CS 152
Computer Programming
Fundamentals
Arrays

Brooke Chenoweth

University of New Mexico

Spring 2024

Why Arrays?

- Do you remember your math?

$$\overline{x} = \frac{\sum_{i=1}^{N} x_i}{N}$$

  What is this?
- Right. . . The arithmetic mean. . .
- So, if you have N variables, of the same type, but different values, you need N variable declarations in order to store those values.
- And in a loop, a way of accessing all those variables in order

Averaging Numbers in Java

```
double x1;
double x2;
double x3;
double x4;

// initialize values here...

double sum = x1 + x2 + x3 + x4;
double average = sum / 4;
```

Averaging a few values is easy, but what if we have more?

Averaging Numbers in Java

```
double x1;
double x2;
double x3;
double x4;
double x5;
double x6;
double x7;
double x8;
double x9;
double x10;

// initialize values here...

double sum = x1 + x2 + x3 + x4 + x5
           + x6 + x7 + x8 + x9 + x10;
double average = sum / 10;
```

This is getting ugly...

It would help if we could loop over the variables

What is an array?

- An array is basically an indexed variable, just like the formula on the earlier slide.
- Java arrays are 0-based arrays, so array indices always start at 0 (zero).
- The number of elements in the array can be accessed through by reading the `length` variable in the object.

Averaging Numbers in an Array

```
double[] x = new double[10];

// initialize values here...

double sum = 0;
for(int i = 0; i < x.length;  i++) {
   sum += x[i];
}

double average = sum / x.length;
```

Create array to hold 10 doubles

Ask array its length

Access array element at index i

Array declaration

- The standard form is:

  ```
  <type>[] <variableName>;
  ```

- You can declare arrays of *any* type you want
- The above doesn't tell you how many elements there should be in the array.
- We haven't initialized the array yet, so the variable refers to `null`

Array declaration

- The standard form is:

  ```
  <type>[] <variableName> = new <type>[<size>];
  ```

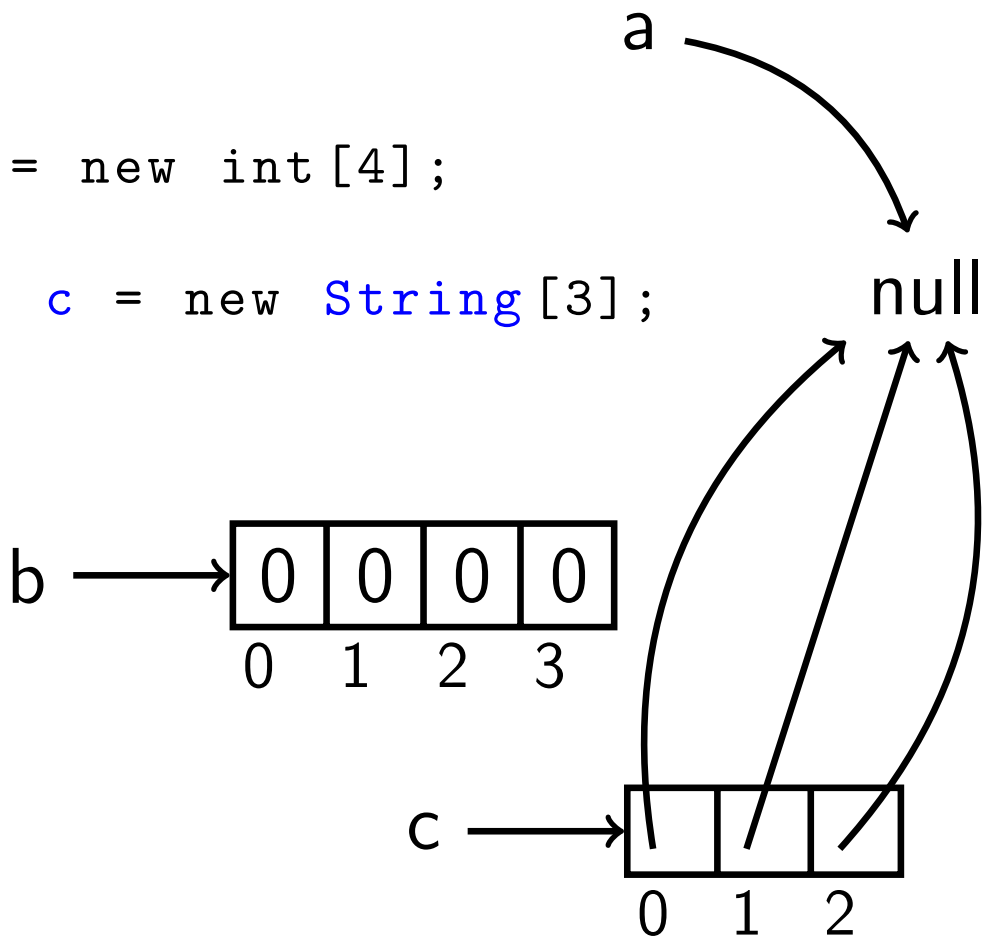
- The `size` tells us how many elements are in that array
- Arrays are initialized by default (on creation), this means:
  - Arrays of numbers contain all 0's
  - Arrays of reference types contains all `null`
- If you didn't create the array, you can still find out the length of it by using the `<variableName>.length` expression
  - This means, access the `length` instance variable in the array object referred to by the variable `<variableName>`.

