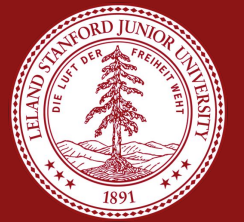
Arrays

Lecture 16

CS106A, Summer 2019

Sarai Gould && Laura Cruz-Albrecht

With inspiration from slides created by Keith Schwarz, Mehran Sahami, Eric Roberts, Stuart Reges, Chris Piech and others.



Announcements

GREAT JOB ON THE MIDTERM!

Announcements

- Midterms solutions will be released later this week.

Plan for Today

- Review: Characters and Strings
- Data Structures
- Arrays
- Storing Coffee Prices in an Array
- Pass by Reference and Pass by Value
- Cat Wake Ups
- ArrayLists (quickly)

Review: Looping over Strings

A common String programming pattern is looping over a String and operating on each character.

```
for (int i = 0; i < str.length(); i++) {
    char ch = str.charAt(i);
    // do something with ch here
}
```

Review: Looping over Strings

Another common String programming pattern is **building up a new string** by adding characters to it over time.

```
// Creates a new String containing digits 0 through 4
String str = "";
for (int i = 0; i < 5; i++) {
    str += i;
}
println(str);  // 01234
```

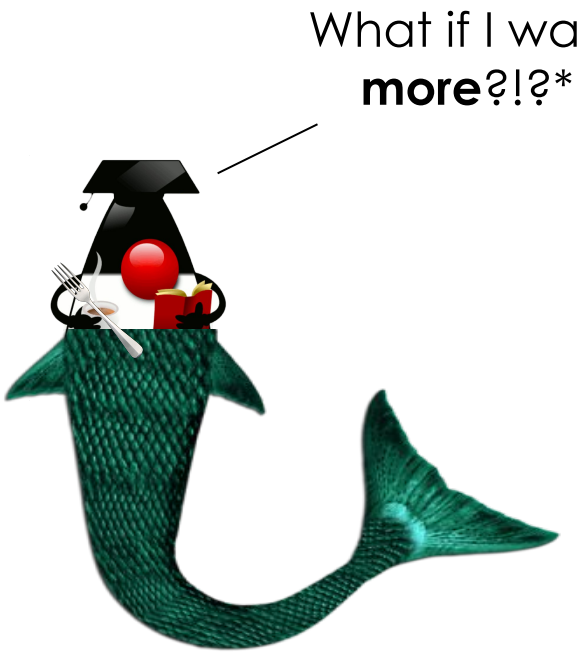
How Can We Store MORE Information?

How Can We Store MORE Information?

Most variables we've learned about so far can only store very limited data:

```
int num = 5;
double fraction = 0.2;
boolean isSummer = true;
char letter = 'c';
String phrase = "Hi!";
```

How Can We Store MORE Information?



* it's a little mermaid reference 🐟

Data Structures

Data Structures allow us to store more data in more interesting ways.

An **array** is one of these data structures!

Arrays

Arrays allow us to store data in a fixed size list.

With arrays, we can store lots of one type of data! We can have an array of `ints`, `booleans`, `GRects`, you name it!

Creating Our First Arrays

We can create arrays a few different ways.

Creating Our First Arrays

We can **set the values** we want in our arrays.

```
// A few examples
String[] bestFriends = {“Duke”, “Karel”}; // [“Duke”, “Karel”]

int[] favNumbers = {4, 7, 23}; // [4, 7, 23]

double[] specialNums = {1.41, 1.61, 2.83, 3.14}; // [1.41, 1.61, 2.83, 3.14]
```

Creating Our First Arrays

```
type[] name = {elements, to, add, to, array};
```

```
// A few examples
String[] bestFriends = {“Duke”, “Karel”}; // [“Duke”, “Karel”]

int[] favNumbers = {4, 7, 23}; // [4, 7, 23]

double[] specialNums = {1.41, 1.61, 2.83, 3.14}; // [1.41, 1.61, 2.83, 3.14]
```

Creating Our First Arrays

We can **set the length of the array** and let it have **default values**.

```
// A few examples
String[] fourStrings = new String[4]; // [null, null, null, null]

int[] sixInts = new int[6]; // [0, 0, 0, 0, 0, 0]

double[] threeDoubles = new int[3]; // [0.0, 0.0, 0.0]
```

Creating Our First Arrays

```
type[] name = new type[numberOfElements];
```

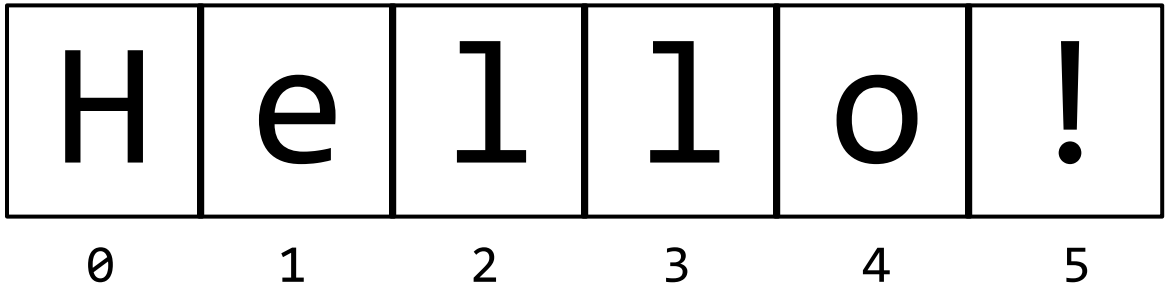
```
// A few examples
String[] fourStrings = new String[4]; // [null, null, null, null]

int[] sixInts = new int[6]; // [0, 0, 0, 0, 0, 0]

double[] threeDoubles = new int[3]; // [0.0, 0.0, 0.0]
```

Arrays Remind Me of Strings

There are some similarities that arrays have to Strings!

