Summary 16 – Arrays and ArrayList's

- What is this?

An ***array*** is a *collection of data organized by a single name*. One refers to the elements of the collection by using ***indices*** *(aka subscripts)*.

An ***ArrayList*** is also a *collection of data organized by a single name*. ArrayList is a class, hence has methods, can be extended, and so forth. They also resize themselves as needed. Because ArrayList's are objects and because they resize themselves, they are generally more convenient than the more primitive arrays.

- Examples

1. **Declaring** an array or ArrayList or (more generically) List.

```
double[] data;
Employee[] employees;

ArrayList<Double> scores;
ArrayList<Student> students;

List<Double> scores;
List<Student> students;
```

2. **Allocating space** for an array or ArrayList. Assuming the declarations above:

```
data = new double[50];
employees = new Employee[numberOfEmployees];

scores = new ArrayList<Double>();
students = new ArrayList<Student>();

// You also can declare and allocate space
// in a single statement:

double[] moreData = new double[100];
List<Student> moreStudents
                = new ArrayList<Student>();
// Note the use of interface-type (to be done later)
// for the above declaration.
```

3. **Getting (accessing) elements** in an array or ArrayList. Examples (assuming declarations above, and also see examples that follow):

```
... data[4] ...
... employees[k] ...

... scores.get(4) ...
... students.get(k) ...
```

4. **Setting (mutating) elements** in an array or ArrayList. Examples (assuming declarations above, and also see examples that follow):

```
data[4] = valueFromFile;
employees[k] = new Employee(...);

scores.set(4, scoreFromFile);
students.set(k, new Student(...)) ...

// You can also can add elements to the end
// of an ArrayList, and more:
students.add(new Student(...));
students.remove(j); // Remove element at index j,
                    // shifting the elements behind it
```

5. **Initializing** all elements of an array or ArrayList, **using old-style loops**:

```
Dog[] dogs = new Dog[numberOfDogs];

for (int k = 0; k < dogs.length; k++) {
    dogs[k] = new Dog(...);
}

List<Cat> cats = new ArrayList<Cat>();

for (int k = 0; k < numberOfCats; k++) {
    cats.add(new Cat(...));
}
```

6. **Counting** array or ArrayList elements that satisfy a given property, **using old-style loops**:

```
int count = 0;
for (int k = 0; k < dogs.length; k++) {
    if (... dogs[k] ...) {
        count++;
    }
}

int otherCount = 0;
for (int k = 0; k < cats.size(); k++) {
    if (... cats.get(k) ...) {
        otherCount++;
    }
}
```

7. **Summing** array or ArrayList elements, **using NEW-style loops**:

```
double totalWeight = 0;
for (Dog dog : dogs) {
    totalWeight += dog.weight();
}

int totalAge = 0;
for (Cat cat : cats) {
    totalAge += cat.age();
}
```

Note: you can NOT use the enhanced for loop to *modify* an element of an array or ArrayList and stick it back in the collection.

8. The "**histogram loop pattern**":
   a. Declare an array or ArrayList that will hold histogram values 0 .. *n*, where events are numbered and *n* is the biggest event that can occur.
   b. Initialize all elements of the histogram to 0.
   c. Repeatedly:
      i. Get one event (call it *m*) from which the histogram is to be built.
      ii. Increment the histogram count at index *m*.

The array or ArrayList now holds the desired histogram.

9. **Two (or more) dimensional arrays or ArrayList's**, as in these examples:

```
Integer[m][n] matrix = new Integer[m][n];
for (int j = 0; j < matrix.length; j++) {
    for (int k = 0; k < matrix[j].length; k++) {
        matrix[j][k] = ...
    }
}

List<List<Integer>> matrix =
                    new ArrayList<List<Integer>>();
for (int j = 0; j < m; j++) {
    matrix.add(new ArrayList<Integer>());
    for (int k = 0; k < n; k++) {
        matrix.get(j).add(...);
    }
}
```

The above examples show why arrays are often preferable to ArrayList's when two or more dimensions are required.

- For further study:
  - Chapter 7 of *Big Java*
  - The notes on:
    - Arrays and ArrayList's (10 ArraysAndArrayLists)
    - Wrapper classes and autoboxing (10 ArraysAndArrayLists)
    - Two-dimensional arrays (11 TwoDArrays)
  - The Arrays and Collections classes
  - This summary's *author:* David Mutchler