Module 1 - Lecture 4

Loops and Arrays

Review

- Behavior in programs
- Expressions vs. Statements
- Code Blocks
- Scope
- Methods
- Comparison and Logical Operators
- Conditional statements

The problem...

So far, we have learned how to store data using variables. What would you do if I asked you to store a collection of Instructors' names and then print them to the screen? There must be a better way, right?

```
String tomName = "Tom";

String bethName = "Beth";

String waltName = "Walt";

System.out.println(tomName);

System.out.println(bethName);

System.out.println(waltName);
```

The problem continues...

What if I wanted to write a method that accepts any number of instructor names and be able to print all of them? Can I do that with variables?

Arrays!

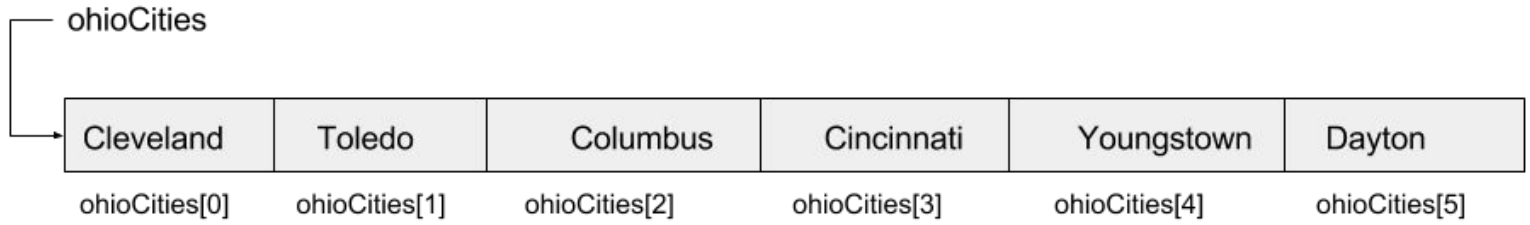
Arrays

An **array** is a series of objects of which all are the **same size and type**. Each item in an array is called an element.

You want to use arrays when you have data that is all related and act as the same kind of thing in the code, but you need multiple values for, like student names, quiz scores or inning scores for a baseball game.

Think of an array like an egg carton or a roller coaster train!

Declaration



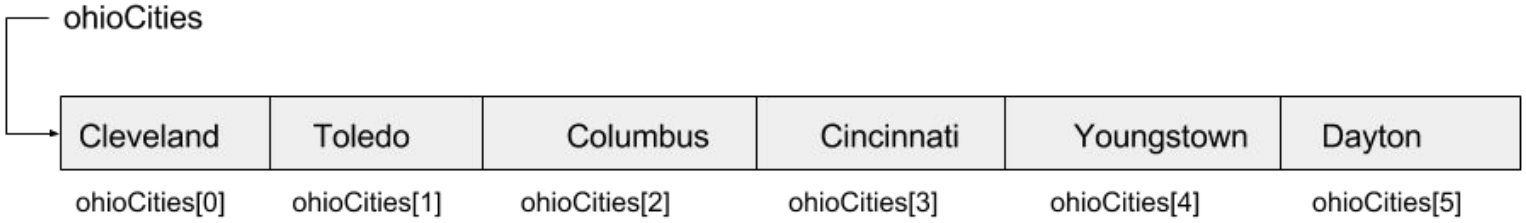

```
String[] ohioCities;              // Declaration

ohioCities = new String[6];       // Initialization

String[] ohioCities = new String[6]; // All in one
```

<element type> <variable name> <number of elements>

Array Assignment

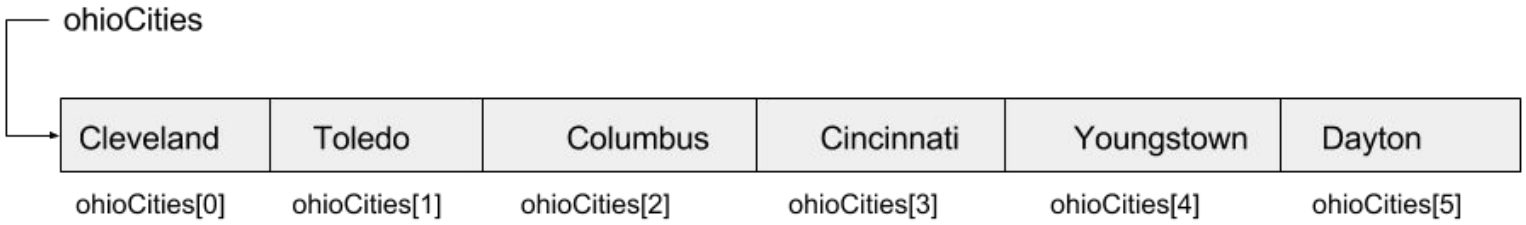



```
String[] ohioCities = new String[6];

ohioCities[0] = "Cleveland";        // Starts at 0!?
ohioCities[1] = "Toledo";
ohioCities[5] = "Dayton";
ohioCities[3] = "Cincinnati";
ohioCities[4] = "Youngstown";
ohioCities[2] = "Columbus";

// Alternative Syntax
String[] ohioCities = {"Cleveland","Toledo","Columbus"}
```