Strings

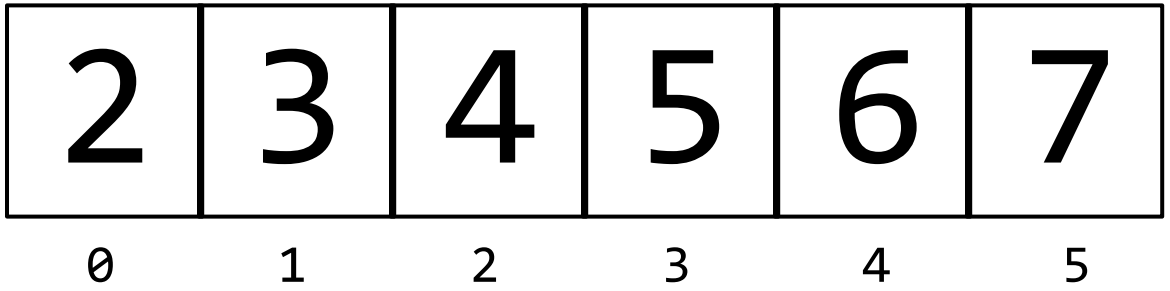
- Each character is assigned an index, going from 0 to length-1.
- There is a `char` at each index.




```
int strLen = text.length();             // 6
char last = text.charAt(strLen - 1);    // '!'
```

Arrays

- Each location is assigned an index, going from 0 to length-1.
- The type of data at each index depends on the type of array!




```
int arrayLen = myArray.length;       // 6
int last = myArray[arrayLen - 1];    // 7
```

Arrays vs. Strings

Array:

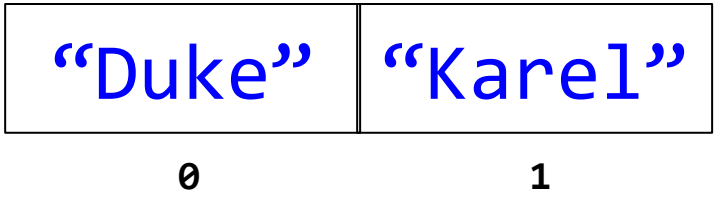
```
int[] myArray = new int[5];
OR
int[] myArray = {2, 3, 4, 5, 6, 7};


int arrayLen = myArray.length;

int first = myArray[0];

int last = myArray[arrayLen - 1];

// In arrays, we can change elements!
myArray[0] = 1;
```

String:

```
String text = "Hello!";


int strLen = text.length();

char first = text.charAt(0);

char last = text.charAt(strLen - 1);

// In Strings, we can't change elements!
// :(
```

Getting Array Elements

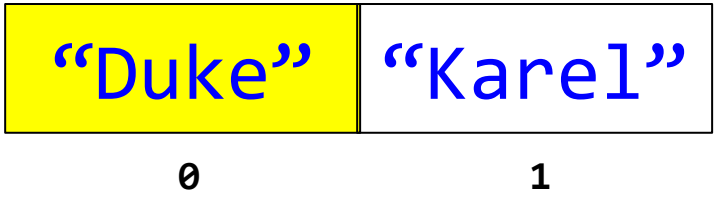
We can get values from our arrays!




```
// A few examples
String[] bestFriends = {“Duke”, “Karel”}; // [“Duke”, “Karel”]
```

Getting Array Elements

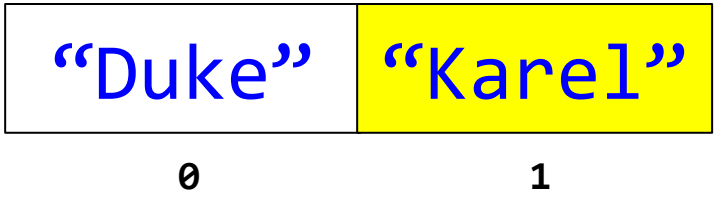
This is very similar to how we get elements from Strings. The indices also start at 0!




```
// A few examples
String[] bestFriends = {“Duke”, “Karel”}; // [“Duke”, “Karel”]

String firstFriend = bestFriends[0];
println(firstFriend); // “Duke”
```

Getting Array Elements

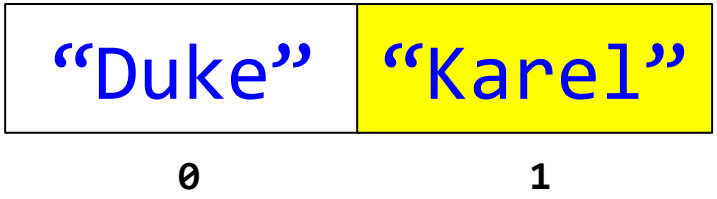
We can get values from our arrays!




```
// A few examples
String[] bestFriends = {“Duke”, “Karel”}; // [“Duke”, “Karel”]

String firstFriend = bestFriends[0];
println(firstFriend); // “Duke”

String secondFriend = bestFriends[1];
println(secondFriend); // “Karel”
```

Getting Array Elements

This is one way to get the last element in an array.

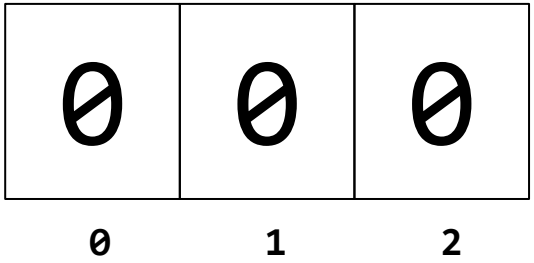



```
// A few examples
String[] bestFriends = {"Duke", "Karel"}; // ["Duke", "Karel"]

String firstFriend = bestFriends[0];
println(firstFriend); // "Duke"

String lastFriend = bestFriends[bestFriends.length - 1];
println(lastFriend); // "Karel"
```

Setting Array Elements

Maybe I want to change the elements in my array!

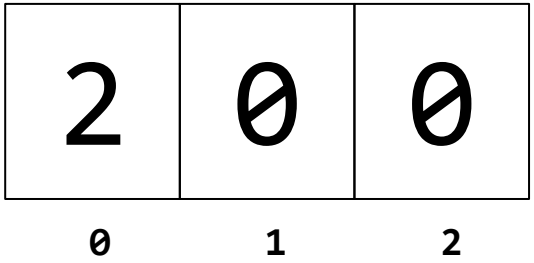
```
// A few examples
int[] evenNumbers = new int[3] ; // [0, 0, 0]
```



Setting Array Elements

Unlike Strings, in arrays you can set what different indexes are equal to!

