# Computer Science 1

Module 5
Arrays

# Learning Goals

- You know what arrays are and how to declare them
- You can use an array to store and access values
- You can combine loops and arrays to iterate over values
- You are aware of the limitations of arrays
- You know what multi-dimensional arrays are and how to use them
- You know what *pass-by-value* means and what the consequences are
- You know the difference between single variables and arrays
- You are aware of common pitfalls when using arrays

# Limitations of Variables

Up to now, you need to know how many you need, BEFORE the program starts.

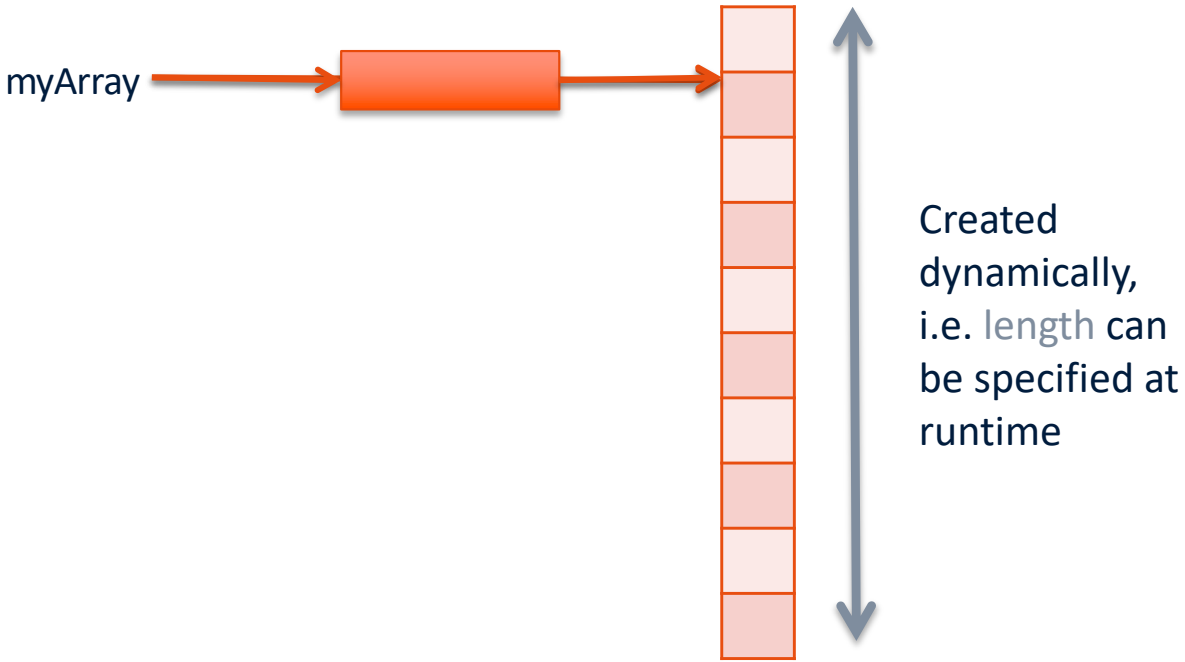
Consider the following task:

> "Write a program that reads in an unspecified number of values that ends with a letter and write those values out in sorted order."

# Arrays

Solution:

use variables that can store a number of values



# Array Definition

A collection of data items, all of the same type, packaged under a single name/identifier

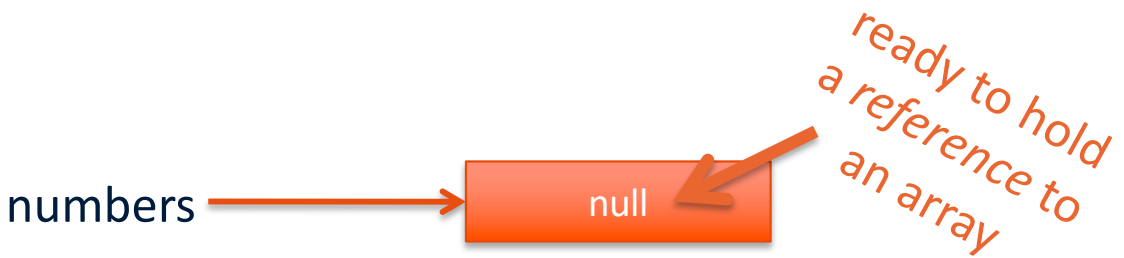
- Like String or Scanner, it is not a basic data type but an “object”
- Like Scanner, it needs to be declared AND created

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

# Array Declaration

```
int[] numbers;
String[] words;
double[] realnumbers;
```

```
int numbers[];
String words[];
double singlenumber, realnumbers[];
```

