

E2 Review

Topics

Everyone		CS 113	
Arrays	S&W 1.4	2d Arrays	pages 106-110
		Reading from Files	FileReader in HWs (not SW 1.5)
Methods	S&W 2.1	Recursion	S&W 2.3

Arrays

declaration	<pre>double[] doubArr; boolean[] boolArr;</pre>	create a name but not space
create	<pre>doubArr=new doubArr[5];</pre>	Actually make the array
	<pre>int siz = 5; boolArr = new boolArr[siz];</pre>	The size of an array can be set at run time
Initialize	<pre>for (int i=0; i<doubArr.length; i++) { doubArr[i]=i*i; }</pre>	Arrays usually have a "default" value (but I never remember). Use for loops.
Use	<pre>double sum=0.0; for (int i=(doubArra.length-1); i>=0; i++) { sum += doubArray[i]; }</pre>	For loop always use XXX.length

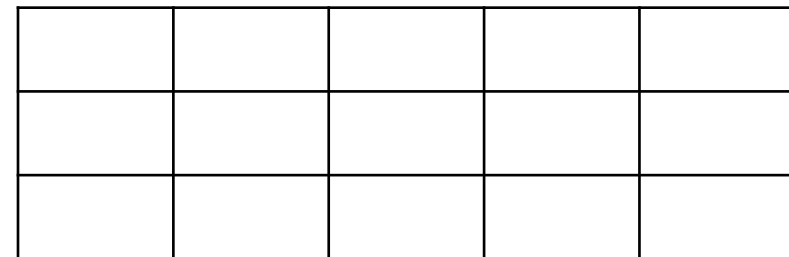
Reverse -- in place

- Write a complete program that reverses order of the command line arguments

2-D arrays

- `int[][] matr = new int[3][5];`

- Simple picture

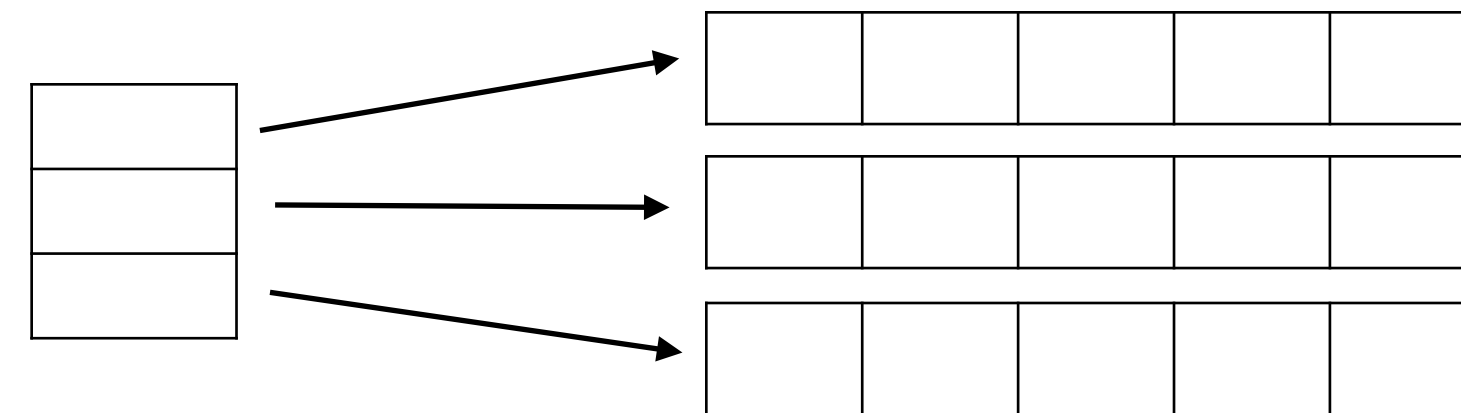


- Better Picture

- do not talk of rows and columns
- talk about first dimension, second dimension

- Size

- `matr.length == 3`
- `matr[0].length == 5` (note in better picture `matr[0].length ?? matr[1].length`)



Sum the lower diagonal of 2-d Array

Methods

Signature

public static double squareSum(double[] dArr)