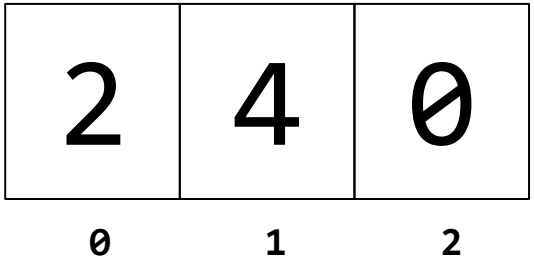
```
// A few examples
int[] evenNumbers = new int[3] ; // [0, 0, 0]

// Let's change the first element (index 0) to 2!
evenNumbers[0] = 2; // [2, 0, 0]
```



Setting Array Elements

Unlike Strings, in arrays you can set what different indexes are equal to!

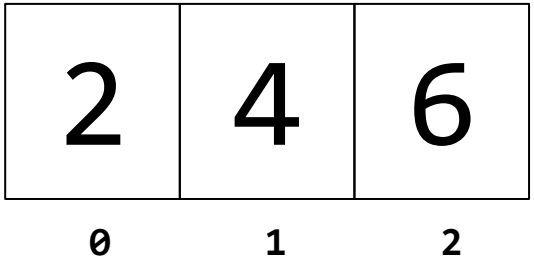
```
// A few examples
int[] evenNumbers = new int[3] ; // [0, 0, 0]

// Let's change the first element (index 0) to 2!
evenNumbers[0] = 2; // [2, 0, 0]

// Let's change the second element (index 1) to 4!
evenNumbers[1] = 4; // [2, 4, 0]
```



Setting Array Elements

Unlike Strings, in arrays you can set what different indexes are equal to!

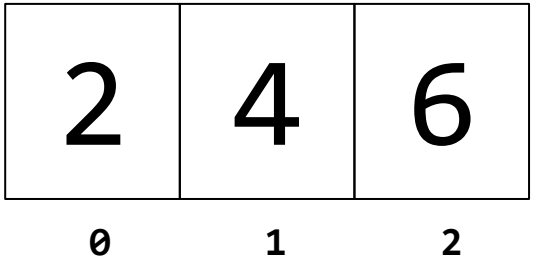
```
// A few examples
int[] evenNumbers = new int[3] ; // [0, 0, 0]

// Let's change the first element (index 0) to 2!
evenNumbers[0] = 2; // [2, 0, 0]

// Let's change the second element (index 1) to 4!
evenNumbers[1] = 4; // [2, 4, 0]

// Let's change the third element (index 2) to 6!
evenNumbers[2] = 6; // [2, 4, 6]
```



Setting Array Elements

This is one way to set the last index of an array!

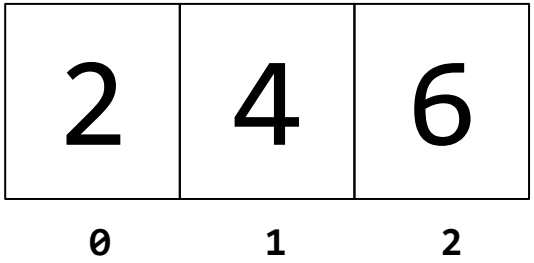
```
// A few examples
int[] evenNumbers = new int[3] ; // [0, 0, 0]

// Let's change the first element (index 0) to 2!
evenNumbers[0] = 2; // [2, 0, 0]

// Let's change the second element (index 1) to 4!
evenNumbers[1] = 4; // [2, 4, 0]

// Let's change the last element (index 2) to 6!
evenNumbers[evenNumbers.length - 1] = 6; // [2, 4, 6]
```



Setting Array Elements

```
name[index] = newValueAtIndex;
```

```
// A few examples
int[] evenNumbers = new int[3] ; // [0, 0, 0]

// Let's change the first element (index 0) to 2!
evenNumbers[0] = 2; // [2, 0, 0]

// Let's change the second element (index 1) to 4!
evenNumbers[1] = 4; // [2, 4, 0]

// Let's change the last element (index 2) to 6!
evenNumbers[evenNumbers.length - 1] = 6; // [2, 4, 6]
```



Why Do We Care About Arrays?

Why Do We Care About Arrays?

Storing Lots of Values

What if I wanted to find out the maximum amount I spent on coffee this week?

Storing Lots of Values

Let's use an array to store information about coffee spending!

Max Coffee Spending

What type of data are we storing?

Max Coffee Spending

What type of data are we storing?

`Double`!

Money is usually stored as a **`double`**.

Max Coffee Spending

What's the Pseudocode?

Max Coffee Spending

What's the Pseudocode?

Repeat for 7 days of the week

    Ask how much money was spent

    Store money spent in array

Repeat for length of array

    Compare current value to previous maximum spent

    if current value is greater then previous maximum

        Store current value as new max value

Let’s Code It!

Max Coffee Spending

What's the Pseudocode?

Repeat for 7 days of the week

    Ask how much money was spent

    Store money spent in array

Use Array method to sort the array smallest -> biggest

The last element is the most money spent!

Arrays

Using an array helped us store data and allowed us to use some helpful methods to make our lives easier.

Arrays as Parameters

What if we want to pass arrays and elements in arrays into a method? What happens then?

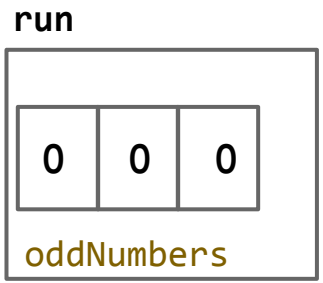
Arrays as Parameters

```
public void run(){
     int[] oddNumbers = new int[3]; // [0, 0, 0]
     makeArrayOdd(oddNumbers);
}

private void makeArrayOdd(int[] arrayToChange){
     arrayToChange[0] = 1;
     arrayToChange[1] = 3;
     arrayToChange[2] = 5;
}
```

What happens to this array?

