CS 152
Computer Programming Fundamentals
Arrays

Brooke Chenoweth
University of New Mexico
Fall 2018
Why Arrays?

• Do you remember your math?

\[ \bar{x} = \frac{\sum_{i=1}^{N} x_i}{N} \]

What is this?
Why Arrays?

• Do you remember your math?

\[ \bar{x} = \frac{\sum_{i=1}^{N} x_i}{N} \]

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
What is an array?

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

<table>
<thead>
<tr>
<th>Math</th>
<th>Java</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_i$</td>
<td>$x[i-1]$</td>
</tr>
<tr>
<td>$y = x_3$</td>
<td>$y = x[2]$;</td>
</tr>
<tr>
<td>$x_5 = 15.67$</td>
<td>$x[4] = 15.67;$</td>
</tr>
<tr>
<td>$k = \frac{x_1-x_2}{y_1-y_2}$</td>
<td>$k = (x[0]-x[1])/(x[0]-x[1]);$</td>
</tr>
</tbody>
</table>
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 arrays are 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:

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

- In the above example both variables `arr1` and `arr2` now refer to the array `"Goodbye", "World"`, 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".
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.

- **Find length**
  - String: `s.length()`
  - char array: `a.length`

- **Find \( i^{th} \) char**
  - String: `s.charAt(i)`
  - char array: `a[i]`

- **Convert to other**
  - String to char array: `s.toCharArray()`
  - Char array to String: `new String(a)`