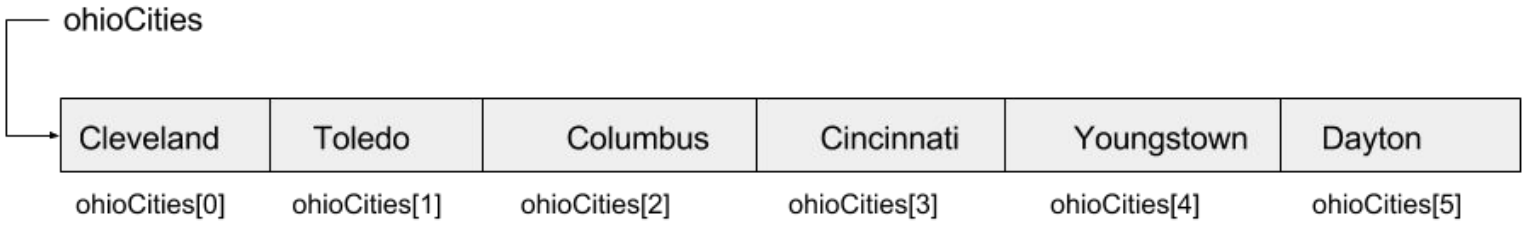
Access




```
// Print "Columbus" to the console.
System.out.println(ohioCities[2]);
```

Let's Code!

The gotchas!



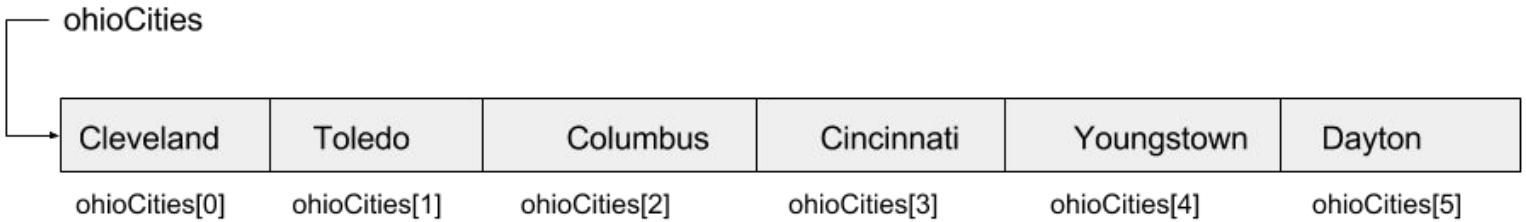
- We access arrays using square brackets.
- Arrays are zero indexed.
- Arrays are a fixed length.
- Arrays must be initialized before use.
- All elements must be of the same type.
- Arrays are reference types. WHAT?
- Arrays are stored together in memory (maybe). This is why we can use an index to address an element. The index is the offset from the start!

Loops!

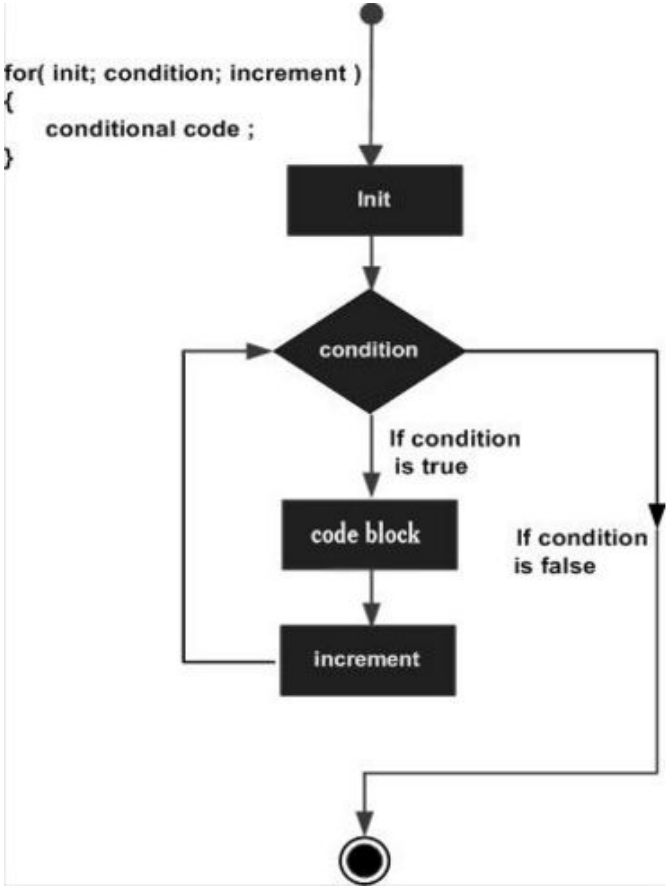
Remember this problem?