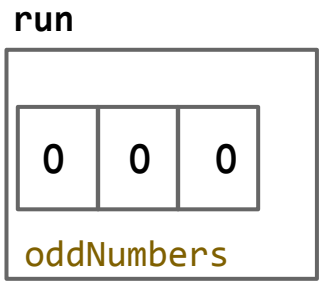
Arrays as Parameters

```
public void run(){
     int[] oddNumbers = new int[3]; // [0, 0, 0]
     makeArrayOdd(oddNumbers);
}
```



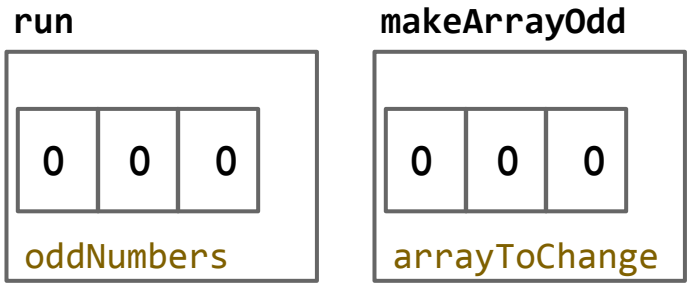
Arrays as Parameters

```
public void run(){
     int[] oddNumbers = new int[3]; // [0, 0, 0]
     makeArrayOdd(oddNumbers);
}
```



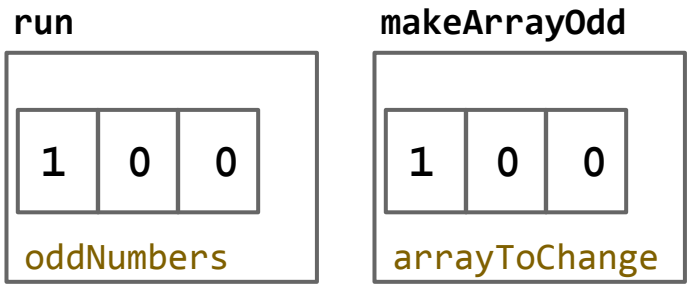
Arrays as Parameters

```
public void run(){
}
    private void makeArrayOdd(int[] arrayToChange){
        arrayToChange[0] = 1;
        arrayToChange[1] = 3;
        arrayToChange[2] = 5;
    }
```



Arrays as Parameters

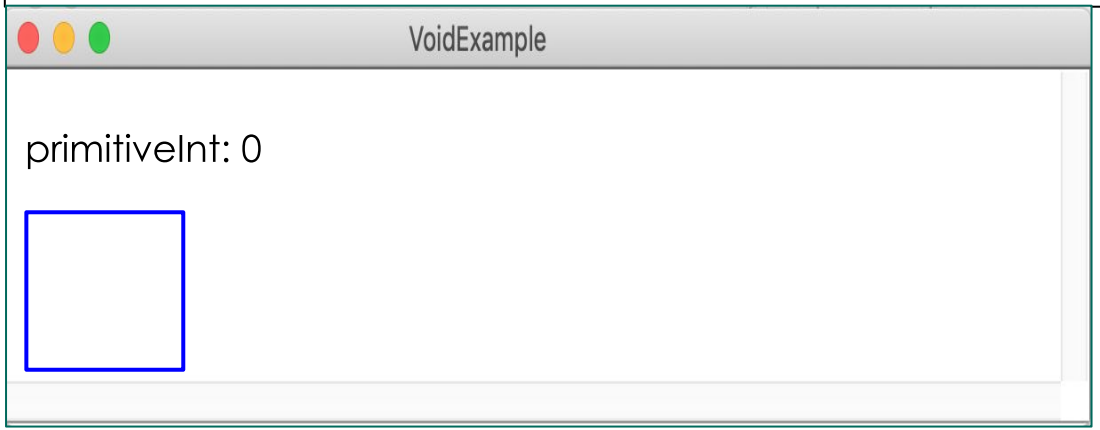
```
public void run(){
}
    private void makeArrayOdd(int[] arrayToChange){
        arrayToChange[0] = 1;
        arrayToChange[1] = 3;
        arrayToChange[2] = 5;
    }
```

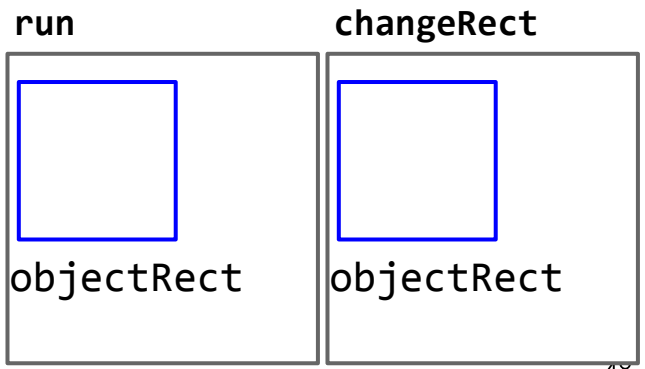


Wait, they both changed!
We've seen this before...

Passing an Object by **Reference**

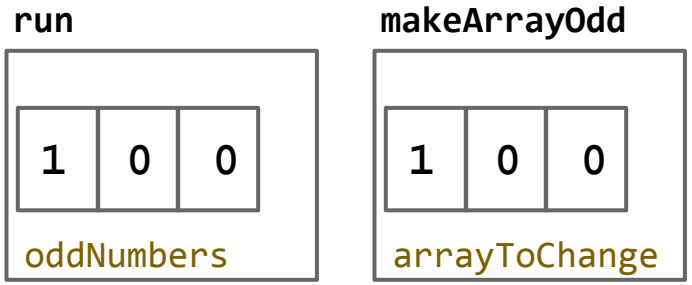
```
public void run(){
    add(objectRect, 100, 100);
    changeRect(objectRect);
}

private void changeRect(GRect objectRect){
    objectRect.setFilled(false);
    objectRect.setColor(Color.GREEN);
}
```





Arrays as Parameters

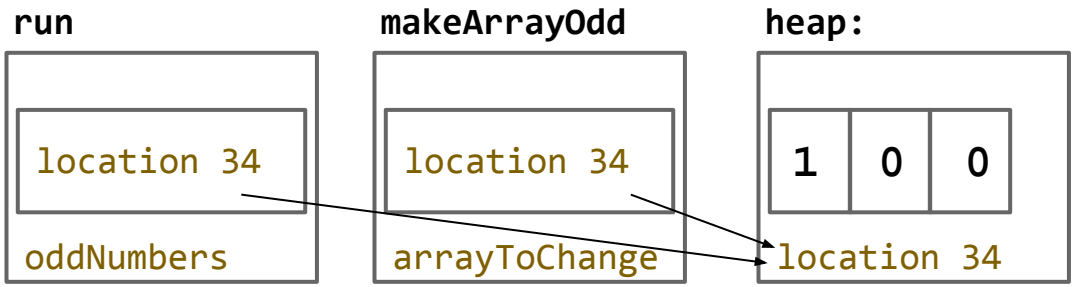
```
public void run(){
}
    private void makeArrayOdd(int[] arrayToChange){
        arrayToChange[0] = 1;
        arrayToChange[1] = 3;
        arrayToChange[2] = 5;
    }
```



It looks like arrays are **also passed by reference**. Arrays are also objects!