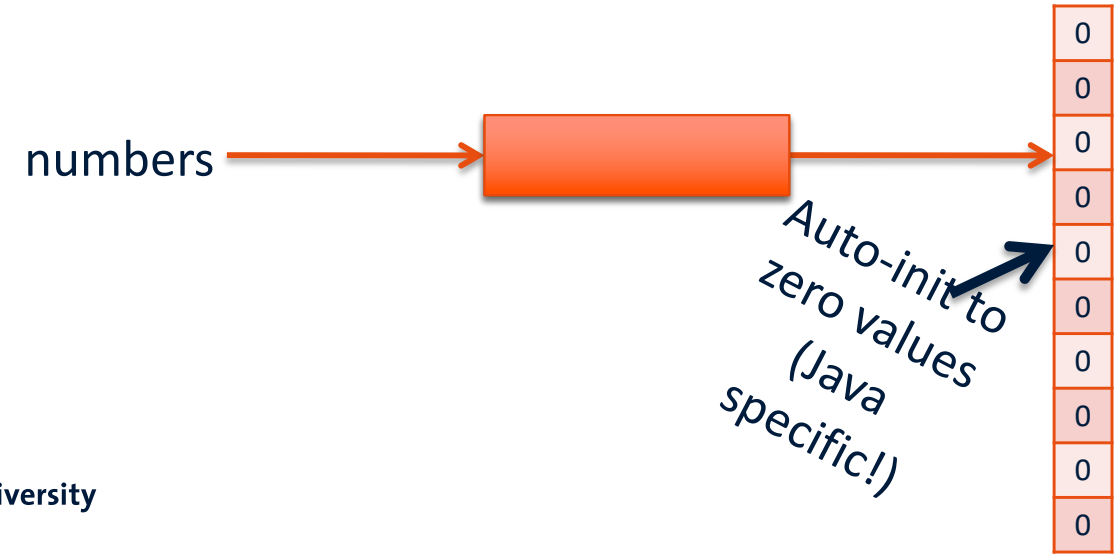


# Array Creation

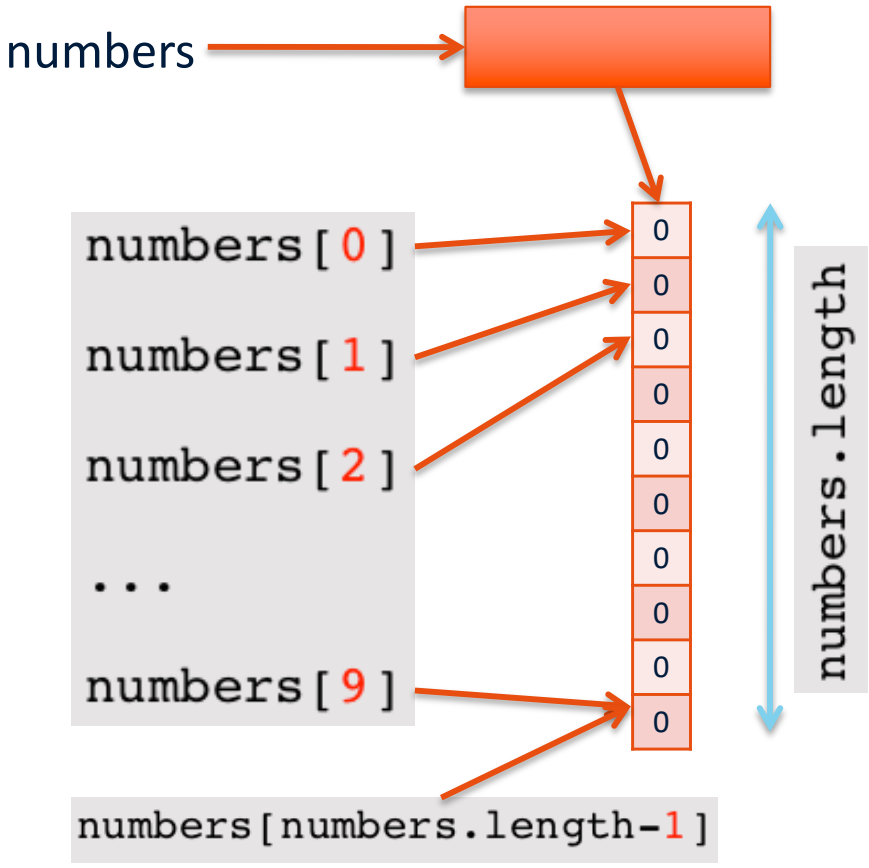
```
int length = ...;

numbers = new int[10];
realnumbers = new double[length*2-1];
```

new <type>[<length>]



# Access to Values




<arrayName>[<index>]

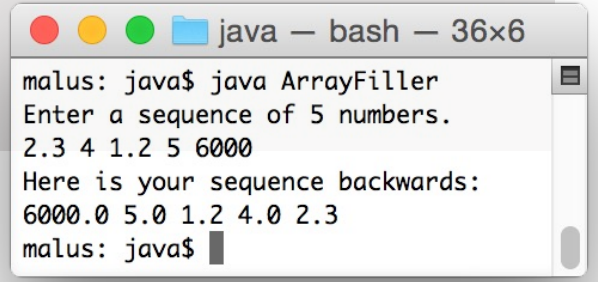
```
int i = ...;

numbers[i]
```

# An example

```
public static void main(String[] args) {
    Scanner input = new Scanner(System.in);
    //Make an array
    final int LENGTH = 5;
    double[] r = new double[LENGTH];
    // Ask for input
    System.out.println("Enter a sequence of " + LENGTH + " numbers.");
    // Read in all numbers
    for (int i=0; i<LENGTH; i++) {
        r[i] = input.nextDouble();
    }
    // Print out the numbers backwards
    System.out.println("Here is your sequence backwards:");
    for (int i=LENGTH-1; i>=0; i--) {
        System.out.print(r[i] + " ");
    }
    System.out.println();
}
```

Doesn't need to be a CONSTANT!

