E2 Review
# Topics

<table>
<thead>
<tr>
<th>Everyone</th>
<th>CS 113</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arrays</td>
<td>S&amp;W 1.4</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Methods</td>
<td>S&amp;W 2.1</td>
</tr>
</tbody>
</table>
## Arrays

| Declaration | double[] doubArr;  
|            | boolean[] boolArr; | create a name but not space |
| Create     | doubArr=new doubArr[5]; | Actually make the array |
| Initialize | int siz = 5;  
|            | .....  
|            | boolArr = new boolArr[siz]; | The size of an array can be set at run time |
| Use        | for (int i=0; i<doubArr.length; i++) {  
|            | doubArr[i]=i*i;  
|            | } | Arrays usually have a "default" value (but I never remember). Use for loops. |
| Use        | double sum=0.0;  
|            | for (int i=(doubArr.length-1); i>=0; i++) {  
|            | sum += doubArray[i];  
|            | } | 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)
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
    • use as param to another method
double d = Math.pow(3,2);
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
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));
}