U6L2: Traversing Arrays

Looping through an array

It is often useful to loop through all the elements of an array one at time. For example, if we were given the following array of Scooby Doo characters

names

0	1	2	3	4
"Shaggy"	"Scooby"	"Velma"	"Daphne"	"Fred"

We could write a for loop that iterates from `0` to `4` to print each out on a separate line.

```
1 for(int i = 0; i < 5; i++){
2     System.out.println(names[i]);
3 }
```

A better way to write this would be to use `names.length` instead of the *magic number* `5`

```
1 for(int i = 0; i < names.length; i++){
2     System.out.println(names[i]);
3 }
```

Off By One Errors

The most common errors when traversing arrays are "off by one" types. The following loop is supposed to print all names in reverse order. Can you find the error in the following code?

```
1 for(int i = names.length; i >= 0; i--){
2     System.out.println(names[i]);
```

For Each Loops

Also called *enahnced for loops* were added to Java in version 5. They result is a more readable for loop that can be used with any Java *collection* such as *Array* and *ArrayList* objects.

To use a for each loop, we must use the following header syntax:

```
for(arrayType varName: arrayName)
```

arrayType is the data type held in the array.
varName is the name of a temporary variable that will hold the current element of an array.
arrayName is the variable name of the array.

Our Scooby Doo example can now be rewritten as:

```
1 for(String name: names){
2     System.out.println(name);
3 }
```

Advantages of the *for each* loop
readability
eliminates the chance of *off by one* errors
programmer does not need to be concerned with *indexing*.

For each loops can't modify an array.

Suppose I have a `grades` array that has been populated as shown below

```
1 int[] grades = {77, 92, 88, 95, 100};
```

I want to design a method that takes an integer array such as `grades` and resets all the values to zero. The following method would do this:

```
1 public static void resetGrades(int[] grades){
2     for(int i = 0; i < grades.length; i++){
3         grades[i] = 0;
4     }
5 }
```

However, the for-each version would not. This is because `grade` is a local copy of the primitive value `grades[i]`. We must use a traditional for loop if we want to modify the array.

```
1 public static void resetGrades(int[] grades){
2     for(int grade: grades){
3         grade = 0;
4     }
5 }
```

LAB 023 Starter Code

```
 1 // write this in the same project as Student.java from LAB-022
 2 import java.util.Arrays;
 3
 4 public class Course{
 5
 6     public static final int MAX_STUDENTS = 30;
 7     private String name;
 8     private numStudents;
 9     private Student[] students;
10
11     // constructor
12
13     // methods
14
15 }
```

LAB-023 TODO: Student Class

Create the following constructor and instance methods

Constructor

Course(String name): takes a string used to set the `name` attribute. Initialize the `students` array here. Use `MAX_STUDENTS` to set the size. Set `numStudents` to zero.

Instance Methods

addStudent(): Takes a Student object and adds the student to `students` array **only if** there is room for the student in the class. Increment the `numStudents` variable.
getNumImproved(): Traverses the `students` array and returns the number of students in the course that have improved.
getMaxAverage(): Traverses the array of Students and returns the maximum average grade in the course.
getMinAverage(): Traverses the array of Students and returns the minimum average grade in the course.
printRoster(): Prints the roster of a course as example shown on next slide. Include numbers, names, and averages. Use the `for-each` type for this.

Output of printRoster() method

Roster for AP Computer Science A
1. Shaggy 71.0
2. Scooby 100.0
3. Velma 99.8
4. Daphne 92.4
5. Fred 81.5

- make your own test code, but you don't need to submit it.