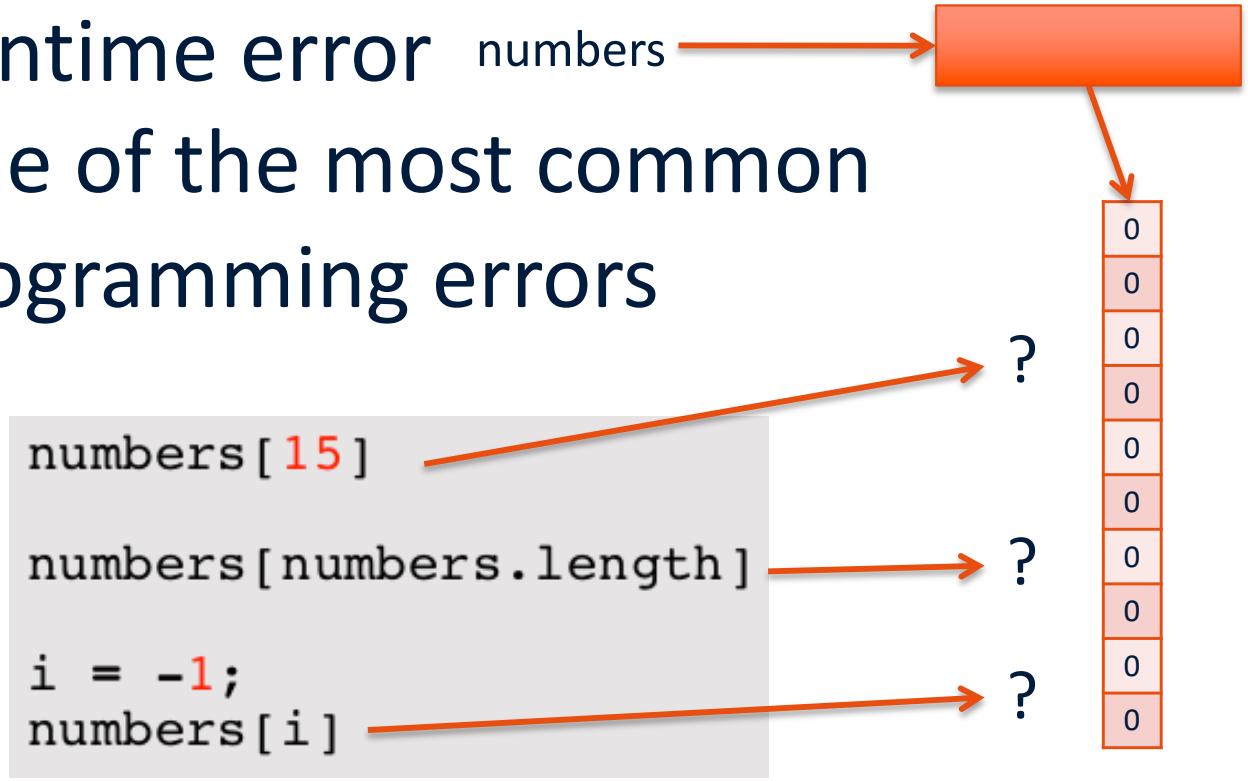


# Exercise together: SimpleArray

- Create an array of 10 elements
- Populate it such that element i = i
- Print out the array, iterating over its elements with a *for* loop
- Print out the array again, this time using the static method `Arrays.toString()`

