1 Lab Assignment 5

This is a lab exercise to have you build out some code you will use in a later homework assignment as well as verify that you can use `JUnit` for testing.

1.1 The Application

A standard double-12 set of dominoes has 91 dominoes each of which has two sides with some number of dots. The dominoes and their dots are distributed as follows.

- $(0, 0)$
- $(1, 0)$ $(1, 1)$
- $(2, 0)$ $(2, 1)$ $(2, 2)$
- $(3, 0)$ $(3, 1)$ $(3, 2)$ $(3, 3)$
- etc.
- $(12, 0)$ $(12, 1)$ $(12, 2)$ ... $(12, 12)$

This makes a total of $0 + 1 + ... + 13 = 91$ individual dominoes.

Many dominoes games involve each player choosing a "hand" of dominoes and then chaining the dominoes together based on the dots. (And the order of which number comes first in the pairs mentioned above does not matter.) A hand consisting of $(1, 1)$, $(1, 7)$, $(1, 10)$, $(1, 12)$, $(6, 7)$, $(6, 10)$, for example, could be chained together to form

$$(12, 1)(1, 1)(1, 7)(7, 6)(6, 10)(10, 1)$$

1.2 The Lab

In your later homework exercise you will be asked to chain dominoes using recursion. For now, the assignment is just to build out the underlying code structure.

1. The data payload layer of your code is a class for an individual domino. You will need instance variables for the two dot counts and an accessor for the dot counts. I recommend you use an `ArrayList` of length two to

store the dot counts and `static final int` constants perhaps called `SIDEA` and `SIDEB` so that you can write one accessor. The two sides of the domino are intended to be interchangeable, so your code should be written to go either side.

You will also need a `boolean` instance variable perhaps named `dealt` to indicate whether or not that domino has been dealt to a player.

You have been given an `interface` for the `OneDomino` class.

2. You have been given a driver program for this lab. It will invoke methods in a middle-layer application class to construct a game (by constructing the list of 91 dominos) and to deal a hand of twelve dominos.

   Dealing a hand consists of generating random numbers using `Random` to subscript into your `ArrayList` of 91 dominoes and "move" that `OneDomino` from the list of available dominoes into an `ArrayList` for the hand.

   THINK ABOUT HOW YOU WANT TO DO THIS!!!! The purpose of the `dealt` variable is so your list of available dominoes is of fixed length (but some have already been used). You can either make the hand a list of integer subscripts into the original array, or you can copy the dominoes, or you can ....

   You have been given an `interface` for the `DominoGame` class.

1.3 Testing with `JUnit`.

You are now to write a `JUnit` module that will test certain aspects of your code.

Specifically, you need to

1. Test that you have exactly 91 dominoes in your game.
2. Test that you have exactly 12 dominoes in the hand that was dealt.
3. Run a double loop to verify that you have no duplicate dominoes in your hand. (This would indicate that your selection process hadn't looked properly at the `dealt` variable.)