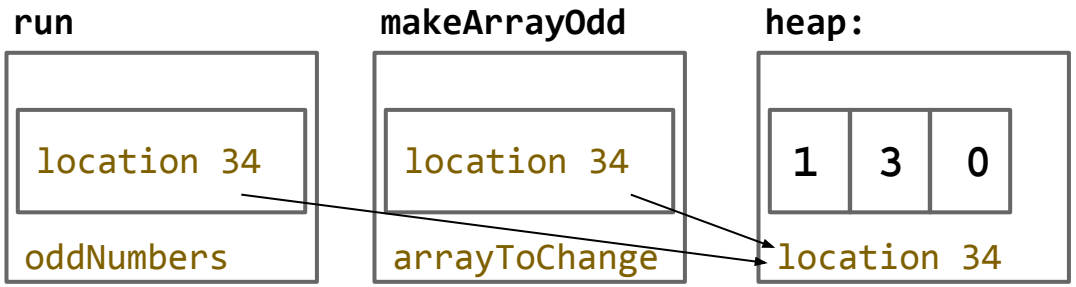
Arrays as Parameters

```
public void run(){
}
    private void makeArrayOdd(int[] arrayToChange){
        arrayToChange[0] = 1;
        arrayToChange[1] = 3;
        arrayToChange[2] = 5;
    }
```



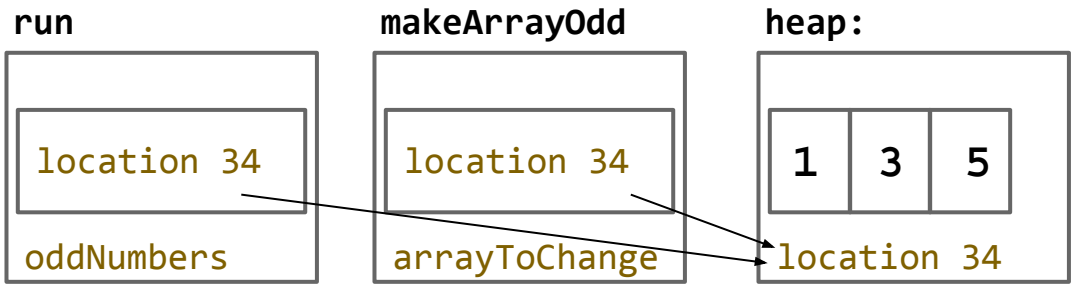
Arrays as Parameters

```
public void run(){
}
    private void makeArrayOdd(int[] arrayToChange){
        arrayToChange[0] = 1;
        arrayToChange[1] = 3;
        arrayToChange[2] = 5;
    }
```



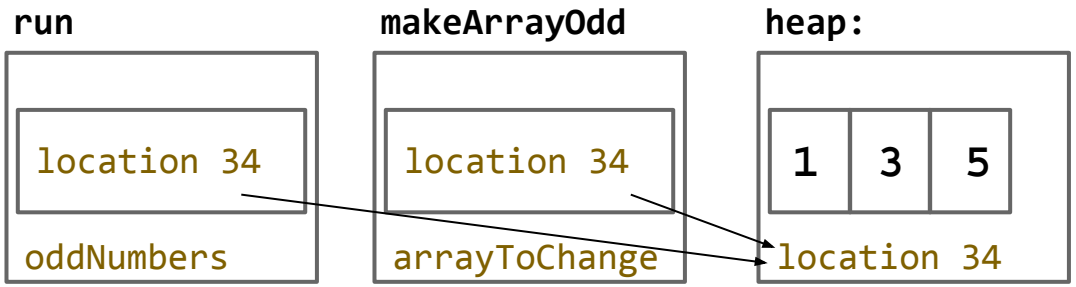
Arrays as Parameters

```
public void run(){
}
    private void makeArrayOdd(int[] arrayToChange){
        arrayToChange[0] = 1;
        arrayToChange[1] = 3;
        arrayToChange[2] = 5;
    }
```



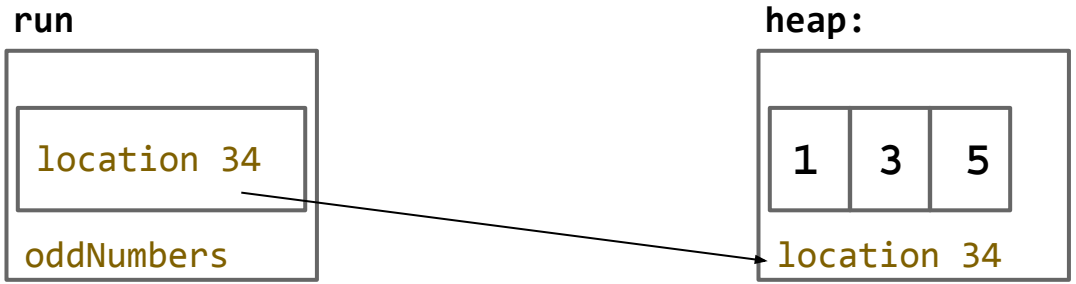
Arrays as Parameters

```
public void run(){
}
    private void makeArrayOdd(int[] arrayToChange){
        arrayToChange[0] = 1;
        arrayToChange[1] = 3;
        arrayToChange[2] = 5;
    }
```



Arrays as Parameters

```
public void run(){
     int[] oddNumbers = new int[3]; // [0, 0, 0]
     makeArrayOdd(oddNumbers); // [1, 3, 5]
}
```



Arrays are Passed by Reference

Because arrays are **passed by reference** this means they CAN be changed by passing them into a method.