# ArrayOutOfBoundsException

- Runtime error
- One of the most common programming errors

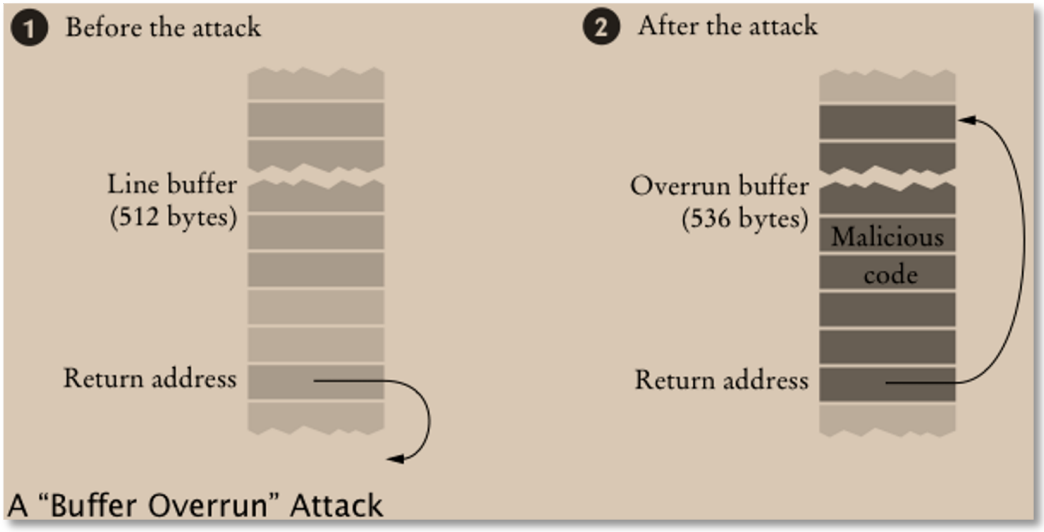


# A little bit of history

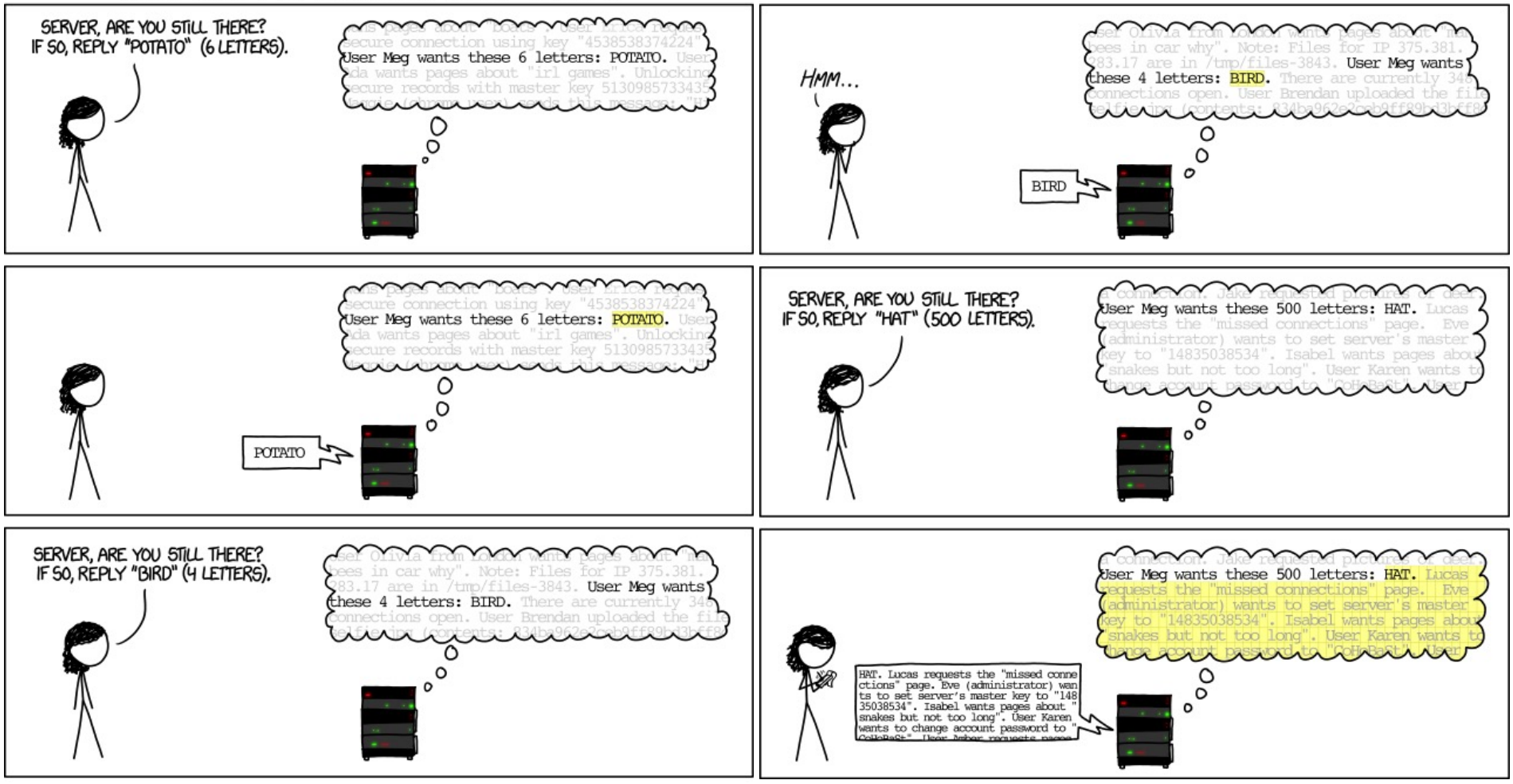
Java catches you when you read/write out of array bounds

This wasn't always the case with earlier programming languages

Early internet worm



# HOW THE HEARTBLEED BUG WORKS:



# Instantiating Arrays

Like Strings, arrays can be instantiated

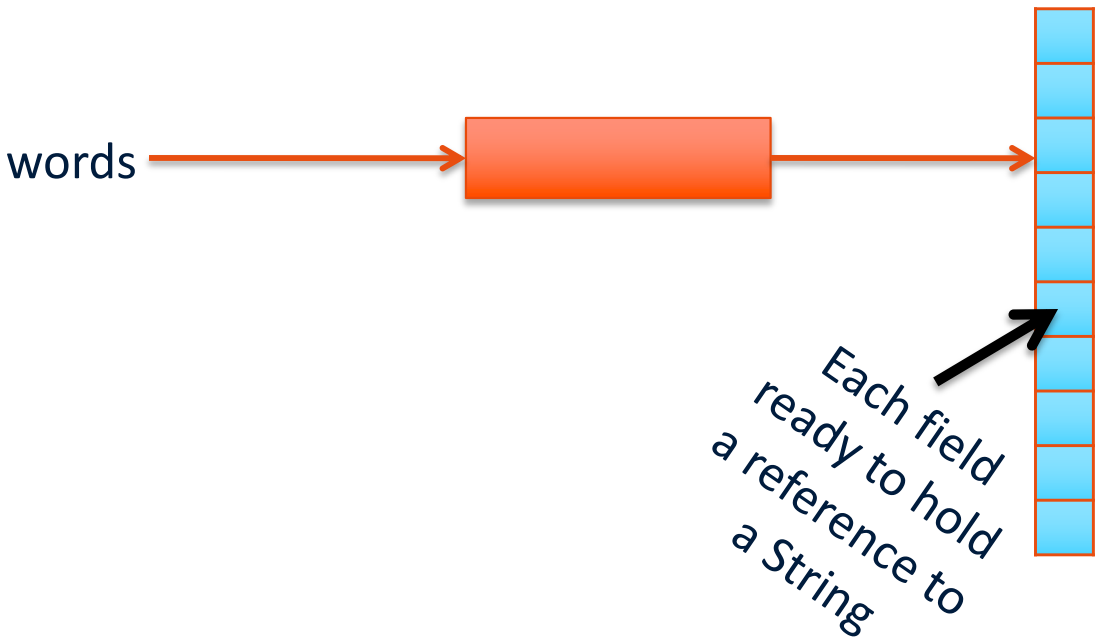
[Very useful for testing!!]

```
double[] ar = {1.0, 2.0, 3.0, 4.0};
String[] words = {"The","quick","brown","fox","jumps"};
```

