods are not yet written ∴ won't be able to check it out yet

b. `anotherPlayIsPossible` — This method should also utilize the helper methods. It should be very short.

in ElevensBoard

```
/**
 * Determine if there are any legal plays left on the board.
 * In Elevens, there is a legal play if the board contains
 * (1) a pair of non-face cards whose values add to 11, or (2) a group
 * of three cards consisting of a jack, a queen, and a king in some order.
 * @return true if there is a legal play left on the board;
 *         false otherwise.
 */
@Override
public boolean anotherPlayIsPossible()
```

`ElevensBoard` helper methods:

c. `containsPairSum11` — This method determines if the selected elements of `cards` contain a pair of cards whose point values add to 11.

```
/**
 * Check for an 11-pair in the selected cards.
 * @param selectedCards selects a subset of this board.  It is this list
 *                      of indexes into this board that are searched
 *                      to find an 11-pair.
 * @return true if the board entries indexed in selectedCards
 *              contain an 11-pair; false otherwise.
 */
private boolean containsPairSum11(List<Integer> selectedCards)
```

d. `containsJQK` — This method determines if the selected elements of `cards` contains a jack, a queen, and a king in some order.

```
/**
 * Check for a JQK in the selected cards.
 * @param selectedCards selects a subset of this board.  It is this list
 *                      of indexes into this board that are searched
 *                      to find a JQK-triplet.
 * @return true if the board entries indexed in selectedCards
 *              include a jack, a queen, and a king; false otherwise.
 */
private boolean containsJQK(List<Integer> selectedCards)
```

When you have completed these methods, run the `main` method found in `ElevensGUIRunner.java`. Make sure that the Elevens game works correctly. Note that the cards directory must be in the same directory with your `.class` files.

Questions:

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

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

3. 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();
}
```

Activity 10: ThirteensBoard (~~Optional~~ Not?)

Introduction:

The purpose of this activity is to create the Thirteens game using the knowledge gained from implementing the Elevens game.

Exploration:

The rules for the Thirteens game are repeated below:

Thirteens

A game related to Elevens, called Thirteens, uses a 10-card board. Ace, 2, ... , 10, jack, queen correspond to the point values of 1, 2, ..., 10, 11, 12. Pairs of cards whose point values add up to 13 are selected and removed. Kings are selected and removed singly. Chances of winning are claimed to be about 1 out of 2.

Exercises:

1. The **Activity10 Starter Code** folder contains all the code for a complete working Elevens game. Review the code in the `ElevensBoard` class. Identify the changes that would be necessary to implement the Thirteens game.

2. Copy and paste the `ElevensBoard.java` file into a new file, `ThirteensBoard.java`. Make the necessary changes to this file to implement the Thirteens game.

3. The **Activity10 Starter Code** folder also contains the `ElevensGUIRunner.java` file that is shown below. This program creates the board (an `ElevensBoard` object). Then it creates the GUI (a `CardGameGUI` object). Finally, it displays the GUI by calling its `displayGame` method. Review the code in the `ElevensGUIRunner` class as shown below. Identify the changes that would be necessary to implement the Thirteens game.

```
/**
 * This is a class that plays the GUI version of the Elevens game.
 * See accompanying documents for a description of how Elevens is played.
 */
public class ElevensGUIRunner {

   /**
    * Plays the GUI version of Elevens.
    * @param args is not used.
```

```
     */
    public static void main(String[] args) {
        Board board = new ElevensBoard();
        CardGameGUI gui = new CardGameGUI(board);
        gui.displayGame();
    }
}
```

4. Copy and paste the `ElevensGUIRunner.java` file into a new file, `ThirteensGUIRunner.java`. Make the necessary changes to this file to implement the Thirteens game.

↑ Be thoughtful about this!!

5. Run the `ThirteensGUIRunner` program and test your new Thirteens game.