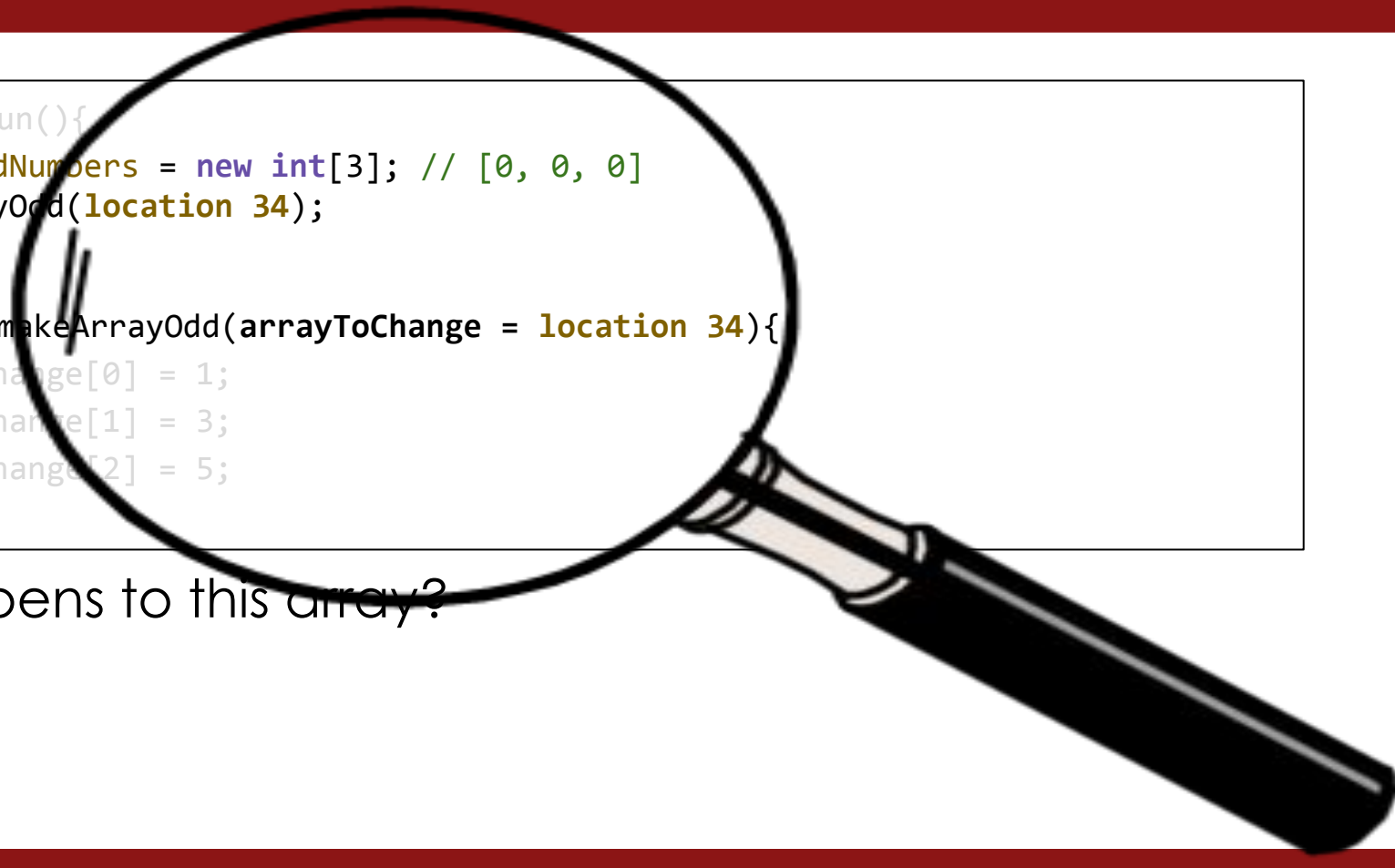
Arrays as Parameters

```
public void run(){
     int[] oddNumbers = new int[3]; // [0, 0, 0]
     makeArrayOdd(oddNumbers);
}

private void makeArrayOdd(int[] arrayToChange){
     arrayToChange[0] = 1;
     arrayToChange[1] = 3;
     arrayToChange[2] = 5;
}
```

A Closer Look at Parameters

```
public void run(){
     int[] oddNumbers = new int[3]; // [0, 0, 0]
     makeArrayOdd(location 34);
}

private void makeArrayOdd(arrayToChange = location 34){
     arrayToChange[0] = 1;
     arrayToChange[1] = 3;
     arrayToChange[2] = 5;
}
```

What happens to this array?



Because arrays are **passed by reference** this means they CAN be changed by passing them into a method.

What about array elements?

Array Values as Parameters

```
public void run(){
     int[] oddNumbers = new int[3]; // [0, 0, 0]
     makeArrayOdd(oddNumbers[0], oddNumbers[1], oddNumbers[2]);
}

private void makeArrayOdd(int a, int b, int c){
     a = 1;
     b = 3;
     c = 5;
}
```

What happens to the array now?

Array Values as Parameters

```
public void run(){
     int[] oddNumbers = new int[3]; // [0, 0, 0]
     makeArrayOdd(oddNumbers[0], oddNumbers[1], oddNumbers[2]);
}

private void makeArrayOdd(int a, int b, int c){
     a = 1;
     b = 3;
     c = 5;
}
```

This does **not** change our array! By passing in **primitive values from the array**, we simply created copies of the numbers in the array!

If our parameters are primitives, nothing is changing outside the method!

A Closer Look

```
public void run(){
     int[] oddNumbers = new int[3]; // [0, 0, 0]
     makeArrayOdd(0,          0,          0);
}

private void makeArrayOdd(a = 0, b = 0, c = 0){
     a = 1;
     b = 3;
     c = 5;
}
```

This does **not** change our array! By passing in **primi** **es from the array**, we simply created copies of the numbers in the a

If our parameters are primitives, nothing is changing outside the method!