Only possible for basic type arrays and String arrays (and arrays of arrays of those)

# Arrays of Objects

```
words = new String[10];
```



# main(String[] args)

```
public static void main(String[] args) {
    ...
}
```

Command line arguments passed into the java program

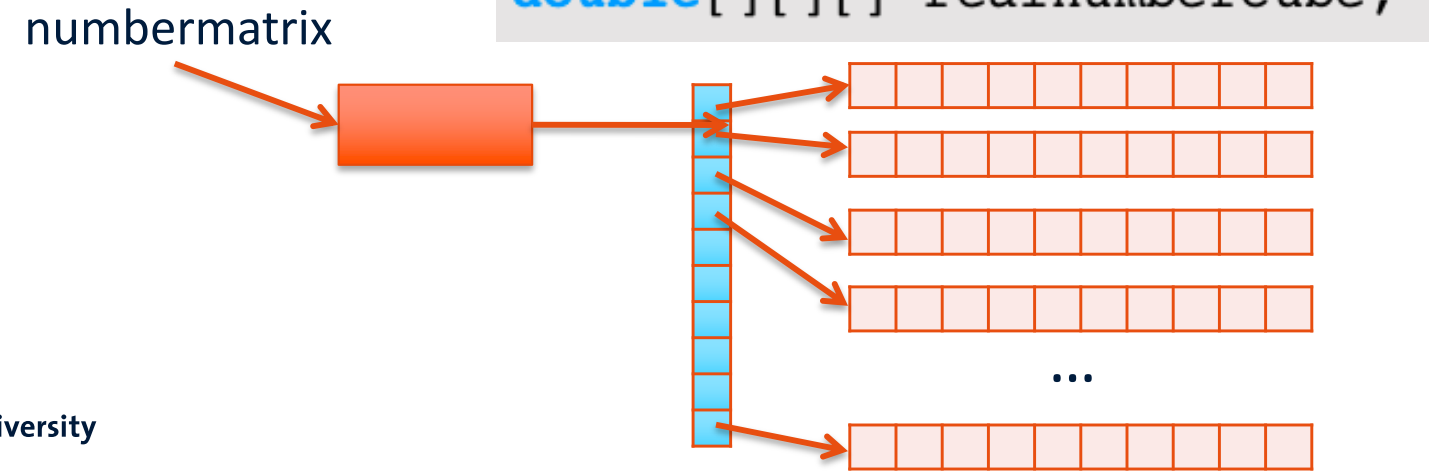
```
wopr: java$ java MyGame -players 2 -difficulty high
```

```
String[] args = {"-players","2","-difficulty","high"};
```

# Arrays of Arrays

Arrays are objects!
Arrays can hold objects!
= multi-dimensional arrays

```
int[][] numbermatrix;
double[][][] realnumbercube;
```



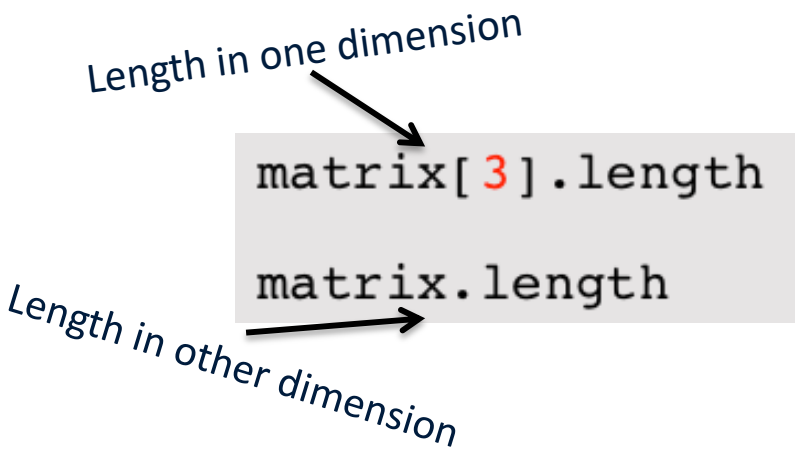
# Multidimensional Arrays

```
matrix[4][5]

matrix[i][j]

cube[3][4][7]

cube[i][j][k]
```

