Model 1 Example Code

The following examples are found in *Model1.java*. Open this file on your computer, and run the program. Record the output of each example in the space below.

```
1 int[] nums;
2 nums = new int[3];
3
4 nums[0] = 74;
5 nums[1] = 11;
6 nums[2] = 21;
7
8 System.out.println(nums.length);
9 System.out.println(nums);
```

The output is:

```
3
[I@7440e464
```

```
1 ArrayList<Integer> nums;
2 nums = new ArrayList<Integer>();
3
4 nums.add(74);
5 nums.add(11);
6 nums.add(21);
7
8 System.out.println(nums.size());
9 System.out.println(nums);
```

The output is:

```
3
[74, 11, 21]
```

Questions (20 min) Start time:

1. Compare the examples line by line, and summarize the differences.

a) Line 1:

Arrays are declared with square brackets; ArrayLists are declared with angle brackets. The array example uses the primitive type int, but the ArrayList example uses Integer.

b) Line 2:

When creating the array, you have to specify the length in brackets. When creating the ArrayList, you need parentheses after the <>'s.

c) Lines 3–6:

Arrays use square brackets to index a specific element. ArrayLists use the add method, and no indexes are required.

d) Line 8:

Arrays have a length attribute. ArrayLists have a size method.

2. What is the main difference in the output of these two examples?

Printing the array just displays its memory address. Printing the ArrayList shows the actual contents. (The reason why is because ArrayList has a toString method.)

3. What happens if you add the following code after Line 6 in the array example? Verify your answer by editing *Model1.java* and running the program.

```
nums[3] = 59;
```

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException:
Index 3 out of bounds for length 3
```

4. In *Model1.java*, comment out the line you just added in the previous question. Then add the following line to the ArrayList example. What is the resulting output?

```
nums.add(59);
```

```
4
[74, 11, 21, 59]
```

5. Based on your previous answer, what ability do ArrayLists have that arrays do not?

Their size can change, i.e., they can "grow" as you add new elements.
(They also provide a toString method, so you can print them directly.)

6. Add the following line to the ArrayList example. What is the result?

```
nums[0] = 33;
```

Compiler error:
The type of the expression must be an array type but it resolved to ArrayList<Integer>.

7. In the `ArrayList` example, is `nums` an *array* or an *object*? Justify your answer.

It's an object, because it has methods like add and size. You can't use it like an array, as shown in the compiler error.