

ARRAYS

CS302 – Introduction to Programming
University of Wisconsin – Madison
Lecture 11

By Matthew Bernstein – matthewb@cs.wisc.edu

Introduction To Arrays

- Let's say we need to store a bunch of values (say thousands) of the same type. For example, we want to read a long sequence of numbers and then perform some analysis on these numbers (max, min, average, mode, etc.)
- How do we do this?
- Well, we could store each of the values in a variable:

```
double val_1;  
double val_2;  
....  
double val_1000;
```


- However, this is a horrible solution...it doesn't scale

Introduction to Arrays


- We use arrays to store many values
- Arrays are a fundamental object in java
- The following code creates an array called “values” with room to store 10 doubles:

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

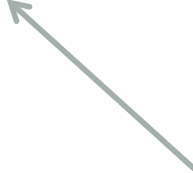
The data type
is an array that
stores doubles




The name of our
variable is “values”



The “**new**” operator
creates the array
by allocating the
space in memory




We specify that
we want the array to
hold 10 values



Alternative Ways of Creating Arrays

- We can also create an array with initial values:

```
double[] values = {32, 3, 355, 9.564, 2.49};
```



This creates an array
that stores 5 values.
These values are initialized
here.

- When we create an array this way, we don't need the “**new**” operator. The compiler automatically counts the number of values in the curly braces.

Accessing Values in Arrays

- Let's say we create the following array:

```
String[] names = {"Bob", "Lindsey", "Taylor"};
```

- We can access the i^{th} value as follows:

```
names[i];
```

- The following example gets the 0th value (the first value in the array):

```
String firstName = names[0]; // firstName = "Bob"
```

Assigning Values in Arrays

- Similarly we can assign the value at a specific index of an array:

```
String[] names = new String[3];  
names[0] = "Bob";  
names[1] = "Lindsey";  
names[2] = "Taylor";
```

Trying to Access Array at Invalid Index

- This is a common run-time error that will cause your program to crash with a **bounds error**.
- For example you create an array with 3 values:

```
String[] names = {"Bob", "Lindsey", "Taylor"};
```

- And you try to access the 4th value (index 3):

```
names[3]
```

Another Common Error

- Your declaration and initialization must be consistent.
- The following example is **WRONG**:

```
double[] data = new int[10];
```


Arrays – Their Lengths are Fixed

- Array's lengths are FIXED
- Let's say you create an array that can store 10 values and you would like to allow your array to grow to store more values...unfortunately you cannot do this
- In order to “grow” an array, you would have to create a new, larger array and copy all elements from your previous array to this new array (We will show how to do this later)

The “.length” Field

- We can find the length of an array by accessing the array’s “length” field
- Example:

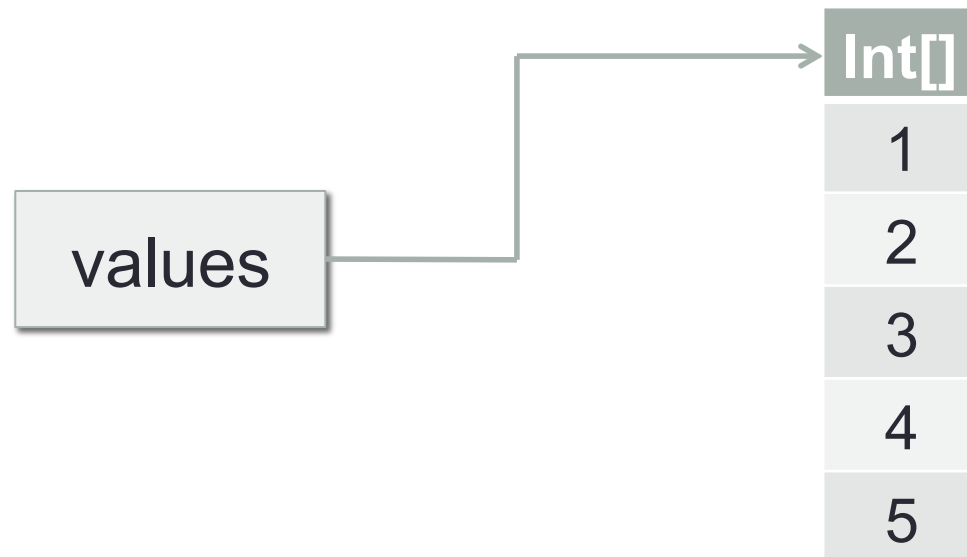
```
int[] arr = new int[10];
```

```
int x = arr.length;           // This is 10
```

Array Variables are References

- When you declare an array variable, the variable itself does not store any values. Rather, the variable points to the location in memory where the values are stored:

```
int[] values = {1, 2, 3, 4, 5};
```