Array declaration

```
int[] a;

int[] b = new int[4];

String[] c = new String[3];
```



Accessing array values

- Just like in math, we can read and assign to different indices of our variables.
- In the following example, I'm assuming that indexed variables in math are 1-based, and that appropriate Java arrays (of the right type) have already been created.

Math	**Java**
x_i	`x[i-1]`
$y = x_3$	`y = x[2];`
$x_5 = 15.67$	`x[4] = 15.67;`
$k = \frac{x_1 - x_2}{y_1 - y_2}$	`k = (x[0]-x[1])/(x[0]-x[1]);`

Array Initialization

- Arrays can be directly initialized to values by using what's called "Array Initializers":
  - `int[] arr = {5, 3, 8, 4};`
    Creates an int array of length 4 with above values.
  - `String[] sArr = {"Hello", "World"};`
    Creates a String array of length 2 with the above values
- Note! Java array structure is *immutable* once created. This means:
  - You can change values of the elements
  - You can not change the length of the array once it's been created.

Assigning arrays to each other

Since Java arrays are reference types we have to take some special considerations when trying to assign one to another:

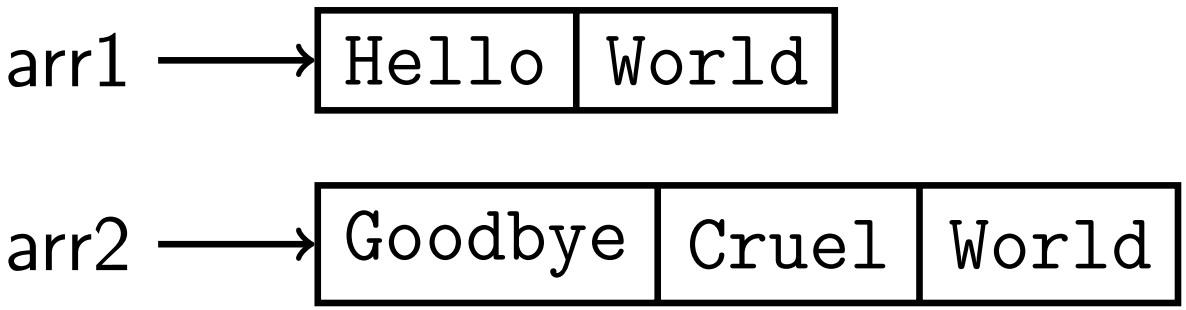
```
String[] arr1 = { "Hello", "World" };
String[] arr2 = { "Goodbye", "Cruel", "World" };
arr1 = arr2; // Array assignment
```

- In the above example both variables `arr1` and `arr2` now refer to the second array, and no variable refers to the original `arr1`.
- When an object (in this case an array) no longer has any variables referring to it, its memory is eventually recycled by means of the "garbage collector".

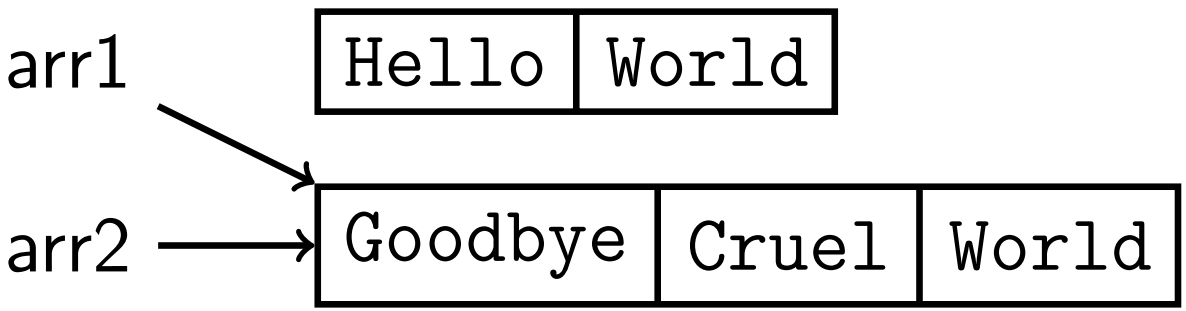
Assigning arrays to each other

```
String[] arr1 = { "Hello", "World" };
String[] arr2 = { "Goodbye", "Cruel", "World" };
arr1 = arr2; // Array assignment
```

