Arrays and References
The Original (Partial) Story

• An **array** is a group of similar variables, all of the same type, and with systematically related names that involve special syntax using […]

• Each **array element**, e.g., `a[0]`, `a[1]`, …, acts like a single variable of the type used in the declaration of array `a`

• The variable named `a.length` contains the number of array elements
int[] a = { 6, 18, 9, -10 };
The Full Story

• In addition, you need to know:
  – Arrays are *reference types*
  – The name of the array (e.g., `a` in the example) is a *reference to the entire collection of element variables* `a[0], a[1], ..., and a.length`
The Full Picture

```java
int[] a = { 6, 18, 9, -10 };
```

![Diagram of an array `a` with elements 6, 18, 9, -10 and `a.length` set to 4]
Arrays Are Reference Types

• You should now be able to predict what happens when you do the following:
  – Assign one array to another using =
  – Pass an array as a parameter to a method
  – Return an array from a method
  – Compare two arrays for equality with ==
  – But... what does equals do?
One of the Many Warts of Java

• The `equals` method for arrays does arguably the wrong thing: it compares reference values just like `==`
  
  – You might expect it would compare arrays “element-wise”, and the lengths of the arrays, but it does not
  
  – Fortunately, SpotBugs flags the use of `equals` and explains it is equivalent to `==`
What Can Be Done?

• You can try to write your own code to check whether two arrays are element-wise equal (but this is surprisingly hard to get right!)

• You can use code from the Java libraries in the package `java.util`
  – See the class `Arrays`
  – Use the static method `Arrays.deepEquals`
What Can Be Done?

• You can try to write your own code to check whether two arrays are element-wise equal (but this is surprisingly hard to get right!)

• You can use code from the Java libraries in the package `java.util`:
  – See the class `Arrays`
  – Use the static method `Arrays.deepEquals`
Best Practices for Arrays

• Avoid them in industrial-strength software
  – OK in exercises intended to demonstrate the basics of arrays (because there is so much Java code “in the wild” that uses arrays), and in simple throw-away programs

• Recommended alternatives:
  – Java libraries: `java.util.List` interface with `ArrayList` implementation
  – OSU CSE components: `Array`, `Sequence`
Resources

- **Java Tutorials**
  - [http://docs.oracle.com/javase/tutorial/java/nutsandbolts/arrays.html](http://docs.oracle.com/javase/tutorial/java/nutsandbolts/arrays.html)

- **Java for Everyone, Section 6.1**

- **Java Libraries API: Arrays**
  - [http://docs.oracle.com/javase/8/docs/api/](http://docs.oracle.com/javase/8/docs/api/)

- **Effective Java, Item 25**