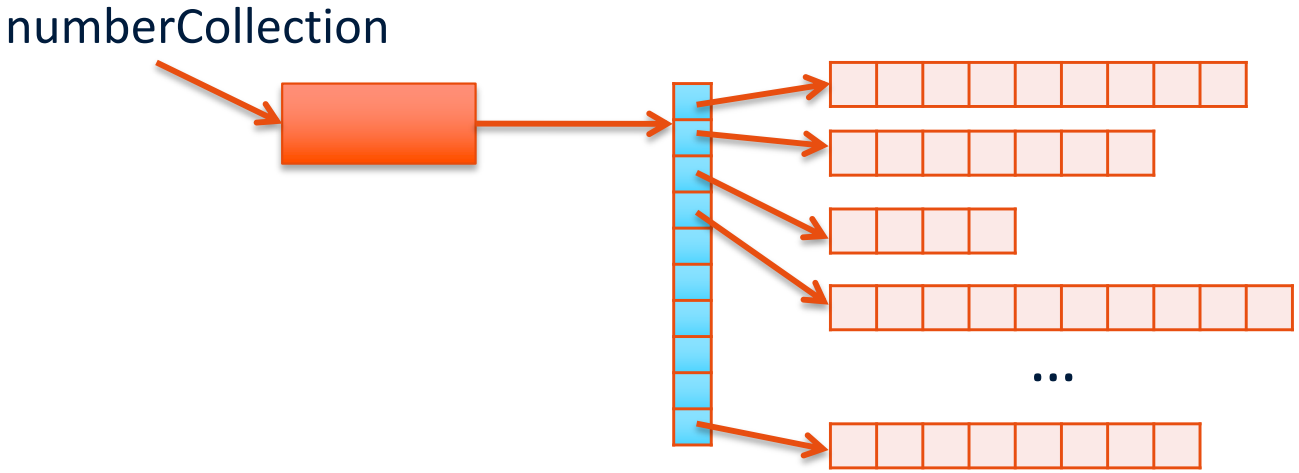



```
for (int i=0; i < matrix.length; i++)
    for (int j=0; j < matrix[i].length; j++)
        ... matrix[i][j] ...
```

# Making Multidimensional Arrays


```
double[][] numberCollection;
```

# Multi-dimensional Array of Different Lengths




```
for (int i=0; i < matrix.length; i++)
    for (int j=0; j < matrix[i].length; j++)
        ... matrix[i][j] ...
```

# Exercise together: MultidimensionalArray

- Make a multidimensional array initialized with the following values:

```
{{1,2,3,4,5}, {10,20}, {100,200,300,400}}
```

- Print it with a nested for loop
- Print it using the `Arrays.toString()` method

# Shortcut for normal people!!


When multi-dimensional arrays are rectangular:

When it represents a board, matrix, image, …

All dimension-lengths can be given in one go:

```
int board = new int[3][3];
double matrix = new double[rows][cols];
```

⇒ 99.9% or more of all arrays will be initialized this way

# Array Length

Once an array is created, its length can not be changed.

- What if the number of elements in it is unknown at creation?

*E.g. "read in an unknown number of integers, ending with a letter, then sort the numbers"*

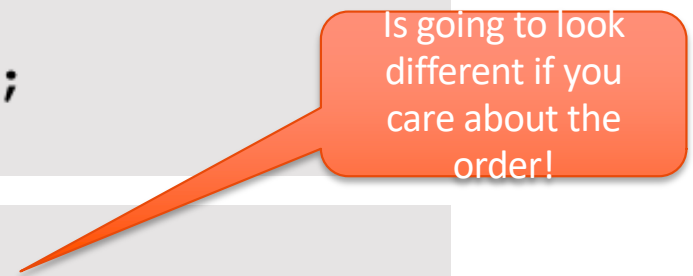
# Array Length Solutions

1. Create an array larger than the largest expected/reasonable number of entries

```
//init
int numbers[] = new int[1000];
int length = 0;
```

```
//Adding an element
numbers[length] = ...;
length++;
```

```
//Removing an element
numbers[index] = numbers[length-1];
length--;
```



# Array Length Solutions (cont.)

## 2. Replace array with array of correct length

```
//init
int[] numbers = new int[0];
```

```
//Adding an element
int[] newnumbers = new int[numbers.length+1];
for (int i=0; i<numbers.length; i++)
    newnumbers[i] = numbers[i];
numbers = newnumbers;
numbers[numbers.length-1] = ...;
```

```
//Removing an element
int[] newnumbers = new int[numbers.length-1];
for (int i=0; i<numbers.length-1; i++)
    newnumbers[i] = numbers[i];
if (index < numbers.length-1)
    newnumbers[index]=numbers[numbers.length-1];
numbers = newnumbers;
```

# Array Length Solutions (cont.)

## 3. Replace with larger array only when necessary

```
//init
int[] numbers = new int[2];
int length = 0;
```

```
//Adding an element
if (length < numbers.length) {
    numbers[length] = ...;
    length++;
} else {
    //length == numbers.length
    int[] newnumbers = new int[numbers.length*2];
    for (int i=0; i<numbers.length; i++)
        newnumbers[i] = numbers[i];
    numbers = newnumbers;
    numbers[length] = ...;
    length++;
}
```

```
//Removing an element
numbers[index] = numbers[length-1];
length--;
```

# Array Length Solutions (cont.)

4. Use an ArrayList!

More on this topic next week (and in project 1.1)

# Exercise together: RemovingElement

- Make an array with values: `{1,2,3,4,5,6,7,8,0,0}`
- Declare a variable toDelete indicating which element to delete
- Declare a variable length that will act as the valid length of the array
- Delete the requested element in the array

# Exercise together: DeleteFromArray

- Make an array with values: `{1,2,3,4,5,6,7,8,0,0}`
- Declare a variable toDelete indicating which element to delete