Swapping Elements

How can we swap elements in an array?

Think: if we have two boxes, how can we swap what's in the boxes?

Swapping Elements

How can we swap elements in an array?

Think: if we have two boxes, how can we swap what's in the boxes?

Answer: We need a temporary box to store one of the elements while we're doing the swap!

Swapping Elements: Example 1

```
public void run(){
     int[] sortedArray = {1, 3, 2, 4}; // [1, 3, 2, 4]
     swapElements(sortedArray[1], sortedArray[2]);
}

private void swapElements(int val1, int val2){
     int temp = val1;
     val1 = val2;
     val2 = temp;
}
```

What happens to this array?

Swapping Elements: Example 2

```
public void run(){
     int[] sortedArray = {1, 3, 2, 4}; // [1, 3, 2, 4]
     swapElements(sortedArray, 1, 2);
}

private void swapElements(int[] arr, int pos1, int pos2){
     int temp = arr[pos1];
     arr[pos1] = arr[pos2];
     arr[pos2] = temp;
}
```

What happens to this array?

Reference vs. Value

In Example 1, we made copies of the primitive elements in the array and they were **passed by value**.

In Example 2, we passed in the array itself, and it was **passed by reference**.

When we pass in the array as a parameter, we can change it within a method!

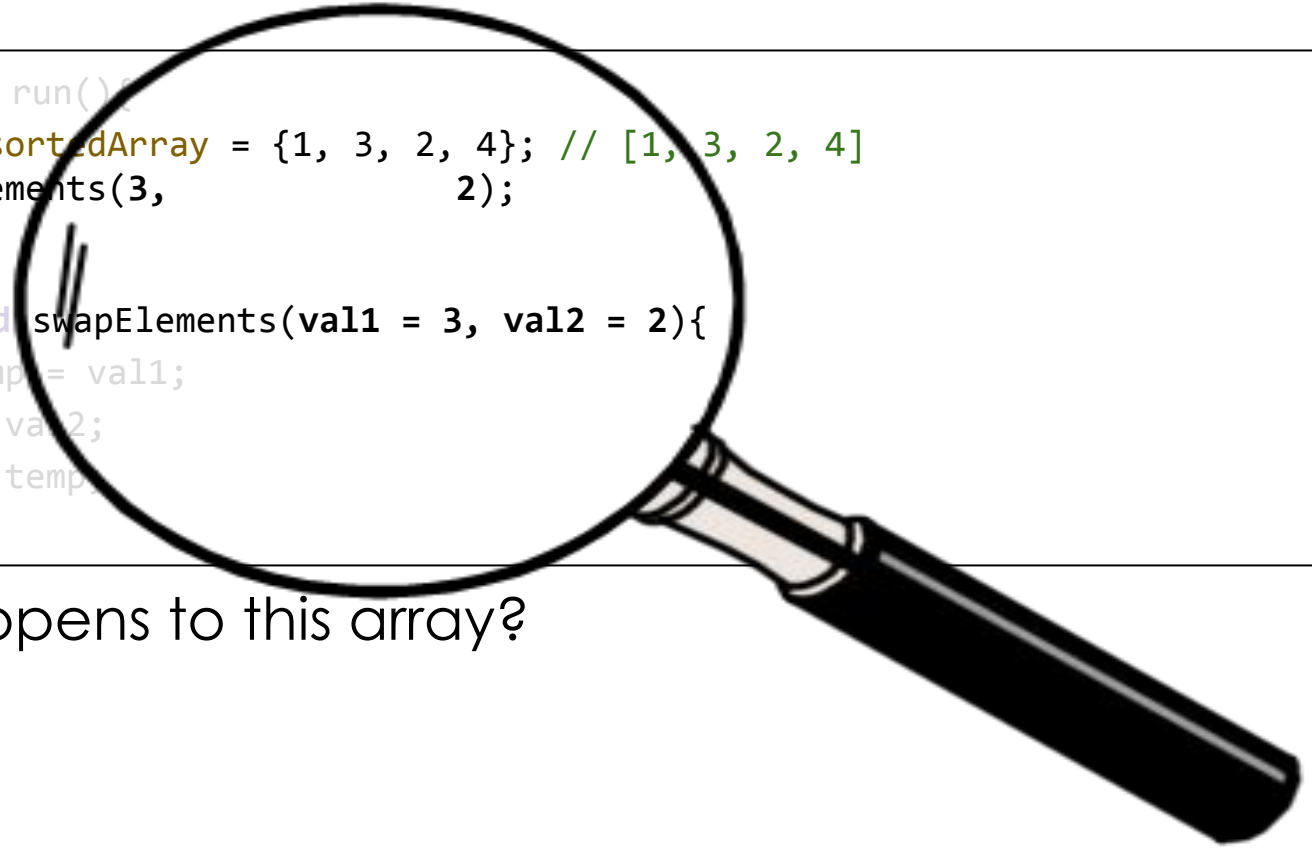
Swapping Elements: Example 1

```
public void run(){
     int[] sortedArray = {1, 3, 2, 4}; // [1, 3, 2, 4]
     swapElements(sortedArray[1], sortedArray[2]);
}

private void swapElements(int val1, int val2){
     int temp = val1;
     val1 = val2;
     val2 = temp;
}
```

What happens to this array?

Swapping Elements: Example 1

```
public void run(){
    int[] sortedArray = {1, 3, 2, 4}; // [1, 3, 2, 4]
    swapElements(3,                   2);
}

private void swapElements(val1 = 3, val2 = 2){
    int temp = val1;
    val1 = val2;
    val2 = temp;
}
```



What happens to this array?