- Parts of definition

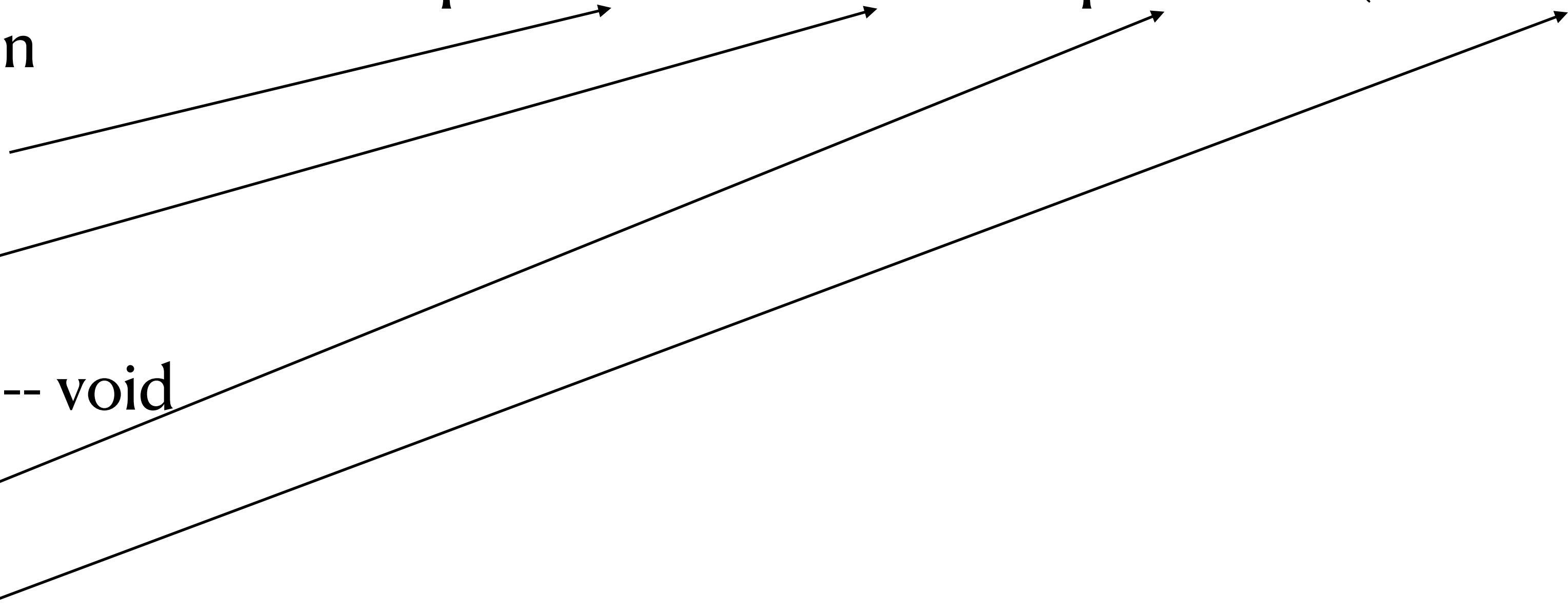
- "public static"

- return type

- special case -- void

- name

- parameters



Methods

Use

- `System.out.println("aa");` // void return
- `Math.pow(2,3);` // double return
- `Integer.parseInt("5");` // int return

- If non-void return

- use the return or ignore it?

- set to variable

```
double d = Math.pow(3,2);
```

- use as param to another method

```
System.out.println(Math.pow(3,2));
```

Methods

- Scope
 - All of the variables in a method are independent of the vars from where it was called
 - None of the vars from where it was called are available in method
- Passing Arrays
 - remember that arrays are pointers so when you pass an array you do not pass

--	--	--	--

you pass 

Recursion

- Method calls itself
 - with a slightly simpler problem
- 2 parts of a recursive method
 - Base Case
 - be sure to "return" from base case.
 - Action

Longest Increasing Subsequence

Using Recursion

```
public static void main(String[] args) {  
    if (args.length == 0) {  
        System.out.println("Give me a positive integer");  
        return;  
    }  
    int siz = Integer.parseInt(args[0]);  
    if (siz < 1) {  
        System.out.println("Give me a positive integer please");  
        return;  
    }  
    double[] dd = new double[siz];  
    for (int i = 0; i < dd.length; i++) {  
        dd[i] = Math.random();  
        System.out.println(dd[i]);  
    }  
    System.out.println(longIncreas(dd, 1, 1, 1));  
}
```