- Produce a new array with all elements except the deleted one

# More Shortcuts

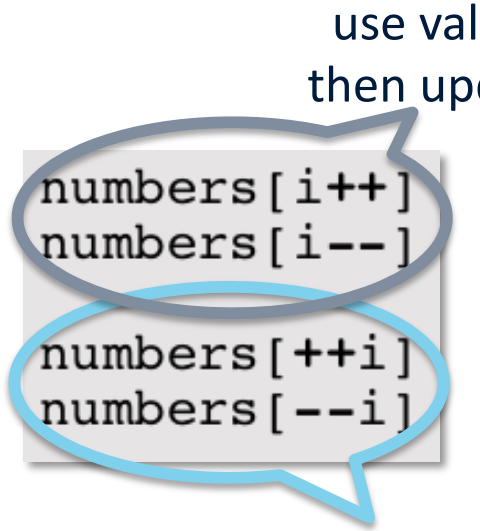
## Using an index + updating

```
public static void main(String[] args) {

    Scanner prompt = new Scanner(System.in);
    System.out.println("Type in a series of numbers,");
    System.out.println("end with a letter.");

    int[] numbers = new int[1000];
    int count = 0;
    while (prompt.hasNextInt()) {
        numbers[count++] = prompt.nextInt();
    }

    while (count > 0) {
        System.out.print(numbers[--count] + " ");
    }
    System.out.println();
}
```




```
Type in a series of numbers,
end with a letter.
12 23 34 45 56 67 78 89 90 q
90 89 78 67 56 45 34 23 12
```

# Even More Shortcuts (2)

## Copying an array

```
for (int i=0; i<numbers.length; i++)
    newnumbers[i] = numbers[i];
```

```
System.arraycopy(<fromArray>,<startCopyAt>,<toArray>,<startPasteAt>,<count>);
```

```
System.arraycopy(numbers,0,newnumbers,0,numbers.length);
```

# Exercise together: MoreShortcuts

- Make three arrays:
  - a: Initialize it with {1,2,3,4,5}
  - b: int array of size 5
  - c: int array of size 5

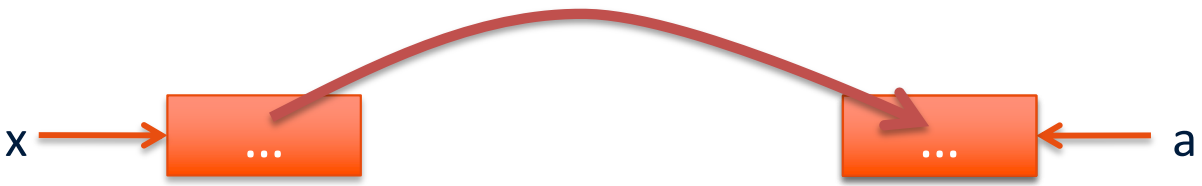
- Copy a into b with arraycopy
- Copy a into c with an assignment
- Print all three arrays and make sense of the differences
- Modify the contents of a[0] and print again

# Arrays as parameters

Java is “pass-by-value”!?!

```
public static int doSomething(int a) {
    ...
}
```

```
int x = ...;
doSomething(x);
```