Swapping Elements: Example 2

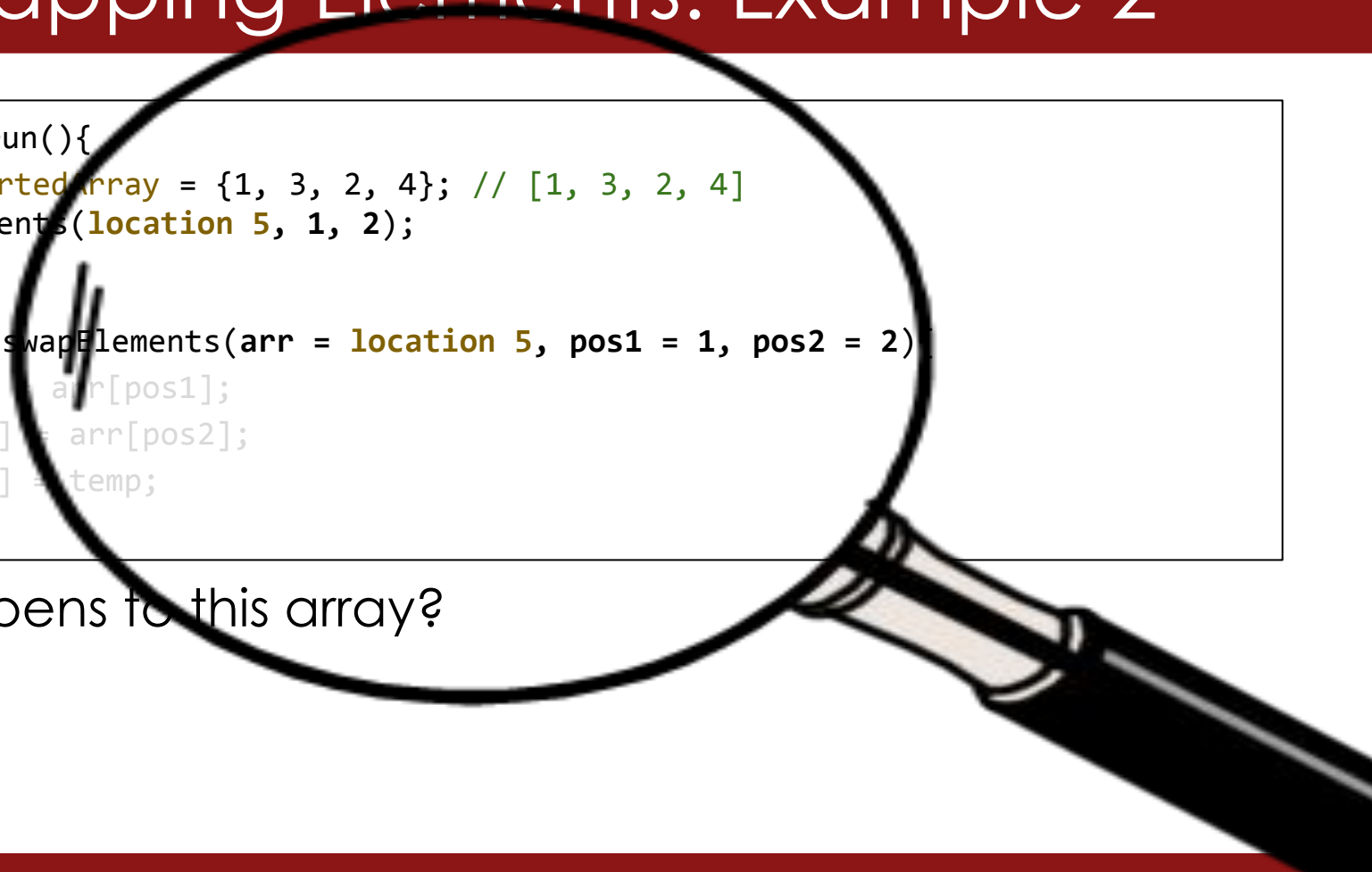
```
public void run(){
     int[] sortedArray = {1, 3, 2, 4}; // [1, 3, 2, 4]
     swapElements(sortedArray, 1, 2);
}

private void swapElements(int[] arr, int pos1, int pos2){
     int temp = arr[pos1];
     arr[pos1] = arr[pos2];
     arr[pos2] = temp;
}
```

What happens to this array?

Swapping Elements: Example 2

```
public void run(){
     int[] sortedArray = {1, 3, 2, 4}; // [1, 3, 2, 4]
     swapElements(location 5, 1, 2);
}

private void swapElements(arr = location 5, pos1 = 1, pos2 = 2){
     int temp = arr[pos1];
     arr[pos1] = arr[pos2];
     arr[pos2] = temp;
}
```

What happens to this array?



Reference vs. Value

In Example 1, we made copies of the primitive elements in the array and they were **passed by value**.

In Example 2, we passed in the array itself, and it was **passed by reference**.

When we pass in the array as a parameter, we can change it within a method!

Another Example

Let's say I want to store information about my 5 cats (that I wish I could have).

We'll store some data about them in an array and then retrieve that data from our array!

A Small Lie...

Here is my cat (she lives with my mother).

Her name is Chi.

Another Example

I love my cat, but she has been constantly waking me up at night!

It makes me pretty tired, so my doctor wants to know how many wake ups I've been receiving! Can we write a program to record and relay the number of times I've been woken up by my (beautiful) cat*?

*Who I love more than I love most things

Let’s Code It!

Quick Sidenote:

`ArrayLists` will be covered in Thursday's lecture, but you will need them for ParaKarel, so we're doing a quick intro now!

`ArrayLists` are a bit like fancy arrays (more on Thursday).

Arrays vs. ArrayLists

Array:

```
int[] myArray = new int[5];
OR
int[] myArray = {2, 3, 4, 5, 6, 7};

int arrayLen = myArray.length;

int first = myArray[0];

int last = myArray[arrayLen - 1];

myArray[last] = 1;
```

ArrayList:

```
ArrayList<Integer> myAL = new ArrayList<>();

int arrayListLen = myAL.size();

int first = myAL.get(0);

int last = myAL.get(arrayListLen - 1);

myAL.add(1);
```

Arrays vs. ArrayLists

ArrayList:

```
// How to create an ArrayList that stores Strings
ArrayList<String> myAL = new ArrayList<>();

// Getting the size of the ArrayList
int arrayListLen = myAL.size();

// Getting the first and last element in an ArrayList of Strings
String first = myAL.get(0);
String last = myAL.get(arrayListLen - 1);

// Adding an element to the ArrayList
myAL.add("defenestrate");
```

Plan for Today

- Review: Characters and Strings
- Data Structures
- Arrays
- Storing Coffee Prices in an Array
- Pass by Reference and Pass by Value
- Cat Wake Ups
- ArrayLists (quickly)