What if I wanted to write a method that accepts any number of instructor names and be able to print all of them?

How would we solve this problem for the array below?



Loops



Often we want to write code that runs multiple times. We do this with **loops!**

Unary operators and Shortcuts

Increment variable by 1 -> `x++;` `// equivalent to x = x + 1;`

Decrement variable by 1. -> `x--;` `// equivalent to x = x - 1;`

NOTE: Do not use the statements above outside of loops!

Assignment shortcuts.

```
x += 2;  // equivalent to x = x + 2;

x -= 5;  // equivalent to x = x - 5;

x *= 3;  // equivalent to x = x * 3;
```

Primitive vs. Reference

Primitive vs. Reference Types

Primitive or Value Types

Type	Size (bits)
boolean	1*
char	16
byte	8
short	16
int	32
long	64
float	32
double	64

*it actually depends on the JVM

Reference Types

Type	Size (bits)
Strings	???
Arrays	???
Objects	???

... really everything other than primitives.

Reference type assignment

What will be printed by the code below?

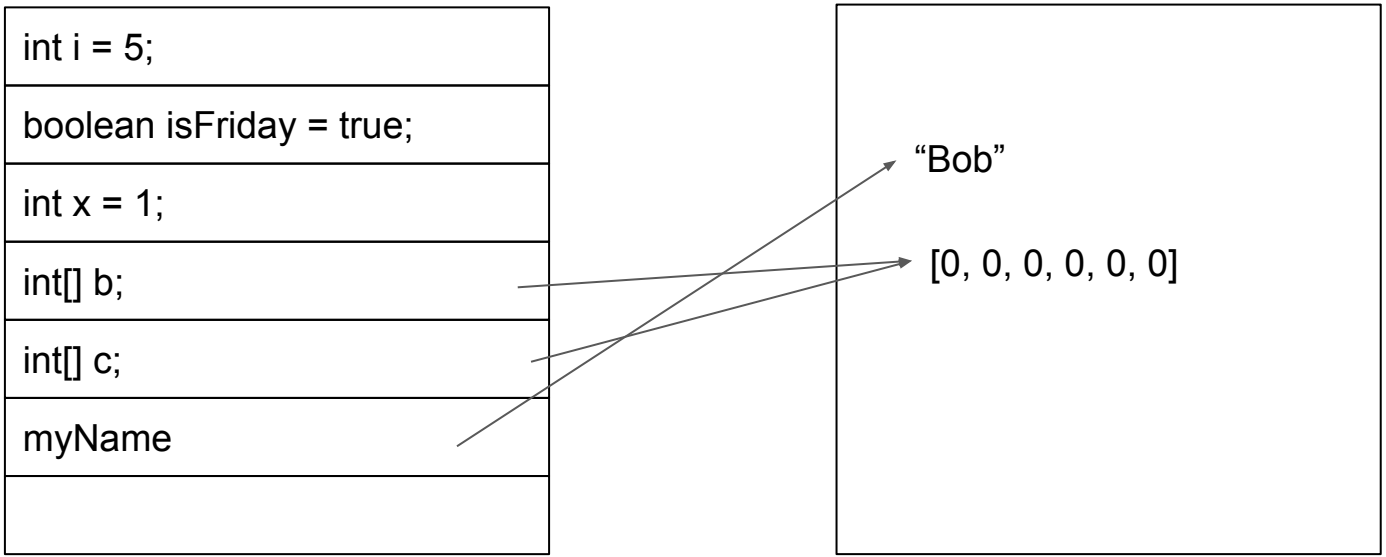
```
boolean[] first = { true, true, false, false };

boolean[] second = first;

second[2] = true;

System.out.print(first[2]);
```

Stack vs. Heap



QUESTIONS?