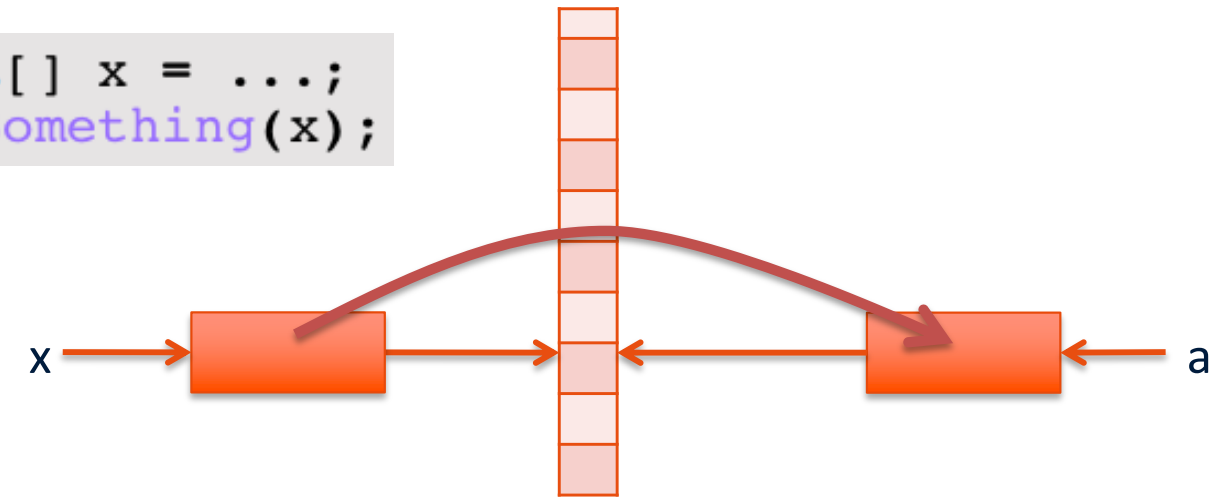


# Arrays as parameters (2)

Array references can be passed as parameters

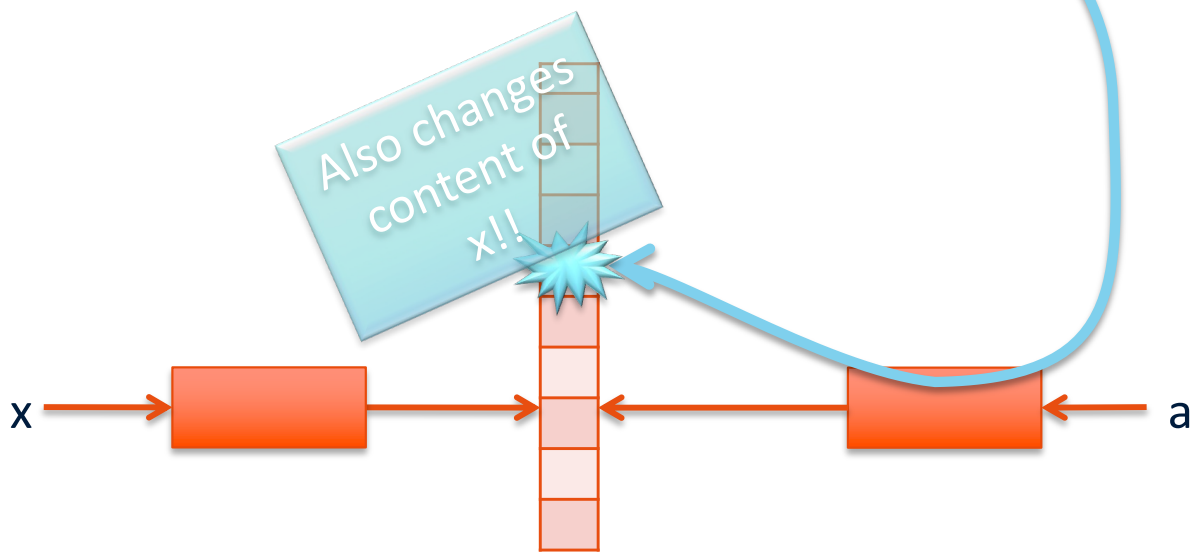
```
public static int doSomething(int[] a) {
    ...
}
```

```
int[] x = ...;
doSomething(x);
```



# Side effects!

```
public static int doSomething(int[] a) {
    ...
    a[i] = ...;
    ...
}
```



# Exercise together: ArrayPassByValue

- Make an array initialized with `{4, 8, 15, 16, 23, 42}`
- Create a method `public static void changeElement (int elementArray)` that assigns 10 to `elementArray`
- Create a method `public static void changeElement (int[] array)` that assigns 10 to `array[0]`

- Invoke both methods and print the resulting array at every stage

# Simple Array Algorithms

## Finding the minimum/maximum

Initializing *min* with the value in array[0] we skip one step in the for loop

```
public static int arrayMin(int[] array) {
    int min = array[0];
    for (int i=1; i<array.length; i++)
        min = Math.min(min,array[i]);
    return min;
}
```

# Simple Array Algorithms (2)

## Conditional counting

```
public static int positiveCount(int[] array) {
    int count = 0;
    for (int i=0; i<array.length; i++)
        if (array[i] >= 0)
            count++;
    return count;
}
```

# Simple Array Algorithms (3)

## Conditional selection

```
public static int[] positiveCollection(int[] array) {
    int[] coll = new int[array.length];
    int amount = 0;
    for (int i=0; i<array.length; i++)
        if (array[i] >= 0) {
            coll[amount] = array[i];
            amount++;
        }
    int[] result = new int[amount];
    System.arraycopy(coll,0,result,0,amount);
    return result;
}
```

# Simple Array Algorithms (4)

## Finding a value

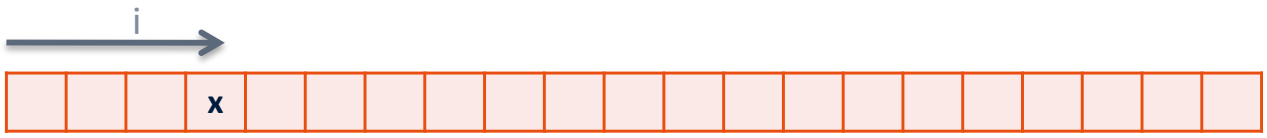
```
public static boolean contains(int x, int[] array) {
    boolean found = false;
    for (int i=0; i<array.length; i++) {
        if (array[i]==x)
            found = true;
    }
    return found;
}
```


n steps

# Simple Array Algorithms (5)

## Finding a value with better performance

```
public static boolean contains(int x, int[] array) {
    boolean found = false;
    int i = 0;
    while (!found && i<array.length) {
        if (array[i]==x)
            found = true;
        i++;
    }
    return found;
}
```




on average n/2 steps

# Simple Array Algorithms (6)

## Finding a value in a sorted array

```
public static boolean contains(int x, int[] array) {
    boolean found = false;
    int start = 0;
    int end = array.length-1;
    while (!found && start<=end) {
        int middle = (start+end)/2;
        if (array[middle]<x)
            start = middle+1;
        else if (x<array[middle])
            end = middle-1;
        else
            found = true;
    }
    return found;
}
```



Binary search, worst case $O(\log_2(n))$ steps

# Sorting an Array

(Almost) a research topic on its own …

- Lots of algorithms to sort an array (you cover this in detail in DSA course)
- We might see an example in some future homework

# Summary

Arrays

- Declaration and Creation
- Referencing values
- Array lengths
- Array references as parameters
- Some (simple) algorithms

# Coming up next!

Book: Check Canvas
Homework 5: 19 tasks available
Quiz 5
Practical session this Friday!
You can already start thinking about assignment 6
Q&A on Tuesday

Practice, practice ... practice!

- BAD NEWS: Complexity is increasing rapidly!!
- GOOD NEWS: We are reaching exam-level complexity!