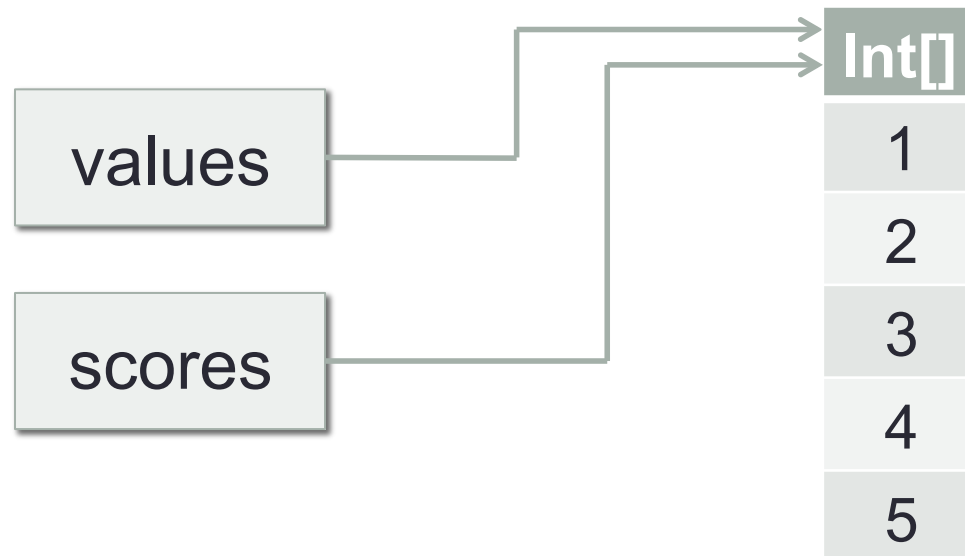


Copying Array Variables

- If we assign an array variable to another array variable, we are actually assigning the reference from the first variable to the second:

```
int [] values = {1, 2, 3, 4, 5}
```

```
int [] scores = values;
```



Example

- What will the following code print to the console?

```
int[] values = {1, 2, 3, 4, 5};
```

```
int[] scores = values;
```

```
values[3] = 6;
```

```
System.out.println( scores[3] );
```

Partially Filled Arrays

- Remember we cannot change the size of an array after it is initialized. However, what if we don't know how large to make our array when we initialize it?
- We can create a large array (larger than we think we'll need) and partially fill this array
- However, we'll need to keep track of how full the array is at any point. Thus, we keep a **companion variable** that tells us how full it is.

Partially Filled Arrays - Example

- Partially Filling an array:

```
Scanner in = new Scanner(System.in);

double[] values = new double[1000];
int currentSize = 0;
while ( in.hasNextDouble() )
{
    if (currentSize < values.length)
    {
        values[currentSize] = in.nextDouble();
        currentSize++;
    }
}
```

Processing a Partially Filled Array

- Instead of iterating over the entire array, we iterate only over the part of the array that is filled:

```
for (int i = 0; i < currentSize; i++)  
{  
    System.out.println( values[i] );  
}
```


The Enhanced “for” Loop

- As we’ve seen, it is common that we will need to process all of the contents of an array. To do this, we used the following:

```
for (int i = 0; i < values.length; i++)  
{  
    // Process values[i]  
}
```

- The Enhanced “for” loop provides an easier way of doing this:

```
for (double element : values)  
{  
    // Process each element of values  
}
```

The Enhanced “for” Loop

- How it works:

The type of elements stored in the array we are processing

The name of the variable we use to store each element of values

The array on which we are iterating over

```
for (type variable_name : array_variable)
```

A colon is needed here

Example

```
double[] values = {1.0, 2.0, 3.0, 4.0, 5.0};
```

```
for (double v : values)  
{  
    System.out.println(v);  
}
```

Programming Exercise

- Write a program that allows a user to enter a sequence of integers. The program should prompt the user for the number of integers the user will input and then will allow the user to input exactly that many values.
- The program should store these values in an array
- The program should then process the array and compute the mean, median, and mode.

Calculating Median

- We need to sort our array
- How do we do this?
- We need a sorting algorithm:
 - There are several algorithms for sorting a list which you will cover in data structure and algorithms courses.
 - We will cover the most basic algorithm:
Bubble Sort

Calculate Mode

- **Do this on your own as a programming exercise**
- **Idea**
 - Iterate through our array. Keep a count for the element with the highest number of appearances.

Cool CS Link of the Day

- Karl Sim's Evolved Creatures: Using a super-computer to simulate evolution
- http://www.youtube.com/watch?v=JBgG_VSP7f8

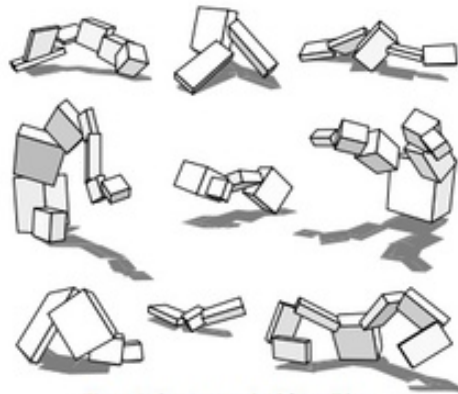


Figure 7: Creatures evolved for walking.

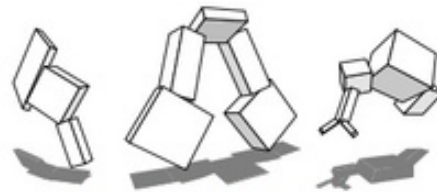


Figure 8: Creatures evolved for jumping.