Before assignment

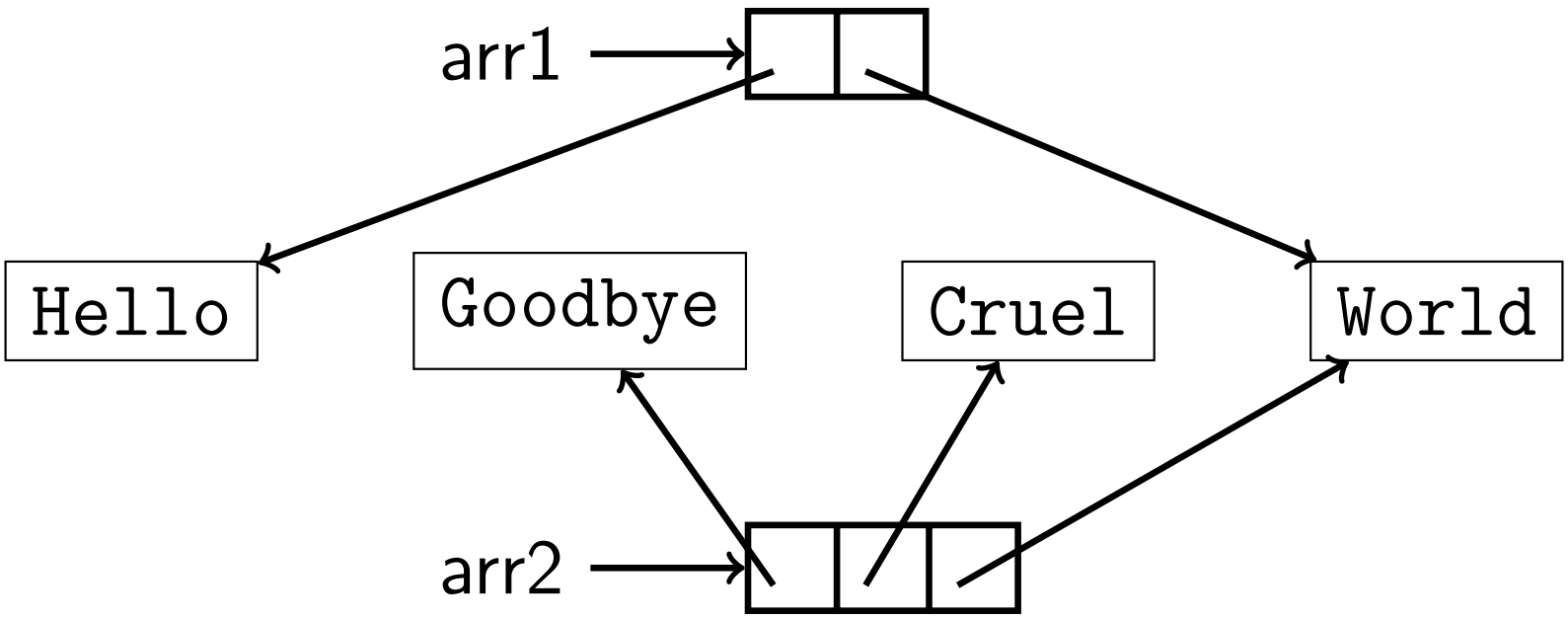


After assignment



Array of Objects

```
String[] arr1 = { "Hello", "World" };
String[] arr2 = { "Goodbye", "Cruel", "World" };
```



Array Example

```
public class ArrayExample1 {
  public static void main ( String[] args ) {
    int[] a = new int[15]; // Array with 15 elements
    int[] b = new int[15];

    // Give each element a value
    for ( int i = 0; i < a.length; i++ ) {
      a[i] = i;
      b[i] = a.length - i - 1;
    }
    // Print out every element in the array
    for ( int element: a ) {
      System.out.println ( element );
    }
    // Copy values from one array to the other
    for ( int i = 0; i < a.length; i++ ) {
      b[i] = a[i];
    }
  }
}
```

`String` vs `char[]`

A String is not the same as an array of `chars`.

	`String s;`	`char[] a;`
Find length	`s.length()`	`a.length`
Find i^{th} char	`s.charAt(i)`	`a[i]`
Convert to other	`s.toCharArray()`	`new String(a)`

Array of Arrays

- An array is a object type, so we could make an array to hold arrays.
- The following code makes an array of arrays of ints.

```
int[][] arr2d = new int[3][4];
```

This array has three elements, each consisting of an array of four ints.