Exercise 1: Trace Race (Practice with ArrayLists)

Here are some of the methods you can use with an ArrayList of Strings:

```java
void add(String s)
    Add s to the end of the list.

void add(int index, String s)
    Add s at the specified position (move other strings over to make room). Error if the position is not valid.

String get(int index)
    Return the string at the specified position. Error if the position is not valid.

String remove(int index)
    Remove and return the string at the specified position. Error if the position is not valid.

String set(int index, String s)
    Replace the string at the specified position with s, and return the replaced string. Error if the position is not valid.

int size()
    Return the number of strings currently in the list.

int indexOf(String s)
    Return the position of the first occurrence of s in the list, or -1 if it is not in the list.
```

Divide into teams. Each team has two variables, L1 and L2, both of type ArrayList<String>. Arrange your cards so that L1 contains the following 13 strings, and L2 is empty:

"OH GREAT WIZARD CAN YOU SPARE A CUP OF JAVA? WHAT!!"

Now your Team Leader will give you a piece of code that modifies the two lists. Your job is to act out each line of code, moving strings from L1 to L2 as appropriate. To act out the substring operation, you can tear the card.

See who gets to the end of the code (correctly) first!
Exercise 2: Crazy Arrays (Scaler and Array Parameters)

In Java, arrays are Objects. This means that they’re treated differently than primitives (e.g., int, double, or boolean) when passed as parameters. To help you understand the difference, act out the code below, with one person playing the part of each method, and everyone else taking turns to tell them what to do when the next statement executes. Remember that in Java, when a method is called, the calling method passes copies of the values of the arguments to the called method.

```java
public static void changeAll(int num, int[] numArray) {
    int k;
    for (k=0; k<numArray.length; k++) {
        numArray[k] += num;
    }
    num = 0;
    numArray = null;
    if (numArray == null) {
        System.out.println("The array is null");
    } else {
        System.out.println("The array is NOT null");
    }
}

public static void main(String[] args) {
    int k;
    int myNum;
    int[] myArray;
    myNum = 5;
    myArray = new int[6];
    for (k=0; k<myArray.length; k++) {
        myArray[k] = k;
    }
    changeAll(myNum, myArray);
    if (myArray == null) {
        System.out.println("The array is null");
    } else {
        System.out.println("The array is NOT null");
        for (k=0; k<myArray.length; k++) {
            System.out.println(myArray[k]);
        }
    }
    System.out.println("myNum is: " + myNum);
}
```
Exercise 3: PigLatin (More Coding Fun)

The rules for translating an English word to Pig-Latin are as follows:

- If the word starts with a vowel, it is unchanged.
- Otherwise, all of the consonants at the beginning of the word (up to the first vowel) are moved to the end of the word, preceded by a dash (for readability), and followed by “ay”.

For example, the sentence “I love Madison in the springtime” would be translated to “I ove-lay adison-May in e-thay ingtime-spray”.

For this exercise, you will complete the PigLatin class started on the next page, which will allow users of the class to translate an English word to Pig-Latin.

First, make sure you understand the translation rules by translating the following phrases into Pig-Latin:

- Hello world
- I love WESCS
- the car says ooga ooga

Part (a): Take a look at the incomplete PigLatin class on the next page. Make sure you understand what each method is supposed to do.
import java.util.*;

public class PigLatin {

    // translate: translate the English word to Pig-Latin
    public static String translate(String englishWord) {
        /* part (c) */
    }

    // firstVowelPos: return the position of the first vowel
    // in the English word; return -1 if there
    // is NO vowel in the word
    public static int firstVowelPos(String englishWord) {
        /* part (b) */
    }

    // main: get an English word from the user
    // then translate to Pig Latin and print the result
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter English word to translate: ");
        String word = sc.next();
        String translation = translate(word);
        System.out.println("In Pig Latin, your word is " + translation);
    }
}

Part (b): Divide into groups, each working on one laptop. Write the firstVowelPos method, which returns the position of the first vowel in the englishWord or returns -1 if there is no vowel in the word. Here are some examples:

<table>
<thead>
<tr>
<th>englishWord</th>
<th>Result of calling firstVowelPos</th>
</tr>
</thead>
<tbody>
<tr>
<td>hello</td>
<td>1</td>
</tr>
<tr>
<td>ice</td>
<td>0</td>
</tr>
<tr>
<td>spring</td>
<td>3</td>
</tr>
<tr>
<td>qyzzx</td>
<td>-1</td>
</tr>
</tbody>
</table>

Part (c): Now describe in English how the translate method of the PigLatin class should work (using the rules for translating from English to Pig-Latin given above, and making use of the firstVowelPos method). Then write the actual code and test it.

If you finish early, change the main method so that it asks the user to type in a whole English sentence, then translates it word-by-word, then prints the whole translation.
Exercise 4: Boats, Planes, but no Automobiles (Logical Thinking)

Puzzle 1: Alice and Bob are on separate islands. Bob is sick, and Alice has the medicine. Eve has a boat and a chest that can be locked. She is willing to transport objects between Alice and Bob, but only in the chest, and if the chest is unlocked, she will steal whatever is inside. If both Alice and Bob have a padlock and a key such that their own key only opens their own lock, how can Alice send Bob the medicine so that Eve won’t steal it?

Puzzle 2: Three identical airplanes start at the same airport. Each plane has a full fuel tank holding just enough fuel to allow the plane to travel half the distance around the world. These airplanes have the special ability to transfer fuel between their tanks in mid-flight. Devise a scheme that will allow one airplane to travel all the way around the world without ever landing to get more fuel. (Assume that when a plane is out of fuel it can glide in to land.)
L2.add(L1.remove(1));
L2.add(0, L1.remove(10));
L2.add(L1.remove(8));
L2.add(L1.indexOf("CAN") + 1, L1.remove(3));
if (L1.get(4).equals("A")) {
    L2.add(1, L1.remove(4));
}
L2.add(L1.remove(7));
if (L1.size() > 3) {
    L2.add(5, L1.remove(3));
} else {
    L2.add(2, L1.remove(3));
}
L2.set(5, L2.get(5).substring(2,5));
if (L1.indexOf("OH") == -1) {
    L2.add(4, L1.remove(0));
} else {
    L2.add(4, L1.remove(1));
}
OH GREAT WIZARD

OH GREAT WIZARD CAN YOU SPARE A CUP OF JAVA WHAT

!! OH GREAT WIZARD CAN YOU
SPARE A CUP

OF JAVA?

WHAT!! OH

GREAT WIZARD CAN

CUP OF JAVA

YOU SPARE A
OH GREAT WIZARD

CAN YOU SPARE

A CUP OF

JAVA WHAT

!!
<table>
<thead>
<tr>
<th>SPARE</th>
<th>SPARE</th>
<th>SPARE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPARE</td>
<td>SPARE</td>
<td>SPARE</td>
</tr>
<tr>
<td>SPARE</td>
<td>SPARE</td>
<td>SPARE</td>
</tr>
<tr>
<td>SPARE</td>
<td>SPARE</td>
<td>SPARE</td>
</tr>
<tr>
<td>SPARE</td>
<td>SPARE</td>
<td>SPARE</td>
</tr>
<tr>
<td>SPARE</td>
<td>SPARE</td>
<td>SPARE</td>
</tr>
<tr>
<td>SPARE</td>
<td>SPARE</td>
<td>SPARE</td>
</tr>
<tr>
<td>SPARE</td>
<td>SPARE</td>
<td>SPARE</td>
</tr>
<tr>
<td>SPARE</td>
<td>SPARE</td>
<td>SPARE</td>
</tr>
</tbody>
</table>